Applets

Applets are small programs that could be run in Java compatible web browsers. Applets provide mechanisms to run programs in the client side and interact with the server hosting it. However, due to security reasons an applet is a restricted version of standard Java applications.

In order to create a Java applet containing swing components, a subclass of `JApplet` class could be declared. `JApplet` is a top level container pane which has a content pane. The controls have to be added to this content not to the `JApplet` directly. The content pane's default layout manager is border layout.

A simple applet could be created simply by subclassing `JApplet` and overriding `init()` method which contains the startup code (corresponds to `main()` method of standard applications) as:

```
import javax.swing.*;
import java.awt.*;
public class SimpleApplet extends JApplet {
        public void init() {
                getContentPane().add(new JLabel("Hello
World!"));
        }
}
```

In order to run above applet inside a web browser, `<APPLET>` tag could be added to an HTML page and applet is located to the same directory with the web page. The applet can be referred in the HTML page

```
<APPLET code="SimpleApplet" width="120" height="40">
</APPLET>
```

It is also possible to run applets from the command prompt by using `appletviewer` tool (available in Sun's JDK) as:

```
appletviewer SimpleApplet.java
```

Applet Creation and Execution

There are four milestone methods that control the creation and execution of an applet. According to the needs these methods could be overridden in the created `JApplet` subclass. These methods and their use are as follows:

`public void init()`: This method is called automatically when the applet is loaded in a web page. It does the initialization of the applet and it should be overridden while subclassing `JApplet`.

`public void start()`: This method is called automatically after the applet is loaded (or the page containing the applet is revisited). It is used to start up the applet's normal execution.

`public void stop()`: This method is called automatically when the page is leaved (or the browser is quitted). It is

used to shut off the applet's normal execution.

`public void destroy()`: This method is called automatically when the applet is being unloaded from the page. It can be used to release resources allocated by the applet.

Applet Benefits and Restrictions

Each time a web page containing an applet is visited, the applet is downloaded from the web server. Therefore, there will be no deployment and installation issues. However, the download time could be very large for large applets. Caching of applets in hosts may reduce the download time but it is not guaranteed.
Most of the capabilities of standard applications are also available for applets. However, there are some restrictions on applet's abilities. These are mainly due to security reasons and change from browser to browser. Some of the restrictions for an applet running on a host are as follows:

Applets cannot load libraries, read/write files, and access certain system properties on the host that is executing it. However, it can access classes and retrieve files located in the host it comes from (by specifying a URL relative to the URL of the page or applet).
Applets can only make network connections to the host it comes from.

It cannot start any program on the host that is executing it. However, it can request the web browser to display HTML documents and call public methods of other applets residing in the same page.

Interaction with Browser and Other Applets

Java enabled web browsers allow an applet to display short/single line messages on the browser's status bar. In order to display a message on the browser's status bar `showStatus()` method of the applet class could be used as:

```
showStatus("this is a simple message");
```

`AppletContext` interface returned by `getAppletContext()` method of the applet class corresponds to an applet's environment. By using this interface, an applet may obtain information about document containing the applet and other applets in the same web page. For example, `showDocument()` method could be called to request web browser to display page specified by the url as:

```
getAppletContext().showDocument(url);
```

`AppletContext` interface could also be used to learn other applets in the same page. For this purpose, the following methods could

be used:

`getAppletContext().getApplets()`: This method returns an enumeration of all applets in the current page. This method could be used as:

```
Enumeration e = getAppletContext().getApplets();
while(e.hasMoreElements()){
  Applet a = (Applet) e.nextElement();
  ...
}
```

`getAppletContext().getApplet(name)`: this method returns the applet with the specified name. If the applet is found, the instance of the applet class is returned by the method. Then it is possible to call public methods of the returned applet. Applet names could be specified in the <APPLET> tag as:

```
<APPLET NAME="myApplet" ...> </APPLET>
```

Parameters

In order to customize an applet's operation according to the user's preferences, applet parameters could be used. A URL, an integer, a floating point, a boolean, or a string value can be given to an applet by using parameters. Applet parameters could be supplied to an applet in the `<APPLET>` tag as:

```
<APPLET ...>
        <PARAM NAME="xxx" VALUE="yyy">
        ...
</APPLET>
```

Applet parameters could be used in the code by calling `getParameter(name)` method of the applet class. For example, for the above applet, the value of parameter `xxx` could be learned as:

```
getParameter("xxx");
```

In addition to defining parameters for an applet, a description of these parameters could be defined by overriding `getParameterInfo()` method of the applet class. This will enable browsers to help the user to set appropriate parameters for the applet. The `getParameterInfo()` method should return an array containing information about parameters as:

```
public String[][] getParameterInfo() {
String[][] info = {
  {"parameter name", "parameter value", "description"},
        ...
  };
  return info;
}
```

Loading Data Files

In some cases, an applet requires loading some data files to

perform its operations. Such files could be downloaded from the host where the applet is loaded from by specifying a complete or relative `URL` of the file requested.
In order to learn where the applet is loaded from, `getCodeBase()` and `getDocumentBase()` methods could be used. The `getCodeBase()` method returns the URL where the applet's classes reside. The `getDocumentBase()` method returns the URL where the HTML page containing the applet is located. There are special methods in applet class to download an image or a audio clip file by specifying a complete url of the file or a url and a name (a path relative to the specified url) for the file. These methods are as follows:

```
getImage(url);
getImage(url, name);
getAudioClip(url);
getAudioClip(url, name);
```

For other file types, it is possible to use getClass() method of the applet class to get a stream for the file. Then that stream could be used to read the contents of the file. An input stream for a resource on the server could be created as:

```
InputStream s = getClass().getResourceAsStream(name);
```

Using Java Archives

If an applet makes use of too many classes or data files, the download time of the applet may be very large. In this case, an achieve file which bundles all necessary files in a single file could be formed. This will reduce the download time especially when the archive file is compressed.
The files required by an applet could be packaged in a JAR (the standard java archive format based on the ZIP file format) file by using the jar utility. For example, the class and image files required by an applet could be packaged by issuing the following command:

```
jar cvf myapplet.jar *.class *.gif
```

In order to specify the JAR file containing the applet code, archieve attribute of <APPLET> could be used as:

```
&ltAPPLET code=MyClass.class archive=myapplet.jar
...></APPLET>
```