The Java Class Library

This appendix is a reference for many of the classes in the Java standard class library. We list the variables, constants, constructors, and methods of each class. Items within a class are grouped according to their purpose. The classes are listed in alphabetical order. The package each class is contained in is given in parentheses after the class name.

AbstractButton (`javax.swing`)

A public abstract class, derived from `JComponent` and implementing `ItemSelectable` and `SwingConstants`, that represents the common behaviors for buttons and menu items.

Methods

```
public void addActionListener(ActionListener listener)
public void addChangeListener(ChangeListener listener)
public void addItemListener(ItemListener listener)
```

Adds a specific type of listener to this button.

```
public void doClick()
public void doClick(int pressTime)
```

Performs a button click programmatically (as if the user had used the mouse). The button stays visually "pressed" for `pressTime` milliseconds if specified.

```
public Icon getDisabledIcon()
public void setDisabledIcon(Icon disabeledIcon)
```

Gets or sets the icon used by this button when it is disabled.

```
public Icon getDisabledSelectedIcon()
public void setDisabledSelectedIcon(Icon disabledSelectedIcon)
```

Gets or sets the icon used by this button when it is disabled and selected.

```
public int getHorizontalAlignment()
public void setHorizontalAlignment(int alignment)
public int getVerticalAlignment()
```

```
public void setVerticalAlignment(int alignment)
```
Gets or sets the horizontal or vertical alignment of the icon and text.
```
public int getHorizontalTextPosition()
public void setHorizontalTextPosition(int position)
public int getVerticalTextPosition()
public void setVerticalTextPosition(int position)
```
Gets or sets the horizontal or vertical position of the text relative to the icon.
```
public Icon getIcon()
public void setIcon(Icon icon)
```
Gets or sets the default icon for this button.
```
public Insets getMargin()
public void setMargin(Insets insets)
```
Gets or sets the margin between this button's border and the label.
```
public int getMnemonic()
public void setMnemonic(int mnemonic)
```
Gets or sets this button's keyboard mnemonic.
```
public Icon getPressedIcon()
public void setPressedIcon(Icon icon)
```
Gets or sets the icon used by this button when it is pressed.
```
public Icon getRolloverIcon()
public void setRolloverIcon(Icon icon)
```
Gets or sets the icon used by this button when the mouse rolls over it.
```
public Icon getSelectedIcon()
public void setSelectedIcon(Icon icon)
```
Gets or sets the icon used by this button when it is selected.
```
public String getText()
public void setText(String text)
```
Gets or sets the text displayed on this button.
```
public void setEnabled(boolean flag)
```
Enables or disables this button.
```
public void setRolloverEnabled(boolean flag)
```
Enables or disables the rollover property for this button. Rollover effects will not occur if this property is disabled.
```
public isRolloverEnabled()
```
Returns true if this button currently has its rollover property enabled.

```
public void setSelected(boolean flag)
```
Selects or deselects ths button.
```
public boolean isSelected()
```
Returns true if this button is currently selected.

ActionEvent (java.awt.event)

A public class, derived from `AWTEvent`, that represents an AWT action event.

Variables and Constants

```
public static final int ALT_MASK
public static final int CTRL_MASK
public static final int META_MASK
public static final int SHIFT_MASK
```
Constant values which represent masks for the Alt, Control, Meta, and Shift keys being pressed during an action event.
```
public static final int ACTION_FIRST
public static final int ACTION_LAST
```
Constant values that represent the index of the first and last action event ids.
```
public static final int ACTION_PERFORMED
```
A constant value that represents an action performed AWT event type.

Constructors

```
public ActionEvent(Object src, int type, String cmd)
public ActionEvent(Object src, int type, String cmd, int keys)
```
Creates a new instance of an `ActionEvent` from the specified source object, event type, and command string. Additionally, a mask value can be set that defines the types of keys depressed during the event.

Methods

```
public String getActionCommand()
```
Returns the command string associated with this action.
```
public int getModifiers()
```
Returns the mask of the modifiers (special keys) depressed during this event.
```
public String paramString()
```
Returns a string containing the parameters of this `ActionEvent`.

AdjustmentEvent (java.awt.event)

A public class, derived from AWTEvent, that represents an AWT adjustment event.

Variables and Constants

```
public static final int ADJUSTMENT_FIRST
public static final int ADJUSTMENT_LAST
```
Constant values that represent the index of the first and last adjustment event ids.

```
public static final int ADJUSTMENT_VALUE_CHANGED
```
A constant value that represents an adjustment value change event.

```
public static final int BLOCK_DECREMENT
public static final int BLOCK_INCREMENT
```
Constant values that represent block decrement and increment events.

```
public static final int TRACK
```
A constant value which represents an absolute tracking adjustment event.

```
public static final int UNIT_DECREMENT
public static final int UNIT_INCREMENT
```
Constant values which represent unit decrement and increment events.

Constructors

```
public AdjustmentEvent(Adjustable source, int id, int type, int
val)
```
Creates a new instance of an AdjustmentEvent from a specified source and having a specified id, type, and value.

Methods

```
public Adjustable getAdjustable()
```
Returns the adjustable object that originated this AWT AdjustmentEvent.

```
public int getAdjustmentType()
```
Returns the type of adjustment for this event.

```
public int getValue()
```
Returns the current value of this AdjustmentEvent.

```
public String paramString()
```
Returns a string containing the parameters of this event.

Applet (`java.applet`)

A public class, derived from `Panel`, that is intended to be used as a program running inside a Web page.

Constructors

`public Applet()`
Creates a new instance of an applet for inclusion on a Web page.

Methods

`public void destroy()`
Destroys the applet and all of its resources. This method contains no functionality and should be overridden by subclasses.

`public AppletContext getAppletContext()`
Returns this applet's context (the environment in which it is running).

`public String getAppletInfo()`
Returns a string representation of information regarding this applet. This method contains no functionality and should be overridden by subclasses.

`public AudioClip getAudioClip(URL audio)`

`public AudioClip getAudioClip(URL base, String filename)`
Returns the `AudioClip` requested. The location of the audio clip can be given by the base URL and the file name relative to that base.

`public URL getCodeBase()`

`public URL getDocumentBase()`

`public Locale getLocale()`
Returns the URL of this applet, the document that contains this applet, or the locale of this applet.

`public Image getImage(URL image)`

`public Image getImage(URL base, String filename)`
Returns the image requested. The location of the image can be given by the base URL and the filename relative to that base.

`public String getParameter(String param)`

`public String[][] getParameterInfo()`
Returns the value of the specified parameter for this applet. An array of string elements containing information about each parameter for this applet can also be obtained. Each element of the returned array should be comprised of three strings (parameter name, type, and description). This method contains no functionality and should be overridden by subclasses.

`public void init()`

This method provides initialization functionality to the applet prior to the first time that the applet is started. It is automatically called by the browser or the appletviewer program. This method contains no functionality and should be overridden by subclasses.

`public boolean isActive()`

Returns a true value if this applet is currently active. An applet is considered active just prior to execution of its start method and is no longer active just after execution of its stop method.

`public void play(URL source)`

`public void play(URL base, String filename)`

Plays the audio clip located at source. The location of the audio clip can be given as a base URL and the filename relative to that base.

`public void resize(Dimension dim)`

`public void resize(int w, int h)`

Resizes this applet according to the specified dimension.

`public final void setStub(AppletStub stub)`

Sets the interface between this applet and the browser or appletviewer program.

`public void showStatus(String message)`

Prints the specified message in the browser's status window.

`public void start()`

This method generally contains functionality relevant to the starting of this applet. It is called after the applet has been initialized (with the `init` method) and every time the applet is reloaded in the browser or appletviewer program. This method contains no functionality and should be overridden by subclasses.

`public void stop()`

This method generally contains functionality relevant to the stopping of this applet. It is called by the browser (when the containing Web page is replaced) or appletviewer program. This method contains no functionality and should be overridden by subclasses.

ArrayList (`java.util`)

A public class, derived from `AbstractList`, that represents a resizable array implementation of a list. Similar to `Vector`, but unsynchronized.

Constructors

`public ArrayList()`

`public ArrayList(int initialCapacity)`
Creates a new list with the specified initial capacity (ten by default).

`public ArrayList(Collection col)`
Creates a new list containing the elements of the specified collection.

Methods

`public void add(int index, Object element)`
Inserts the specified element into this list at the specified index.

`public boolean add(Object obj)`
Appends the specified element to the end of this list.

`public boolean addAll(Collection col)`

`public boolean addAll(int index, Collection col)`
Inserts all of the elements in the specified collection into the list at the specified index, or appends them to the end of the list if no index is specified.

`public void clear()`
Removes all of the elements from this list.

`public boolean contains(Object obj)`
Returns true if this list contains the specified object.

`public void ensureCapacity(int minimumCapacity)`
Increases the capacity of this list to the specified value if necessary.

`public Object get(int index)`
Returns the element at the specified index. Throws `IndexOutOfBoundsException` if the index is out of range.

`public int indexOf(Object obj)`
Returns the index of the first occurrence of the specified object (based on the equals method) or –1 if it is not found.

`public boolean isEmpty()`
Returns true if this list contains no elements.

`public int lastIndexOf(Object obj)`
Returns the index of the last occurrence of the specified object (based on the equals method) or –1 if it is not found.

`public Object remove(int index)`
Removes and returns the object at the specified index in this list. Throws `IndexOutOfBoundsException` if the index is out of range.

`protected void removeRange(int fromIndex, int toIndex)`
Removes the elements at the indexes in the specified range, exclusive.

```
public Object set(int index, Object obj)
```
Replaces the element at the specified index with the specified object.

```
public int size()
```
Returns the number of elements in this list.

```
public Object[] toArray()
```
Returns an array containing the elements in this list.

```
public void trimToSize()
```
Trims the capacity of this list to the current size.

AWTEvent (`java.awt`)

A public class, derived from `EventObject`, that is the root class for all of the AWT event classes.

Variables and Constants

```
public final static long ACTION_EVENT_MASK
public final static long ADJUSTMENT_EVENT_MASK
public final static long COMPONENT_EVENT_MASK
public final static long CONTAINER_EVENT_MASK
public final static long FOCUS_EVENT_MASK
public final static long ITEM_EVENT_MASK
public final static long KEY_EVENT_MASK
public final static long MOUSE_EVENT_MASK
public final static long MOUSE_MOTION_EVENT_MASK
public final static long TEXT_EVENT_MASK
public final static long WINDOW_EVENT_MASK
```
Constant values representing the AWT event masks for various events.

```
protected boolean consumed
```
A variable representing the state of the event. A true value means that it has not been sent to the appropriate peer, false indicates that it has.

```
protected int id
```
The numeric identification for this event.

Constructors

```
public AWTEvent(Event evt)
```
Creates a new `AWTEvent` from the specified event.

```
public AWTEvent(Object src, int type)
```
Creates a new `AWTEvent` from a specified source, and having a defined type.

Methods

`protected void consume()`
Targets this `AWTEvent` to be sent to the appropriate peer.

`public int getID()`
Returns this event's type.

`protected boolean isConsumed()`
Returns a true value if this `AWTEvent` has been sent to the appropriate peer.

`public String paramString()`
Returns the parameter string for this `AWTEvent`.

`public String toString()`
Returns a string representation of this `AWTEvent`.

BigDecimal (`java.math`)

A public class, derived from `Number`, which can be used to represent a decimal number with a definable precision.

Variables and Constants

`ROUND_CEILING`
A constant that represents a rounding mode in which the value of the `BigDecimal` is rounded up (away from zero) if the number is positive, and down (closer to zero) if the number is negative.

`ROUND_DOWN`
A constant that represents a rounding mode in which the value of the `BigDecimal` is rounded closer to zero (decreasing a positive number and increasing a negative number).

`ROUND_FLOOR`
A constant that represents a rounding mode in which the value of the `BigDecimal` is rounded down (closer to zero) if the number is positive, and up (away from zero) if the number is negative.

`ROUND_HALF_DOWN`
A constant that represents a rounding mode in which the value of the `BigDecimal` is rounded as in `ROUND_UP` if the fraction of the number is greater than 0.5 and as `ROUND_DOWN` in all other cases.

`ROUND_HALF_EVEN`
A constant that represents a rounding mode in which the value of the `BigDecimal` is rounded as in `ROUND_HALF_UP` if the number to the left of the decimal is odd and as `ROUND_HALF_DOWN` when the number is even.

`ROUND_HALF_UP`
A constant that represents a rounding mode in which the value of the `BigDecimal` is rounded as in `ROUND_UP` if the fraction of the number is greater than or equal to 0.5 and as in `ROUND_DOWN` in all other cases.

`ROUND_UNNECESSARY`
A constant that represents a rounding mode in which the value of the `BigDecimal` is not rounded (if possible) and an exact result is returned.

`ROUND_UP`
A constant that represents a rounding mode in which the value of the `BigDecimal` is rounded away from zero (increasing a positive number, and decreasing a negative number).

Constructors

```
public BigDecimal(BigInteger arg)
public BigDecimal(BigInteger arg, int scale) throws
NumberFormatException
public BigDecimal(double arg) throws NumberFormatException
public BigDecimal(String arg) throws NumberFormatException
```

Creates an instance of a `BigDecimal` from `arg`. The string argument may contain a preceding minus sign indicating a negative number. The resulting `BigDecimal`'s scale will be the number of integers to the right of the decimal point in the string, a specified value, or 0 (zero) if none are present.

Methods

```
public double doubleValue()
public float floatValue()
public int intValue()
public long longValue()
public BigInteger toBigInteger()
public String toString()
```

Converts this `BigDecimal` to either a Java primitive type or a `BigInteger`.

```
public BigDecimal abs()
```

Returns the absolute value of this `BigDecimal` with the same scale as this `BigDecimal`.

```
public BigDecimal add(BigDecimal arg)
public BigDecimal subtract(BigDecimal arg)
```

Returns the result of `arg` added to or subtracted from this `BigDecimal`, with the resulting scale equal to the larger of the two `BigDecimal`'s scales.

```
public int compareTo(BigDecimal arg)
```
This method compares this `BigDecimal` to `arg` and will return a –1 if this `BigDecimal` is less than `arg`, 0 if equal to `arg` or a 1 if greater than `arg`. If the values of the two `BigDecimals` are identical and the scales are different, they are considered equal.

```
public BigDecimal divide(BigDecimal arg, int mode) throws
ArithmeticException, IllegalArgumentException
```

```
public BigDecimal divide(BigDecimal arg, int scale, int mode)
throws ArithmeticException, IllegalArgumentException
```
Returns the result of this `BigDecimal` divided by `arg`. If required the rounding mode is used. The resulting `BigDecimal`'s scale is identical to this `BigDecimal`'s scale or a specified value.

```
public boolean equals(Object arg)
```
Returns a true value if this `BigDecimal`'s value and scale are equal to `arg`'s value and scale.

```
public int hashCode()
```
Returns the hash code of this `BigDecimal`.

```
public BigDecimal max(BigDecimal arg)
```

```
public BigDecimal min(BigDecimal arg)
```
Returns the greater or lesser of this `BigDecimal` and `arg`.

```
public BigDecimal movePointLeft(int num)
```

```
public BigDecimal movePointRight(int num)
```
Returns this `BigDecimal` with the decimal point moved `num` positions.

```
public BigDecimal multiply(BigDecimal arg)
```
Returns the result of this `BigDecimal` multiplied with the value of `arg`. The scale of the resulting `BigDecimal` is the result of the addition of the two `BigDecimal`'s scales.

```
public BigDecimal negate()
```
Returns the negation of this `BigDecimal`'s value with the same scale.

```
public int scale()
```
Returns the scale of this `BigDecimal`.

```
public BigDecimal setScale(int val) throws ArithmeticException,
IllegalArgumentException
```

```
public BigDecimal setScale(int val, int mode) throws
ArithmeticException, IllegalArgumentException
```
Returns a `BigDecimal` whose value is the same as this `BigDecimal`'s and has a new scale specified by `val`. If rounding is necessary, a rounding mode can be specified.

```
public int signum()
```
Returns a –1 if this `BigDecimal` is negative, 0 if zero, and 1 if positive.

```
public static BigDecimal valueOf(long value)
public static BigDecimal valueOf(long value, int scale) throws
NumberFormatException
```
Returns a `BigDecimal` with a defined value. The scale of the returned number is specified or it defaults to 0 (zero).

BigInteger (`java.math`)

A public class, derived from `Number`, that can be used to represent an integer in a two's complement format of any precision.

Constructors

```
public BigInteger(byte[] arg) throws NumberFormatException
public BigInteger(int signum, byte[] magnitude) throws
NumberFormatException
```
Creates an instance of a `BigInteger` from the specified byte array. The sign of the number can be placed in `signum` (where –1 is negative, 0 is zero, and 1 is positive).

```
public BigInteger(String arg) throws NumberFormatException
public BigInteger(String arg, int radix) throws
NumberFormatException
```
Creates an instance of a `BigInteger` from the string `arg`, which can contain decimal numbers preceded by an optional minus sign. The argument `radix` specifies the base of the `arg` value.

```
public BigInteger(int size, Random rand) throws
IllegalArgumentException
public BigInteger(int size, int prob, Random rand)
```
Creates a (generally) prime instance of a `BigInteger` from a random integer, `rand`, of a specified length, `size`. The certainty parameter (`prob`) represents the amount of probability that the generated number is a prime.

Methods

```
public double doubleValue()
public float floatValue()
public int intValue()
public long longValue()
public String toString()
```

```
public String toString(int base)
```
Converts this `BigDecimal` to either a Java primitive type or a `BigInteger`. The base can specify the radix of the number value returned.

```
public BigInteger abs()
```
Returns the absolute value of this `BigInteger`.

```
public BigInteger add(BigInteger arg) throws ArithmeticException
public BigInteger subtract(BigInteger arg)
```
Adds the argument to, or subtracts `arg` from this `BigInteger` and returns the result.

```
public BigInteger and(BigInteger arg)
public BigInteger andNot(BigInteger arg)
public BigInteger not()
public BigInteger or(BigInteger arg)
public BigInteger xor(BigInteger arg)
```
Returns the result of a logical operation of this `BigInteger` and the value of `arg`. The `not` method returns the logical not of this `BigInteger`.

```
public int bitCount()
```
Returns the number of bits from this `BigInteger` that are different from the sign bit.

```
public int bitLength()
```
Returns the number of bits from this `BigInteger`, excluding the sign bit.

```
public BigInteger clearBit(int index) throws ArithmeticException
```
Returns the modified representation of this `BigInteger` with the bit at position `index` cleared.

```
public int compareTo(BigInteger arg)
```
Compares this `BigInteger` to the parameter `arg`. If this `BigInteger` is less than `arg`, a –1 is returned, if equal to `arg` a 0 (zero) is returned, and if greater than `arg`, a 1 is returned.

```
public BigInteger divide(BigInteger arg) throws
ArithmeticException
public BigInteger[] divideAndRemainder(BigInteger arg) throws
ArithmeticException
```
Returns the result of this `BigInteger` divided by `arg`. The `divideAndRemainder` method returns as the first element ([0]) the quotient, and the second element ([1]) the remainder.

```
public boolean equals(Object arg)
```
Returns a true value if this `BigInteger` is equal to the parameter `arg`.

`public BigInteger flipBit(int index) throws ArithmeticException`

Returns the modified representation of this `BigInteger` with the bit at position `index` flipped.

`public BigInteger gcd(BigInteger arg)`

Returns the greatest common denominator of the absolute value of this `BigInteger` and the absolute value of the parameter `arg`.

`public int getLowestSetBit()`

Returns the index of the rightmost bit that is equal to one from this `BigInteger`.

`public int hashCode()`

Returns the hash code of this `BigInteger`.

`public boolean isProbablePrime(int prob)`

Returns a true value if this `BigInteger` is probably a prime number. The parameter `prob` represents the certainty of the decision.

`public BigInteger max(BigInteger arg)`

`public BigInteger min(BigInteger arg)`

Returns the larger or smaller of this `BigInteger` or `arg`.

`public BigInteger mod(BigInteger arg)`

`public BigInteger modInverse(BigInteger arg) throws ArithmeticException`

`public BigInteger modPow(BigInteger exp, BigInteger arg)`

Returns the result of this `BigInteger` mod `arg`. The `modInverse` returns the modular multiplicative inverse. `modPow` returns the result of this (`BigInteger ** exp`) mod `arg`.

`public BigInteger multiply(BigInteger arg)`

Returns the result of this `BigInteger` multiplied by `arg`.

`public BigInteger negate()`

Returns this `BigInteger` negated (this `BigInteger` * –1).

`public BigInteger pow(int exp) throws ArithmeticException`

Returns the result of this `BigInteger` ** exp.

`public BigInteger remainder(BigInteger arg) throws ArithmeticException`

Returns the result of this `BigInteger` mod `arg`.

`public BigInteger setBit(int index) throws ArithmeticException`

Returns the result of this `BigInteger` with the bit at the specified index set.

`public BigInteger shiftLeft(int num)`

`public BigInteger shiftRight(int num)`

Returns the result of this `BigInteger` shifted `num` bits.

`public int signum()`
Returns a –1 if the value of this `BigInteger` is negative, 0 if zero, and 1 if positive.

`public boolean testBit(int index) throws ArithmeticException`
Returns a true value if the bit at the specified index is set.

`public byte[] toByteArray()`
Returns the two's complement of this `BigInteger` in an array of bytes.

`public static BigInteger valueOf(long arg)`
Returns a `BigInteger` from the value of `arg`.

BitSet (`java.util`)

A public final class, derived from `Object` and implementing `Cloneable` and `Serializable`, that allows for the manipulation of a vectored array of bits.

Constructors

`public BitSet()`

`public BitSet(int size)`
Creates a new instance of a bit sequence of `size` bits (the default is 64). Each of the initial bits are set to false.

Methods

`public void and(BitSet arg)`

`public void or(BitSet arg)`

`public void xor(BitSet arg)`
Places all of the bits from both this `BitSet` AND/OR/XORed with the bits of `arg` into this `BitSet`.

`public void clear(int index)`

`public void set(int index)`
Clears or sets the bit (sets it to false) at location `index`.

`public Object clone()`
Returns a clone of this `BitSet`.

`public boolean equals(Object arg)`
Returns a true if `arg` is not null and all bits are equal to this `BitSet`.

`public boolean get(int index)`
Returns the boolean value of the bit at location `index`.

`public int hashCode()`
Returns the hash code of this `BitSet`.

`public int size()`
Returns the size of this `BitSet`.

`public String toString()`
Returns a string representation of this `BitSet` in set notation (i.e., {1, 2, 5})

Boolean (`java.lang`)

A public final class, derived from `Object` and implementing `Serializable`, that contains boolean logic operations, constants, and methods as a wrapper around the Java primitive type `boolean`.

Variables and Constants

`public final static Boolean TRUE`

`public final static Boolean FALSE`
Boolean constant values of true or false.

`public final static Class TYPE`
The `Boolean` constant value of the boolean type class.

Constructors

`public Boolean(boolean arg)`

`public Boolean(String arg)`
Creates an instance of the `Boolean` class from the parameter `arg`.

Methods

`public boolean booleanValue()`
The boolean value of the current object.

`public boolean equals(Object arg)`
Returns the result of an equality comparison against `arg`. Here `arg` must be a boolean object with the same value as this `Boolean` for a resulting true value.

`public static boolean getBoolean(String str)`
Returns a `Boolean` representation of the system property named in `str`.

`public int hashCode()`
Returns the hash code for this object.

`public String toString()`
Returns the string representation of the state of the current object (i.e., "true" or "false").

`public static Boolean valueOf(String str)`
Returns a new Boolean initialized to the value of `str`.

BorderFactory (javax.swing)

A public class, derived from `Object`, that represents a factory for creating GUI borders.

Methods

```
public static Border createBevelBorder(int type)
```

```
public static Border createBevelBorder(int type, Color
highlight, Color shadow)
```

```
public static Border createBevelBorder(int type, Color
outerHighlight, Color innerHighlight, Color outerShadow, Color
innerShadow)
```

Returns a bevel border with the specified type (`BevelBorder.LOWERED` or `BevelBorder.RAISED`) and shading.

```
public static CompoundBorder createCompoundBorder(Border
outside, Border inside)
```

Returns a border composed of the two other specified borders.

```
public static Border createEmptyBorder()
```

```
public static Border createEmptyBorder(int top, int left, int
bottom, int right)
```

Returns an empty (invisible) border with the specified dimensions, which default to 0.

```
public static Border createEtchedBorder()
```

```
public static Border createEtchedBorder(Color highlight, Color
shadow)
```

```
public static Border createEtchedBorder(int type)
```

```
public static Border createEtchedBorder(int type, Color
highlight, Color shadow)
```

Returns an etched border with the specified type (`EtchedBorder.RAISED` or `EtchedBorder.LOWERED`) and shading.

```
public static Border createLineBorder(Color color)
```

```
public static Border createLineBorder(Color color, int
thickness)
```

Returns a line border with the specified color and thickness. If unspecified, the thickness defaults to one pixel.

```
public static Border createLoweredBevelBorder()
```

```
public static Border createRaisedBevelBorder()
```

Returns a border with a lowered or raised beveled edge.

```
public static MatteBorder createMatteBorder(int top, int left,
int bottom, int right, Color color)
```

```
public static MatteBorder createMatteBorder(int top, int left,
int bottom, int right, Icon icon)
```
Returns a matte border with the specified edge sizes. The border is made up either of the specified color or the specified icon.

```
public static TitledBorder createTitledBorder(Border border)
public static TitledBorder createTitledBorder(String title)
public static TitledBorder createTitledBorder(Border border,
String title)
public static TitledBorder createTitledBorder(Border border,
String title, int justification, int position)
public static TitledBorder createTitledBorder(Border border,
String title, int justification, int position, Font font)
public static TitledBorder createTitledBorder(Border border,
String title, int justification, int position, Font font, Color
color)
```
Returns a titled border with the specified border and the specified title text, justification, position, font, and color. Justification and position are defined by constants in the `TitledBorder` class. Justification can be: `LEFT`, `CENTER`, `RIGHT`, `LEADING`, or `TRAILING` (default is `LEADING`). Position specifies the title's vertical position in relation to the border and can be: `TOP`, `BELOW_TOP`, `ABOVE_BOTTOM`, `BOTTOM`, or `BELOW_BOTTOM` (default is `TOP`).

BorderLayout (`java.awt`)

A public class, derived from `Object` and implementing `LayoutManager2` and `Serializable`, that lays out a container using five distinct areas (North, South, East, West, and Center).

Variables and Constants

```
public final static String CENTER
public final static String EAST
public final static String NORTH
public final static String SOUTH
public final static String WEST
```
Constant values indicating areas of the border layout manager.

Constructors

```
public BorderLayout()
public BorderLayout(int hgap, int vgap)
```

Creates a new instance of a `BorderLayout`. If no initial horizontal and vertical gaps are specified, they default to zero.

Methods

```
public void addLayoutComponent(Component item, Object
constraints)
public void removeLayoutComponent(Component item)
```

Adds or removes a component to this layout manager. When adding a component, it is possible to restrict the component to the specified constraints.

```
public int getHgap()
public int getVgap()
```

Returns the horizontal or vertical gap of components laid out by this layout manager.

```
public float getLayoutAlignmentX(Container cont)
public float getLayoutAlignmentY(Container cont)
```

Returns the horizontal or vertical alignment value of the specified container.

```
public void invalidateLayout(Container cont)
```

Forces this layout manager to discard any cached layout information about the specified container.

```
public void layoutContainer(Container cont)
```

Lays out the specified container with this layout manager.

```
public Dimension maximumLayoutSize(Container cont)
public Dimension minimumLayoutSize(Container cont)
public Dimension preferredLayoutSize(Container cont)
```

Returns the maximum, minimum or preferred size of the specified container when laid out by this layout manager.

```
public void setHgap(int hgap)
public void setVgap(int vgap)
```

Sets the horizontal or vertical gap in pixels of components laid out by this layout manager.

Box (javax.swing)

A public class, derived from `JComponent` and implementing `Accessible`, that represents a lightweight container that uses a box layout.

Constructors

```
public Box(int axis)
```
Creates a box that displays its components along the specified axis (`BoxLayout.X_AXIS` or `BoxLayout.Y_AXIS`).

Methods

```
public static Box createHorizontalBox()
public static Box createVerticalBox()
```
Returns a box that displays its components horizontally (from left to right) or vertically (from top to bottom).

```
public static Component createGlue()
public static Component createHorizontalGlue()
public static Component createVerticalGlue()
```
Returns an invisible glue component that expands as much as necessary to fill the space between neighboring components.

```
public static Component createRigidArea(Dimension dim)
```
Returns an invisible component with the specified size.

```
public static Component createHorizontalStrut(int width)
public static Component createVerticalStrut(int height)
```
Returns an invisible component with a fixed width or height.

BoxLayout (javax.swing)

A public class, derived from `Object` and implementing `LayoutManager2` and `Serializable`, that lays out components either vertically or horizontally.

Variables and Constants

```
public static final int X_AXIS
public static final int Y_AXIS
```
Specifies that components should be laid out left to right or top to bottom.

Constructors

```
public BoxLayout(Container target, int axis)
```
Creates a box layout for the specified target container along the specified axis.

Methods

```
public float getLayoutAlignmentX(Container cont)
public float getLayoutAlignmentY(Container cont)
```
Returns the horizontal or vertical alignment value of the specified container.

```
public void invalidateLayout(Container cont)
```
Forces this layout manager to discard any cached layout information about the specified container.

```
public void layoutContainer(Container cont)
```
Lays out the specified container with this layout manager.

```
public Dimension maximumLayoutSize(Container cont)
public Dimension minimumLayoutSize(Container cont)
public Dimension preferredLayoutSize(Container cont)
```
Returns the maximum, minimum, or preferred size of the specified container when laid out by this layout manager.

BufferedReader (`java.io`)

A public class, derived from `Reader`, that provides a buffered stream of character-based input.

Constructors

```
public BufferedReader(Reader rdr)
public BufferedReader(Reader rdr, int size)
```
Creates a `BufferedReader` from the specified `Reader`, by using a specified size (in characters). The default size is 8192 characters.

Methods

```
public void close() throws IOException
```
Closes this `BufferedReader`.

```
public void mark(int readAheadLimit) throws IOException
```
Sets a mark in the stream where attempts to reset this `BufferedReader` will return to. The `readAheadLimit` determines how far ahead the stream can be read before the mark expires.

```
public boolean markSupported()
```
An overridden method from `Reader` that determines if this stream supports the setting of a mark.

```
public int read() throws IOException
```

```
public String readLine() throws IOException
```
Reads a single character or an entire line from this `BufferedReader` stream. The character is returned as an `int`, the line as a string. A line of text is considered to be a series of characters ending in a carriage return (`\r`), a line feed (`\n`), or a carriage return followed by a line (`\r\n`).

```
public int read(char[] dest, int offset, int size) throws
IOException
```
Reads `size` characters from this `BufferedReader` stream. Reading will skip `offset` characters into the current location in the stream, and place them in the destination array. This method will return the number of characters read from the stream or a –1 if the end of the stream was reached.

```
public boolean ready() throws IOException
```
Returns a true value if this `BufferedReader` is capable of being read from. This state can only be true if the buffer is not empty.

```
public void reset() throws IOException
```
Resets this `BufferedReader` to the last mark.

```
public long skip(long num) throws IOException
```
Skips forward `num` characters in the stream and returns the actual number of characters skipped.

BufferedWriter (`java.io`)

A public class, derived from `Writer`, that represents a character output stream that buffers characters for efficiency.

Constructors

```
public BufferedWriter(Writer out)
```

```
public BufferedWriter(Writer out, int size)
```
Creates a buffered output stream using the specified `Writer` stream and a buffer of the specified size.

Methods

```
public void close()
```
Closes this stream.

```
public void flush()
```
Flushes this stream.

```
public void newLine()
```
Writes a line separator to this stream.

```
public void write(int ch)
public void write(String str)
public void write(String str, int offset, int length)
public void write(char[] buffer)
public void write(char[] buffer, int offset, int length)
```
Writes a single character, string, or character array to this stream. A portion of the string or character array can be specified.

ButtonGroup (javax.swing)

A public class, derived from `Object` and implementing `Serializable`, that represents a set of mutually exclusive buttons.

Constructors

```
public ButtonGroup()
```
Creates an empty button group.

Methods

```
public void add(AbstractButton button)
public void remove(AbstractButton button)
```
Adds or removes the specified button to this group.

```
public int getButtonCount()
```
Returns the number of buttons in this group.

Byte (java.lang)

A public final class, derived from `Number`, that contains byte logic operations, constants, and methods as a wrapper around the Java primitive type `byte`.

Variables and Constants

```
public final static byte MAX_VALUE
public final static byte MIN_VALUE
```
A constant value that holds the maximum (127) and minimum (–128) values a byte can contain.

```
public final static Class TYPE
```
The `Byte` constant value of the byte type class.

Constructors

```
public Byte(byte arg)
public Byte(String arg) throws NumberFormatException
```
Creates a new instance of a `Byte` from `arg`.

Methods

```
public byte byteValue()
public double doubleValue()
public float floatValue()
public int intValue()
public long longValue()
public short shortValue()
```
Returns the value of this `Byte` as a Java primitive type.

```
public static Byte decode(String str) throws
NumberFormatException
```
Returns the given string (`str`) as a `Byte`. The parameter string may be encoded as an octal, hexadecimal, or binary number.

```
public boolean equals(Object arg)
```
Returns a true value if this `Byte` is equal to the parameter object `arg`.

```
public int hashCode()
```
Returns the hash code of this `Byte`.

```
public static byte parseByte(String str) throws
NumberFormatException
public static byte parseByte(String str, int base) throws
NumberFormatException
```
Returns the value of the parsed string (`str`) as a byte. The radix of the string can be specified in `base`.

```
public String toString()
public static String toString(byte prim)
```
Returns a string representation of this `Byte` or the specified primitive byte (`prim`), whose radix is assumed to be 10.

```
public static Byte valueOf(String str) throws
NumberFormatException
```

```
public static Byte valueOf(String str, int base) throws
NumberFormatException
```

Returns a `Byte` object whose initial value is the result of the parsed parameter (`str`). The parameter is assumed to be the text representation of a byte and its radix 10 (unless specified in `base`).

Calendar (`java.util`)

A public abstract class, derived from `Object` and implementing `Cloneable` and `Serializable`, that allows for the manipulation of a `Date` object.

Variables and Constants

```
public static final int AM
public static final int PM
```

Constant values that represent ante and post meridian.

```
public static final int ERA
public static final int YEAR
public static final int MONTH
public static final int WEEK_OF_YEAR
public static final int WEEK_OF_MONTH
public static final int DATE
public static final int DAY_OF_MONTH
public static final int DAY_OF_YEAR
public static final int DAY_OF_WEEK
public static final int DAY_OF_WEEK_IN_MONTH
public static final int AM_PM
public static final int HOUR
public static final int HOUR_OF_DAY
public static final int MINUTE
public static final int SECOND
public static final int MILLISECOND
public static final int ZONE_OFFSET
public static final int DST_OFFSET
```

Constant values that represent the index to the field where particular data is stored representing an instance of time (to millisecond precision). The combination of all of these fields yields a full representation of a moment of time with respect to a particular calendar (i.e., `GregorianCalendar`).

```
public static final int JANUARY
public static final int FEBRUARY
public static final int MARCH
public static final int APRIL
public static final int MAY
public static final int JUNE
public static final int JULY
public static final int AUGUST
public static final int SEPTEMBER
public static final int OCTOBER
public static final int NOVEMBER
public static final int DECEMBER
public static final int UNDECIMBER
```

Constant values representing various calendar months. `UNDECIMBER` represents the 13th month of a Gregorian calendar (lunar month).

```
public static final int SUNDAY
public static final int MONDAY
public static final int TUESDAY
public static final int WEDNESDAY
public static final int THURSDAY
public static final int FRIDAY
public static final int SATURDAY
```

Constant values representing the days of a week.

```
protected boolean areFieldsSet
```

A boolean flag that indicates if the time fields have been set for this `Calendar`.

```
public static final int FIELD_COUNT
```

A constant value that represents the number of date/time fields stored by a `Calendar`.

```
protected int fields[]
```

The integer array that contains the values that make up the information about this `Calendar`.

```
protected boolean isSet[]
```

The boolean array that contains status values used to indicate if a corresponding time field has been set.

```
protected boolean isTimeSet
```

A boolean flag field that is used to indicate if the time is set for this `Calendar`.

```
protected long time
```

A `long int` field that contains the time set for this `Calendar`.

Methods

`public abstract void add(int field, int val)`

Adds (or subtracts in the case of a negative `val`) an amount of days or time from the specified `field`.

`public abstract boolean after(Object arg)`

`public abstract boolean before(Object arg)`

Returns a true value if this `Calendar` date is after or before the date specified by `arg`.

`public final void clear()`

`public final void clear(int field)`

Clears the value from the specified time `field` from this `Calendar`. The `clear` method will clear all of the values from this `Calendar`.

`public Object clone()`

Returns a clone of this `Calendar`.

`protected void complete()`

Attempts to complete any empty date/time fields by calling the `completeTime()` and `completeFields()` methods of this `Calendar`.

`protected abstract void computeFields()`

`protected abstract void computeTime()`

Computes the values of the time fields based on the currently set time (`computeFields()`) or computes the time based on the currently set time fields (`computeTime()`) for this `Calendar`.

`public abstract boolean equals(Object arg)`

Returns a true value if this `Calendar` is equal to the value of `arg`.

`public final int get(int fld)`

Returns the value of the specified time field from this `Calendar`.

`public static synchronized Locale[] getAvailableLocales()`

Returns the list of locales that are available.

`public int getFirstDayOfWeek()`

`public void setFirstDayOfWeek(int val)`

Returns or sets the first day of the week to `val` for this `Calendar`.

`public abstract int getGreatestMinimum(int fld)`

Returns the largest allowable minimum value for the specified field.

`public static synchronized Calendar getInstance()`

`public static synchronized Calendar getInstance(Locale locale)`

`public static synchronized Calendar getInstance(TimeZone tz)`

```
public static synchronized Calendar getInstance(TimeZone tz,
Locale locale)
```
Returns an instance of a `Calendar` based on the default time zone and locale, or from a specified time zone and/or locale.

```
public abstract int getLeastMaximum(int fld)
```
Returns the smallest allowable maximum value for the specified field.

```
public abstract int getMaximum(int fld)
public abstract int getMinimum(int fld)
```
Returns the largest or smallest allowable value for the specified field.

```
public int getMinimalDaysInFirstWeek()
public void setMinimalDaysInFirstWeek(int val)
```
Returns or sets the smallest allowable number of days in the first week of the year, based on the locale.

```
public final Date getTime()
public final void setTime(Date dt)
```
Returns or sets the time for this `Calendar`.

```
protected long getTimeInMillis()
protected void setTimeInMillis(long ms)
```
Returns or sets the time in milliseconds for this `Calendar`.

```
public TimeZone getTimeZone()
public void setTimeZone(TimeZone val)
```
Returns or sets the time zone for this `Calendar`.

```
protected final int internalGet(int fld)
```
An internal method used to obtain field values to be used by subclasses of `Calendar`.

```
public boolean isLenient()
public void setLenient(boolean flag)
```
Returns or sets the flag indicating leniency for date/time input.

```
public final boolean isSet(int fld)
```
Returns a true value if a value is set for the specified field.

```
public abstract void roll(int fld, boolean direction)
```
Adds one single unit of time to the specified date/time field. A true value specified for `direction` increases the field's value, false decreases it.

```
public final void set(int fld, int val)
```
Sets a single specified field to a value.

```
public final void set(int year, int month, int date)
public final void set(int year, int month, int date, int hour,
int min)
```

```
public final void set(int year, int month, int date, int hour,
int min, int sec)
```

Sets the year, month, date, hour, minute, and seconds of the time fields for this `Calendar`.

CardLayout (`java.awt`)

A public class, derived from `Object` and implementing `LayoutManager2` and `Serializable`, that lays out components in a series of separate cards, only one of which is visible at any time. The visibility of the cards can be changed, essentially providing the ability to sequence through the cards.

Constructors

```
public CardLayout()
public CardLayout(int hg, int vg)
```

Creates a new instance of a card layout with a specified horizontal and vertical gap (or no gap in the case of the first constructor).

Methods

```
public void addLayoutComponent(Component item, Object constr)
public void removeLayoutComponent(Component item)
```

Adds or removes a component to this layout manager. While adding, it is possible to restrict the component to the specified constraints (`constr`).

```
public void first(Container cont)
public void last(Container cont)
```

Moves to the first or last card in the layout. `cont` is the container that is laid out by this layout manager.

```
public int getHgap()
public int getVgap()
```

Returns the horizontal or vertical gap between the components laid out by this layout manager.

```
public float getLayoutAlignmentX(Container parent)
public float getLayoutAlignmentY(Container parent)
```

Returns the horizontal or vertical alignment value of the specified container.

```
public void invalidateLayout(Container cont)
```

Forces this layout manager to discard any cached layout information about the specified container.

```
public void layoutContainer(Container cont)
```
Lays out the specified container with this layout manager.

```
public Dimension maximumLayoutSize(Container cont)
public Dimension minimumLayoutSize(Container cont)
public Dimension preferredLayoutSize(Container cont)
```
Returns the maximum, minimum, or preferred size of the specified container when laid out by this layout manager.

```
public void next(Container cont)
public void previous(Container cont)
```
Cycles to the next or previous card. `cont` is the container that is laid out by this layout manager.

```
public void setHgap(int hg)
public void setVgap(int vg)
```
Sets the horizontal or vertical gap in pixels of components laid out by this layout manager.

```
public void show(Container cont, String str)
```
Cycles to the card the contains the component with the name `str`. When found, the specified container is laid out with this layout manager.

```
public String toString()
```
Returns a string representation of this layout manager.

Character (`java.lang`)

A public class, derived from `Object` and implementing `Serializable`, that contains character constants and methods to convert and identify characters.

Variables and Constants

```
public final static byte COMBINING_SPACING_MARK
public final static byte CONNECTOR_PUNCTUATION
public final static byte CONTROL
public final static byte CURRENCY_SYMBOL
public final static byte DASH_PUNCTUATION
public final static byte DECIMAL_DIGIT_NUMBER
public final static byte ENCLOSING_MARK
public final static byte END_PUNCTUATION
public final static byte FORMAT
public final static byte LETTER_NUMBER
```

```
public final static byte LINE_SEPARATOR
public final static byte LOWERCASE_LETTER
public final static byte MATH_SYMBOL
public final static byte MODIFIER_LETTER
public final static byte MODIFIER_SYMBOL
public final static byte NON_SPACING_MARK
public final static byte OTHER_LETTER
public final static byte OTHER_NUMBER
public final static byte OTHER_PUNCTUATION
public final static byte OTHER_SYMBOL
public final static byte PARAGRAPH_SEPARATOR
public final static byte PRIVATE_USE
public final static byte SPACE_SEPARATOR
public final static byte START_PUNCTUATION
public final static byte SURROGATE
public final static byte TITLECASE_LETTER
public final static byte UNASSIGNED
public final static byte UPPERCASE_LETTER
```

Constant values representing various character symbols and types.

```
public final static int MAX_RADIX
```

A constant value that represents the largest possible value of a radix (base).

```
public final static char MAX_VALUE
```

A constant value that represents the largest possible value of a character in Java = `\uffff'`.

```
public final static int MIN_RADIX
```

A constant value that represents that smallest possible value of a radix (base).

```
public final static char MIN_VALUE
```

A constant value that represents the smallest possible value of a character in Java = `\u0000'`.

```
public final static Class TYPE
```

The `Character` constant value of the character type class.

Constructors

```
public Character(char prim)
```

Creates an instance of the `Character` class from the primitive parameter `prim`.

Methods

`public char charValue()`

Returns the value of this `Character` as a primitive character.

`public static int digit(char c, int base)`

`public static char forDigit(int c, int base)`

Returns the numeric value or the character depiction of the parameter `c` in radix `base`.

`public boolean equals(Object arg)`

Returns a true value if this `Character` is equal to the parameter `arg`.

`public static int getNumericValue(char c)`

Returns the Unicode representation of the character parameter (`c`) as a nonnegative integer. If the character has no numeric representation, a –1 is returned. If the character cannot be represented as a nonnegative number, –2 will be returned.

`public static int getType(char c)`

Returns an integer value that represents the type of character the parameter `c` is.

`public int hashCode()`

Returns a hash code for this `Character`.

`public static boolean isDefined(char c)`

`public static boolean isISOControl(char c)`

Returns a true value if the parameter `c` has a defined meaning in Unicode or is an ISO control character.

`public static boolean isIdentifierIgnorable(char c)`

Returns a true value if the parameter `c` is a character that can be ignored in a Java identifier (such as control characters).

`public static boolean isJavaIdentifierPart(char c)`

`public static boolean isJavaIdentifierStart(char c)`

Returns a true value if the parameter `c` can be used in a valid Java identifier in any but the leading character. `isJavaIdentifierStart` returns a true value if the parameter `c` can be used as the leading character in a valid Java identifier.

`public static boolean isDigit(char c)`

`public static boolean isLetter(char c)`

`public static boolean isLetterOrDigit(char c)`

`public static boolean isLowerCase(char c)`

`public static boolean isSpaceChar(char c)`

`public static boolean isTitleCase(char c)`

```
public static boolean isUnicodeIdentifierPart(char c)
public static boolean isWhitespace(char c)
public static boolean isUnicodeIdentifierStart(char c)
public static boolean isUpperCase(char c)
```

Returns a true value if the parameter `c` is a digit; letter; letter or a digit; lowercase character; space character; titlecase character; can be used in a valid Unicode identifier in any but the leading character; a white space character; can be used as the leading character in a valid Unicode identifier or an uppercase character (respectively).

```
public static char toLowerCase(char c)
public String toString()
public static char toTitleCase(char c)
public static char toUpperCase(char c)
```

Returns a lowercase character, string representation, titlecase, or uppercase character of the parameter `c`.

Class (`java.lang`)

A public final class, derived from `Object` and implementing `Serializable`, that describes both interfaces and classes in the currently running Java program.

Methods

```
public static Class forName(String class) throws
ClassNotFoundException
```

Returns a `Class` object that corresponds with the named `class`. The name of the specified class must be a fully qualified class name (as in `java.io.Reader`).

```
public Class[] getClasses()
public Class[] getDeclaredClasses() throws SecurityException
```

Returns an array of `Classes` that contains all of the interfaces and classes that are members of this `Class` (excluding superclasses). `getClasses` returns only the list of public interfaces and classes.

```
public ClassLoader getClassLoader()
```

Returns the `ClassLoader` for this `Class`.

```
public Class getComponentType()
```

Returns the `Component` type of the array that is represented by this `Class`.

```
public Constructor getConstructor(Class[] types) throws
NoSuchMethodException, SecurityException
```

```
public Constructor[] getConstructors() throws SecurityException
```
Returns the `Constructor` object or an array containing the public constructors for this class. The signature of the public constructor that is returned must match exactly the types and sequence of the parameters specified by the `types` array.

```
public Constructor getDeclaredConstructor(Class[] types) throws
NoSuchMethodException, SecurityException
```

```
public Constructor[] getDeclaredConstructors() throws
SecurityException
```
Returns the `Constructor` object or an array containing the constructors for this class. The signature of the public constructor that is returned must match exactly the types and sequence of the parameters specified by the `types` array parameter.

```
public Field getDeclaredField(String field) throws
NoSuchFieldException, SecurityException
```

```
public Field[] getDeclaredFields() throws SecurityException
```
Returns the `Field` object or an array containing all of the fields for the specified matching `field` name for this `Class`.

```
public Method getDeclaredMethod(String method, Class[] types)
throws NoSuchMethodException, SecurityException
```

```
public Method[] getDeclaredMethods() throws SecurityException
```
Returns a `Method` object or an array containing all of the methods for the specified `method` of this `Class`. The requested method's parameter list must match identically the types and sequence of the elements of the `types` array.

```
public Class getDeclaringClass()
```
Returns the declaring class of this `Class`, provided that this `Class` is a member of another class.

```
public Field getField(String field) throws NoSuchFieldException,
SecurityException
```

```
public Field[] getFields() throws SecurityException
```
Returns a `Field` object or an array containing all of the fields of a specified matching `field` name for this `Class`.

```
public Class[] getInterfaces()
```
Returns an array containing all of the interfaces of this `Class`.

```
public Method getMethod(String method, Class[] types) throws
NoSuchMethodException, SecurityException
```

`public Method[] getMethods() throws SecurityException`
Returns a `Method` object or an array containing all of the public methods for the specified public `method` of this `Class`. The requested method's parameter list must match identically the types and sequence of the elements of the `types` array.

`public int getModifiers()`
Returns the encoded integer visibility modifiers for this `Class`. The values can be decoded using the `Modifier` class.

`public String getName()`
Returns the string representation of the name of the type that this `Class` represents.

`public URL getResource(String arg)`
Returns a URL representing the system resource for the class loader of this `Class`.

`public InputStream getResourceAsStream(String arg)`
Returns an input stream representing the `named` system resource from the class loader of this `Class`.

`public Object[] getSigners()`
Returns an array of `Objects` that contains the signers of this `Class`.

`public Class getSuperclass()`
Returns the superclass of this `Class`, or null if this `Class` is an interface or of type `Object`.

`public boolean isArray()`
Returns a true value if this `Class` represents an array type.

`public boolean isAssignableFrom(Class other)`
Returns a true value if this `Class` is the same as a superclass or superinterface of the `other` class.

`public boolean isInstance(Object target)`
Returns a true value if the specified `target` object is an instance of this `Class`.

`public boolean isInterface()`

`public boolean isPrimitive()`
Returns a true value if this `Class` represents an interface class or a primitive type in Java.

`public Object newInstance() throws InstantiationException, IllegalAccessException`
Creates a new instance of this `Class`.

`public String toString()`
Returns a string representation of this `Class` in the form of the word class or interface, followed by the fully qualified name of this `Class`.

Color (java.awt)

A public final class, derived from `Object` and implementing `Serializable`, that is used to represent colors. A color is defined by three components, red, blue, and green, that each have a value ranging from 0 to 255.

Variables and Constants

```
public final static Color black
public final static Color blue
public final static Color cyan
public final static Color darkGray
public final static Color gray
public final static Color green
public final static Color lightGray
public final static Color magenta
public final static Color orange
public final static Color pink
public final static Color red
public final static Color white
public final static Color yellow
```

A constant value that describes the colors black (0, 0, 0), blue (0, 0, 255), cyan (0, 255, 255), darkGray (64, 64, 64), gray (128, 128, 128), green (0, 255, 0), lightGray (192, 192, 192), magenta (255, 0, 255), orange (255, 200, 0), pink (255, 175, 175), red (255, 0, 0), white (255, 255, 255) and yellow (255, 255, 0) as a set of RGB values.

Constructors

```
public Color(float r, float g, float b)
public Color(int rgb)
public Color(int r, int g, int b)
```

Creates a new instance of the color described by the `rgb` value. When passed as a single integer value, the red component is represented in bits 16 to 23, green in 15 to 8, and blue in 0 to 7.

Methods

```
public Color brighter()
public Color darker()
```

Returns a brighter or darker version of this color.

```
public static Color decode(String str) throws
NumberFormatException
```
Returns the color specified by `str`.

```
public boolean equals(Object arg)
```
Returns a true value if this color is equal to `arg`.

```
public int getBlue()
public int getGreen()
public int getRed()
```
Returns the blue, green, or red component value for this color.

```
public static Color getColor(String str)
public static Color getColor(String str, Color default)
public static Color getColor(String str, int default)
```
Returns the color represented in the string `str` (where its value is an integer). If the value is not determined, the color `default` is returned.

```
public static Color getHSBColor(float h, float s, float b)
```
Returns a color specified by the Hue-Saturation-Brightness model for colors, where `h` is the hue, `s` is the saturation, and `b` is the brightness of the desired color.

```
public int getRGB()
```
Returns an integer representation of the RGB value for this color.

```
public int hashCode()
```
Returns the hash code for this color.

```
public static int HSBtoRGB(float hue, float saturation, float
brightness)
```
Converts a `hue`, `saturation`, and `brightness` representation of a color to an RGB value.

```
public static float[] RGBtoHSB(int r, int g, int b, float[]
hsbvals)
```
Converts an RGB representation of a color to an HSB value, placing the converted values into the `hsbvals` array. The RGB value is represented via a red (`r`), green (`g`), and blue (`b`) value.

```
public String toString()
```
Returns a string representation of this color.

Component (`java.awt`)

A public abstract class, derived from `Object` and implementing `ImageObserver`, `MenuContainer`, and `Serializable`, that is the superclass to every AWT item that is represented on screen with a specific size and position.

Variables and Constants

```
public final static float BOTTOM_ALIGNMENT
public final static float LEFT_ALIGNMENT
public final static float RIGHT_ALIGNMENT
public final static float TOP_ALIGNMENT
```
Constant values that represent specified alignments within the component.

```
protected Locale locale
```
Holds the locale for this component.

Constructors

```
protected Component()
```
Creates a new instance of a component.

Methods

```
public synchronized void add(PopupMenu popmenu)
public synchronized void remove(MenuComponent popmenu)
```
Adds or removes the specified popup menu to this component.

```
public synchronized void addComponentListener(ComponentListener
listener)
public synchronized void addFocusListener(FocusListener
listener)
public synchronized void addKeyListener(KeyListener listener)
public synchronized void addMouseListener(MouseListener
listener)
public synchronized void
addMouseMotionListener(MouseMotionListener listener)
public synchronized void
removeComponentListener(ComponentListener listener)
public synchronized void removeFocusListener(FocusListener
listener)
public synchronized void removeKeyListener(KeyListener listener)
public synchronized void removeMouseListener(MouseListener
listener)
public synchronized void
removeMouseMotionListener(MouseMotionListener listener)
```
Adds or removes the specified `listener` to this component.

```
public void addNotify()
public void removeNotify()
```
Notifies the component that a peer must be created or destroyed.

```
public int checkImage(Image img, ImageObserver obs)
public int checkImage(Image img, int width, int height,
ImageObserver obs)
```
Returns the status of the construction of a scaled image `img`. The image created can be scaled to a `width` and `height`. The image `obs` will be informed of the status of the image.

```
public boolean contains(int x, int y)
public boolean contains(Point pt)
```
Returns a true value if this component contains the specified position.

```
public Image createImage(ImageProducer prod)
public Image createImage(int width, int height)
```
Returns a new image created from `prod`. The second method creates another image which is generally offscreen (having `width` and `height`), used for double-buffering drawings.

```
protected final void disableEvents(long mask)
protected final void enableEvents(long mask)
```
Disables or enables all events specified by the `mask` for this component.

```
public final void dispatchEvent(AWTEvent event)
```
Dispatches an `AWTEvent` to this component or one of its subcomponents.

```
public void doLayout()
```
Lays out this component.

```
public float getAlignmentX()
public float getAlignmentY()
```
Returns the horizontal or vertical alignment for this component.

```
public Color getBackground()
public Color getForeground()
public void setBackground(Color clr)
public void setForeground(Color clr)
```
Returns or sets the background or foreground color for this component.

```
public Rectangle getBounds()
public void setBounds(int x, int y, int width, int height)
```
Returns or sets the bounds of this component. Setting the bounds resizes and reshapes this component to the bounding box of `<x, y>` to `<x+width, y+height>`.

```
public ColorModel getColorModel()
```
Returns the color model of this component.

```
public Component getComponentAt(int x, int y)
public Component getComponentAt(Point pt)
```
Returns the component located at the specified point.

```
public Cursor getCursor()
public synchronized void setCursor(Cursor csr)
```
Returns or sets the cursor set for this component.
```
public Font getFont()
public void setFont(Font ft)
```
Returns or sets the font of this component.
```
public FontMetrics getFontMetrics(Font ft)
```
Returns the font metrics of the specified font.
```
public Graphics getGraphics()
```
Returns the graphics context for this component.
```
public Locale getLocale()
public void setLocale(Locale locale)
```
Returns or sets the locale for this component.
```
public Point getLocation()
public Point getLocationOnScreen()
```
Returns the location of this component relative to the containing or screen space.
```
public Dimension getMaximumSize()
public Dimension getMinimumSize()
public Dimension getPreferredSize()
```
Returns the maximum, minimum, or preferred size of this component.
```
public String getName()
public void setName(String str)
```
Returns or sets the name of this component.
```
public Container getParent()
```
Returns the parent container of this component.
```
public Dimension getSize()
public void setSize(Dimension dim)
public void setSize(int width, int height)
```
Returns the size of or resizes this component to the specified dimension(s).
```
public Toolkit getToolkit()
```
Returns the toolkit of this component.
```
public final Object getTreeLock()
```
Returns the AWT object that is used as the base of the component tree and layout operations for this component.
```
public boolean imageUpdate(Image src, int flags, int x, int y,
```

```
int width, int height)
```
Draws more of an image (`src`) as its information becomes available. The exact value of the `x`, `y`, `width`, and `height` variables is dependent on the value of the `flags` variable.

```
public void invalidate()
```
Forces this component to be laid out again by making it "invalid."

```
public boolean isEnabled()
public void setEnabled(boolean toggle)
```
Returns or sets the enabled state of this component.

```
public boolean isFocusTraversable()
```
Returns a true value if this component can be traversed using Tab or Shift-Tab sequences.

```
public boolean isShowing()
public boolean isValid()
```
Returns a true value if this component is visible on screen or does not need to be laid out (valid).

```
public boolean isVisible()
public void setVisible(boolean toggle)
```
Returns or sets the state of this component's visibility.

```
public void list()
public void list(PrintStream outstrm)
public void list(PrintStream outstrm, int spc)
public void list(PrintWriter outstrm)
public void list(PrintWriter outstrm, int spc)
```
Prints a listing of this component's parameters to the print writer stream `outstrm` (default of `System.out`), indenting `spc` spaces (default of 0).

```
public void paint(Graphics gc)
public void print(Graphics gc)
```
Paints or prints this component with the graphics context `gc`.

```
public void paintAll(Graphics gc)
public void printAll(Graphics gc)
```
Paints or prints this component and all of its subcomponents with the graphics context `gc`.

```
protected String paramString()
```
Returns a string describing the parameters of this component.

```
public boolean prepareImage(Image src, ImageObserver obs)
```

```
public prepareImage(Image src, int width, int height,
ImageObserver obs)
```
Downloads the `src` for display. The image can be scaled to a `width` and `height`. The `obs` is informed of the status of the image.

```
protected void processComponentEvent(ComponentEvent event)
protected void processFocusEvent(FocusEvent event)
protected void processKeyEvent(KeyEvent event)
protected void processMouseEvent(MouseEvent event)
protected void processMouseMotionEvent(MouseEvent event)
```
Processes the specified event for this component, sending the event to a registered event listener.

```
protected void processEvent(AWTEvent event)
```
Processes an AWT event for this component, sending it to the appropriate processing routine (i.e., `processComponentEvent` method) for further handling.

```
public void repaint()
public void repaint(int x, int y, int width, int height)
```
Repaints a rectangular portion of this component from `<x, y>` to `<x+width, y+height>`.

```
public void repaint(long msec)
public void repaint(long msec, int x, int y, int width, int
height)
```
Repaints a rectangular portion of this component from `<x, y>` to `<x+width, y+height>` after a delay of `msec` milliseconds.

```
public void requestFocus()
```
Requests that this component get the input focus.

```
public void setLocation(int x, int y)
public void setLocation(Point pt)
```
Moves this component to the specified point in the containing space.

```
public String toString()
```
Returns a string representation of this component.

```
public void transferFocus()
```
Transfers focus from this component to the next component.

```
public void update(Graphics gc)
```
Updates this component using graphics context `gc`.

```
public void validate()
```
Validates this component if needed.

ComponentAdapter (java.awt.event)

A public abstract class, derived from `Object` and implementing `ComponentListener`, that permits a derived class to override the predefined no-op component events.

Constructors

```
public ComponentAdapter()
```

Creates a new instance of a `ComponentAdapter`.

Methods

```
public void componentHidden(ComponentEvent event)
public void componentMoved(ComponentEvent event)
public void componentResized(ComponentEvent event)
public void componentShown(ComponentEvent event)
```

Empty methods that should be overridden in order to implement event handling for AWT components.

ComponentEvent (java.awt.event)

A public class, derived from `AWTEvent`, that represents an AWT component event.

Variables and Constants

```
public static final int COMPONENT_FIRST
public static final int COMPONENT_LAST
```

Constant values that represent the index of the first and last component event ids.

```
public static final int COMPONENT_MOVED
public static final int COMPONENT_RESIZED
public static final int COMPONENT_SHOWN
public static final int COMPONENT_HIDDEN
```

Constant values that represent AWT component event ids.

Constructors

```
public ComponentEvent(Component src, int type)
```

Creates a new instance of a `ComponentEvent` from the specified source and of a specific type.

Methods

```
public Component getComponent()
```
Returns the AWT component that triggered this event.

```
public String paramString()
```
Returns a string containing the parameters of this event.

Container (`java.awt`)

A public abstract class, derived from `Component`, that is the superclass to any AWT component that can contain one or more AWT components.

Constructors

```
protected Container()
```
Creates a new instance of a container.

Methods

```
public Component add(Component item)
public Component add(Component item, int idx)
public void add(Component item, Object constr)
public void add(Component item, Object constr, int idx)
public Component add(String str, Component item)
```
Adds component `item` to this container at index `idx` (or to the end by default). The new item can have constraints (`constr`) applied to it. A string name can be associated with the added component in the case of the last constructor.

```
public void addContainerListener(ContainerListener listener)
public void removeContainerListener(ContainerListener listener)
```
Adds or removes the specified `listener` to this container.

```
protected void addImpl(Component item, Object constr, int idx)
```
Adds component `item` to this container at index `idx`, and passes the constraints for the new item (`constr`) to the layout manager for this container.

```
public void addNotify()
public void removeNotify()
```
Creates or destroys this container's peer.

```
public void doLayout()
```
Lays out the components of this container.

```
public float getAlignmentX()
```

```
public float getAlignmentY()
```
Returns the horizontal or vertical alignment value of this container.

```
public Component getComponent(int idx) throws
ArrayIndexOutOfBoundsException
public Component getComponentAt(int x, int y)
public Component getComponentAt(Point pt)
```
Returns the component that is located at the specified point or index.

```
public int getComponentCount()
```
Returns the number of components in this container.

```
public Component[] getComponents()
```
Returns an array of all of the components in this container.

```
public Insets getInsets()
```
Returns the insets of this container.

```
public LayoutManager getLayout()
public void setLayout(LayoutManager layout)
```
Returns or sets the layout manager of this container.

```
public Dimension getMaximumSize()
public Dimension getMinimumSize()
public Dimension getPreferredSize()
```
Returns the maximum, minimum, or preferred size of this container.

```
public void invalidate()
```
Marks the layout of this container as invalid, forcing the need to lay out the components again.

```
public boolean isAncestorOf(Component comp)
```
Returns a true value if the specified component (`comp`) is contained in the component hierarchy of this container.

```
public void list(PrintStream outstream, int spaces)
public void list(PrintWriter outstream, int spaces)
```
Prints a listing of all of the components of this container to print stream `outstream`, indented a specified number of `spaces` (default of 0).

```
public void paint(Graphics gwin)
public void print(Graphics gwin)
```
Paints or prints this container with graphics context `gwin`.

```
public void paintComponents(Graphics gwin)
public void printComponents(Graphics gwin)
```
Repaints or prints all of the components in this container with graphics context `gwin`.

`protected String paramString()`
Returns a string representation of this container's parameters.

`protected void processContainerEvent(ContainerEvent event)`
Processes any container `event`, passing the event to a registered container listener.

`protected void processEvent(AWTEvent event)`
Handles any `AWTEvent`, invoking `processContainerEvent` for container events, and passing the event to the superclass' `processEvent` otherwise.

`public void remove(Component comp)`

`public void remove(int idx)`
Removes the specified component (or the component at the specified index) from this container.

`public void removeAll()`
Removes all components from this container.

`public void validate()`
Validates this container and all of the subcomponents in it.

`protected void validateTree()`
Validates this container and all subcontainers in it.

ContainerAdapter (`java.awt.event`)

A public abstract class, derived from `Object` and implementing `ContainerListener`, that permits a derived class to override the predefined no-op container events.

Constructors

`public ContainerAdapter()`
Creates a new instance of a `ContainerAdapter`.

Methods

`public void componentAdded(ContainerEvent event)`

`public void componentRemoved(ContainerEvent event)`
Empty methods that should be overridden in order to implement event handling for AWT containers.

ContainerEvent (java.awt.event)

A public class, derived from `ComponentEvent`, that describes a particular AWT container event.

Variables and Constants

`public static final int COMPONENT_ADDED`

`public static final int COMPONENT_REMOVED`

Constant values that represent various container events (a component being added or removed to this container).

`public static final int CONTAINER_FIRST`

`public static final int CONTAINER_LAST`

Constant values that represent the index of the first and last component event ids.

Constructors

`public ContainerEvent(Component src, int type, Component comp)`

Creates a new instance of a `ContainerEvent` with a specified source component, event type, and a defined component (which is being added or removed).

Methods

`public Component getChild()`

Returns the child component that was added or removed, triggering this event.

`public Container getContainer()`

Returns the container in which this event was triggered.

`public String paramString()`

Returns a string containing the parameters of this `ComponentEvent`.

Cursor (java.awt)

A public class, derived from `Object` and implementing `Serializable`, that represents the different states and images of the mouse cursor in a graphical application or applet.

Variables and Constants

`public final static int CROSSHAIR_CURSOR`

`public final static int DEFAULT_CURSOR`

`public final static int E_RESIZE_CURSOR`

```
public final static int HAND_CURSOR
public final static int MOVE_CURSOR
public final static int N_RESIZE_CURSOR
public final static int NE_RESIZE_CURSOR
public final static int NW_RESIZE_CURSOR
public final static int S_RESIZE_CURSOR
public final static int SE_RESIZE_CURSOR
public final static int SW_RESIZE_CURSOR
public final static int TEXT_CURSOR
public final static int W_RESIZE_CURSOR
public final static int WAIT_CURSOR
```

Constant values that represent various cursors.

```
protected static Cursor predefined[]
```

An array used to hold the cursors as they are defined and implemented.

Constructors

```
public Cursor(int cursortype)
```

Creates a new instance of a cursor of the specified type (`cursortype`).

Methods

```
public static Cursor getDefaultCursor()
```

Returns the default cursor.

```
public static Cursor getPredefinedCursor(int cursortype)
```

Returns the cursor of the specified type (`cursortype`).

```
public int getType()
```

Returns the type of this cursor.

Date (`java.util`)

A public class, derived from `Object` and implementing `Serializable` and `Cloneable`, that creates and manipulates a single moment of time.

Constructors

```
public Date()
public Date(long date)
```

Creates a new instance of a `Date` from a specified `date` (time in milliseconds since midnight, January 1, 1970 GMT) or by using the current time.

Methods

```
public boolean after(Date arg)
public boolean before(Date arg)
```
Returns a true value if this `Date` is after/before the date specified in `arg`.
```
public boolean equals(Object arg)
```
Returns a true value if this `Date` is equal to `arg`.
```
public long getTime()
public void setTime(long tm)
```
Returns or sets the time specified by this `Date`. The time is represented as a long integer equal to the number of seconds since midnight, January 1, 1970 UTC.
```
public int hashCode()
```
Returns the hash code for this `Date`.
```
public String toString()
```
Returns a string representation of this `Date`.

DateFormat (`java.text`)

A public abstract class, derived from `Cloneable`, that is used to convert date/time objects to locale-specific strings, and vice versa.

Variables and Constants

```
public static final int DEFAULT
public static final int FULL
public static final int LONG
public static final int MEDIUM
public static final int SHORT
```
Constant values that represent formatting styles.
```
public static final int AM_PM_FIELD
public static final int DATE_FIELD
public static final int DAY_OF_WEEK_FIELD
public static final int DAY_OF_WEEK_IN_MONTH_FIELD
public static final int DAY_OF_YEAR_FIELD
public static final int ERA_FIELD
public static final int HOUR0_FIELD
public static final int HOUR1_FIELD
public static final int HOUR_OF_DAY0_FIELD
```

```
public static final int HOUR_OF_DAY1_FIELD
public static final int MILLISECOND_FIELD
public static final int MINUTE_FIELD
public static final int MONTH_FIELD
public static final int SECOND_FIELD
public static final int TIMEZONE_FIELD
public static final int WEEK_OF_MONTH_FIELD
public static final int WEEK_OF_YEAR_FIELD
public static final int YEAR_FIELD
```

Constant values that represent various fields for date/time formatting.

```
protected Calendar calendar
```

Holds the calendar that this `DateFormat` uses to produce its date/time formatting.

```
protected NumberFormat numberFormat
```

Holds the number format that this `DateFormat` uses to produce its number formatting.

Constructors

```
protected DateFormat()
```

Creates a new instance of a `DateFormat`.

Methods

```
public Object clone()
```

Returns a copy of this `DateFormat`.

```
public boolean equals(Object arg)
```

Returns a true value if this `DateFormat` is equal to `arg`.

```
public final String format(Date src)
```

Formats the specified `Date` object into a string.

```
public abstract StringBuffer format(Date src, StringBuffer dest,
FieldPosition pos)
public final StringBuffer format(Object src, StringBuffer dest,
FieldPosition pos)
```

Formats the source object into the specified destination, starting at field `pos`. This method returns the same value as the destination buffer.

```
public static Locale[] getAvailableLocales()
```

Returns the set of available locales for this `DateFormat`.

```
public Calendar getCalendar()
public void setCalendar(Calendar cal)
```

Returns or sets the calendar associated with this `DateFormat`.

```
public static final DateFormat getDateInstance()
public static final DateFormat getDateInstance(int style)
public static final DateFormat getDateInstance(int style, Locale
locale)
```
Returns the `DateFormat` for the specified or default locale (using the default or specified date formatting style).

```
public static final DateFormat getDateTimeInstance()
public static final DateFormat getDateTimeInstance(int dstyle,
int tstyle)
public static final DateFormat getDateTimeInstance(int dstyle,
int tstyle, Locale locale)
```
Returns the `DateFormat` for the specified or default locale (using the default or specified date and time formatting styles).

```
public static final DateFormat getInstance()
```
Returns the `DateFormat` for the default locale using the short formatting style.

```
public NumberFormat getNumberFormat()
public void setNumberFormat(NumberFormat format)
```
Returns or sets the `NumberFormat` for this `DateFormat`.

```
public static final DateFormat getTimeInstance()
public static final DateFormat getTimeInstance(int style)
public static final DateFormat getTimeInstance(int style, Locale
locale)
```
Returns the `DateFormat` for the specified or default locale (using the default or specified time formatting style).

```
public TimeZone getTimeZone()
public void setTimeZone(TimeZone tz)
```
Returns or sets the time zone for this `DateFormat`.

```
public int hashCode()
```
Returns the hash code for this `DateFormat`.

```
public boolean isLenient()
public void setLenient(boolean lenient)
```
Returns or sets the state of the leniency for this `DateFormat`.

```
public Date parse(String src) throws ParseException
```
Parses the specified source to a `Date` object.

```
public abstract Date parse(String src, ParsePosition pos)
public Object parseObject(String src, ParsePosition pos)
```
Parses the specified source string to a `Date` or `Object`, starting at the specified position.

DateFormatSymbols (java.text)

A public class, derived from `Object` and implementing `Serializable` and `Cloneable`, that contains functionality for formatting both date and time values. This class is usually utilized as part of a `DateFormat` class (or subclass).

Constructors

`public DateFormatSymbols()`

`public DateFormatSymbols(Locale locale)`

Creates a new instance of `DateFormatSymbols` using the specified or default locale.

Methods

`public Object clone()`

Returns a clone of this `DateFormatSymbols`.

`public boolean equals(Object arg)`

Returns a true value if this `DateFormatSymbols` is equal to `arg`.

`public String[] getAmPmStrings()`

`public void setAmPmStrings(String[] newstr)`

Returns or sets the AM/PM strings for this set of symbols.

`public String[] getEras()`

`public void setEras(String[] newstr)`

Returns or sets the eras for this set of symbols.

`public String getLocalPatternChars()`

`public void setLocalPatternChars(String newchars)`

Returns or sets the local pattern characters for date and time for this set of symbols.

`public String[] getMonths()`

`public void setMonths(String[] newmon)`

Returns or sets the full names of months for this set of symbols.

`public String[] getShortMonths()`

`public void setShortMonths(String[] newmon)`

Returns or sets the short names of months for this set of symbols.

`public String[] getShortWeekdays()`

`public void setShortWeekdays(String[] newdays)`

Returns or sets the short names of weekdays for this set of symbols.

`public String[] getWeekdays()`

```
public void setWeekdays(String[] newdays)
```
Returns or sets the full names of weekdays for this set of symbols.

```
public String[][] getZoneStrings()
public void setZoneStrings(String[][] newzone)
```
Returns or sets the time zone strings for this set of symbols.

```
public int hashCode()
```
Returns the hash code for this set of symbols.

DecimalFormat (java.text)

A public class, derived from `NumberFormat`, that is used to format decimal numbers to locale-based strings, and vice versa.

Constructors

```
public DecimalFormat()
public DecimalFormat(String str)
public DecimalFormat(String str, DecimalFormatSymbols sym)
```
Creates a new instance of a `DecimalFormat` from the specified or default pattern, specified or default symbols and using the default locale.

Methods

```
public void applyLocalizedPattern(String str)
public String toLocalizedPattern()
```
Sets or returns the pattern of this `DecimalFormat`. The specified pattern is in a locale-specific format.

```
public void applyPattern(String str)
public String toPattern()
```
Sets or returns the pattern of this `DecimalFormat`.

```
public Object clone()
```
Returns a copy of this `DecimalFormat`.

```
public boolean equals(Object arg)
```
Returns a true value if this `DecimalFormat` is equal to `arg`.

```
public StringBuffer format(double num, StringBuffer dest,
FieldPosition pos)
public StringBuffer format(long num, StringBuffer dest,
FieldPosition pos)
```
Formats the specified Java primitive type starting at `pos`, according to this `DecimalFormat`, placing the resulting string in the specified destination buffer. This method returns the value of the string buffer.

```
public DecimalFormatSymbols getDecimalFormatSymbols()
public void setDecimalFormatSymbols(DecimalFormatSymbols
symbols)
```
Returns or sets the decimal number format symbols for this `DecimalFormat`.

```
public int getGroupingSize()
public void setGroupingSize(int val)
```
Returns or sets the size of groupings for this `DecimalFormat`.

```
public int getMultiplier()
public void setMultiplier(int val)
```
Returns or sets the value of the multiplier for use in percent calculations.

```
public String getNegativePrefix()
public void setNegativePrefix(String val)
```
Returns or sets the prefix for negative numbers for this `DecimalFormat`.

```
public String getNegativeSuffix()
public void setNegativeSuffix(String val)
```
Returns or sets the suffix for negative numbers for this `DecimalFormat`.

```
public String getPositivePrefix()
public void setPositivePrefix(String val)
```
Returns or sets the prefix for positive numbers for this `DecimalFormat`.

```
public String getPositiveSuffix()
public void setPositiveSuffix(String val)
```
Returns or sets the suffix for positive numbers for this `DecimalFormat`.

```
public int hashCode()
```
Returns the hash code for this `DecimalFormat`.

```
public boolean isDecimalSeparatorAlwaysShown()
public void setDecimalSeparatorAlwaysShown(boolean toggle)
```
Returns or sets the state value that allows/prevents the display of the decimal point when formatting integers.

```
public Number parse(String src, ParsePosition pos)
```
Parses the specified string as a long (if possible) or double, starting at position `pos`, and returns a `Number`.

DecimalFormatSymbols (`java.text`)

A public class, derived from `Object` and implementing `Serializable` and `Cloneable`, that contains functionality for formatting decimal values. This class is usually utilized as part of a `DecimalFormat` class (or subclass).

Constructors

`public DecimalFormatSymbols()`

`public DecimalFormatSymbols(Locale locale)`

Creates a new instance of `DecimalFormatSymbols` using the specified or default locale.

Methods

`public Object clone()`

Returns a clone of this `DecimalFormatSymbols`.

`public boolean equals(Object arg)`

Returns a true value if this `DecimalFormatSymbols` is equal to `arg`.

`public char getDecimalSeparator()`

`public void setDecimalSeparator(char separator)`

Returns or sets the character used to separate decimal numbers in this set of symbols.

`public char getDigit()`

`public void setDigit(char num)`

Returns or sets the character used as a digit placeholder in a pattern for this set of symbols.

`public char getGroupingSeparator()`

`public void setGroupingSeparator(char separator)`

Returns or sets the character used to separate groups of thousands for this set of symbols.

`public String getInfinity()`

`public void setInfinity(String str)`

Returns or sets the string used to represent the value of infinity for this set of symbols.

`public char getMinusSign()`

`public void setMinusSign(char minus)`

Returns or sets the character used to represent the minus sign for this set of symbols.

`public String getNaN()`

`public void setNaN(String str)`

Returns or sets the character used to represent a NAN value for this set of symbols.

`public char getPatternSeparator()`

```
public void setPatternSeparator(char separator)
```
Returns or sets the character used to separate positive and negative numbers in a pattern from this set of symbols.

```
public char getPercent()
public void setPercent(char percent)
```
Returns or sets the character used as a percent sign for this set of symbols.

```
public char getPerMill()
public void setPerMill(char perMill)
```
Returns or sets the character used as a mille percent sign for this set of symbols.

```
public char getZeroDigit()
public void setZeroDigit(char zero)
```
Returns or sets the character used to represent zero for this set of symbols.

```
public int hashCode()
```
Returns the hash code for this set of symbols.

Dimension (`java.awt`)

A public class, derived from `Object` and implementing `Serializable`, that is used to encapsulate an object's dimensions (height and width).

Variables and Constants

```
public int height
public int width
```
Variables which contain the height and width of an object.

Constructors

```
public Dimension()
public Dimension(Dimension dim)
public Dimension(int width, int height)
```
Creates a new instance of a dimension from specified dimensions (or 0 width and 0 height by default).

Methods

```
public boolean equals(Object arg)
```
Returns a true value if this dimension is equal to `arg`.

```
public Dimension getSize()
public void setSize(Dimension dim)
```

```
public void setSize(int width, int height)
```
Returns or sets the size of this dimension.

```
public String toString()
```
Returns the string representation of this dimension.

Double (java.lang)

A public final class, derived from `Number`, that contains floating point math operations, constants, methods to compute minimum and maximum numbers, and string manipulation routines related to the `double` primitive type.

Variables and Constants

```
public final static double MAX_VALUE
public final static double MIN_VALUE
```
Constant values that contain the maximum (1.79769313486231570e+308d) and minimum (4.94065645841246544e2324d) possible values of an integer in Java.

```
public final static double NaN
```
A constant value that contains the representation of the Not-A-Number double (0.0d).

```
public final static double NEGATIVE_INFINITY
public final static double POSITIVE_INFINITY
```
Constant values that contain the negative (–1.0d / 0.0d) and positive (1.0d / 0.0d) infinity double.

```
public final static Class TYPE
```
A constant value of the `Double` type class.

Constructors

```
public Double(double arg)
public Double(String arg) throws NumberFormatException
```
Creates an instance of the `Double` class from the parameter `arg`.

Methods

```
public byte byteValue()
public double doubleValue()
public float floatValue()
public int intValue()
public long longValue()
```

```
public short shortValue()
```
Returns the value of the current object as a Java primitive type.

```
public static long doubleToLongBits(double num)
public static double longBitsToDouble(long num)
```
Returns a long bit stream or a double representation of parameter `num`. Bit 63 of the returned `long` is the sign bit, bits 52 to 62 are the exponent, and bits 0 to 51 are the mantissa.

```
public boolean equals(Object param)
```
Returns a true value if this `Double` is equal to the specified parameter (`param`).

```
public int hashCode()
```
Returns a hash code for this `Double`.

```
public boolean isInfinite()
public static boolean isInfinite(double num)
```
Returns true if the current object or `num` is positive or negative infinity, false in all other cases.

```
public boolean isNaN()
public static boolean isNaN(double num)
```
Returns true if the current object or `num` is Not-A-Number, false in all other cases.

```
public static double parseDouble(String str) throws
NumberFormatException
```
Returns the double value represented by `str`.

```
public String toString()
public static String toString(double num)
```
Returns the string representation of the current object or `num` in base 10 (decimal).

```
public static Double valueOf(String str) throws
NumberFormatException
```
Returns a `Double` initialized to the value of `str`.

Error (`java.lang`)

A public class, derived from `Throwable`, that is used to signify program-terminating errors that should not be caught.

Constructors

```
public Error()
public Error(String str)
```
Creates a new instance of an error. A message can be provided via `str`.

Event (java.awt)

A public class, derived from `Object`, that represents event obtained from a graphical user interface.

Variables and Constants

```
public final static int ACTION_EVENT
```
A constant that represents the user desires an action.

```
public final static int ALT_MASK
public final static int CTRL_MASK
public final static int META_MASK
public final static int SHIFT_MASK
```
Constant values which represent the mask for Alt, Control, Meta, and Shift keys modifying events.

```
public Object arg
```
An optional argument used by some events.

```
public final static int BACK_SPACE
public final static int CAPS_LOCK
public final static int DELETE
public final static int DOWN
public final static int END
public final static int ENTER
public final static int ESCAPE
public final static int F1
public final static int F2
public final static int F3
public final static int F4
public final static int F5
public final static int F6
public final static int F7
public final static int F8
public final static int F9
public final static int F10
public final static int F11
public final static int F12
public final static int HOME
public final static int INSERT
public final static int LEFT
```

```
public final static int NUM_LOCK
public final static int PAUSE
public final static int PGDN
public final static int PGUP
public final static int PRINT_SCREEN
public final static int RIGHT
public final static int SCROLL_LOCK
public final static int TAB
public final static int UP
```
Constant values that represent keyboard keys.

```
public int clickCount
```
The number of consecutive clicks during a `MOUSE_DOWN` event.

```
public Event evt
```
The next event to take place, as in a linked list.

```
public final static int GOT_FOCUS
```
An id field constant that represents when an AWT component gets the focus.

```
public int id
```
The numeric identification for this event.

```
public int key
```
The keyboard key that was pressed during this event.

```
public final static int KEY_ACTION
public final static int KEY_ACTION_RELEASE
```
Constant values that represent when the user presses or releases a function key.

```
public final static int KEY_PRESS
public final static int KEY_RELEASE
```
Constant values that represent when the user presses or releases a keyboard key.

```
public final static int LIST_DESELECT
public final static int LIST_SELECT
```
Constant values that represent when the user deselects or selects a list item.

```
public final static int LOAD_FILE
public final static int SAVE_FILE
```
Constant values that represent when a file load or save event occurs.

```
public final static int LOST_FOCUS
```
An id field constant that represents when an AWT component loses the focus.

```
public int modifiers
```
Value of any key modifiers for this event.

```
public final static int MOUSE_DOWN
```

```
public final static int MOUSE_DRAG
public final static int MOUSE_ENTER
public final static int MOUSE_EXIT
public final static int MOUSE_MOVE
public final static int MOUSE_UP
```
Constant values that represent mouse events.

```
public final static int SCROLL_ABSOLUTE
```
An id field constant that represents when the user has moved the bubble in a scroll bar.

```
public final static int SCROLL_BEGIN
public final static int SCROLL_END
```
Constant values that represent the scroll begin or ending event.

```
public final static int SCROLL_LINE_DOWN
public final static int SCROLL_LINE_UP
```
Constant values that represent when the user has clicked in the line down or up area of the scroll bar.

```
public final static int SCROLL_PAGE_DOWN
public final static int SCROLL_PAGE_UP
```
Constant values that represent when the user has clicked in the page down or up area of the scroll bar.

```
public Object target
```
The object that this event was created from or took place over.

```
public long when
```
The time stamp of this event. Represented as the number of milliseconds since midnight, January 1, 1970 UTC.

```
public final static int WINDOW_DEICONIFY
public final static int WINDOW_DESTROY
public final static int WINDOW_EXPOSE
public final static int WINDOW_ICONIFY
public final static int WINDOW_MOVED
```
Constant values that represent various window events.

```
public int x
public int y
```
The horizontal or vertical coordinate location of this event.

Constructors

```
public Event(Object obj, int id, Object arg)
public Event(Object obj, long ts, int id, int x, int y, int key,
int state)
```

```
public Event(Object obj, long ts, int id, int x, int y, int key,
int state, Object arg)
```

Creates a new instance of an event with an initial target `Object` (`obj`), `id`, `x` location, `y` location, `key`, modifier `state`, time stamp (`ts`), and argument (`arg`).

Methods

```
public boolean controlDown()
public boolean metaDown()
public boolean shiftDown()
```

Returns a true value if the Control, Meta, or Shift key is down for this event.

```
protected String paramString()
```

Returns the parameter string for this event.

```
public String toString()
```

Returns a string representation of this event.

```
public void translate(int xval, int yval)
```

Translates this event, modifying the `x` and `y` coordinates for this event by adjusting the `x` location by `xval` and the `y` location by `yval`.

Exception (`java.lang`)

A public class, derived from `Throwable`, that catches conditions that are thrown by methods.

Constructors

```
public Exception()
public Exception(String str)
```

Creates a new instance of an exception. A message can be provided via `str`.

Float (`java.lang`)

A public final class, derived from `Number`, that contains floating point math operations, constants, methods to compute minimum and maximum numbers, and string manipulation routines related to the primitive `float` type.

Variables and Constants

```
public final static float MAX_VALUE
public final static float MIN_VALUE
```
Constant values that contain the maximum possible value (3.40282346638528860e+38f) or the minimum possible value (1.40129846432481707e245f) of a float in Java.

```
public final static float NaN
```
A constant value that contains the representation of the Not-A-Number float (0.0f).

```
public final static float NEGATIVE_INFINITY
public final static float POSITIVE_INFINITY
```
Constant values that contain the representation of the negative (–1.0f / 0.0f) or positive (1.0f / 0.0f) infinity float.

```
public final static Class TYPE
```
The `Float` constant value of the float type class.

Constructors

```
public Float(double arg)
public Float(float arg) throws NumberFormatException
public Float(String arg)
```
Creates an instance of the `Float` class from the parameter `arg`.

Methods

```
public byte byteValue()
public float floatValue()
public double doubleValue()
public int intValue()
public long longValue()
public short shortValue()
```
Returns the value of the current object as a Java primitive type.

```
public boolean equals(Object arg)
```
Returns the result of an equality comparison against `arg`.

```
public static int floatToIntBits(float num)
public static float intBitsToFloat(int num)
```
Returns the bit stream or float equivalent of the parameter `num` as an `int`. Bit 31 of the `int` returned value is the sign bit, bits 23 to 30 are the exponent, while bits 0 to 22 are the mantissa.

```
public int hashCode()
```
Returns a hash code for this object.

```
public boolean isInfinite()
public static boolean isInfinite(float num)
```
Returns true if the current object or `num` is positive or negative infinity, false in all other cases.

```
public boolean isNaN()
public static boolean isNaN(float num)
```
Returns true if the current object or `num` is Not-A-Number, false in all other cases.

```
public static float parseFloat(String str) throws
NumberFormatException
```
Returns the float value represented by `str`.

```
public String toString()
public static String toString(float num)
```
Returns the string representation of the current object or `num`.

```
public static Float valueOf(String str) throws
NumberFormatException
```
Returns a `Float` initialized to the value of `str`.

FlowLayout (`java.awt`)

A public class, derived from `Object` implementing `LayoutManager` and `Serializable`, that lays out components in a sequential horizontal order using their preferred size.

Variables and Constants

```
public final static int CENTER
public final static int LEFT
public final static int RIGHT
```
Constant values indicating areas of the flow layout manager.

Constructors

```
public FlowLayout()
public FlowLayout(int al)
public FlowLayout(int al, int hg, int vg)
```
Creates a new instance of a flow layout and gives it `al` alignment (default of centered) with a `vg` vertical and `hg` horizontal gap (default of 0).

Methods

```
public void addLayoutComponent(String str, Component cpnt)
public void removeLayoutComponent(Component cpnt)
```
Adds or removes a component to/from this layout manager. When adding a component, a name may be specified.

```
public int getAlignment()
public void setAlignment(int alg)
```
Returns or sets the alignment value for this layout manager.

```
public int getHgap()
public int getVgap()
```
Returns the value of the horizontal or vertical gap between components laid out by this layout manager.

```
public void layoutContainer(Container cont)
```
Lays out the specified container with this layout manager.

```
public Dimension minimumLayoutSize(Container cont)
public Dimension preferredLayoutSize(Container cont)
```
Returns the minimum or preferred size of the specified container when laid out by this layout manager.

```
public void setHgap(int hg)
public void setVgap(int vg)
```
Sets the horizontal or vertical gap for this layout manager.

```
public String toString()
```
Returns a string representation of this layout manager.

FocusAdapter (`java.awt.event`)

A public abstract class, derived from `Object` and implementing `FocusListener`, that permits derived classes to override the predefined no-op focus events.

Constructors

```
public FocusAdapter()
```
Creates a new instance of a `FocusAdapter`.

Methods

```
public void focusGained(FocusEvent event)
public void focusLost(FocusEvent event)
```
Empty methods that should be overridden in order to implement event handling for AWT focus-based events.

FocusEvent (java.awt.event)

A public class, derived from `ComponentEvent`, that describes a particular AWT focus event.

Variables and Constants

```
public static final int FOCUS_FIRST
public static final int FOCUS_LAST
```
Constant values that represent the index of the first and last focus event ids.

```
public static final int FOCUS_GAINED
public static final int FOCUS_LOST
```
Constant values that represent the gain and loss of focus events.

Constructors

```
public FocusEvent(Component src, int type)
public FocusEvent(Component src, int type, boolean toggle)
```
Creates a new instance of a `FocusEvent` from the specified source, having a defined event type and toggling this event as a temporary change of focus (false by default).

Methods

```
public boolean isTemporary()
```
Returns the status value of the temporary focus toggle.

```
public String paramString()
```
Returns a string containing the parameters of this `FocusEvent`.

Font (java.awt)

A public class, derived from `Object` and implementing `Serializable`, that represents a GUI font.

Variables and Constants

```
public final static int BOLD
public final static int ITALIC
public final static int PLAIN
```
Constant values that indicate the style of the font.

`protected String name`
The name of the font.

`protected int size`
The size of the font in pixels.

`protected int style`
The style of the font.

Constructors

`public Font(String str, int st, int sz)`
Creates a new font with an initial name (`str`), style (`st`), and size (`sz`).

Methods

`public static Font decode(String arg)`
Returns the requested font from a specified string.

`public boolean equals(Object obj)`
Returns a true value if this font is equal to `obj`.

`public String getFamily()`
Returns the name of the family this font belongs to.

`public static Font getFont(String str)`

`public static Font getFont(String str, Font ft)`
Returns the font named `str`. If the font cannot be located, the second method returns `ft` as the default.

`public String getName()`
Returns the name of this font.

`public FontPeer getPeer()`
Returns the peer of this font.

`public int getSize()`

`public int getStyle()`
Returns the size or style of this font.

`public int hashCode()`
Returns the hash code for this font.

`public boolean isBold()`

`public boolean isItalic()`

`public boolean isPlain()`
Returns a true value if this font is bolded, italicized, or plain.

`public String toString()`
Returns a string representation of this font.

FontMetrics (java.awt)

A public class, derived from `Object` and implementing `Serializable`, that provides detailed information about a particular font.

Variables and Constants

```
protected Font font
```
The font upon which the metrics are generated.

Constructors

```
protected FontMetrics(Font f)
```
Creates a new instance of metrics from a given font `f`.

Methods

```
public int bytesWidth(byte[] src, int offset, int size)
public int charsWidth(char[] src, int offset, int size)
```
Returns the advance width for displaying the subarray of `src`, starting at index `offset`, and having a length of `size`.

```
public int charWidth(char c)
public int charWidth(int c)
```
Returns the advance width of the character `c` for the font in this font metric.

```
public int getAscent()
public int getDescent()
```
Returns the amount of ascent or descent for the font in this font metric.

```
public Font getFont()
```
Returns the font in this font metric.

```
public int getHeight()
```
Returns the standard height of the font in this font metric.

```
public int getLeading()
```
Returns the standard leading of the font in this font metric.

```
public int getMaxAdvance()
```
Returns the maximum amount of advance for the font in this font metric.

```
public int getMaxAscent()
public int getMaxDescent()
```
Returns the maximum amount of ascent or descent for the font in this font metric.

`public int[] getWidths()`
Returns an `int` array containing the advance widths of the first 256 characters of the font.

`public int stringWidth(String str)`
Returns the advance width of the string `str` as represented by the font in this font metric.

`public String toString()`
Returns a string representation of the font metrics.

Format (`java.text`)

A public abstract class, derived from `Object` and implementing `Cloneable` and `Serializable`, which is used to format locale-based values into Strings, and vice versa.

Constructors

`public Format()`
Creates a new instance of a `Format`.

Methods

`public Object clone()`
Returns a copy of this `Format`.

`public final String format(Object arg)`
Returns a formatted string from `arg`.

`public abstract StringBuffer format(Object arg, StringBuffer dest, FieldPosition pos)`
Formats the specified argument (starting at field `pos`) into a string, and appends it to the specified `StringBuffer`. This method returns the same value as the destination buffer.

`public Object parseObject(String src) throws ParseException`
Parses the specified source string into a formatted object.

`public abstract Object parseObject(String src, ParsePosition pos)`
Parses the specified source string into a formatted object starting at the specified `ParsePosition`.

Graphics (java.awt)

A public abstract class, derived from `Object`, that provides many useful drawing methods and tools for the manipulation of graphics. A `Graphics` object defines a context in which the user draws.

Constructors

```
protected Graphics()
```

Creates a new `Graphics` instance. This constructor cannot be called directly.

Methods

```
public abstract void clearRect(int x, int y, int width, int
height)
```

Draws a rectangle (with no fill pattern) in the current background color at position `<x, y>`, and having a `width` and `height`.

```
public abstract void clipRect(int x, int y, int width, int
height)
```

Sets a clipping rectangle at position `<x, y>` and having a `width` and `height`.

```
public abstract void copyArea(int x, int y, int width, int
height, int xoffset, int yoffset)
```

Copies a graphic rectangular area at position `<x, y>` and having a `width` and `height`, to position `newx` and `newy`.

```
public abstract Graphics create()
public Graphics create(int x, int y, int width, int height)
```

Returns a copy of this graphics context from position `<x, y>`, and having a `width` and `height`. In the case of the first method, the entire area is copied.

```
public abstract void dispose()
```

Disposes this graphics context.

```
public void draw3DRect(int x, int y, int width, int height,
boolean toggle)
```

Draws a 3D rectangle at position `<x, y>` and having a `width` and `height`. If `toggle` is true, the rectangle will appear raised; otherwise, it will appear indented.

```
public abstract void drawArc(int x, int y, int width, int
height, int sAngle, int aAngle)
```

Draws an arc with a starting position `<x, y>` and having a `width` and `height`. The start angle (`sAngle`) and arc angle (`aAngle`) are both measured in degrees and describe the starting and ending angle of the arc.

```
public void drawBytes(byte[] src, int index, int ln, int x,
int y)
public void drawChars(char[] src, int index, int ln, int x,
int y)
```

Draws `ln` bytes or characters of array `src` (starting at the offset `index`) at position `<x, y>`.

```
public abstract boolean drawImage(Image src, int x, int y, Color
bgc, ImageObserver obsv)
public abstract boolean drawImage(Image src, int x, int y,
ImageObserver obsv)
```

Draws a graphic image (`src`) at position `<x, y>`. Any transparent color pixels are drawn as `bgc`, and the `obsv` monitors the progress of the image.

```
public abstract boolean drawImage(Image src, int x, int y, int
width, int height, Color bgc, ImageObserver obsv)
public abstract boolean drawImage(Image src, int x, int y, int
width, int height, ImageObserver obsv)
```

Draws a graphic image (`src`) at position `<x, y>` and having a `width` and `height`. Any transparent color pixels are drawn as `bgc`, and the `obsv` monitors the progress of the image.

```
public abstract boolean drawImage(Image src, int xsrc1, int
ysrc1, int xsrc1, int ysrc2, int xdest1, int ydest1, int xdest1,
int ydest2, Color bgc, ImageObserver obsv)
public abstract boolean drawImage(Image src, int xsrc1, int
ysrc1, int xsrc1, int ysrc2, int xdest1, int ydest1, int xdest1,
int ydest2, ImageObserver obsv)
```

Draws a graphic image (`src`) from the area defined by the bounding rectangle `<xsrc1, ysrc1>` to `<xsrc2, ysrc2>` in the area defined by the bounding rectangle `<xdest1, ydest1>` to `<xdest2, ydest2>`. Any transparent color pixels are drawn as `bgc`, and the `obsv` monitors the progress of the image.

```
public abstract void drawLine(int xsrc, int ysrc, int xdest, int
ydest)
```

Draws a line from position `<xsrc, ysrc>` to `<xdest, ydest>`.

```
public abstract void drawOval(int xsrc, int ysrc, int width, int
height)
```

Draws an oval starting at position `<xsrc, ysrc>` and having a `width` and `height`.

```
public abstract void drawPolygon(int[] x, int[] y, int num)
public void drawPolygon(Polygon poly)
```

Draws a polygon constructed from `poly` or an array of `x` points, `y` points, and a number of points in the polygon (`num`).

```
public void drawRect(int xsrc, int ysrc, int width, int height)
```

```
public abstract void drawRoundRect(int xsrc, int ysrc, int
width, int height, int awd, int aht)
```
Draws a rectangle with or without rounded corners at position <xsrc, ysrc> and having a width and height. The shape of the rounded corners are determined by the width of the arc (awd) and the height of the arc (aht).

```
public abstract void drawString(String str, int x, int y)
```
Draws the string str at position <x, y> in this Graphic's current font and color.

```
public void fill3DRect(int x, int y, int width, int height,
boolean toggle)
```
Draws a filled 3D rectangle at position <x, y> and having a width and height. The rectangle is filled with this Graphic's current color, and if toggle is true, the rectangle is drawn raised. (Otherwise it is drawn indented.)

```
public abstract void fillArc(int x, int y, int width, int
height, int sAngle, int aAngle)
```
Draws a filled arc at position <x, y> and having a width and height. The arc has a starting angle of sAngle and an ending angle of aAngle.

```
public abstract void fillOval(int x, int y, int width, int
height)
```
Draws a filled oval at position <x, y> and having a width and height.

```
public abstract void fillPolygon(int[] x, int[] y, int num)
public void fillPolygon(Polygon poly)
```
Draws a filled polygon defined by poly or the arrays x, y and the number of points in the polygon, num.

```
public abstract void fillRect(int x, int y, int width, int
height)
public abstract void fillRoundRect(int x, int y, int width, int
height, int aWidth, int aHeight)
```
Draws a filled rectangle with or without rounded corners at position <x, y> and having a width and height. The shape of the rounded corners are determined by the width of the arc (aWidth) and the height of the arc (aHeight).

```
public void finalize()
```
Disposes of the current graphics context.

```
public abstract Shape getClip()
```
Returns a shape object of the current clipping area for this graphics context.

```
public abstract Rectangle getClipBounds()
```
Returns a rectangle describing the bounds of the current clipping area for this graphics context.

```
public abstract Color getColor()
```

```
public abstract void setColor(Color clr)
```
Returns or sets the current color for this graphics context.

```
public abstract Font getFont()
public abstract void setFont(Font ft)
```
Returns or sets the current font of this graphics context.

```
public FontMetrics getFontMetrics()
public abstract FontMetrics getFontMetrics(Font fn)
```
Returns the font metrics associated with this graphics context or font `fn`.

```
public abstract void setClip(int x, int y, int width, int
height)
public abstract void setClip(Shape shp)
```
Sets the clipping area for this graphics context to be at position `<x, y>` and having a `width` and `height` or to be of a specified shape (`shp`).

```
public abstract void setPaintMode()
```
Sets the current graphics context's paint mode to overwrite any subsequent destinations with the current color.

```
public abstract void setXORMode(Color clr)
```
Sets the current graphics context's paint mode to overwrite any subsequent destinations with the alternating current color and `clr` color.

```
public String toString()
```
Returns a string representation of this graphics context.

```
public abstract void translate(int x, int y)
```
Modifies the origin of this graphics context to be relocated to `<x, y>`.

GregorianCalendar (`java.util`)

A public class, derived from `Calendar`, that represents the standard world Gregorian calendar.

Variables and Constants

```
AD
BC
```
Constant values representing periods of an era.

Constructors

```
public GregorianCalendar()
public GregorianCalendar(Locale locale)
public GregorianCalendar(TimeZone zone)
```

```
public GregorianCalendar(TimeZone zone, Locale locale)
```
Creates a new `GregorianCalendar` from the current time in the specified time zone (or the default) and the specified locale (or the default).

```
public GregorianCalendar(int year, int month, int date)
public GregorianCalendar(int year, int month, int date, int
hour, int min)
public GregorianCalendar(int year, int month, int date, int
hour, int min, int sec)
```
Creates a new `GregorianCalendar`, setting the year, month, date, hour, minute, and seconds of the time fields.

Methods

```
public void add(int field, int val)
```
Adds (or subtracts in the case of a negative `val`) an amount of days or time from the specified `field`.

```
public boolean after(Object arg)
public boolean before(Object arg)
```
Returns a true value if this `GregorianCalendar` date is after or before the date specified by `arg`.

```
public Object clone()
```
Returns a clone of this `GregorianCalendar`.

```
protected void computeFields()
protected void computeTime()
```
Computes the values of the time fields based on the currently set time (`computeFields()`) or computes the time based on the currently set time fields (`computeTime()`) for this `GregorianCalendar`.

```
public boolean equals(Object arg)
```
Returns a true value if this `GregorianCalendar` is equal to the value of `arg`.

```
public int getGreatestMinimum(int fld)
public int getLeastMaximum(int fld)
```
Returns the largest allowable minimum or smallest allowable maximum value for the specified field.

```
public final Date getGregorianChange()
public void setGregorianChange(Date dt)
```
Returns or sets the date of the change from Julian to Gregorian calendars for this calendar. The default value is October 15, 1582 (midnight local time).

```
public int getMaximum(int fld)
public int getMinimum(int fld)
```
Returns the largest or smallest allowable value for the specified field.

`public synchronized int hashCode()`
Returns the hash code for this `GregorianCalendar`.

`public boolean isLeapYear(int year)`
Returns a true value if the specified year is a leap year.

`public void roll(int fld, boolean direction)`
Adds one single unit of time to the specified date/time field. A true value specified for `direction` increases the field's value, false decreases it.

GridBagConstraints (`java.awt`)

A public class, derived from `Object` and implementing `Cloneable`, that specifies the layout constraints for each component laid out with a `GridBagLayout`.

Variables and Constants

`public int anchor`
Determines where to place a component that is smaller in size than its display area in the gridbag.

`public final static int BOTH`
`public final static int HORIZONTAL`
`public final static int NONE`
`public final static int VERTICAL`
Constant values that indicate the direction(s) that the component should grow.

`public final static int CENTER`
`public final static int EAST`
`public final static int NORTH`
`public final static int NORTHEAST`
`public final static int NORTHWEST`
`public final static int SOUTH`
`public final static int SOUTHEAST`
`public final static int SOUTHWEST`
`public final static int WEST`
Constant values that indicate where the component should be placed in its display area.

`public int fill`
Determines how to resize a component that is smaller than its display area in the gridbag.

`public int gridheight`

`public int gridwidth`
: Specifies the number of vertical and horizontal cells the component shall occupy.

`public int gridx`

`public int gridy`
: Describes horizontal and vertical cell locations (indices) in the gridbag, where `gridx=0` is the leftmost cell and `gridy=0` is the topmost cell.

`public Insets insets`
: Defines the amount of space (in pixels) around the component in its display area.

`public int ipadx`

`public int ipady`
: Defines the amount of space (in pixels) to add to the minimum horizontal and vertical size of the component.

`public final static int RELATIVE`
: A constant that specifies that this component is the next to last item in its gridbag row or that it should be placed next to the last item added to the gridbag.

`public final static int REMAINDER`
: A constant that specifies that this component is the last item in its gridbag row.

`public double weightx`

`public double weighty`
: Specifies the weight of horizontal and vertical growth of this component relative to other components during a resizing event. A larger value indicates a higher percentage of growth for this component.

Constructors

`public GridBagConstraints()`
: Creates a new instance of `GridBagConstraints`.

Methods

`public Object clone()`
: Creates a copy of these gridbag constraints.

GridBagLayout (`java.awt`)

A public class, derived from `Object` and implementing `Serializable` and `LayoutManager`, that creates a gridlike area for component layout. Unlike

GridLayout, GridBagLayout does not force the components to be the same size or to be constrained to one cell.

Variables and Constants

```
public double columnWeights[]
public int columnWidths[]
```
Holds the weights and widths of each column of this GridBagLayout.

```
protected Hashtable comptable
```
A hashtable of the components managed by this layout manager.

```
protected GridBagConstraints defaultConstraints
```
Holds the default constraints for any component laid out by this layout manager.

```
protected GridBagLayoutInfo layoutInfo
```
Holds specific layout information (such as the list of components or the constraints of this manager) for this GridBagLayout.

```
protected final static int MAXGRIDSIZE
```
A constant value that contains the maximum (512) number of grid cells that can be laid out by this GridBagLayout.

```
protected final static int MINSIZE
```
A constant value that contains the minimum (1) number of cells contained within this GridBagLayout.

```
protected final static int PREFERREDSIZE
```
A constant value that contains the preferred (2) number of cells contained within this GridBagLayout.

```
public int rowHeights[]
public double rowWeights[]
```
Holds the heights and weights of each row of this GridBagLayout.

Constructors

```
public GridBagLayout()
```
Creates a new instance of a GridBagLayout.

Methods

```
public void addLayoutComponent(Component item, Object
constraints)
```
Adds the component item to this layout manager using the specified constraints on the item.

`public void addLayoutComponent(String str, Component item)`
Adds the component `item` to this layout manager and names it `str`.

`protected void AdjustForGravity(GridBagConstraints constraints, Rectangle rect)`
Sets the characteristics of `rect` based on the specified constraints.

`protected void ArrangeGrid(Container parent)`
Arranges the entire grid on the parent.

`public GridBagConstraints getConstraints(Component item)`
Returns a copy of the constraints for the `item` component.

`public float getLayoutAlignmentX(Container parent)`

`public float getLayoutAlignmentY(Container parent)`
Returns the horizontal and vertical alignment values for the specified container.

`public int[][] getLayoutDimensions()`
Returns a two-dimensional array in which the zero index of the first dimension holds the minimum width of each column and the one index of the first dimension holds the minimum height of each column.

`protected GridBagLayoutInfo GetLayoutInfo(Container parent, int sizeflag)`
Computes and returns a `GridBagLayoutInfo` object for components associated with the specified parent container.

`public Point getLayoutOrigin()`
Returns this layout's point of origin.

`public double[][] getLayoutWeights()`
Returns a two-dimensional array in which the zero index of the first dimension holds the weight in the *x* direction of each column and the one index of the first dimension holds the weight in the *y* direction of each column.

`protected Dimension GetMinSize(Container parent, GridBagLayoutInfo info)`
Returns the minimum size for the specified parent container based on laying out the container using the specified `GridBagLayoutInfo`.

`public void invalidateLayout(Container cont)`
Forces this layout manager to discard any cached layout information about the specified container.

`public void layoutContainer(Container cont)`
Lays out the specified container with this layout manager.

`public Point location(int x, int y)`
Returns the upper right corner of the cell in this `GridBagLayout` with dimensions greater than the specified `<x, y>` coordinate.

```
protected GridBagConstraints lookupConstraints(Component item)
```
Returns the actual constraints for the specified component.

```
public Dimension maximumLayoutSize(Container cont)
public Dimension minimumLayoutSize(Container cont)
public Dimension preferredLayoutSize(Container cont)
```
Returns the maximum, minimum, or preferred size of the specified container when laid out by this layout manager.

```
public void removeLayoutComponent(Component comp)
```
Removes the specified component from this layout manager.

```
public void setConstraints(Component item, GridBagConstraints
constraints)
```
Sets the `constraints` for the `item` component in this layout manager.

GridLayout (`java.awt`)

A public class, derived from `Object` and implementing `Serializable` and `LayoutManager`, that creates a grid area of equal sized rectangles to lay out components in.

Constructors

```
public GridLayout()
public GridLayout(int r, int c)
```
Creates a new instance of a `GridLayout` with a dimension of `r` rows and `c` columns (default of 1 by any).

```
public GridLayout(int r, int c, int hg, int vg)
```
Creates a new instance of a `GridLayout` with a dimension of `r` rows and `c` columns. The grid cells have a `hg` pixel horizontal gap and a `vg` pixel vertical gap.

Methods

```
public void addLayoutComponent(String str, Component comp)
public void removeLayoutComponent(Component comp)
```
Adds or removes the specified component. When adding, the component can be given a name (`str`).

```
public int getColumns()
public void setColumns(int val)
```
Returns or sets the number of columns of this layout manager.

```
public int getHgap()
```

```
public int getVgap()
```
Returns the value of the horizontal or vertical gap for this layout manager.

```
public int getRows()
public void setRows(int val)
```
Returns or sets the number of rows of this layout manager.

```
public void layoutContainer(Container cont)
```
Lays out the specified container with this layout manager.

```
public Dimension minimumLayoutSize(Container cont)
public Dimension preferredLayoutSize(Container cont)
```
Returns the minimum or preferred size of the specified container when laid out with this layout manager.

```
public void setHgap(int val)
public void setVgap(int val)
```
Sets the horizontal or vertical gap for this layout manager to `val`.

Hashtable (`java.util`)

A public class, derived from `Dictionary` and implementing `Serializable` and `Cloneable`, that allows for the storing of objects that have a relationship with a key. You can then use this key to access the object stored.

Constructors

```
public Hashtable()
public Hashtable(int size)
public Hashtable(int size, float load) throws
IllegalArgumentException
```
Creates a new instance of a hashtable, setting the initial capacity (or using the default size of 101) and a load factor (default of 0.75). The initial capacity sets the number of objects the table can store, and the load factor value is the percentage filled the table may become before being resized.

Methods

```
public void clear()
```
Removes all keys and elements from this `Hashtable`.

```
public Object clone()
```
Returns a clone of this `Hashtable` (the keys and values are not cloned).

```
public boolean contains(Object arg) throws NullPointerException
```
Returns a true value if this `Hashtable` contains a key that is related to the element `arg`.

```
public boolean containsKey(Object obj)
```
Returns a true value if this `Hashtable` contains an entry for the key at `obj`.

```
public Enumeration elements()
public Enumeration keys()
```
Returns an enumerated list of all of the elements or keys of this `Hashtable`.

```
public Object get(Object obj)
public Object put(Object obj, Object arg) throws
NullPointerException
public Object remove(Object obj)
```
Returns, inserts, or removes the element `arg` that corresponds to the key `obj`.

```
public boolean isEmpty()
```
Returns a true value if the `Hashtable` is empty.

```
protected void rehash()
```
Resizes this `Hashtable`. The method is invoked automatically when the number of keys exceeds the capacity and load factor.

```
public int size()
```
Returns the number of elements in this `Hashtable`.

```
public String toString()
```
Returns a string representation of this `Hashtable`'s key-element pairings.

Image (`java.awt`)

A public abstract class, derived from `Object`, that is used to manage graphic images.

Variables and Constants

```
public final static int SCALE_AREA_AVERAGING
public final static int SCALE_DEFAULT
public final static int SCALE_FAST
public final static int SCALE_REPLICATE
public final static int SCALE_SMOOTH
```
Constant values used to indicate specific scaling algorithms.

```
public final static Object UndefinedProperty
```
A constant value that is returned whenever an undefined property for an image is attempted to be obtained.

Constructors

```
public Image()
```
Creates a new instance of an image.

Methods

```
public abstract void flush()
```
Frees the cache memory containing this image.

```
public abstract Graphics getGraphics()
```
Returns a newly created graphics context for drawing off-screen images.

```
public abstract int getHeight(ImageObserver obs)
public abstract int getWidth(ImageObserver obs)
```
Returns the height or width of this image. If the height is not known, a –1 is returned and the `obs` is informed later.

```
public abstract Object getProperty(String property,
ImageObserver obs)
```
Returns the value of the `property` for this image. If the value is not known, a null is returned and `obs` is informed later.

```
public Image getScaledInstance(int width, int height, int algo)
```
Returns a scaled version of this image. The new image is scaled to `width` pixels by `height` pixels using the specified scaling algorithm (`algo`). If either of the new width or height values are –1, then the new image will maintain the aspect ratios of the old image.

```
public abstract ImageProducer getSource()
```
Returns the source image producer for this image.

ImageIcon (`javax.swing`)

A public class, derived from `Object` and implementing `Accessible`, `Icon`, and `Serializable`, that represents an icon based on an image.

Constructors

```
public ImageIcon()
public ImageIcon(byte[] imageData)
public ImageIcon(byte[] imageData, String description)
public ImageIcon(Image image)
public ImageIcon(Image image, String description)
public ImageIcon(String filename)
```

```
public ImageIcon(String filename, String description)
public ImageIcon(URL location)
public ImageIcon(URL location, String description)
```
Creates an icon using an image described by raw image data (in a supported format such as GIF or JPEG), and `Image` object, a file, or a URL. An optional description can be specified as well.

Methods

```
public String getDescription()
```
Returns the description of this image icon.
```
public int getIconHeight()
public int getIconWidth()
```
Returns this icon's height or width.
```
public Image getImage()
```
Returns this icon's image.
```
public void paintIcon(Component observer, Graphics page, int x,
int y)
```
Paints this icon on the specified graphics context at the specified location using the specified image observer.
```
public setDescription(String description)
public setImage(Image image)
```
Sets the description or the image for this icon.

InputEvent (`java.awt.event`)

A public abstract class, derived from `ComponentEvent`, that describes a particular AWT input event.

Variables and Constants

```
public static final int ALT_MASK
public static final int BUTTON1_MASK
public static final int BUTTON2_MASK
public static final int BUTTON3_MASK
public static final int CTRL_MASK
public static final int META_MASK
public static final int SHIFT_MASK
```
Constant values which represent various keyboard and mouse masks.

Methods

`public void consume()`
Consumes this event, preventing it from being passed to its peer component.

`public int getModifiers()`
Returns the modifiers for this event.

`public long getWhen()`
Returns the time stamp of this event.

`public boolean isConsumed()`
Returns a true value if this event is consumed.

`public boolean isAltDown()`

`public boolean isControlDown()`

`public boolean isMetaDown()`

`public boolean isShiftDown()`
Returns a true value if the Alt, Control, Meta, or Shift key is depressed during this event.

InputStream (`java.io`)

A public abstract class, derived from `Object`, that is the parent class of any type of input stream that reads bytes.

Constructors

`public InputStream()`
Generally called only by subclasses, this constructor creates a new instance of an `InputStream`.

Methods

`public int available() throws IOException`
Returns the number of available bytes that can be read. This method returns a 0 (zero) value and should be overridden by a subclass implementation.

`public void close() throws IOException`
Closes the input stream. This method has no functionality and should be overridden by a subclass implementation.

`public void mark(int size)`
Sets a mark in the input stream, allowing a rereading of the stream data to occur if the reset method is invoked. The `size` parameter indicates how many bytes may be read following the mark being set, before the mark is considered invalid.

`public boolean markSupported()`

Returns a true value if this `InputStream` object supports the mark and reset methods. This method always returns a false value and should be overridden by a subclass implementation.

`public abstract int read() throws IOException`

Reads the next byte of data from this `InputStream` and returns it as an `int`. This method has no functionality and should be implemented in a subclass. Execution of this method will block until data is available to be read, the end of the input stream occurs, or an exception is thrown.

`public int read(byte[] dest) throws IOException`

`public int read(byte[] dest, int offset, int size) throws IOException`

Reads from this `InputStream` into the array `dest`, and returns the number of bytes read. `size` specifies the maximum number of bytes read from this `InputStream` into the array `dest[]` starting at index `offset`. This method returns the actual number of bytes read or –1, indicating that the end of the stream was reached. To read `size` bytes and throw them away, call this method with `dest[]` set to null.

`public synchronized void reset() throws IOException`

Resets the read point of this `InputStream` to the location of the last mark set.

`public long skip(long offset) throws IOException`

Skips over `offset` bytes from this `InputStream`. Returns the actual number of bytes skipped, as it is possible to skip over less than `offset` bytes.

InputStreamReader (`java.io`)

A public class, derived from `Reader`, that is an input stream of characters.

Constructors

`public InputStreamReader(InputStream input)`

`public InputStreamReader(InputStream input, String encoding) throws UnsupportedEncodingException`

Creates an instance of `InputStreamReader` from the `InputStream input` with a specified `encoding`.

Methods

`public void close() throws IOException`

Closes this `InputStreamReader`.

`public String getEncoding()`
Returns the string representation of this `InputStreamReader`'s encoding.

`public int read() throws IOException`
Reads a single character from this `InputStreamReader`. The character read is returned as an `int`, or a –1 is returned if the end of this `InputStreamReader` was encountered.

`public int read(char[] dest, int offset, int size) throws IOException`
Reads no more than `size` bytes from this `InputStreamReader` into the array `dest[]` starting at index `offset`. This method returns the actual number of bytes read or –1, indicating that the end of the stream was reached. To read `size` bytes and throw them away, call this method with `dest[]` set to null.

`public boolean ready() throws IOException`
Returns a true value if this `InputStreamReader` is capable of being read from. This state can only be true if the buffer is not empty.

Insets (`java.awt`)

A public class, derived from `Object` and implementing `Serializable` and `Cloneable`, that specify the margins of a container.

Variables and Constants

`public int bottom`

`public int left`

`public int right`

`public int top`
Contains the value of the inset for a particular margin.

Constructors

`public Insets(int t, int l, int b, int r)`
Creates an instance of insets with initial top (`t`), bottom (`b`), left (`l`), and right (`r`) inset values.

Methods

`public Object clone()`
Creates a copy of this group of inset values.

`public boolean equals(Object arg)`
Returns a true value if this inset is equal to the object `arg`.

```
public String toString()
```
Returns a string representation of this group of inset values.

Integer (java.lang)

A public final class, derived from `Number`, that contains integer math operations, constants, methods to compute minimum and maximum numbers, and string manipulation routines related to the primitive `int` type.

Variables and Constants

```
public final static int MAX_VALUE
public final static int MIN_VALUE
```
Constant values that contain the maximum possible value (2147483647) or minimum possible value (22174783648) of an integer in Java.

```
public final static Class TYPE
```
The `Integer` constant value of the integer type class.

Constructors

```
public Integer(int num)
public Integer(String num) throws NumberFormatException
```
Creates an instance of the `Integer` class from the parameter `num`.

Methods

```
public byte byteValue()
public double doubleValue()
public float floatValue()
public int intValue()
public long longValue()
public short shortValue()
```
Returns the value of this integer as a Java primitive type.

```
public static Integer decode(String str) throws
NumberFormatException
```
Decodes the given string (`str`) and returns it as an `Integer`. The decode method can handle octal, hexadecimal, and decimal input values.

```
public boolean equals(Object num)
```
Returns the result of an equality comparison against `num`.

```
public static Integer getInteger(String str)
public static Integer getInteger(String str, int num)
```

```
public static Integer getInteger(String str, Integer num)
```
Returns an `Integer` representation of the system property named in `str`. If there is no property corresponding to `num`, or the format of its value is incorrect, then the default `num` is returned as an `Integer` object.

```
public int hashCode()
```
Returns a hash code for this object.

```
public static int parseInt(String str) throws
NumberFormatException
public static int parseInt(String str, int base) throws
NumberFormatException
```
Evaluates the string `str` and returns the `int` equivalent in radix `base`.

```
public static String toBinaryString(int num)
public static String toHexString(int num)
public static String toOctalString(int num)
```
Returns the string representation of parameter `num` in base 2 (binary), 8 (octal), or 16 (hexadecimal).

```
public String toString()
public static String toString(int num)
public static String toString(int num, int base)
```
Returns the string representation of this integer or `num`. The radix of `num` can be specified in `base`.

```
public static Integer valueOf(String str) throws
NumberFormatException
public static Integer valueOf(String str, int base) throws
NumberFormatException
```
Returns an `Integer` initialized to the value of `str` in radix `base`.

ItemEvent (`java.awt.event`)

A public class, derived from `AWTEvent`, that represents an AWT item event (from a component such as a `Checkbox`, `CheckboxMenuItem`, `Choice`, or `List`).

Variables and Constants

```
public static final int DESELECTED
public static final int SELECTED
```
Constant values representing the deselection or selection of an AWT item component.

```
public static final int ITEM_FIRST
```

`public static final int ITEM_LAST`

Constant values that represent the index of the first and last item event ids.

`public static final int ITEM_STATE_CHANGED`

A constant value that represents the event of the change of state for an AWT item.

Constructors

`public ItemEvent(ItemSelectable src, int type, Object obj, int change)`

Creates a new instance of an `ItemEvent` from the specified source, having a specific `type`, item `object`, and state `change`.

Methods

`public Object getItem()`

Returns the specific item that triggered this event.

`public ItemSelectable getItemSelectable()`

Returns the `ItemSelectable` object that triggered this event.

`public int getStateChange()`

Returns the state change type (deselection or selection) that triggered this event.

`public String paramString()`

Returns a parameter string containing the values of the parameters for this event.

JApplet (`javax.swing`)

A public class, derived from `Applet` and implementing `Accessible` and `RootPaneContainer`, that represents a primary applet container.

Constructors

`public JApplet()`

Creates an applet container.

Methods

`public Container getContentPane()`

`public Component getGlassPane()`

`public JLayeredPane getLayeredPane()`

```
public JRootPane getRootPane()
```
Returns the content pane, glass pane, layered pane, or root pane for this applet.

```
public void setContentPane(Container contenetPane)
public void setGlassPane(Component glassPane)
public void setLayeredPane(JLayeredPane layeredPane)
public void setRootPane(JRootPane rootPane)
```
Sets the content pane, glass pane, layered pane, or root pane for this applet.

```
public void remove(Component comp)
```
Removes the specified component from this applet.

```
public JMenuBar getJMenuBar()
public void setJMenuBar setJMenuBar(JMenuBar menuBar)
```
Gets or sets the menu bar for this applet.

JButton (`javax.swing`)

A public class, derived from `AbstractButton` and implementing `Accessible`, that represents a GUI push button.

Constructors

```
public JButton()
public JButton(Icon icon)
public JButton(String text)
public JButton(String text, Icon icon)
```
Creates a button with the specified text and icon.

Methods

```
public boolean isDefaultButton()
```
Returns true if this button is the current default button for its root pane.

```
public boolean isDefaultCapable()
public void setDefaultCapable(boolean capable)
```
Gets or sets the property that determines if this button can be the default button for its root pane.

JCheckBox (javax.swing)

A public class, derived from `JToggleButton` and implementing `Accessible`, that represents a GUI component that can be selected or deselected (displaying its state to the user).

Constructors

```
public JCheckBox()
public JCheckBox(Icon icon)
public JCheckBox(Icon icon, boolean selected)
public JCheckBox(String text)
public JCheckBox(String text, boolean selected)
public JCheckBox(String text, Icon icon)
public JCheckBox(String text, Icon icon, boolean selected)
```

Creates a check box with the specified text, icon, and selected state (which defaults to unselected).

JCheckBoxMenuItem (javax.swing)

A public class, derived from `JMenuItem` and implementing `Accessible` and `SwingConstants`, that represents a menu item that can be selected or deselected.

Constructors

```
public JCheckBoxMenuItem()
public JCheckBoxMenuItem(Icon icon)
public JCheckBoxMenuItem(String text)
public JCheckBoxMenuItem(String text, boolean selected)
public JCheckBoxMenuItem(String text, Icon icon)
public JCheckBoxMenuItem(String text, Icon icon, boolean
selected)
```

Creates a menu check box with the specified text, icon, and selected state (which defaults to unselected).

JColorChooser (javax.swing)

A public class, derived from `JComponent` and implementing `Accessible`, that represents a pane of controls that allows a user to define and select a color. A color chooser can be displayed as a dialog box or within any container.

Constructors

```
public JColorChooser()
public JColorChooser(Color initialColor)
```

Creates a color chooser with the specified initial color (white by default).

Methods

```
public Color getColor()
public void setColor(Color color)
public void setColor(int color)
public void setColor(int red, int green, int blue)
```

Gets or sets the current color for this color chooser.

```
public static Color showDialog(Component parent, String title,
Color initialColor)
```

Shows a color chooser dialog box, returning the selected color when the user presses the OK button.

JComboBox (javax.swing)

A public class, derived from `JComponent` and implementing `ItemSelectable`, `ListDataListener`, `ActionListener`, and `Accessible`, that represents a GUI component that combines a button (or editable field) and a drop down list.

Constructors

```
public JComboBox()
public JComboBox(Object[] items)
public JComboBox(Vector items)
```

Creates a combo box containing the specified items.

Methods

```
public addActionListener(ActionListener listener)
public addItemListener(ItemListener listener)
```
Adds a specific type of listener to this combo box.

```
public void addItem(Object item)
public insertItemAt(Object item, int index)
```
Adds the specified item to the end of the item list or inserts it at the specified index.

```
public Object getItemAt(int index)
```
Returns the item at the specified index.

```
public int getItemCount()
```
Returns the number of items in the list.

```
public Object getSelectedItem()
```
Returns the currently selected item.

```
public void setEditable(boolean flag)
```
Sets whether this combo box is editable.

```
public boolean isEditable()
```
Returns true if this combo box is editable.

```
public void setEnabled(boolean flag)
```
Enables or disables this combo box. When disabled, items cannot be selected.

```
public void removeAllItems()
public void removeItem(Object item)
public void removeItemAt(int index)
```
Removes all items, a specific item, or the item at a specific index, from the list.

JComponent (`javax.swing`)

A public abstract class, derived from `Component` and implementing `Serializable`, that represents the base class for all Swing components (except top-level containers).

Methods

```
public float getAlignmentX()
public void setAlignmentX(float alignment)
public float getAlignmentY()
public void setAlignmentY(float alignment)
```
Gets or sets the horizontal or vertical alignment for this component.

```
public Border getBorder()
public void setBorder(Border border)
```
Gets or sets the border for this component.
```
public Graphics getGraphics()
```
Returns the graphics context for this component.
```
public int getHeight()
public int getWidth()
```
Returns the height or width of this component.
```
public Dimension getMaximumSize()
public void setMaximumSize(Dimension size)
public Dimension getMinimumSize()
public void setMinimumSize(Dimension size)
public Dimension getPreferredSize()
public void setPreferredSize(Dimension size)
```
Gets or sets the maximum, minimum, or preferred size for this component.
```
public JRootPane getRootPane()
```
Returns the root pane ancestor for this component.
```
public String getToolTipText()
public void setToolTipText(String text)
```
Gets or sets the text for this component's tool tip.
```
public int getX()
public int getY()
```
Returns the x or y coordinate of this component.
```
public void setEnabled(boolean enabled)
```
Enables or disables this component.
```
public void setFont(Font font)
```
Sets the font for this component.
```
public void setBackground(Color color)
public void setForeground(Color color)
```
Sets the background or foreground color for this component.
```
public setVisible(boolean flag)
```
Makes this component visible or invisible.

JFileChooser (`javax.swing`)

A public class, derived from `JComponent` and implementing `Accessible`, that represents a GUI component that allows the user to select a file from a file system.

Variables and Constants

```
public static final int APPROVE_OPTION
```
Return value if approval (Yes, Ok) is chosen.

```
public static final int CANCEL_OPTION
```
Return value if Cancel is chosen.

```
public static final int ERROR_OPTION
```
Return value if an error occurred.

Constructors

```
public JFileChooser()
public JFileChooser(File directory)
public JFileChooser(FileSystemView view)
public JFileChooser(String path)
public JFileChooser(File directory, FileSystemView view)
public JFileChooser(String path, FileSystemView view)
```
Creates a file chooser with the specified directory or path and optional file system view.

Methods

```
public File getCurrentDirectory()
public void setCurrentDirectory(File directory)
```
Gets or sets the current directory for this file chooser.

```
public String getDescription(File file)
public String getName(File file)
```
Returns the description or name of the specified file.

```
public boolen getDraggedEnabled()
public void setDraggedEnabled(boolean flag)
```
Gets or sets the property that determines whether the user can drag to select files.

```
public File getSelectedFile()
public File[] getSelectedFiles()
```
Gets the currently selected file or files.

```
public boolean isMultiSelectionEnabled()
```
Returns true if multiple files can be selected.

```
public void setDialogTitle(String title)
```
Sets the title of the dialog box.

```
public void setFileFilter(FileFilter filter)
```
Sets the current file filter.

```
public void setSelectedFile(File file)
public void setSelectedFiles(File[] files)
```
Sets the selected file or files.

```
public int showDialog(Component parent, String
approveButtonText)
```
Displays a custom file chooser dialog with the specified approve button text.

```
public int showOpenDialog(Component parent)
```
Displays an "open file" file chooser dialog.

```
public int showSaveDialog(Component parent)
```
Displays a "save file" file chooser dialog.

JFrame (`javax.swing`)

A public class, derived from `Frame` and implementing `WindowConstants`, `Accessible`, and `RootPaneContainer`, that represents a primary GUI window.

Variables and Constants

```
public static final int EXIT_ON_CLOSE
```
Represents the exit application default window close operation.

Constructors

```
public JFrame()
public JFrame(String title)
```
Creates a frame with the specified title.

Methods

```
public Container getContentPane()
public Component getGlassPane()
public JLayeredPane getLayeredPane()
public JRootPane getRootPane()
```
Returns the content pane, glass pane, layered pane, or root pane for this frame.

```
public void setContentPane(Container contenetPane)
public void setGlassPane(Component glassPane)
public void setLayeredPane(JLayeredPane layeredPane)
public void setRootPane(JRootPane rootPane)
```
Sets the content pane, glass pane, layered pane, or root pane for this frame.

```
public void remove(Component comp)
```
Removes the specified component from this frame.

```
public JMenuBar getJMenuBar()
public void setJMenuBar setJMenuBar(JMenuBar menuBar)
```
Gets or sets the menu bar for this frame.
```
public void setDefaultCloseOperation(int operation)
```
Sets the default operation when the user closes this frame.

JLabel (javax.swing)

A public class, derived from `JComponent` and implementing `Accessible` and `SwingConstants`, that represents a GUI display area for a string, and image, or both.

Constructors

```
public JLabel()
public JLabel(String text)
public JLabel(Icon icon)
public JLabel(String text, int horizontalAlignment)
public JLabel(Icon icon, int horizontalAlignment)
public JLabel(String text, Icon icon, int horizontalAlignment)
```
Creates a label containing the specified icon and string, and using the specified horizontal alignment.

Methods

```
public int getHorizontalAlignment()
public void setHorizontalAlignment(int alignment)
public int getVerticalAlignment()
public void setVerticalAlignment(int alignment)
```
Gets or sets the horizontal or vertical alignment of the icon and text.
```
public int getHorizontalTextPosition()
public void setHorizontalTextPosition(int position)
public int getVerticalTextPosition()
public void setVerticalTextPosition(int position)
```
Gets or sets the horizontal or vertical position of the text relative to the icon.
```
public Icon getIcon()
public void setIcon(Icon icon)
```
Gets or sets the default icon for this button.
```
public String getText()
```

```
public void setText(String text)
```
Gets or sets the text displayed on this button.

```
public Component getLabelFor()
public void setLabelFor(Component comp)
```
Gets or sets the component that this label describes.

JList (javax.swing)

A public class, derived from `JComponent` and implementing `Accessible` and `Scrollable`, that represents a GUI component that allows the user to select one or more objects from a list.

Variables and Constants

```
public static final int HORIZONTAL_WRAP
```
Indicates that cells flow horizontally, then vertically.

```
public static final int VERTICAL
```
Indicates one column of cells (the default).

```
public static final int VERTICAL_WRAP
```
Indicates that cells flow vertically, then horizontally.

Constructors

```
public JList()
public JList(Object[] items)
public JList(Vector items)
```
Creates a list that displays the specified items.

Methods

```
public void addListSelectionListener(ListSelectionListener
listener)
```
Adds the specified listener to this list.

```
public void clearSelection()
```
Clears the selection (no items will be selected).

```
public void ensureIndexIsVisible(int index)
```
Scrolls the list to make the specified item visible.

```
public int getLastVisibleIndex()
```
Returns the index of the last visible cell.

```
public int getLayoutOrientation()
```

```
public void setLayoutOrientation(int orientation)
```
Gets or sets the layout orientation for this list.

```
public int getMaxSelectionIndex()
public int getMinSelectionIndex()
```
Returns the largest or smallest selected cell index.

```
public int getSelectedIndex()
public void setSelectedIndex(int index)
public int[] getSelectedIndices()
public void setSelectedIndex(int[] indices)
```
Gets or sets the selected index or indices.

```
public void setSelectionInterval(int from, int to)
```
Selects the specified index interval.

```
public Object getSelectedValue()
public Object[] getSelectedValues()
```
Returns the currently selected value or values.

```
public Color getSelectionBackground()
public void setSelectionBackground(Color color)
public Color getSelectionForeground()
public void setSelectionForeground(Color color)
```
Gets or sets the background or foreground color of the selection.

```
public boolean isSelectedIndex(int index)
```
Returns true if the specified index is selected.

```
public boolean isSelectionEmpty()
```
Returns true if no item is currently selected.

```
public void setDragEnabled(boolean flag)
```
Enables or disables the property allowing the user to select multiple items by dragging the mouse.

```
public void setListData(Object[] items)
public void setListData(Vector items)
```
Sets the contents of the list to the specified items.

```
public void setSelectionMode(int selectionMode)
```
Sets the selection mode for this list using `ListSelectionModel` constants.

JOptionPane (`javax.swing`)

A public class, derived from `JComponent` and implementing `Accessible`, that provides methods for creating standard dialog boxes.

Variables and Constants

```
public static final int CANCEL_OPTION
public static final int OK_OPTION
public static final int YES_OPTION
```
Return value if a specific button option is chosen.
```
public static final int CLOSED_OPTION
```
Return value if the user closes the window without selecting anything.
```
public static final int DEFAULT_OPTION
public static final int YES_NO_OPTION
public static final int YES_NO_CANCEL_OPTION
public static final int OK_CANCEL_OPTION
```
Specifies the types of buttons to use in the dialog.
```
public static final int ERROR_MESSAGE
public static final int INFORMATION_MESSAGE
public static final int WARNING_MESSAGE
public static final int QUESTION_MESSAGE
public static final int PLAIN_MESSAGE
```
Specifies a message style.

Methods

```
public static int showConfirmDialog(Component parent, Object
message)
public static int showConfirmDialog(Component parent, Object
message, String title, int buttonSet)
public static int showConfirmDialog(Component parent, Object
message, String title, int buttonSet, int messageStyle)
public static int showConfirmDialog(Component parent, Object
message, String title, int buttonSet, int messageStyle, Icon
icon)
```
Displays a dialog box allowing the user to confirm an option. Uses the specified message, title, button set, message style, and icon.
```
public static int showInputDialog(Component parent, Object
message)
public static int showInputDialog(Component parent, Object
message, Object initialSelectionValue)
public static int showInputDialog(Component parent, Object
message, String title, int messageStyle)
public static int showInputDialog(Object message)
```

```
public static int showInputDialog(Object message, Object
initialSelectionValue)
public static int showInputDialog(Component parent, Object
message, String title, int messageStyle, Icon icon, Object[]
selectionValues, Object initialSelectionValue)
```

Displays a dialog box allowing the user to enter input. Uses the specified message, title, and message style. An initial selection and options can also be specified.

```
public static void showMessageDialog(Component parent, Object
message)
public static void showMessageDialog(Component parent, Object
message, String title, int messageStyle)
public static void showMessageDialog(Component parent, Object
message, String title, int buttonSet, int messageStyle, Icon
icon)
```

Displays a dialog box presenting a message. Uses the specified message, title, message style, and icon.

```
public static int showOptionDialog(Component parent, Object
message, String title, int buttonSet, int messageStyle, Icon
icon, Object[] options, Object initialValue)
```

Displays a dialog box allowing the user to make a general choice. Uses the specified message, title, button set, message style, and icon. An initial selection and options can also be specified.

JPanel (`javax.swing`)

A public class, derived from `JComponent` and implementing `Accessible`, that represents a lightweight GUI container used to organize other components.

Constructors

```
public JPanel()
public JPanel(LayoutManager manager)
```

Creates a panel with the specified layout manager, which defaults to a flow layout.

JPasswordField (`javax.swing`)

A public class, derived from `JTextField`, that represents a GUI text field into which the user can type a password. The password itself is not displayed as it is typed, but a visual indication that characters are being typed is shown.

Constructors

```
public JPasswordField()
public JPasswordField(int columns)
public JPasswordField(String text)
public JPasswordField(String text, int columns)
```

Creates a password field with the specified number of columns, initialized to the specified text.

Methods

```
public char[] getPassword()
```

Returns the text contained in this password field.

```
public char getEchoChar()
public void setEchoChar(char ch)
```

Gets or sets the character that is displayed as the user types into this field.

JRadioButton (`javax.swing`)

A public class, derived from `JToggleButton` and implementing `Accessible`, that represents a radio button, used as part of a button group (`ButtonGroup`), to present a set of mutually exclusive options.

Constructors

```
public JRadioButton()
public JRadioButton(String text)
public JRadioButton(Icon icon)
public JRadioButton(String text, boolean selected)
public JRadioButton(Icon icon, boolean selected)
public JRadioButton(String text, Icon icon)
public JRadioButton(String text, Icon icon, boolean selected)
```

Creates a radio button with the specified text, icon, and initial selection status (unselected by default).

JScrollPane (`javax.swing`)

A public class, derived from `JComponent` and implementing `Accessible` and `ScrollPaneConstants`, that represents a lightweight GUI container with a scrollable view.

Constructors

```
public JScrollPane()
public JScrollPane(Component comp)
public JScrollPane(int verticalPolicy, int horizontalPolicy)
public JScrollPane(Component comp, int verticalPolicy, int
horizontalPolicy)
```

Creates a scroll pane displaying the specified component and using the specified horizontal and vertical scroll bar policies.

Methods

```
public int getHorizontalScrollBarPolicy()
public void setHorizontalScrollBarPolicy(int policy)
public int getHorizontalScrollBarPolicy()
public void setHorizontalScrollBarPolicy(int policy)
```

Gets or sets the horizontal or vertical scroll bar policy for this scroll pane.

JSlider (`javax.swing`)

A public class, derived from `JComponent`, that represents a GUI component that allows the user to select a numeric value by sliding a knob within a bounded interval.

Constructors

```
public JSlider()
public JSlider(int orientation)
public JSlider(int min, int max)
public JSlider(int min, int max, int initialValue)
public JSlider(int orientation, int min, int max, int
initialValue)
```

Creates a new slider with the specified orientation, minimum value, maximum value, and initial value. The default orientation is horizontal, the default minimum value is 0, the default maximum value is 100, and the default initial value is the range midpoint.

Methods

```
public void addChangeListener(ChangeListener listener)
```

Adds a `ChangeListener` to this slider.

```
public int getExtent()
```
Returns the range of values covered by the knob.
```
public int getMajorTickSpacing()
public int getMinorTickSpacing()
```
Returns the major or minor tick spacing of this slider.
```
public int getMinimum()
public int getMaximum()
```
Returns the minimum or maximum value of this slider.
```
public int getOrientation()
```
Returns this slider's orientation.
```
public boolean getPaintLabels()
public boolean getPaintTicks()
public boolean getPaintTrack()
```
Returns true if this slider's labels, tick marks, or track are to be painted.
```
public boolean getSnapToTicks()
```
Returns true if this slider's knob snaps to the closest tick mark when the user moves the knob.
```
public int getValue()
```
Returns this slider's values.
```
public boolean getValueIsAdjusting()
```
Returns true if the slider knob is being dragged.
```
public void setExtent(int extent)
```
Sets the size of the range covered by this slider's knob.
```
public void setMajorTickSpacing(int value)
public void setMinorTickSpacing(int value)
```
Sets the major or minor tick spacing for this slider.
```
public void setMinimum(int minimumValue)
public void setMaximum(int maximumValue)
```
Sets the minimum or maximum value for this slider.
```
public void setOrientation(int orientation)
```
Sets the orientation for this slider.
```
public void setPaintLabels(boolean flag)
public void setPaintTicks(boolean flag)
public void setPaintTrack(boolean flag)
```
Determines whether this slider's labels, tick marks, or track are to be painted.

```
public void setSnapToTicks(boolean flag)
```
Determines whether the knob (and value) snaps to the closest tick mark when the user moves the knob.

```
public void setValue(int value)
```
Sets this slider's current value.

JTabbedPane (javax.swing)

A public class, derived from `JComponent` and implementing `Accessible`, `Serializable`, and `SwingConstants`, that represents a GUI container that allows the user to switch between a group of components by clicking on a tab.

Variables and Constants

```
public static final int SCROLL_TAB_LAYOUT
```
Specifies a tab layout that provides a scrollable region of tabs when all tabs won't fit in a single run.

```
public static final int WRAP_TAB_LAYOUT
```
Specifies a tab layout that wraps tabs in multiple rows when all tabs won't fit in a single run.

Constructors

```
public JTabbedPane()
public JTabbedPane(int tabPlacement)
public JTabbedPane(int tabPlacement, int tabLayoutPolicy)
```
Creates a tabbed pane with the specified tab placement and tab layout policy. The tab placement is specified using `SwingConstants`.

Methods

```
public Component add(String title, Component comp)
```
Adds the specified component to a tab with the specified title.

```
public int getTabCount()
```
Returns the number of tabs in this tabbed pane.

```
public Color getBackgroundAt(int index)
public void setBackgroundAt(int index, Color color)
public Color getForegroundAt(int index)
public void setForegroundAt(int index, Color color)
```
Gets or sets the background or foreground color of the tab at the specified index.

JTextArea (javax.swing)

A public class, derived from `JTextComponent`, that represents a multi-line area for displaying or editing text.

Constructors

```
public JTextArea()
public JTextArea(int rows, int columns)
public JTextArea(String text)
public JTextArea(String text, int rows, int columns)
```

Creates a text area with the specified initial text and an initial size goverened by the specified number of rows and columns.

Methods

```
public int getColumns()
public void setColumns(int columns)
public int getRows()
public void setRows(int rows)
```

Gets or sets the number of rows or columns for this text area.

```
public int getLineCount()
```

Returns the number of lines cotained in this text area.

```
public boolean getLineWrap()
public void setLineWrap(boolean flag)
```

Gets or sets the property that determines if lines are wrapped in this text area.

```
public boolean getWrapStyleWord()
public void setWrapStyleWord(boolean flag)
```

Gets or sets the property that determines if lines are wrapped by words or characters (if they are wrapped at all).

```
public void append(String str)
```

Appends the specified string to the end of the document in this text area.

```
public void insert(String str, int position)
```

Inserts the specified string into this text area's document at the specified position.

```
public void setFont(Font font)
```

Sets the font for this text area.

JTextField (javax.swing)

A public class, derived from `JTextComponent` and implementing `SwingConstants`, that represents a single line area for displaying or editing text (often used as an input field).

Constructors

```
public JTextField()
public JTextField(int columns)
public JTextField(String text)
public JTextField(String text, int columns)
```

Creates a text field with the specified initial text and an initial size goverened by the specified number of columns.

Methods

```
public void addActionListener(ActionListener listener)
```

Adds an action listener to this text field.

```
public int getColumns()
public void setColumns(int columns)
```

Gets or sets the number of columns for this text field.

```
public int getHorizontalAlignment()
public void setHorizontalAlignment(int alignment)
```

Gets or sets the horizontal alignment for this text field.

```
public void setFont(Font font)
```

Sets the font for this text field.

JToggleButton (javax.swing)

A public class, derived from `AbstractButton` and implementing `Accessible`, that represents a two-state button.

Constructors

```
public JToggleButton()
public JToggleButton(String text)
public JToggleButton(String text, boolean selected)
public JToggleButton(Icon icon)
```

```
public JToggleButton(Icon icon, boolean selected)
public JToggleButton(String text, Icon icon)
public JToggleButton(String text, Icon icon, boolean selected)
```
Creates a toggle button with the specified string, icon, and selection state.

JToolTip (javax.swing)

A public class, derived from `JComponent` and implementing `Accessible`, that represents a text tip that is displayed when the mouse cursor rests momentarily over a GUI component.

Constructors

```
public JToolTip()
```
Creates a tool tip.

Methods

```
public JComponent getComponent()
public void setComponent(JComponent comp)
```
Gets or sets the component to which this tool tip applies.
```
public String getTipText()
public void setTipText(String text)
```
Gets or sets the text shown when this tool tip is displayed.

KeyAdapter (java.awt.event)

A public abstract class, derived from `Object` and implementing `KeyListener`, that permits derived classes to override the predefined no-op keyboard events.

Constructors

```
public KeyAdapter()
```
Creates a new instance of a `KeyAdapter`.

Methods

```
public void keyPressed(KeyEvent event)
public void keyReleased(KeyEvent event)
public void keyTyped(KeyEvent event)
```
Empty methods that should be overridden in order to implement event handling for keyboard events.

KeyEvent (java.awt.event)

A public class, derived from `InputEvent`, that represents an AWT keyboard event.

Variables and Constants

```
public static final int VK_0
public static final int VK_1
public static final int VK_2
public static final int VK_3
public static final int VK_4
public static final int VK_5
public static final int VK_6
public static final int VK_7
public static final int VK_8
public static final int VK_9
```

Constant values that represent the keyboard keys 0–9.

```
public static final int KEY_FIRST
public static final int KEY_LAST
```

Constant values that represent the index of the first and last key event ids.

```
public static final int KEY_PRESSED
public static final int KEY_RELEASED
public static final int KEY_TYPED
```

Constant values that represent the ids of a key being pressed, released, or typed.

```
public static final char CHAR_UNDEFINED
```

A constant value that represents an event of a key press or release that does not correspond to a Unicode character.

```
public static final int VK_LEFT
public static final int VK_RIGHT
public static final int VK_UP
public static final int VK_DOWN
public static final int VK_HOME
public static final int VK_END
public static final int VK_PAGE_UP
public static final int VK_PAGE_DOWN
```

Constant values that represent various keyboard directional keys.

```
public static final int VK_INSERT
public static final int VK_DELETE
```

Constant values that represent various keyboard editing control keys.

```
public static final int VK_NUMPAD0
public static final int VK_NUMPAD1
public static final int VK_NUMPAD2
public static final int VK_NUMPAD3
public static final int VK_NUMPAD4
public static final int VK_NUMPAD5
public static final int VK_NUMPAD6
public static final int VK_NUMPAD7
public static final int VK_NUMPAD8
public static final int VK_NUMPAD9
public static final int VK_ADD
public static final int VK_SUBTRACT
public static final int VK_MULTIPLY
public static final int VK_DIVIDE
public static final int VK_ENTER
public static final int VK_DECIMAL
```

Constant values that represent various keyboard number pad keys.

```
public static final int VK_PERIOD
public static final int VK_EQUALS
public static final int VK_OPEN_BRACKET
public static final int VK_CLOSE_BRACKET
public static final int VK_BACK_SLASH
public static final int VK_SLASH
public static final int VK_COMMA
public static final int VK_SEMICOLON
public static final int VK_SPACE
public static final int VK_BACK_SPACE
public static final int VK_QUOTE
public static final int VK_BACK_QUOTE
public static final int VK_TAB
public static final int VK_SLASH
```

Constant values that represent various keyboard character keys.

```
public static final int VK_PAUSE
public static final int VK_PRINTSCREEN
```

```
public static final int VK_SHIFT
public static final int VK_HELP
public static final int VK_CONTROL
public static final int VK_ALT
public static final int VK_ESCAPE
public static final int VK_META
public static final int VK_ACCEPT
public static final int VK_CANCEL
public static final int VK_CLEAR
public static final int VK_CONVERT
public static final int VK_NONCONVERT
public static final int VK_MODECHANGE
public static final int VK_SEPARATER
public static final int VK_KANA
public static final int VK_KANJI
public static final int VK_FINAL
```

Constant values that represent various keyboard command and control keys.

```
public static final int VK_UNDEFINED
```

A constant value for KEY_TYPED events for which there is no defined key value.

```
public static final int VK_F1
public static final int VK_F2
public static final int VK_F3
public static final int VK_F4
public static final int VK_F5
public static final int VK_F6
public static final int VK_F7
public static final int VK_F8
public static final int VK_F9
public static final int VK_F10
public static final int VK_F11
public static final int VK_F12
```

Constant values that represent the keyboard keys F1–F12.

```
public static final int VK_CAPS_LOCK
public static final int VK_NUM_LOCK
public static final int VK_SCROLL_LOCK
```

Constant values that represent various keyboard control keys.

```
public static final int VK_A
```

```
public static final int VK_B
public static final int VK_C
public static final int VK_D
public static final int VK_E
public static final int VK_F
public static final int VK_G
public static final int VK_H
public static final int VK_I
public static final int VK_J
public static final int VK_K
public static final int VK_L
public static final int VK_M
public static final int VK_N
public static final int VK_O
public static final int VK_P
public static final int VK_Q
public static final int VK_R
public static final int VK_S
public static final int VK_T
public static final int VK_U
public static final int VK_V
public static final int VK_W
public static final int VK_X
public static final int VK_Y
public static final int VK_Z
```

Constant values that represent the keyboard keys A–Z.

Constructors

```
public KeyEvent(Component src, int id, long when, int modifiers,
int keyCode)
```

```
public KeyEvent(Component src, int id, long when, int modifiers,
int keyCode, char keyChar)
```

Creates a new instance of a `KeyEvent` from the specified source, having a specific type (`id`), time stamp, modifiers, key code, and/or key character.

Methods

`public char getKeyChar()`

`public void setKeyChar(char character)`

Returns or sets the character associated with this `KeyEvent`. For events that have no corresponding character, a `CHAR_UNDEFINED` is returned.

`public int getKeyCode()`

`public void setKeyCode(int code)`

Returns or sets the code associated with this `KeyEvent`. For events that have no corresponding code, a `VK_UNDEFINED` is returned.

`public static String getKeyModifiersText(int mods)`

`public static String getKeyText(int keyCode)`

Returns a string representation of the `KeyEvent` modifiers key code (i.e., "Meta+Shift" or "F1").

`public boolean isActionKey()`

Returns a true value if this event is from an action key.

`public String paramString()`

Returns a string representation of the parameters of this event.

`public void setModifiers(int mods)`

Sets the key event modifiers for this event.

Locale (`java.util`)

A public class, derived from `Object` and implementing `Serializable` and `Cloneable`, that represents geographic-specific or political-specific information.

Variables and Constants

```
public static final Locale CANADA
public static final Locale CANADA_FRENCH
public static final Locale CHINA
public static final Locale FRANCE
public static final Locale GERMANY
public static final Locale ITALY
public static final Locale JAPAN
public static final Locale KOREA
public static final Locale PRC
public static final Locale TAIWAN
public static final Locale UK
```

```
public static final Locale US
```
Constant values that represent locales based on countries.
```
public static final Locale CHINESE
public static final Locale ENGLISH
public static final Locale FRENCH
public static final Locale GERMAN
public static final Locale ITALIAN
public static final Locale JAPANESE
public static final Locale KOREAN
public static final Locale SIMPLIFIED_CHINESE
public static final Locale TRADITIONAL_CHINESE
```
Constant values that represent locales based on languages.

Constructors

```
public Locale(String lang, String country)
public Locale(String lang, String country, String var)
```
Creates a new locale from the specified two-character ISO codes for a language and country. A computer and browser variant of a locale can also be included. These usually take the form of WIN for Windows or MAC for Macintosh.

Methods

```
public Object clone()
```
Returns a copy of this locale.
```
public boolean equals(Object arg)
```
Returns a true value if this locale is equal to `arg`.
```
public String getCountry()
public String getLanguage()
public String getVariant()
```
Returns the character code for the name of this locale's country, language, or variant.
```
public static synchronized Locale getDefault()
public static synchronized void setDefault(Locale locale)
```
Returns or sets the default locale.
```
public final String getDisplayCountry()
public String getDisplayCountry(Locale displaylocale)
```
Returns the display version of the country name for this locale in either the specified or default locales.
```
public final String getDisplayLanguage()
```

`public String getDisplayLanguage(Locale displaylocale)`
Returns the display version of the language name for this locale in either the specified or default locales.

`public final String getDisplayName()`

`public String getDisplayName(Locale displaylocale)`
Returns the display version of the name for this locale in either the specified or default locales.

`public final String getDisplayVariant()`

`public String getDisplayVariant(Locale displaylocale)`
Returns the display version of the variant for this locale in either the specified or default locales.

`public String getISO3Country() throws MissingResourceException`

`public String getISO3Language() throws MissingResourceException`
Returns the three-character ISO abbreviation for the country or language for this locale.

`hashCode()`
Returns the hash code for this locale.

`toString()`
Returns a string representation of this locale.

Long (`java.lang`)

A public final class, derived from `Number`, that contains long integer math operations, constants, methods to compute minimum and maximum numbers, and string manipulation routines related to the primitive `long` type.

Variables and Constants

`public final static long MAX_VALUE`

`public final static long MIN_VALUE`
Constant values that contain the maximum possible value (9223372036854775807L) or minimum possible value (-9223372036854775808L) of a `long` in Java.

`public final static Class TYPE`
The `Integer` constant value of the integer type class.

Constructors

```
public Long(long num)
public Long(String num) throws NumberFormatException
```

Creates an instance of the `Long` class from the parameter `num`.

Methods

```
public byte byteValue()
public double doubleValue()
public float floatValue()
public int intValue()
public long longValue()
public short shortValue()
```

Returns the value of this `Long` as a Java primitive type.

```
public boolean equals(Object arg)
```

Returns the result of the equality comparison between this `Long` and the parameter `arg`.

```
public static Long getLong(String prop)
public static Long getLong(String prop, long num)
public static Long getLong(String prop, long num)
```

Returns a `Long` representation of the system property named in `prop`. If there is no property corresponding to `prop`, or the format of its value is incorrect, then the default `num` is returned.

```
public int hashCode()
```

Returns a hash code for this `Long`.

```
public static long parseLong(String str) throws
NumberFormatException
public static long parseLong(String str, int base) throws
NumberFormatException
```

Evaluates the string `str` and returns the long equivalent in radix `base`.

```
public static String toBinaryString(long num)
public static String toHexString(long num)
public static String toOctalString(long num)
```

Returns the string representation of parameter `num` in base 2 (binary), 8 (octal), or 16 (hexadecimal).

```
public String toString()
public static String toString(long num)
```

```
public static String toString(long num, int base)
```
Returns the string representation of this `long` or `num` in base 10 (decimal). The radix of the returned number can also be specified in `base`.

```
public static Long valueOf(String str) throws
NumberFormatException
public static Long valueOf(String str, int base) throws
NumberFormatException
```
Returns a `Long` initialized to the value of `str` in radix `base`.

Math (`java.lang`)

A public final class, derived from `Object`, that contains integer and floating point constants, and methods to perform various math operations, compute minimum and maximum numbers, and generate random numbers.

Variables and Constants

```
public final static double E
public final static double PI
```
Constant values that contain the natural base of logarithms (2.718281828459045235 4) and the ratio of the circumference of a circle to its diameter (3.14159265358979323846).

Methods

```
public static double abs(double num)
public static float abs(float num)
public static int abs(int num)
public static long abs(long num)
```
Returns the absolute value of the specified parameter.

```
public static double acos(double num)
public static double asin(double num)
public static double atan(double num)
```
Returns the arc cosine, arc sine, or arc tangent of parameter `num` as a double.

```
public static double atan2(double x, double y)
```
Returns the component *e* of the polar coordinate {r,*e*} that corresponds to the cartesian coordinate <x, y>.

```
public static double ceil(double num)
```
Returns the smallest integer value that is not less than the argument `num`.

```
public static double cos(double angle)
public static double sin(double angle)
public static double tan(double angle)
```
Returns the cosine, sine, or tangent of parameter `angle` measured in radians.

```
public static double exp(double num)
```
Returns *e* to the `num`, where *e* is the base of natural logarithms.

```
public static double floor(double num)
```
Returns a double that is the largest integer value that is not greater than the parameter `num`.

```
public static double IEEEremainder(double arg1, double arg2)
```
Returns the mathematical remainder between `arg1` and `arg2` as defined by IEEE 754.

```
public static double log(double num) throws ArithmeticException
```
Returns the natural logarithm of parameter `num`.

```
public static double max(double num1, double num2)
public static float max(float num1, float num2)
public static int max(int num1, int num2)
public static long max(long num1, long num2)
```
Returns the larger of parameters `num1` and `num2`.

```
public static double min(double num1, double num2)
public static float min(float num1, float num2)
public static int min(int num1, int num2)
public static long min(long num1, long num2)
```
Returns the minimum value of parameters `num1` and `num2`.

```
public static double pow(double num1, double num2) throws
ArithmeticException
```
Returns the result of `num1` to `num2`.

```
public static double random()
```
Returns a random number between 0.0 and 1.0.

```
public static double rint(double num)
```
Returns the closest integer to parameter `num`.

```
public static long round(double num)
public static int round(float num)
```
Returns the closest `long` or `int` to parameter `num`.

```
public static double sqrt(double num) throws ArithmeticException
```
Returns the square root of parameter `num`.

MessageFormat (java.text)

A public class, derived from Format, that is used to build formatted message strings.

Constructors

```
public MessageFormat(String str)
```
Creates a new instance of a MessageFormat from the specified string pattern.

Methods

```
public void applyPattern(String str)
public String toPattern()
```
Sets and returns the pattern for this MessageFormat.

```
public Object clone()
```
Returns a copy of this MessageFormat.

```
public boolean equals(Object arg)
```
Returns a true value if this MessageFormat is equal to arg.

```
public final StringBuffer format(Object src, StringBuffer dest,
FieldPosition ignore)
public final StringBuffer format(Object[] src, StringBuffer
dest, FieldPosition ignore)
```
Formats the specified source object with this MessageFormat, placing the result in dest. This method returns the value of the destination buffer.

```
public static String format(String str, Object[] args)
```
Formats the given string applying specified arguments. This method allows for message formatting with the creation of a MessageFormat.

```
public Format[] getFormats()
public void setFormats(Format[] newFormats)
```
Returns and sets the formats for this MessageFormat.

```
public Locale getLocale()
public void setLocale(Locale locale)
```
Returns and sets the locale for this MessageFormat.

```
public int hashCode()
```
Returns the hash code for this MessageFormat.

```
public Object[] parse(String src) throws ParseException
public Object[] parse(String src, ParsePosition pos)
```
Parses the string source (starting at position pos, or 0 by default), returning its objects.

```
public Object parseObject(String src, ParsePosition pos)
```
Parses the string source (starting at position `pos`, or 0 by default), returning one object.

```
public void setFormat(int var, Format fmt)
```
Sets an individual format at index `var`.

MouseAdapter (`java.awt.event`)

A public abstract class, derived from `Object` and implementing `MouseListener`, that permits derived classes to override the predefined no-op mouse events.

Constructors

```
public MouseAdapter()
```
Creates a new instance of a `MouseAdapter`.

Methods

```
public void mouseClicked(MouseEvent event)
public void mouseEntered(MouseEvent event)
public void mouseExited(MouseEvent event)
public void mousePressed(MouseEvent event)
public void mouseReleased(MouseEvent event)
```
Empty methods which should be overridden in order to implement event handling for mouse events.

MouseEvent (`java.awt.event`)

A public class, derived from `InputEvent`, that represents events triggered by the mouse.

Variables and Constants

```
public static final int MOUSE_CLICKED
public static final int MOUSE_DRAGGED
public static final int MOUSE_ENTERED
public static final int MOUSE_EXITED
public static final int MOUSE_MOVED
```

```
public static final int MOUSE_PRESSED
public static final int MOUSE_RELEASED
```
Constant values that represent a variety of mouse events.
```
public static final int MOUSE_FIRST
public static final int MOUSE_LAST
```
Constant values that represent the index of the first and last mouse event ids.

Constructors

```
public MouseEvent(Component src, int type, long timestamp, int
mods, int x, int y, int clickCount, boolean popupTrigger)
```
Creates a new instance of a `MouseEvent` from a given source, with a specified type, time stamp, keyboard modifiers, x and y locations, number of clicks, and a state value, if this event triggers a popup menu.

Methods

```
public int getClickCount()
```
Returns the number of mouse clicks in this event.
```
public Point getPoint()
```
Returns the point location of this event, relative to the source component's space.
```
public int getX()
public int getY()
```
Returns the *x* or *y* location of this event, relative to the source component's space.
```
public boolean isPopupTrigger()
```
Returns a true value if this event is a trigger for popup-menus.
```
public String paramString()
```
Returns a string representation of the parameters of this `MouseEvent`.
```
public synchronized void translatePoint(int xoffset, int
yoffset)
```
Offsets the *x* and *y* locations of this event by the specified amounts.

MouseMotionAdapter (`java.awt.event`)

A public abstract class, derived from `Object` and implementing `MouseMotionListener`, that permits a derived class to override the predefined no-op mouse motion events.

Constructors

```
public MouseMotionAdapter()
```

Creates a new instance of a `MouseMotionAdapter`.

Methods

```
public void mouseDragged(MouseEvent event)
public void mouseMoved(MouseEvent event)
```

Empty methods that should be overridden in order to implement event handling for mouse motion events.

Number (`java.lang`)

A public abstract class, derived from `Object` and implementing `Serializable`, that is the parent class to the wrapper classes `Byte`, `Double`, `Integer`, `Float`, `Long`, and `Short`.

Constructors

```
public Number()
```

Creates a new instance of a `Number`.

Methods

```
public byte byteValue()
public abstract double doubleValue()
public abstract float floatValue()
public abstract int intValue()
public abstract long longValue()
public short shortValue()
```

Returns the value of this `Number` as a Java primitive type.

NumberFormat (`java.text`)

A public abstract class, derived from `Format` and implementing `Cloneable`, that is used to convert number objects to locale-specific strings, and vice versa.

Variables and Constants

```
public static final int FRACTION_FIELD
public static final int INTEGER_FIELD
```
Constant values that indicate field locations in a `NumberFormat`.

Constructors

```
public NumberFormat()
```
Creates a new instance of a `NumberFormat`.

Methods

```
public Object clone()
```
Returns a copy of this `NumberFormat`.

```
public boolean equals(Object arg)
```
Returns a true value if this `NumberFormat` is equal to `arg`.

```
public final String format(double num)
public final String format(long num)
```
Formats the specified Java primitive type according to this `NumberFormat`, returning a string.

```
public abstract StringBuffer format(double num, StringBuffer
dest,FieldPosition pos)
public abstract StringBuffer format(long num, StringBuffer dest,
FieldPosition pos)
public final StringBuffer format(Object num, StringBuffer dest,
FieldPosition pos)
```
Formats the specified Java primitive type (or object) starting at `pos`, according to this `NumberFormat`, placing the resulting string in the specified destination buffer. This method returns the value of the string buffer.

```
public static Locale[] getAvailableLocales()
```
Returns the available locales.

```
public static final NumberFormat getCurrencyInstance()
public static NumberFormat getCurrencyInstance(Locale locale)
```
Returns the `NumberFormat` for currency for the default or specified locale.

```
public static final NumberFormat getInstance()
public static NumberFormat getInstance(Locale locale)
```
Returns the default number format for the default or specified locale.

```
public int getMaximumFractionDigits()
```

`public void setMaximumFractionDigits(int val)`
Returns or sets the maximum number of fractional digits allowed in this `NumberFormat`.

`public int getMaximumIntegerDigits()`
`public void setMaximumIntegerDigits(int val)`
Returns or sets the maximum number of integer digits allowed in this `NumberFormat`.

`public int getMinimumFractionDigits()`
`public void setMinimumFractionDigits(int val)`
Returns or sets the minimum number of fractional digits allowed in this `NumberFormat`.

`public int getMinimumIntegerDigits()`
`public void setMinimumIntegerDigits(int val)`
Returns or sets the minimum number of integer digits allowed in this `NumberFormat`.

`public static final NumberFormat getNumberInstance()`
`public static NumberFormat getNumberInstance(Locale locale)`
Returns the `NumberFormat` for numbers for the default or specified locale.

`public static final NumberFormat getPercentInstance()`
`public static NumberFormat getPercentInstance(Locale locale)`
Returns the `NumberFormat` for percentages for the default or specified locale.

`public int hashCode()`
Returns the hash code for this `NumberFormat`.

`public boolean isGroupingUsed()`
`public void setGroupingUsed(boolean toggle)`
Returns or sets the toggle flag for the use of the grouping indicator by this `NumberFormat`.

`public boolean isParseIntegerOnly()`
`public void setParseIntegerOnly(boolean toggle)`
Returns or sets the toggle flag for the use of parsing numbers as integers only by this `NumberFormat`.

`public Number parse(String str) throws ParseException`
Parses the specified string as a number.

`public abstract Number parse(String str, ParsePosition pos)`
`public final Object parseObject(String str, ParsePosition pos)`
Parses the specified string as a long (if possible) or double, starting at position `pos`. Returns a number or an object.

Object (java.lang)

A public class that is the root of the hierarchy tree for all classes in Java.

Constructors

`public Object()`
Creates a new instance of the object class.

Methods

`protected Object clone() throws OutOfMemoryError, CloneNotSupportedException`
Returns an exact copy of the current object.

`public boolean equals(Object arg)`
Returns a true value if the current object is equal to `arg`.

`protected void finalize() throws Throwable`
The `finalize` method contains code that is called as the object is being destroyed.

`public final Class getClass()`
Returns the class of the current object.

`public int hashCode()`
Returns a hash code for the current object.

`public final void notify() throws IllegalMonitorStateException`

`public final void notifyAll() throws IllegalMonitorStateException`
Informs a paused thread that it may resume execution. `notifyAll` informs all paused threads.

`public String toString()`
Returns a string representation of the current object.

`public final void wait() throws IllegalMonitorStateException, InterruptedException`

`public final void wait(long msec) throws IllegalMonitorStateException, InterruptedException`

`public final void wait(long msec, int nsec) throws IllegalMonitorStateException, InterruptedException, IllegalArgumentException)`
Causes a thread to suspend execution for `msec` milliseconds and `nsec` nanoseconds. The `wait()` method (without parameters) causes a thread to suspend execution until further notice.

ParsePosition (java.text)

A public class, derived from `Object`, that is used to track the position of the index during parsing. This class is generally used by the `Format` class (and its subclasses).

Constructors

```
public ParsePosition(int index)
```
Creates a new instance of a `ParsePosition` from the specified index.

Methods

```
public int getIndex()
public void setIndex(int num)
```
Returns or sets the parse position.

Point (java.awt)

A public class, derived from `Object` and implementing `Serializable`, that defines and manipulates a location on a two-dimensional coordinate system.

Variables and Constants

```
public int x
public int y
```
The `x` and `y` locations of this point.

Constructors

```
public Point()
public Point(Point pt)
public Point(int x, int y)
```
Creates a new instance of a `Point` from the specified coordinates, the specified point, or using <0, 0> by default.

Methods

```
public boolean equals(Object arg)
```
Returns a true value if this point is identical to `arg`.

```
public Point getLocation()
public void move(int x, int y)
public void setLocation(Point pt)
```

`public void setLocation(int x, int y)`
Returns or relocates the position of this point.

`public int hashCode()`
Returns the hash code of this point.

`public String toString()`
Returns a string representation of this point.

`public void translate(int xoffset, int yoffset)`
Relocates this point to `<x+xoffset, y+yoffset>`.

Polygon (`java.awt`)

A public class, derived from `Object` and implementing `Shape` and `Serializable`, that maintains a list of points that define a polygon shape.

Variables and Constants

`protected Rectangle bounds`
The bounds of this polygon.

`public int npoints`
The total number of points of this polygon.

`public int xpoints[]`

`public int ypoints[]`
The arrays of `x` and `y` locations for the points of this polygon.

Constructors

`public Polygon()`

`public Polygon(int[] x, int[] y, int np)`
Creates a new instance of a polygon, initially defined by the arrays of `x` and `y` locations `<x, y>` and comprised of `np` points. The default constructor creates a new polygon that contains no points.

Methods

`public void addPoint(int newx, int newy)`
Adds the point located at `<newx, newy>` to this polygon.

`public boolean contains(int x, int y)`

`public boolean contains(Point pt)`
Returns a true value if this polygon contains the specified point.

```
public Rectangle getBounds()
```
Returns the bounds of this polygon.

```
public void translate(int xoffset, int yoffset)
```
Relocates all of the `x` and `y` points of this polygon by `xoffset` and `yoffset`.

PrintStream (`java.io`)

A public class, derived from `FilterOutputStream`, that provides methods to print data types in a format other than byte-based.

Constructors

```
public PrintStream(OutputStream out)
public PrintStream(OutputStream out, boolean autoflush)
```
Creates a new instance of a `PrintStream` on `out`. If the `autoflush` value is set to true, then the output buffer is flushed at every occurrence of a newline.

Methods

```
public boolean checkError()
```
Flushes this print stream's buffer and returns a true value if an error occurred.

```
public void close()
```
Closes this print stream.

```
public void flush()
```
Flushes this print stream's buffer.

```
public void print(boolean b)
public void print(char c)
public void print(char[] s)
public void print(double d)
public void print(float f)
public void print(int i)
public void print(long l)
public void print(Object obj)
public void print(String s)
public void println()
public void println(boolean b)
public void println(char c)
public void println(char[] s)
public void println(double d)
```

```
public void println(float f)
public void println(int i)
public void println(long l)
public void println(Object obj)
public void println(String s)
```
Prints the specified Java primitive type, `Object`, or blank line to this print stream. When using a character, only the lower byte is printed.

```
public void write(int b)
public void write(byte[] b, int off, int len)
```
Writes a byte or `len` bytes from the array `b`, starting at index `off` to this print stream.

Random (java.util)

A public class, derived from `Object` and implementing `Serializable`, that produces sequences of pseudo-random numbers.

Constructors

```
public Random()
public Random(long rnd)
```
Creates a new instance of a random class using the value of `rnd` as the random number seed. When the default constructor is used, the current time in milliseconds is the seed.

Methods

```
protected int next(int b)
```
Returns the next random number (from the specified number of bits).

```
public void nextBytes(byte[] b)
```
Generates an array of random bytes as defined by `b[]`.

```
public double nextDouble()
public float nextFloat()
```
Returns a random number between 0.0 and 1.0 in the specified primitive type.

```
public int nextInt()
public long nextLong()
```
Returns a random integer value from all possible int or long values (positive and negative).

```
public double nextGaussian()
```
Returns a Gaussian double random number with a mean value of 0.0 and a standard deviation of 1.0.

```
public void setSeed(long rnd)
```
Sets the seeds for this random number generator to `rnd`.

Rectangle (`java.awt`)

A public class, derived from `Object` and implementing `Shape` and `Serializable`, that represents a rectangular shape that is described by an `x` and `y` location, and a width and height.

Variables and Constants

```
public int height
public int width
```
The height and width of this rectangle.

```
public int x
public int y
```
The `x` and `y` locations of the upper-left corner of this rectangle.

Constructors

```
public Rectangle()
public Rectangle(Dimension dim)
public Rectangle(Point pt)
```
Creates a new instance of a `Rectangle` with an initial location of the corresponding values of `pt` or `dim`, with a height of 0 and width of 0. If neither `pt` or `dim` are specified, then the initial location is <0, 0> and the height and width are set to 0.

```
public Rectangle(Rectangle rect)
public Rectangle(Point pt, Dimension dim)
```
Creates a new instance of a `Rectangle` with initial location and size values the same as corresponding values in `rect`, or with an initial location of the corresponding values of `pt`, and with a width and height corresponding to the values of `dim`.

```
public Rectangle(int width, int height)
public Rectangle(int x, int y, int width, int height)
```
Creates a new instance of a `Rectangle` with an initial location of <`x`, `y`> (or <0, 0> by default), and with a `height` and `width`.

Methods

`public void add(int x, int y)`

`public void add(Point point)`

`public void add(Rectangle rect)`

Adds the specified point in space, defined by coordinates, a point, or the initial location of the specified `Rectangle`, to this `Rectangle`. This method may expand the `Rectangle` (if the point lies outside) or reduce the `Rectangle` (if the point lies inside).

`public boolean contains(int x, int y)`

`public boolean contains(Point pt)`

Returns a true value if this `Rectangle` contains the specified point.

`public boolean equals(Object rect2)`

Returns a true value if this `Rectangle` and the rectangle `rect2` are identical.

`public Rectangle getBounds()`

Returns the bounds of this `Rectangle`.

`public Point getLocation()`

`public Dimension getSize()`

Returns the location or size of this `Rectangle`.

`public void grow(int width, int height)`

Increases this `Rectangle` by `height` and `width` pixels.

`public int hashCode()`

Returns the hash code for this `Rectangle`.

`public Rectangle intersection(Rectangle rect2)`

Returns the intersection of this `Rectangle` and the specified rectangle (`rect2`).

`public boolean intersects(Rectangle rect2)`

Returns a true value if this `Rectangle` intersects `rect2`.

`public boolean isEmpty()`

Returns a true value if this `Rectangle` is empty (height and width <= 0).

`public void setBounds(int x, int y, int width, int height)`

`public void setBounds(Rectangle rect)`

Resets the `x` and `y` locations, width, and height of this `rectangle` to the respective values of `rect` or the specified values of `x`, `y`, `width`, and `height`.

`public void setLocation(int x, int y)`

`public void setLocation(Point pt)`

Resets the location of this `Rectangle` to the specified point.

`public void setSize(Dimension dim)`

`public void setSize(int width, int height)`

Resets the size to `width` and `height`, or the corresponding values of `dim`.

```
public String toString()
```
Returns a string representation of this `Rectangle`.

```
public void translate(int width, int height)
```
Adds the specified `width` and `height` to this `Rectangle`'s width and height values.

```
public Rectangle union(Rectangle rect2)
```
Returns the union of this `Rectangle` and `rect2`.

Scanner (`java.util`)

A public final class, derived from `Object` and implementing `Iterator`, that represents a simple text scanner that can parse primitive types and strings using regular expressions.

Constructors

```
public Scanner (InputStream source)
public Scanner (File source)
public Scanner (String source)
```
Sets up a new instance of `Scanner` to scan values from the specified `source`.

Methods

```
public String next()
```
Returns the next input token as a character string.

```
public String next(String pattern)
public String next(Pattern pattern)
```
Returns the next token that matches the specified pattern.

```
public String nextLine()
```
Returns as a character string all input remaining on the current line.

```
public boolean nextBoolean()
public byte nextByte()
public double nextDouble()
public float nextFloat()
public int nextInt()
public long nextLong()
public short nextShort()
```
Returns the next input token as the indicated type. Throws `InputMismatchException` if the next token is inconsistent with the type.

```
public boolean hasNext()
```
Returns true if the scanner has another token in its input.

```
public boolean hasNext(String pattern)
public boolean hasNext (Pattern pattern)
```
Returns true if the next token matches the specified pattern.
```
public boolean hasNextBoolean()
public boolean hasNextByte()
public boolean hasNextDouble()
public boolean hasNextFloat()
public boolean hasNextInt()
public boolean hasNextLong()
public boolean hasNextShort()
```
Returns true if the next token can be interpreted as the indicated type.
```
public Scanner useDelimiter (String pattern)
public Scanner useDelimiter (Pattern pattern)
```
Sets the scanner's delimiting pattern.
```
public Pattern delimiter()
```
Returns the pattern the scanner is currently using to match delimiters.
```
public String findInLine (String pattern)
public String findInLine (Pattern pattern)
```
Attempts to find the next occurrence of the specified pattern, ignoring delimiters.
```
public Scanner skip (String pattern)
public Scanner skip (Pattern pattern)
```
Skips input that matches the specified pattern.

Short (java.lang)

A public class, derived from `Number`, that contains integer math operations, constants, methods to compute minimum and maximum numbers, and string manipulation routines related to the primitive `short` type.

Variables

```
public final static short MAX_VALUE
public final static short MIN_VALUE
```
A constant value that contains the maximum possible value (32767) or minimum possible value (–32768) of an integer in Java.
```
public final static Class TYPE
```
The `Short` constant value of the short type class.

Constructors

```
public Short(short num)
public Short(String num) throws NumberFormatException
```

Creates a new instance of a `Short` from the specified `num`.

Methods

```
public byte byteValue()
public double doubleValue()
public float floatValue()
public int intValue()
public long longValue()
public short shortValue()
```

Returns the value of this `Short` as a Java primitive type.

```
public static Short decode(String str) throws
NumberFormatException
```

Returns the short representation of the coded argument (`str`). The argument can be coded in decimal, hexadecimal, or octal formats.

```
public boolean equals(Object arg)
```

Returns a true value if this `Short` is equal to the parameter `arg`.

```
public int hashCode()
```

Returns the hash code for this `Short`.

```
public static short parseShort(String str) throws
NumberFormatException
public static short parseShort(String str, int base) throws
NumberFormatException
```

Returns the string argument (`str`) as a short in base 10. The radix of the returned number can be specified in `base`.

```
public static String toString(short num)
public String toString()
```

Returns a string representation of this `Short` or `num`.

```
public static Short valueOf(String str) throws
NumberFormatException
public static Short valueOf(String str, int base) throws
NumberFormatException
```

Returns an instance of a new `Short` object initialized to the value specified in `str`. The radix of the returned number can be specified in `base`.

SimpleDateFormat (java.text)

A public class, derived from DateFormat, that allows for the parsing of dates to locale-based strings, and vice versa.

Constructors

```
public SimpleDateFormat()
public SimpleDateFormat(String str)
public SimpleDateFormat(String str, Locale locale)
```
Creates a new instance of a SimpleDateFormat using the specified or default pattern and the specified or default locale.

```
public SimpleDateFormat(String str, DateFormatSymbols format)
```
Creates a new instance of a SimpleDateFormat using the specified pattern and format data.

Methods

```
public void applyLocalizedPattern(String str)
public String toLocalizedPattern()
```
Sets or returns the locale-based string that describes this SimpleDateFormat.

```
public void applyPattern(String str)
public String toPattern()
```
Sets or returns the non-locale-based string that describes this SimpleDateFormat.

```
public Object clone()
```
Returns a copy of this SimpleDateFormat.

```
public boolean equals(Object arg)
```
Returns a true value if this SimpleDateFormat is equal to arg.

```
public StringBuffer format(Date date, StringBuffer dest,
FieldPosition pos)
```
Formats the specified string, starting at field pos, placing the result in the specified destination buffer. This method returns the value of the buffer.

```
public DateFormatSymbols getDateFormatSymbols()
public void setDateFormatSymbols(DateFormatSymbols symbols)
```
Returns or sets the date/time formatting symbols for this SimpleDateFormat.

```
public int hashCode()
```
Returns the hash code for this SimpleDateFormat.

```
public Date parse(String str, ParsePosition pos)
```
Parses the specified string, starting at position pos, and returns a Date object.

SimpleTimeZone (java.util)

A public class, derived from TimeZone, that represents a time zone in a Gregorian calendar.

Constructors

```
public SimpleTimeZone(int offset, String id)
```

```
public SimpleTimeZone(int offset, String id, int stMonth, int
stNthDayWeekInMonth, int stDayOfWeek, int stTime, int endMonth,
int endNthDayWeekInMonth, int endDayOfWeek, int endTime)
```

Creates a new SimpleTimeZone from an offset from GMT and a time zone id. ID should be obtained from the TimeZone.getAvailableIDs method. You can also define the starting and ending times for daylight savings time. Each period has a starting and ending month (stMonth, endMonth), day of the week in a month (stNthDayWeekInMonth, endNthDayWeekInMonth), day of the week (stDayOfWeek, endDayOfWeek), and time (stTime, endTime).

Methods

```
public Object clone()
```

Returns a copy of this SimpleTimeZone.

```
public boolean equals(Object arg)
```

Returns a true value if this SimpleTimeZone is equal to arg.

```
public int getOffset(int era, int year, int month, int day, int
dayOfWeek, int millisec)
```

Returns the offset from the Greenwich Mean Time (GMT), taking into account daylight savings time.

```
public int getRawOffset()
```

```
public void setRawOffset(int millisec)
```

Returns or sets the offset from Greenwich Mean Time (GMT) for this SimpleTimeZone. These methods do not take daylight savings time into account.

```
public synchronized int hashCode()
```

Returns the hash code for this SimpleTimeZone.

```
public boolean inDaylightTime(Date dt)
```

Returns a true value if the specified date falls within daylight savings time.

```
public void setEndRule(int month, int dyWkInMo, int dyWk, int
tm)
```

```
public void setStartRule(int month, int dyWkInMo, int dyWk, int
tm)
```
Sets the starting and ending times for daylight savings time for this `SimpleTimeZone` to a specified month, day of a week in a month, day of a week, and time (in milliseconds).

```
public void setStartYear(int year)
```
Sets the daylight savings starting year for this `SimpleTimeZone`.

```
public boolean useDaylightTime()
```
Returns a true value if this `SimpleTimeZone` uses daylight savings time.

Stack (`java.util`)

A public class, derived from `Vector`, that represents a last-in-first-out stack.

Constructors

```
public Stack()
```
Creates a new instance of an empty stack.

Methods

```
public boolean empty()
```
Returns a true value if this stack contains no elements.

```
public Object peek() throws EmptyStackException
```
Returns the item on the top of the stack, but does not remove it.

```
public Object pop() throws EmptyStackException
```

```
public Object push(Object obj)
```
Returns and removes the item on the top of the stack (`pop`) or pushes a new item onto the stack (`push`).

```
public int search(Object obj)
```
Returns the relative position of item `obj` from the top of the stack, or –1 if the item is not in this stack.

String (`java.lang`)

A public final class, derived from `Object` and implementing `Serializable`, that contains methods for creating and parsing strings. Because the contents of a string cannot be modified, many of the methods return a new string.

Constructors

```
public String()
public String(byte[] arg)
public String(byte[] arg, int index, int count)
public String(byte[] arg, String code) throws
UnsupportedEncodingException
public String(byte[] arg, int index, int count, String code)
throws UnsupportedEncodingException
```

Creates a new instance of the `String` class from the array `arg`. The parameter `index` indicates which element of `arg` is the first character of the resulting string, and the parameter `count` is the number of characters to add to the new string. The `String()` method creates a new string of no characters. The characters are converted using `code` encoding format.

```
public String(char[] chars)
public String(char[] chars, int index, int count) throws
StringIndexOutOfBoundsException
```

Creates an instance of the `String` class from the array `chars`. The parameter `index` indicates which element of `chars` is the first character of the resulting string, and the parameter `count` is the number of characters to add to the new string.

```
public String(String str)
public String(StringBuffer str)
```

Creates an instance of the `String` class from the parameter `str`.

Methods

```
public char charAt(int idx) throws
StringIndexOutOfBoundsException
```

Returns the character at index `idx` in the current object. The first character of the source string is at index 0.

```
public int compareTo(String str)
```

Compares the current object to `str`. If both strings are equal, 0 (zero) is returned. If the current string is lexicographically less than the argument, an `int` less than zero is returned. If the current string is lexicographically greater than the argument, an `int` greater than zero is returned.

```
public String concat(String source)
```

Returns the product of the concatenation of argument `source` to the end of the current object.

```
public static String copyValueOf(char[] arg)
public static String copyValueOf(char[] arg, int index, int
count)
```

Returns a new `String` that contains the characters of `arg`, beginning at index `index`, and of length `count`.

```
public boolean endsWith(String suff)
```

Returns true if the current object ends with the specified suffix.

```
public boolean equals(Object arg)
public boolean equalsIgnoreCase(String arg)
```

Returns true if the current object is equal to `arg`. `arg` must not be null, and must be of exact length and content as the current object. `equalsIgnoreCase` disregards the case of the characters.

```
public byte[] getBytes()
public byte[] getBytes(String enc) throws
UnsupportedEncodingException
```

Returns the contents of the current object in an array of bytes decoded with `enc`. When a decoding format is not present, the platform default it used.

```
public void getChars(int start, int end, char[] dest, int
destStart)
```

Copies the contents of the current object starting at index `start` and ending at `end` into the character array `dest` starting at index `destStart`.

```
public int hashCode()
```

Returns the hash code of the current object.

```
public int indexOf(char c)
public int indexOf(char c, int index)
```

Returns the index of the first occurrence of the character `c` in the current object not less than `index` (default of 0). Returns a –1 if there is no such occurrence.

```
public int indexOf(String str)
public int indexOf(String str, int index)
```

Returns the index of the first occurrence of the string `str` in the current object not less than `index` (default of 0). Returns a –1 if there is no such occurrence.

```
public String intern()
```

Creates a new canonical string with identical content to this string.

```
public int lastIndexOf(char c)
public int lastIndexOf(char c, int index)
```

Returns the index of the last occurrence of the character `c` in the current object not less than `index` (default of 0). Returns a –1 if there is no such occurrence.

```
public int lastIndexOf(String str)
```

```
public int lastIndexOf(String str, int index)
```
Returns the index of the last occurrence of the string `str` in the current object not less than `index` (default of 0). Returns a –1 if there is no such occurrence.

```
public int length()
```
Returns the integer length of the current object.

```
public boolean regionMatches(boolean case, int cindex, String
str, int strindex, int size)
public boolean regionMatches(int cindex, String str, int
strindex, int size)
```
Returns a true result if the subregion of parameter `str` starting at index `strindex` and having length `size`, is identical to a substring of the current object starting at index `cindex` and having the same length. If `case` is true, then character case is ignored during the comparisons.

```
public String replace(char oldC, char newC)
```
Returns a new string with all occurrences of the `oldC` replaced with the `newC`.

```
public boolean startsWith(String str)
public boolean startsWith(String str, int index)
```
Returns a true if the current object starts with the string `str` at location `index` (default of 0).

```
public String substring(int startindex) throws
StringIndexOutOfBoundsException
public String substring(int startindex, int lastindex) throws
StringIndexOutOfBoundsException
```
Returns the substring of the current object starting with `startindex` and ending with `lastindex-1` (or the last index of the string in the case of the first method).

```
public char[] toCharArray()
public String toString()
```
Returns the current object as an array of characters or a string. Is present due to the automatic use of the `toString` method in output routines.

```
public String toLowerCase()
public String toLowerCase(Locale loc)
```
Returns the current object with each character in lowercase, taking into account variations of the specified locale (`loc`).

```
public String toUpperCase()
public String toUpperCase(Locale loc)
```
Returns the current object with each character in uppercase, taking into account variations of the specified locale (`loc`).

```
public String trim()
```
Returns the current object with leading and trailing white space removed.
```
public static String valueOf(boolean arg)
public static String valueOf(char arg)
public static String valueOf(char[] arg)
public static String valueOf(char[] arg, int index, int size)
public static String valueOf(double arg)
public static String valueOf(float arg)
public static String valueOf(int arg)
public static String valueOf(long arg)
public static String valueOf(Object arg)
```
Returns a string representation of the parameter `arg`. A starting `index` and specified `size` are permitted.

StringBuffer (`java.lang`)

A public class, derived from `Object` and implementing `Serializable`, that contains methods for creating, parsing, and modifying string buffers. Unlike a `String`, the content and length of a `StringBuffer` can be changed dynamically.

Constructors

```
public StringBuffer()
public StringBuffer(int size) throws NegativeArraySizeException
```
Creates an instance of the `StringBuffer` class that is empty but has an initial capacity of `size` characters (16 by default).
```
public StringBuffer(String arg)
```
Creates an instance of the `StringBuffer` class from the string `arg`.

Methods

```
public StringBuffer append(boolean arg)
public StringBuffer append(char arg)
public StringBuffer append(char[] arg)
public StringBuffer append(char[] arg, int index, int size)
public StringBuffer append(double arg)
public StringBuffer append(float arg)
public StringBuffer append(int arg)
public StringBuffer append(long arg)
```

```
public StringBuffer append(Object arg)
public StringBuffer append(String arg)
```
Returns the current object with the `String` parameter `arg` appended to the end. A substring of a character array can be appended by specifying an `index` and `size`.

```
public int capacity()
```
Returns the capacity of this `StringBuffer`.

```
public char charAt(int idx) throws
StringIndexOutOfBoundsException
```
Returns the character at the specified index of this `StringBuffer`.

```
public void ensureCapacity(int min)
```
Sets the minimum capacity of this `StringBuffer` to be no less than `min`. The new capacity set by this method may actually be greater than `min`.

```
public void getChars(int start, int end, char[] dest, int
destindex) throws StringIndexOutOfBoundsException
```
Copies the characters at index `start` to `end` from this `StringBuffer` to `dest`, starting at index `destindex`.

```
public StringBuffer insert(int index, boolean arg) throws
StringIndexOutOfBoundsException
public StringBuffer insert(int index, char arg) throws
StringIndexOutOfBoundsException
public StringBuffer insert(int index, char[] arg) throws
StringIndexOutOfBoundsException
public StringBuffer insert(int index, double arg) throws
StringIndexOutOfBoundsException
public StringBuffer insert(int index, float arg) throws
StringIndexOutOfBoundsException
public StringBuffer insert(int index, int arg) throws
StringIndexOutOfBoundsException
public StringBuffer insert(int index, long arg) throws
StringIndexOutOfBoundsException
public StringBuffer insert(int index, Object arg) throws
StringIndexOutOfBoundsException
public StringBuffer insert(int index, String arg) throws
StringIndexOutOfBoundsException
```
Inserts the string representation of parameter `arg` into this `StringBuffer` at index `index`. Characters to the right of the specified index of this `StringBuffer` are shifted to the right.

```
public int length()
```
Returns the length of this `StringBuffer`.

```
public StringBuffer reverse()
```
Returns the value of this `StringBuffer` with the order of the characters reversed.

```
public void setCharAt(int idx, char c)
```
Sets the character at the specified index to `c`.

```
public void setLength(int size) throws
StringIndexOutOfBoundsException
```
Truncates this `StringBuffer`, if needed, to the new length of `size`.

```
public String toString()
```
Returns the `String` representation of this `StringBuffer`.

StringTokenizer (`java.util`)

A public class, derived from `Object` and implementing `Enumeration`, that manipulates string values into tokens separated by delimiter characters.

Constructors

```
public StringTokenizer(String arg)
public StringTokenizer(String arg, String delims)
public StringTokenizer(String arg, String delims, boolean
tokens)
```
Creates a new instance of a `StringTokenizer` with the string initialized to `arg`, and utilizing the specified delimiters or the defaults (" `\t\n\r`": a space, tab, newline, and carriage return). If `tokens` is true, the delimiters are treated as words within the string and are subject to being returned as tokens.

Methods

```
public int countTokens()
```
Returns the number of tokens present in this string tokenizer.

```
public boolean hasMoreElements()
public boolean hasMoreTokens()
```
Returns a true value if there are more tokens to be returned by this string tokenizer. `hasMoreElements()` is identical to `hasMoreTokens()` and is implemented to complete the implementation of the `Enumerated` interface.

```
public Object nextElement() throws NoSuchElementException
public String nextToken() throws NoSuchElementException
```

```
public String nextToken(String delims) throws
NoSuchElementException
```

Returns the next token in the string. `nextElement()` is identical to `nextToken()` and is implemented to complete the implementation of the `Enumerated` interface. New delimiters can be specified in the last method, and stay in effect until changed.

System (`java.lang`)

A public final class, derived from `Object`, that contains the standard input, output, and error streams, as well as various system related methods.

Variables

```
public static PrintStream err
public static InputStream in
public static PrintStream out
```

Constant values that are the standard error output stream (`stderr`), standard input stream (`stdin`), and the standard output stream (`stdout`).

Methods

```
public static void arraycopy(Object source, int srcindex,
Object dest, int destindex, int size) throws
ArrayIndexOutOfBoundsException, ArrayStoreException
```

Copies a subarray of `size` objects from `source`, starting at index `srcindex`, to `dest` starting at `destindex`.

```
public static long currentTimeMillis()
```

Returns the current system in milliseconds from midnight, January 1st, 1970 UTC.

```
public static void exit(int num) throws SecurityException
```

Exits the program with the status code of `num`.

```
public static void gc()
```

Executes the `gc` method of the `Runtime` class, which attempts to garbage collect any unused objects, freeing system memory.

```
public static Properties getProperties() throws
SecurityException
public static void setProperties(Properties newprops) throws
SecurityException
```

Returns or sets the current system properties.

```
public static String getProperty(String name) throws
SecurityException
public static String getProperty(String name, String default)
throws SecurityException
```

Returns the system property for `name`, or returns the value `default` as a default result if no such `name` exists.

```
public static SecurityManager getSecurityManager()
public static void setSecurityManager(SecurityManager mgr)
throws SecurityException
```

Returns or sets the security manager for the current application. If no security manager has been initialized, then a null value is returned by the `get` method.

```
public static int identityHashCode(Object arg)
```

Returns the hash code for the specified object. This will return the default hash code, in the event that the object's `hashCode` method has been overridden.

```
public static void load(String name) throws
UnsatisfiedLinkError, SecurityException
```

Loads `name` as a dynamic library.

```
public static void loadLibrary(String name) throws
UnsatisfiedLinkError, SecurityException
```

Loads `name` as a system library.

```
public static void runFinalization()
```

Requests that the Java Virtual Machine execute the finalize method on any outstanding objects.

```
public static void runFinalizersOnExit(boolean toggle)
```

Allows the execution of the finalizer methods for all objects, when `toggle` is true.

```
public static void setErr(PrintStream strm)
public static void setIn(InputStream strm)
public static void setOut(PrintStream strm)
```

Reassigns the error stream, input stream, or output stream to `strm`.

SystemColor (`java.awt`)

A public final class, derived from `Color` and implementing `Serializable`, that represents the current window system color for the current system. If the user changes the window system colors for this system and the window system can update the new color selection, these color values will change as well.

Variables and Constants

`public final static int ACTIVE_CAPTION`

Constant index to the active caption color in the system color array.

`public final static int ACTIVE_CAPTION_BORDER`

`public final static int ACTIVE_CAPTION_TEXT`

Constant indices to the active caption border and text colors in the system color array.

`public final static int CONTROL`

Constant index to the control color in the system color array.

`public final static int CONTROL_DK_SHADOW`

`public final static int CONTROL_SHADOW`

Constant indices to the control shadow and control dark shadow colors in the system color array.

`public final static int CONTROL_HIGHLIGHT`

`public final static int CONTROL_LT_HIGHLIGHT`

Constant indices to the control highlight and light highlight colors in the system color array.

`public final static int CONTROL_TEXT`

Constant index to the control text color in the system color array.

`public final static int DESKTOP`

Constant index to the desktop color in the system color array.

`public final static int INACTIVE_CAPTION`

Constant index to the inactive caption color in the system color array.

`public final static int INACTIVE_CAPTION_BORDER`

`public final static int INACTIVE_CAPTION_TEXT`

Constant indices to the inactive caption border and text colors in the system color array.

`public final static int INFO`

Constant index to the information (help) text background color in the system color array.

`public final static int INFO_TEXT`

`public final static int MENU_TEXT`

Constant indices to the information (help) and menu text colors in the system color array.

`public final static int NUM_COLORS`

Constant value that holds the number of colors in the system color array.

`public final static int SCROLLBAR`

Constant index to the scroll bar background color in the system color array.

```
public final static int TEXT
```
Constant index to the background color of text components in the system color array.

```
public final static int TEXT_HIGHLIGHT
public final static int TEXT_HIGHLIGHT_TEXT
```
Constant indices to the background and text colors for highlighted text in the system color array.

```
public final static int TEXT_INACTIVE_TEXT
```
Constant index to the inactive text color in the system color array.

```
public final static int TEXT_TEXT
```
Constant index to the color of text components in the system color array.

```
public final static int WINDOW
```
Constant index to the background color of windows in the system color array.

```
public final static int WINDOW_BORDER
public final static int WINDOW_TEXT
```
Constant indices to the border and text colors of windows in the system color array.

```
public final static SystemColor activeCaption
```
The system's background color for window border captions.

```
public final static SystemColor activeCaptionBorder
public final static SystemColor activeCaptionText
```
The system's border and text colors for window border captions.

```
public final static SystemColor control
```
The system's color for window control objects.

```
public final static SystemColor controlDkShadow
public final static SystemColor controlShadow
```
The system's dark shadow and regular shadow colors for control objects.

```
public final static SystemColor controlHighlight
public final static SystemColor controlLtHighlight
```
The system's highlight and light highlight colors for control objects.

```
public final static SystemColor controlText
```
The system's text color for control objects.

```
public final static SystemColor desktop
```
The system's color of the desktop background.

```
public final static SystemColor inactiveCaption
```
The system's background color for inactive caption areas of window borders.

```
public final static SystemColor inactiveCaptionBorder
```

`public final static SystemColor inactiveCaptionText`
The system's border and text colors for inactive caption areas of window borders.

`public final static SystemColor info`
The system's background color for information (help) text.

`public final static SystemColor infoText`
The system's text color for information (help) text.

`public final static SystemColor menu`
The system's background color for menus.

`public final static SystemColor menuText`
The system's text color for menus.

`public final static SystemColor scrollbar`
The system's background color for scroll bars.

`public final static SystemColor text`
The system's color for text components.

`public final static SystemColor textHighlight`
The system's background color for highlighted text.

`public final static SystemColor textHighlightText`

`public final static SystemColor textInactiveText`
The system's text color for highlighted and inactive text.

`public final static SystemColor textText`
The system's text color for text components.

`public final static SystemColor window`
The system's background color for windows.

`public final static SystemColor windowBorder`

`public final static SystemColor windowText`
The system's border and text colors for windows.

Methods

`public int getRGB()`
Returns the RGB values of this `SystemColor`'s symbolic color.

`public String toString()`
Returns a string representation of this `SystemColor`'s values.

Thread (`java.lang`)

A public class, derived from `Object` and implementing `Runnable`, that handles the implementation and management of Java execution threads.

Variables and Constants

```
public final static int MAX_PRIORITY
public final static int MIN_PRIORITY
public final static int NORM_PRIORITY
```

Constant values that contain the maximum (10), minimum (1), and normal (6) priority values a thread can have.

Constructors

```
public Thread()
```

Creates a new instance of a thread.

```
public Thread(Runnable arg)
```

Creates a new instance of a thread. `arg` specifies which object's `run` method is invoked to start the thread.

```
public Thread(String str)
public Thread(Runnable arg, String str)
```

Creates a new instance of a thread, named `str`. `arg` specifies which object's run method is invoked to start the thread.

```
public Thread(ThreadGroup tgrp, String str) throws
SecurityException
public Thread(ThreadGroup tgrp, Runnable arg) throws
SecurityException
public Thread(ThreadGroup tgrp, Runnable arg, String str) throws
SecurityException
```

Creates a new instance of a thread, named `str` and belonging to thread group `tgrp`. The `arg` parameter specifies which object's `run` method is invoked to start the thread.

Methods

```
public static int activeCount()
```

Returns the number of active threads in this thread's group.

```
public void checkAccess() throws SecurityException
```

Validates that the current executing thread has permission to modify this thread.

```
public static Thread currentThread()
```

Returns the currently executing thread.

```
public void destroy()
```

Destroys this thread.

```
public static void dumpStack()
```
Dumps a trace of the stack for the current thread.

```
public static int enumerate(Thread[] dest)
```
Copies each of the members of this thread's group into the thread array `dest`.

```
public final String getName()
public final int getPriority()
public final ThreadGroup getThreadGroup()
```
Returns the name, priority, or thread group of this thread.

```
public void interrupt()
```
Interrupts this thread's execution.

```
public static boolean interrupted()
```
Returns a true value if the current thread's execution has been interrupted.

```
public final boolean isAlive()
public boolean isInterrupted()
```
Returns a true value if this thread's execution is alive or has been interrupted.

```
public final boolean isDaemon()
```
Returns a true value if this thread is a daemon thread.

```
public final void join() throws InterruptedException
public final void join(long msec) throws InterruptedException
public final void join(long msec, int nsec) throws
InterruptedException
```
Waits up to `msec` milliseconds and `nsec` nanoseconds for this thread to die. The `join()` method waits forever for this thread to die.

```
public void run()
```
Method containing the main body of the executing thread code. Run methods can run concurrently with other thread run methods.

```
public final void setDaemon(boolean flag) throws
IllegalThreadStateException
```
Sets this thread as a daemon thread, if `flag` is true.

```
public final void setName(String str) throws SecurityException
public final void setPriority(int val) throws SecurityException
```
Sets the name of this thread to `str` or the priority to `val`.

```
public static void sleep(long msec) throws InterruptedException
public static void sleep(long msec, int nsec) throws
InterruptedException
```
Causes the current thread to sleep for `msec` milliseconds and `nsec` nanoseconds.

`public void start() throws IllegalThreadStateException`
Start this thread's execution, calling this thread's `run` method.

`public String toString()`
Returns a string representation of this thread.

`public static void yield()`
Causes the currently executing thread to pause in execution, allowing other threads to run.

Throwable (`java.lang`)

A public class, derived from `Object` and implementing `Serializable`, that is the superclass of all of the errors and exceptions thrown.

Constructors

`public Throwable()`

`public Throwable(String str)`
Creates a new instance of a throwable object with the specified message (`str`) or none present.

Methods

`public Throwable fillInStackTrace()`
Fills in the executable stack trace for this throwable object.

`public String getLocalizedMessage()`
Returns a locale-specific description of this object. Locale-specific messages should override this method; otherwise, the same message that the `getMessage` method produces will be returned.

`public String getMessage()`
Returns the detail message for this throwable.

`public void printStackTrace()`

`public void printStackTrace(PrintStream stream)`

`public void printStackTrace(PrintWriter stream)`
Prints the stack trace for this throwable to the standard error stream or to the specified `stream`.

`public String toString()`
Returns a string representation of this throwable object.

Timer (javax.swing)

A public class, derived from `Object` and implementing `Serializable`, that fires an action event after a specified delay. Often used to control animations.

Constructors

```
public Timer(int delay, ActionListener listener)
```
Creates a timer that notifies the specified action listener every `delay` milliseconds.

Methods

```
public void addActionListener(ActionListener listener)
```
Adds the specified action listener to this timer.

```
public int getDelay()
public void setDelay(int delay)
```
Gets or sets this timer's delay (in milliseconds).

```
public void start()
public void stop()
```
Starts or stops this timer.

```
public boolean isRunning()
```
Returns true if this timer is currently running.

TimeZone (java.util)

A public abstract class, derived from `Object` and implementing `Serializable` and `Cloneable`, that represents an amount of time offset from GMT that results in local time. Functionality is provided to allow for daylight savings time within a time zone.

Methods

```
clone()
```
Returns a copy of this `TimeZone`.

```
public static synchronized String[] getAvailableIDs()
public static synchronized String[] getAvailableIDs(int offset)
```
Returns a list of all of the supported time zone ids, or only those for a specified time zone offset.

```
public static synchronized TimeZone getDefault()
```

```
public static synchronized void setDefault(TimeZone tz)
```
Returns or sets the default time zone.

```
public String getID()
```
Returns the id of this time zone.

```
public abstract int getOffset(int era, int year, int month, int
day, int dayOfWeek, int milliseconds)
```
Returns the offset from the Greenwich Mean Time (GMT), taking into account daylight savings time.

```
public abstract int getRawOffset()
public abstract void setRawOffset(int millisec)
```
Returns or sets the offset from Greenwich Mean Time (GMT) for this `SimpleTimeZone`. These methods do not take daylight savings time into account.

```
public static synchronized TimeZone getTimeZone(String id)
```
Returns the time zone corresponding to the specified id value.

```
public abstract boolean inDaylightTime(Date dt)
```
Returns a true result if the specified date falls within the daylight savings time for this `TimeZone`.

```
public void setID(String id)
```
Sets the id value of this `TimeZone`.

```
public abstract boolean useDaylightTime()
```
Returns a true value if this `TimeZone` uses daylight savings time.

URL (`java.net`)

A public final class, derived from `Object` and implementing `Serializable`, that represents a Web Uniform Resource Locator (URL).

Constructors

```
public URL(String arg) throws MalformedURLException
public URL(URL url, String type) throws MalformedURLException
```
Creates a URL instance from a string argument, or by parsing a `type` (http, gopher, ftp) and the remaining base.

```
public URL(String proto, String source, int num, String doc)
throws MalformedURLException
public URL(String proto, String source, String doc) throws
MalformedURLException
```
Creates a URL instance using a defined protocol (`proto`), `source` system, destination port `num`, and document (`doc`).

Methods

`public boolean equals(Object obj)`
Returns a true value if this URL is equal in all respects (protocol, source, port, and document) to `obj`.

`public final Object getContent() throws IOException`
Returns the retrieved contents as an `Object`.

`public String getFile()`

`public String getRef()`
Returns the name of the file (document) or its anchor this URL will attempt to retrieve.

`public String getHost()`

`public int getPort()`
Returns the name of the host (source) or the port this URL will attempt to connect to.

`public String getProtocol()`
Returns the protocol this URL will use in retrieving the data.

`public int hashCode()`
Returns the hash code for this URL.

`public URLConnection openConnection() throws IOException`

`public final InputStream openStream() throws IOException`
Returns a connection to this URL and returns the connection or a stream.

`public boolean sameFile(URL arg)`
Returns a true value if this URL retrieves the same file as the `arg` URL.

`protected void set(String proto, String source, int num, String doc, String anchor)`
Sets the protocol (`proto`), `source`, port `num`, file (`doc`) and reference (`anchor`) for this URL.

`public static void setURLStreamHandlerFactory(URLStreamHandlerFactory fac) throws Error`
Sets the URL `StreamHandlerFactory` for this application to `fac`.

`public String toExternalForm()`

`public String toString()`
Returns a string representation of this URL.

Vector (java.util)

A public class, derived from `Object` and implementing `Serializable` and `Cloneable`, that manages an array of objects. Elements can be added or removed from this list and the size of the list can change dynamically.

Variables and Constants

```
protected int capacityIncrement
```

The amount of element spaces to be added to the vector each time that an increase must occur. A `capacityIncrement` of 0 indicates that the list will double in size at every resizing.

```
protected int elementCount
protected Object elementData[]
```

The number of elements and the array containing the elements currently in this `Vector`.

Constructors

```
public Vector()
public Vector(int size)
public Vector(int size, int incr)
```

Creates a new instance of a vector with an initial size of `size` (or using the default of 10). An initial `capacityIncrement` can also be specified.

Methods

```
public final void addElement(Object arg)
public final void insertElementAt(Object arg, int index) throws
ArrayIndexOutOfBoundsException
```

Adds element `arg` to the end of this `Vector` or at a specific `index`. The capacity of the vector is adjusted if needed.

```
public final int capacity()
public final void ensureCapacity(int size)
```

Returns the current capacity of this `Vector`, or ensures that it can contain at least `size` elements.

```
public Object clone()
```

Returns the clone of this `Vector`.

```
public final boolean contains(Object arg)
```

Returns a true value if this `Vector` contains object `arg`.

```
public final void copyInto(Object[] dest)
```
Copies each of the elements of this `Vector` into the array `dest`.

```
public final Object elementAt(int index) throws
ArrayIndexOutOfBoundsException
```
Returns the element at location `index` from this `Vector`.

```
public final Enumeration elements()
```
Returns an `Enumeration` of the elements in this `Vector`.

```
public final Object firstElement() throws NoSuchElementException
public final Object lastElement() throws NoSuchElementException
```
Returns the first or last element in this `Vector`.

```
public final int indexOf(Object arg)
public final int indexOf(Object arg, int index)
```
Returns the index of the first occurrence of element `arg`, starting at `index`. A –1 value is returned if the element is not found.

```
public final boolean isEmpty()
```
Returns a true value if this `Vector` contains no elements.

```
public final int lastIndexOf(Object arg)
public final int lastIndexOf(Object arg, int index)
```
Returns the first index that object `arg` occurs at in this `Vector`, starting a backwards search at the specified index. If the object is not located, a –1 is returned.

```
public final void removeAllElements()
public final boolean removeElement(Object arg)
public final void removeElementAt(int index) throws
ArrayIndexOutOfBoundsException
```
Removes element `arg` and returns a true value. If the object requested is not located, a false value is returned. An element can also be removed at a specific `index` value, or all elements can be removed.

```
public final void setElementAt(Object arg, int index) throws
ArrayIndexOutOfBoundsException
```
Sets the element at the specified `index` equal to object `arg`.

```
public final void setSize(int size)
```
Sets the size of this `Vector` to `size`.

```
public final int size()
```
Returns the number of elements in this `Vector`.

```
public final String toString()
```
Returns a string representation of this `Vector`.

```
public final void trimToSize()
```
Reduces the size of this `Vector` to contain all of the elements present.

Void (java.lang)

An uninstantiable class that acts as a placeholder for the primitive `void` type in the `Class` object.

Variables and Constants

```
public final static Class TYPE
```
The `Void` constant value of the void type class.

Window (java.awt)

A public class, derived from `Container`, that creates a graphical area that has no borders or menus and can be used to contain AWT components.

Constructors

```
public Window(Frame frm)
```
Creates a new instance of a window that has a parent frame (`frm`). The window is initially not visible.

Methods

```
public void addNotify()
```
Creates this window's peer.

```
public synchronized void addWindowListener(WindowListener
listener)
public synchronized void removeWindowListener(WindowListener
listener)
```
Removes or adds the specified window listener (`listener`) for this window.

```
public void dispose()
```
Removes this window and deletes any resources used by this window.

```
public Component getFocusOwner()
```
Returns the component from this active window that currently has the focus.

```
public Locale getLocale()
```
Returns the locale for this window.

```
public Toolkit getToolkit()
```
Returns the toolkit for this window.

```
public final String getWarningString()
```
Returns the warning string for this window.

`public boolean isShowing()`
Returns a true value if this window is currently visible on the screen.

`public void pack()`
Causes all of the components of this window to be laid out according to their preferred size.

`protected void processEvent(AWTEvent event)`
Processes the specified event for this window. If the event is a `WindowEvent`, then this method calls the process `WindowEvent` method of this window, otherwise it will call the parent class' `processEvent` method.

`protected void processWindowEvent(WindowEvent event)`
Handles any `WindowEvent` (`event`) generated on this window, and passes them to a registered listener for that event.

`public void show()`
Makes this window visible to the user and brings it to the front (on top of other windows).

`public void toBack()`

`void toFront()`
Sends this window to the back or front of other windows currently displayed on the screen.

WindowAdapter (`java.awt.event`)

A public abstract class, derived from `Object` and implementing `WindowListener`, that permits a derived class to override the predefined no-op AWT window events.

Constructors

`public WindowAdapter()`
Creates a new instance of a `WindowAdapter`.

Methods

`public void windowActivated(WindowEvent event)`

`public void windowClosed(WindowEvent event)`

`public void windowClosing(WindowEvent event)`

`public void windowDeactivated(WindowEvent event)`

`public void windowDeiconified(WindowEvent event)`

```
public void windowIconified(WindowEvent event)
public void windowOpened(WindowEvent event)
```
Empty methods that should be overridden in order to implement event handling for window events.

WindowEvent (java.awt.event)

A public class, derived from `ComponentEvent`, that describes a particular AWT window-based event.

Variables and Constants

```
public static final int WINDOW_ACTIVATED
public static final int WINDOW_CLOSED
public static final int WINDOW_CLOSING
public static final int WINDOW_DEACTIVATED
public static final int WINDOW_DEICONIFIED
public static final int WINDOW_FIRST
public static final int WINDOW_ICONIFIED
public static final int WINDOW_LAST
public static final int WINDOW_OPENED
```
Constant values which represent a variety of window event types.

Constructors

```
public WindowEvent(Window src, int type)
```
Creates a new instance of a `WindowEvent` from a specified source window and having a specific event type.

Methods

```
public Window getWindow()
```
Returns the source window that this event was triggered in.

```
public String paramString()
```
Returns a string containing the parameters for this `WindowEvent`.