

## Chapter 8: Applets and Application

[Credit: 3 hrs]

### 8.1. Fundamental concept of applet

Applet is a Java program that can be embedded into a web page. It runs inside the web browser and works at client side. After an applet arrives on the client, it has limited access to resources so that it can produce a graphical user interface and run complex computations without introducing the risk of viruses or breaching data integrity. Applets differ from console-based applications in several key areas. Applet is embedded in a HTML page using the APPLET or OBJECT tag and hosted on a web server.

Applets are used to make the web site more dynamic and entertaining.

#### Characteristics of Applet

1. Applets are small Java applications that can be accessed on an Internet server, transported over Internet, and can be automatically installed and run as apart of a web document.
2. After a user receives an applet, the applet can produce a graphical user interface. It has limited access to resources so that it can run complex computations without introducing the risk of viruses or breaching data integrity.
3. Any applet in Java is a class that extends the java.applet.Applet class.
4. An Applet class does not have any main() method. It is viewed using JVM. The JVM can use either a plug-in of the Web browser or a separate runtime environment to run an applet application.
5. JVM creates an instance of the applet class and invokes init() method to initialize an Applet.

#### Advantages of Applets

1. It takes very less response time as it works on the client side.
2. It can be run on any browser which has JVM running in it. Hence, platform independent
3. Execution of applets is easy in a Web browser and does not require any installation or deployment procedure in real-time programming (where as servlets require).
4. Writing and displaying (just opening in a browser) graphics and animations is easier than applications.
5. In GUI development, constructor, size of frame, window closing code etc. are not required (but are required in applications).

#### Disadvantages:

1. Requires java plug-in

#### Two Types of Applet

There are two varieties of applets.

1. The first type of applets uses the **Abstract Window Toolkit (AWT)** to provide the graphic user interface (or use no GUI at all). This style of applet has been available since Java was first created.
2. The second type of applets is those based on the Swing class **JApplet**. Swing applets use the Swing classes to provide the GUI. *Swing offers a richer and often easier-to-use user interface than does the AWT.* Thus, Swing-based applets are now the most popular. However, traditional AWT-based applets are still used, especially when only a very simple user interface is required.

Thus, both AWT- and Swing-based applets are valid. Because **JApplet** inherits **Applet**, all the features of **Applet** are also available in **JApplet**.

## 8.2. Simple applet and Applet & Applications

### A Simple Java “Hello World” Applet

```
import java.awt.Graphics;
import java.applet.Applet;
public class Hello extends Applet
{
    public void paint(Graphics g)
    {
        g.drawString("Hello World",25,20);
    }
}
```

1. This applet begins with two *import* statements. The first imports the *Abstract Window Toolkit* (AWT) classes. Applets interact with the user (either directly or indirectly) through the AWT, not through the console-based I/O classes. The AWT contains support for a window-based, graphical user interface.
2. The second *import* statement imports the *applet* package, which contains the class *Applet*. Every applet that we create must be a subclass of *Applet*.
3. The next line in the program declares the class *Hello*. This class must be declared as *public*, because it will be accessed by code that is outside the program.
4. Inside *Hello* class, *paint()* is declared. This method is defined by the AWT and must be overridden by the *applet*. *paint()* is called each time that the applet must redisplay its output. This situation can occur for several reasons. For example, *the window in which the applet is running can be overwritten by another window and then uncovered*. Or, *the applet window can be minimized and then restored*. *paint()* is also called when the applet begins execution. Whatever the cause, whenever the applet must redraw its output, *paint()* is called. The *paint()* method has one parameter of type *Graphics*. This parameter contains the graphics context, which describes the graphics environment in which the applet is running. This context is used whenever output to the applet is required.
5. Inside *paint()* is a call to *drawString()*, which is a member of the *Graphics* class. This method outputs a string beginning at the specified X,Y location. It has the following general form:

*void drawString(String message, int x, int y)*

6. Here, *message* is the string to be output beginning at x,y. In a Java window, the upper-left corner is location 0,0. The call to *drawString()* in the applet causes the message “Hello World” to be displayed beginning at location 20,20.
7. Notice that the applet *does not have a main()* method. Unlike Java programs, applets do not begin execution at *main()*. In fact, most applets don’t even have a *main()* method. Instead, an

applet begins execution *when the name of its class is passed to an applet viewer* or to a network browser.

8. After we enter the source code for *Hello*, compile in the same way that we have been compiling programs. However, running *Hello* involves a different process. In fact, there are two ways in which you can run an applet:

- Executing the applet within a Java-compatible web browser.
- Using an applet viewer,

### Viewing the Applet:

Save the above code as Hello.java and compile it. To compile: Go to command prompt >type `javac Hello.java` and press enter. You will get a class file *Hello.class*

Then create a html file and save it as Test.html and run it.

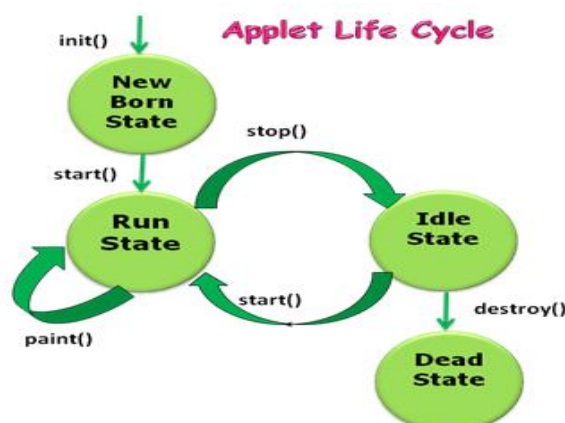
```
<html>
  <body bgcolor = "yellow">
    <applet code= "Hello.class" width= "100" height = "200" >
    </applet>
  </body>
</html>
```

In command prompt;

**C:\> appletviewer Test.html**

We will get a view of applet.

### Applet life cycle



#### 1. New Born State

The life cycle of an applet begins when the applet is first loaded into browser and called the *init()* method. At this stage, new objects to the applet are created, initial values are set, images are loaded and the colors of the images are set. An applet is initialized only once in its lifetime.

Its general form is:

```
public void init( )
{
    //action to be performed
}
```

### 2. Running State

An Applet achieves the running state when the system calls the **start( )** method. This occurs as soon as the Applet is initialized.

General form is:

```
public void start( )
{
    //action to be performed
}
```

### 3. Idle State

An Applet comes in Idle state when its execution has been stopped either implicitly or explicitly. An Applet is implicitly stopped when we leave the page containing the current applets. An Applet is explicitly stopped when we call **stop( )** method to stop its execution.

General form is:

```
public void stop( )
{
    //action to be performed
}
```

### 4. Dead state

An Applet is in dead state when it has been removed from the memory. This can be done using **destroy( )** method.

General form is:

```
public void destroy( )
{
    //action to be performed
}
```

Apart from the above stages, Java Applet also possess **paint( )** method. This method helps in drawing, writing and creating colored backgrounds of the applet. It takes an argument of the graphic class. To use the graphics, it imports the **package.awt.Graphics.**

## Methods used in Applet Life Cycle

Methods	Description
<code>public void init()</code>	Called once by the <i>applet container</i> when an applet is loaded for execution. This method initializes an applet. Typical actions performed here are initializing fields, creating GUI components, loading sounds to play, loading images to display and creating threads.
<code>public void start()</code>	Called by the <i>applet container</i> after method <i>init</i> completes execution. In addition, if the user browses to another website and later returns to the applet's HTML page, method <i>start</i> is called again. The method performs any tasks that <i>must be completed</i> when the applet is loaded for the first time and that must be performed every time the applet's HTML page is revisited. Actions performed here might include starting an animation or starting other threads of execution.
<code>public void paint(Graphics g)</code>	Called by the <i>applet container</i> after methods <i>init</i> and <i>start</i> . Method <i>paint</i> is also called when the applet needs to be repainted. For example, if the user covers the applet with another open window on the screen and later uncovers it, the <i>paint</i> method is called. Typical actions performed here involve drawing with the <i>Graphics</i> object <i>g</i> that's passed to the <i>paint</i> method by the applet container.
<code>public void stop()</code>	The <i>applet container</i> calls this method when the user leaves the applet's web page by browsing to another web page. Since it's possible that the user might return to the web page containing the <i>applet</i> , method <i>stop</i> performs tasks that might be required to suspend the applet's execution, so that the applet does not use computer processing time when it's not displayed on the screen. Typical actions performed here would stop the execution of animations and threads.
<code>public void destroy()</code>	The <i>applet container</i> calls this method when the applet is being removed from memory. This occurs when the user exits the browsing session by closing all the browser windows and may also occur at the browser's discretion when the user has browsed to other web pages. The method performs any tasks that are required to <i>clean up</i> resources allocated to the applet.

## An Applet Skeleton

Four of these methods, *init()*, *start()*, *stop()*, and *destroy()*, apply to all applets and are defined by *Applet*. Default implementations for all of these methods are provided below:

```
// An Applet skeleton.
import java.awt.*;
import java.applet.*;
/*
<applet code="AppletSkel" width=300 height=100>
</applet>
*/
public class AppletSkel extends Applet {
    // Called first.
    public void init() {
        // initialization
    }
    /* Called second, after init(). Also called whenever the applet is restarted. */
    public void start() {
        // start or resume execution
    }
    // Called when the applet is stopped.
    public void stop() {
        // suspends execution
    }
    /* Called when applet is terminated. This is the last method executed. */
    public void destroy() {
        // perform shutdown activities
    }
    // Called when an applet's window must be restored.
    public void paint(Graphics g) {
        // redisplay contents of window
    }
}
```

## The Applet classes

The **Applet** class defines the methods shown in following table. **Applet** provides all necessary support for applet execution, such as *starting and stopping*. It also provides methods that *load and display images*, and *methods that load and play audio clips*. **Applet** extends the **AWT class Panel**. In turn, **Panel** extends **Container**, which extends **Component**. These classes provide support for Java's window-based, graphical interface. Thus, **Applet** provides all of the necessary support for window based activities.

Methods	Description
<code>void destroy()</code>	Called by the browser just before an applet is terminated.
<code>AccessibleContext getAccessibleContext()</code>	Returns the accessibility context for the invoking object.
<code>AppletContext getAppletContext()</code>	Returns the context associated with the applet.
<code>String getAppletInfo()</code>	Returns a string that describes the applet.
<code>AudioClip getAudioClip(URL url)</code>	Returns an AudioClip object that encapsulates the audio clip found at the location specified by url.
<code>URL getCodeBase()</code>	Returns the URL associated with the invoking applet.
<code>Image getImage(URL url)</code>	Returns an Image object that encapsulates the image found at the location specified by url.
<code>Locale getLocale()</code>	Returns a Locale object that is used by various locale sensitive classes and methods.
<code>String getParameter(String paramName)</code>	Returns the parameter associated with <i>paramName</i> . <i>null</i> is returned if the specified parameter is not found.
<code>void init()</code>	Called when an applet begins execution. It is the first method called for any applet.
<code>boolean isActive()</code>	Returns true if the applet has been started. It returns false if the applet has been stopped.
<code>void play(URL url)</code>	If an audio clip is found at the location specified by url, the clip is played.
<code>void play(URL url, String clipName)</code>	If an audio clip is found at the location specified by url with the name specified by clipName, the clip is played.
<code>void start()</code>	Called by the browser when an applet should start (or resume) execution. It is automatically called after <i>init()</i> when an applet first begins.
<code>void stop()</code>	Called by the browser to suspend execution of the applet. Once stopped, an applet is restarted when the browser calls <i>start()</i> .
<code>void showStatus(String str)</code>	Displays str in the status window of the browser or applet viewer. If the browser does not support a status window, then no action takes place.

And many more.. (refer to text book)

Table: The Methods Defined by Applet

## Converting Java Applet to Application and vice-versa

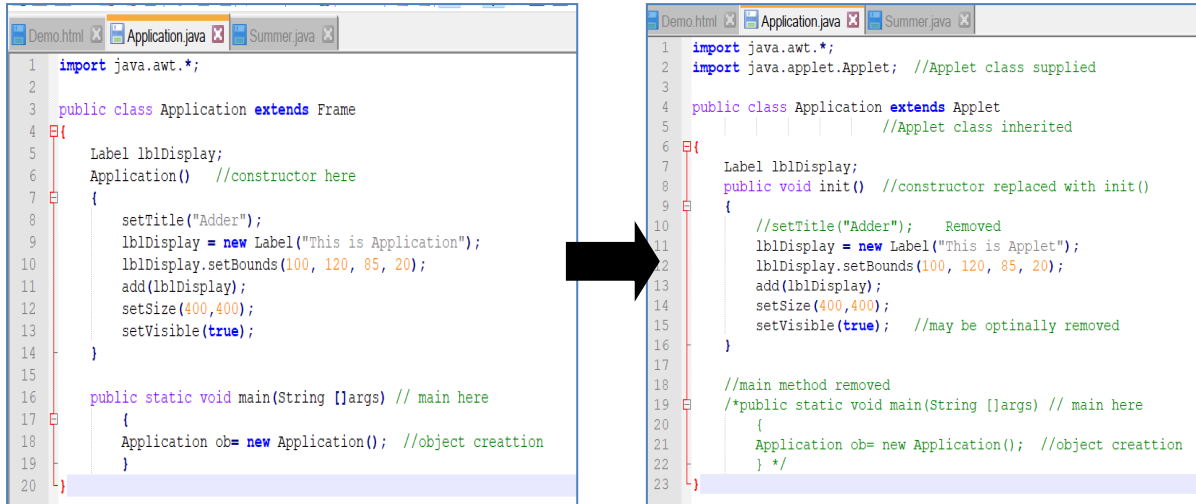
### 1. Converting a java application into java applet

The basic steps to follow to convert an application program into an applet program are:

1. Make a HTML page with appropriate tag to load the applet code.
2. Supply a subclass of **Applet** or **JApplet** class. Make this class public.
3. Eliminate the **main()** method.
4. Override the **init()** method to initialize your applet's resources the same way the **main()** method initializes the applicant's resources. **init()** might be called more than once and should be designed accordingly.

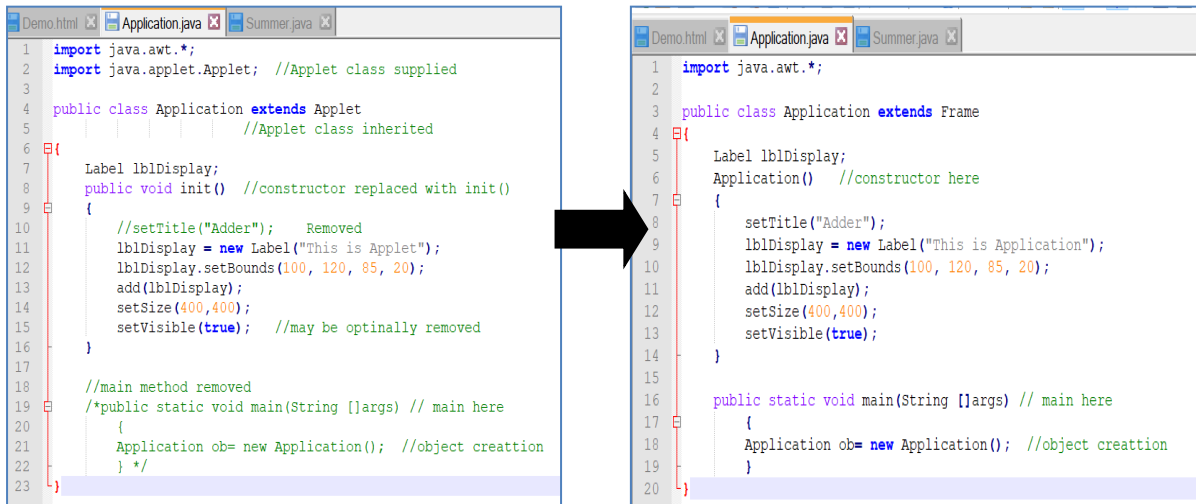


- Remove the call to **setSize** as setting the size of the applet is done in the HTML code with the **WIDTH** and **HEIGHT** parameters. Also, remove the call to **setDefaultCloseOperation** and **setTitle** which is not relevant for applets.
- Since the applet is displayed automatically, do not call **show()** or **setVisible(true)**.



## 2. Converting a java applet into java application

- Supply a subclass of **Frame**
- Eliminate the **init()** method and replace it with constructor or methods.
- Introduce **setSize()** and **setVisible()** methods.
- Introduce a **main()** method.



Refer: <http://www.mathcs.emory.edu/~cheung/Courses/377/Syllabus/8JDBC/GUI/Applets/applet.html>

## Applet Vs Application

Features	Application	Applet
main() method	Present	Not present
Execution	Requires JRE	Requires a browser or applet viewer
Nature	Called as stand-alone application as application can be executed from command prompt	Requires some third party tool help like a browser to execute
Restrictions	Can access any data or software available on the system	cannot access any thing on the system except browser's services
Security	Does not require any security	Requires highest security for the system as they are untrusted

## The html APPLET tag

The APPLET tag is used to start an applet from both an HTML document and from an applet viewer. An applet viewer will execute each APPLET tag that it finds in a separate window, while web browsers will allow many applets on a single page.

The syntax for a fuller form of the APPLET tag is shown below; bracketed items are optional while *code*, *width* and *height* attributes are required.

```
<APPLET
[CODEBASE= codebase URL]
[CODE= appletFile]
[ALT= alternate text]
[NAME=appletInstanceName]
[WIDTH=pixels]
[HEIGHT=pixels]
[ALIGN=alignment]
[VSPACE= pixels]
[HSPACE=pixels] >

[<PARAM NAME= AttributeName VALUE =Attribute Value>]
[<PARAM NAME= AttributeName2 VALUE =Attribute Value >]
.....
[HTML displayed in the absence of Java]
</APPLET>
```

Table below shows the different attributes of APPLET tag and their purposes:

S.NO.	Attributes of APPLET tag	Description
1.	CODEBASE	<ul style="list-style-type: none"> <li>It species the base URL of the applet code, which is the directory</li> </ul>



		<p>that will be searched for the applet's executable class file.</p> <ul style="list-style-type: none"> <li>It is optional.</li> </ul>
2.	CODE	<ul style="list-style-type: none"> <li>It is a required attribute that gives the name of the file containing applet's compiled <code>.class</code> file.</li> <li>This file is relative to the code base URL of the applet, which is the directory that the HTML file was in or the directory indicated by CODEBASE if set.</li> </ul>
3.	ALT	<ul style="list-style-type: none"> <li>It is used to specify a short text message that should be displayed if the browser recognizes the APPLET tag but cannot currently run Java applets.</li> <li>It is optional.</li> </ul>
4.	NAME	<ul style="list-style-type: none"> <li>It is used to specify a name for the applet instance.</li> <li>Applet must be named in order for other applets on the same page to find them by name and communicate with them.</li> <li>To obtain an applet by name, use <code>getApplet()</code>, which is defined by the <code>AppletContext</code> interface.</li> </ul>
5.	WIDTH and HEIGHT	<ul style="list-style-type: none"> <li>These are required attributes that give the size (in pixels) to the applet display area.</li> </ul>
6.	ALIGN	<ul style="list-style-type: none"> <li>It specifies the alignment of the applet.</li> <li>The values can be: LEFT, RIGHT, TOP, BOTTOM, MIDDLE, BASELINE, TEXTTOP, ABSMIDDLE and ABSBOTTOM.</li> <li>It is optional.</li> </ul>
7.	VSPACE and HSPACE	<ul style="list-style-type: none"> <li>VSPACE specifies the space, in pixels, above and below the applet.</li> <li>HSPACE specifies the space, in pixels, on each side of the applet.</li> <li>These are optional attributes.</li> </ul>
8.	PARAM NAME	<ul style="list-style-type: none"> <li>PARAM tag allows us to specify applet-specific arguments in an HTML page.</li> <li>Applets access their attributes with the <code>getParameter()</code> method.</li> </ul>

### 8.3. Parameters to Applet

We can pass parameters to applets using the `<param>` tag and retrieving the values of parameters using `getParameter()` method..

The `<param>` tag is a sub tag of the `<applet>` tag. The `<param>` tag contains two attributes: *name* and *value* which are used to specify the name of the parameter and the value of the parameter respectively. For example, the param tags for passing name and age parameters looks as shown below:

```
<param name="name" value="Ramesh" />
<param name="age" value="25" />
```

Now, these two parameters can be accessed in the applet program using the `getParameter()` method of the `Applet` class.

```
String getParameter(String param-name)
```

Let's look at a sample program which demonstrates the `<param>` HTML tag and the `getParameter()` method:

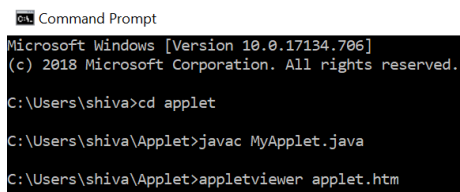
Creating a java file as MyApplet.java

```
import java.awt.*;
import java.applet.*;
public class MyApplet extends Applet
{
    String n;
    String a;
    public void init()
    {
        n = getParameter("name");
        a = getParameter("age");
    }
    public void paint(Graphics g)
    {
        g.drawString("Name is: " + n, 20, 20);
        g.drawString("Age is: " + a, 20, 40);
    }
}
```

Creating a HTML file as applet.html

```
<html>
  <body bgcolor = "yellow">
    <applet code="MyApplet" height="300" width="500">
      <param name="name" value="Ramesh" />
      <param name="age" value="25" />
    </applet>
  </body>
</html>
```

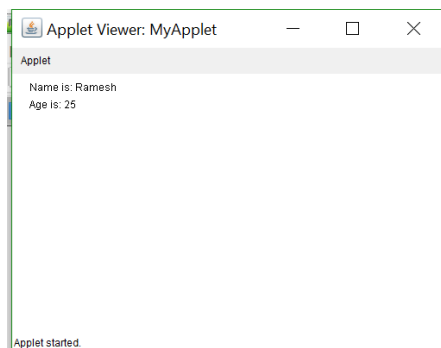
In the command prompt, we create a class file and run the applet viewer by commanding as:



```
Command Prompt
Microsoft Windows [Version 10.0.17134.706]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\shiva>cd applet
C:\Users\shiva\Applet>javac MyApplet.java
C:\Users\shiva\Applet>appletviewer applet.htm
```

The output is obtained as:



## 8.4. Applet architecture

An applet is a window-based program. As such, its architecture is different from the console-based programs. First, applets are event driven. It is important to understand in a general way that the eventdriven architecture impacts the design of an applet.

An applet resembles a set of *interrupt service routines*. Here is how the process works. An applet waits until an **event** occurs. The **run-time system** notifies the applet about an event by calling an event handler that has been provided by the applet. Once this happens, the applet must take appropriate action and then quickly return. This is a crucial point. For the most part, applet should not enter a “mode” of operation in which it maintains control for an extended period. Instead, *it must perform specific actions in response to events and then return control to the run-time system*. In those situations in which applet needs to perform a repetitive task on its own (for example, displaying a scrolling message across its window), we must start an additional thread of execution.

Second, the user initiates interaction with an applet. These interactions are sent to the applet *as events to which the applet must respond*. For example, when the user clicks the mouse inside the applet’s window, a mouse-clicked event is generated. If the user presses a key while the applet’s window has input focus, a *keypress* event is generated. **Applets** can contain various controls, such as push buttons and check boxes.

When the user interacts with one of these controls, an event is generated.

## 8.5. Applet Security policies

Because applets are designed to be loaded from a remote site and then executed locally, security becomes vital. If a user enables Java in the browser, the browser will download all the applet code on the web page and execute it immediately. The user never gets a chance to confirm or to stop individual applets from running. For this reason, applets (unlike applications) are restricted in what they can do. The applet security manager throws a **SecurityException** whenever an applet attempts to violate one of the access rules. The restricted execution environment for applets is often called the “sandbox.” Applets playing in the “sandbox” cannot alter the user’s system or spy on it.

In particular, when running in the sandbox:

- Applets can never run any local executable program.
- Applets cannot communicate with any host other than the server from which they were downloaded; that server is called the **originating host**. This rule is often called “applets can only phone home.” This protects applet users from applets that might try to spy on intranet resources.
- Applets cannot read from or write to the local computer’s file system.
- Applets cannot find out any information about the local computer, except for the Java version used, the name and version of the operating system, and the characters used to separate files (for instance, / or \), paths (such as : or ;), and lines (such as *ln* or *lrln*). In particular, applets cannot find out the user’s name, e-mail address, and so on.
- All windows that an applet pops up carry a warning message.

To allow for different levels of security under different situations, we can use **signed applets**. A signed applet carries with it a certificate that indicates the identity of the signer. Cryptographic techniques ensure that such a certificate cannot be forged. If we trust the signer, we can choose to give the applet additional rights.

Programs from vendors that are known to be somewhat flaky can be given access to some, but not all, privileges.

Unknown applets can be restricted to the sandbox.

To sum up, Java has three separate mechanisms for enforcing security:

1. Program code is interpreted by the Java Virtual Machine, not executed directly.

2. A security manager checks all sensitive operations in the Java runtime library.
3. Applets can be signed to identify their origin.

## Swing Vs Applet

S.No.	Swing	Applet
1.	Swing is light weight Component.	Applet is heavy weight Component.
2.	Swing have look and feel according to user view you can change look and feel using UIManager.	Applet does not provide this facility.
3.	Swing used for standalone Applications. Swing have <i>main()</i> method to execute the program.	Applet needs HTML code to run the Applet.
4.	Swing uses MVC Model view Controller.	Applet does not.
5.	Swing have its own Layout like most popular Box Layout.	Applet uses AWT Layouts like flowlayout.
6.	Swings have some Thread rules.	Applet doesn't have any rule.
7.	To execute Swing, no need of any browser	To execute Applet program, we need any one browser like: Appletviewer, web browser.

## Exam Questions:

1. What is applet? Explain about the applet life cycle. [2018 spring]
2. Explain applet architecture. How can you convert applet to application? Explain with suitable example. [2018 fall]
3. Write a code to create a simple applet in Java. Describe the differences between applet and application. [2017 fall]
4. What is applet? Explain the lifecycle of an applet and demonstrate the same using a simple program. [2017 spring]
5. Explain about the lifecycle of Applet? Write a simple but meaning applet program. [2016 fall]
6. What is applet? Explain about Applet architecture and security. [2016 spring]
7. How can you convert a java application into java applet? Explain with an example code for both applet and application. [2015 spring]
8. What are Applet security policies as compared to application? Give the suitable example to access parameter value to applet application. [2015 fall]
9. Differentiate between java application and java applet. Write a java applet to display an image. [2014 spring]
10. What are the parameters to applets? Also describe different applet security policies. [2014 Fall]
11. How does applet differ from applications? Explain the purposes of different attributes of Applet tags. [2013 spring]

\*\*\*