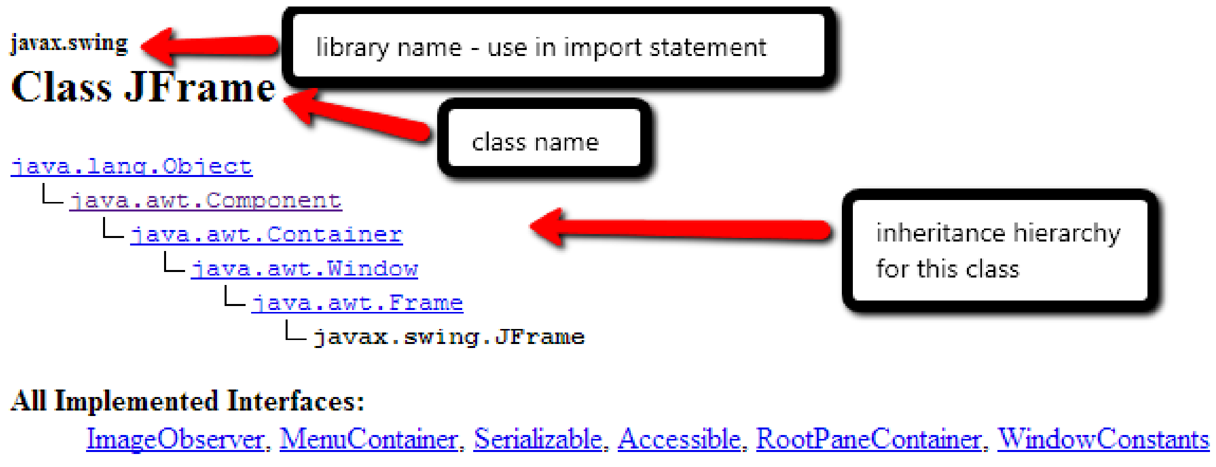
Reading an API page

Top of page:


library name - use in import statement
class name
inheritance hierarchy for this class

javax.swing

Class JFrame

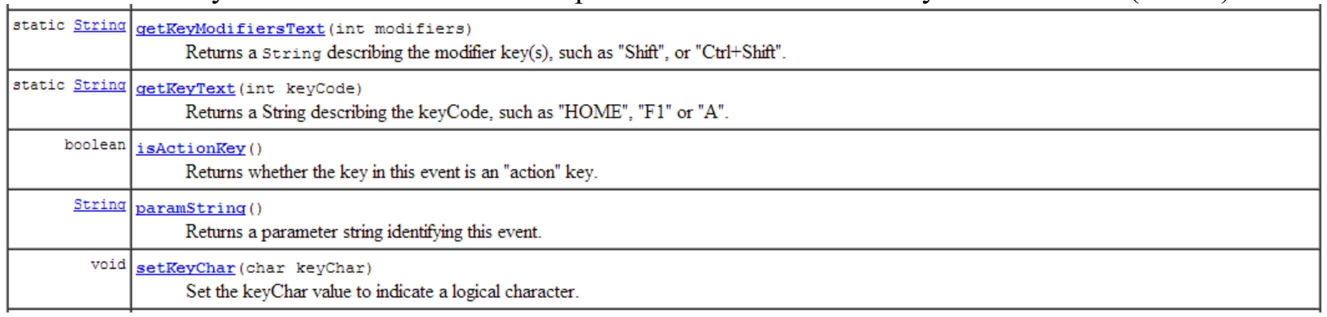
```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └java.awt.Window
              └java.awt.Frame
                  └javax.swing.JFrame
```

All Implemented Interfaces:

ImageObserver, MenuContainer, Serializable, Accessible, RootPaneContainer, WindowConstants

Depending on the class, this section may be followed by a brief (or lengthy) documentation section that describes the class and its uses. Scrolling down, you may encounter sections that deal with nested classes, data fields (variables and constants) and constructors. For now, scroll past all of this; we want to look at the methods sections.

Method summary: the illustration below is a part of the method summary section from a(nother) class:

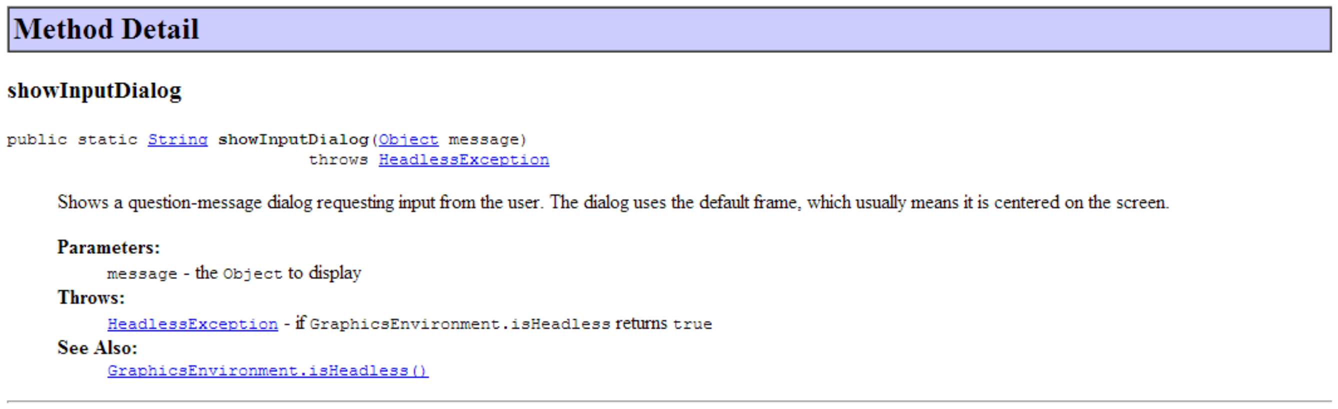


static String	getKeyModifiersText(int modifiers) Returns a String describing the modifier key(s), such as "Shift", or "Ctrl+Shift".
static String	getKeyText(int keyCode) Returns a String describing the keyCode, such as "HOME", "F1" or "A".
boolean	isActionKey() Returns whether the key in this event is an "action" key.
String	paramString() Returns a parameter string identifying this event.
void	setKeyChar(char keyChar) Set the keyChar value to indicate a logical character.

The method summary is organized as follows:

- The box on the left side lists the return type of the method and indicates whether or not the method is static
- The box on the right lists the name of the method and any required parameter(s).

If you click on any method name, or just continue to scroll down until you get past the method summary section, you will reach the detail section(s) describing data fields, constructors, and methods. An example of a method detail entry is shown below:



Method Detail

showInputDialog

```
public static String showInputDialog(Object message)
                              throws HeadlessException
```

Shows a question-message dialog requesting input from the user. The dialog uses the default frame, which usually means it is centered on the screen.

Parameters:
message - the Object to display

Throws:
HeadlessException - if GraphicsEnvironment.isHeadless returns true

See Also:
GraphicsEnvironment.isHeadless()

Methods, method specifications, method calls

Method: encoded algorithm

- Set of instructions to follow in order to solve a problem or reach a goal
- Each method has a unique name, by which we can **call** the method when we want it to **execute**
- A method is an **abstraction** for its algorithm: to do these instructions, call this method

Method specification: tells us minimum information we need to call the method, including:

- whether the method should be called from a class (static) or an object (instance)
- what, if any, kind of value is returned by the method
- method name
- parameter list: indicates number, data type, and order of arguments to be passed to the method when it is called

static data type name (parameter(s))

Examples:

API Class	Method signature	Example of method call
Math	static double pow (double a, double b)	double d2 = Math.pow(x, 2);
Scanner	int nextInt()	int choice = kb.nextInt();
Graphics	void drawRect(int x, int y, int width, int height)	g.drawRect(50,50,95, 125);

Using code from the API

- import statements
- static methods vs. instance methods
- instantiating objects:
  ClassName objectName = new ClassName(arg(s)); // or:
  ClassName objectName;
  objectName = new ClassName(arg(s));

  Examples:
  ```
  Scanner kb = new Scanner(System.in);
  Random rg;
  rg = new Random();
  Color c = new Color(200, 39, 102);
  c = new Color(95, 201, 7);
  ```
- Method calls (Note: a method may be an instruction in and of itself or may be part of another; more on this later):
  - instance methods: *objectName.methodName(arg(s))*
    Examples:
    ```
    System.out.println("Here's a method with an argument");
    int randomNum = rg.nextInt();
    g.setColor(c);
    g.drawArc(x, y, wid, ht, start, sweep);
    ```
  - static methods:
    ClassName.methodName(arg(s))
    Example:
    ```
    double qRoot1 =
          ((-1 * b) +
          Math.sqrt(Math.pow(b, 2) - (4 * a * c)))/(2 * a);
    ```