Swami Sahajanand College of Computer Science

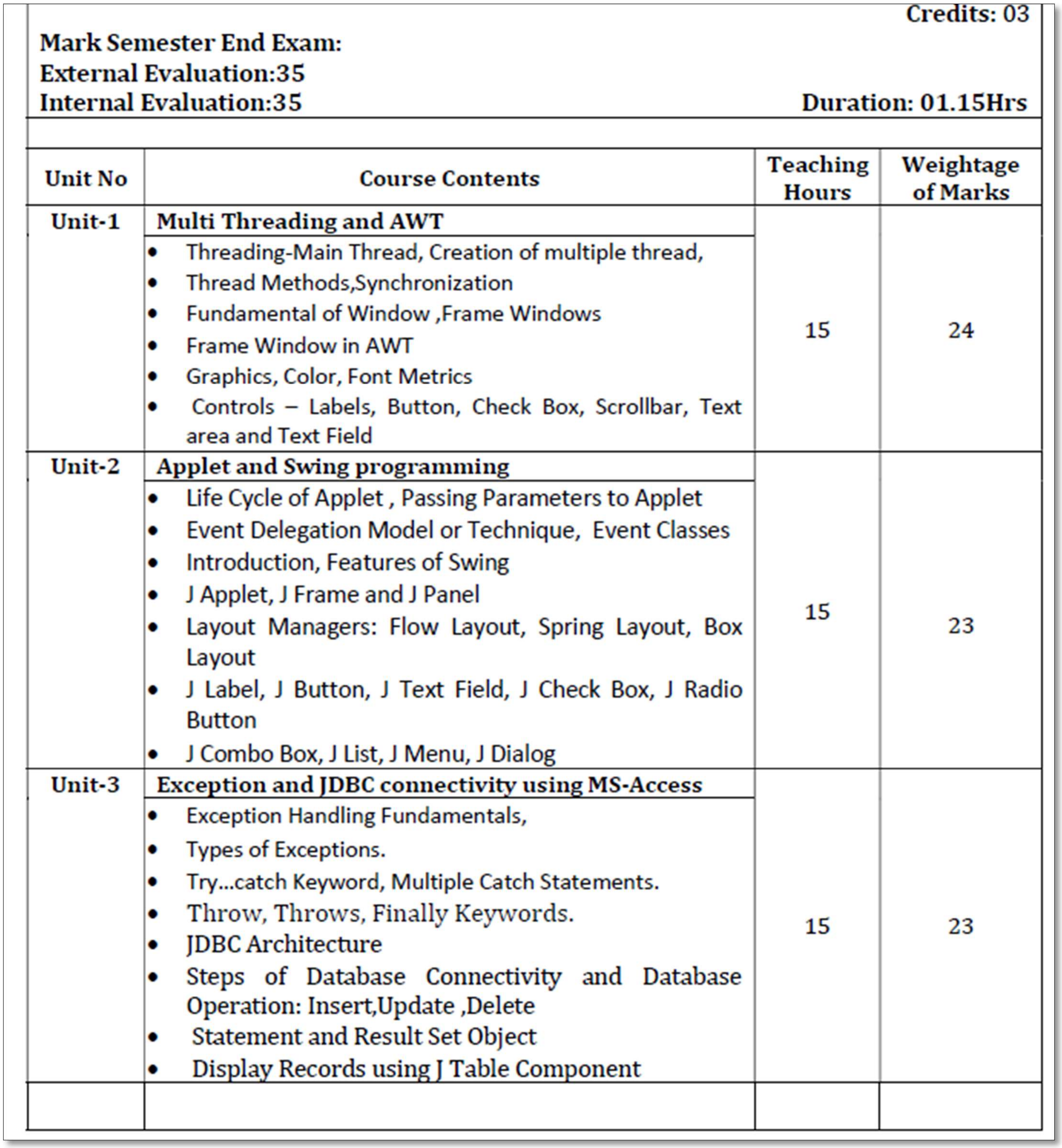
B.C.A. SEM-VI [NEP]

Subject: ADVANCE JAVA
Major (Core) - 27197

UNIT 1

MULTI THREADING AND AWT

- Threading-Main Thread, Creation of multiple threads.
- Thread Methods, Synchronization.
- Fundamental of Window, Frame Windows.
- Frame Window in AWT.
- Graphics, Color, Font Metrics.
- Controls – Labels, Button, Check Box, Scrollbar, Text area and Text Field.



Credits: 03

Mark Semester End Exam:
External Evaluation:35
Internal Evaluation:35 **Duration: 01.15Hrs**

Unit No	Course Contents	Teaching Hours	Weightage of Marks
Unit-1	**Multi Threading and AWT** • Threading-Main Thread, Creation of multiple thread, • Thread Methods,Synchronization • Fundamental of Window ,Frame Windows • Frame Window in AWT • Graphics, Color, Font Metrics • Controls – Labels, Button, Check Box, Scrollbar, Text area and Text Field	15	24
Unit-2	**Applet and Swing programming** • Life Cycle of Applet , Passing Parameters to Applet • Event Delegation Model or Technique, Event Classes • Introduction, Features of Swing • J Applet, J Frame and J Panel • Layout Managers: Flow Layout, Spring Layout, Box Layout • J Label, J Button, J Text Field, J Check Box, J Radio Button • J Combo Box, J List, J Menu, J Dialog	15	23
Unit-3	**Exception and JDBC connectivity using MS-Access** • Exception Handling Fundamentals, • Types of Exceptions. • Try...catch Keyword, Multiple Catch Statements. • Throw, Throws, Finally Keywords. • JDBC Architecture • Steps of Database Connectivity and Database Operation: Insert,Update ,Delete • Statement and Result Set Object • Display Records using J Table Component	15	23

❖Explain java Thread model.

- When a modern operating system wants to start running a program, it creates a new process.
- A process is a program that is currently executing.
- A thread is a smallest unit of executable code that performs a particular task.
- Each & every process has at least one thread running within it.
- Threads are multiple flows of control within a single program or process.
- Multithreading can allow a single program or process to give the appearance of working on more than one thing at the same time.
- With multithreading, application can handle more than one activity at the same time.
- Sometimes threads are referred to as lightweight processes.
- Most modern operating system allows more than one thread to be running at a time within a process.
- When the Java Virtual Machine is started by the operating system, a new process is created. Within that process, many threads can be spawned (Created).
- A thread is a path of code execution through a program, and each thread has its own local variables, program counter and lifetime.
- There are three parts of thread.

  - Code
    - Code can be shared by multiple threads, independent of data.
    - Two threads shared the same code when they execute code from instances of the same class.
  - Data
    - Data may or may not be shared by multiple threads, independent of code.
    - Two threads share the same data when they share access to a common object.
  - CPU
    - In Java programming, the virtual CPU is encapsulated in an instance of thread class.
    - When a thread is constructed, the code and the data are specified by the object passed to its constructor.

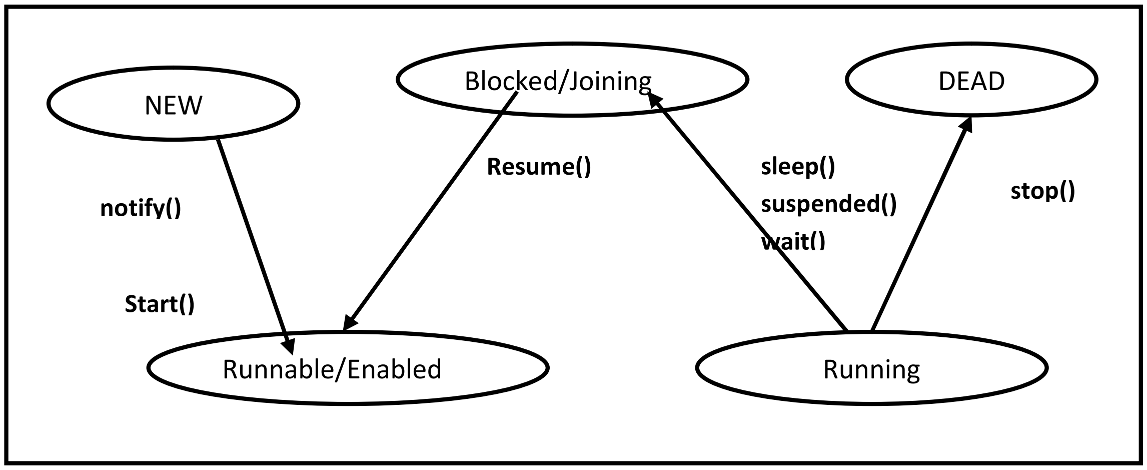
What are Thread class and the Runnable interface?

- In java we can achieve multithreading using Thread class, its methods and its companion interface Runnable.
- Thread encapsulates a thread of execution.
- To create a new thread, our program will either extend Thread class or implement the Runnable interface.
- Thread class defines several methods that help manage threads.

Method	Meaning
getName()	Obtain a thread's name
getPriority()	Obtain a thread's priority
isAlive()	Determine if a thread is still running
Join()	Wait for thread to terminate
run()	Entry point for the thread
sleep()	Suspend a thread for a period of time
start()	Start a thread by calling its run method

❖Explain life cycle of thread.

- A thread is a smallest unit of executable code that performs a particular task.
- Every process has at least one thread running within it.
- A thread can exist in several different states throughout its lifetime.
- Following is the pictorial representation of thread life cycle.



- A thread has just created is in a born state.
- The thread does not immediately start running after it is created.
- It waits for a start() method to be called and then it is ready for run state.
- In runnable state you can temporary stop the execution of the thread using sleep().
- Following table explains the possible state transitions of threads.

From State	To State	Reasons of state change
Enabled	Running	The system schedules the thread for execution
Running	Enabled	System preempts the thread and schedules another
	Waiting	The thread executes wait()
	Sleeping	The thread executes sleep()
	Joining	Thread executes join()
	Running	The thread was interrupted
	Dead	The thread exited run() by returning or by throwing an exception
	Sleeping	Enabled due to sleeping period expired
Joining	Enabled	The thread u being joined died, or the join timed out
	Enabled	The thread was interrupted; throws InterruptedException when run
Waiting	Locking	Another thread executed notify() or notifyAll()
	Locking	The wait for the lock on o timed out

❖How we can create thread? Explain in detail.

- We can create a thread by instantiating an object of type Thread.
- Java defines two ways in which this can be accomplished.
- Extending the Thread class
- Implementing the Runnable interface
- Extending the Thread class
- Steps to create a new thread are….
    - Extend Thread class
    - Override the run() method of Thread class in this subclass
    - Create an instance of this new class
    - Invoke the start() method on the instance
- Syntax to extend Thread class

```
class class_name extends Threadc
{
        //body
        public void run()
        {
              //implementation
        }
}
```

- Example:

```
class a extends Thread
{
        public void run()
        {
                For (int i=1;i<=5;i++)
                {
                        System.out.println("New thread:- " + i);
                }
        }
}
Class b extends Thread
{
        public void run()
        {
                For (int j=1;j<=5;j++)
                {
                        System.out.println("New thread:- " + j);
                }
        }
}
        Public static void main(String args[])
        {
                a a1=new a();
                b b1=new b();
                a1.start();
                b1.start();
        }
}
```

❖What is Synchronization?

- While working with multiple threads, it may be possible that more than one thread share the same data at a time.
- For example one thread might try to read the data, when other thread tries to change it.
- It will create an error and data may be corrupted.
- In this case, it is necessary to allow one thread to finish its task completely and then allow next thread to execute.
- This can be done with the help of synchronized () method.
- Synchronized () method allows programmer to control threads that are sharing data.
- Easy way of achieving synchronization is to create synchronized method within a class.
- Synchronized block ensures that a call to a method that is a member of object occurs only after the current thread has successfully entered object's monitor.
- When two or more threads need access to share data, they need some way to ensure that, the data will be used by only one thread at a time.
- Example:

```
class Numbers
{
        synchronized void printNumbers(int start,int end)
        {
                System.out.print("[");
                for(int i=start;i<=end;i++)
                {
                        try
                        {
                                if(i!=end)
                                        System.out.print(i + ",");
                                else
                                        System.out.print(i);
                        Thread.sleep(1000);
                        }
                        catch(InterruptedException e)
                        {
                                System.out.println("Exception Interrupted");
                        }
                }
        System.out.print("]\n");
        }
}
class Printer implements Runnable
{
		int starting,ending;
		Numbers numbers;
		Thread t;
		public Printer(int start,int end,Numbers nums)
		{
			starting = start;
			ending = end;
			numbers = nums;
			t = new Thread(this);
			t.start();
		}
		public void run()
		{
			numbers.printNumbers(starting,ending);
		}
	}
	class SyncMethod
	{
		public static void main(String args[])
		{
			Numbers n = new Numbers();
			Printer p1 = new Printer(1,5,n);
			Printer p2 = new Printer(11,15,n);
			Printer p3 = new Printer(21,25,n);
			try
			{
				p1.t.join();
				p2.t.join();
				p3.t.join();
			}
			catch(InterruptedException e)
			{
				System.out.println("Exception Interrupted");
			}
		}
	}
```

❖AWT Classes

- Full form of AWT IS Abstract Window Toolkit.
- AWT is package in java.
- It has many classes and methods which are used to create and manages windows and its components like button, textbox, checkbox etc in Graphical user interface (GUI).
- AWT classes are used to develop desktop application as well as applets.
- Now a day's Swing classes are used instead of applet class to develop user interface in applet.
- The reason of using swing classes is it provides rich implementation of many common GUI control like textbox, button, checkbox etc.
- However all the many swing class inherits AWT classes?

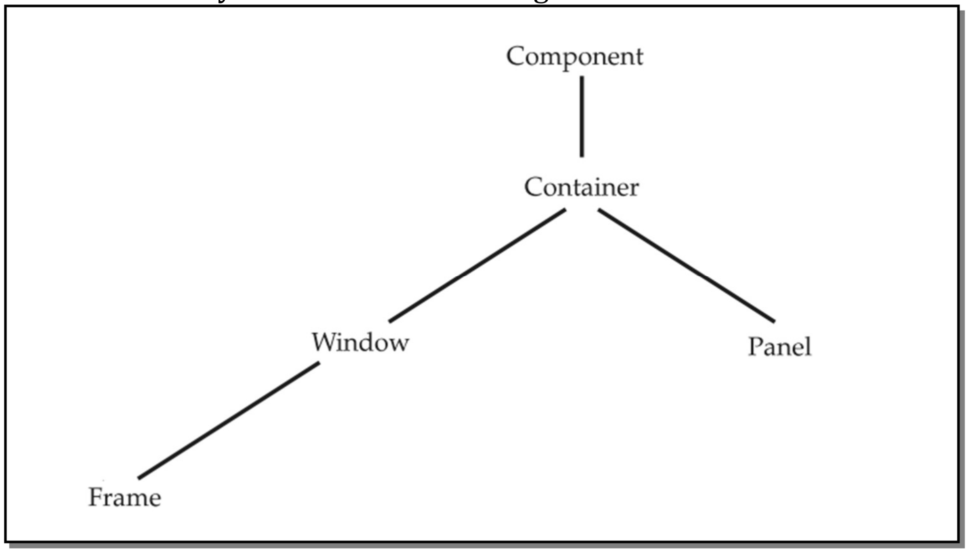
- The AWT classes are contained in the java.awt package. It is one of Java's largest packages.
- All the classes are logically organized in a top-down, hierarchical fashion.
- The following table list some of the commonly used AWT classes.

Class	**Description**
AWTEvent	include AWT events.
BorderLayout	The border layout manager.
Button	Creates a push button control.
Canvas	A blank, semantics-free window.
CardLayout	The card layout manager. Card layouts try to be like index cards. Only the one on top is showing.
Checkbox	Creates a check box control.
CheckboxGroup	Creates a group of check box controls.
CheckboxMenuItem	Creates an on/off menu item.
Choice	Creates a pop-up list.
Color	Manages colors in a por table, platform-independent fashion.
Component	An abstract super-class for various AWT components.
Container	A subclass of Component that can hold other components.
Dialog	Creates a dialog window.
Dimension	Specifies the dimensions of an object. The width is stored in width, and the height is stored in height member variables.
Event	Includes events.
EventQueue	Queues events.
FileDialog	Creates a window from which a file can be selected.
FlowLayout	The flow layout manager. Flow layout positions components left to right, top to bottom.
Font	Includess a type font.
FontMetrics	include various information related to a font. It is used to display text in a window.
Frame	Creates a standard window that has a title bar, resize corners, and a menu bar.

Graphics	Include the graphics context. This context is used by the various output methods to display output in a window.
GridLayout	The grid layout manager. Grid layout displays components in a two-dimensional grid.
Image	Used to show a graphical images.
Insets	Includes the borders of a container.
Label	Creates a label that displays a string.
List	Creates a list from which the user can choose. Similar to the standard Windows list box.
Menu	Creates a pull-down menu.
MenuBar	Creates a menu bar.
MenuComponent	An abstract class implemented by various menu classes.
MenuItem	Creates a menu item.
MenuShortcut	Includes a keyboard shor tcut for a menu item.
Panel	The simplest concrete subclass of Container.
Point	Includes a Car tesian coordinate pair, stored in x and y.
Polygon	Used to draw a polygon.
PopupMenu	Encapsulates a pop-up menu.
Rectangle	Used to draw a rectangle.
Scrollbar	Creates a scroll bar control.
TextArea	Creates a multiline edit control.
TextField	Creates a single-line edit control.

❖Window Fundamentals

- The AWT defines windows according to a class hierarchy, the added functionality and specificity with each level.
- AWT classes can define two types of windows.
- The two most common windows are those derived from Panel, which is used by applets.
- The other type of window is derived from Frame, which creates a standard window in desktop application.
- Functionality of these two types of windows is derived from their parent classes.
- So let us see the hierarchy of these classes in diagrams.



- The Component and Container classes are two of the most important classes in the java.awt package.
- The Component class provides a common superclass for all classes that implement GUI controls.
- The Container class is a subclass of the Component class and can contain other AWT components.
- The Window class is a subclass of the Container class that provides a common set of methods for implementing windows.
- The Window class has two subclasses, Frame and Dialog that are used to create Window objects.
- The Frame class is used to create a main application window, and the Dialog class is used to implement dialog boxes.

Component

- Component class is base class in AWT class hierarchy.
- It is an abstract class that includes all of the attributes of a visual component.
- All user interface elements that are displayed on the screen and that interact with the user are subclasses of Component.
- It defines more than 100 hundred public methods for managing events, such as mouse and keyboard input, positioning and sizing the window, and repainting.
- It also remembers current background and foreground color and font name.

Container

- The Container class is a subclass of Component.
- It has additional methods that allow other Component objects to be nested within it.
- Other Container objects can be stored inside of a Container.

- It allows nested control in windows.
- A container is responsible for positioning any components that it contains.
- It does this through the use of various layout managers

Panel

- The Panel class is a concrete subclass of Container.
- It does not add any new methods.
- It implements Container.
- A Panel may be thought of as a recursively nest able, concrete screen component.
- Panel is the superclass for Applet. When screen output is directed to an applet, it is drawn on the surface of a Panel object.
- A Panel is a window that does not contain a title bar, menu bar, or border. This is why you don't see these items when an applet is run inside a browser.
- When you run an applet using an applet viewer, the applet viewer provides the title and border.
- Other components can be added to a Panel object by its add() method (inherited from Container).
- Once these components have been added, you can position and resize them manually using the setLocation(), setSiize(), or setBounds() methods defined Component.

Window

- The Window class creates top-level window.
- A top-level window is not included within any other object.
- It appears directly on the desktop.
- In normal situation it is not directly created. It is created via its subclass "Frame".

Frame

- Frame class include all the things commonly included in "window."
- It is a subclass of Windows and has a title bar, menu bar, border, and resizing corners.
- If you create Frame object from within an applet, browser display warning message "window has been created" to inform the user that it is a window created by applet not by the operating system.
- This is done to warn the user about possible security threat.

Canvas

- Canvas includes a blank window upon which you can draw.
- It is not part of hierarchy.

❖Frame Windows

- Frame is one of the most commonly required windows after Applet. It create standard window.
- It is created by deriving Frame class.
- It either appears as child window within Applet or top level window or child window for standalone application.
- Frame window has two constructors.
- Frame()
    - This constructor create standard window without any title.
- Frame(String Title)
    - This constructor creates with title specified with its argument.
- One can not specify size of frame window at the time of creating it. It must be given by calling setSize() method.
- Now let us see the important methods of Frame class.

Method Name	**setSize( )**

Overloaded Versions	void setSize(int newWidth, int newHeight) void setSize(Dimension newSize)
Description	• The setSize( )method is used to set the dimensions of the window. • The new size of the window is specified by newWidth and newHeight, or by the width and height fields of the Dimension object passed in newSize. • The dimensions are specified in terms of pixels.
Method Name	**Dimension getSize( )**
Description	• The getSize( ) method is used to obtain the current size of a window • This method returns the current size of the window contained within the width and height fields of a Dimension object.
Method Name	**void setVisible(boolean visibleFlag)**
Description	• It is used to either hide or show frame window. • It has one Boolean argument. If it is true frame window become visiable and if it is false it invisible. • By default it is always invisible.
Method Name	**setTitle(String newTitle)**
Description	• This method is used to change title of the window.

❖Working with Graphics (introduction)

- Like all other programming language such as C, C++; Java also supports graphics.
- Graphics is core part of Graphical user interface in modern operating system.
- Graphics includes foreground & background color, Font size, font name, object like square, round, custom drawing etc.
- In java AWT supports graphics using variety of library functions.
- All these function draws graphics relative to window. It can be applet window, child window of applet or stand alone window.
- Output to a window takes place through a graphics context.
- A graphics context is provided by the Graphics class and is obtained in any one of two following ways:
  - It is passed to an applet when one of its various methods, such as paint( ) or update( ), is called.
  - It is returned by the getGraphics( ) method of Component.
- Graphics class has many methods which can draw object with only outline or filled object.
- Let us see some methods to draw object using graphics class.

Method Name	**void drawLine(int startX, int startY, int endX, int endY)**
Description	• drawLine( ) displays a line in the current drawing color that begins at startX,startY and ends at endX,endY
Method Name	**void drawRect(int top, int left, int width, int height)**
Description	• The drawRect( ) method is used to draw rectangle with border from top , left position to top+left, left+right position.
Method Name	**void fillRect(int top, int left, int width, int height)**
Description	• The fillRect() method is used to draw rectangle with current background color border from top , left position to top+left, left+right position.

Method Name	**void drawRoundRect(int top, int left, int width, int height, int xDiam, int yDiam)**
Description	• the drawRoundRect() method is used to draw rounded rectangle with border from top , left position to top+width, bottom+right position. • The diameter of the rounding arc along the X axis is specified by xDiam. The diameter of the rounding arc along the Y axis is specified by yDiam
Method Name	**void fillRoundRect(int top, int left, int width, int height, int xDiam, int yDiam)**
Description	• the fillRoundRect () method is used to draw rounded rectangle with current background color from top , left position to top+height, left+width position. • The diameter of the rounding arc along the X axis is specified by xDiam. The diameter of the rounding arc along the Y axis is specified by yDiam
Method Name	void drawOval(int top, int left, int width, int height)
Description	• The drawOval() method is used to draw Oval with border from top , left position to top+height, left+width position.
Method Name	**void fillOval(int top, int left, int width, int height)**
Description	• The fillOval () method is used to draw Oval with current background color from top , left position to top+height, left+width position.
Method Name	• **void drawArc(int top, int left, int width, int height, int startAngle,int sweepAngle)**
Description	• drawarc() method is used to draw arc with following description. • The arc is bounded by the rectangle whose upper-left corner is specified by top,left and whose width and height are specified by width and height. • The arc is drawn from startAngle through the angular distance specified by sweepAngle. • Angles are specified in degrees. Zero degrees is on the horizontal, at the three o'clock position. • The arc is drawn counterclockwise if sweepAngle is positive, and clockwise if sweepAngle is negative.
Method Name	• **void fillArc(int top, int left, int width, int height, int startAngle, int sweepAngle)**
Description	• fillArc() method is used to draw arc with current background color with same description given with drawArc() method.

❖Working with Colors

- Java allows using various standard colors to draw object or display text in window.
- Java provides portable and device independent colors.
- Java finds the best match for that color, according to the limits of the display hardware currently executing your program or applet.
- So developer can write their code and use color he wants without warring about hardware device that runs applet code.
- Color is provided by the Color class.

- Java's abstract color model uses 24-bit color, wherein a color is represented as a combination of red, green, and blue values.
- Each component of the color can have a number between 0 and 255.
- 0,0,0 is black, 255,255,255 is white, and Java can represent millions of colors between as well.
- Standard colors are provided using constant defined in color class.

No.	**Color Name**	**RGB Value**
1.	Color.white	255,255,255
2.	Color.black	0,0,0
3.	Color.lightGray	192,192,192
4.	Color.gray	128,128,128
5.	Color.darkGray	64,64,64
6.	Color.red	255,0,0
7.	Color.green	0,255,0
8.	Color.blue	0,0,255
9.	Color.yellow	255,255,0
10.	Color.magenta	255,0,255
11.	Color.cyan	0,255,255
12.	Color.pink	255,175,175
13.	Color.orange	255,200,0

- For example to use red color one can use Color.red constant.
- It is also possible to make custom colors using color class constructor.
- So let us see constructors in detail

Constructor 1	new Color(int,int,int)
Description	• The first constructor takes three integers that specify the color as a mix of red, green, and blue. • These values must be between 0 and 255, • **example: color mycolor= new Color(255, 100, 100);**
Constructor 2	new Color(int rgbValue)
Description	• The second color constructor takes a single integer that contains the mix of red, green, and blue packed into an integer. • The integer is organized with red in bits 16 to 23, green in bits 8 to 15, and blue in bits 0 to 7 • **Example :** • **int newRed = (0xff000000 | (0xc0 << 16) | (0x00 << 8) | 0x00);**

	• **Color darkRed = new Color(newRed);**
Constructor 3	new Color(float,float,float)

Description	• The third constructor, Color(float, float, float), takes three float values (between 0.0 and 1.0) that specify the relative mix of red, green, and blue.
	• Example
	publicvoid paint(Graphics g)
	{
	Color one = new Color(0.99F,0.01F,0.00F); //RED color
	g.setColor(one); // set foreground color
	g.drawString("color code demo ",10,10); // show message
	}

❖Explain Color class method/ Color Methods.

Method Name	**int getRed( )**
Description	• Return the red component of color as integer
Method Name	**int getGreen( )**
Description	• Return the Green component of color as integer
Method Name	**int getBlue( )**
Description	• Return the Blue component of color as integer
Method Name	**int getRGB( )**
Description	• Return the RGB component of color as integer
Method Name	**void setColor(Color newColor)**
Description	• setColor method is used to change foreground color of particular control.
Method Name	**Color getColor()**
Description	• getColor() method is used to obtain current foreground color.

```
/* Random color box using applet */
import java.awt.Graphics;
import java.awt.Color;

public class ColorBox extends java.applet.Applet
```

```
{
 public void paint(Graphics g)
 {
        int rval, gval, bval;
        for (int j = 30; j < (this.getHeight() - 25); j += 30)
        {
                for (int i = 5; i < (this.getWidth() - 25); i += 30)
                {
                        rval = (int) Math.floor(Math.random() * 256);
                        gval = (int) Math.floor(Math.random() * 256);
                        bval = (int) Math.floor(Math.random() * 256);
                        g.setColor(new Color(rval, gval, bval));
                        g.fillRect(i, j, 25, 25);
                        g.setColor(Color.black);
                        g.drawRect(i - 1, j - 1, 25, 25);
                }
        }
 }
}
```

❖Explain Color Mode

- There are two color modes available in java.
- These are given below
  - Hue, Saturation, and Brightness
  - Red, Green, Blue
- One can specify color using any one of the two modes.

Hue, Saturation, and Brightness

- The hue-saturation-brightness (HSB) color model is an alternative to red-green-blue (RGB) for specifying particular colors.
- In this color mode hue is a wheel of color. The hue is specified with a number between 0.0 and 1.0.
- Saturation is another scale ranging from 0.0 to 1.0, representing light pastels to intense hues. Brightness values also range from 0.0 to 1.0, where 1 is bright white and 0 is black.
- HSB color can be converted into RGB combination using below function.
- static int HSBtoRGB(float hue, float saturation, float brightness)

Red, Green and Blue

- The Red, Green, Blue model is standard color model.
- Most of the time color is specified using RGB model.
- In this mode first red specify red color, green specifies green color, and blue specified blue color.
- Range of the color must be between 0 to 255.
- RGB color can be converted ino HSB color using following method.
- static float[ ] RGBtoHSB(int red, int green, int blue, float values[ ].

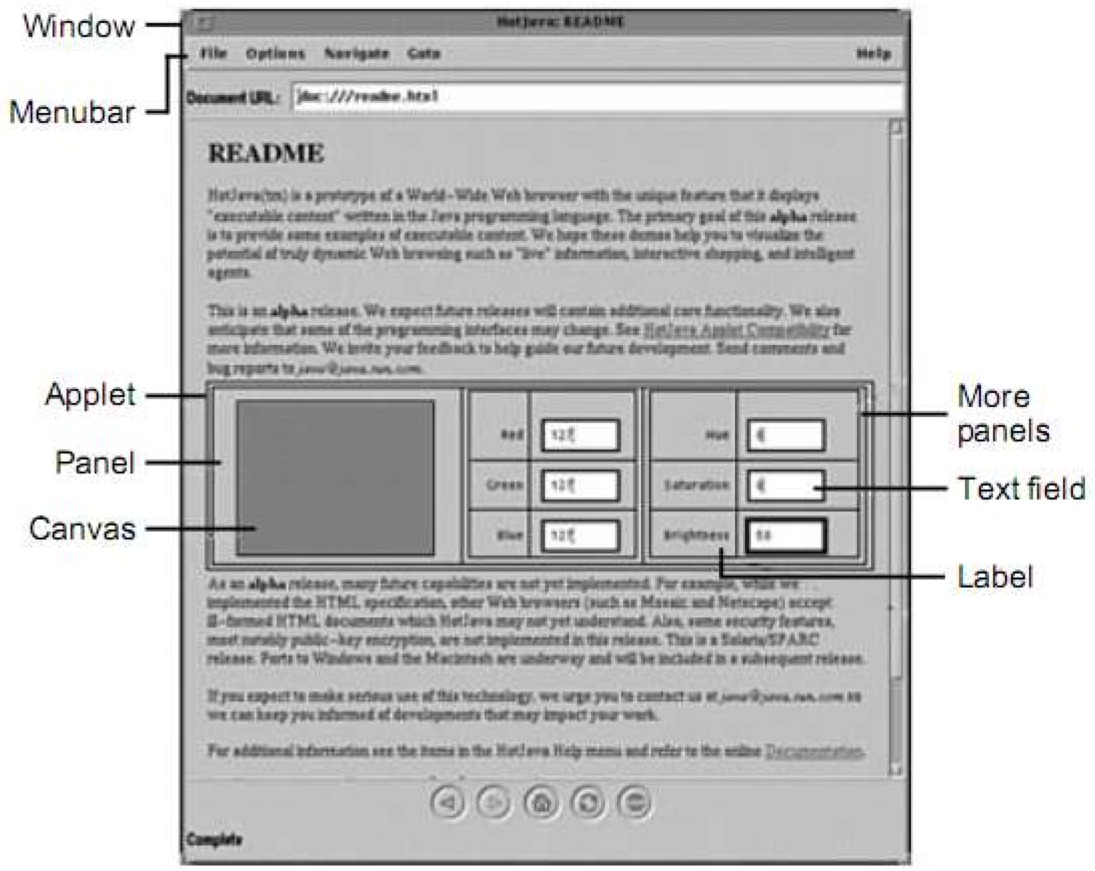
❖Working with Fonts Metrics

- Font means a set of cast metal character form which has fixed size and format.
- Fonts are used to display text or print documents.
- There are various types of font which can be used in java.
- The AWT supports multiple type fonts.
- The AWT provides flexibility by providing 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.
- Fonts are represented by the Font class.
- It has following main methods.

Method	**Description**
boolean equals(Object FontObj)	Returns true if the calling object contains the same font as that specified by FontObj. Otherwise, it returns false.
String getFamily( )	Returns the name of the font family to which the invoking font belongs.
String getFontName()	Returns the face name of the invoking font.
String getName( )	Returns the logical name of the invoking font.
int getSize( )	Returns the size, in points, of the invoking font.
int getStyle( )	Returns the style values of the invoking font.
boolean isBold( )	Returns true if the font includes the BOLD style value. Otherwise, false is returned.
boolean isItalic( )	Returns true if the font includes the ITALIC style value. Otherwise, false is returned.
boolean isPlain( )	Returns true if the font includes the PLAIN style value. Otherwise, false is returned.
String toString( )	Returns the string equivalent of the invoking font.

❖AWT Controls

- The basic concept behind the AWT is that a Java window is a set of nested components, starting from the outermost window all the way down to the smallest UI component (control).
- Abstract Window Toolkit provides various control/components to take input from user or to display output to user.
- Controls are components that allow a user to interact with application in various way.
- Components can include things you can actually see on the screen, such as windows, menu-bars, buttons, and text fields, and they can also include containers, which in turn can contain other components.
- For example, a commonly used control is the push button which is used to start execution of unit of code known as module or method or function. It mainly process inputted data.
- This nesting of components within containers creates a hierarchy of components.
- The hierarchy of components decides the arrangement of items on the screen and inside other items, the order in which they are displayed, and how events are passed from one component to another.
- See the below given figure for better understanding.



❖Type of components in AWT Containers.

- Containers are type of AWT components that can contain other components, as well as other containers.
- The most common form of container is the panel, which represents a container that can be displayed on screen.

- Applets are a form of panel

Canvases.

- A canvas is a simple drawing surface.
- Although you can draw on panels canvases are good for painting images or other graphics operations.

UI components.

- UI components include buttons, lists, simple popup menus, checkboxes, test fields, and other typical elements of a user interface.
- The AWT supports the following types of UI controls/components:
- Labels
    - Text editing
    - Push buttons
    - Check boxes
    - Choice lists
    - Scroll bars
- These controls are subclasses of Component.

How to add controls?

- To display a control in a window, you must add it to the window.
- To do this, you must first create an object / instance of the desired control and then add it to a window by calling add( ), which is defined by Container.
- The add( ) method has many overloaded versions. One commonly used version is
    - Component add(Component compObj)
- Here, compObj is an instance of the control that you want to add.
- A reference to compObj is returned by the add method
- Once a control has been added, it will automatically be visible whenever its parent window is displayed.

How to remove control?

- To remove control from the window one need to call remove() method defined in Container class.
- It has this general form:
    - void remove(Component obj)
- Here, obj is a reference to the control you want to remove.
- You can remove all controls by calling removeAll( ).

❖Labels

- Label is one of the simplest type of control available in AWT.
- It is used to show text message/description about other control in applet window.
- Label's Text string is always read only so end user can not directly change text of Label.
- Labels are passive controls that do not support any interaction with the user.
- There is no event for Label.
- It has mainly three constructor
    - Label( ) throws HeadlessException
    - Label(String str) throws HeadlessException
    - Label(String str, int how) throws HeadlessException
- The first version creates a blank label.
- The second version creates a label that contains the string specified by str. This string is left-justified.
- The third version creates a label that contains the string specified by str using the alignment specified by how.

- The value of how must be one of these three constants: Label.LEFT, Label.RIGHT, or Label.CENTER.
- Let us see one example of label control.

```
// Label Control demo
import java.applet.Applet;
import java.awt.Label;
public class LabelDemo extends Applet
{
 public void init()
 {
        Label lblone = new Label("Swami Vivekanand college of computer science");
        Label lbltwo = new Label("Java Applet",Label.RIGHT);
        Label lblthree = new Label();
        // add labels to applet window
        this.add(lblone);    this.add(lbltwo);                add(lblthree);
        String temp = "text in lblone is " + lblone.getText();
        // getText method is used to retrive text of label
        lblthree.setText(temp);
        //setText method is used to set text of label
 }
}
```

- It has following commonly used methods.

No	Method	Description
1	setText(String str)	This method is used to set or change Text of control.
2	String getText()	This method is used to retrieve current Text of control.
3	void setAlignment(int how)	This method is used to set alignment of Text of control. The value of how must be either LEFT,CENTER,RIGHT
4	Int getAlignment()	This method is used to get current alignment of Text of control.

❖Button

- One of the most used control in applet is Button control.
- Button represents push Button with Text label on it.
- Buttons are simple UI components that trigger some action in your interface when they are pressed.
- For example, a calculator applet might have buttons for each number and operator, or a dialog box might have buttons for “OK” and “Cancel.”
- It has following constructors.
    - Button( ) throws HeadlessException
    - Button(String str) throws HeadlessException
- The first version creates an empty button.
- The second creates a button that contains str as a label.
- It has following commonly used methods.

No	Method	Description

1	setLabel(String str)	This method is used to set Label of push button.
2	String getLabel()	This method is used to get current Label of push button.

How to handle events with Buttons?

- Each time a button is pressed, an action event is generated.
- This is sent to any listeners that previously registered an interest in receiving action event notifications from that button.
- Each listener implements the ActionListener interface.
- That interface has one abstract actionPerformed( ) method, which is called when an event occurs.
- An ActionEvent object is supplied as the argument to this method. It contains a reference to the button that generated the event and a reference to the action command string associated with the button.
- By default, the action command string is the label of the button.
- Usually, either the button reference or the action command string can be used to identify the button.
- Let us see one example.

```
// button control demo
import java.applet.Applet;
import java.awt.Label;
import java.awt.Button;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
public class ButtonDemo extends Applet implements ActionListener
{
 Button btnone,btntwo,btnthree;
 Label lblquestion,lblanswer;
 public void init()
 {
        btnone= new Button("Microsoft");
        btntwo = new Button("Linux");
        btnthree = new Button("Mac OS");
        lblquestion = new Label("Which Operating System You Like? ");
        lblanswer = new Label("");
        this.add(lblquestion);              this.add(btnone);
        this.add(btntwo);                   this.add(btnthree);
        this.add(lblanswer);
        btnone.addActionListener(this); // used to send event to the listener
        btntwo.addActionListener(this);
        btnthree.addActionListener(this);
 }
```

```
        //actionPerformed method is called when user clicks on button
        @Override
        public void actionPerformed(ActionEvent pressed_button)
        {
                String labeltext = pressed_button.getActionCommand();
                //getActionCommand Returns Text Label of pressed button
                if(labeltext.equals("Microsoft"))
                {
                        lblanswer.setText("You Liked Microsoft Operating system");
                }
                else if(labeltext.equals("Linux"))
                {
                        lblanswer.setText("You Liked Linux Operating system");
                }
                else if(labeltext.equals("Mac OS"))
                {
                        lblanswer.setText("You Liked Macnitosh Operating system");
                }
        }
       }
```

❖Checkbox

- A check box is a control that is used to turn an option on or off.
- It consists of a small box that can either contain a check mark or not.
- There is a label associated with each check box that describes what option the box represents. You change the state of a check box by clicking on it.
- Check boxes can be used individually or as part of a group.
- Check boxes are objects of the Checkbox class.
- Checkbox has following constructors:
    - Checkbox( ) throws HeadlessException
    - Checkbox(String str) throws HeadlessException
    - Checkbox(String str, boolean on) throws HeadlessException
- First version creates empty unchecked checkbox.
- Second version creates unchecked checkbox with label specified with str.
- Third version creates checked or unchecked checkbox with label specified with str.State of checkbox is specified by on. It can be true or false.

No	Method	Description
1	setState(Boolean on)	This method is used to change state of checkbox. On can be true or false.
2	Boolean getState()	This method is used to obtain current state of checkbox.
3	setLabel(String str)	This method is used to set Label of push button.
4	String getLabel()	This method is used to get current Label of push button.

How to handle CheckBox Event?

- Each time a check box is selected or deselected, an item event is generated.
- This is sent to any listeners that previously registered an interest in receiving item event notifications from that control.
- Each listener that handle checkbox events, implements the ItemListener interface.
- That interface has one abstract method itemStateChanged( ) method.
- An ItemEvent object is supplied as the argument to this method. It contains information about the event
- Let us see one example.

```
// checkbox control demo
import java.applet.Applet;
import java.awt.Font;
import java.awt.Label;
import java.awt.Button;
import java.awt.Checkbox;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
public class CheckBoxDemo extends Applet implements ActionListener, ItemListener
{
 Button btnremove;
 Label lblsample;
 Checkbox chkbold,chkitalic;
 public void init()
 {
        btnremove= new Button("Remove Formatting");
        lblsample = new Label("Formatting will apply here.... ");
        chkbold = new Checkbox("Bold Effect");
        chkitalic = new Checkbox("Italic Effect");
        this.add(lblsample);                this.add(chkbold);
        this.add(chkitalic);           this.add(btnremove);
        chkbold.addItemListener(this); // used to send the event to the listner
        chkitalic.addItemListener(this);
        btnremove.addActionListener(this);
 }
 //actionPerformed method is called when user clicks on button
 @Override
 public void actionPerformed(ActionEvent pressed_button)
 {
         Font f = new Font("Monospaced",Font.PLAIN,14);
         lblsample.setFont(f);
         chkbold.setState(false);   chkitalic.setState(false);
 }
 //itemStateChanged is called when user click on checkbox
 @Override
 public void itemStateChanged(ItemEvent clicked_box)
 {
         Font f;
         if(chkbold.g void setBlockIncrement(int newIncr) etState()==true &&
chkitalic.getState()==true)
         {
                 f = new Font("Monospaced",Font.BOLD+Font.ITALIC,14);
         }
         else if (chkitalic.getState()==true)
         {
                 f = new Font("Monospaced",Font.ITALIC,14);
         }
         else if (chkbold.getState()==true)
         {
                 f = new Font("Monospaced",Font.BOLD,14);
         }
         else
         {
                f = new Font("Monospaced",Font.PLAIN,14);
         }
         lblsample.setFont(f);
 }
}
```

❖Scroll Bars

- Scroll bars are used to select continuous values between a specified minimum and maximum.
- Scroll bars may be oriented horizontally or vertically.
- A scroll bar is actually a made by several individual parts.
- Each end has an arrow that you can click to move the current value of the scroll bar one unit in the direction of the arrow.

- The current value of the scroll bar relative to its minimum and maximum values is indicated by the slider box (or thumb) for the scroll bar.
- The slider box can be dragged by the user to a new position. The scroll bar will then reflect this value.
- In the background space on either side of the thumb, the user can click to cause the thumb to jump in that direction by some increment larger than 1.
- Usually this action translates into some form of page up and page down.
    - Scroll bars are represented by the Scrollbar class.
    - Scrollbar defines the thee constructors:
    - Scrollbar( ) throws HeadlessException
    - Scrollbar(int style) throws HeadlessException
    - Scrollbar(int style, int initialValue, int thumbSize, int min, int max) throws HeadlessException
- The first version creates a vertical scroll bar.
- The second and third version allow you to specify the orientation of the scroll bar.
- If style is Scrollbar.VERTICAL, a vertical scroll bar is created.
- If style is Scrollbar.HORIZONTAL, the scroll bar is horizontal.
- In the third form of the constructor, the initial value of the scroll bar is passed in initialValue. The number of units represented by the height of the thumb is passed in thumbSize.
- The minimum and maximum values for the scroll bar are specified by min and max.
- If you construct a scroll bar by using one of the first two constructors, then you need to set its parameters by using setValues( ), before it can be used:

No	Method	Description
1	void setValues(int initialValue, int thumbSize, int min, int max)	This method is used to set various values for scrollbar such as initial values, size of scrollbar, minimum and maximum value. It is same as third version of scrollbar constructor
2	Int getvalue()	This method is used to get current value of scrollbar.
3	void setValue(int newValue)	This method is used to set current value of scrollbar specified by newvalue.
4	int getMinimum( )	This method return the minimum value of scrollbar.
5	int getMaximum( )	This method return the maximum value of scrollbar.
6	void setUnitIncrement(int newIncr)	This method is used to set step value of scrollbar. Default step value is 1.
7	void setBlockIncrement(int newIncr)	This method is used to set block increment value of scrollbar which occur when user use page up or page down keys. Default block increment is 10.

How to handle scrollbar events?

- To process scroll bar events, one need to implement the AdjustmentListener interface.
- Each time a user interacts with a scroll bar, an AdjustmentEvent object is generated.
- Its getAdjustmentType( ) method can be used to determine the type of the adjustment.
- Type of adjustment events are following

BLOCK_DECREMENT	A page-down event has been generated.
BLOCK_INCREMENT	A page-up event has been generated.
UNIT_DECREMENT	The line-down button in a scroll bar has been pressed.

UNIT_INCREMENT The line-up button in a scroll bar has been pressed.

```
import java.applet.Applet;
import java.awt.Label;
import java.awt.Scrollbar;
import java.awt.Font;
import java.awt.event.AdjustmentEvent;
import java.awt.event.AdjustmentListener;
public class ScrollBarDemo extends Applet implements AdjustmentListener
{
 Label lblmsg;
 Scrollbar hsb,vsb;
 Font f1;
 public void init()
 {
        lblmsg=new Label("Current value of horizontal scroll bar is :- ");
        vsb=new Scrollbar(); // default constructor
        vsb.setValues(1,10,1,100); // used to set values for scrollbar
        //void setValues(int initialValue, int thumbSize, int min, int max)
        hsb=new Scrollbar(Scrollbar.HORIZONTAL,15,20,10,50);
        this.add(lblmsg);
        this.add(hsb);
        add(vsb);
        hsb.addAdjustmentListener(this); // set Listner for scrollbar
        vsb.addAdjustmentListener(this);
        hsb.setUnitIncrement(5); // set unit increment for scrollbar
        hsb.setBlockIncrement(12); // set block increment for scrollbar
 }
 //adjustmentValueChanged method called when user change value of scrollbar
 @Override
 public void adjustmentValueChanged(AdjustmentEvent e)
 {
        lblmsg.setText("Current value of horizontal scroll bar is :-" + hsb.getValue());
        f1=new Font(Font.SANS_SERIF,Font.BOLD,vsb.getValue());
        lblmsg.setFont(f1);
 }
}
```

❖TextField

- TextField represent TextBox control which is used to accept one-line text from user.
- Text fields allow the user to enter strings and to edit the text using the arrow keys, cut and paste keys, and mouse selections.
- TextField is a subclass of TextComponent.
- TextField has following constructors:
  - TextField( ) throws HeadlessException
  - TextField(int numChars) throws HeadlessException
  - TextField(String str) throws HeadlessException
  - TextField(String str, int numChars) throws HeadlessException
- The first version creates a blank text field.
- The second version creates a text field that is numChars characters wide.
- The third version initializes the text field with the string contained in str.
- The fourth version initializes a text field and sets its width.

No	Method	Description
1	String getText( )	This method is used to obtain current text of textfield.
2	void setText(String str)	This method is used to set text of texfield specified by str.
3	String getSelectedText( )	This method is used to obtain selected text of textfield.
4	void select(int startIndex, int endIndex)	This method is used to obtain string from current text start & ends with starindex and endindex
5	boolean isEditable( )	This method returns true if textfield is editable, otherwise false
6	void setEditable(boolean canEdit)	This method set whether user can modify textfield or not. Argument can be true or false.
7	void setEchoChar(char ch)	This method specifies a single character that the TextField will display when characters are entered so the actual characters typed will not be shown
8	boolean echoCharIsSet( )	This method returns true if echochar is set, otherwise it return false.
9	char getEchoChar( )	This method is used to obtain echo character.

How to handle TextField events?

- Since text fields perform their own editing functions, your program generally will not respond to individual key events that occur within a text field.
- However, one may want to respond when the user presses ENTER.
- When this occurs, an action event is generated.
- Let us see one example

```
// TextField demo
import java.applet.Applet;
import java.awt.Graphics;
import java.awt.Label;
import java.awt.TextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
public class TextFieldDemo extends Applet implements ActionListener
```

```
{
 TextField txtname, txtpass;
 public void init()
 {
        Label lblname = new Label("Name: ", Label.RIGHT);
        Label lblpass = new Label("Password: ", Label.RIGHT);
        txtname = new TextField(12);
        txtpass = new TextField(8);
        txtpass.setEchoChar('?'); // set password character
        add(lblname);       add(txtname);
        add(lblpass);       add(txtpass);
        // register to receive action events
        txtname.addActionListener(this);
        txtpass.addActionListener(this);
 }
 // actionPerformed method called when user press enter key in TextField.
 public void actionPerformed(ActionEvent ae)
 {
        repaint();
 }
 public void paint(Graphics g)
 {
        g.drawString("Name: " + txtname.getText(), 6, 60);
        g.drawString("Selected text in name: "
        + txtname.getSelectedText(), 6, 80);
        g.drawString("Password: " + txtpass.getText(), 6, 100);
 }
}
```

❖TextArea

- TextArea control is used when user needs to given multiline input in applet program.
- To handle these situations, the AWT includes a simple multiline editor called TextArea.
- It has following five constrctor.
    - TextArea( ) throws HeadlessException
    - TextArea(int numLines, int numChars) throws HeadlessException
    - TextArea(String str) throws HeadlessException
    - TextArea(String str, int numLines, int numChars) throws HeadlessException
    - TextArea(String str, int numLines, int numChars, int sBars) throws HeadlessException
- Here, numLines specifies the height, in lines, of the text area, and numChars specifies its width, in characters. Initial text can be specified by str.
- In the fifth form, you can specify the scroll bars that you want the control to have.

- sBars must be one of these values:

SCROLLBARS_BOTH	SCROLLBARS_NONE
SCROLLBARS_HORIZONTAL_ONLY	SCROLLBARS_VERTICAL_ONLY

- First 6 methods that we have seen in TextField control is also applicable to TextArea control as well as it has following additional method.

No	Method	Description
1	void append(String str)	The append( ) method appends the string specified by str to the end of the current text.
2	void insert(String str, int index)	The insert( ) method inserts the string passed in str at the specified index.
3	void replaceRange(String str, int startIndex, int endIndex)	The replaceRange() method replaces the characters fromstartIndex to endIndex–1, with the replacement text passed in str.

- Text areas only generate got-focus and lost-focus events. Normally, your program simply obtains the current text when it is needed.

//TextField control demo

```
import java.applet.Applet;
import java.awt.TextArea;
public class TextAreaDemo extends Applet
{
 public void init()
 {
         TextArea txtaddress = new TextArea(10,50);
         String address = "ABC Corporation, \n kunj villa, jawahar nagar, new delhi \n
phone no :- 9999888899";
         txtaddress.setText(address);
         this.add(txtaddress);

 }
}
```