Java Lab Programs

1. Develop a java program to add two matrices of order of n.
(The value of n should be read from command line arguments)

```java
package matrixaddition;

import java.util.Scanner;

public class MatrixAddition{
        public static void main(String[] args){
                Scanner scanner=new Scanner(System.in);

                // Input the order of the matrix
                System.out.print("Enter the number of rows (N) : ");
                int N=scanner.nextInt();

                // Initialize the two matrices
                int[][] matrix1=new int[N][N];
                int[][] matrix2=new int[N][N];

                // Input the elements of the matrices
                System.out.println("Enter the elements of the first matrix :");
                for(int i=0;i<N;i++)
                        for(int j=0;j<N;j++)
                                matrix1[i][j]=scanner.nextInt();

                System.out.println("Enter the elements of the second matrix :");
                for(int i=0;i<N;i++)
                        for(int j=0;j<N;j++)
                                matrix2[i][j]=scanner.nextInt();

                // Add the matrices then store the results in new matrix and display them
                int[][] resultMatrix=new int[N][N];
                System.out.println("Resultant matrix after addition :");
                for(int i=0;i<N;i++){
                        for(int j=0;j<N;j++){
                                resultMatrix[i][j]=matrix1[i][j]+matrix2[i][j];
                                System.out.print(resultMatrix[i][j]+" ");
                        }
                        System.out.println("");
                }
                scanner.close();
        }
}
```

Output

```
Enter the number of rows (N) : 1
Enter the elements of the first matrix :
1
Enter the elements of the second matrix :
1
Resultant matrix after addition :
2
```

2. Develop a stack class to hold a maximum of IO integers with suitable methods.
Develop a java class method to illustrate Stack Operation.

```
package stackop;

public class StackOp{
        private int[] stackArray;
        private int top;
        private static final int MAX_SIZE=10;

        public StackOp(){
                stackArray=new int[MAX_SIZE];
                top=-1;
        }

        public void push(int data){
                if(isFull())
                        System.out.println(data+" : Stack is full, cannot Push");
                else{
                        stackArray[++top]=data;
                        System.out.println(data+" : Pushed on to the Stack");
                }
        }

        public int pop(){
                if(isEmpty()){
                        System.out.println("Stack is empty, cannot Pop !");
                        return -1;
                }else{
                        int poppedItem=stackArray[top--];
                        System.out.println(poppedItem+" : Popped from the Stack");
                        return poppedItem;
                }
        }

        public boolean isEmpty(){
                return top==-1;
        }

        public boolean isFull(){
                return top==MAX_SIZE-1;
        }

        public static void main(String[] args){
                StackOp stack=new StackOp();
                for(int i=0;i<=10;i++)
                        stack.push(5*i);

                System.out.println("Is the Stack empty ? "+stack.isEmpty());
                System.out.println("Is the Stack full ? "+stack.isFull());

                System.out.println("\nPopping elements from the Stack :");
                while(!stack.isEmpty())
                        stack.pop();

                System.out.println("Is the Stack empty ? "+stack.isEmpty());
                System.out.println("Is the Stack full ? "+stack.isFull());

                System.out.println("\nPopping elements from the Stack :");
                while(!stack.isEmpty())
                        stack.pop();

                System.out.println("Is the Stack empty ? "+stack.isEmpty());
                System.out.println("Is the Stack full ? "+stack.isFull());
        }
}
```

Output

```
0 : Pushed on to the Stack
5 : Pushed on to the Stack
10 : Pushed on to the Stack
15 : Pushed on to the Stack
20 : Pushed on to the Stack
25 : Pushed on to the Stack
30 : Pushed on to the Stack
35 : Pushed on to the Stack
40 : Pushed on to the Stack
45 : Pushed on to the Stack
50 : Stack is full, cannot Push
Is the Stack empty ? false
Is the Stack full ? true

Popping elements from the Stack :
45 : Popped from the Stack
40 : Popped from the Stack
35 : Popped from the Stack
30 : Popped from the Stack
25 : Popped from the Stack
20 : Popped from the Stack
15 : Popped from the Stack
10 : Popped from the Stack
5 : Popped from the Stack
0 : Popped from the Stack
Is the Stack empty ? true
Is the Stack full ? false

Popping elements from the Stack :
Is the Stack empty ? true
Is the Stack full ? false
```

3. Make a class called Employee, which models an employee with an id, name and salary designed as shown in the following class diagram. The method `raiseSalary(percent)` increase the salary given by the percentage. Develop the Employee class and suitable method of demonstration.

```
package raisesalary;

import java.util.Scanner;

class Employee{
        int empID;
        String empName;
        float empSalary;

        public void getDetailes(){
                Scanner sc=new Scanner(System.in);
                System.out.print("\nEnter the ID : ");
                empID=sc.nextInt();
                System.out.print("Enter the Name : ");
                empName=sc.next();
                System.out.print("Enter the Salary : ");
                empSalary=sc.nextInt();
        }

        public void raiseSalary(double percentage){
                double increaseAmount=(percentage/100)*empSalary;
                empSalary+=increaseAmount;
                System.out.println(empName+"'s Salary increased by "+percentage+"%");
                System.out.println("New Salary "+empSalary);
        }
}

public class RaiseSalary{
        public static void main(String[] args){
                Employee e1=new Employee();
                Employee e2=new Employee();
                Employee e3=new Employee();

                e1.getDetailes();
                e1.raiseSalary(100);
                e2.getDetailes();
                e2.raiseSalary(100);
                e3.getDetailes();
                e3.raiseSalary(200);
        }
}
```

Output

```
Enter the ID : 1
Enter the Name : a
Enter the Salary : 100
a's Salary increased by 100.0%
New Salary 200.0

Enter the ID : 2
Enter the Name : b
Enter the Salary : 200
b's Salary increased by 100.0%
New Salary 400.0

Enter the ID : 3
Enter the Name : c
Enter the Salary : 300
c's Salary increased by 200.0%
New Salary 900.0
```

4. A class called `MyPoint`,which models a 2D point with x and y coordinates,is designed as follows :

- Two instance variables `x(int)` and `y(int)`
- A default(or "no-arg") constructor that construct a point at the default location of (0,0)
- A overloaded constructor that constructs a point with the given x and y coordinates
- A method `setXY()` to set both x and y
- A method `getXY()` which returns the x and y in a 2-element int array
- A `toString()` method that returns a string description of the instance in the format "(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;

        public MyPoint(){
                x=0;
                y=0;
        }
        public MyPoint(int x,int y){
                this.x=x;
                this.y=y;
        }
        public void setXY(int x,int y){
                this.x=x;
                this.y=y;
        }

        public int[] getXY(){
                int[] coordinates={x,y};
                return coordinates;
        }

        public double distance(int x,int y){
                int dx=this.x-x;
                int dy=this.y-y;
                return Math.sqrt(dx*dx+dy*dy);
        }

        public double distance(MyPoint another){
                int dx=this.x-another.x;
                int dy=this.y-another.y;
                return Math.sqrt(dx*dx+dy*dy);
        }

        public double distance(){
                return Math.sqrt(x*x+y*y);
        }

        @Override
        public String toString(){
                return "("+x+","+y+")";
        }
}

public class TestMyPoint{
        public static void main(String[] args){
                MyPoint point1=new MyPoint();
                MyPoint point2=new MyPoint(3,4);

                point1.setXY(1,2);
                int[] coordinates=point1.getXY();
                System.out.println("Coordinates of point1 : ("+coordinates[0]+","+coordinates[1]+")");

                double distance1=point1.distance(5,6);
                System.out.println("Distance from point1 to (5,6) : "+distance1);

                double distance2=point1.distance(point2);
                System.out.println("Distance from point1 to point2 : "+distance2);

                double distance3=point2.distance();
                System.out.println("Distance from point2 to the origin : "+distance3);

                System.out.println("Point1 : "+point1.toString());
                System.out.println("Point2 : "+point2.toString());
        }
}
```

Output

```
Coordinates of point1 : (1,2)
Distance from point1 to (5,6) : 5.656854249492381
Distance from point1 to point2 : 2.8284271247461903
Distance from point2 to the origin : 5.0
Point1 : (1,2)
Point2 : (3,4)
```

5. Develop a java program to create a class named Shape. Create 3 sub classes each class has two member functions named `draw()` & `erase()`. Demonstrate the polymorphism by developing suitable methods defining member data and main program.

Shape

```
public class Shape{
        void draw(){
                System.out.println("Draw a Shape\n");
        }

        void erase(){
                System.out.println("Erase a Shape\n");
        }

        public static void main(String[] args){
                Shape s1=new Circle();
                s1.draw();
                s1.erase();

                Shape s2=new Triangle();
                s2.draw();
                s2.erase();

                Shape s3=new Square();
                s3.draw();
                s3.erase();
        }
}
```

Circle

```
public class Circle extends Shape{
        void draw(){
                System.out.println("Draw a Circle\n");
        }

        void erase(){
                System.out.println("Erase a Circle\n");
        }
}
```

Triangle

```
public class Triangle extends Shape{
        void draw(){
                System.out.println("Draw a Triangle\n");
        }

        void erase(){
                System.out.println("Erase a Triangle\n");
        }
}
```

Square

```
public class Square extends Shape{
        void draw(){
                System.out.println("Draw a Square\n");
        }

        void erase(){
                System.out.println("Erase a Square\n");
        }
}
```

Output

```
Draw a Circle

Erase a Circle

Draw a Triangle

Erase a Triangle

Draw a Square

Erase a Square
```

6. Develop a java program to create an Abstract class Shape with abstract method `calculateArea()`.
Create subclass Circle & Triangle that extend the shape class and implement the respective methods to calculate the area and perimeter of each sape.

Shape

```
abstract class Shapes{
        abstract public double calculateArea();
        abstract public double calculatePerimeter();
}

class Circle_S extends Shapes{
        private double radius;

        public Circle_S(double radius){
                this.radius=radius;
        }

        @Override
        public double calculateArea(){
                return Math.PI*radius*radius;
        }

        @Override
        public double calculatePerimeter(){
                return 2*Math.PI*radius;
        }
}

class Triangle_S extends Shapes{
        private double side1,side2,side3;

        public Triangle_S(double side1,double side2,double side3){
                this.side1=side1;
                this.side2=side2;
                this.side3=side3;
        }

        @Override
        public double calculateArea(){
                double S=(side1+side2+side3)/2;
                return Math.sqrt(S*(S-side1)*(S-side2)*(S-side3));
        }

        @Override
        public double calculatePerimeter(){
                return side1+side2+side3;
        }
}
```

Abstract

```
public class Abstract{
        public static void main(String[] args){
                Circle_S circle=new Circle_S(5.0);
                Triangle_S triangle=new Triangle_S(3.0,4.0,5.0);

                System.out.println("The area of the Circle is : "+circle.calculateArea());
                System.out.println("The perimeter of the Circle is : "+circle.calculatePerimeter());

                System.out.println("The area of the Triangle is : "+triangle.calculateArea());
                System.out.println("The perimeter of Triangle the is : "+triangle.calculatePerimeter());
        }
}
```

Output

```
The area of the Circle is : 78.53981633974483
The perimeter of the Circle is : 31.41592653589793
The area of the Triangle is : 6.0
The perimeter of Triangle the is : 12.0
```

7. Develop a java program to create an outer class with a function `display`.
Create another class named inner with function called `display` & call the functions in the `main` class.

```
class Outer{
        void display(){
                System.out.println("You are inside Outer class");
        }

        class inner{
                void display(){
                        System.out.println("You are inside Inner class");
                }
        }
}

public class OuterClass{
        public static void main(String[] args){
                Outer out=new Outer();
                out.display();

                Outer.inner in=out.new inner();
                in.display();
        }
}
```

Output

```
You are inside Outer class
You are inside Inner class
```

8. Develop a java program to create an interface resizable with method `resizableWidth` and `resizableHeight` that allows an object to be resized, create a class Rectangle that implements the resizable interface & implement resiable method

Intersize

```
public interface Intersize{
        void resizeWidth(int width);
        void resizeHeight(int height);
}

class Rectangle implements Intersize{
        private int width,height;

        public Rectangle(int width,int height){
                this.width=width;
                this.height=height;
        }

        public void resizeWidth(int width){
                this.width=width;
        }

        public void resizeHeight(int height){
                this.height=height;
        }

        public void display(){
                System.out.println("The width is : "+width+"\nThe height is : "+height);
        }
}
```

Interface

```
public class Interface{
        public static void main(String[] args){
                Rectangle r=new Rectangle(20,30);

                System.out.println("The Original Rectangle");
                r.display();
                r.resizeWidth(40);
                r.resizeHeight(50);

                System.out.println("\nThe Resized Rectangle");
                r.display();
        }
}
```

Output

```
The Original Rectangle
The width is : 20
The height is : 30

The Resized Rectangle
The width is : 40
The height is : 50
```

9. Develop a JAVA program to raise a custom exception (user defined exception) for DivisionByZero using try, catch, throw and finally

```
public class CustomException{
        public static void main(String[] args){
                int a=25,b=0;
                try{
                        if(b==0)
                                throw new ArithmeticException("Divizion by 0 error");
                        int result=a/b;
                        System.out.println(a + "/" + b + "=" + result);
                }catch(ArithmeticException e){
                        System.out.println("Raised exception is " + e);
                }finally{
                        System.out.println("Execution Finished");
                }
        }
}
```

Output

```
Raised exception is java.lang.ArithmeticException: Divizion by 0 error
Execution Finished
```

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

```java
package MyPack;
class MyClass{
        public void displayMessage(){
                System.out.println("This is a message from the MyClass in mypack package.");
        }
}

public class MainApp{
        public static void main(String[] args){
                MyClass myObject=new MyClass();
                myObject.displayMessage();
        }
}
```

Output

```
This is a message from the MyClass in mypack package.
```

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).

```
class MyRunnable implements Runnable{
        public void run(){
                try{
                        Thread.sleep(500);
                        System.out.println("Thread ID : "+Thread.currentThread().getId()+" is running !");
                }catch(InterruptedException e){
                        System.out.println("Thread Interrupted : "+e.getMessage());
                }
        }
}

public class MainThreads{
        public static void main(String[] args){
                Thread thread1=new Thread(new MyRunnable());
                Thread thread2=new Thread(new MyRunnable());
                Thread thread3=new Thread(new MyRunnable());

                thread1.start();
                thread2.start();
                thread3.start();
        }
}
```

Output

```
Thread ID : 28 is running !
Thread ID : 29 is running !
Thread ID : 30 is running !
```

12. Develop a program to

    1. create a class MyThread in this class a constructor
    2. call the base class constructor, using super and start the thread.
    3. The run method of the class starts after this.

    It can be observed that both main thread and created child thread are executed concurrently.

```
class MyThread extends Thread{
        public MyThread(String name){
                super(name);
                start();
        }

        public void run(){
                for(int i=1;i<=5;i++){
                        System.out.println("Child Thread : "+i);
                        try{
                                Thread.sleep(1000);
                        } catch (InterruptedException e){
                                System.out.println("Child thread interrupted !");
                        }
                }
        }
}

public class MainThread{
        public static void main(String[] args){
                MyThread myThread=new MyThread("ChildThread");

                for(int i=1;i<=5;i++){
                        System.out.println("Main Thread : "+i);
                        try{
                                Thread.sleep(1000);
                        } catch (InterruptedException e){
                                System.out.println("Main thread interrupted !");
                        }
                }
        }
}
```

Output

```
Child Thread : 1
Main Thread : 1
Child Thread : 2
Main Thread : 2
Child Thread : 3
Main Thread : 3
Main Thread : 4
Child Thread : 4
Main Thread : 5
Child Thread : 5
```