14EE706/14OE706

Hall Ticket Number:

IV/IV B.Tech (Regular/Supplementary) DEGREE EXAMINATION

November, 2019 | **Electrical & Electronics Engineering**

Seventh Semester | **Java Programming**

Time: Three Hours | **Maximum :** 60 Marks

Answer Question No.1 compulsorily. (1X12 = 12 Marks)

Answer ONE question from each unit. (4X12=48 Marks)

1 Answer all questions (1X12=12 Marks)

a) Define scope and lifetime.
b) Give the syntax for StringBuffer.
c) What is method overriding?
d) What are the types of exceptions?
e) Specify need of synchronization of threads.
f) State the different methods of applet life cycle.
g) Differentiate frames and applets.
h) Write methods of Color class.
i) List out AWT components.
j) Which class supports radio buttons?
k) Define swing.
l) List the types of JDBC drivers.

UNIT I

2 a) Define constructor. Explain about constructor overloading. 6M
b) What is the use of super keyword? Explain with an example. 6M

(OR)

3 a) What is the importance of static keyword? Explain with examples. 6M
b) What is a Package? Explain the procedure for creating and accessing packages in java. 6M

UNIT II

4 a) Explain the usage of try, catch and finally keywords with an example. 6M
b) Define Thread. Explain the life cycle of thread with a neat sketch. 6M

(OR)

5 a) Explain about Graphics class in applets with examples. 6M
b) Explain how we can pass parameters to applets using an example. 6M

UNIT III

6 a) List different layout managers that AWT supports? Write a Java program for one layout manager. 6M
b) Explain about mouse events with example. 6M

(OR)

7 a) Explain about the delegation event model. 6M
b) What is an adapter class? Explain its purpose and functionality. 6M

UNIT IV

8 a) Write a Java program to demonstrate check boxes in swing. 6M
b) Differentiate between AWT and Swing. 6M

(OR)

9 a) Write the steps involved in JDBC connectivity process with example. 6M
b) Discuss about JDBC drivers. 6M

IV/IV B.Tech (Regular/Supplementary) DEGREE EXAMINATION

November, 2019 **Electronics & Electrical Engineering**
Seventh Semester **JAVA Programming (14EE706/CS02)**
Time: Three Hours **Maximum : 60 Marks**

SCHEME OF EVALUATION

Prepared By : **J.KUMARARAJA**
Assistant Professor,
Dept. of CSE,
BEC, BAPATLA.

(Signature)

(Signature)
Head of the Dept,
Dept. of CSE,
BEC, BAPATLA.

SCHEME OF EVALUATION
Allocation of Marks

UNIT – I

2. a) Define constructor. Explain about constructor overloading.

Explanation	**–**	**3Marks**
Program	**–**	**3Marks**

2. b) What is the use of super keyword? Explain with an example.

Explanation	**–**	**3Marks**
Program	**–**	**3Marks**

(or)

3. a) What is the importance of static keyword? Explain with examples.

Explanation	**–**	**3Marks**
Program	**–**	**3Marks**

3. b) What is a Package? Explain the procedure for creating and accessing packages in java.

Explanation	**–**	**3Marks**
Program	**–**	**3Marks**

UNIT – II

4. a) Explain the usage of try, catch and finally keywords with an example.

Explanation	**–**	**3Marks**
Program	**–**	**3Marks**

4. b) Define Thread. Explain the life cycle of thread with a neat sketch.

Diagram	**–**	**3Marks**
Explanation	**–**	**3Marks**

(or)

5. a) Explain about Graphics class in applets with examples.

Explanation	**–**	**3Marks**
Program	**–**	**3Marks**

5. b) Explain how we can pass parameters to applets using an example.

Explanation	**–**	**3Marks**
Program	**–**	**3Marks**

<u>UNIT – III</u>

6. a) List different layout managers that AWT supports? Write a Java program for one layout manager.

Explanation	**–**	**3Marks**
Program	**–**	**3Marks**

6. b) Explain about mouse events with example.

Explanation	**–**	**3Marks**
Program	**–**	**3Marks**

(or)

7. a) Explain about the delegation event model.

Diagram	**–**	**3Marks**
Explanation	**–**	**3Marks**

7. b) What is an adapter class? Explain its purpose and functionality.

Explanation	**–**	**3Marks**
Program	**–**	**3Marks**

<u>UNIT – IV</u>

8. a) Write a Java program to demonstrate check boxes in swing.

Program	**–**	**6Marks**

8. b) Differentiate between AWT and Swing.

Explanation	**–**	**6Marks**

(or)

9. a) Write the steps involved in JDBC connectivity process with example.

Explanation	**–**	**2Marks**
Program	**–**	**4Marks**

9. b) Discuss about JDBC drivers.

Listing the names	**–**	**2Marks**
Explanation	**–**	**4Marks**

- - - - -

IV/IV B.Tech (Regular/Supplementary) DEGREE EXAMINATION

November, 2019 **Electronics & Electrical Engineering**

Seventh Semester **JAVA Programming (14EE706/CS02)**

Time: Three Hours **Maximum : 60 Marks**

SCHEME OF EVALUATION - ANSWERS

Answer Questions No. 1 compulsorily ***(1 X 12 = 12 Marks)***

Answer ONE question from each unit. ***(4 X 12 = 48 Marks)***

1. Answer all questions **(1 X 12 = 12 Marks)**

a) Define scope and lifetime.

Scope of a variable refers to in which areas or sections of a program can the variable be accessed and **lifetime** of a variable refers to how long the variable stays alive in memory.

General convention for a variable's scope is, it is accessible only within the block in which it is declared.

b) Give the syntax for StringBuffer.

Java StringBuffer class is used to create mutable (modifiable) string. The StringBuffer class in java is same as String class except it is mutable i.e. it can be changed.

c) What is method overriding?

If subclass (child class) has the same method as declared in the parent class, it is known as **method overriding in Java**.

In other words, If a subclass provides the specific implementation of the method that has been declared by one of its parent class, it is known as method overriding.

Usage of Java Method Overriding

- ❖ Method overriding is used to provide the specific implementation of a method which is already provided by its superclass.
- ❖ Method overriding is used for runtime polymorphism.

d) What are the types of exceptions?

There are mainly two types of exceptions: checked and unchecked. Here, an error is considered as the unchecked exception. According to Oracle, there are three types of exceptions:

1. Checked Exception
2. Unchecked Exception 3. Error

e) Specify need of synchronization of threads.

Synchronization in java is the capability *to control the access of multiple threads to any shared resource*.

Java Synchronization is better option where we want to allow only one thread to access the shared resource.

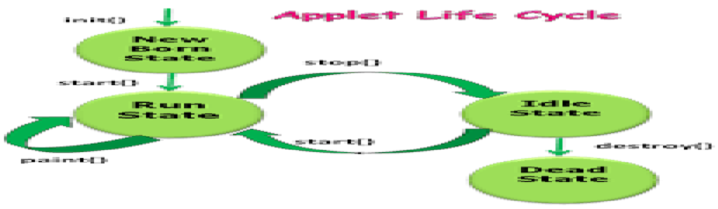
Why use Synchronization

The synchronization is mainly used to

1. To prevent thread interference.
2. To prevent consistency problem.

f) State the different methods of applet life cycle.

Java *applet* inherits features from the class *Applet.* Thus, whenever an *applet* is created, it undergoes a series of changes from initialization to destruction. Various stages of an *applet* life cycle are depicted in the figure below:



- *init( )*
- *start( )*
- *paint( )*
- *stop( )*
- *destroy( )*

g) Differentiate frames and applets.

Frames on the other hand is a window which is decorated that is contains border, title, the maximise minimise and close buttons.

Frame based applications come under the category of desktop Java applications and run on their own without the need of web browser.

h) Write methods of Color class.

- ❖ **static Color getColor(String nm)** - Finds a color in the system properties.
- ❖ **static setColor(Color name)** - Sets a color in the system properties.

i) List out AWT components.

Java AWT (Abstract Window Toolkit) is *an API to develop GUI or window-based applications* in java.

Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavyweight i.e. its components are using the resources of OS.

The java.awt package provides classes for AWT api such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc.

j) Which class supports radio buttons?

The JRadioButton class is used to create a radio button. It is used to choose one option from multiple options. It is widely used in exam systems or quiz.

It should be added in ButtonGroup to select one radio button only.

k) Define swing.

Java Swing tutorial is a part of Java Foundation Classes (JFC) that is *used to create window-based applications*. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.

Unlike AWT, Java Swing provides platform-independent and lightweight components.

The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

l) List the types of JDBC drivers.

JDBC Driver is a software component that enables java application to interact with the database. There are 4 types of JDBC drivers:

1. JDBC-ODBC bridge driver
2. Native-API driver (partially java driver)
3. Network Protocol driver (fully java driver)
4. Thin driver (fully java driver)

2. a) Define constructor. Explain about constructor overloading.

A **constructor** looks more like a method but without return type. Moreover, the name of the constructor and the class name should be the same.

The advantage of constructors over methods is that they are **called implicitly** whenever an object is created. In case of methods, they must be called explicitly. To create an object, the constructor must be called. Constructor gives properties to an object at the time of creation itself (else, it takes some method calls with extra code to do the same job). Programmer uses constructor for initializing variables, instantiating objects.

Constructor Overloading

Just like method overloading, constructors also can be overloaded. Same constructor declared with different parameters in the same class is known as constructor overloading. Compiler differentiates which constructor is to be called depending upon the number of parameters and their sequence of data types.

Ex. Prg:

```
                // Prg. to demonstrate constructor overloading
import java.lang.*;
class Perimeter
{
        public Perimeter()
        {
                System.out.println("From default");
        }
        public Perimeter(int x)
        {
                System.out.println("Circle perimeter: " + 2*Math.PI*x);
        }

        public Perimeter(int x, int y)
        {
                System.out.println("Rectangle perimeter: " +2*(x+y));
        }
        public static void main(String args[])
        {
                Perimeter p1 = new Perimeter();
                Perimeter p2 = new Perimeter(10);
                Perimeter p3 = new Perimeter(10, 20);
        }
}
```

2. b) What is the use of super keyword? Explain with an example.

The **super** keyword in Java is a reference variable which is used to refer immediate parent class object.

Whenever you create the instance of subclass, an instance of parent class is created implicitly which is referred by super reference variable.

Usage of Java super Keyword

1. super can be used to refer immediate parent class instance variable.
2. super can be used to invoke immediate parent class method.
3. super( ) can be used to invoke immediate parent class constructor.

Ex.Prg:

```
      // Prg. demonstrate the usage of super keyword
class Person
{
    public String name;
    Person(String s)
    {
         name = s;
    }
    public void display()
    {
      System.out.println("Name of Person = " + name);
    }
}
class Staff extends Person
{
    private int staffId;
    Staff(String name,int staffId)
    {
      super(name); // invoke Person class constructor
      this.staffId = staffId;
    }
    public void display()
    {
      super.display();
        System.out.println("Staff Name from child class is  =
                                                   " +super.name);
      System.out.println("Staff Id is  = " + staffId);
    }
}
```

```
class TemporaryStaff extends Staff

{

    private int days;
    private int hoursWorked;
    TemporaryStaff(String sname,int id,int days,int hoursWorked)
    {
       super(sname,id); // invoke Staff class constructor
       this.days  = days;
       this.hoursWorked = hoursWorked;
    }
    public int Salary()
    {
        int salary = days * hoursWorked * 50;

      return salary;
    }
    public void display()
    {
      super.display(); // invoke Staff class display()
      System.out.println("No. of Days = " + days);
      System.out.println("No. of Hours Worked = " +
                                      hoursWorked +" Hrs.");
      System.out.println("Total Salary (Rs./-)= " + Salary());
    }
}
class main
{
    public static void main(String args[])
    {
        TemporaryStaff ts = new TemporaryStaff("Kumar",999,10,20);
        ts.display();
    }
}
```

3. a) What is the importance of static keyword? Explain with examples.

The **static keyword** in Java is used for memory management mainly. We can apply java static keyword with variables, methods, blocks and nested class. The static keyword belongs to the class than an instance of the class.

The static can be:

1. Variable (also known as a class variable)
2. Method (also known as a class method) 3. Block 4. Nested class

1) Java static variable

If you declare any variable as static, it is known as a static variable.

- The static variable can be used to refer to the common property of all objects (which is not unique for each object), for example, the company name of employees, college name of students, etc.
- The static variable gets memory only once in the class area at the time of class loading.

Advantages of static variable

It makes your program **memory efficient** (i.e., it saves memory).

Understanding the problem without static variable

```
class Student
{
     int rollno;
    String name;
    String college="ITS";
}
```

Suppose there are 500 students in my college, now all instance data members will get memory each time when the object is created. All students have its unique rollno and name, so instance data member is good in such case. Here, "college" refers to the common property of all objects. If we make it static, this field will get the memory only once.

Ex. Prg:

```
        // prg. to demonstrate the usage of static keyword
import java.lang.*;
class Student
{
     int rollno;
     String name;
     static String college = "CEC";

     Student(int r, String n)
         {
                rollno = r;
                name = n;
     }
         static void change()
         {
                college = "BEC";
     }
        static
        {
                System.out.println("\n<--Student BioData-->\n");
                System.out.println("\tRNo.\tName\tCollege");
                System.out.println("\t----\t----\t----");

        }

    void display ()
        {
                System.out.println("\t"+rollno+"\t"+name+"\t"+college);
        }
    public static void main(String args[])
        {

                 Student s1 = new Student(111,"Karan");
                 Student s2 = new Student(222,"Aryan");
                 Student s3 = new Student(333,"Sonoo");

                Student.change();

                s1.display();
                s2.display();
                s3.display();
    }
}
```

3. b) What is a Package? Explain the procedure for creating and accessing packages in java.

A **java package** is a group of similar types of classes, interfaces and sub-packages.

Package in java can be categorized in two form, built-in package and user-defined package.

There are many built-in packages such as java.lang, java.awt, javax.swing, java.net, java.io, java.util, java.sql etc.

Advantages:

Java package is used to categorize the classes and interfaces so that they can be easily maintained.
Java package provides access protection.
Java package removes naming collision.

Program 1:

```
// Prg. Creating a package pack with
Addition Class
package pack;    //pack is the package name
public class Addition
{
      //instance vars
      private int n1,n2;
      public Addition(int a,int b)
      {
            n1=a;
            n2=b;
      }
      // method to find sum of two numbers
      public void sum()
      {
            System.out.println
                              ("Sum :"+(n1+n2));
      }
```

Compile the above program as s

javac - d . Addition.java

Program 2:

```
       // Prg : To  access the package pack
pack mypack;
import pack.*;
class Use
{
       public static void main(String args[])
       {
```

UNIT – II

4. a) Explain the usage of try, catch and finally keywords with an example.

Java try block

Java **try** block is used to enclose the code that might throw an exception. It must be used within the method.

If an exception occurs at the particular statement of try block, the rest of the block code will not execute. So, it is recommended not to keeping the code in try block that will not throw an exception.

Java try block must be followed by either catch or finally block.

Syntax of Java try-catch

```
try
{
        //code that may throw an exception
}
catch(Exception_class_Name ref)
{
        //code that may handle exception
}
```

Syntax of try-finally block

```
try
{
        //code that may throw an exception
}
finally
{
        //code that may end the execution
}
```

Java finally block

Java finally block is a block that is used *to execute important code* such as closing connection, stream etc.
Java finally block is always executed whether exception is handled or not.
Java finally block follows try or catch block.
Finally block in java can be used to put "cleanup" code such as closing a file, closing connection etc.

Ex. Prg:

```
// Prg. to demonstrate on Exception Handling
import java.lang.*;
import java.io.*;
class Ex2
{
        public static void main(String args[])
        {
                try
                {
                        // Open the files
                        System.out.println ("Open Files");
                        // Do Some Processing
                        int n=args.length;
                        System.out.println ("No of Arguments:"+n);

                        int a=45/n; //causes a exception if no args. passed

                        System.out.println ("Value of a is:"+a);
                }
                catch(ArithmeticException ae)
                {
                        // Display the Exception Details
                                System.out.println (ae);
                        //      Display any message to the user
                                System.out.println ("Please pass data while
running this program:");
                }
                finally
                {
                                // Close files
                                System.out.println ("Close files");
                }
        }
}
```

4. b) Define Thread. Explain the life cycle of thread with a neat sketch.

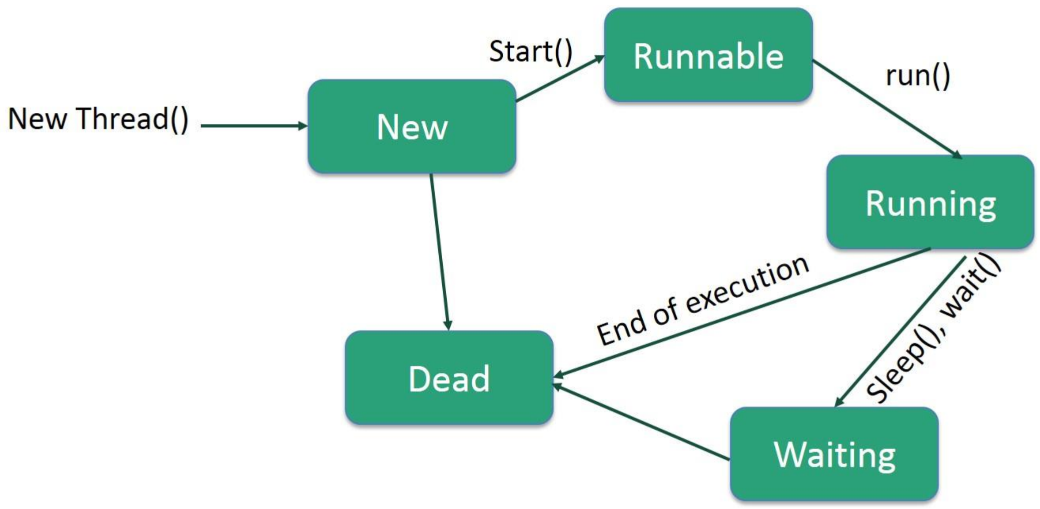
A thread can be in one of the five states. According to sun, there is only 4 states in **thread life cycle in java** new, runnable, non-runnable and terminated. There is no running state.

But for better understanding the threads, we are explaining it in the 5 states.

The life cycle of the thread in java is controlled by JVM. The java thread states are as follows:

1. New
2. Runnable

3. Running
4. Non-Runnable (Blocked or Waiting)
5. Terminated



New

The thread is in new state if you create an instance of Thread class but before the invocation of start( ) method.

Runnable

The thread is in runnable state after invocation of start( ) method, but the thread scheduler has not selected it to be the running thread.

Running

The thread is in running state if the thread scheduler has selected it.

Non-Runnable (Blocked or waiting)

This is the state when the thread is still alive, but is currently not eligible to run.

Terminated (Dead)

A thread is in terminated or dead state when its run( ) method exits.

5. a) Explain about Graphics class in applets with examples.

java.awt.Graphics class provides many methods for graphics programming.

Commonly used methods of Graphics class:

1. **public abstract void drawString(String str, int x, int y):** is used to draw the specified string.
2. **public void drawRect(int x, int y, int width, int height):** draws a rectangle with the specified width and height.
3. **public abstract void fillRect(int x, int y, int width, int height):** is used to fill rectangle with the default color and specified width and height.
4. **public abstract void drawOval(int x, int y, int width, int height):** is used to draw oval with the specified width and height.
5. **public abstract void fillOval(int x, int y, int width, int height):** is used to fill oval with the default color and specified width and height.
6. **public abstract void drawLine(int x1, int y1, int x2, int y2):** is used to draw line between the points(x1, y1) and (x2, y2).
7. **public abstract boolean drawImage(Image img, int x, int y, ImageObserver observer):** is used draw the specified image.
8. **public abstract void drawArc(int x, int y, int width, int height, int startAngle, int arcAngle):** is used draw a circular or elliptical arc.
9. **public abstract void fillArc(int x, int y, int width, int height, int startAngle, int arcAngle):** is used to fill a circular or elliptical arc.
10. **public abstract void setColor(Color c):** is used to set the graphics current color to the specified color.
11. **public abstract void setFont(Font font):** is used to set the graphics current font to the specified font.

Example of Graphics in applet:

Ex.Prg:

```
      // Prg to demonstrate on Graphics class

import java.applet.Applet;
import java.awt.*;

/* <applet code="GraphicsDemo.class" width="300" height="300">
</applet>   */

public class GraphicsDemo extends Applet
{

      public void paint(Graphics g)
      {
             g.setColor(Color.red);
             g.drawString("Welcome",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.pink);
                g.fillOval(170,200,30,30);
                g.drawArc(90,150,30,30,30,270);
                g.fillArc(270,150,30,30,0,180);
        }
}
```

5. b) Explain how we can pass parameters to applets using an example.

Passing parameters to applets using the param tag and retrieving the values of parameters using getParameter method.

Parameters specify extra information that can be passed to an applet from the HTML page. Parameters are specified using the HTML's *param* tag.

Param Tag

The <param> tag is a sub tag of the <applet> tag. The <param> tag contains two attributes: *name* and *value* which are used to specify the name of the parameter and the value of the parameter respectively. For example, the param tags for passing name and age parameters looks as shown below:

<param name="name" value="Ramesh" /><param name="age" value="25" />

Now, these two parameters can be accessed in the applet program using the *getParameter()* method of the *Applet* class.

getParameter( ) Method

The *getParameter()* method of the *Applet* class can be used to retrieve the parameters passed from the HTML page. The syntax of *getParameter()* method is as follows:

String getParameter(String param-name);

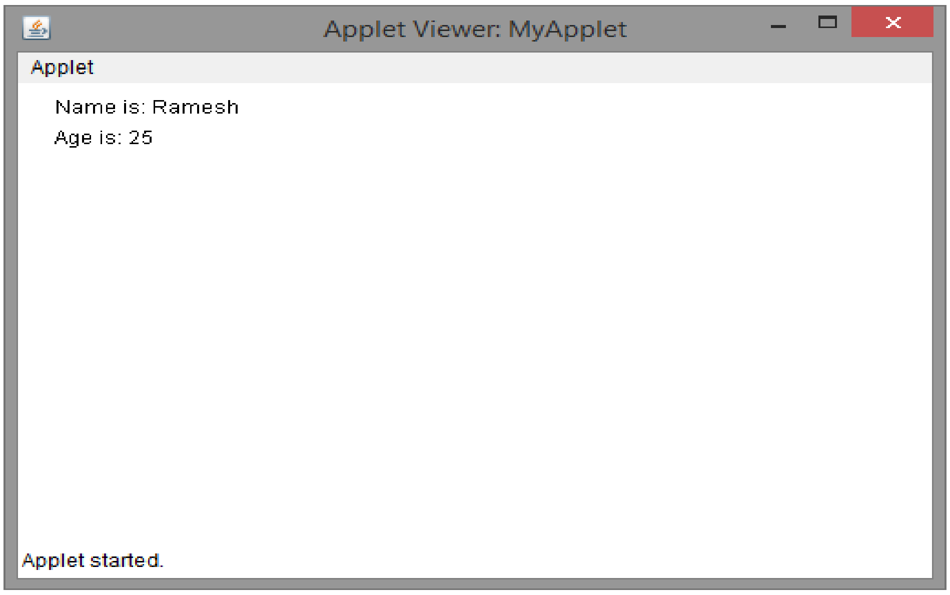
Let's look at a sample program which demonstrates the <param> HTML tag and the *getParameter( )* method:

Ex.Prg:

```
// Prg to demonstrate on Applets

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

```
/*
      <applet code="MyApplet" height="300" width="500">
             <param name="name" value="Ramesh" />
             <param name="age" value="25" />
      </applet>
*/
public class MyApplet extends Applet
{
      String n;
      String a;
      public void init()
      {
             n = getParameter("name");
             a = getParameter("age");
      }
      public void paint(Graphics g)
      {
             g.drawString("Name is: " + n, 20, 20);
             g.drawString("Age is: " + a, 20, 40);
      }
}
```

Output of the above program is as follows:



```
Name is: Ramesh
Age is: 25
```

Applet started.

6. a) List different layout managers that AWT supports? Write a Java program for one layout manager.

The Layout Managers are used to arrange components in a particular manner.

LayoutManager is an interface that is implemented by all the classes of layout managers.

- There are following classes that represents the layout managers:

Sr. No.	LayoutManager & Description
1	**BorderLayout** The borderlayout arranges the components to fit in the five regions: east, west, north, south and center.
2	**CardLayout** The CardLayout object treats each component in the container as a card. Only one card is visible at a time.
3	**FlowLayout** The FlowLayout is the default layout.It layouts the components in a directional flow.
4	**GridLayout** The GridLayout manages the components in form of a rectangular grid.
5	**GridBagLayout** This is the most flexible layout manager class.The object of GridBagLayout aligns the component vertically,horizontally or along their baseline without requiring the components of same size.

Ex.Prg:

```
//Prg to demonstrate on creating GridLayout
```

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

public class MyGridLayout
{
      JFrame f;
      MyGridLayout()
      {
             f=new JFrame();

             JButton b1=new JButton("1");
             JButton b2=new JButton("2");
             JButton b3=new JButton("3");
             JButton b4=new JButton("4");
             JButton b5=new JButton("5");
        JButton b6=new JButton("6");
        JButton b7=new JButton("7");
             JButton b8=new JButton("8");
        JButton b9=new JButton("9");

             f.add(b1);f.add(b2);f.add(b3);
             f.add(b4);f.add(b5);f.add(b6);
             f.add(b7);f.add(b8);f.add(b9);

             f.setLayout(new GridLayout(3,3));
             //setting grid layout of 3 rows and 3 columns

             f.setSize(300,300);
             f.setVisible(true);
      }
      public static void main(String[] args)
      {
             new MyGridLayout();
      }
}
```

6. b) Explain about mouse events with example.

The Java MouseListener is notified whenever you change the state of mouse. It is notified against MouseEvent. The MouseListener interface is found in java.awt.event package. It has five methods.

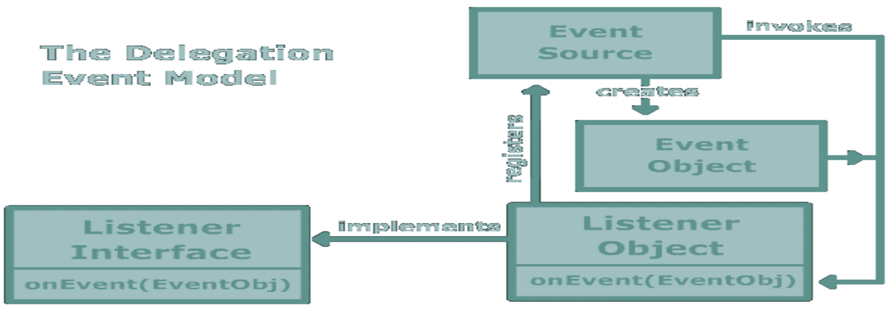
Ex. Prg:

```
//Prg. to Demonstrate on handling MouseEvents by MouseListener
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
       <applet code="MouseEvents.class" width=300 height=100>
       </applet>
*/
public class MouseEvents extends Applet implements MouseListener,
MouseMotionListener
{
       String msg = " ";
       int mouseX = 0, mouseY = 0; // coordinates of mouse
       public void init()
       {
              setBackground(Color.pink);
              addMouseListener(this);
              addMouseMotionListener(this);
       }
       // Handle mouse clicked.
       public void mouseClicked(MouseEvent me)
       {
              setBackground(Color.blue);
              // save coordinates
              mouseX = 0;
              mouseY = 10;
              msg = "Mouse clicked.";
              repaint();
       }
              // Handle mouse entered.
       public void mouseEntered(MouseEvent me)
       {
              setBackground(Color.cyan);
              // save coordinates
              mouseX = 0;
              mouseY = 10;
              msg = "Mouse entered.";
              repaint();
       }
              // Handle mouse exited.
       public void mouseExited(MouseEvent me)
       {
              setBackground(Color.red);
              // save coordinates
              mouseX = 0;
              mouseY = 10;
              msg = "Mouse exited.";
              repaint();
       }
              // Handle button pressed.
       public void mousePressed(MouseEvent me)
       {
              setBackground(Color.orange);
              // save coordinates
              mouseX = me.getX();
              mouseY = me.getY();
              msg = "Mouse Pressed";
              repaint();
       }
              // Handle button released.
       public void mouseReleased(MouseEvent me)
       {
              setBackground(Color.green);
              // save coordinates
              mouseX = me.getX();
              mouseY = me.getY();
              msg = "Mouse Released";
              repaint();
       }
              // Handle mouse dragged.
       public void mouseDragged(MouseEvent me)
       {
              // save coordinates
              mouseX = me.getX();
              mouseY = me.getY();
              msg = "Mouse Dragging";
       showStatus("Dragging mouse at " + mouseX + ", " + mouseY);
              repaint();
       }
       // Handle mouse moved.
       public void mouseMoved(MouseEvent me)
       {
              // show status
showStatus("Moving mouse at " + me.getX() + ", " + me.getY());
       }

       // Display msg in applet window at current X,Y location.
       public void paint(Graphics g)
       {

              g.drawString(msg, mouseX, mouseY);
       }
}
```

7. a) Explain about the delegation event model.

The modern approach to handling events is based on the delegation event model. In this model, an event is sent to component form which it originated. The component registers a listener object with the program. The listener processes the event and then returns.

For e.g.: When you click a button, the action to be performed is handled by an object registered to handle the button click event. The following diagram illustrates the delegation event model



Every event has a corresponding listener interface that specifies the methods that are required to handle the event. Event object are sent to registered listeners. To enable a component to handle events, you must register an appropriate listener for it.

7. b) What is an adapter class? Explain its purpose and functionality.

Java adapter classes *provide the default implementation of listener* *interfaces*. If you inherit the adapter class, you will not be forced to provide the implementation of all the methods of listener interfaces. So it *saves code*.The adapter classes are found in **java.awt.event**, **java.awt.dnd** and **javax.swing.event** packages.

- The Adapter classes with their corresponding listener interfaces are given below.

Adapter class	**Listener interface**
WindowAdapter	WindowListener
KeyAdapter	KeyListener
MouseAdapter	MouseListener
MouseMotionAdapter	MouseMotionListener
FocusAdapter	FocusListener
ComponentAdapter	ComponentListener
ContainerAdapter	ContainerListener
HierarchyBoundsAdapter	HierarchyBoundsListener

Ex. Prg:

```
// Prg. To create a Frame and then close it using Window Adapter
Class

import java.awt.*;
import java.awt.event.*;
class MyFrame4 extends Frame
{

       public static void main(String args[])
       {

              // create a Frame with title

                     MyFrame4 f=new MyFrame4();

              // set title to Frame

                     f.setTitle("My AWT Frame");
              // set the size of the Frame

                     f.setSize(300,250);

              // Dispaly the Frame

                     f.setVisible(true);

              // Close the Frame

                     f.addWindowListener(new Myclass());
       }
}
class Myclass extends WindowAdapter
{
              public void windowClosing(WindowEvent e)
              {
                     System.exit(0);
              }
}
```

UNIT – IV

8. a) Write a Java program to demonstrate check boxes in swing.

JCheckBox

The JCheckBox class is used to create a checkbox. It is used to turn an option on (true) or off (false). Clicking on a CheckBox changes its state from "on" to "off" or from "off" to "on ".It inherits JToggleButton class.

JCheckBox class declaration

Let's see the declaration for javax.swing.JCheckBox class.

Ex.Prg:

```
      // Prg. To creating check box and text area box.

import javax.swing.*;
import java.awt.*;      // Container class
import java.awt.event.*;

class Check extends JFrame implements ActionListener
{

      // vars
      JCheckBox cb1,cb2;
      JTextArea ta;
      String msg=" ";

      Check()
      {
             //create the content pane

             Container c=getContentPane();

             // set flow layout to content pane

             c.setLayout(new FlowLayout());

      //create a text area with 10 rows and 20 chars per row

             ta=new JTextArea(10,20);
             ta.setBackground(Color.red);

      //create two checkboxes

             cb1=new JCheckBox("C++",true);
             cb2=new JCheckBox("JAVA");

      // add the checkboxes,textarea to the cotainer
             c.add(cb1);
             c.add(cb2);
             c.add(ta);
// add action listeners. we need not add listener to text area
      //since the user clicks on the checkboxes only

             cb1.addActionListener(this);
             cb2.addActionListener(this);

// close the frame upon clicking

      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
public void actionPerformed(ActionEvent ae)
{
             // know which components are selected by user

             if(cb1.getModel().isSelected())

                    msg+="\nC++";

             if(cb2.getModel().isSelected())

                    msg+="\nJAVA";

      // display the selected message message in text area
                    ta.setText(msg);

      // reset the message to empty string
                    msg=" ";
      }
      public static void main(String args[])
      {
             // create the frame

             Check cr =new Check();

             // set title for the frame

             cr.setTitle("My checkboxes and Radio buttons");

             // Set the size 300 X 300 px

             cr.setSize(500, 400);

             cr.setVisible(true);
      }
}
```

8. b) Differentiate between AWT and 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**.

9. a) Write the steps involved in JDBC connectivity process with example.

JDBC stands for **J**ava **D**ata**b**ase **C**onnectivity, which is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases, especially data stored in a Relational Database.

JDBC works with Java on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX.

Methods/Stages used to access database:

There are 5 steps/stages to connect any java application with the database in java using JDBC. They are as follows:

- Register the driver
- Creating connection/Connecting to database
- Creating statement/Preparing SQL statement in java
- Executing queries/Retrieving the results
- Closing connection

Register the driver

We should first declare a driver which is going to be used for communication which is going to be used for communication with the database server in java program.

```
DriverManager.registerDriver(new sun.jdbc.odbc.jdbcodbcDriver());
```

Create the connection

In this stage, we establish a connection with a specific dadabase through the driver which is already registered in the previous step.

```
Connection con=DriverManager.getConnection
                                       ("jdbc:oracle:oradsn","system","password");
```

Create the connection/Preparing SQL statement in java

We should create an SQL statement in our java program using any of the interfaces like Statement, PreapredStatement etc., which are available in java.sql package.

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

Execute Queries/Retrieving the results

The results obtained by executing the SQL statements can be stored in an object with the help of interfaces like ResultSet,ResultSetMetaData etc.,

```
ResultSet rs=stmt.executeQuery( “select *from employee”);
```

Closing the Connection

We should close the connection between java program and the database by using close( ) metod of Connection class.

```
Connection.close( );
```

9. b) Discuss about JDBC drivers.

JDBC stands for **J**ava **D**ata**b**ase **C**onnectivity, which is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases, especially data stored in a Relational Database.

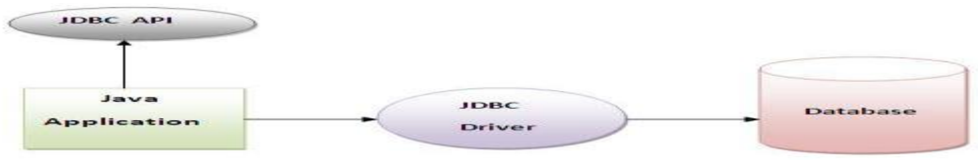
JDBC works with Java on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX.

JDBC Driver is a software component that enables java application to interact with the database. There are 4 types of JDBC drivers:

5. JDBC-ODBC bridge driver
6. Native-API driver (partially java driver)
7. Network Protocol driver (fully java driver)
8. Thin driver (fully java driver)

JDBC-ODBC bridge driver

The JDBC-ODBC bridge driver uses ODBC driver to connect to the database. The JDBC-ODBC bridge driver converts JDBC method calls into the ODBC function calls. This is now discouraged because of thin driver.



Native-API driver

The Native API driver uses the client-side libraries of the database. The driver converts JDBC method calls into native calls of the database API. It is not written entirely in java.

Network Protocol driver

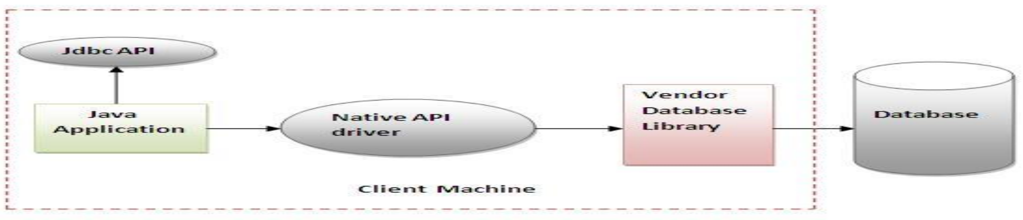


Figure- Native API Driver

Network Protocol driver

The Network Protocol driver uses middleware (application server) that converts JDBC calls directly or indirectly into the vendor-specific database protocol. It is fully written in java.

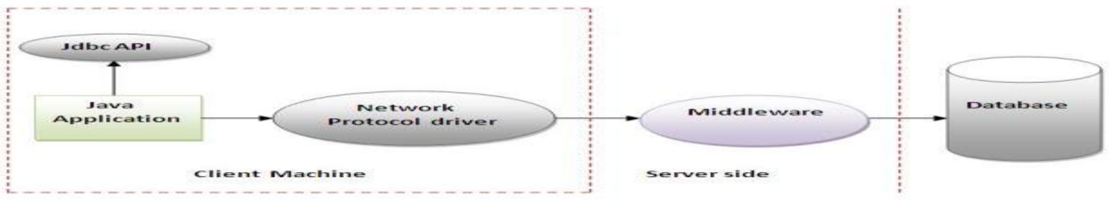


Figure- Network Protocol Driver

Thin driver

The thin driver converts JDBC calls directly into the vendor-specific database protocol. That is why it is known as thin driver. It is fully written in Java language.

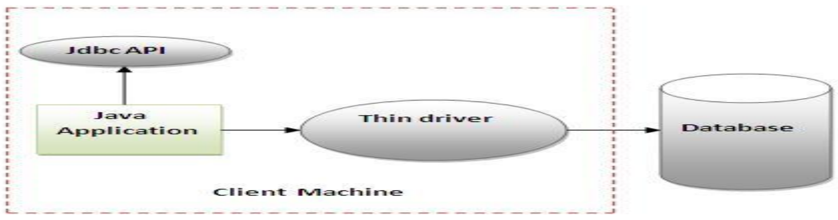


Figure- Thin Driver

- - - - -