INTERFACE TYPES

If a method has a parameter of an interface type, then you can supply an object of any class that implements the interface type.

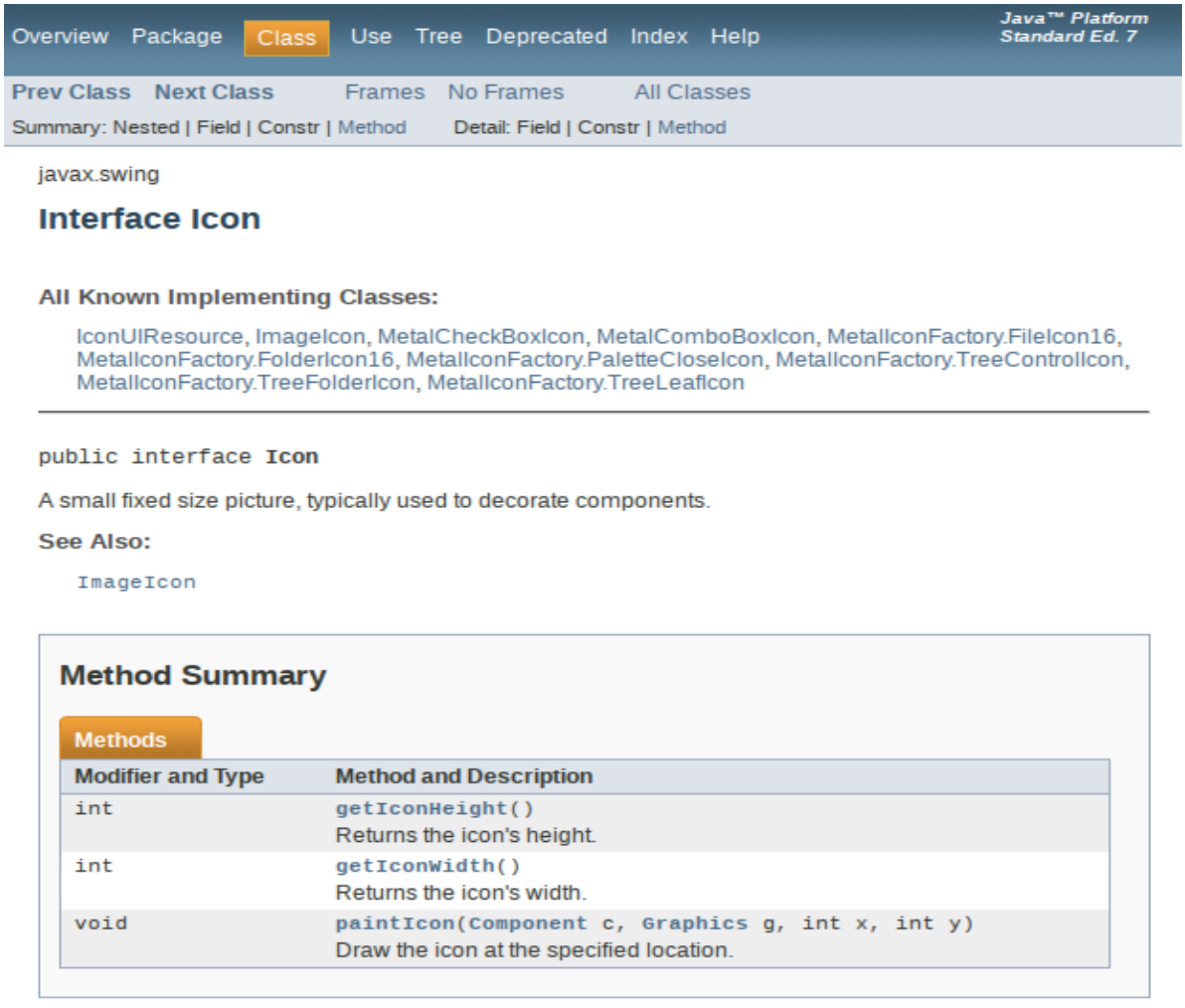
An interface type specifies a set of methods, but it does not implement them.

If a class implements an interface type, its objects can be assigned to variables of the interface type. The *type of an object is never an interface type*. However, the *type of a variable* can be an interface type. Such a variable contains a reference to an object whose class implements the interface type.

Icon Example

JDK defines an Icon interface:

```
public interface Icon{
   int getIconWidth();
   int getIconHeight();
   void paintIcon(Component c, Graphics g, int x, int y);
}
```



Overview Package Class Use Tree Deprecated Index Help — Java™ Platform Standard Ed. 7

Prev Class Next Class Frames No Frames All Classes

Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

javax.swing

Interface Icon

All Known Implementing Classes:

IconUIResource, ImageIcon, MetalCheckBoxIcon, MetalComboBoxIcon, MetalIconFactory.FileIcon16, MetalIconFactory.FolderIcon16, MetalIconFactory.PaletteCloseIcon, MetalIconFactory.TreeControlIcon, MetalIconFactory.TreeFolderIcon, MetalIconFactory.TreeLeafIcon

---

`public interface Icon`

A small fixed size picture, typically used to decorate components.

See Also:

`ImageIcon`

Method Summary

Methods

Modifier and Type	Method and Description
`int`	`getIconHeight()` Returns the icon's height.
`int`	`getIconWidth()` Returns the icon's width.
`void`	`paintIcon(Component c, Graphics g, int x, int y)` Draw the icon at the specified location.

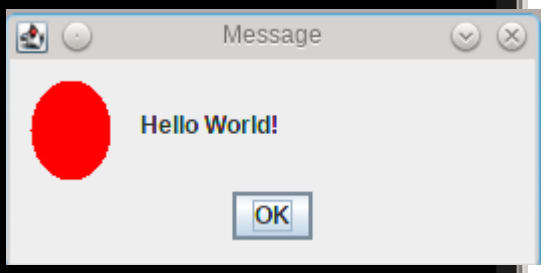
ImageIcon is a class in Java swing package that implements the Icon interface. You can pass an object of ImageIcon class whenever an Icon interface type is expected, as in the following example.

```
JoptionPane.showMessageDialog(
        null, // parent window
        "Hello World!", // message
        "Message", // window title
        JOptionPane.INFORMATION_MESSAGE, // message type
                new ImageIcon("face.png"));
```

Your own Icon class

You can define your own class that implements the Icon interface, and pass it to methods requiring an Icon interface type.

```
import java.awt.*;
import java.awt.geom.*;
import javax.swing.*;

/**
   An Icon
*/
public class myIcon implements Icon {
    private int width;
    private int height;

    public myIcon(int width, int height) {
        this.width=width;
        this.height=height;
    }

    public int getIconWidth() {
        return width;
    }

    public int getIconHeight() {
        return height;
    }

    public void paintIcon(Component c, Graphics g, int x, int y){
        Graphics2D g2 = (Graphics2D) g;
        Ellipse2D.Double ellipse = new Ellipse2D.Double(x, y, width, height);
;
        g2.setColor(Color.RED);
        g2.fill(ellipse);
    }
}

class TestIcon {
    public static void main(String[] args) {
        myIcon icn=new myIcon(40,50);
        JOptionPane.showMessageDialog(null,"Hello World!","Message", JOption
Pane.INFORMATION_MESSAGE, icn);
    }
}
```



Look at JDK Shape interface and some of the implementing classes such as Rectangle2D.Double.

http://docs.oracle.com/javase/7/docs/api/java/awt/Shape.html

You can construct many geometric shapes with the classes implementing Shape interface.