

GUI Programming in JAVA

Course: CSL304

Prof. Juhi Ganwani

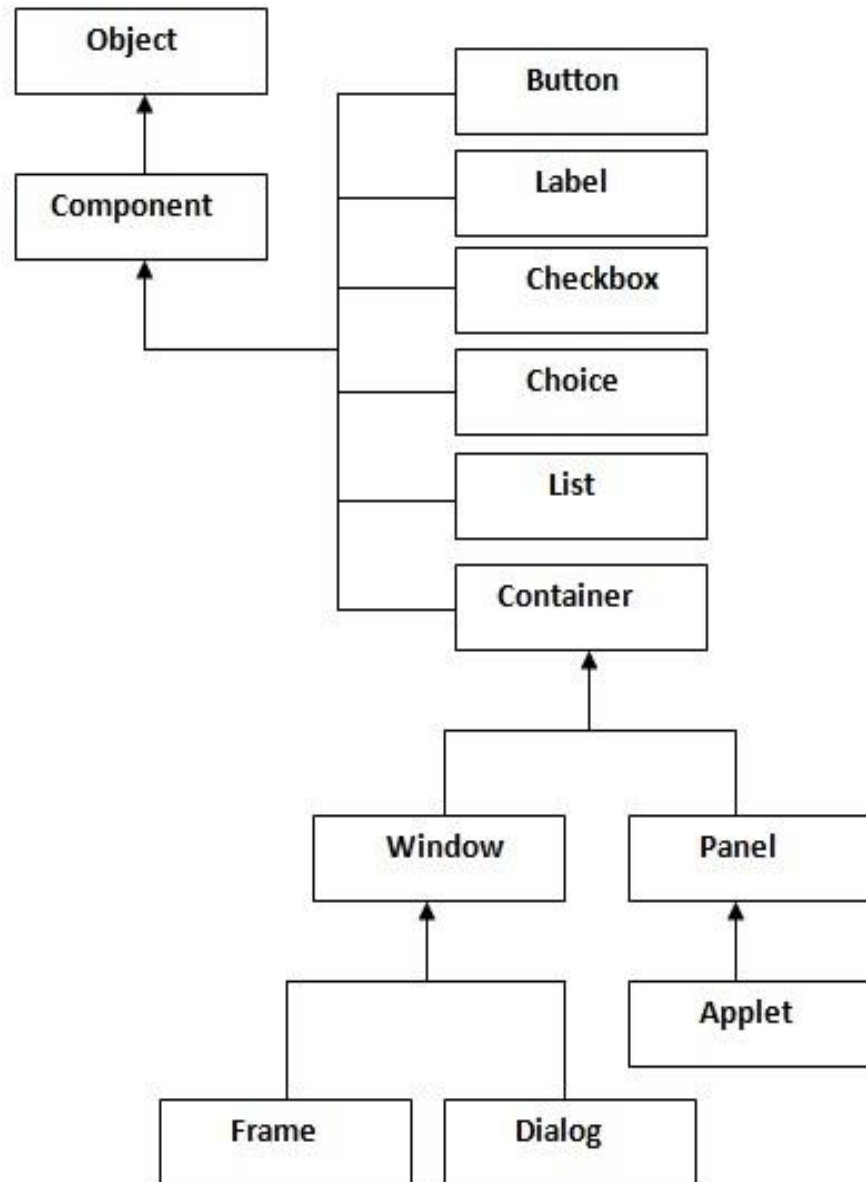
CONTENT

- Applet: Applet & applet life cycle, Creating applets, Graphics class functions, Parameter passing to applet, Font & Color class, Event handling using event class
- AWT: working with windows, using AWT controls for GUI design, Swing class in JAVA
- Introduction to JDBC, JDBC-ODBC connectivity, JDBC architecture

AWT: Abstract Window Toolkit

- Java AWT is an API to develop GUI or window-based applications in java.
- The AWT hides you from the underlying details of the GUI your application will be running on and thus is at **very high level of abstraction**.
- java.awt package provides classes for AWT api such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc. These are also known as components.
- Using these components, we can create an interactive user interface for an application.

Java AWT Hierarchy



Components & containers

- All the elements like buttons, text fields, scrollbars etc are known as **components**.
- In AWT we have classes for each component.
- To have everything placed on a screen to a particular position, we have to add them to a container. ...(Frames, Windows, Panel, Dialog Boxes)
- A **container** is like a screen wherein we are placing components like buttons, text fields, checkbox etc. In short a container contains and controls the layout of components.

Containers

Window - container that have no borders and menu bars.

Panel - does not contain title bar, menu bar or border. Region internal to a Frame, used for grouping components together, not bounded by a visible border.

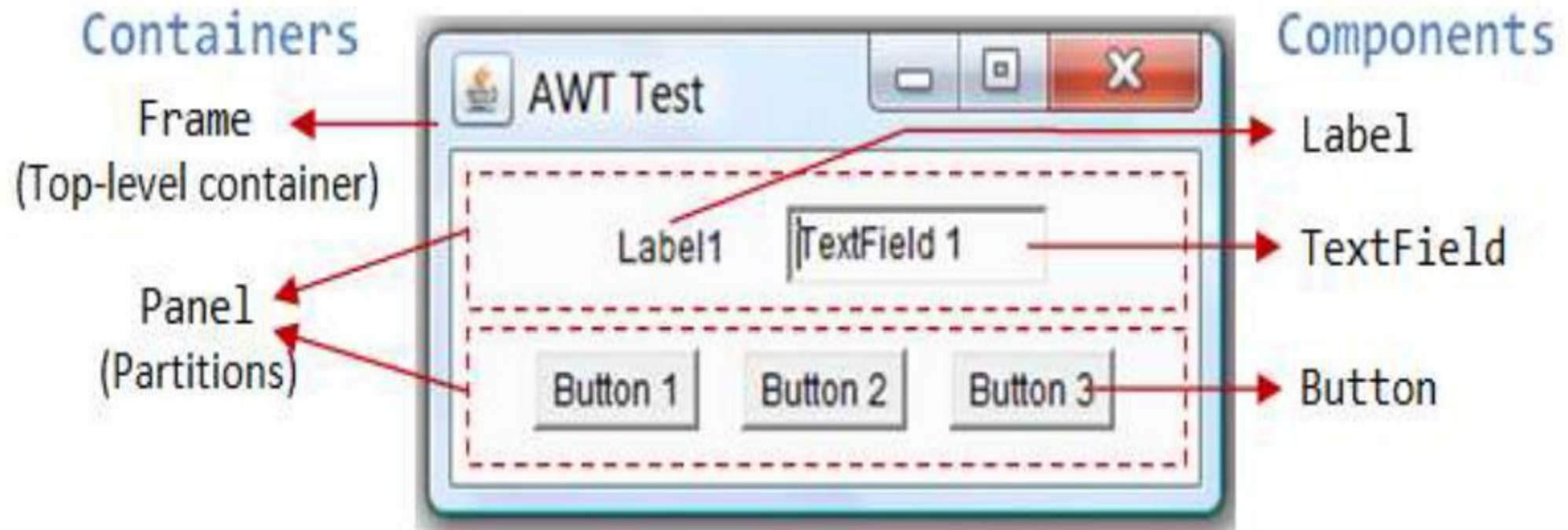
Frame - is the container that contain title bar and can have menu bars. It can have other components like button, textfield etc. Most widely used container.

Dialog – it has border and title. An instance of the Dialog class cannot exist without an associated instance of the Frame class.

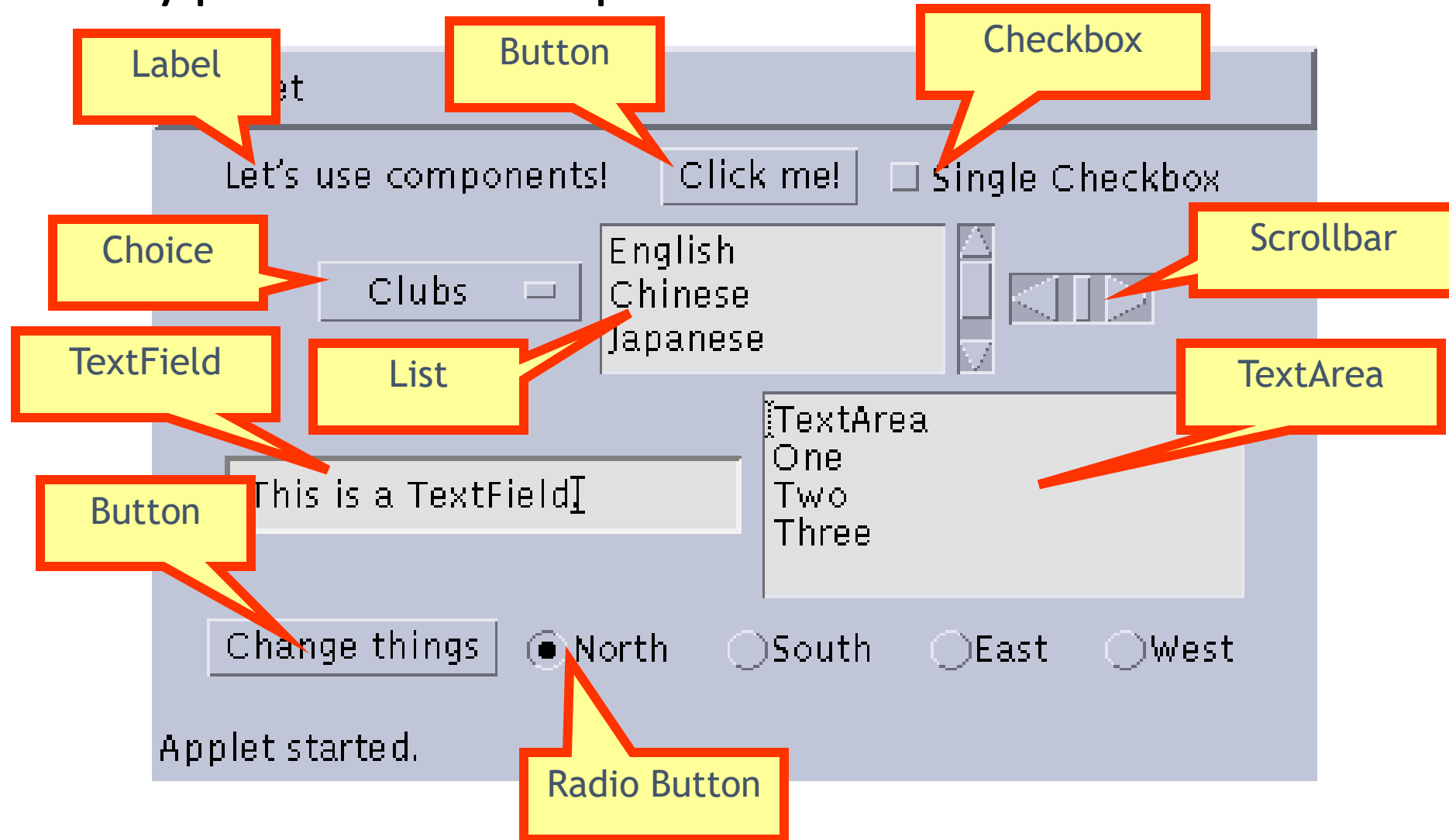
Useful methods of Component class

Method	Description
<code>public void add(Component c)</code>	inserts a component on this component.
<code>public void setSize(int width,int height)</code>	sets the size (width and height) of the component.
<code>public void setLayout(LayoutManager m)</code>	defines the layout manager for the component.
<code>public void setVisible(boolean status)</code>	changes the visibility of the component, by default false.

Example



Some types of components



button

menus

title bar

menu bar

combo box



scroll
bars

Calculator GUI



Creating a Frame

- There are two ways to create a Frame. They are,
 1. By extending Frame class (inheritance)
 2. By creating the object of Frame class (instantiating)

By extending Frame class

```
import java.awt.*;

class FirstFrame extends Frame
{
    FirstFrame()
    {
        Button b=new Button("click me");
        setTitle("This is my First AWT example");
        b.setBounds(30,100,80,30);      // setting button position
        add(b);                          //adding button into frame
        setSize(300,300);                //frame size 300 width and 300 height
        setLayout(null);                 //no layout manager
        setVisible(true);                //now frame will be visible, by default not visible
    }
    public static void main(String args[])
    {
        FirstFrame f=new FirstFrame();
    }
}
```

This is my First AWT ...

click me

B.1139]
rights reserved.

Programs\Module6

s\Module6>javac FirstFrame.java

s\Module6>java FirstFrame

By creating the object of Frame class

```
import java.awt.*;

class SecondFrame
{
    SecondFrame()
    {
        Frame f=new Frame();
        Button b=new Button("click me");
        f.setTitle("This is my Second AWT example");
        b.setBounds(30,100,80,30);
        f.add(b);
        f.setSize(300,300);
        f.setLayout(null);
        f.setVisible(true);
    }
    public static void main(String args[])
    {
        SecondFrame f=new SecondFrame();
    }
}
```



This is my Second A...



click me

```
18363.1139]  
All rights reserved.
```

```
C:\Users\juhig\OneDrive\00PM\Programs\Module6
```

```
grams\Module6>javac SecondFrame.java
```

```
1 error
```

```
1 error  
C:\Users\juhig\OneDrive\00PM\Programs\Module6>javac SecondFrame.java
```

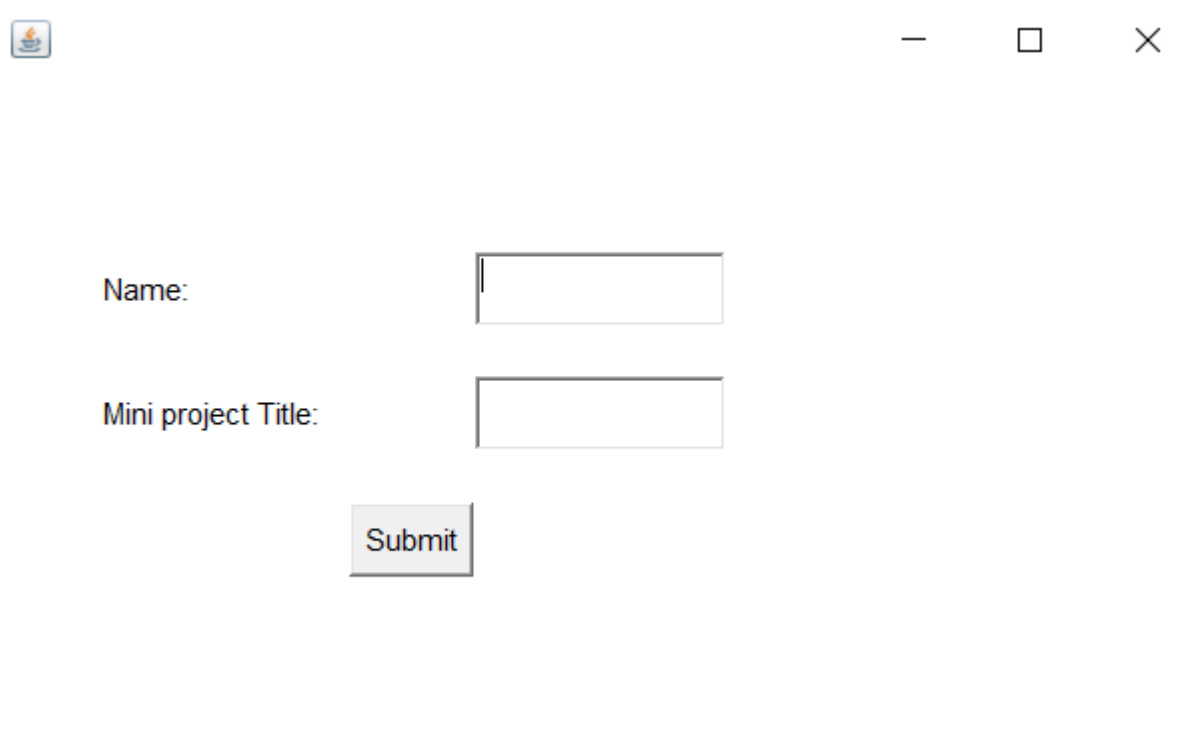
```
C:\Users\juhig\OneDrive\00PM\Programs\Module6>javac SecondFrame.java
```


```
C:\Users\juhig\OneDrive\00PM\Programs\Module6>javac SecondFrame.java
```

```
C:\Users\juhig\OneDrive\00PM\Programs\Module6>javac SecondFrame.java
```


```
C:\Users\juhig\OneDrive\00PM\Programs\Module6>javac SecondFrame.java
```


Example: Creating a form





—



✕

Name:

Mini project Title:

Submit

```
import java.awt.*;
public class Form {
    Form(){
        //Creating Frame
        Frame fr=new Frame();

        //Creating a label
        Label lb1 = new Label("Name: ");
        lb1.setBounds(50,100,100,30);
        //adding label to the frame
        fr.add(lb1);

        //Creating Text Field
        TextField t1 = new TextField();
        t1.setBounds(200,100,100,30);
        //adding text field to the frame
        fr.add(t1);
        Label lb2 = new Label("Mini project Title: ");
        lb2.setBounds(50,150,100,30);
        fr.add(lb2);
```

```
        TextField t2 = new TextField();
        t2.setBounds(200,150,100,30);
        fr.add(t2);

        Button b=new Button("Submit");
        b.setBounds(150,200,50,30);
        fr.add(b);
        //setting frame size
        fr.setSize(500, 300);

        //Setting the layout for the Frame
        fr.setLayout(null);

        fr.setVisible(true);
    }
    public static void main(String args[]){
        Form f = new Form();  }
}
```

Event handling

- Changing the state of an object is known as an event.
- For example, click on button, dragging mouse etc.
- The `java.awt.event` package provides many event classes and Listener interfaces for event handling.

Event handling by implementing ActionListener

- The Java ActionListener is notified whenever you click on the button or menu item.
- It is notified against ActionEvent.
- The ActionListener interface is found in java.awt.event package.
- It has only one method: actionPerformed().
- The actionPerformed() method is invoked automatically whenever you click on the registered component.

```
public abstract void actionPerformed(ActionEvent e);
```

Event handling by implementing ActionListener

1) Implement the ActionListener interface in the class:

```
public class ActionListenerExample Implements ActionListener
```

2) Register the component with the Listener:

```
component.addActionListener(instanceOfListenerclass);
```

3) Override the actionPerformed() method:

```
public void actionPerformed(ActionEvent e){  
    //Write the code here  
}
```

Button example with ActionListener



Welcome

click me

```
import java.awt.*;
import java.awt.event.*;
class ButtonExample extends Frame
implements ActionListener{
    TextField tf;
    ButtonExample(){

//create components
    tf=new TextField();
    tf.setBounds(60,50,170,20);
    Button b=new Button("click me");
    b.setBounds(100,120,80,30);

//register listener
    b.addActionListener(this);//passing current
    instance
```

```
//add components and set size, layout and
visibility
    add(b);add(tf);
    setSize(300,300);
    setLayout(null);
    setVisible(true);
}

    public void actionPerformed(ActionEvent
    e){
        tf.setText("Welcome");
    }

    public static void main(String args[]){
        new ButtonExample();
    }
}
```

TextField example with ActionListener



—



5

2

3

+

-


```

import java.awt.*;
import java.awt.event.*;
public class TextFieldExample extends Frame implements ActionListener{
    TextField tf1,tf2,tf3;
    Button b1,b2;
    TextFieldExample(){
        tf1=new TextField();
        tf1.setBounds(50,50,150,20);
        tf2=new TextField();
        tf2.setBounds(50,100,150,20);
        tf3=new TextField();
        tf3.setBounds(50,150,150,20);
        tf3.setEditable(false);
        b1=new Button("+");
        b1.setBounds(50,200,50,50);
        b2=new Button("-");
        b2.setBounds(120,200,50,50);
        b1.addActionListener(this);
        b2.addActionListener(this);
    }

```

```

        add(tf1);add(tf2);add(tf3);add(b1);add(b2);
        setSize(300,300);
        setLayout(null);
        setVisible(true); }
    public void actionPerformed(ActionEvent e) {
        String s1=tf1.getText();
        String s2=tf2.getText();
        int a=Integer.parseInt(s1);
        int b=Integer.parseInt(s2);
        int c=0;
        if(e.getSource()==b1){
            c=a+b;
        }else if(e.getSource()==b2){
            c=a-b;
        }
        String result=String.valueOf(c);
        tf3.setText(result); }
    public static void main(String[] args) {
        new TextFieldExample();
    } }

```

TextArea example with ActionListener

Words: 5 Characters: 40

Example of TextArea with ActionListener.

Count Words

```
import java.awt.*;
import java.awt.event.*;
public class TextAreaExample extends Frame implements ActionListener{
    Label l1,l2;
    TextArea area;
    Button b;
    TextAreaExample(){
        l1=new Label();
        l1.setBounds(50,50,100,30);
        l2=new Label();
        l2.setBounds(160,50,100,30);
        area=new TextArea();
        area.setBounds(20,100,300,300);
        b=new Button("Count Words");
        b.setBounds(100,400,100,30);
        b.addActionListener(this);
        add(l1);add(l2);add(area);add(b);
        setSize(400,450);
        setLayout(null);
        setVisible(true); }
}
```

```
public void actionPerformed(ActionEvent e){
    String text=area.getText();
    String words[]=text.split("\\s");
    l1.setText("Words: "+words.length);
    l2.setText("Characters: "+text.length());
}
public static void main(String[] args) {
    new TextAreaExample();
}
}
```

Event handling by implementing ItemListener

- The Java ItemListener is notified whenever you click on the checkbox or choice
- It is notified against ItemEvent.
- The ItemListener interface is found in java.awt.event package.
- It has only one method: itemStateChanged().
- **itemStateChanged() method**
- The itemStateChanged() method is invoked automatically whenever you click or unclick on the registered checkbox component.

```
public abstract void itemStateChanged(ItemEvent e);
```

CheckBox example with ItemListener

 CheckBox Example — □ ✕

C++ Checkbox: unchecked

☐ C++

☒ Java

```

import java.awt.*;
import java.awt.event.*;
public class CheckBoxExample implements ItemListener{
    Checkbox checkBox1,checkBox2;
    Label label;
    CheckBoxExample(){
        Frame f= new Frame("CheckBox Example");
        label = new Label();
        label.setAlignment(Label.CENTER);
        label.setSize(400,100);
        checkBox1 = new Checkbox("C++");
        checkBox1.setBounds(100,100, 50,50);
        checkBox2 = new Checkbox("Java");
        checkBox2.setBounds(100,150, 50,50);
        f.add(checkBox1); f.add(checkBox2); f.add(label);
        checkBox1.addItemListener(this);
        checkBox2.addItemListener(this);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);  }

```

```

public void itemStateChanged(ItemEvent e) {
    if(e.getSource()==checkBox1)
        label.setText("C++ Checkbox: "+
            (e.getStateChange()==1?"checked":"unchecked"));
    if(e.getSource()==checkBox2)
        label.setText("Java Checkbox: "+
            (e.getStateChange()==1?"checked":"unchecked"));
    }
public static void main(String args[])
{
    new CheckBoxExample();
}
}

```

Choice example with ActionListener

— ☐

Programming language Selected: Java

Java

```

import java.awt.*;
import java.awt.event.*;
public class ChoiceExample implements ActionListener
{
    Choice c;
    Label label;
    ChoiceExample(){
        Frame f= new Frame();
        label = new Label();
        label.setAlignment(Label.CENTER);
        label.setSize(400,100);
        Button b=new Button("Show");
        b.setBounds(200,100,50,20);
        c=new Choice();
        c.setBounds(100,100, 75,75);
        c.add("C");
        c.add("C++");
        c.add("Java");
        c.add("PHP");
        c.add("Android");

```

```

        f.add(c);f.add(label); f.add(b);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
        b.addActionListener(this) ;
    }

    public void actionPerformed(ActionEvent e) {
        String data = "Programming language Selected: "+
        c.getItem(c.getSelectedIndex());
        label.setText(data);
    }

    public static void main(String args[])
    {
        new ChoiceExample();
    }
}

```


JAVA SWING

```
mirror_mod = modifier_ob.  
# Add mirror object to mirror  
mirror_mod.mirror_object  
operation == "MIRROR_X":  
mirror_mod.use_x = True  
mirror_mod.use_y = False  
mirror_mod.use_z = False  
operation == "MIRROR_Y":  
mirror_mod.use_x = False  
mirror_mod.use_y = True  
mirror_mod.use_z = False  
operation == "MIRROR_Z":  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True  
  
# Selection at the end - add  
mirror_ob.select= 1  
modifier_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier_ob))  
mirror_ob.select = 0  
= bpy.context.selected_object  
data.objects[one.name].select  
print("please select exactly  
-- OPERATOR CLASSES ----  
  
types.Operator):  
on X mirror to the selected  
object.mirror_mirror_x"  
mirror X"  
  
context):  
context.active_object is not
```

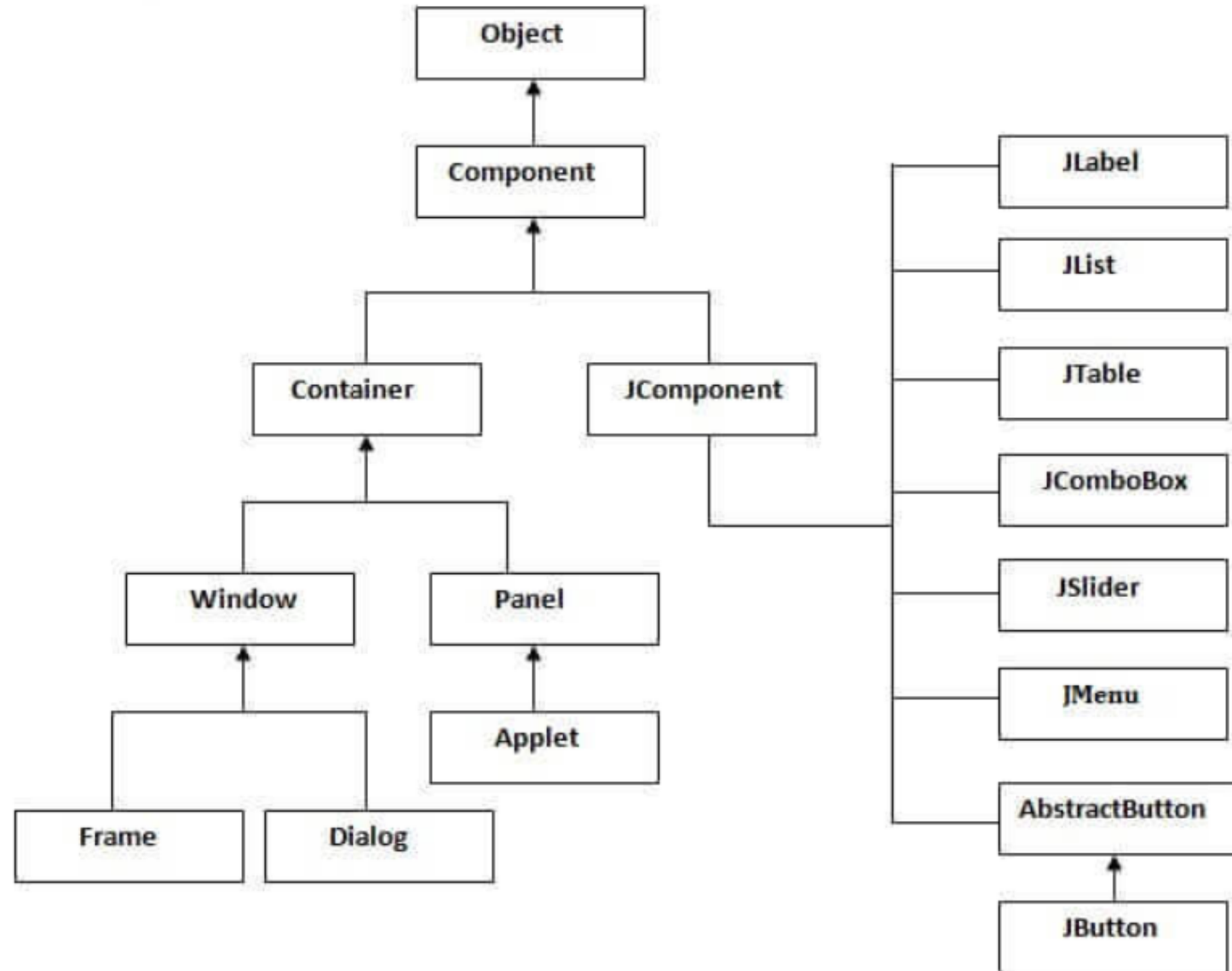
Introduction

- Java Swing tutorial is a part of Java Foundation Classes (JFC) that is used to create window-based applications.
- Unlike AWT, Java Swing provides platform-independent and lightweight components.
- javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

AWT Vs Swing

No.	Java AWT	Java Swing
1)	AWT components are platform-dependent .	Java swing components are platform-independent .
2)	AWT components are heavyweight .	Swing components are lightweight .
3)	AWT doesn't support pluggable look and feel .	Swing supports pluggable look and feel .
4)	AWT provides less components than Swing.	Swing provides more powerful components such as tables, lists, scrollpanes, colorchooser, tabbedpane etc.
5)	AWT doesn't follows MVC (Model View Controller) where model represents data, view represents presentation and controller acts as an interface between model and view.	Swing follows MVC .

Hierarchy of java Swing API



Java Swing example

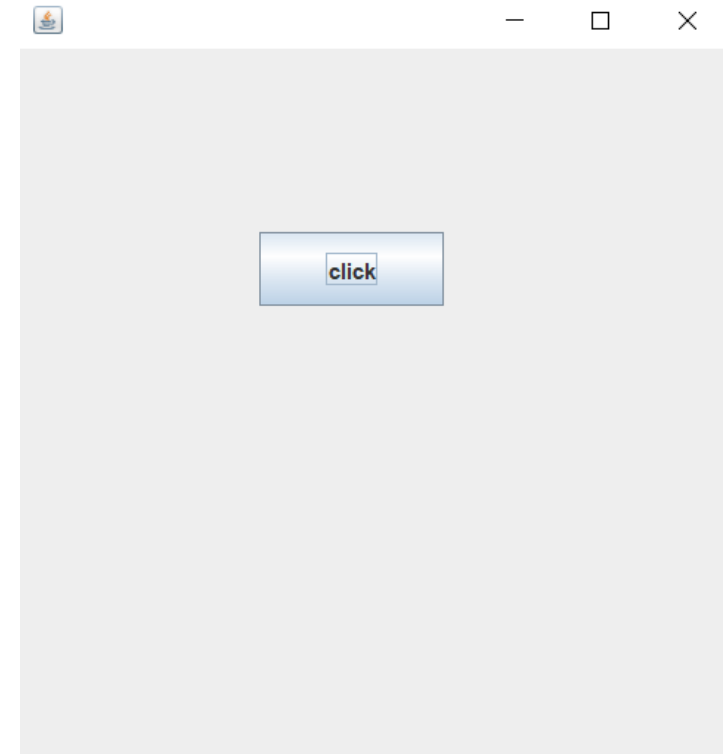
```
import javax.swing.*;


public class FirstSwingExample {
    public static void main(String[] args) {
        JFrame f=new JFrame();//creating instance of JFrame

        JButton b=new JButton("click");//creating instance of JButton
        b.setBounds(130,100,100, 40);//x axis, y axis, width, height

        f.add(b);//adding button in JFrame

        f.setSize(400,500);//400 width and 500 height
        f.setLayout(null);//using no layout managers
        f.setVisible(true);//making the frame visible
    }
}
```



 ActionListener in Java Swing Examples — □

OK Button is Clicked

OK

Cancel

Java Swing
example with
ActionListener

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

class ActionListenerExample extends JFrame implements
ActionListener {
    JLabel lblData;
    JButton btnOk,btnCancel;
    ActionListenerExample() {
        setLayout(new FlowLayout());
        lblData = new JLabel("Click any button to display data");
        btnOk=new JButton("OK");
        btnCancel = new JButton("Cancel");
        btnOk.addActionListener(this);
        btnCancel.addActionListener(this);
        add(lblData);
        add(btnOk);
        add(btnCancel); }
}
```

```
public void actionPerformed(ActionEvent e) {
    if(e.getSource() == btnOk)
        lblData.setText("OK Button is Clicked ");
    else if(e.getSource() == btnCancel)
        lblData.setText("Cancel Button is Clicked "); }
}

class SecondSwingExample
{ public static void main(String args[])
    {
        ActionListenerExample frame = new ActionListenerExample();
        frame.setTitle("ActionListener in Java Swing Examples");
        frame.setBounds(200,150,180,150);
        frame.setVisible(true);
    }
}
```



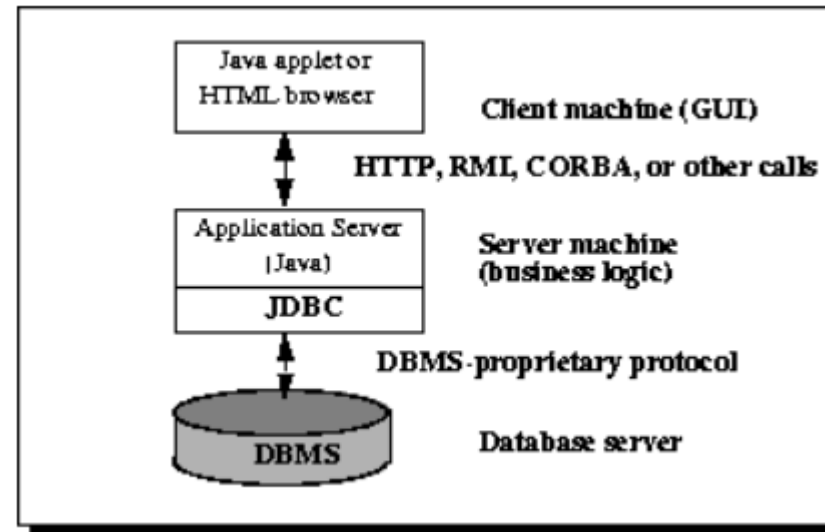
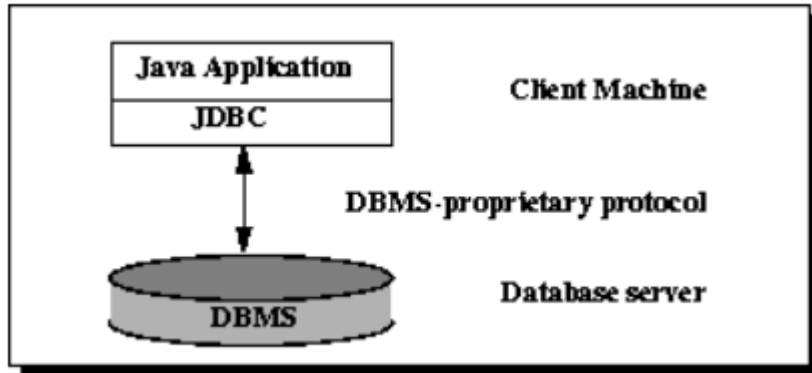
JDBC Connectivity

Introduction

- Java application cannot directly communicate with database
- This is because a database can interpret only SQL statements and not java language statements for this reason, we need a mechanism to translate java statements into SQL statements
- JDBC provides the mechanism for the kind of translation

JDBC

- JDBC stands for **J**ava **D**atabase **C**onnectivity
- It is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases



Steps to connect to the database

There are 5 steps to connect any java application with the database using JDBC.

These steps are as follows:

1. Register the Driver class
2. Create connection
3. Create statement
4. Execute queries
5. Close connection

1. Register the driver class

- forName method of Class class is used to register the driver class
- This method is used to dynamically load the driver class

Syntax of forName method:

```
public static void forName (String className )throws ClassNotFoundException
```

Calling a method:

```
Class.forName("com.mysql.cj.jdbc.Driver");
```

2. Create the connection object

getConnection method of DriverManager class is used to establish connection with the database.

Syntax of getConnection() method:

- public static Connection getConnection(String url)throws SQLException
- public static Connection getConnection(String url,String name,String password)throws SQLException

Calling a method:

```
Connection con=  
DriverManager.getConnection("jdbc:mysql://localhost:3306/Student",  
root","root")
```

3. Create the Statement object

- `createStatement()` method of `Connection` interface is used to create statement.
- object of statement is responsible to execute queries with the database.

Syntax of `createStatement()` method:

```
public Statement createStatement()throws SQLException
```

Calling a method:

```
Statement stmt=con.createStatement();
```

4. Execute the query

- `executeQuery()` method of `Statement` interface is used to execute queries to the database.
- This method returns the object of `ResultSet` that can be used to get all the records of a table

Syntax of `executeQuery()` method

```
public ResultSet executeQuery (String sql )throws SQLException
```

Calling a method:

```
ResultSet rs= stmt.executeQuery ("select * from stud");
```

5. Close the connection object

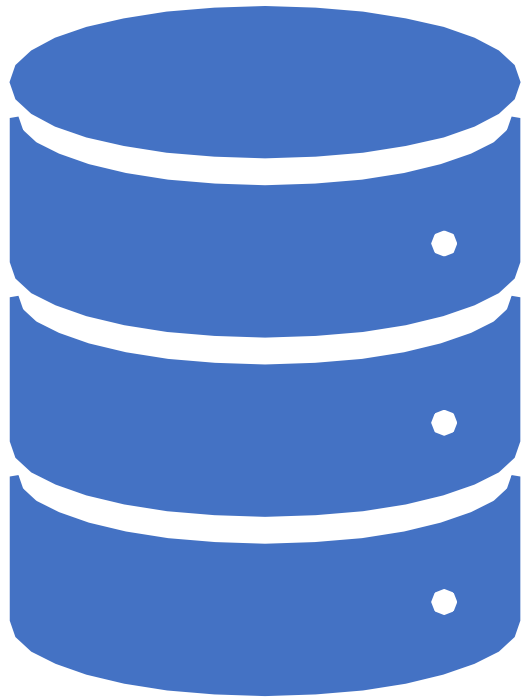
- By closing connection object, Statement and ResultSet will be closed automatically
- The close() method of Connection interface is used to close the connection

Syntax of close() method:

```
public void close() throws SQLException
```

Calling a method:

```
con.close();
```

JDBC program to insert record
in a stud table of database
Student

```
Import java.sql.*;

public class JDBCDEMO
{ // JDBC driver name and database URL

static final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";

static final String DB_URL =
"jdbc:mysql://localhost:3306/Student";

// Database credentials

static final String USER = "root";
static final String PASS = "root";

public static void main(String[] args) {

Connection conn = null;
Statement stmt = null;

try{

//Register JDBC driver
Class.forName(JDBC_DRIVER );

//Open a connection
System.out.println("Connecting to database...");

conn =DriverManager.getConnection(DB_URL,USER,PASS);
```

```
//Execute a query

System.out.println("Creating statement...");

stmt = conn.createStatement();

String sql;

sql = "insert into stud(name,rollno) values ('xyz','3')";

int r= stmt.executeUpdate(sql);

ResultSet rs = stmt.executeQuery("Select * from stud");

//Extract data from result set

while(rs.next()){

System.out.println("Roll no: "+rs.getInt(2)+" Name:
"+rs.getString(1));      }

//Clean-up environment

rs.close();  stmt.close();  conn.close();
}catch(SQLException se){      se.printStackTrace(); }

catch(Exception e){      e.printStackTrace(); }
System.out.println("Goodbye!");

}
```

Output

Command Prompt

```
C:\Users\juhig\OneDrive\OOPM\Programs\Module6>javac JDBCDEMO.java
```

```
C:\Users\juhig\OneDrive\OOPM\Programs\Module6>java -cp .;"C:\Program Files\Java\jdk-14.0.2\lib\ext1\mysql-connector-java-8.0.22.jar" JDBCDEMO
```

```
Connecting to database...
```

```
Creating statement...
```

```
Roll no: 3 Name: xyz
```

```
Goodbye!
```

```
C:\Users\juhig\OneDrive\OOPM\Programs\Module6>_
```

Applets



Introduction

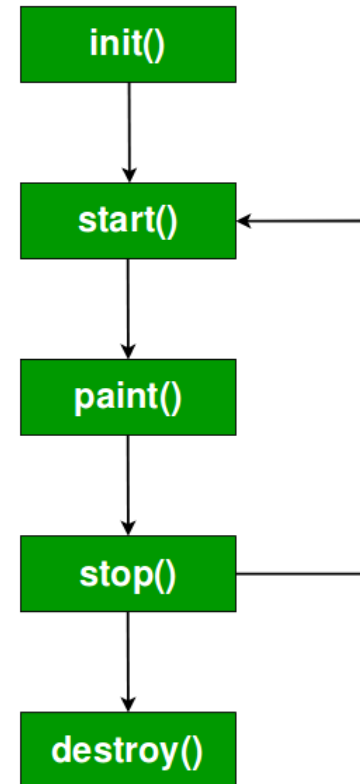
- An applet is a Java program that can be embedded into a web page.
- It runs inside the web browser and works at client side.
- In other words, we can say that Applets are small Java applications that can be accessed on an Internet server, transported over Internet, and can be automatically installed and run as apart of a web document.

Important points

1. All applets are sub-classes (either directly or indirectly) of java.applet.Applet class.
2. Applets are not stand-alone programs. Instead, they run within either a web browser or an applet viewer. JDK provides a standard applet viewer tool called applet viewer.
3. In general, execution of an applet does not begin at main() method.
4. Output of an applet window is not performed by System.out.println(). Rather it is handled with various AWT methods, such as drawString().

Applet lifecycle

Applet Lifecycle



Executing an applet

- There are two ways to run an applet:
 - 1.By html file.
 - 2.By appletViewer tool (for testing purpose).

Creating applet by html file

To execute the applet by html file, create an applet and compile it. After that create an html file and place the applet code in html file. Now click the html file.

First.java

```
import java.applet.Applet;  
import java.awt.Graphics;  
public class First extends Applet{  
  
    public void paint(Graphics g){  
        g.drawString("welcome",150,150);  
    }  
  
}
```

myApplet.html

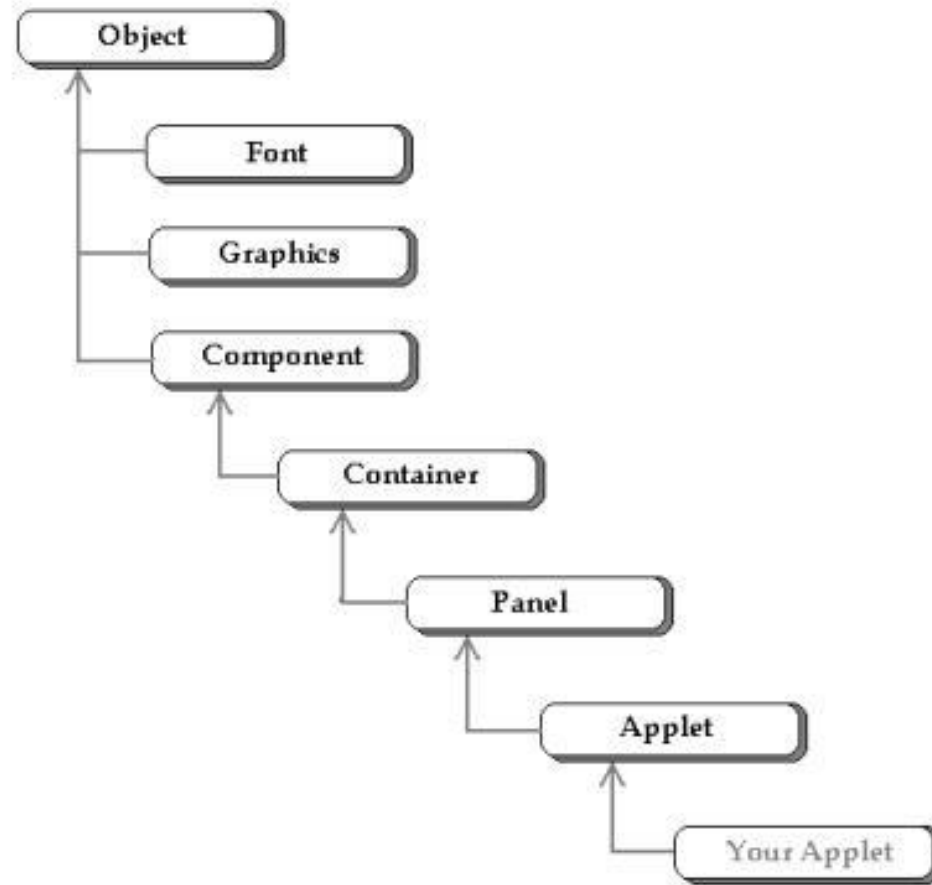
```
<html>  
<body>  
<applet code="First.class"  
width="300" height="300">  
</applet>  
</body>  
</html>
```

Creating applet by appletviewer tool

- To execute the applet by appletviewer tool, create an applet that contains applet tag in comment and compile it. After that run it by: **appletviewer First.java**. Now Html file is not required but it is for testing purpose only.

```
//First.java
import java.applet.Applet;
import java.awt.Graphics;
/*
<applet code="First.class" width="300" height="300">
</applet>
*/
public class First extends Applet{
    public void paint(Graphics g){
        g.drawString("welcome to applet",150,150);
    } }
```

Applet Class Hierarchy



Graphics in Applet

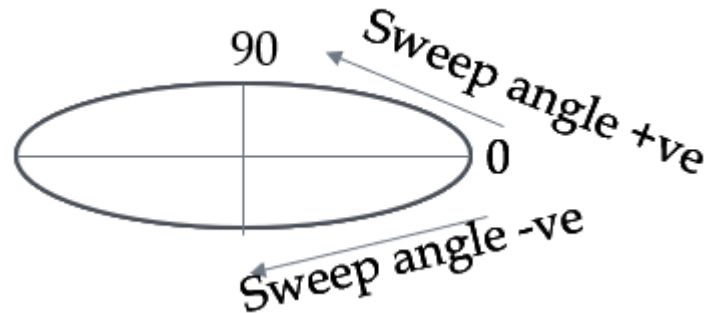
- The Graphics class (in awt package) defines a number of drawing functions.
- Each shape can be drawn edge-only or filled.
- Objects are drawn and filled in the currently selected graphics color (black by default).
- When a graphics object is drawn that exceeds the dimensions of the window, output is automatically clipped.

Methods in Graphic class

To draw	Method	Syntax
Line	drawLine()	void drawLine(int <i>startX</i> , int <i>startY</i> , int <i>endX</i> , int <i>endY</i>)
Rectangle	drawRect() fillRect()	void drawRect(int <i>top</i> , int <i>left</i> , int <i>width</i> , int <i>height</i>) void fillRect(int <i>top</i> , int <i>left</i> , int <i>width</i> , int <i>height</i>)
Rounded Rectangle	drawRoundRect() fillRoundRect()	void drawRoundRect(int <i>top</i> , int <i>left</i> , int <i>width</i> , int <i>height</i> , int <i>xDiam</i> , int <i>yDiam</i>) void fillRoundRect(int <i>top</i> , int <i>left</i> , int <i>width</i> , int <i>height</i> , int <i>xDiam</i> , int <i>yDiam</i>)
Oval	drawOval() fillOval()	void drawOval(int <i>top</i> , int <i>left</i> , int <i>width</i> , int <i>height</i>) void fillOval(int <i>top</i> , int <i>left</i> , int <i>width</i> , int <i>height</i>)
Arc	drawArc() fillArc()	void drawArc(int <i>top</i> , int <i>left</i> , int <i>width</i> , int <i>height</i> , int <i>startAngle</i> , int <i>sweepAngle</i>) void fillArc(int <i>top</i> , int <i>left</i> , int <i>width</i> , int <i>height</i> , int <i>startAngle</i> , int <i>sweepAngle</i>)
Polygon	drawPolygon() fillPolygon()	void drawPolygon(int <i>x</i> [], int <i>y</i> [], int <i>numPoints</i>) void fillPolygon(int <i>x</i> [], int <i>y</i> [], int <i>numPoints</i>)

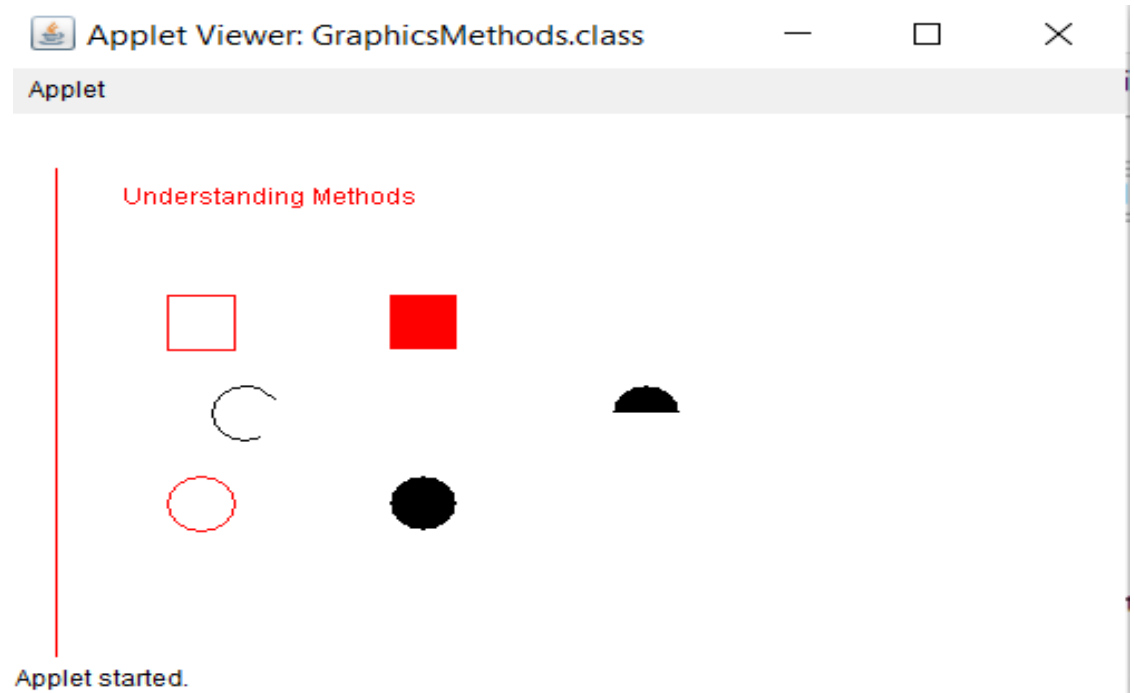
drawArc()

- void drawArc(int top, int left, int width, int height, int startAngle, int sweepAngle)



Graphics programming

```
import java.applet.Applet;  
import java.awt.*;  
public class GraphicsMethods extends Applet{  
    public void paint(Graphics g){  
        g.setColor(Color.red);  
        g.drawString("Understanding Methods",50, 50);  
        g.drawLine(20,30,20,300);  
        g.drawRect(70,100,30,30);  
        g.fillRect(170,100,30,30);  
        g.drawOval(70,200,30,30);  
        g.setColor(Color.black);  
        g.fillOval(170,200,30,30);  
        g.drawArc(90,150,30,30,30,270);  
        g.fillArc(270,150,30,30,0,180); }  
}
```



Draw a smiley in Java Applet

```
import java.applet.*;
import java.awt.*;
public class Smiley extends Applet {
    public void paint(Graphics g)
    {
        // Oval for face outline
        g.drawOval(80, 70, 150, 150);
        // Ovals for eyes
        // with black color filled
        g.setColor(Color.BLACK);
        g.fillOval(120, 120, 15, 15);
        g.fillOval(170, 120, 15, 15);
        // Arc for the smile
        g.drawArc(130, 180, 50, 20, 0, -180);
    }
}
```

