18CS IT 304

Hall Ticket Number:

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

November, 2019 **Common for CSE & IT**

Third Semester **OBJECT ORIENTED PROGRAMMING**

Time: Three Hours **Maximum:** 50 Marks

Answer Question No.1 compulsorily. (10X1 = 10 Marks)

Answer ONE question from each unit. (4X10=40 Marks)

1. Answer all questions (10X1=10 Marks)
   a) Write the advantages of this keywords.
   b) What is an object and write its syntax?
   c) What is an abstract class?
   d) Define Collection.
   e) What is the use of final keyword?
   f) How to create a user defined Exception?
   g) Define Thread.
   h) Write the syntax for HTML Applet Tag.
   i) What is Delegation event model?
   j) List the constructors for Button class.

UNIT I

2. a) Explain the following OOP principles in detail. 5M
   Encapsulation, Inheritance, Polymorphism.
   b) Define method overloading and explain with suitable examples. 5M

(OR)

3. a) Explain about static keyword in java with an example program. 5M
   b) Define method overriding and explain with suitable examples. 5M

UNIT II

4. What is inheritance? Explain different types of inheritance with suitable examples. 10M

(OR)

5. a) Write a shot notes on interface. Write a java program to demonstrate the multiple inheritance. 5M
   b) Explain any five string handling methods with an example program. 5M

UNIT III

6 a) Define Exception. Explain Exception handling mechanism with an example program. 5M
   b) Explain the concept of synchronization in Java with an example program. 5M

(OR)

7 a) Define deadlock? Write a java program to create deadlock situation java. 5M
   b) Write a java program to copy the content from one file to another file using Byte stream classes. 5M

UNIT IV

8 a) Define Applet, Explain applet life cycle methods with an example. 5M
   b) Write a Java application to handle Mouse Events. 5M

9 a) Discuss about FlowLayout Mangers to place the components in an Applet. 5M
   b) Explain the creation of JTables with an example program. 5M

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

November, 2019 **Common for CSE & IT**

Third Semester **OBJECT ORIENTED PROGRAMMING**

Time: Three Hours **Maximum:** 50 Marks

Answer Question No.1 compulsorily. (10X1 = 10 Marks)

Answer ONE question from each unit. (4X10=40 Marks)

Scheme of Evaluation & Solutions

1. Answer all questions (10X1=10 Marks)

a) Write the advantages of this keywords.

- this() can be used to invoke current class constructor.
- this can be used to resolve the ambiguity between instance variable and local variable.
- this can be passed as an argument in the method call.
- this can be used to return the current class instance from the method.

[Any two advantages]

b) What is an object and write its syntax?

Object is instance of class that is process of allocating the memory to class at run time is called object.

Syntax: ClassName object-name = new ClassName ( );

c) What is an abstract class?

A class contain at least one abstract method is called abstract class.

d) Define Collection.

Group of objects is called collection.

e) What is the use of final keyword?

- It is used to create constant variables, to restrict method overriding, to restrict inheritance.

f) How to create a user defined Exception?

1. By extends Exception class.
2. By using throw keyword.

g) Define Thread.

A thread is a lightweight process.

h) Write the syntax for HTML Applet Tag.

< APPLET
[CODEBASE = *codebaseURL*]
CODE = *appletFile*
[ALT = *alternateText*]
[NAME = *appletInstanceName*]
WIDTH = *pixels* HEIGHT = *pixels*
[ALIGN = *alignment*]
[VSPACE = *pixels*] [HSPACE = *pixels*]
>
[< PARAM NAME = *AttributeName* VALUE = *AttributeValue*>]
[< PARAM NAME = *AttributeName2* VALUE = *AttributeValue*>]
. . .
[*HTML Displayed in the absence of Java*]
</APPLET>

i) What is Delegation event model?

a *source* generates an event and sends it to one or more *listeners.* In this scheme, the listener simply waits until it receives an event. Once an event is received, the listener processes the event and then returns.

j) List the constructors for Button class.

Button( ) throws HeadlessException
Button(String *str*) throws HeadlessException

2. a) Explain the following OOP principles in detail. 5M
Encapsulation, Inheritance, Polymorphism.

Data Encapsulation(1.5 M)

- The process of grouping or wrapping or binding the data and code (functions) into a single unit.
- The advantage of encapsulation is that data cannot access directly. It is only accessible through the member functions of the class.
- The data encapsulation is achieved by using class.
- Syntax

```
class className
{
        Data
        Code
}
```

Inheritance(1.5 M)

- Inheritance is a mechanism to obtaining the one class properties into another class and it creates new class from existing class.
- The class which is giving properties is called **base class** (or) **parent class** (or) **super class**.
- The class which is taking properties is called **derived class** (or) **child class** (or) **sub class**.
- Inheritance is basically used for reducing the overall code size of the program (**code reusability**).
- Syntax

```
class Base_class_Name extends Derived_Class_Name
{
        Data
        Code
}
```

Polymorphism(1.5 M)

- Polymorphism is basic and important concept of OOPS. Polymorphism, a Greek term, means the ability to take more than one form.
- Polymorphism can be achieved by **overloading** and **overriding** concepts
- Compile time and runtime polymorphism with explanation.

[A small example] (0.5 M)

2. b) Define method overloading and explain with suitable examples. 5M

Description: 1M Example: 4M

Method overloading:

- A class contain **Two or more methods having same method name and but different in number of parameters and/or type of parameters** is called method overloading.
- The function would perform different operations depending upon the argument list in the function call.

Example program:

```
class Shapes  {
    double  a;
    void Area(double r)
    {
            a = 3.4 * r * r;
            System.out.println("Aria of circle :"+a);
    }
    void Area(double b, double h)
    {
            a = 0.5 * b * h;
            System.out.println("Aria of Triangle :"+ a);
    }
        void Area(double b1, double b2, double h)
    {
            a = h * (b1 + b2) / 2;
            System.out.println("Aria of Trapezoidal :"+a);
    }
}
class MethodOverloading  {
    Public static void main(String args[])
        {
            shapes s=new Shapes( );
            s.Area(6.4);            //area of circle
            s.Area(3.2, 6.7);       //area of Triangle
            s.Area(4.2, 3.1, 5.0); //area of Trapezoidal
        }
}
```

Output:

Area of Circle : 128.614
Area of Triangle : 10.72
Area of Trapezoidal :18.25

[Any valid example program]

3. a) Explain about static keyword in java with an example program. 5M

Description: 3M Example: 2M

static keyword:

- The **static keyword** in Java is used for memory management.
- We can apply java static keyword with **variables, methods, blocks and nested class**.
- The static keyword belongs to the class than an instance (**object**) of the class.
- When a member is declared static, it can be accessed before any objects of its class are created, and without reference to any object.

1) static variable (class variable)

1. A variable that declared with static keyword is known as a **static variable**.
2. When a variable is declared as static, then a **single copy of variable is created** and shared among all objects at class level.
3. The static variable gets memory only once in the class area at the time of class loading.
4. Syntax:**static** datatype variablename**;**

2) static method (class method)

1. A method that declared with static keyword is known as a **static method**.
2. The most common example of a static method is ***main( )*** method.
3. **Restrictions:**
   - They can only directly call other **static** methods.
   - They can only directly access **static** variable.
   - They cannot refer to **this** or **super** in any way.

Syntax:static returntype methodname(par-list) {
Body;
}

3) static block

1. static block used to initialize the static data member.
2. It is executed before the main method at the time of class loading.

Syntax: **static** {
Body;
}

Example: Demonstrate static variables, methods, and blocks.

```
class UseStatic {
    static int a = 3;
    static int b;
    static {
        System.out.println("Static block initialized.");
        b = a * 4;
    }
    static void meth(int x) {
            System.out.println("x = " + x);
            System.out.println("a = " + a);
            System.out.println("b = " + b);
    }
    public static void main(String args[])
    {
            UseStatic.meth(42);
    }
}
```

Note: the static members of outside class can be accessed through **class name**.

[Any related program]

3. b) Define method overriding and explain with suitable examples. 5M

Description: 2M Example: 3M

Method Overriding:

- A derived class has the same method heading (return type, name, and parameters) as a method in its base class, then the method in the derived class is said to *override* the method in the base class.

Dynamic method dispatch:

- A mechanism in which call the derived class override method by using the base class reference is called dynamic method dispatch.
- Dynamic method dispatch resolved at runtime polymorphism.

Rules for method overriding:

- Base class and derived class method headings must be same.
- If a method cannot be inherited then it cannot be overridden.
- Instance methods can be overridden only if they are inherited by the subclass.
- A method declared static cannot be overridden but can be re-declared.
- A method declared final cannot be overridden.
- Constructors cannot be overridden.
- The method in base is need not to inherit in derived class.

Example: // Dynamic Method Dispatch

```
class A {
        void display() {
                System.out.println("Inside A's display method");
        }
}
class B extends A {
        void display() {
                System.out.println("Inside B's display method");
        }
}
class Dispatch {
        public static void main(String args[]) {
                A obj1 = new A();
                B obj2 = new B();
                A ref;
                ref = obj1;
                ref.display();
                ref = obj2;
                ref.display();
        }
}
```

Output:

```
Inside A's display method
Inside B's display method
Inside C's display method
```

[Any related program]

4. What is inheritance? Explain different types of inheritance with suitable examples. 10M
Description: 3 M Diagram: 2 M Example: 5 M

Inheritance Basics: Inheritance is a mechanism to obtaining the one class properties into another class and it creates new class from existing class.

Types of inheritance:there are 5 types of inheritances.

1. **Single inheritance**
2. **Multiple inheritance**
3. **Hierarchical inheritance**
4. **Multilevel inheritance**
5. **Hybrid inheritance**

1. **Single inheritance**

Single derived class with only one base class, is called single inheritance.

(Or)

A class inherits members of only one class is called single inheritance.

2. **Multiple inheritance**

A derived class with several base class, is called multiple inheritance.

(Or)

A class inherits members of two or more classes is called multiple inheritance.

Note: Single inheritance achieved through interface, not classes

3. **Hierarchical inheritance**

Several derived classes with only one base class is called hierarchical inheritance.

(Or)

Two or more classes inherits members of only one class is called hierarchical inheritance.

4. **Multilevel inheritance**

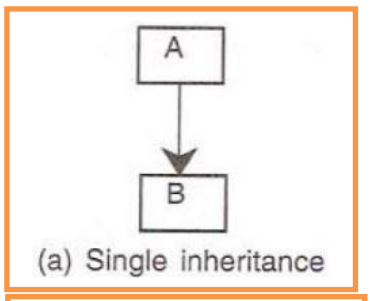
A derived class with another derived class is called multilevel inheritance.

(Or)

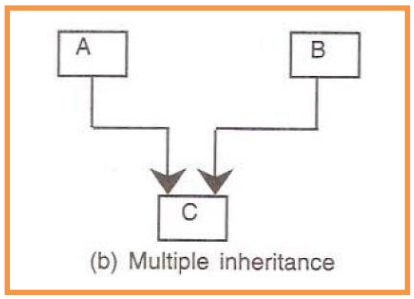
A class inherits members another derived class is called multilevel inheritance.

5. **Hybrid inheritance**

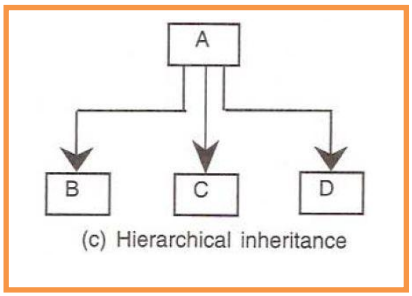
The combination of two or more inheritances is called hybrid inheritance.



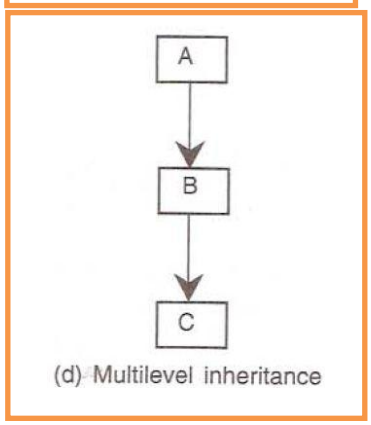
(a) Single inheritance



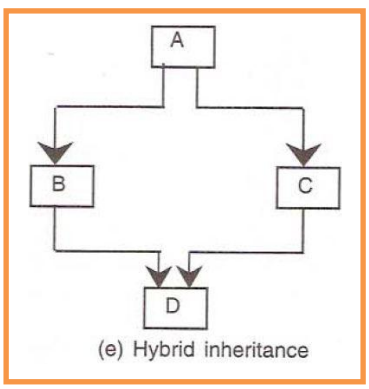
(b) Multiple inheritance



(c) Hierarchical inheritance



(d) Multilevel inheritance



(e) Hybrid inheritance

[Any related program for each mechanism]

5. a) Write a shot note on interface. Write a java program to demonstrate the multiple inheritance. 5M
Description: 2M Example: 3M

Interfaces: An **interface is** a collection of static constants and abstract methods.

- **Syntax:** defining interface

```
access interface name{
        return-type method-name1(parameter-list);
        type final-varname1= value;
        //...
        return-type method-nameN(parameter-list);
        type final-varnameN = value;
}
```

Characteristics:

- Interfaces specify what a class must do and not how. It is the blueprint of the class.
- Like **abstract classes**, interfaces **cannot** be used to create objects.
- Interface methods do not have a body - the body is provided by the "implement" class on implementation of an interface, you must override all of its methods
- Interface methods are by default **public** and **abstract**
- Interface attributes are by default **public**, **static** and **final**
- An interface cannot contain a constructor (as it cannot be used to create objects)

Multiple interface: java program to demonstrate the multiple inheritance

```
interface Sports{
        void SportsInfo( );
}
interface Exam{
        void Marks( );
}
class Student implements Sports, Exam{
        void SportsInfo( )          {
                System.out.println("He is playing cricket");
                System.out.println("Position: All-rounder");
        }
        void Marks( )    {
                int s1=60, s2=60, s3=60, s4=60, s5=60, s6=60;
                float per;
                per=( (s1+s2+s3+s4+s5+s6)/600) *100'
                System.out.println("Percentage :"+per);
        }
        public static void main(String args[ ])     {
                Student obje=new Student();
                obj. SportsInfo( );
                obj.Marks( )
        }
}
```

Output:

```
He is playing cricket
Position: All-rounder
Percentage :60.00
```

[Any related example]

5 b) Explain any five string handling methods with an example program. 5M

Description for each method caries on 0.5 marks
Example program for string handling function 2.5 marks

6 a) Define Exception. Explain Exception handling mechanism with an example program. 5M

Description: 3M Example: 2M

- Exception is a runtime error.
- Exception is an event that interrupt the flow of program execution.
- In java, exceptions are handled by using 5 keywords

  1) try 2) catch 3) throw 4) throws 5) finally

1. **try:** It contains program statements that you want to monitor for exceptions. If an exception occurs within the **try** block, it is thrown.
2. **catch:** Your code can catch this exception (using **catch**) and handle it in some rational manner.System-generated exceptions are automatically thrown by the Java runtime system.
3. **throw:** To manually throw an exception, use the keyword **throw**.
4. **thrown:** Any exception that is thrown out of a method must be specified as such by a **throws** clause.
5. **finally:** Any code that absolutely must be executed after a **try** block completes is put in a **finally** block.

Syntax:

```
try {
        // block of code to monitor for errors
}
catch (ExceptionType1 exOb) {
        // exception handler for ExceptionType1
}
catch (ExceptionType2 exOb) {
        // exception handler for ExceptionType2
}
// ...
finally {
        // block of code to be executed after try block ends
}
```

Here, *ExceptionType* is the type of exception that has occurred.

Example: demonstrate the **try, catch, and finally blocks**.

```
import java.util.*;
class ExceptionDemo
{
        public static void main(String ar[])
{
                        try     {
                                String lan[]={"C","C++","C#","JAVA"};
                                for(int i=0;i<10;i++)
                                        System.out.println("Language "+(i+1)+" :"+lan[i]);
                        }
                        catch(Exception ie)     {
                                System.out.println("Index was not found");
                        }
                        finally                 {
                                System.out.println("Final block is executed");
                        }

        }
}
```

[Any related example]

6 b) Explain the concept of synchronization in Java with an example program. 5M

Description: 1M Example: 4M

- 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.
- The synchronization is mainly used to to prevent thread interference and to prevent consistency problem.

```
// This program uses a synchronized block.
class Callme {
        void call(String msg) {
                System.out.print("[" + msg);
                try {
                        Thread.sleep(1000);
                }
                catch (InterruptedException e) {
                        System.out.println("Interrupted");
                }
                System.out.println("]");
        }
}
class Caller implements Runnable {
        String msg;
        Callme target;
        Thread t;
        public Caller(Callme targ, String s) {
                target = targ;
                msg = s;
                t = new Thread(this);
                t.start();
        }
        // synchronize calls to call()
        public void run() {
                synchronized(target) { // synchronized block
                        target.call(msg);
                }
        }
}
class Synch1 {
        public static void main(String args[]) {
                Callme target = new Callme();
                Caller ob1 = new Caller(target, "Hello");
                Caller ob2 = new Caller(target, "Synchronized");
                Caller ob3 = new Caller(target, "World");
                // wait for threads to end
                try {
                        ob1.t.join();
                        ob2.t.join();
                        ob3.t.join();
                }
                catch(InterruptedException e) {
                        System.out.println("Interrupted");
                }
        }
}
```

[Any related example]

7 a) Define deadlock? Write a java program to create deadlock situation java. 5M

Description: 1M Example: 4M

Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.

Example:

```
class Shared {
   synchronized void Resource1(Shared s2)     {
      System.out.println("Resource1 is used");
      try {            Thread.sleep(1000);        }
     catch (InterruptedException e) {
         System.out.println(e);
      }
      s2.Resource2(this);
      System.out.println("Resource1 is end");
   }
   synchronized void Resource2(Shared s1)     {
      System.out.println("Resource2 is used");
      try {   Thread.sleep(1000);
      } catch (InterruptedException e) {
         System.out.println(e);
      }
      s1.Resource1(this);
      System.out.println("Resource2 is end");
   }
}
class Thread1 extends Thread {
   private Shared s1, s2;
   public Thread1(Shared s1, Shared s2) {
      this.s1 = s1;
      this.s2 = s2;
   }
   public void run()  {
      s1.Resource1(s2);
   }
}
class Thread2 extends Thread {
   private Shared s1, s2;
   public Thread2(Shared s1, Shared s2) {
      this.s1 = s1;
      this.s2 = s2;
   }
   public void run() {
      s2.Resource2(s1);
   }
}
public class DeadlockDemo3 {
   public static void main(String[] args)     {
      Shared s1 = new Shared();
      Shared s2 = new Shared();
      Thread1 t1 = new Thread1(s1, s2);
      t1.start();
      Thread2 t2 = new Thread2(s1, s2);
      t2.start();
   }
}
```

[Any related example]

7. b) Write a java program to copy the content from one file to another file using Byte stream classes. **5M**

Declaration: 2M **Implementation: 3M**

```
import java.io.*;
class FileDemo
{
	public static void main(String args[]) throws IOException
	{
		int ch;
		FileInputStream fin=null;
		FileOutputStream fout=null;
		try
		{
			fin=new FileInputStream("D:/java/alphabets.txt");
			fout=new FileOutputStream("D:/java/newalphabets.txt");
			ch = fin.read();
			while(ch !=-1)
			{
				fout.write((char)ch);
				ch = fin.read();
			}
			System.out.println("Copy completed");
		}
		 catch(Exception e)
		{
			System.out.println("File is not found");
		}
		finally
		{
			fin.close();
			fout.close();
		}
	}
}
```

Output:

Copy completed

[With window]

[Any related example]

UNIT IV

8 a) Define Applet. Explain applet life cycle methods with an example. 5M

Description: 3M Example: 2M

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

Lifecycle of Java Applet

Applet is initialized.

Applet is started.

Applet is painted.

Applet is stopped.

Applet is destroyed

Four methods in the Applet class gives you the framework on which you build any serious applet –

init − This method is intended for whatever initialization is needed for your applet. It is called after the param

tags inside the applet tag have been processed.

start − This method is automatically called after the browser calls the init method. It is also called whenever

the user returns to the page containing the applet after having gone off to other pages.

stop − This method is automatically called when the user moves off the page on which the applet sits. It can,

therefore, be called repeatedly in the same applet.

destroy − This method is only called when the browser shuts down normally. Because applets are meant to

live on an HTML page, you should not normally leave resources behind after a user leaves the page that

contains the applet.

paint − Invoked immediately after the start() method, and also any time the applet needs to repaint itself in the

browser. The paint() method is actually inherited from the java.awt.

```
import java.applet.Applet;
import java.awt.Graphics;
import java.awt.*;
/*<applet code="AppletLifeCycle.class" width="350" height="150"></applet>*/
public class AppletLifeCycle extends Applet
{
        public void init() {
                setBackground(Color.CYAN);
                System.out.println("init() called");
        }
        public void start(){
                System.out.println("Start() called");
        }
        public void paint(Graphics g)     {
                System.out.println("Paint(() called");
        }
        public void stop()          {
                System.out.println("Stop() Called");
        }
        public void destroy()     {
                System.out.println("Destroy() Called");
        }
}
```

Output

[Any related example]

8 b) Write a Java application to handle Mouse Events. 5M

Declaration: 2M Implementation: 3M

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
<applet code="MouseEventDemo" width=500 height=400>
</applet>
*/
public class MouseEventDemo extends Applet implements MouseListener
{
        String msg="";
        String key="";
        public void mouseEntered(MouseEvent m)
        {
                msg = "Mouse Enterd";
                repaint();
        }
        public void mouseExited(MouseEvent m)
        {
                msg = "Mouse Exited";
                repaint();
        }
        public void mouseClicked(MouseEvent m)
        {
                msg = "Mouse clicked";
                repaint();
        }
        public void mousePressed(MouseEvent m)
        {
                showStatus("Mouse pressed");
        }
        public void mouseReleased(MouseEvent m)
        {
                showStatus("Mouse Released");
        }
        public void init()
        {
                addMouseListener(this);
                addMouseMotionListener(this);
        }
        public void paint(Graphics g)    {
                g.drawString(msg,10,40);
        }
}
```

Output

[Any related program]

9 a) Discuss about FlowLayout Mangers to place the components in an Applet. 5M

Description: 2.5 M Example: 2.5 M

FlowLayout is used to arrange components in a sequence one after the other. The default layout of applet and panel is FlowLayout.FlowLayout is the default layout manager.

Constructors:

1. **FlowLayout():** It will Construct a new FlowLayout with centered alignment.The horizontal and vertical gap will be 5 pixels.
2. **FlowLayout(int align) :** It will Construct a new FlowLayout with given alignment.The horizontal and vertical gap will be 5 pixels.
3. **FlowLayout(int align, int HorizontalGap, int VerticalGap):** It will Construct a new FlowLayout with given alignment, the given horizontal and vertical gap between the components.

Methods:

1. **setTitle(String Text):** This Method is used to set Title of JFrame. The title you want to set is passed as a string.
2. **getAlignment():** Returns the alignment for this layout.
3. **setAlignment(int align):** used to set the alignment for this layout.
4. **removeLayoutComponent(Component comp):** Used to remove the component passed as argument from the layout.

Below programs will illustrate the Example of FlowLayout in java.

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*<applet code="FlowLayoutDemo" width=240 height=200></applet>*/
public class FlowLayoutDemo extends Applet implements ItemListener {
        String msg = "";
        Checkbox windows, android, solaris, mac;
        public void init() {
        // set left-aligned flow layout
                setLayout(new FlowLayout(FlowLayout.LEFT));
                windows = new Checkbox("Windows", null, true);
                android = new Checkbox("Android");
                solaris = new Checkbox("Solaris");
                mac = new Checkbox("Mac OS");
                add(windows);
                add(android);
                add(solaris);
                add(mac);
                windows.addItemListener(this);
                android.addItemListener(this);
                solaris.addItemListener(this);
                mac.addItemListener(this);
        }
        public void itemStateChanged(ItemEvent ie) {
                repaint();
        }
        public void paint(Graphics g) {
                msg = "Current state: ";
                g.drawString(msg, 6, 80);
                msg = " Windows: " + windows.getState();
                g.drawString(msg, 6, 100);
                msg = " Android: " + android.getState();
                g.drawString(msg, 6, 120);
                msg = " Solaris: " + solaris.getState();
                g.drawString(msg, 6, 140);
                msg = " Mac: " + mac.getState();
                g.drawString(msg, 6, 160);
        }
}
```

[Any related example]

9 b) Explain the creation of JTables with an example program. 5M

Description: 2.5 M Example: 2.5 M

- **JTable** is a component that displays rows and columns of data.
- You can drag the cursoron column boundaries to resize columns.
- **JTable** constructor:

JTable(Object *data*[ ][ ], Object *colHeads*[ ])

- A **JTable** can generate several different events. The two most fundamental to a table'soperation are **ListSelectionEvent** and **TableModelEvent**.

Here are the steps required to set up a simple **JTable** that can be used to display data:

1. Create an instance of **JTable**.
2. Create a **JScrollPane** object, specifying the table as the object to scroll.
3. Add the table to the scroll pane.
4. Add the scroll pane to the content pane.

```
// Demonstrate JTable.
import java.awt.*;
import javax.swing.*;
/* <applet code="JTableDemo" width=400 height=200></applet> */
public class JTableDemo extends JApplet {
        public void init() {
                try {
                        SwingUtilities.invokeAndWait(
                                new Runnable() {
                                        public void run() {
                                                makeGUI();
                                        }
                                }
                        );
                }
                catch (Exception exc) {
                        System.out.println("Can't create because of " + exc);
                }
        }
        private void makeGUI() {
                // Initialize column headings.
                String[] colHeads = { "Name", "Extension", "ID#" };
                // Initialize data.
                Object[][] data = {
                { "Gail", "4567", "865" },
                { "Ken", "7566", "555" },
                { "Viviane", "5634", "587" },
                { "Melanie", "7345", "922" },
                { "Anne", "1237", "333" },
                { "John", "5656", "314" },
                { "Matt", "5672", "217" },
                { "Claire", "6741", "444" },
                { "Erwin", "9023", "519" },
                { "Ellen", "1134", "532" },
                { "Jennifer", "5689", "112" },
                { "Ed", "9030", "133" },
                { "Helen", "6751", "145" }
                };
                JTable table = new JTable(data, colHeads);// Create the table.
                JScrollPane jsp = new JScrollPane(table);// Add the table to a scroll pane.
                add(jsp);// Add the scroll pane to the content pane.
        }
}
```

[Any related example]