STELLA MARY'S COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CS3381 OBJECT ORIENTED PROGRAMMING LAB MANUAL

(Regulation-2022)

Name of the Student: _____

Register Number: _____

Year & Branch: _____

Semester: _____

UNIVERSITY SYLLABUS

CS3381 OBJECT ORIENTED PROGRAMMING LABORATORY L T P C0 0 3 1.5

OBJECTIVES:

The student should be made to:

- To build software development skills using java programming for real-world applications.
- To understand and apply the concepts of classes, packages, interfaces, inheritance, exception handling and file processing.
- To develop applications using generic programming and event handling.

LIST OF EXPERIMENTS:

1. Solve problems by using sequential search, binary search, and quadratic sorting algorithms (selection,insertion).
2. Develop stack and queue data structures using classes and objects.
   Develop a java application with an Employee class with Emp_name, Emp_id, Address, Mail_id, Mobile_no as members. Inherit the classes, Programmer, Assistant Professor, Associate Professor and Professor from employee class. Add Basic Pay (BP) as the member of all the inherited classes with 97% of BP as DA, 10 % of BP as HRA, 12% of BP as PF, 0.1% of BP for staff club funds. Generate pay slips for the employees with their gross and net salary.
3. Write a Java Program to create an abstract class named Shape that contains two integers and an empty method named printArea(). Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape. Each one of the classes contains only the method printArea( ) that prints the area of the given shape.
   Solve the above problem using an interface.
4. Implement exception handling and creation of user defined exceptions.
5. Write a java program that implements a multi-threaded application that has three threads. First thread generates a random integer every 1 second and if the value is even, the second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of the cube of the number.
6. Write a program to perform file operations.
7. Develop applications to demonstrate the features of generics classes.
8. Develop applications using JavaFX controls, layouts and menus.
9. Develop a mini project for any application using Java concepts.

TOTAL: 45 PERIODS

LAB REQUIREMENTS:

Number of Students in batch: For a batch of 30 students

Operating Systems: Linux / Windows

Front End Tools: Eclipse IDE / Netbeans IDE

Course Outcomes:

CO1	Design and develop java programs using object oriented programming concepts
CO2	Develop simple applications using object oriented concepts such as package, exceptions
CO3	Implement multithreading, and generics concepts
CO4	Create GUIs and event driven programming applications for real world problems
CO5	Implement and deploy web applications using Java

SL.NO	DATE	EXPERIMENT NAME	REMARKS
1 a		LINEAR SEARCH	
1 b		BINARY SEARCH	
1 c		SELECTION SORT	
1 d		INSERTION SORT	
2 a		STACK IMPLEMENTATION	
2 b		QUEUEIMPLEMENTATION	
3		EMPLOYEE PAYROLL	
4		ABSTRACT CLASS EXAMPLE	
5		INTERFACES - DEVELOPING USER DEFINED INTERFACES	
6		HANDLING USERDEFINED EXCEPTION	
7		MULTITHREAD CONCEPT	
8		FILE CONCEPT	
9		FIND MAXIMUM ELEMENT USING GENERIC FUNCTION	
10		MINI PROJECT: TRAFFIC LIGHT SIMULATION:	

DATE:

1.SOLVE PROBLEMS BY USING SEQUENTIAL SEARCH, BINARY SEARCH, AND QUADRATIC SORTING ALGORITHMS (SELECTION, INSERTION)

1.a .LINEAR SEARCH

AIM:

To write java program to implement linear searching algorithm.

ALGORITHM:

Step1: Start with the first item in the list

Step2: Compare the current item to the key

Step 3: If the current value matches the key then update with the location

Step 4: Else repeat from 2.

PROGRAM:

```
import java.util.Scanner;
class LinearSearchExample2
{
  public static void main(String args[])
  {
    int c, n, search, array[];
    Scanner in = new Scanner(System.in);
    System.out.println("Enter number of elements");
    n = in.nextInt();
    array = new int[n];
    System.out.println("Enter those " + n + " elements");
    for (c = 0; c < n; c++)
      array[c] = in.nextInt();
    System.out.println("Enter value to find");
    search = in.nextInt();
    for (c = 0; c < n; c++)
```

```
    {
      if (array[c] == search)     /* Searching element is present */
      {
         System.out.println(search + " is present at location " + (c + 1) + ".");
          break;
      }
    }
   if (c == n)  /* Element to search isn't present */
      System.out.println(search + " isn't present in array.");
  }
}
```

OUTPUT:

RESULT

Thus the java program to perform linear searching algorithms was executed and verified successfully.

DATE:

1.b.BINARY SEARCH

AIM:

To write java program to implement binary searching algorithm.

ALGORITHM:

Step1: Choose the middle element in the list

Step2: If it matches the middle element, its position in the list is returned.

Step 3: If the target value is less than or greater than the middle element, the search continues in the lower or upper half of the array, respectively, eliminating the other half from consideration

PROGRAM:

```
import java.util.Scanner;
class Main {
int binarySearch(int array[], int element, int low, int high) {
while (low <= high) {
int mid = low + (high - low) / 2;
if (array[mid] == element)
return mid;
if (array[mid] < element)
low = mid + 1;
else
high = mid - 1;
}
return -1;
}
public static void main(String args[])
{
Main obj = new Main();
```

```
int[] array = { 3, 4, 5, 6, 7, 8, 9 };
int n = array.length;
Scanner input = new Scanner(System.in);
System.out.println("Enter element to be searched:");
int element = input.nextInt();
input.close();
int result = obj.binarySearch(array, element, 0, n - 1);
if (result == -1)
System.out.println("Not found");
else
System.out.println("Element found at index " + result);
}
}
```

OUTPUT:

RESULT

Thus the java program to perform binary searching algorithms was executed and verified successfully.

DATE:

1.c.SELECTION SORT

AIM:

To write java program to implement selection sorting algorithm.

ALGORITHM:

Step 1 − Set Min_Index to 0

Step 2 − Search for the smallest element in the array

Step 3 − Swap with value with the element at the Min_Index

Step 4 − Increment Min_Index to point to next element

Step 5 − Repeat until the complete array is sorted

PROGRAM:

```
class SelectionSort
{
void sort(int array[])
{
int n = array.length;
for (int i = 0; i < n-1; i++)
{
int min_element = i;
for (int j = i+1; j < n; j++)
if (array[j] < array[min_element])
min_element = j;
int temp = array[min_element];
array[min_element] = array[i];
array[i] = temp;
}
}
void printarrayay(int array[])
{
```

```
int n = array.length;
for (int i=0; i<n; ++i)
System.out.print(array[i]+" ");
System.out.println();
}
public static void main(String args[])
{
SelectionSort ob = new SelectionSort();
int array[] = {15, 10, 99, 53, 36};
ob.sort(array);
System.out.println("Sorted arrayay");
ob.printarrayay(array);
}
}
```

OUTPUT:

RESULT

Thus the java program to perform selection sorting algorithms was executed and verified successfully.

DATE:

1.d.INSERTION SORT

AIM:

To write java program to implement insertion sorting algorithm.

ALGORITHM

Step1: Choose the first two elements in the list

Step2: If it is sorted leave as it is

Step 3: If not swap the element with the previous

Step 4: Check whether the elements in the sub list is sorted

Step 5: Repeat until list is sorted

PROGRAM:

```
import java.util.Scanner;
public class CodesCracker
{
public static void main(String[] args)
{
int n, i, j, element;
Scanner scan = new Scanner(System.in);
System.out.print("Enter the Size of Array: ");
n = scan.nextInt();
int[] arr = new int[n];
System.out.print("Enter " +n+ " Elements: ");
for(i=0; i<n; i++)
arr[i] = scan.nextInt();
for(i=1; i<n; i++)
{
element = arr[i];
for(j=(i-1); j>=0 && arr[j]>element; j--)
arr[j+1] = arr[j];
```

```
arr[j+1] = element;
}
System.out.println("\nThe new sorted array is: ");
for(i=0; i<n; i++)
System.out.print(arr[i]+ " ");
}
}
```

OUTPUT:

RESULT

Thus the java program to perform selection sorting algorithms was executed and verified successfully.

DATE:

2. DEVELOP STACK AND QUEUE DATA STRUCTURES USING CLASSES AND OBJECTS.

2 a) STACK IMPLEMENTATION

AIM:

To write a Java program to develop Stack data structure using classes and objects.

ALGORITHM

Step 1: push() method inserts an item at the top of the stack (i.e., above its current top element).

Step2: pop() method removes the object at the top of the stack and returns that object from the function. The stack size will be decremented by one.

Step 3: isEmpty() method tests if the stack is empty or not.

Step 4:isFull() method tests if the stack is full or not.

Step 5:peek() method returns the object at the top of the stack without removing it from the stack or modifying the stack in any way.

Step 5: size () method returns the total number of elements present in the stack.

PROGRAM:

```
import java.util.*;
//Stack class
class Stack {
   int top;
   int maxsize = 5;
   int[] stack_arry = new int[maxsize];
   Stack()
{          initially top = -1
      top = -1;
   }
   boolean isEmpty(){
      return (top < 0);
   }
  boolean push (int val)
{
      if(top == maxsize-1)
{
         System.out.println("Stack Overflow !!");
         return false;
      }
      else  {
         top++;
```

```
            stack_arry[top]=val;
            return true;
        }
    }
    boolean pop () {            //pop () method
        if (top == -1) {
            System.out.println("Stack Underflow !!");
            return false;
        }
        else   {

            System.out.println("\nItem popped: " + stack_arry[top--]);
            return true;
        }
    }
    void display () {           //print the stack elements
        System.out.println("Printing stack elements .....");
        for(int i = top; i>=0;i--) {
            System.out.print(stack_arry[i] + " ");
        }
    }
}

public class Main {
public static void main(String[] args) {
    //define a stack object
    Stack stck = new Stack();
    System.out.println("Initial Stack Empty : " + stck.isEmpty());
    //push elements
    stck.push(10);
    stck.push(20);
    stck.push(30);
    stck.push(40);
    System.out.println("After Push Operation...");
    //print the elements
    stck.display();
    //pop two elements from stack
    stck.pop();
    stck.pop();
    System.out.println("After Pop Operation...");
        stck.display();
}
}
```

OUTPUT:

RESULT

Thus the java program to develop Stack data structure using classes and objects was executed and verified successfully.

2.b) QUEUE IMPLEMENTATION

AIM:

To write a Java program to develop Queue data structure using classes and objects.

ALGORITHM

Step 1: The Enqueue () method inserts an item at the rear of the queue.

Step2: The Dequeue ()method removes the object from the front of the queue and returns it, thereby decrementing queue size by one.

Step 3: Peek() method returns the object at the front of the queue without removing it.

Step 4: IsEmpty() tests if the queue is empty or not.

Step 5: Size() method returns the total number of elements present in the queue.

PROGRAM:

```
class Queue
{
   private static int front, rear, capacity;
   private static int queue[];

   Queue(int size) {
      front = rear = 0;
      capacity = size;
      queue = new int[capacity];
   }

   // insert an element into the queue
   static void queueEnqueue(int item)  {
      // check if the queue is full
      if (capacity == rear) {
         System.out.printf("\nQueue is full\n");
         return;
      }

      // insert element at the rear
      else {
         queue[rear] = item;
         rear++;
      }
      return;
   }
   //remove an element from the queue
   static void queueDequeue()  {
      // check if queue is empty
      if (front == rear) {
```

```
        System.out.printf("\nQueue is empty\n");
        return;
    }

            else {
        for (int i = 0; i < rear - 1; i++) {
            queue[i] = queue[i + 1];
        }

  // set queue[rear] to 0
        if (rear < capacity)
            queue[rear] = 0;

        // decrement rear
        rear--;
    }
    return;
}

// print queue elements
static void queueDisplay()
{
    int i;
    if (front == rear) {
        System.out.printf("Queue is Empty\n");
        return;
    }

    // traverse front to rear and print elements
    for (i = front; i < rear; i++) {
        System.out.printf(" %d , ", queue[i]);
    }
    return;
}

// print front of queue
static void queueFront()
{
    if (front == rear) {
        System.out.printf("Queue is Empty\n");
        return;
    }
    System.out.printf("\nFront Element of the queue: %d", queue[front]);
    return;
```

```
    }
}
    public class QueueArrayImplementation {
    public static void main(String[] args) {
        Queue q = new Queue(4);
        System.out.println("Initial Queue:");
        q.queueDisplay();
        q.queueEnqueue(10);
        q.queueEnqueue(30);
        q.queueEnqueue(50);
        q.queueEnqueue(70);
       System.out.println("Queue after Enqueue Operation:");
        q.queueDisplay();
        q.queueFront();
        q.queueEnqueue(90);
        q.queueDisplay();
        q.queueDequeue();
        q.queueDequeue();
        System.out.printf("\nQueue after two dequeue operations:");
        q.queueDisplay();
        q.queueFront();
    }
}
```

OUTPUT:

RESULT

Thus the java program to develop Queue data structure using classes and objects was executed and verified successfully.

DATE:

3. EMPLOYEE PAYROLL

AIM:

To Develop a java application with Employee class with Emp_name, Emp_id, Address, Mail_id, Mobile_no as members. Inherit the classes, Programmer, Assistant Professor, Associate Professor and Professor from employee class. Add Basic Pay (BP) as the member of all the inherited classes with 97% of BP as DA, 10 % of BP as HRA, 12% of BP as PF, 0.1% of BP for staff club fund. Generate pay slips for the employees with their gross and net salary.

ALGORITHM:

1. Create a class called Employee
2. Get the employee details from the user.
3. By using the above information calculate the salary of an employee
4. Display the details.

PROGRAM:

```
import java.util.Scanner;
 class Employee
{
   int id;
   String name, address, mail, mno;
   Scanner get = new Scanner(System.in);
   Employee()
   {
      System.out.println("Enter Name of the Employee:");
      name = get.nextLine();
      System.out.println("Enter mailid of the Employee:");
         mail = get.nextLine();
      System.out.println("Enter Address of the Employee:");
```

```
        address = get.nextLine();
        System.out.println("Enter mobile no. of the Employee:");
        mno = get.nextLine();
        System.out.println("Enter id:");
        id = get.nextInt();
    }
    void display()
    {
        System.out.println("Employee Name: "+name);
        System.out.println("ID: "+id);
        System.out.println("Mail Id: "+mail);
        System.out.println("Address: "+address);
        System.out.println("Mobile No.: "+mno);
    }
}
 class Programmer extends Employee
{
    float grosssalary, netsalary;
    float bpay;
    Programmer()
    {
        System.out.println("Enter Basic Pay:");
        bpay = get.nextFloat();
            grosssalary=(0.97f*bpay)+(0.10f*bpay)+(0.12f*bpay)+(0.001f*bpay)+bpay;
            netsalary= (0.97f*bpay)+(0.10f*bpay)+bpay;
    }
    void display()
                                                                                                {
System.out.println("==============================="+"\n"+"Programmer"+"\n"+"===
============================"+"\n");
        super.display();
```

```
        System.out.println("Gross Salary: "+grosssalary);
        System.out.println("Net Salary: "+netsalary);
    }
}
 class AssistantProfessor extends Employee
{
    float grosssalary, netsalary;
    float bpay;
    AssistantProfessor()
    {
        System.out.println("Enter Basic Pay:");
        bpay = get.nextFloat();
            grosssalary= (0.97f*bpay)+(0.10f*bpay)+(0.12f*bpay) + (0.001f*bpay)+bpay;
            netsalary= (0.97f*bpay)+(0.10f*bpay)+bpay;
    }
    void display()
    {
                        System.out.println("==============================="+"\n"+"Assistant
Professor"+"\n"+"==============================="+"\n");
        super.display();
        System.out.println("Gross Salary: "+grosssalary);
        System.out.println("Net Salary: "+netsalary);
    }
}
class AssociateProfessor extends Employee
{
    float grosssalary, netsalary;
    float bpay;
    AssociateProfessor()
    {
        System.out.println("Enter Basic Pay:");
```

```
        bpay = get.nextFloat();
            grosssalary=(0.97f*bpay)+(0.10f*bpay)+(0.12f*bpay)+ 0.001f*bpay)+bpay;
            netsalary= (0.97f*bpay)+(0.10f*bpay)+bpay;
    }
    void display()
    {
                    System.out.println("================================="+"\n"+"Associate
Professor"+"\n"+"================================="+"\n");
        super.display();
        System.out.println("Gross Salary: "+grosssalary);
        System.out.println("Net Salary: "+netsalary);
    }
}
class Professor extends Employee
{
    float grosssalary, netsalary;
    float bpay;
    Professor()
    {
        System.out.println("Enter Basic Pay:");
        bpay = get.nextFloat();
            grosssalary=(0.97f*bpay)+(0.10f*bpay)+(0.12f*bpay)+(0.001f*bpay)+bpay;
            netsalary= (0.97f*bpay)+(0.10f*bpay)+bpay;
    }
    void display()
    {
System.out.println("================================="+"\n"+"Professor"+"\n"+"=====
============================"+"\n");
        super.display();
        System.out.println("Gross Salary: "+grosssalary);
        System.out.println("Net Salary: "+netsalary);
```

```
    }
}
class Employees
{
    public static void main(String args[])
    {
            System.out.println("==================================="+"\n"+"Enter Professor
Details"+"\n"+"==================================="+"\n");
        Professor ob1 = new Professor();
        ob1.display();
        AssociateProfessor ob2 = new AssociateProfessor();
           System.out.println("==================================="+"\n"+"Enter Associate
Professor Details"+"\n"+"==================================="+"\n");
        ob2.display();
        AssistantProfessor ob3 = new AssistantProfessor();
            System.out.println("==================================="+"\n"+"Enter Assistant
Professor Details"+"\n"+"==================================="+"\n");
        ob3.display();
        Programmer ob4 = new Programmer();
                        System.out.println("==================================="+"\n"+"Enter
Programmer Details"+"\n"+"==================================="+"\n");
        ob4.display();
    }
}
```

OUTPUT:

RESULT:

Thus the java program to implement employee payroll was executed successfully.

DATE:

4. a ABSTRACT CLASS EXAMPLE

AIM:

To create an abstract class named shape that contains two integers and an empty method named printArea .Provide three classes named Rectangle ,Triangle and Circle subclass that each one of the classes extends the Class Shape .Each one of the classes contains only the method printArea() that prints the area of Shape.

ALGORITHM:

1. Start the program
2. Create a class with the name Shape.
3. Create the Rectangle,Triangle,Circle extends Shape.
4. Create the objects to the individual classes.
5. Call the PrintArea() on individual object.

PROGRAM:

```
abstract class shape
{
public int x, y;
public abstract void printArea();
}
class Rectangle extends shape
{
public void printArea()
{
System.out.println("Area of Rectangle is " + x * y);
}
}
class Triangle extends shape
```

```
{
public void printArea()
 {
System.out.println("Area of Triangle is " + (x * y) / 2);
}
}
class Circle extends shape
{
public void printArea()
 {
System.out.println("Area of Circle is " + (22 * x * x) / 7);
}
}
public class Abstex
{
/**
* @param args the command line arguments
*/
public static void main(String[] args)
{
// TODO code application logic here
Rectangle r = new Rectangle();
r.x = 10;
r.y = 20;
r.printArea();
System.out.println("-------------------------------------");
Triangle t = new Triangle();
t.x = 30;
t.y = 35;
t.printArea();
System.out.println("-------------------------------------");
```

```
Circle c = new Circle();
c.x = 2;
c.printArea();
System.out.println("-------------------------------------");
}
```

OUTPUT

RESULT:

Thus the program for abstract class was executed successfully.

DATE:

4. b.ABSTRACT CLASS EXAMPLE

AIM:

To write a java program to implement abstract class.

ALGORITHM:

1. Start the program
2. Create a class with the name Shape.
3. Create the Rectangle,Triangle,Circle extends Shape.
4. Create the objects to the individual classes.
5. Call the numberOfSides() on individual object.

PROGRAM

```
import java.io.*;
import java.util.*;
abstract class Shape
{
 abstract void numberOfSides();
  }
class Trapezoid extends Shape
{
void numberOfSides()
 {
        System.out.println(" Trapezoidal has four sides");
}
  }
class Triangle extends Shape
{
```

```
 void numberOfSides()
{
 System.out.println("Triangle has three sides");
}
  }
 class Hexagon extends Shape
{
void numberOfSides()
{
System.out.println("Hexagon has six sides");
 }
 }
class ShapeDemo
{
 public static void main(String args[ ])
 {
Trapezoid t=new Trapezoid();
Triangle r=new Triangle();
Hexagon h=new Hexagon();
Shape s;
s=t;
s.numberOfSides();
s=r;
s.numberOfSides();
s=h;
s.numberOfSides();
}
}
```

OUTPUT:

RESULT:

Thus the program for abstract class was executed successfully.

DATE:

5.INTERFACES - DEVELOPING USER DEFINED INTERFACES

AIM:

To write a java program to develop user defined interfaces.

ALGORITHM:

1. Start the program
2. Declare the interface called Area and declare the function called compute in it.
3. Define the class called Rectangle and implement the interface Area.
4. Define the class called Circle and implement the interface Area.
5. Display the result.

PROGRAM:

```
import java.io.*;
interface Area
{
final static float pi=3.14F;
float compute(float x, float y);
}
class Rectangle implements Area
{
public float compute(float x, float y)
{
return(x*y);
}
}
class Circle implements Area
{
```

```
public float compute(float x, float y)
{
return(pi*x*x);
}
}
class inter
{
public static void main(String args[])
{
Rectangle rect=new Rectangle(); // Object to class ‘Rectangle’
Circle cir=new Circle(); // Object to class ‘Circle’
System.out.println("Area of Rectangle ="+ rect.compute(10,20));
System.out.println("Area of Circle ="+ cir.compute(10,0));
}
}
```

OUTPUT:

RESULT:

Thus java program to implement user defined interface was created, executed and output was verified successfully.

DATE:

6. HANDLING USERDEFINED EXCEPTION

AIM:

To implement the user defined exception concept in java.

ALGORITHM:

1. Create a user defined exception class called MyException
2. Throw an exception of user defined type as an argument in main()
3. Exception is handled using try, catch block
4. Display the user defined exception

PROGRAM:

```
import java.lang.Exception;
class MyException extends Exception
{
int a;
MyException(int b)
{
a=b;
}
public String toString()
{
return ("Exception Number = " +a) ;
}
}
class JavaException
{
public static void main(String args[])
```

```
{
try
{
throw new MyException(2); // throw is used to create a new exception and throw it.
}
catch(MyException e)
{
System.out.println(e) ;
}
}
}
```

OUTPUT:

RESULT:

Thus the java program to implement user defined exception was executed successfully.

DATE:

7.MULTITHREAD CONCEPT

AIM:

To implements a multithread application that has three threads .First thread generates random integer for every second and if the value is even ,second thread computes the square of number and prints .If the value is odd ,the third thread will print the value of cube of number.

ALGORITHM:

1. Start the program
2. Create a class with the name „even implements Runnable‟ and „odd implements Runnable‟.
3. Create thread objects and Random class object.
4.Pass the objects of our class to thread class .
5. Call the start method.

PROGRAM

```
import java.util.*;
class even implements Runnable {
public int x;
public even(int x) {
this.x = x;
}
public void run() {
System.out.println("Thread Name:Even Thread and " + x + "is even Number and Square of " + x
+ " is: " + x * x);
}
}
class odd implements Runnable {
public int x;
```

```
public odd(int x) {
this.x = x;
}
public void run() {
System.out.println("Thread Name:ODD Thread and " + x + " is odd number and Cube of " + x +
" is: " + x * x * x);
}
}
class A extends Thread {
public String tname;
public Random r;
public Thread t1, t2;
public A(String s) {
tname = s;
}
public void run() { int num = 0;
r = new Random();
try {
for (int i = 0; i < 50; i++) {
num = r.nextInt(100);
System.out.println("Main Thread and Generated Number is " + num);
if (num % 2 == 0) {
t1 = new Thread(new even(num));
t1.start();
} else {
t2 = new Thread(new odd(num));
t2.start();
}
Thread.sleep(1000);
System.out.println("---------------------------------------");
}
```

```
}
catch (Exception ex)
{
System.out.println(ex.getMessage());
}
}
}
public class Mthread
{
public static void main(String[] args)
{
A a = new A("One");
a.start();
}
}
```

OUTPUT:

RESULT:

Thus the multithreaded program was executed successfully.

DATE:

8.FILE CONCEPT

AIM:

To write a java program read a file name from the user, then displays information about whether the file exists, whether the file is readable, whether the file is writable, the type of file and the length of the file in bytes.

ALGORITHM:

1. Start the program
2. Create a class with the name FileDemo.
3. Get the file name from the user.
4.Check whether the file exist ot not.
5.If yes check whether the file is readable,Writable.
6.Finally disply the file details.

PROGRAM:

```
import java.io.File;
class FileDemo
{
  static void p(String s)
{
    System.out.println(s);
  }
  public static void main(String args[ ])
{
    File f1 = new File(args[0]);
    p("File Name: " + f1.getName());
    p("Path: " + f1.getPath());
```

```
    p("Abs Path: " + f1.getAbsolutePath());
    p("Parent: " + f1.getParent());
    p(f1.exists() ? "exists" : "does not exist");
    p(f1.canWrite() ? "is writeable" : "is not writeable");
    p(f1.canRead() ? "is readable" : "is not readable");
    p("is " + (f1.isDirectory() ? "" : "not" + " a directory"));
    p(f1.isFile() ? "is normal file" : "might be a named pipe");
    p(f1.isAbsolute() ? "is absolute" : "is not absolute");
      p("File last modified: " + f1.lastModified());
    p("File size: " + f1.length() + " Bytes");
  }
}
```

OUTPUT:

RESULT:

Thus the java program to display the file details was executed successfully.

DATE:

9.FIND MAXIMUM ELEMENT USING GENERIC FUNCTION

AIM:

To write a java program to find maximum element using generic function..

ALGORITHM:

1. Start the program.
2. Create the class named as MainClass.
3. Create the generic function x,y,z using function template.
4. Find the maximum of three elements x,y,z.

PROGRAM:

```
public class MainClass
{
  // determines the largest of three Comparable objects
  public static <T extends Comparable<T>> T maximum(T x, T y, T z) {
    T max = x; // assume x is initially the largest
    if (y.compareTo(max) > 0)
      max = y; // y is the largest so far
    if (z.compareTo(max) > 0)
      max = z; // z is the largest
    return max; // returns the largest object
  } // end method maximum
  public static void main(String args[])
{
    System.out.printf("Maximum of %d, %d and %d is %d\n\n", 3, 4, 5, maximum(3, 4, 5));
          System.out.printf("Maximum of %.1f, %.1f and %.1f is %.1f\n\n", 6.6, 8.8, 7.7,
maximum(6.6,
```

```
        8.8, 7.7));
        System.out.printf("Maximum of %s, %s and %s is %s\n", "pear", "apple", "orange", maximum(
        "pear", "apple", "orange"));
    }
}
```

OUTPUT:

RESULT:

Thus the java program to find maximum using generic function was executed successfully.

10. a . MINI PROJECT: TRAFFIC LIGHT SIMULATION:

AIM:

To simulate a Traffic Light. The program lets the use select one of three lights :red, yellow or Green with radio buttons. On selecting radio button, an appropriate message with “stop” or “Ready” or “GO” should appear above the button in selected color. Intially ,there is no message Shown.

ALGORITHM:

1. Start the program
2. Create a class with the name A implements ItemListener.
3. Create the ButtonGroup ,JRadiobuttons,JPanel.
4. Add to the JFrame
5. Register the components to the Jframe.
6. Close the Jframe.

PROGRAM:

```
import javax.swing.*;
import javax.swing.event.*;
import java.awt.*;
import java.awt.event.*;
class A extends JFrame implements ItemListener {
public JLabel l1, l2;
public JRadioButton r1, r2, r3;
public ButtonGroup bg;
public JPanel p, p1;
public A() {
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setLayout(new GridLayout(2, 1));
setSize(800, 400);
```

```
p = new JPanel(new FlowLayout());
p1 = new JPanel(new FlowLayout());
l1 = new JLabel();
Font f = new Font("Verdana", Font.BOLD, 60);
l1.setFont(f);
add(l1);
p.add(l1);
add(p);
l2 = new JLabel("Select Lights");
p1.add(l2);
JRadioButton r1 = new JRadioButton("Red Light");
r1.setBackground(Color.red);
p1.add(r1);
r1.addItemListener(this);
JRadioButton r2 = new JRadioButton("Yellow Light");
r2.setBackground(Color.YELLOW);
p1.add(r2);
r2.addItemListener(this);
JRadioButton r3 = new JRadioButton("Green Light");
r3.setBackground(Color.GREEN);
p1.add(r3);
r3.addItemListener(this);
add(p1);
bg = new ButtonGroup();
bg.add(r1);
bg.add(r2);
bg.add(r3);
setVisible(true);
}
public void itemStateChanged(ItemEvent i) {
JRadioButton jb = (JRadioButton) i.getSource();
```

```
switch (jb.getText()) {
case "Red Light": {
l1.setText("STOP");
l1.setForeground(Color.red);
}
break;
case "Yellow Light": {
l1.setText("Ready");
l1.setForeground(Color.YELLOW);
}
break;
case "Green Light": {
l1.setText("GO");
l1.setForeground(Color.GREEN);
}
break;
}
}
}
public class TLights {
public static void main(String[] args) {
A a = new A();
}
}
```

RESULT:

Thus the java program for Traffic Light simulation has been simulated successfully.

11.b. Develop a java application to implement currency converter (Dollar to INR, EURO to INR, Yen to INR and vice versa), distance converter (meter to KM, miles to KM and vice versa) , time converter (hours to minutes, seconds and vice versa) using packages.

Package Code: For Currency converter

```
package MyPack1;      // creating a package “MyPack1”
import java.util.Scanner;
public class currency
{
    public currency()
   {
      char us_dollar_sym = 36;
      char rupee_sym = 163;
      char yen_sym = 165;
      char euro_sym = 8364;
      String us_dollar = "Dollars";
      String rupee = "Rupees";
      String yen = "Yen";
      String euro = "Euros";
      double rate = 0;
      // Interface
      System.out.println("Welcome to the Currency Converter Program \n");
         System.out.println("Use the following codes to input your currency choices: \n 1 - US dollars \n 2 - Euros \n 3 - Indian Rupees \n 4 - Japanese Yen \n");
      System.out.println("Please choose the input currency:");
      Scanner in = new Scanner(System.in);
      int choice = in.nextInt();

     String inType = null;
      switch(choice) {
```

```
    case 1: inType = "US Dollars >> " + us_dollar_sym;  break;
    case 2: inType = "Euros >> " + euro_sym; break;
    case 3: inType = "Indian Rupees >> " + rupee_sym; break;
    case 4: inType = "Japanese Yen >> " + yen_sym; break;
    default:
    System.out.println("Please restart the program & enter a number from the list.");
    return;
   }

    System.out.println("Please choose the output currency");
    int output = in.nextInt();

    System.out.printf("Now enter the input in " + inType);
    double input = in.nextDouble();

   if (choice == output)
    System.out.println("Same currency no need to convert");

   if (choice == 1 && output == 2)
    {
      double dollar_euro_rate = 83;
      rate = input * dollar_euro_rate;
          System.out.printf( "%s" + input + " at a conversion rate of " + dollar_euro_rate + "
Dollars to %s =
                                %.2f\n", (char)us_dollar_sym, euro, rate);
    }
    else if (choice == 1 && output == 3){
      double dollar_rupee_rate = 66.82;
      rate = input * dollar_rupee_rate;
          System.out.printf( "%s" + input + " at a conversion rate of " + dollar_rupee_rate + "
Dollars to %s
```

```
                                = %.2f\n", (char)us_dollar_sym, rupee, rate);
        }
        else if (choice == 1 && output == 4){
          double dollar_yen_rate = 109.12;
          rate = input * dollar_yen_rate;
            System.out.printf( "%s" + input + " at a conversion rate of " + dollar_yen_rate + " Dollars
to %s =
                                        %.2f\n", (char)us_dollar_sym, yen, rate);
        }
       if (choice == 2 && output == 1)
       {
          double euro_dollar_rate = 1.20;
          rate = input * euro_dollar_rate;
            System.out.printf( "%s" + input + " at a conversion rate of " + euro_dollar_rate + " Euros
to %s =
                                        %.2f\n", (char)euro_sym, us_dollar, rate);
        }
        else if (choice == 2 && output == 3)
        {
          double euro_rupee_rate = 80.03;
          rate = input * euro_rupee_rate;
            System.out.printf( "%s" + input + " at a conversion rate of " + euro_rupee_rate + " Euros
to %s =
                                %.2f\n", (char)euro_sym, rupee, rate);
        }
        else if (choice == 2 && output == 4)
        {
          double euro_yen_rate = 130.69;
          rate = input * euro_yen_rate;
            System.out.printf( "%s" + input + " at a conversion rate of " + euro_yen_rate + " Euros to
%s =
```

```
                        %.2f\n", (char)euro_sym, yen, rate);
        }
      if (choice == 3 && output == 1)
      {
         double rupee_dollar_rate = 0.015;
             System.out.printf( "%s" + input + " at a conversion rate of " + rupee_dollar_rate + "
Pounds to %s
                        = %.2f\n", (char)rupee_sym, us_dollar, rate);
        }
        else if (choice == 3 && output == 2)
        {
         double rupee_euro_rate = 0.012;
          System.out.printf( "%s" + input + " at a conversion rate of " + rupee_euro_rate + " Pounds
to %s =
                                %.2f\n", (char)rupee_sym, euro, rate);
        }
        else if (choice == 3 && output == 4)
        {
         double rupee_yen_rate = 1.641;
           System.out.printf( "%s" + input + " at a conversion rate of " + rupee_yen_rate + " Pounds
to %s =
                                %.2f\n", (char)rupee_sym, yen, rate);
        }
      if (choice == 4 && output == 1)
      {
         double yen_dollar_rate = 0.0092;
           System.out.printf( "%s" + input + " at a conversion rate of " + yen_dollar_rate + " Yen to
%s =
                                %.2f\n", (char)yen_sym, us_dollar, rate);
        }
        else if (choice == 4 && output == 2)
```

```
      {
        double yen_euro_rate = 0.0077;
          System.out.printf( "%s" + input + " at a conversion rate of " + yen_euro_rate + " Yen to
%s =
                                        %.2f\n", (char)yen_sym, euro, rate);
      }
      else if (choice == 4 && output == 3)
      {
        double yen_rupee_rate = 0.61;
          System.out.printf( "%s" + input + " at a conversion rate of " + yen_rupee_rate + " Yen to
%s =
                                          %.2f\n", (char)yen_sym, rupee, rate);
      }
     System.out.println("Thank you for using the currency converter");
   }
}
```

Package Code: For Distance converter

```
package MyPack;
import java.util.Scanner;

public class DConverter {



        public DConverter(){
        // Variable declarations
        int num;//num entered by user
        double meters;

        // Scanner object for keyboard input
        Scanner keyboard = new Scanner(System.in);
```

```
// Ask user for meters
System.out.print("Enter a distance in meters, e.g 100: ");
meters = keyboard.nextDouble();

// Check for input greater than zero
while(meters <=0)
{
String error = "Please enter a num greater than zero";
System.out.println(error);
meters = keyboard.nextDouble();
}

// Menu options
System.out.print("\nEnter 1-4 from the menu options: ");
System.out.println("\n1. Convert to kilometers\n2. Convert to inches\n" +
"3. Convert to feet\n4. End");
num = keyboard.nextInt();

// num entered by the user
while(num <=0 || num>=5)
{
String invalid = "Invalid option entered. Please use 1 through 4";
System.out.println(invalid);
meters = keyboard.nextDouble();
}
switch (num)

{

case 1:
```

```
dpKilometers(meters);
break;

case 2:
dpInches(meters);
break;

case 3:
dpFeet(meters);
break;
case 4:
System.exit(0);
}
}
public static void dpKilometers(double meters)
{
double km;
km = meters * .001;
System.out.println(+ meters + " meters equals to " +km+ " kilometers");
}
public static void dpInches(double meters)
{
double in;
in = meters * 39.37;
System.out.println(meters + " meters equals to " + in + " inches.");
}
public static void dpFeet(double meters)
{
double ft;
ft = meters * 3.281;
System.out.println(meters + " meters equals to " + ft + " feet.");
```

```
        }
        }
```

Package Code: For Time Conversion

```
package MyPack;
import java.util.Scanner;
public class TConverter
{
    public TConverter()
   {

       String hr = "Hours";
       String min = "Minutes";
       String sec = "Seconds";
       double time = 0, rate=0;
       // Interface
       System.out.println("Welcome to the Time Converter Program \n");
        System.out.println("Use the following codes to input your Time choices: \n 1 - Hours \n 2 -
Minutes \n 3 - Seconds \n");
       System.out.println("Please choose the input time format:");
       Scanner in = new Scanner(System.in);
       int choice = in.nextInt();
      String inType = null;
       switch(choice) {
       case 1: inType = "Hours >> " ;  break;
       case 2: inType = "Minutes >> "; break;
       case 3: inType = "Seconds >> "; break;
       default:
       System.out.println("Please restart the program & enter a number from the list.");
       return;
     }
```

```
  System.out.println("Please choose the output time format");
  int output = in.nextInt();
  System.out.printf("Now enter the input in " + inType);
  double input = in.nextDouble();
if (choice == output)
  System.out.println("Same time format no need to convert");
if (choice == 1 && output == 2)
  {
     double hrs_min_rate = 60;
     rate = input * hrs_min_rate;
     System.out.printf(input+ " Hours converted to "+rate+" Minutes");
  }
  else if (choice == 1 && output == 3){
     double hrs_sec_rate = 3600;
     rate = input * hrs_sec_rate;
    System.out.printf(input+ " Hours converted to "+rate+" seconds");
  }
if (choice == 2 && output == 1)
{
    double min_hrs_rate = 0.01667;
    rate = input * min_hrs_rate;
     System.out.printf(input+ " Minutes converted to "+rate+" Hours");
  }
  else if (choice == 2 && output == 3)
  {
    double min_sec_rate = 60;
    rate = input * min_sec_rate;
      System.out.printf(input+ " Minutes converted to "+rate+" Seconds");
  }
if (choice == 3 && output == 1)
```

```
    {
      double sec_hrs_rate = 0.0002778;
      rate = input * sec_hrs_rate;
        System.out.printf(input+ " Seconds converted to "+rate+" Hours");
     }
     else if (choice == 3 && output == 2)
     {
      double sec_min_rate = 0.01667;
     rate = input * sec_min_rate;
        System.out.printf(input+ " Seconds converted to "+rate+" Minutes");       }
    System.out.println("Thank you for using the Time converter");
  }
}
```

Using Package:

```
import MyPack.*;
public class Ex{
        public static void main(String args[]){
                currency c=new currency();
                DConverter dc=new DConverter();
                TConverter tc=new TConverter();
        }
}
```

OUTPUT:

VIVA QUESTIONS AND ANSWERS

1. What is The Java Features ?
   - Compiled and Interpreted
   - Platform-Independent and Portable
   - Object – Oriented
   - Robust and Secure
   - Distributed
   - Simple, Small and Familiar
   - Multithreaded and interactive
   - High Performance
   - Dynamic and Extensible

2 .How Java Differs From C ?
   - Java does not include the C unique statement keywords goto, size of, and type def.
   - Java does not contain the data type struct,union and enum.
   - Java does not define the type modifiers keywords auto,extern ,register,signed,and unsigned.
   - Java does not support an explicit pointer type.

3. How Java Differs From C ++ ?
   - java does not support operator overloading.
   - Java does not have template classes as in C++.
   - Java does not multiple inheritance of classes. This is accomplished using a new feature called “interface”.

4. What is The Java Components?
   - Object and Classes
   - Data Abstraction and Encapsulation
   - Inheritance
   - Polymorphism
   - Dynamic Binding
   - Message Communication.

5. What is the Java Development kit (JDK) ?

- appletviewer(for viewing Java applets)
- javac(Java compiler)
- java(Java interpreter)
- javap(Java disassembler)
- javah(for C header files)
- javadoc(for creating HTML documents)
- jdb(Java debugger)

6. What is the Variables ?

A variable is an identifier that denotes a storage location used to store a data value.

7. What are Class?

Class is a template for multiple objects with similar features and it is a blue print for objects. It defines a type of object according to the data the object can hold and the operations the object can perform.

8. What are Primitive data types?

Primitive data types are 8 types and they are: byte, short, int, long, float, double, boolean, char.

9. What is the method overloading.?

Function with same name but different argument perform different task known as method overloading.

10. What are inner class and anonymous class?

Inner class : classes defined in other classes, including those defined in methods are called inner classes. An inner class can have any accessibility including private. Anonymous class : Anonymous class is a class defined inside a method without a name and is instantiated and declared in the same place and cannot have explicit constructors.

11.What is the if statement ?

The if statement is a powerful decision making statement and is used to control the flow of execution of statements.

12. What is the If-else statement ?

The if…..else statement is an extension of the simple if statement.

13. What is the Switch-case statement ?

The switch statement tests the value of a given variable(or expression) against a list of case values and when a match is found , a block of statements associated with that case is executed.

14. What are the Break ?

The break statement is used to terminate a statement sequence.

15.What are Continue ?

The continue statement causes control to be transferred directly to the conditional expression that controls the loop.

16.What is the While loop ?

The while loop is Java's most fundamental looping statement.It repeats a statement or block while its controlling expression is true.

17.What is the Do – while loop ?

The do –while is an exit – controlled loop.Based on a condition, the control is transferred back to a particular point in the program.

18.What is the for loop ?

The for is an entry-controlled loop and is used when an action is to be repeated for a predetermined number of itmes.

19.What is the Operators ?

Operators are used in programs to manipulate data and variables.

20.What is the Arithmetic Expressions ?

An arithmetic expression is a combination of variables , constants , and operators arranged as per the syntax of the language.

21. What is the Constructor ?

Constructor is a special member function of class call automatically when object is created.

22. What is the Parameterized constructors ?

The constructors that can take argument are called parameterized constructors.

23. What is OOPs?

Object oriented programming organizes a program around its data, i. e. , objects and a set of well defined interfaces to that data. An object-oriented program can be characterized as data controlling access to code.

24. What are Encapsulation?

Encapsulation is the mechanism that binds together code and data it manipulates and keeps both safe from outside interference and misuse.

25. What are Polymorphism?

Polymorphism is the feature that allows one interface to be used for general class actions.

26. What is the difference between procedural and object-oriented programs?

a) In procedural program, programming logic follows certain procedures and the instructions are executed one after another. In OOP program, unit of program is object, which is nothing but combination of data and code.

b) In procedural program, data is exposed to the whole program whereas in OOPs program, it is accessible with in the object and which in turn assures the security of the code.

27. What is an Object and how do you allocate memory to it?

Object is an instance of a class and it is a software unit that combines a structured set of data with a set of operations for inspecting and manipulating that data. When an object is created using new operator, memory is allocated to it.

28. What is the difference between constructor and method?

Constructor will be automatically invoked when an object is created whereas method has to be called explicitly.

29. What are methods and how are they defined?

Methods are functions that operate on instances of classes in which they are defined.

30. What is the use of bin and lib in JDK?

Bin contains all tools such as javac, appletviewer, awt tool, etc., whereas lib contains API and all packages.

31. What is casting?

Casting is used to convert the value of one type to another.

32. How many ways can an argument be passed to a subroutine and explain them? An argument can be passed in two ways. They are passing by value and passing by reference. Passing by value: This method copies the value of an argument into the formal parameter of the subroutine. Passing by reference: In this method, a reference to an argument (not the value of the argument) is passed to the parameter.

33. What is the difference between an argument and a parameter?

While defining method, variables passed in the method are called parameters. While using those methods, values passed to those variables are called arguments.

34. What are different types of access modifiers?

public: Any thing declared as public can be accessed from anywhere. private: Any thing declared as private can't be seen outside of its class.

protected: Any thing declared as protected can be accessed by classes in the same package and subclasses in the other packages.

default modifier : Can be accessed only to classes in the same package.

35. What is UNICODE?

Unicode is used for internal representation of characters and strings and it uses 16 bits to represent each other.

36. What is the difference between String and String Buffer?

a) String objects are constants and immutable whereas StringBuffer objects are not. b) String class supports constant strings whereas StringBuffer class supports growable and modifiable strings.

37. What is the difference between Array and vector?

Array is a set of related data type and static whereas vector is a growable array of objects and dynamic.

38. What are wrapper classes?

Wrapper classes are classes that allow primitive types to be accessed as objects.

39. What are Vector?

Vector : The Vector class provides the capability to implement a growable array of object.

41. What is the Inheritance?

Inheritance is the process by which object of one class acquire the properties of object of another class.

42. What is the Package?

Package is the collection of class stored in a folder. These class can be used in any java program.

43. What is meant by Inheritance and what are its advantages?

Inheritance is the process of inheriting all the features from a class. The advantages of inheritance are reusability of code and accessibility of variables and methods of the super class by subclasses.

44. What is an abstract class?

An abstract class is a class designed with implementation gaps for subclasses to fill in and is deliberately incomplete.

45. What is a clone able interface and how many methods does it contain?

It is not having any method because it is a TAGGED or MARKER interface. 46- What is the difference between abstract class and interface?

a) All the methods declared inside an interface are abstract whereas abstract class must have at least one abstract method and others may be concrete or abstract.

b) In abstract class, key word abstract must be used for the methods whereas interface we need not use that keyword for the methods.

c) Abstract class must have subclasses whereas interface can't have subclasses.

47. Can you have an inner class inside a method and what variables can you access?

Yes, we can have an inner class inside a method and final variables can be accessed.

48. What is the class and interface in java to create thread and which is the most advantageous method?

Thread class and Runnable interface can be used to create threads and using Runnable interface is the most advantageous method to create threads because we need not extend thread class here.

49. What are the states associated in the thread?

Thread contains ready, running, waiting and dead states.

50. What is deadlock?

When two threads are waiting each other and can't precede the program is said to be deadlock.

51. What is the Interfaces?

Using the keyword interface ,you can fully abstract a class' interface from its implementation.

52.What is The Try-Catch-Finally ?

- Java uses a keyword Try to handle a run-time error, simply enclose the code that you want to monitor inside a try block. Immediately following the try block.
- include a catch clause that specifies the exception type that you wish to catch.
- finally block can be used to handle any exception generated within a try block.
- When a finally block is defined , this is guaranteed to execute ,regardless of whether or not an exception is thrown.

53.What are the Throw statements ?

you have only been catching exceptions that are thrown by the Java run-time system.

54.What are the Throws ?

If a method is capable of causing an exception that it does not handle, it must specify this behavior so that callers of the method can guard themselves against that exception.

55. What is Java?

Java is an object-oriented programming language developed initially by James Gosling and colleagues at Sun Microsystems.

56. Can you have virtual functions in Java?

Yes, all functions in Java are virtual by default.

57.What is the NullPointerException ?

When an object is not initialized, the default value is null.

58.Name the containers which uses Border Layout as their default layout? Containers which uses Border Layout as their default are: window, Frame and Dialog classes.

59. What is Collection API

The Collection API is a set of classes and interfaces that support operation on collections of objects.

60.What is a transient variable in Java?

A transient variable is a variable that may not be serialized. If you don't want some field to be serialized, you can mark that field transient or static.

61. What is the exception ?

An exception is a condition that is caused by a run-time error in the program.

62. What is the difference between an argument and a parameter?

While defining method, variables passed in the method are called parameters. While using those methods, values passed to those variables are called arguments.

63. What are different types of access modifiers?

public: Any thing declared as public can be accessed from anywhere. private: Any thing declared as private can’t be seen outside of its class. protected: Any thing declared as protected can be accessed by classes in the same package and subclasses in the other packages. default modifier : Can be accessed only to classes in the same package.

64. What is Garbage Collection and how to call it explicitly?

When an object is no longer referred to by any variable, java automatically reclaims memory used by that object. This is known as garbage collection. System. gc() method may be used to call it explicitly.

65. What is finalize() method? A5-finalize () method is used just before an object is destroyed and can be called just prior to garbage collection.

66. What are Transient and Volatile Modifiers?

Transient: The transient modifier applies to variables only and it is not stored as part of its object's Persistent state. Transient variables are not serialized. Volatile: Volatile modifier applies to variables only and it tells the compiler that the variable modified by volatile can be changed unexpectedly by other parts of the program.

67. What is the difference between this() and super()?

this() can be used to invoke a constructor of the same class whereas super() can be used to invoke a super class constructor.

68. What is the difference between superclass and subclass?

A super class is a class that is inherited whereas sub class is a class that does the inheriting.

69. What modifiers may be used with top-level class?

public, abstract and final can be used for top-level class.

70. What is the difference between exception and error?

The exception class defines mild error conditions that your program encounters. Exceptions can occur when trying to open the file, which does not exist, the network connection is disrupted, operands being manipulated are out of prescribed ranges, the class file you are interested in loading is missing. The error class defines serious error conditions that you should not attempt to recover from. In most cases it is advisable to let the program terminate when such an error is encountered.

71. What is the multithreaded?

A multithreaded program contains two or more parts that can run concurrently .Each part of such a program is called a thread , and each thread define a separate path of execution.

72. What is the thread ?

A thread is similar to a program that has a single flow of control.It has a beginning,a body,and an end ,and executes commands sequentially.

73. What is the Inter thread communication ?

Java includes an elegant interprocess communication mechanism via the wait( ), notify( ), and notifyAll( ) methods.

74. What is the Threads synchronization ?

When two or more threads need access to a shared resource, they need some way to ensure that the resource will be used by only one thread at a time. The process by which this is achieved is called synchronization

75. What is the synchronized Methods ?

There are Three synchronized Methods : The call( ) method, run( ) method,and sleep() method.

76. What is the difference between process and thread?

Process is a program in execution whereas thread is a separate path of execution in a program.

77. What are synchronized methods and synchronized statements? Synchronized methods are methods that are used to control access to a method or an object. A thread only executes a synchronized method after it has acquired the lock for the method's object or class. Synchronized statements are similar to synchronized methods. A synchronized statement can only be executed after a thread has acquired the lock for the object or class referenced in the synchronized statement.

78. What are three ways in which a thread can enter the waiting state?

A thread can enter the waiting state by invoking its sleep() method, by blocking on IO, by unsuccessfully attempting to acquire an object's lock, or by invoking an object's wait() method. It can also enter the waiting state by invoking its (deprecated) suspend() method.

79. Can a lock be acquired on a class?

Yes, a lock can be acquired on a class. This lock is acquired on the class's Class object.

80. What is the preferred size of a component?

The preferred size of a component is the minimum component size that will allow the component to display normally.

81. What is the difference between an applet and a servlet?

a) Servlets are to servers what applets are to browsers.

b) Applets must have graphical user interfaces whereas servlets have no graphical user interfaces.

82. What is Inet address?

Every computer connected to a network has an IP address. An IP address is a number that uniquely identifies each computer on the Net. An IP address is a 32-bit number.

83. What is an event and what are the models available for event handling?

An event is an event object that describes a state of change in a source. In other words, event occurs when an action is generated, like pressing button, clicking mouse, selecting a list, etc. There are two types of models for handling events and they are: a) event-inheritance model and b) event-delegation model