Topic 5

Java Applets & Graphics Programming

Total Marks- 20

Specific Objectives:

- **The students will be able to write interactive applets and make use of graphics in programming.**
- **They will also learn to change the background and the foreground color and to use the different fonts.**

Contents:

5.1 Introduction to applets Applet, Applet life cycle (skeleton), Applet tag, Adding Applet To HTML file, passing parameter to applet, embedding <applet>tags in java code, adding controls to applets.

5.2 Graphics Programming Graphics classes, lines, rectangles, ellipse, circle, arcs, polygons, color & fonts, setColor(), getColor(), setForeGround(), setBackGround(), font class, variable defined by font class: name, pointSize, size, style, font methods: getFamily(), getFont(), getFontname(), getSize(), getStyle(), getAllFonts() & getavailablefontfamilyname() of the graphics environment class.

Topic 5 Java Applets & Graphics Programming

5.1.Introduction to applets

Applet Basics-

Basically, an applet is dynamic and interactive java program that inside the web page or applets are small java programs that are primarily used in internet computing. The java application programs run on command prompt using java interpreter whereas the java applets can be transported over the internet form one computer to another and run using the appletviewer or any web browser that supports java.

An applet is like application program which can perform arithmetic operations, display graphics, play sounds accept user input, create animation and play interactive games. To run an applet, it must be included in HTML tags for web page. Web browser is a program to view web page.

Every applet is implemented by creating sub class of Applet class. Following diagram shows the inheritance hierarchy of Applet class.

java.lang.Object

java.awt.Component

java.awt.Container

java.awt.Panel

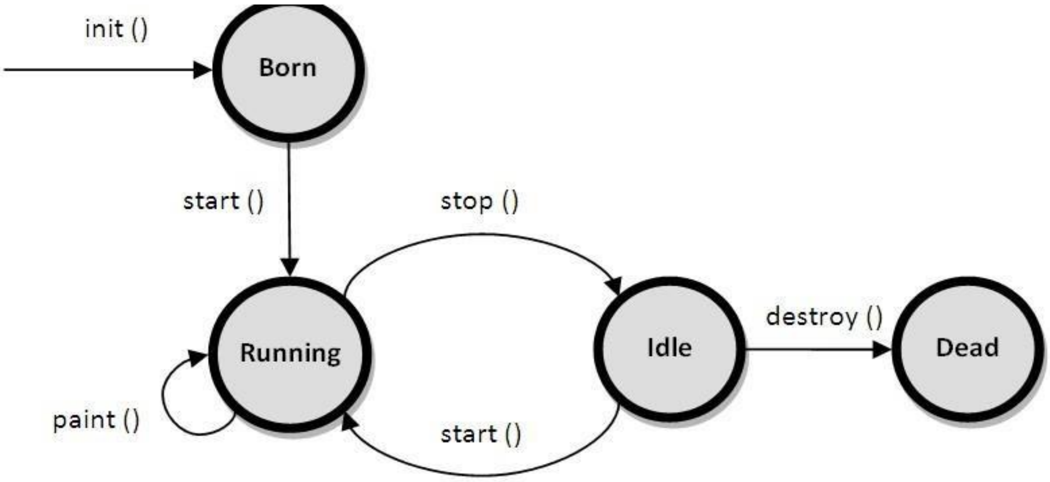
java.applet.Applet

Fig. Chain of classes inherited by Applet class in java

5.1.1 Differentiate between applet and application (4 points). [W-14, S-15, W-15]

Applet	Application
Applet does not use main() method for initiating execution of code	Application use main() method for initiating execution of code
Applet cannot run independently	Application can run independently
Applet cannot read from or write to files in local computer	Application can read from or write to files in local computer
Applet cannot communicate with other servers on network	Application can communicate with other servers on network
Applet cannot run any program from local computer.	Application can run any program from local computer.
Applet are restricted from using libraries from other language such as C or C++	Application are not restricted from using libraries from other language

5.1.2 Applet life Cycle [W-14, S-15]



Applets are small applications that are accessed on an Internet server, transported over the Internet, automatically installed, and run as part of a web document.

The applet states include:

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

a) Born or initialization state [S-15]

Applet enters the initialization state when it is first loaded. This is done by calling the init() method of Applet class. At this stage the following can be done:

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

Initialization happens only once in the life time of an applet.

```
public void init()
        {
        //implementation
        }
```

b) Running state: [S-15]

Applet enters the running state when the system calls the start() method of Applet class. This occurs automatically after the applet is initialized. start() can also be called if the applet is already in idle state. start() may be called more than once. start() method may be overridden to create a thread to control the applet.

```
public void start()
        {
        //implementation
        }
```

c) Idle or stopped state:

An applet becomes idle when it is stopped from running. Stopping occurs automatically when the user leaves the page containing the currently running applet. stop() method may be overridden to terminate the thread used to run the applet.

```
public void stop()
        {
        //implementation
        }
```

d) Dead state:

An applet is dead when it is removed from memory. This occurs automatically by invoking the destroy method when we quit the browser. Destroying stage occurs only once in the lifetime of an applet. destroy() method may be overridden to clean up resources like threads.

```
public void destroy()
        {
        //implementation
        }
```

e) Display state: [S-15]

Applet is in the display state when it has to perform some output operations on the screen. This happens after the applet enters the running state. paint() method is called for this. If anything is to be displayed the paint() method is to be overridden.

```
public void paint(Graphics g)
        {
        //implementation
        }
```

5.1.3 Applet Tag & Attributes [W-15, S-16]

APPLET Tag:

The APPLET tag is used to start an applet from both an HTML document and from an applet viewer.

The syntax for the standard APPLET tag:

```
<APPLET

    [CODEBASE = codebaseURL]
    CODE = appletFile
    [ALT = alternateText]
    [NAME = appletInstanceName]
    WIDTH = pixels HEIGHT = pixels
    [ALIGN = alignment]
    [VSPACE = pixels]  [HSPACE = pixels]>
    [< PARAM NAME = AttributeName1 VALUE = AttributeValue>]
    [<PARAM NAME = AttributeName2    VALUE = AttributeValue>]
     . . .

</APPLET>
```

CODEBASE is an optional attribute that specifies the base URL of the applet code or the directory that will be searched for the applet‟s executable class file.

CODE is a required attribute that give the name of the file containing your applet‟s compiled class file which will be run by web browser or appletviewer.

ALT: Alternate Text. The ALT tag is an optional attribute used to specify a short text message that should be displayed if the browser cannot run java applets.

NAME is an optional attribute used to specifies a name for the applet instance.

WIDTH AND HEIGHT are required attributes that give the size(in pixels) of the applet display area.

ALIGN is an optional attribute that specifies the alignment of the applet.
The possible value is: LEFT, RIGHT, TOP, BOTTOM, MIDDLE, BASELINE, TEXTTOP, ABSMIDDLE, and ABSBOTTOM.

VSPACE AND HSPACE attributes are optional, VSPACE specifies the space, in pixels, about and below the applet. HSPACE VSPACE specifies the space, in pixels, on each side of the applet

PARAM NAME AND VALUE: The PARAM tag allows you to specifies applet-specific arguments in an HTML page applets access there attributes with the get Parameter()method.

Q. Explain <PARAM> tag of applet with suitable example. [S-15]

To pass parameters to an applet <PARAM… > tag is used. Each <PARAM…> tag has a name attribute and a value attribute. Inside the applet code, the applet can refer to that parameter by name to find its value.

The syntax of <PARAM…> tag is as follows

<PARAM NAME = name1 VALUE = value1>

To set up and handle parameters, two things must be done.
1. Include appropriate <PARAM..> tags in the HTML document.
2. Provide code in the applet to parse these parameters.

Parameters are passed on an applet when it is loaded. Generally init() method in the applet is used to get hold of the parameters defined in the <PARAM…> tag.
The getParameter() method, which takes one string argument representing the name of the parameter and returns a string containing the value of that parameter.

Example

```
import java.awt.*;
import java.applet.*;

public class hellouser extends Applet
```

```
{
String str;
public void init()
{
str = getParameter("username");
str = "Hello "+ str;
}
public void paint(Graphics g)
{
g.drawString(str,10,100);
}
}

<HTML>
<Applet code = –hellouser.class‖ width = 400 height = 400>
<PARAM NAME = "username" VALUE = abc>
</Applet>
</HTML>
```

Q. How can parameter be passed to an applet? Write an applet to accept user name in the form of parameter and print „Hello<username>". [W-15]

Passing Parameters to Applet

User defined parameters can be supplied to an applet using <PARAM…..> tags. PARAM tag names a parameter the Java applet needs to run, and provides a value for that parameter.

PARAM tag can be used to allow the page designer to specify different colors, fonts, URLs or other data to be used by the applet.

To set up and handle parameters, two things must be done.

1. Include appropriate <PARAM..>tags in the HTML document.

The Applet tag in HTML document allows passing the arguments using param tag.
The syntax of <PARAM…> tag

```
<Applet code="AppletDemo" height=300 width=300>
<PARAM NAME = name1 VALUE = value1>
</Applet>
```

NAME: attribute name
VALUE: value of attribute named by corresponding PARAM NAME.

2. Provide code in the applet to parse these parameters.

The Applet access their attributes using the getParameter method.
The syntax is : **String getParameter(String name);**

Program for an applet to accept user name in the form of parameter and print „Hello<username>‟ [W-15]

```
import java.awt.*;
import java.applet.*;

public class hellouser extends Applet
{
String str;
public void init()
{
str = getParameter("username");
str = "Hello "+ str;
}
public void paint(Graphics g)
{
g.drawString(str,10,100);
}
}
```

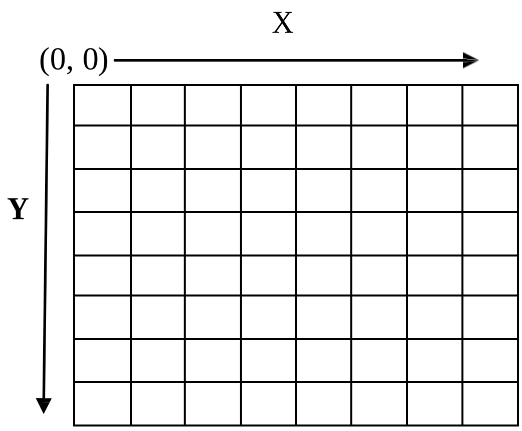
```
<HTML>
<Applet code = –hellouser.class‖ width = 400 height = 400>
<PARAM NAME = "username" VALUE = abc>
</Applet>
</HTML>
```

5.2. Graphics Programming

Graphics can be drawn with the help of java. java applets are written to draw lines, figures of different shapes, images and text in different styles even with the colours in display.

Every applet has its own area of the screen known as canvas, where it creates the display in the area specified the size of applet‘s space is decided by the attributes of <APPLET...> tag.

A java applet draws graphical image inside its space using the coordinate system shown in following fig., which shows java‘s coordinate system has the origin (0, 0) in the upper-left corner, positive x values are to be right, and positive y values are to be the bottom. The values of coordinates x and y are in pixels.



Q. Write a simple applet which display message „Welcome to Java‟. [W-15]

Program:

```
import java. applet.*;
import java.awt.*;

public class Welcome extends Applet
   {
   public void paint( Graphics g)
            {
             g.drawString(―Welcome to java‖,25,50);
            }
   }

/*<applet code= WelcomeJava width= 300 height=300>
</applet>*/
```

Step to run an Applet

1. Write a java applet code and save it with as a class name declared in a program by extension as a .java.
   e.g. from above java code file we can save as a **Welcome.java**

2. Compile the java file in command prompt jdk as shown below
   C:\java\jdk1.7.0\bin> javac Welcome.java

3. After successfully compiling java file, it will create the .class file, e.g Welcome.class. then we have to write applet code to add this class into applet.

4. Applet code

```
<html>
<Applet code= ― Welcome.class‖ width= 500 height=500>
</applet>
</html>
```

5. Save this file with Welcome.html in _bin‘ library folder.

6. Now write the following steps in command prompt jdk.

   C:\java\jdk1.7.0\bin> appletviewer Welcome.java
   C:\java\jdk1.7.0\bin> appletviewer Welcome.html
   (Shows output in applet viewer)
   OR
   C:\java\jdk1.7.0\bin> Welcome.html
   (Shows output in internet browser)

5.2.1. Graphics Class

The Graphics class of java includes methods for drawing different types of shapes, from simple lines to polygons to text in a variety of fonts.

The paint( ) method and a Graphics object is used to display text. To draw shapes, drawing methods in Graphics class is used which arguments representing end points, corners, or starting locations of a shape as a values in the applet‘s coordinate system.

Method	Description
clearRect( )	Erases a rectangular area of the canvas
copyArea( )	Copies a rectangular area of the canvas to another area
drawArc( )	Draws a hollow arc.
drawLine( )	Draws a straight line
drawOval( )	Draws a hollow oval
drawPolygon( )	Draws a hollow polygon
drawRect( )	Draws a hollow rectangle
drawRoundRect( )	Draws a hollow rectangle with rounded corners.
drawstring( )	Displays a text string
fillArc( )	Draws a filled arc
fillOval( )	Draws a filled arc
fillPolygon( )	Draws a filled polygon
fillRect( )	Draws a filled rectangle
fillRoundRect( )	Draws filled rectangle with rounded corners
getColor( )	Retrieves the current drawing color
getFont( )	Retrieves the currently used font
getFontMetrics( )	Retrieves information about the current font.
setColor( )	Sets the drawing color
setFont( )	Seta fonts.

5.2.2. drawString( ) [S-15]

Displaying String:
drawString() method is used to display the string in an applet window

Syntax:

void drawString(String message, int x, int y);
where message is the string to be displayed beginning at x, y

Example:

g.drawString(–WELCOME‖, 10, 10);

5.2.3. Lines and Rectangle.

5.2.3.1. drawLine( )

The drawLine ( ) method is used to draw line which takes two pair of coordinates (x1,y1) and (x2, y2) as arguments and draws a line between them. The graphics object g is passed to paint( ) method.

The syntax is

```
g.drawLine(x1,y1,x2,y2);
```

e.g. g.drawLine(20,20,80,80);

5.2.3.2. drawRect( ) [W-14, S-15,W-15, S-16]

The drawRect() method display an outlined rectangle

Syntax: void drawRect(int top, int left, int width, int height)

This method takes four arguments, the first two represents the x and y co- ordinates of the top left corner of the rectangle and the remaining two represent the width and height of rectangle.

Example: g.drawRect(10,10,60,50);

Q. Design an Applet program which displays a rectangle filled with red color and message as “Hello Third year Students” in blue color. [S-16]

Program-

```
import java.awt.*;
import  java.applet.*;

public class DrawRectangle extends Applet
        {
        public void paint(Graphics g)
                {
                        g.setColor(Color.red);
                        g.fillRect(10,60,40,30);

                        g.setColor(Color.blue);
                        g.drawString("Hello Third year Students",70,100);
                }
        }


/* <applet code="DrawRectangle.class" width="350" height="300"> </applet> */
```

5.2.4. Circle and Ellipse

5.2.4.1. drawOval( ) [W-14, S-15, W-15, S-16]

To draw an Ellipses or circles used drawOval() method can be used.

Syntax: void drawOval(int top, int left, int width, int height)

The ellipse is drawn within a bounding rectangle whose upper-left corner is specified by top and left and whose width and height are specified by width and height to draw a circle or filled circle, specify the same width and height the following program draws several ellipses and circle.

Example: g.drawOval(10,10,50,50);

5.2.4.2. fillOval ( ) [W-14]

Draws an oval within a bounding rectangle whose upper left corner is specified by top, left. Width and height of the oval are specified by width and height.

Syntax- **void fillOval(int top, int left, int width, int height):**

Example g.fillOval(10,10,50,50);

Q. Write a simple applet program which display three concentric circle. [S-16]

Program-

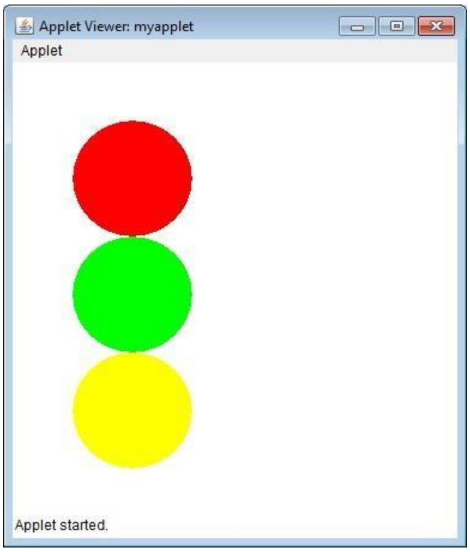
```
import java.awt.*;
import java.applet.*;

public class CircleDemo extends Applet
        {
            public void paint (Graphics g)
                    {

                    g.drawOval(100,100,190,190);
                    g.drawOval(115,115,160,160);
                    g.drawOval(130,130,130,130);
                    }
            }


/*<applet code=‖CircleDemo.class‖ height=300 width=200>
 </applet>*/

(OR)

HTML Source:
<html> <applet code=‖CircleDemo.class‖ height=300 width=200>
</applet>
</html>
```

Q. Write a program to design an applet to display three circles filled with three different colors on screen. [W-14, W-15]

Program-

```
import java.awt.*;
import  java.applet.*;

public class myapplet extends Applet
        {
                public void paint(Graphics g)
                { g.setColor(Color.red);
                g.fillOval(50,50,100,100);

                g.setColor(Color.green);
                g.fillOval(50,150,100,100);

                g.setColor(Color.yellow);
                g.fillOval(50,250,100,100);

                }
        }

/*<applet code=myapplet width= 300 height=300>
</applet>*/
```

Output



5.2.5. Drawing Arcs

5.2.5.1. drawArc( ) [S-15, W-15]

It is used to draw arc

Syntax:

```
void drawArc(int x, int y, int w, int h, int  start_angle, int  sweep_angle);
```

where x, y starting point, w& h are width and height of arc, and start_angle is starting angle of arc sweep_angle is degree around the arc

Example:

```
g.drawArc(10, 10, 30, 40, 40, 90);
```

5.2.6. Drawing polygons
5.2.6.1. drawPolygon( ) [W-14, W-15]

drawPolygon() method is used to draw arbitrarily shaped figures.

Syntax- void drawPolygon(int[ ] xPoints, int[ ] yPoints, int numPoints):

The polygon"s end points are specified by the co-ordinates pairs contained within the x and y arrays. The number of points define by x and y is specified by numPoints.

Example-

```
int  x[ ] = {10, 170, 80};
int  y[ ] = {20, 40, 140};
int n = 3;

g.drawPolygon(x, y, n);
```

Q. Write the syntax and example for each of following graphics methods:

1) drawPoly ( ) 2) drawRect ( ) 3) drawOval ( ) 4) fillOval ( )

For syntax refer above 5.2.3.2 and all........

Example for including all methods in a one program

```
import java.applet.*;
import java.awt.*;

public class DrawGraphics extends Applet
{
        public void paint(Graphics g)
        {
        int x[ ] = {10, 170, 80};
        int y[ ] = {20, 40, 140};
        int n = 3;
        g.drawPolygon(x, y, n);
        g.drawRect(10, 150,100, 80);
        g.drawOval(10, 250, 100, 80);
        g.fillOval(10, 350, 100, 80);
        }
}

/*
<applet code = DrawGraphics.class height = 500 width = 400>
</applet>*/
```

5.2.7. Setting color of an Applet

Background and foreground color of an applet can be set by using followings methods

void setBackground(Color.newColor)

void setForeground (Color.newColor)

where newColor specifies the new color. The class color defines the constant for specific color listed below.

Color.black	Color.white	Color.pink	Color.yellow
Color.lightGray	Color.gray	Color.darkGray	Color.red
Color.green	Color.magenda	Color.orange	Color.cyan

Example

```
setBackground(Color.red);

setForeground (Color.yellow);
```

The following methods are used to retrieve the current background and foreground color.

```
Color getBackground( )
Color  getForeground( )
```

5.2.8. Font class

A font determines look of the text when it is painted. Font is used while painting text on a graphics context & is a property of AWT component.

The Font class defines these variables:

Variable	**Meaning**
String name	Name of the font
float pointSize	Size of the font in points
int size	Size of the font in point
int style	Font style

5.2.8.1. Use of font class [W-14, S-15]

The Font class states fonts, which are used to render text in a visible way. It is used to set or retrieve the screen font.

Syntax to create an object of Font class. [W-14]

To select a new font, you must first construct a Font object that describes that font. Font constructor has this general form:

Font(String fontName, int fontStyle, int pointSize)

fontName specifies the name of the desired font. The name can be specified using either the logical or face name.

All Java environments will support the following fonts:

Dialog, DialogInput, Sans Serif, Serif, Monospaced, and Symbol. Dialog is the font used by once system‟s dialog boxes.

Dialog is also the default if you don‟t explicitly set a font. You can also use any other fonts supported by particular environment, but be careful—these other fonts may not be universally available.

The style of the font is specified by fontStyle. It may consist of one or more of these three constants:

Font.PLAIN, Font.BOLD, and Font.ITALIC. To combine styles, OR them together.

For example,

Font.BOLD | Font.ITALIC specifies a bold, italics style.

The size, in points, of the font is specified by pointSize.

To use a font that you have created, you must select it using setFont( ), which is defined by Component.

It has this general form:

void setFont(Font fontObj)

Here, fontObj is the object that contains the desired font

5.2.8.2. Methods of font class

Q. Describe any three methods of font class with their syntax and example of each. [W-14, S-15]

Sr. No	Methods	Description
1	static Font decode(String *str*)	Returns a font given its name.
2	boolean equals(Object *FontObj*) :	Returns **true** if the invoking object contains the same font as that specified by *FontObj*.Otherwise, it returns **false**.
3	String toString( )	Returns the string equivalent of the invoking font.
4	String getFamily( )	Returns the name of the font family to which the invoking font belongs.
5	static Font getFont(String *property*)	Returns the font associated with the system property specified by *property*. **null** is returned if *property* does not exist.
6	static Font getFont(String *property*,Font *defaultFont*)	Returns the font associated with the System property specified by *property*. The font specified by *defaultFont* is returned if *property* does not exist.
7	String getFontName( )	Returns the face name of the invoking font.
8	String getName( )	Returns the logical name of the invoking font.

9	int getSize( )	Returns the size, in points, of the invoking font.
10	int getStyle( )	Returns the style values of the invoking font.
11	int hashCode( )	Returns the hash code associated with the invoking object.
12	boolean isBold( )	Returns **true** if the font includes the **BOLD** style value. Otherwise, **false** is returned.
13	boolean isItalic( )	Returns **true** if the font includes the **ITALIC** style value. Otherwise, **false** is returned.
14	boolean isPlain( )	Returns **true** if the font includes the **PLAIN** style value. Otherwise, **false** is returned.

Example:-

//program using equals method

```
import java.awt.*;
import java.applet.*;

public class ss extends Applet
	{
	public void paint(Graphics g)
		{
		Font a = new Font ("TimesRoman", Font.PLAIN, 10);
		Font b = new Font ("TimesRoman", Font.PLAIN, 10);

		// displays true since the objects have equivalent settings
		g.drawString(""+a.equals(b),30,60);
		}
	}

/*<applet code=‖ss.class‖ height=200 width=200>
</applet>*/
```

// program using getFontName,getFamily(),getSize(),getStyle(),.getName()

```
import java.awt.*;
import java.applet.*;

public class font1 extends Applet
	{
	Font f, f1;
	String s, msg;
	String fname;
	String ffamily;
	int size;
	int style;
	public void init()
		{
```

```
        f= new Font("times new roman",Font.ITALIC,20);
        setFont(f);
        msg="is interesting";
        s="java programming";
        fname=f.getFontName();
        ffamily=f.getFamily();
        size=f.getSize();
        style=f.getStyle();
        String f1=f.getName();
        }
    public void paint(Graphics g)
        {
        g.drawString("font name"+fname,60,44);
        g.drawString("font family"+ffamily,60,77);
        g.drawString("font size "+size,60,99);
        g.drawString("fontstyle "+style,60,150);
        g.drawString("fontname "+f1,60,190);
        }
    }

/*<applet code=font1.class height=300 width=300>
</applet>*/
```

Q. Write method to set font of a text and describe its parameters. [S-16]

The AWT supports multiple type fonts emerged from the domain of traditional type setting to become an important part of computer-generated documents and displays. The AWT provides flexibility by abstracting font-manipulation operations and allowing for dynamic selection of fonts.

Fonts have a family name, a logical font name, and a face name. The family name is the general name of the font, such as Courier. The logical name specifies a category of font, such as Monospaced. The face name specifies a specific font, such as Courier Italic To select a new font, you must first construct a Font object that describes that font.

One Font constructor has this general form:

Font(String fontName, intfontStyle, intpointSize)

To use a font that you have created, you must select it using setFont( ), which is defined by Component.

It has this general form:

void setFont(Font fontObj)

Example

```
import java.applet.*;
import  java.awt.*;
import  java.awt.event.*;
public class SampleFonts extends Applet
    {
    int next = 0;
    Font f;
    String msg;
```

```
public void init( )
        {
        f = new Font("Dialog", Font.PLAIN, 12);
        msg = "Dialog";
        setFont(f);
public void paint(Graphics g)
        {
        g.drawString(msg, 4, 20);
        }
}
```

Q. State purpose of get Available Font Family Name ( ) method of graphics environment class.

Purpose of getAvailableFontFamilyName() method:

It returns an array of String containing the names of all font families in this Graphics Environment localized for the specified locale

Syntax:

public abstract String[ ] getAvailableFontFamilyNames(Locale 1)

Parameters:

l - a Locale object that represents a particular geographical, political, or cultural region. Specifying null is equivalent to specifying Locale.getDefault().

Or

String[ ] getAvailableFontFamilyNames( )

It will return an array of strings that contains the names of the available font families

Important Questions:-

4 Marks Questions:-

1) Write syntax and example of 1) drawString ( ) 2) drawRect ( ) ; 3) drawOval ( ) 4) drawArc ( ).
2) Describe following states of applet life cycle : a) Initialization state. b) Running state. c) Display state
3) State the use of font class. Describe any three methods of font class with their syntax and example of each.
4) Differentiate applet and application with any four points.
5) State syntax and explain it with parameters for : i)drawRect ( ) ii) drawOral ( )
6) Design an Applet program which displays a rectangle filled with red color and message as –Hello Third year Students‖ in blue color.
7) Describe applet life cycle with suitable diagram.
8) Differentiate between applet and application (any 4 points).
9) Write a program to design an applet to display three circles filled with three different colors on screen.
10) Explain all attributes available in < applet > tag.

6 & 8 Marks Questions:-

1) Explain <PARAM> Tag of applet with suitable example.
2) State the use of font class. Describe any three methods of font class with their syntax and example of each.
3) Write a simple applet program which display three concentric circle.
4) Write method to set font of a text and describe its parameters.
5) Explain <applet> tag with its major attributes only. State purpose of get Available Font Family Name ( ) method of graphics environment class.
6) Design an applet which displays three circles one below the other and fill them red, green and yellow color respectively.
7) Write the syntax and example for each of following graphics methods : 1) drawPoly( ) 2) drawRect ( ) 3) drawOval ( ) 4) fillOval ( )
8) State the use of Font class. Write syntax to create an object of Font class.
9) Describe any 3 methods of Font class with their syntax and example of each.
10) Write syntax and example of following Graphics class methods : (i) drawOval( ) (ii) drawPolygon( ) (iii) drawArc( ) (iv) drawRect( )
11) Differentiate between applet and application and also write a simple applet which display message ‗Welcome to Java'.
12) How can parameters be passed to an applet ? Write an applet to accept user name in the form of parameter and print ‗Hello < username >'.

Topic 6- File I/O & collection frame work

Total Marks- 12

6.1. Stream Classes

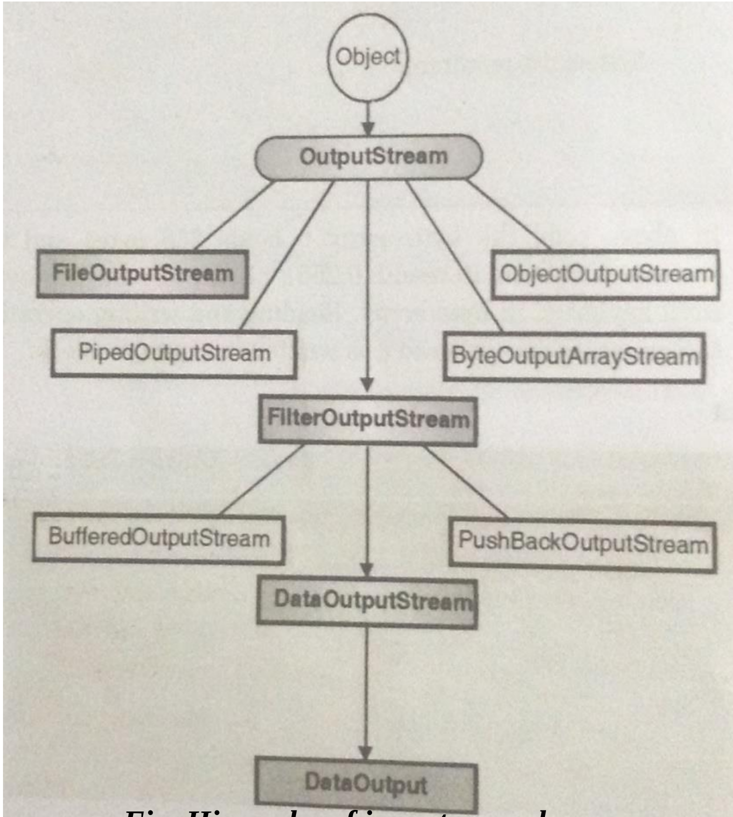


Fig. Hierarchy of java stream classes

1. What are stream classes ? List any two input stream classes from character stream [S-15, S-16]

Definition:

The java. IO package contain a large number of stream classes that provide capabilities for processing all types of data. These classes may be categorized into two groups based on the data type on which they operate.

1. Byte stream classes that provide support for handling I/O operations on bytes.
2. Character stream classes that provide support for managing I/O operations on characters.

Character Stream Class can be used to read and write 16-bit Unicode characters. There are two kinds of character stream classes, namely, reader stream classes and writer stream classes

Reader stream classes:-

It is used to read characters from files. These classes are functionally similar to the input stream classes, except input streams use bytes as their fundamental unit of information while reader streams use characters

Input Stream Classes

1. BufferedReader
2. CharArrayReader
3. InputStreamReader
4. FileReader
5. PushbackReader
6. FilterReader
7. PipeReader
8. StringReader

2. What are streams ? Write any two methods of character stream classes. [W-15]

Java programs perform I/O through streams. A stream is an abstraction that either produces or consumes information (i.e it takes the input or gives the output). A stream is linked to a physical device by the Java I/O system.

All streams behave in the same manner, even if the actual physical devices to which they are linked differ. Thus, the same I/O classes and methods can be applied to any type of device.

Java 2 defines two types of streams: byte and character.

Byte streams provide a convenient means for handling input and output of bytes. Byte streams are used, for example, when reading or writing binary data.

Character streams provide a convenient means for handling input and output of characters.

They use Unicode and, therefore, can be internationalized. Also, in some cases, character streams are more efficient than byte streams.

The Character Stream Classes

Character streams are defined by using two class hierarchies. At the top are two abstract classes, **Reader** and **Writer**. These abstract classes handle Unicode character streams. Java has several concrete subclasses of each of these.

Methods of Reader Class

1) void mark(int *numChars*) : Places a mark at the current point in the input stream that will remain valid until numChars characters are read.

2) boolean markSupported( ) : Returns **true** if **mark( )** / **reset( )** are supported on this stream.

3) int read( ) : Returns an integer representation of the next available character from the invoking input stream. –1 is returned when the end of the file is encountered.

4) int read(char *buffer*[ ]) : Attempts to read up to buffer. Length characters into buffer and returns the actual number of characters that were successfully read. –1 is returned when the end of the file is encountered.

5) abstract int read(char *buffer*[ ],int *offset*,int *numChars*): Attempts to read up to numChars characters into buffer starting at buffer[offset], returning the number of characters successfully read.–1 is returned when the end of the file is encountered.

6) boolean ready( ): Returns **true** if the next input request will not wait. Otherwise, it returns **false**.

7) void reset( ): Resets the input pointer to the previously set mark.

8) long skip(long *numChars*) :- Skips over *numChars* characters of input, returning the number of characters actually skipped.

9) abstract void close( ) :- Closes the input source. Further read attempts will generate an IOException

Writer Class

Writer is an abstract class that defines streaming character output. All of the methods in this class return a **void** value and throw an **IOException** in the case of error

Methods of Writer class are listed below: -

1) abstract void close( ) : Closes the output stream. Further write attempts will generate an **IOException**.

2) abstract void flush( ) : Finalizes the output state so that any buffers are cleared. That is, it flushes the output buffers.

3) void write(int*ch*): Writes a single character to the invoking output stream. Note that the parameter is an **int**, which allows you to call **write** with expressions without having to cast them back to **char**.

4) void write(char *buffer*[ ]): Writes a complete array of characters to the invoking output stream

5) abstract void write(char *buffer*[ ],int *offset*, int *numChars*) :- Writes a subrange of *numChars* characters from the array *buffer*, beginning at *buffer*[*offset*] to the invoking output stream.

6) void write(String *str*): Writes *str* to the invoking output stream.

7) **void write(String *str*, int *offset*,int *numChars*):** Writes a sub range of *numChars* characters from the array *str*, beginning at the specified *offset*.

[Note: any two methods from above list to be considered]**

3. **What is use of stream classes ? Write any two methods FileReader class.[W-14]**

An I/O Stream represents an input source or an output destination.

A stream can represent many different kinds of sources and destinations, including disk files, devices, other programs, and memory arrays.

Streams support many different kinds of data, including simple bytes, primitive data types, localized characters, and objects.

Some streams simply pass on data; others manipulate and transform the data in useful ways. Java"s stream based I/O is built upon four abstract classes: InputStream, OutputStream, Reader, Writer.

They are used to create several concrete stream subclasses, the top level classes define the basic functionality common to all stream classes.

InputStream and OutputStream are designed for byte streams and used to work with bytes or other binary objects.

Reader and Writer are designed for character streams and used to work with character or string.

1. public int read()throws IOException - Reads a single character.

2. public int read(char[ ] cbuf, int offset, int length) throws IOException - Reads characters into a portion of an array.

3. public void close()throws IOException - Closes the stream and releases any system resources associated with it. Once the stream has been closed, further read(), ready(), mark(), reset(), or skip() invocations will throw an IOException. Closing a previously closed stream has no effect

4. public boolean ready()throws IOException - Tells whether this stream is ready to be read. An InputStreamReader is ready if its input buffer is not empty, or if bytes are available to be read from the underlying byte stream

5. public void mark(int readAheadLimit) throws IOException -Marks the present position in the stream. Subsequent calls to reset() will attempt to reposition the stream to this point. Not all character-input streams support the mark() operation.

6. public void reset()throws IOException - Resets the stream. If the stream has been marked, then attempt to reposition it at the mark. If the stream has not been marked, then attempt to reset it in some way appropriate to the particular stream, for example by repositioning it to its starting point. Not all character-input streams support the reset() operation, and some support reset() without supporting mark().

4. **Write any two methods of File and FileInputStream class each.[S-15, W-15]**

File Class Methods

1. **String getName( )** - returns the name of the file.
2. **String getParent( )** - returns the name of the parent directory.

3. boolean exists( ) - returns true if the file exists, false if it does not.

4. void deleteOnExit( ) -Removes the file associated with the invoking object when the Java Virtual Machine terminates.

5. boolean isHidden( )-Returns true if the invoking file is hidden. Returns false otherwise.

FileInputStream Class Methods:

1. int available( )- Returns the number of bytes of input currently available for reading.
2. void close( )- Closes the input source. Further read attempts will generate an IOException.
3. void mark(int numBytes) -Places a mark at the current point in the inputstream that will remain valid until numBytes bytes are read.
4. boolean markSupported( ) -Returns true if mark( )/reset( ) are supported by the invoking stream.
5. int read( )- Returns an integer representation of the next available byte of input. –1 is returned when the end of the file is encountered.
6. int read(byte buffer[ ])- Attempts to read up to buffer.length bytes into buffer and returns the actual number of bytes that were successfully read. –1 is returned when the end of the file is encountered.

5. Explain serialization in relation with stream class. [W-14,W-15, S-16]

Serialization is the process of writing the state of an object to a byte stream. This is useful when you want to save the state of your program to a persistent storage area, such as a file. At a later time, you may restore these objects by using the process of deserialization.

Serialization is also needed to implement Remote Method Invocation (RMI). RMI allows a Java object on one machine to invoke a method of a Java object on a different machine. An object may be supplied as an argument to that remote method. The sending machine serializes the object and transmits it. The receiving machine deserializes it.

Example:

Assume that an object to be serialized has references to other objects, which, inturn, have references to still more objects. This set of objects and the relationships among them form a directed graph. There may also be circular references within this object graph. That is, object X may contain a reference to object Y, and object Y may contain a reference back to object X. Objects may also contain references to themselves. The object serialization and deserialization facilities have been designed to work correctly in these scenarios. If you attempt to serialize an object at the top of an object graph, all of the other referenced objects are recursively located and serialized. Similarly, during the process of deserialization, all of these objects and their references are correctly restored.

6. **Write a program to copy contents of one file to another file using character stream class.[S-15]**

```
import java.io.*;
class CopyData
{
        public static void main(String args[ ])
        {

        //Declare input and output file stream

        FileInputStream fis= null; //input stream

        FileOutputStream  fos=null;  //output Stream
        //Declare a variable to hold a byte

        byte byteRead;
         try
        {
        // connect fis to in.dat
        fis=new FileInputSream(–in.dat‖);
        // connect fos to out.dat
        fos= new FileOutputStream(—out.dat‖);
        //reading bytes from in.dat and write to out.dat

        do
        {
        byteRead =(byte)fis.read( );
        fos.write(byteRead);
        }
        while(byteRead != -1);
        }

        Catch(FileNotFoundException e)

        {

        System.out.println(–file not found‖);

        }

        Catch(IOException  e)

        {

        System.out.pritln(e.getMessage( ));

        }

        finally     // close file

        {
```

```
try
{
fis.close( );
fos.close( );
}
Catch(IOException  e)
{  }
}
}
}
```

7. **What is use of ArrayList Class ? State any three methods with their use from ArrayList.[W-15, S-16]**

Use of ArrayList class:

1. ArrayListsupports dynamic arrays that can grow as needed.
2. ArrayListis a variable-length array of object references. That is, an ArrayListcan dynamically increase or decrease in size. Array lists are created with an initial size. When this size is exceeded, the collection is automatically enlarged. When objects are removed, the array may be shrunk.

Methods of ArrayList class :

1. void add(int index, Object element) Inserts the specified element at the specified position index in this list. Throws IndexOutOfBoundsException if the specified index is is out of range (index < 0 || index >size()).

2. boolean add(Object o) Appends the specified element to the end of this list.

3. booleanaddAll(Collection c) Appends all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's iterator. Throws NullPointerException if the specified collection is null.

4. booleanaddAll(int index, Collection c) Inserts all of the elements in the specified collection into this list, starting at the specified position. Throws NullPointerException if the specified collection is null.

5. void clear() Removes all of the elements from this list.

6. Object clone() Returns a shallow copy of this ArrayList.

7. boolean contains(Object o) Returns true if this list contains the specified element. More formally, returns true if and only if this list contains at least one element e such that (o==null ? e==null : o.equals(e)).

8. void ensureCapacity(intminCapacity) Increases the capacity of this ArrayList instance, if necessary, to ensure that it can hold at least the number of elements specified by the minimum capacity argument.

9. Object get(int index) Returns the element at the specified position in this list. Throws IndexOutOfBoundsException if the specified index is is out of range (index <
0 || index >= size()).

10. intindexOf(Object o) Returns the index in this list of the first occurrence of the specified element, or -1 if the List does not contain this element.

11. intlastIndexOf(Object o) Returns the index in this list of the last occurrence of the specified element, or -1 if the list does not contain this element.

12. Object remove(int index) Removes the element at the specified position in this list. Throws IndexOutOfBoundsException if index out of range (index < 0 || index >= size()).

13. protected void removeRange(intfromIndex, inttoIndex) Removes from this List all of the elements whose index is between fromIndex, inclusive and toIndex, exclusive.

14. Object set(int index, Object element) Replaces the element at the specified position in this list with the specified element. Throws IndexOutOfBoundsException if the specified index is is out of range (index < 0 || index >= size()).

15. int size() Returns the number of elements in this list.

16. Object[ ] toArray() Returns an array containing all of the elements in this list in the correct order. Throws NullPointerException if the specified array is null.

17. Object [] toArray(Object[] a) Returns an array containing all of the elements in this list in the correct order; the runtime type of the returned array is that of the specified array.

18. void trimToSize() Trims the capacity of this ArrayList instance to be the list's current size.

8. Write syntax and function of following methods of Date class :
i) getTime ( ) ii) getDate ( ) [S-15]

The **Date** class encapsulates the current date and time.

i. getTime():

Syntax:

long getTime( )

Returns the number of milliseconds that have elapsed since January 1, 1970.

ii. getDate()

Syntax:

public int getDate()

Returns the day of the month. This method assigns days with the values of 1 to 31.

9. Write syntax and function of following methods of date class :
1) sethme ( ) 2) getDay ( ) [W-14]

1. setTime():

void setTime(long time):
the parameter time - the number of milliseconds.
Sets this Date object to represent a point in time that is time milliseconds after January 1, 1970 00:00:00 GMT

2.getDay()

int getDay():
Returns the day of the week represented by this date.
The returned value (0 = Sunday, 1 = Monday, 2 = Tuesday, 3 = Wednesday, 4 = Thursday, 5 = Friday, 6 = Saturday) represents the day of the week that contains or begins with the instant in time represented by this Date object, as interpreted in the local time zone.

10. Write any four mathematical functions used in Java.[W-14]

1) min() :
Syntax: static int min(int a, int b)
Use: This method returns the smaller of two int values.

2) max() :
Syntax: static int max(int a, int b)
Use: This method returns the greater of two int values.

3) sqrt()
Syntax: static double sqrt(double a)
Use : This method returns the correctly rounded positive square root of a double

value.

4) pow() :
Syntax: static double pow(double a, double b)
Use : This method returns the value of the first argument raised to the power of the second argument.

5) exp()
Syntax: static double exp(double a)
Use : This method returns Euler's number e raised to the power of a double value.

6) round() :
Syntax: static int round(float a)

Use : This method returns the closest int to the argument.

7) abs()
Syntax: static int abs(int a)
Use : This method returns the absolute value of an int value.

11. What is use of setclass ? Write a program using setclass.[W-14]

The Set interface defines a set. It extends Collection and declares the behavior of a collection that does not allow duplicate elements. Therefore, the add( ) method returns false if an attempt is made to add duplicate elements to a set.

The Set interface contains only methods inherited from Collection and adds the restriction that duplicate elements are prohibited.
Set also adds a stronger contract on the behavior of the equals and hashCode operations, allowing Set instances to be compared meaningfully even if their implementation types differ.

The methods declared by Set are summarized in the following table

Sr.No	**Methods**	**Description**
1	add()	Adds an object to the collection
2	clear()	Removes all objects from the collection
3	contains()	Returns true if a specified object is an element within the collection
4	isEmpty()	Returns true if the collection has no elements
5	iterator()	Returns an Iterator object for the collection which may be used to retrieve an object
6	remove()	Removes a specified object from the collection
7	size()	Returns the number of elements in the collection

Following is the example to explain Set functionality:

```
import java.util.*;
public class SetDemo
        {
        public static void main(String args[])
        {
        int count[] = {34, 22,10,60,30,22};
        Set<Integer> set = new HashSet<Integer>();
        try{
        for(int i = 0; i<5; i++){
        set.add(count[i]);
        }
        System.out.println(set);
        TreeSet sortedSet = new TreeSet<Integer>(set);
        System.out.println("The sorted list is:");
        System.out.println(sortedSet);
System.out.println("The First element of the set is: "+ (Integer)sortedSet.first());

System.out.println("The last element of the set is: "+ (Integer)sortedSet.last());
}
catch(Exception e){}
}
}
```

Executing the program.

[34, 22, 10, 30, 60]
The sorted list is:
[10, 22, 30, 34, 60]
The First element of the set is: 10
The last element of the set is: 60

12. State syntax and describe any two methods of map class.[S-16]

The Map Classes Several classes provide implementations of the map interfaces. A map is an object that stores associations between keys and values, or key/value pairs. Given a key, you can find its value. Both keys and values are objects. The keys must be unique, but the values may be duplicated. Some maps can accept a null key and null values, others cannot.

Methods:

void clear // removes all of the mapping from map

booleancontainsKey(Object key) //Returns true if this map contains a mapping for the specified key.

Boolean conainsValue(Object value)// Returns true if this map maps one or more keys to the specified value

Boolean equals(Object o) //Compares the specified object with this map for equality