CHAPTER-1
INTRODUCING APPLET

Introduction:

- Applets are small programs that are designed for **transmission over the Internet and run within a browser**.

  Since Java's virtual machine handles running Java programs, like applets, they're a safe option for downloading and running programs online.

Two general varieties of applets:

1. based on the Abstract Window Toolkit (AWT)
2. based on Swing.

- Both the AWT and Swing support the creation of a graphical user interface (GUI).
- The AWT is the **original GUI toolkit** and Swing is Java‘s **lightweight alternative.**
- Swing-based applets are built upon the same basic architecture as AWT-based applets.
- Swing is built on top of the AWT.

```
// A Minimal AWT-based applet.
import     java.awt.*;
import java.applet.*;
public class SimpleApplet extends Applet {
    public void paint (Graphics g) {
        g.drawString (—Java makes applets easy. —, 20, 20);
    }
}
```

- This applet begins with two import statements. The first imports the Abstract Window Toolkit 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.
- The next import statement imports the applet package. This package contains the class Applet.

- Every applet that you create must be a subclass (either directly or indirectly) of Applet.
- The next line in the program declares the class SimpleApplet. This class must be declared as public because it will be accessed by outside code.
- Inside SimpleApplet, paint( ) is declared.
- This method is defined by the AWT Component class (which is a superclass of Applet) and is overridden by the applet.
- paint( ) is called each time the applet must redisplay its output. This 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 will contain 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.
- Inside paint( ), there 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.

- **General Form: void drawString(String message, int x, int y)**
- 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 to be displayed beginning at location 20,20

Working Of Java when compared to Stand alone Java Application:

- The applet does not have a main( ) method. Unlike the standalone application program, 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 a browser or other applet-enabled program.

- Compiling applet program is similar to stand alone application program. Running an applet is different from running a standalone application.

There are two ways to run an applet:

1. **Inside a browser**(Open an HTML file containing the applet.)
2. **Using AppletViewer**(Run the ‘appletViewer’ command with the HTML file.)

- The tool provided with the standard Java JDK is called appletviewer use it to run the applets.
- We can also run them in your browser, but the appletviewer is much easier to use during development.

- One way to execute an applet (in either a Web browser or the appletviewer) is to write a short HTML text file with the <applet> tag to specify the applet and its dimension .
- Save the HTML file.
- Run the HTML file with appletViewer.
- **Example:** If the preceding HTML file is called StartApp.html, then the following command line will run SimpleApplet: C:\> appletviewer StartApp.html

Preparing to Write Applets

Before writing applets, ensure Java is installed properly and that you have either the Java appletviewer or a Java-enabled browser available.

The steps involved in developing and testing in applet are:

1. Building an applet code(.java file)
2. Creating an executable applet(.class file)
3. Designing a Web page using HTML tags
4. Preparing <APPLET> Tag
5. Incorporating <APPLET> Tag into the web page
6. Creating HTML file
7. Testing the applet code

Building Applet Code

- Applet code uses the services of two classes, namely, **Applet and Graphics** from the Java class library.
- The Applet class which is contained in the java.applet package provides life and behaviour to the applet through its methods such as init(), start() and point().
- Java calls the main() method directly to initiate the execution of the program . When an applet is loaded, Java automatically calls a series of Applet class methods for starting , running, and stopping the applet code

- The paint() method of the Applet class, when it is called, actually displays the result of the applet code on the screen. The output may be text, graphics, or sound.
- The paint() method, which requires a Graphics object as an arguement, is defined as follows.

  public void paint(Graphics g)

- This requires that the applet code imports the java.awt package that contains the Graphics class.
- All output operations of an applet are performed using the methods defined in the Graphics class

Syntax:

```
Import java.awt.*;
Import java.applet.*;
 ……..
 ……..

Public class appletclassname extends Applet
{
 ……
 Public void paint(Graphics g)
  {
   ……..
   ……..
  }
 ……..
 ……..
}
```

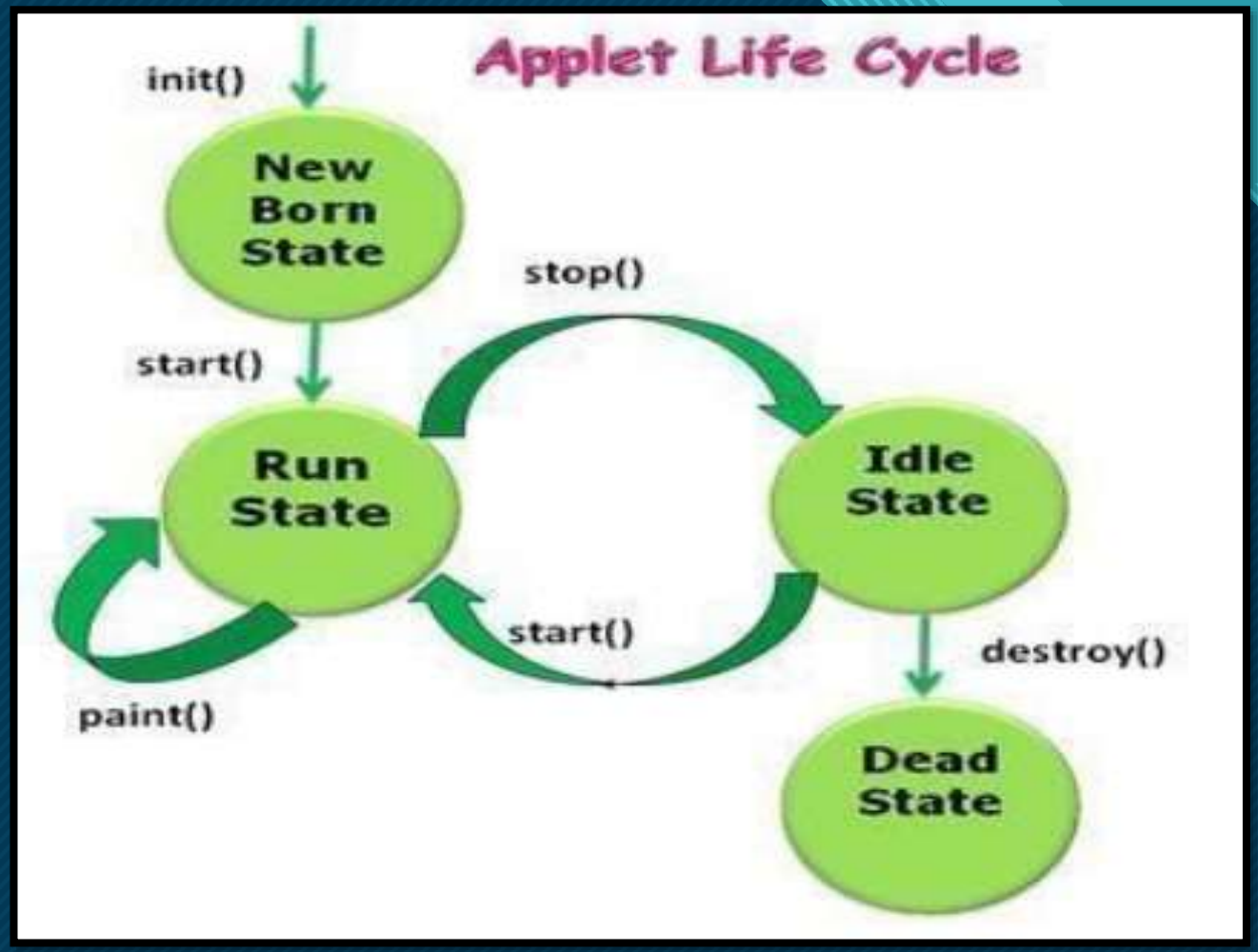
Example:

```
Import java.awt.*;
Import java.applet.*;
Public class HelloJava extends Applet
 {
Public void paint(Graphics g)
  {
g.drawString(—Hello Javal, 10,100);
    }
}
```

Applet Life Cycle

When an applet is loaded, it undergoes a series of changes in its state.

- Born or initialization state
- Running state
- Idle state
- Dead or destroyed state



Initialization State

Applet enters the initialization state when it is first loaded. This is achieved by calling the init() method of Applet class. The applet is born.

At this stage, we may do the following:

- Create objects needed by the applet
- Set up initial values
- Load images or fonts
- Set up colors

```
public void init()
{ ……
  …… }
```

Running State

Applet enters the running state when the system calls the start() method of Applet class. This occurs automatically after the applet is initialized.

```
public voids start() { …… …… }
```

Idle or stopped state

An applet becomes idle when it is stopped from running.

Stopping occurs automatically when we leave the page containing the currently running applet. We can also do so by calling the stop() method.

public void stop()

{……

……... }

Dead State

An applet is said to be dead when it is removed from memory. This occurs by invoking the destroy() method when we quit the browser. Destroying stage occurs only once in the applet‘s life cycle.

public void destroy()

{ ……

…… …… }

Display State Applet moves to the display state whenever it has to perform some output operations on the screen. This happens immediately after the applet enters into the running state. The paint() method is called to accomplish this task.

public void paint(Graphics g) { …… ……..}

The paint() method is defined in the Applet class.

Applet Tag

We have included a pair of tags in the body section.

The tag supplies the name of the applet to be loaded and tells the browser how much space the applet requires. The ellipsis in the tag indicates that it contains certain attributes that must specified.

This HTML code tells the browser to load the compiled Java applet HelloJava.class, which is in the same directory as the HTML fileAnd also specifies the display area for the applet output as 400 pixels width and 200 pixels height.

We can make this display area appear in the centre of the screen by using the CENTER tags shown as follows.

Note That:

<Applet> tag specifies three tings:

1. Name of the applet
2. Width of the applet(in pixels)
3. Height of the applet(in pixels)

```
<APPLET
   CODE=helloJava.class
   WIDTH=400
   HEIGHT=200>
</APPLET>
```

```
<CENTER>
   <APPLET
   ..............
   ..............
</APPLET>
</CENTER>
```

Adding Applet to HTML File We can put together the various components of the Web page and create a file known as HTML file. Insert the <APPLET> tag in the page at place where the output of the applet must appear.

Content of the HTML file that is embedded with <APPLET> tag of out Hellojava applet.

```
<HTML>
<! It specifies the applet to be loaded and executed.
>
<HEAD>
        <TITLE>
              Welcome to Java
        </TITLE>
</HEAD>
<BODY>
        <CENTER>
              <H1> Welcome to the world of Applet </H1>
        </CENTER>
        <BR>
        <CENTER>
```

Passing parameters to Applet

We can supply user-defined parameters to an applet using tags. Each tag has a name attribute such as color, and a value attribute such as red. Inside the applet code, the applet can refer to that parameter by name to find its value. For example, we can change the colour of the text displayed to red by an applet by using a < PARAM... > tag as follows.

```
<APPLET ..... >
<PARAM = color VALUE = "redl >
</APPLET>
```

we can change the text to be displayed by an applet by supplying new text to the applet through a tag as shown below:

```
<PARAM NAME = text VALUE = ‖I love Java —>
```

Passing parameters to an applet code using tag is something similar to passing parameters to the main( ) method using command line arguments. To set up and handle parameters, we need to do two things: 1. include appropriate tags in the HTML document. 2. Provide Code in the applet to parse these parameters. Parameters are passed to an applet when it is loaded. We can define the init( ) method in the applet to get hold of the parameters defined in the tags. This is done using the getParameter( ) method., which takes one string argument representing the name of the parameter and returns a string containing the value of that parameter.

Applet HellojavaParm

```
import java.awt.*;
import java.applet.*;
public class HelloJavaParam extends Applet
{
String str;
public void init( )
{
str = getParameter ("string—);        //Receiving parameter value
if (str = = null)
        str=‖Java‖;
                str= —Hello‖ + str                    //Using the value
          }
public void paint (Graphics g)
{
g.drawString(str, 10, 100);
}
}
```

Now we have to create HTML file that contains this applet. below program shows a web page that passes a parameter whose name is "string" and whose VALUE is "Applet!" to the applet HelloJavaParam.

The HTML file for HelloJavaParam applet

```
<HTML>
<! Parameterized HTML file>
<HEAD>
<TITLE> Welcome to Java Applets </TITLE>
<HEAD>
<BODY>
<APPLET CODE = HelloJavaParam.class
WIDTH = 400
HEIGHT = 200>
<PARAM NAME = "string"
VALUE = "Applet! ">
</APPLET>
</BODY>
</HTML>
```

This will produce output as Hello Applet. if we remove tag from HTML file will produce output as Hello Java