

UNIT 12

Introduction to Java Applets

(1 Hour) — Object Oriented Programming in Java

Course Code: CACS204 Year/Semester: II/III

12.1 Definition of Java Applet

A Java applet is a small Java program designed to run inside an applet-compatible environment. Unlike a normal Java application, an applet does not normally begin execution with a `main()` method. Its execution is controlled by the applet environment through predefined lifecycle methods.

Historically, applets were commonly embedded in web pages and executed using the Java plug-in. The syllabus topic is mainly concerned with understanding the applet concept, its lifecycle, controls, and basic animation.

Basic characteristics

It is a Java program intended for an applet environment.

Its execution is controlled by lifecycle methods such as `init()`, `start()`, `paint()`, `stop()`, and `destroy()`.

It can display graphics and interact with user-interface controls.

12.2 Applet Lifecycle Methods

The applet lifecycle describes the sequence in which the applet environment initializes, starts, paints, stops, and finally removes an applet.

Method	Purpose
<code>init()</code>	Called once when the applet is initialized. Used for initial setup such as creating objects or setting values.
<code>start()</code>	Called when the applet becomes active. It may also be called again after the applet has been stopped.
<code>paint(Graphics g)</code>	Used to display text, shapes, images, and other graphical output.
<code>stop()</code>	Called when the applet becomes inactive, such as when the user leaves the page.
<code>destroy()</code>	Called when the applet is being removed. Used for final cleanup.

Applet Lifecycle

Stage	Flow	Stage	Flow	Stage
init()	→	start()	→	paint()
↑				↓
destroy()	←	stop()	←	inactive

The lifecycle can be viewed as: initialization → execution/display → temporary stopping → restarting when active again → final destruction.

12.3 Building a Simple Applet

A basic applet can extend the Applet class and override lifecycle or drawing methods. The following example demonstrates the traditional applet structure.

```
import java.applet.Applet;
import java.awt.Graphics;

public class SimpleApplet extends Applet {
    public void init() {
        // initialization
    }

    public void paint(Graphics g) {
        g.drawString("Hello Java Applet", 50, 50);
    }
}
```

Explanation

Applet is the base class being extended.

init() is available for one-time initialization.

paint(Graphics g) receives a graphics context and is used to draw the message.

The coordinates (50, 50) specify the approximate position where the text is drawn.

12.4 Using Applet Viewer

The Applet Viewer was a JDK development utility used to test applets without placing them in a web browser. Traditionally, an HTML file could specify the applet class and parameters, and the Applet Viewer could load that file for testing.

Traditional HTML embedding

```
<applet code="SimpleApplet.class" width="400" height="200"></applet>
```

A typical historical workflow was: write the Java applet → compile the source → create the HTML page containing the applet specification → open the page with Applet Viewer.

Note: Java applets and the browser plug-in technology they depended on are obsolete in modern web browsers. Applet Viewer is therefore mainly relevant as a historical/academic syllabus topic.

12.5 Adding Controls

An applet can use graphical user-interface controls to receive input from the user. Traditional Java AWT controls include buttons, labels, text fields, checkboxes, and choices.

Control	Use
Label	Displays non-editable text.
Button	Allows the user to trigger an action by clicking.
TextField	Accepts a single line of text input.
Checkbox	Allows an option to be selected or deselected.
Choice	Provides a drop-down list of options.

Simple control concept

A Button can be added to an applet interface and an event handler can respond when the user clicks it.

A TextField can be used to obtain text entered by the user.

12.6 Animation Concepts

Animation in a graphical program is produced by repeatedly changing the position or appearance of an object and repainting the display. In an applet, a common traditional approach is to use a separate thread or repeated timing mechanism to update the object's state and request repainting.

Basic animation sequence

- Create or initialize the object to be animated.
- Maintain variables representing its position or state.
- Change those values periodically.
- Request the display to be repainted.
- Repeat the update and repaint cycle to create the appearance of movement.

Conceptual animation example

Suppose a circle has an x-coordinate x . On each update, increase x by a small amount and repaint the applet.

If the display is repainted many times per second, the changing position gives the viewer the impression that the circle is moving.

Key Takeaways

- ✓ A Java applet is a small Java program designed for an applet execution environment.
- ✓ The main lifecycle methods are `init()`, `start()`, `paint()`, `stop()`, and `destroy()`.
- ✓ `paint(Graphics g)` is used for graphical output.
- ✓ Applet Viewer was historically used to test applets outside a web browser.
- ✓ AWT controls such as Button, Label, TextField, Checkbox, and Choice can provide user interaction.
- ✓ Animation is achieved by repeatedly updating an object's state and repainting the display.
- ✓ Java applets are obsolete in modern web browsers, but the concepts remain useful for understanding the historical Java syllabus topic.

— *End of Unit 12* —