Classes

JApplet

extends Component (See Methods there also)

Constructors

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

Methods

- `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.

Component

Constructors

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

Methods

- `public synchronized void` **addMouseListener**`(MouseListener listener)`

- `public synchronized void` **addMouseMotionListener**`(MouseMotionListener listener)`

- `public Graphics` **getGraphics**`()`
  Returns the graphics context for this component.

- `public void` **setSize**`(int width, int height)`
  Returns the size of or resizes this component to the specified dimension(s).

- `public void` **paintComponent**`(Graphics gc)`
  Paints this component with the graphics context gc.

- `public void` **repaint**`()`
  Repaints this component.

Graphics

Methods

- `public abstract boolean` **drawImage**`(Image src, int x, int y, ImageObserver obsv)`
  Draws a graphic image (src) at position <x, y>. obsv can be the Applet.
- `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 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 Graphics current font and color.
- `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` **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 abstract Color` **getColor**`()`
- `public abstract void` **setColor**`(Color clr)`
  Sets the current color for this graphics context.

Point

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)`
  Relocates the position of this point to <x, y>.
- `public void` **translate**`(int xoffset, int yoffset)`
  Relocates this point to <x+xoffset, y+yoffset>.

BinaryTree<E>

Variables and Constants

- `private BinaryTree<E> left, right;`
  children; can be null
- `private E data;`

Constructors

- `public` **BinaryTree**`(E data)`
  Constructs leaf node – left and right are null.
- `public` **BinaryTree**`(E data, BinaryTree<E> left, BinaryTree<E> right)`
  Constructs inner node with children left and right.

Methods

- `public boolean` **isInner**`()`
  Is it an inner node?
- `public boolean` **isLeaf**`()`
  Is it a leaf node?
- `public boolean` **hasLeft**`()`
  Does it have a left child?
- `public boolean` **hasRight**`()`
  Does it have a right child?
- `public BinaryTree<E>` **getLeft**`()`
  Return its left child.
- `public BinaryTree<E>` **getRight**`()`
  Return its right child.
- `public void` **setLeft**`(BinaryTree<E> newLeft)`
  Sets its left child to be newLeft.

- `public void` **setRight**`(BinaryTree<E> newRight)`
  Sets its right child to be newRight.
- `public E` **getValue**`()`
  Return its data value.
- `public void` **setValue**`(E newValue)`
  Sets its data value
- `public int` **size**`()`
  Number of nodes (inner and leaf) in tree.
- `public int` **height**`()`
  Longest path to a leaf node from here.
- `public boolean` **equals**`(Object other)`
  Same structure and data?
- `public ArrayList<E>` **fringe**`()`
  Leaves, in order from left to right.
- `private void` **addToFringe**`(ArrayList<E> fringe )`
  Helper for fringe, adding fringe data to the list.
- `public String` **toString**`()`
  Returns a string representation of the tree.
- `private String` **toStringHelper**`(String indent)`
  Recursively constructs a String representation of the tree from this node, starting with the given indentation and indenting further going down the tree.
- `public void` **preorder**`(List<E> dataList )`
  Creates a list storing the the data values in the subtree of a node, ordered according to the preorder traversal of the subtree.
- `public void` **inorder**`(List<E> dataList )`
  Creates a list storing the the data values in the subtree of a node, ordered according to the inorder traversal of the subtree.
- `public void` **postorder**`(List<E> dataList )`
  Creates a list storing the the data values in the subtree of a node, ordered according to the postorder traversal of the subtree.
- `public static <E> BinaryTree<E>` **reconstructTree**`(List<E> preorder, List<E> inorder)`
  Reconstructs tree from a preorder and inorder traversal, assuming that no value is repeated.
  Precondition: the traversals are valid. (Otherwise need lots of error checks.)

MouseEvent

Methods

- `public int` **getClickCount**`()`
  Returns the number of mouse clicks associated with this event.
- `public Point` **getPoint**`()`
  Returns the x,y position of the event relative to the source component.

Interfaces

MouseListener

Methods

- `public void` **mouseClicked**`(MouseEvent e)`
  Invoked when the mouse button has been clicked (pressed and released) on a component.
- `public void` **mouseEntered**`(MouseEvent e)`
  Invoked when the mouse enters a component.
- `public void` **mouseExited**`(MouseEvent e)`
  Invoked when the mouse exits a component.
- `public void` **mousePressed**`(MouseEvent e)`
  Invoked when a mouse button has been pressed on a component.
- `public void` **mouseReleased**`(MouseEvent e)`
  Invoked when a mouse button has been released on a component.

MouseMotionListener

Methods

- `public void` **mouseDragged**`(MouseEvent e)`
  Invoked when a mouse button is pressed on a component and then dragged.
- `public void` **mouseMoved**`(MouseEvent e)`
  Invoked when the mouse cursor has been moved onto a component but no buttons have been pushed.

Iterator<E>

Methods

- `public boolean` **hasNext**`()`
  Returns true if the iteration has more elements.
- `public E` **next**`()`
  Returns the next element in the iteration.
- `public void` **remove**`()`
  Removes from the underlying collection the last element returned by the iterator (optional operation).

List<E>

Methods

- `public boolean` **add**`(E o)`
  Adds the specified element to the end of the list. Returns true if succeeds.
- `public void` **add**`(int index, E o)`
  Inserts the specified element at position index of this list.

- `public void` **clear**`()`
  Removes all of the elements from this list.
- `public boolean` **contains**`(Object o)`
  Returns true if this list contains the specified element.
- `public E` **get**`(int index)`
  Returns the element at position index.
- `public boolean` **isEmpty**`()`
  Returns true if this list contains no elements.
- `public int` **indexOf**`(Object o)`
  Returns the index of the first occurrence of the specified element in the list, or -1 if the list does not contain the element.
- `public Iterator<E>` **iterator**`()`
  Returns an iterator that goes through the elements in this list in the order that they appear in the list.
- `public ListIterator<E>` **listIterator**`()`
  Returns a list iterator that goes through the elements in this list in the order that they appear in the list.
- `public E` **remove**`(int index)`
  Removes the element at the specified position in this list and returns it.
- `public boolean` **remove**`(Object o)`
  Removes the first occurrence of the specified element from this list if it is present. (True if succeeds).
- `public E` **set**`(int index, E element)`
  Replaces the element at the specified position with the given element. Returns the old element.
- `public int` **size**`()`
  Returns the number of elements in this list.

Set<E>

Methods

- `public boolean` **add**`(Object o)`
  Adds the specified element to this set if it is not already present (optional operation).
- `public void` **clear**`()`
  Removes all of the elements from this set (optional operation).
- `public boolean` **contains**`(Object o)`
  Returns true if this set contains the specified element.
- `public boolean` **equals**`(Object o)`
  Compares the specified object with this set for equality.
- `public boolean` **isEmpty**`()`
  Returns true if this set contains no elements.
- `public Iterator<E>` **iterator**`()`
  Returns an iterator over the elements in this set.
- `public boolean` **remove**`(Object o)`
  Removes the specified element from this set if it is present (optional operation).
- `public int` **size**`()`
  Returns the number of elements in this set (its cardinality).

Map<K, V>

Methods

- `public void` **clear**`()`
  Removes all mappings from this map (optional operation).
- `public boolean` **containsKey**`(Object key)`
  Returns true if this map contains a mapping for the specified key.
- `public boolean` **containsValue**`(Object value)`
  Returns true if this map maps one or more keys to the specified value.
- `public Set` **entrySet**`()`
  Returns a set view of the mappings contained in this map.
- `public boolean` **equals**`(Object o)`
  Compares the specified object with this map for equality.
- `public V` **get**`(Object key)`
  Returns the value to which this map maps the specified key.
- `public int` **hashCode**`()`
  Returns the hash code value for this map.
- `public boolean` **isEmpty**`()`
  Returns true if this map contains no key-value mappings.
- `public Set<K>` **keySet**`()`
  Returns a set view of the keys contained in this map.
- `public V` **put**`(K key, V value)`
  Associates the specified value with the specified key in this map (optional operation).
- `public V` **remove**`(Object key)`
  Removes the mapping for this key from this map if it is present (optional operation).
- `public int` **size**`()`
  Returns the number of key-value mappings in this map.