VISVESVARAYATECHNOLOGICALUNIVERSITY

JNANASANGAMA, BELGAVI-590018,KARNATAKA

Semester-III

OBJECT ORIENTED PROGRAMMING WITH JAVA

LABMANUAL (BCS306A)

(As per CBCS Scheme 2022)

Academic Year: 2023-2024

Prepared By

Mr. Praveen Kumar K C
Asst.professor,
Dept. of ISE

ChannabasaveshwaraInstituteofTechnology
(AffiliatedtoVTU,Belgaum&ApprovedbyAICTE,NewDelhi)
(NAAC Accredited & ISO 9001:2015 Certified Institution)
NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka

1. DevelopaJAVAprogramtoaddTWOmatricesofsuitableorderN(ThevalueofNshould be read from command line arguments).

```
import java.util.Scanner;
publicclassAddMatrices{
   publicstaticvoidmain(String[]args){ if
      (args.length != 1) {
         System.out.println("Pleaseprovidetheorderofthematrixasacommand-lineargument."); return;
      }
      intN=0; try
      {
         N= Integer.parseInt(args[0]);
      }catch(NumberFormatExceptione){
         System.out.println("Pleaseprovideavalidintegerfortheorderofthematrix."); return;
      }
      if(N<=0) {
         System.out.println("Pleaseprovideapositivevaluefortheorderofthematrix."); return;
      }
      int[][]matrixA=newint[N][N];
      int[][] matrixB = new int[N][N];
      int[][]sumMatrix=newint[N][N];
      Scanner scanner = new Scanner(System.in);
      System.out.println("Enter the elements of the first matrix:");
      enterMatrixElements(matrixA, scanner);
      System.out.println("Entertheelementsofthesecondmatrix:");
      enterMatrixElements(matrixB, scanner);addMatrices(matrixA,
      matrixB, sumMatrix, N);
```

```
        System.out.println("Thesumofthematricesis:");
        displayMatrix(sumMatrix);
    }
    publicstaticvoidenterMatrixElements(int[][]matrix,Scannerscanner){ for
        (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[0].length; j++) {
                System.out.print("Enterelement["+i+"]["+j+"]:");
                matrix[i][j] = scanner.nextInt();
            }
        }
    }
    publicstaticvoidaddMatrices(int[][]matrixA,int[][]matrixB,int[][]sumMatrix,intN){ for
        (int i = 0; i < N; i++) {
            for(intj =0;j<N;j++){
                sumMatrix[i][j]=matrixA[i][j]+matrixB[i][j];
            }
        }
    }
    publicstaticvoiddisplayMatrix(int[][]matrix){ for
        (int[] row : matrix) {
            for (int element : row) {
                System.out.print(element+"");
            }
            System.out.println();
        }
    }
}
```

2. Developastackclasstoholdamaximumof10integerswithsuitablemethods.DevelopaJAVA main method to illustrate Stack operations.

```
import java.util.Scanner;
publicclassAddMatrices{
   publicstaticvoidmain(String[]args){ if
      (args.length != 1) {
         System.out.println("Pleaseprovidetheorderofthematrixasacommand-lineargument."); return;
      }
      intN=0; try
      {
         N= Integer.parseInt(args[0]);
      }catch(NumberFormatExceptione){
         System.out.println("Pleaseprovideavalidintegerfortheorderofthematrix."); return;
      }
      if(N<=0) {
         System.out.println("Pleaseprovideapositivevaluefortheorderofthematrix."); return;
      }
      int[][]matrixA=newint[N][N];
      int[][] matrixB = new int[N][N];
      int[][]sumMatrix=newint[N][N];
      Scanner scanner = new Scanner(System.in);
      System.out.println("Enter the elements of the first matrix:");
      enterMatrixElements(matrixA, scanner);
      System.out.println("Entertheelementsofthesecondmatrix:");
      enterMatrixElements(matrixB, scanner);
```

```
        addMatrices(matrixA, matrixB, sumMatrix, N);
        System.out.println("Thesumofthematricesis:");
        displayMatrix(sumMatrix);
    }
    publicstaticvoidenterMatrixElements(int[][]matrix,Scannerscanner){ for
        (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[0].length; j++) {
                System.out.print("Enterelement["+i+"]["+j+"]:");
                matrix[i][j] = scanner.nextInt();
            }
        }
    }
    publicstaticvoidaddMatrices(int[][]matrixA,int[][]matrixB,int[][]sumMatrix,intN){ for
        (int i = 0; i < N; i++) {
            for(intj =0;j<N;j++){
                sumMatrix[i][j]=matrixA[i][j]+matrixB[i][j];
            }
        }
    }
    publicstaticvoiddisplayMatrix(int[][]matrix){ for
        (int[] row : matrix) {
            for (int element : row) {
                System.out.print(element+"");
            }
            System.out.println();
        }
    }
}
```

3. A class called Employee, which models an employee with an ID, name and salary, is designed as showninthe following class diagram. The methodraiseSalary (percent) increases the salary bythe givenpercentage. Develop the Employee class and suitable main method for demonstration.

```
publicclassEmployee{
   private int id;
   private  String  name;
   privatedoublesalary;
   publicEmployee(intid,Stringname,doublesalary){
      this.id = id;
      this.name  =  name;
      this.salary=salary;
   }
   publicvoidraiseSalary(doublepercent){ if
      (percent > 0) {
         doubleraise=salary*(percent/100); salary
         += raise;
         System.out.println(name+"'ssalaryraisedby"+percent+"%.Newsalary:"+salary);
      }else{
         System.out.println("Pleaseprovideapositivepercentageforsalaryraise.");
      }
   }
   public void displayInfo() {
      System.out.println("EmployeeID:"+id);
      System.out.println("Name: " + name);
      System.out.println("Salary: " + salary);
   }
   publicstaticvoidmain(String[]args){
      //Creatinganemployeeobject
      Employeeemp=newEmployee(1001,"JohnDoe",50000);
```

```
        // Displaying initial information
        System.out.println("InitialInformation:");
        emp.displayInfo();
        //Raisingsalarybyagiven percentage
        doubleraisePercentage=10;//Example:10%raise
        emp.raiseSalary(raisePercentage);
        // Displaying updated information after the raise
        System.out.println("\nInformationaftersalaryraise:"); emp.displayInfo();
    }
}
```

4. AclasscalledMyPoint,whichmodelsa2Dpointwithxandycoordinates,isdesignedasfollows:

● Twoinstancevariablesx(int)andy(int).

● Adefault(or"no-arg")constructorthatconstructapointatthedefaultlocationof(0,0).

● Aoverloadedconstructorthatconstructsapointwiththegivenxandycoordinates.

● AmethodsetXY()tosetbothxandy.

● AmethodgetXY()whichreturnsthexandyina2-elementint array

.●AtoString()methodthatreturnsastringdescriptionoftheinstanceintheformat"(x,y)".

● A method called distance(int x, int y) that returns the distance from this point to another point at the given (x, y) coordinates

● An overloaded distance(MyPoint another) that returns the distance from this point to the given MyPoint instance (called another)

● Another overloaded distance() method that returns the distance from this point to the origin (0,0) Develop the code for the class MyPoint. Also develop a JAVA program (called TestMyPoint) to test all the methods defined in the class.

```
class MyPoint {
   private int x;
   private int y;
   publicMyPoint(){
      this(0,0);//Defaultconstructorsettingcoordinatesto(0,0)
   }
   publicMyPoint(intx,inty){
      this.x = x;
      this.y=y;
   }
   publicvoidsetXY(intx,inty){ this.x =
      x;
      this.y=y;
   }
   public int[] getXY() {
      returnnewint[]{x,y};
   }
   publicStringtoString(){
      return"("+x+","+y+")";
   }
   publicdoubledistance(intx,inty){ int
      xDiff = this.x - x;
      intyDiff=this.y-y;
      returnMath.sqrt(xDiff*xDiff+yDiff*yDiff);
   }
```

```
    publicdoubledistance(MyPointanother){
      return distance(another.x, another.y);
   }
   publicdoubledistance(){
      return distance(0, 0);
   }
}

publicclassTestMyPoint{
   public static void main(String[] args) {
      MyPoint point1 = new MyPoint();
      MyPointpoint2=newMyPoint(3,4);
      //TestingsetXY()method
      point1.setXY(5, 6);
      //TestinggetXY()method
      int[]coordinates=point1.getXY();
      System.out.println("Point1coordinates:("+coordinates[0]+","+coordinates[1]+")");
      // Testing toString() method
      System.out.println("Point2:"+point2);
      //Testingdistance()methods
      System.out.println("DistancebetweenPoint1andPoint2:"+point1.distance(point2));
      System.out.println("Distance from Point 1 to origin: " + point1.distance());
   }
}
```

5. DevelopaJAVAprogramtocreateaclassnamedshape.Createthreesubclassesnamely:circle, triangleand square, each class has two member functions named draw () and erase (). Demonstrate polymorphismconceptsbydevelopingsuitablemethods,definingmemberdataandmainprogram.

```
//Shapesuperclass class
Shape {
    public void draw() {
        System.out.println("Drawingashape");
    }
    public void erase() {
        System.out.println("Erasingashape");
    }
}
//Circlesubclass
classCircleextendsShape{
    @Override
    public void draw() {
        System.out.println("Drawingacircle");
    }
    @Override
    public void erase() {
        System.out.println("Erasingacircle");
    }
}
//Trianglesubclass
classTriangleextendsShape{
    @Override
    public void draw() {
        System.out.println("Drawingatriangle");
    }
    @Override
    public void erase() {
        System.out.println("Erasingatriangle");
    }
}
//Squaresubclass
classSquareextendsShape{
    @Override
    publicvoiddraw(){
```

```
        System.out.println("Drawingasquare");
    }
    @Override
    public void erase() {
        System.out.println("Erasingasquare");
    }
}

publicclassMain{
    publicstaticvoidmain(String[]args){
        //Creatinginstancesofdifferentshapes
        Shape circle = new Circle();
        Shapetriangle=newTriangle();
        Shape square = new Square();
        //Demonstratingpolymorphismbycallingdrawanderasemethods
        circle.draw();
        circle.erase();
        triangle.draw();
        triangle.erase();
        square.draw();
        square.erase();
    }
}
```

6. Develop a JAVA program to create an abstract class Shape with abstract methods calculate Area()and calculate Perimeter(). Create subclasses Circle and Triangle that extend the Shape class and implement the respective methods to calculate the area and perimeter of each shape.

```
abstractclassShape{
    //Abstractmethodstobeimplementedbysubclasses
    public abstract double calculateArea();
    publicabstractdoublecalculatePerimeter();
}
classCircleextendsShape{
    private double radius;
    // Constructor for Circle
    publicCircle(doubleradius){
        this.radius=radius;
    }
    @Override
    public double calculateArea() {
        returnMath.PI*radius*radius;
    }
    @Override
    publicdoublecalculatePerimeter(){
        return 2 * Math.PI * radius;
    }
}
classTriangleextendsShape{
    private double side1;private
    double side2;private double
    side3;
    //ConstructorforTriangle
    publicTriangle(doubleside1,doubleside2,doubleside3){ this.side1
        = side1;
```

```
        this.side2=side2;
        this.side3=side3;
    }
    @Override
    publicdoublecalculateArea(){
        //UsingHeron'sformulatocalculateareaofatriangle double s =
        (side1 + side2 + side3) / 2;
        returnMath.sqrt(s*(s-side1)*(s -side2)*(s- side3));
    }
    @Override
    publicdoublecalculatePerimeter(){
        return side1 + side2 + side3;
    }
}
publicclassMain{
    publicstaticvoidmain(String[]args){
        //CreatinginstancesofCircleandTriangle Circle
        circle = new Circle(5);
        Triangletriangle=newTriangle(3,4, 5);
        // Calculating and displaying area and perimeter for Circle
        System.out.println("Circle - Area: " + circle.calculateArea());
        System.out.println("Circle-Perimeter:"+circle.calculatePerimeter());
        // Calculating and displaying area and perimeter for Triangle
        System.out.println("Triangle - Area: " + triangle.calculateArea());
        System.out.println("Triangle-Perimeter:"+triangle.calculatePerimeter());
    }
}
```

7. Develop a JAVA program to create an interface Resizable with methods resizeWidth(int width) and resizeHeight(int height) that allow an object to be resized. Create a class Rectangle that implements the Resizable interface and implements the resize methods

```
//Resizableinterface
interfaceResizable{
   void  resizeWidth(int  width);
   voidresizeHeight(intheight);
}
//RectangleclassimplementingResizableinterface class
Rectangle implements Resizable {
   private int width;
   privateintheight;
   //ConstructorforRectangle
   publicRectangle(intwidth,intheight){ this.width
      = width;
      this.height=height;
   }
   //ImplementingresizeWidthmethodfromResizableinterface
   @Override
   publicvoidresizeWidth(intwidth){
      this.width = width;
   }
   //ImplementingresizeHeightmethodfromResizableinterface
   @Override
   publicvoidresizeHeight(intheight){
      this.height = height;
   }
```

```
    //Methodtodisplaycurrentwidthandheightoftherectangle public
    void displaySize() {
        System.out.println("Rectangle Width: " + width);
        System.out.println("RectangleHeight:"+height);
    }
}
publicclassMain{
    publicstaticvoidmain(String[]args){
        // Creating an instance of Rectangle
        Rectanglerectangle=newRectangle(10,20);
        //Displayinginitialsizeoftherectangle
        System.out.println("Initial Size:");
        rectangle.displaySize();
        //Resizingwidthandheightoftherectangle
        rectangle.resizeWidth(15);
        rectangle.resizeHeight(25);
        //Displayingresizedsizeoftherectangle
        System.out.println("\nResized Size:");
        rectangle.displaySize();
    }
}
```

8. Develop a JAVA program to create an outer class with a function display. Create another class inside the outer class named inner with a function called display and call the two functions in the main class.

```
classOuter {
   voiddisplay(){
      System.out.println("Thisisthedisplay()methodoftheouterclass.");
   }
   classInner{
      voiddisplay(){
         System.out.println("Thisisthedisplay()methodoftheinnerclass.");
      }
   }
}
publicclassMain{
   publicstaticvoidmain(String[]args){
      Outer outer = new Outer();
      //Callingthedisplay()methodoftheouterclass outer.display();
      //Creatinganinstanceoftheinnerclassandcallingitsdisplay()method Outer.Inner
      inner = outer.new Inner();
      inner.display();
   }
}
```

9. DevelopaJAVAprogramtoraiseacustomexception(userdefinedexception)for DivisionByZero usingtry, catch, throw and finally.

```
//Customexceptionclassfor DivisionByZero
class DivisionByZeroException extends Exception {
   publicDivisionByZeroException(Stringmessage){
      super(message);
   }
}
publicclassMain{
   publicstaticvoidmain(String[]args){ try
      {
         int numerator = 10;
         intdenominator=0;

         //Performdivisionandthrowexceptionifdenominatoriszero if
         (denominator == 0) {
            thrownewDivisionByZeroException("Divisionbyzeroerror!");
         }
         int result = numerator / denominator;
         System.out.println("Resultofdivision:"+result);
      }catch(DivisionByZeroExceptione){
         System.out.println("CaughtDivisionByZeroException:"+e.getMessage());
      }catch(ArithmeticExceptione){
         System.out.println("CaughtArithmeticException:"+e.getMessage());
      }finally{
         System.out.println("Finallyblockexecuted.");
      }
   }
}
```

10. Develop a JAVA program to create a package named mypack and import & implement it in a suitable

class.

1.1. Creatingthe package:

Create adirectorynamed **mypack** andwithin that directory, createaJavafilenamed **MyPackageClass.java** withthefollowingcontent:

```
packagemypack;

publicclassMyPackageClass{
    public void display() {
        System.out.println("ThisisamethodfromtheMyPackageClassinthe 'mypack'package.");
    }
}
```

2.2. UsingthepackageinanotherJava class:

CreateaJavaclass(let'sname it **MainClass.java**)in aseparate directory(not inside **mypack**) and importandusethe **MyPackageClass** fromthe **mypack** package.

```
importmypack.MyPackageClass;

public class MainClass {
    publicstaticvoidmain(String[]args){
        MyPackageClassmyPackageObj=newMyPackageClass();
        myPackageObj.display();
    }
}
```

11. Write a program to illustrate creation of threads using runnable class. (start method start each of the newly created thread. Inside the run method there is sleep() for suspend the thread for 500 milliseconds).

```
classMyRunnableimplementsRunnable{
   private String threadName;
   publicMyRunnable(StringthreadName){
   this.threadName = threadName;
   }
   @Override
   publicvoidrun(){
      System.out.println("Thread"+threadName+"isrunning."); try {
         Thread.sleep(500);      //Suspendthethreadfor500 milliseconds
      }catch(InterruptedExceptione){
         System.out.println("Thread"+threadName+"interrupted.");
      }
      System.out.println("Thread"+threadName+"isfinished.");
   }}
publicclassMain{
   publicstaticvoidmain(String[]args){
      System.out.println("CreatingthreadsusingRunnableinterface...");
      //CreatingthreadsusingRunnableinterface
      Threadthread1=newThread(newMyRunnable("Thread1"));
      Threadthread2=newThread(newMyRunnable("Thread2"));
      Threadthread3=newThread(newMyRunnable("Thread3"));
      thread1.start();        // Starting threads using start() method
      thread2.start();
      thread3.start();
}}
```

12. Develop a program to create a class MyThread in this class a constructor, call the base class constructor,usingsuperandstartthethread.Therunmethodoftheclassstartsafterthis.Itcanbe observed that both main thread and created child thread are executed concurrently.

```
classMyThreadextendsThread{
   publicMyThread(StringthreadName){
      super(threadName);//Callingbaseclass(Thread)constructor
      start(); // Starting the thread immediately after initialization
   }
   publicvoidrun(){
      System.out.println("Insiderunmethodofthread:"+Thread.currentThread().getName()); try {
         Thread.sleep(1000);//Simulatingsometaskforthethread
      }catch(InterruptedExceptione){
         System.out.println("Thread"+Thread.currentThread().getName()+"interrupted.");
      }
      System.out.println("Thread"+Thread.currentThread().getName()+"isfinished.");
   }
}
publicclassMain{
   public static void main(String[] args) {
      System.out.println("Mainthreadisrunning.");
      //CreatinganinstanceofMyThreadandobservingconcurrentexecution MyThread
      myThread = new MyThread("Child Thread");
      //Continuingexecutioninthemainthread for
      (int i = 0; i < 5; i++) {
         System.out.println("Insidemainthread:"+i); try
         {
            Thread.sleep(500);
         }catch(InterruptedExceptione){
```

```
            System.out.println("Mainthreadinterrupted.");
        }
    }
    System.out.println("Mainthreadisfinished.");
  }
}
```