Program 1

Develop a JAVA program to add TWO matrices of suitable order N (The value of N should be read from command line arguments).

```
public class Matrix {

        public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the value of N (Dimention): ");
        int N = scanner.nextInt();

        int[][] firstMatrix = new int[N][N];
        int[][] secondMatrix = new int[N][N];
        int[][] resultMatrix = new int[N][N];

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

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

        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++) {
                resultMatrix[i][j] = firstMatrix[i][j] + secondMatrix[i][j];
            }
        }

        System.out.println("The resultant matrix is: ");
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++) {
                System.out.print(resultMatrix[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

Program 2

Develop a stack class to hold a maximum of 10 integers with suitable methods. Develop a JAVA main method to illustrate Stack operations.

```
public class Stack_Operations {

        private  int[] stack;
    private int top;
    private int maxSize;


    public Stack_Operations(int maxSize) {
        this.maxSize = maxSize;
        stack = new int[maxSize];
        top = -1;
    }

    public void push(int item) {
        if (!isFull()) {
            stack[++top] = item;
        } else {
            System.out.println("Stack is full");
        }
    }
    public int pop(int i) {
        if (!isEmpty()) {
            return stack[top--];
        } else {
            System.out.println("Stack is empty");
            return -1;
        }
    }
    public int peek() {
        if (!isEmpty()) {
            return stack[top];
        } else {
            System.out.println("Stack is empty");
            return -1;
        }
    }
    public boolean isEmpty() {
        return top == -1;
    }
    public boolean isFull() {
        return top == maxSize - 1;
    }
    public int size() {
```

```
        return top + 1;
    }
    public void printStack() {
        for (int i = 0; i <= top; i++) {
            System.out.print(stack[i] + " ");
        }
        System.out.println();
    }


    public static void main(String[] args)
    {

        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter stack size:");
        int N = scanner.nextInt();

        Stack_Operations stack = new Stack_Operations(N);

        System.out.println("Pushing integers onto the stack:");
        for (int i = 0; i < N; i++)
        {
        stack.push(i);
        }

        System.out.println("Stack after pushing:");
        stack.printStack();
        System.out.println("Eenter how many elements you want
          to pop:");
        int n= scanner.nextInt();
        System.out.println("Popping integers off the stack:");

        for (int i = 0; i < n; i++)
        {
            stack.pop(i);

        }
        System.out.println("Stack after pop:");
    stack.printStack();
    }
}
```

Program 3

A class called Employee, which models an employee with an ID, name and salary, is designed as shown in the following class diagram. The method raiseSalary (double percent) increases the salary by the given percentage. Develop the Employee class and suitable main method for demonstration.

```
public class Employee {

        private int id;
        private String name;
        private double salary;

        public Employee(int id, String name, double salary) {
            this.id = id;
            this.name = name;
            this.salary = salary;
        }

        public void raiseSalary(double percent) {
            double increase = salary * percent / 100;
            salary += increase; // salary =salary + increase;
        }

        public int getId() {
            return id;
        }

        public String getName() {
            return name;
        }

        public double getSalary() {
            return salary;
        }

        public String toString()
        {
             return "Emplyee{"+"name="+ name + ", id=" + id + ",
             Salary=" + salary + '}' ;
        }


        public static void main(String[] args)
        {
              Employee_Sal_Inc employee = new
              Employee_Sal_Inc(123, "Rohan", 75000);
```

```
            System.out.println("Employee before salary increase:
        "+ employee);
            System.out.println("Enter how much percent increment
              you want to give to employee:");
            Scanner sc=new Scanner(System.in);
            double i=sc.nextDouble();
            employee.raiseSalary(i);
            System.out.println("Employee  salary after increase: "
        + employee);
          }
}
```

Program 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.**

```
public class MyPoint {
   private int x;
   private int y;

   public MyPoint() {
       this.x = 0;
      this.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() {
        return new int[]{x, y};
    }

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

    public double distance(int x, int y) {
        return Math.sqrt(Math.pow(this.x - x, 2) + Math.pow(this.y - y,
2));
    }

    public double distance(MyPoint another) {
        return Math.sqrt(Math.pow(this.x - another.x, 2) +
Math.pow(this.y - another.y, 2));
    }

    public double distance() {
        return Math.sqrt(Math.pow(this.x, 2) + Math.pow(this.y, 2));
    }
    public static void main(String[] args)
    {
         MyPoint p1 = new MyPoint();
        System.out.println(p1);
        MyPoint p2 = new MyPoint(3, 4);
        System.out.println(p2);
        System.out.println("Enter coordinates x & y:");
        Scanner sc = new Scanner(System.in);
        int x = sc.nextInt();
        int y=sc.nextInt();
            p1.setXY(x, y);
        System.out.println(p1);
        int[] xy = p1.getXY();
        System.out.println(xy[0]);
        System.out.println(xy[1]);
        System.out.println(p1.distance(2, 3));
        System.out.println(p1.distance(p2));
        System.out.println(p1.distance());

    }
}
```

Program 5

Develop a JAVA program to create a class named shape. Create three sub classes namely: circle, triangle and square, each class has two member functions named draw () and erase (). Demonstrate polymorphism concepts by developing suitable methods, defining member data and main program.

```
abstract class Shape {
        protected String name;

        public Shape(String name) {
             this.name = name;
      }
        public abstract void draw();
        public abstract void erase();

      }

class Circle extends Shape {
        private double r;

        public Circle(String name, int r) {
           super(name);
           this.r = r;
        }
        @Override
        public void draw() {
      System.out.println("Drawing Circle with name: " + name + " and
 radius: " + r);
        }
        @Override
        public void erase() {
           System.out.println("Erasing Circle with name: " + name + "
and radius: " + r);
        }
      }

class Triangle extends Shape {
        private int j;
        private int k;
        public Triangle(String name, int j, int k) {
           super(name);
           this.j = j;
           this.k = k;
        }
        @Override
        public void draw() {
           System.out.println("Drawing Triangle with name: " + name
+ ", base: " + j + " and height: " + k);
        }
```

```
        @Override
        public void erase() {
            System.out.println("Erasing Triangle with name: " + name
+ ", base: " + j + " and height: " + k);
        }
    }

class Square extends Shape {
        private int l;
        public Square(String name, int l) {
            super(name);
            this.l = l;
        }
        @Override
            public void draw() {
                System.out.println("Drawing Square with name: " +
name + " and side: " + l);
                }
        @Override
            public void erase() {
                System.out.println("Erasing Square with name: " +
name + " and side: " + l);
        }

}
```

Program 6

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

```
abstract class Shape {
    private String name;
        public Shape(String name) {
                this.name = name;
        }
        public String getName() {
        return name;
    }
    public abstract double calculateArea();
    public abstract double calculatePerimeter();
 }

class Circle  extends Shape{
    private double r;//radius is declared as variable i
    public Circle(String name, double r) {
        super(name);
        this.r = r;
    }
    @Override
    public double calculateArea() {
        return Math.PI * r * r;
    }
    @Override
    public double calculatePerimeter() {
        return 2 * Math.PI * r;
    }
}

class Tringle extends Shape{
    private double sideA;
    private double sideB;
    private double sideC;
    public Tringle(String name, double sideA, double sideB, double
sideC) {
        super(name);
        this.sideA = sideA;
        this.sideB = sideB;
        this.sideC = sideC;
    }
    public double calculateArea() {
        double semiperimeter = (sideA + sideB + sideC) / 2;
```

```
        return Math.sqrt(semiperimeter * (semiperimeter - sideA) *
(semiperimeter - sideB) * (semiperimeter - sideC));
    }
    public double calculatePerimeter() {
        return sideA + sideB + sideC;
    }
}

public class Shape_Main {
    public static void main(String[] args) {
    double i,j,k,l;

    System.out.println("Enter Radius of circle:");
    Scanner sc=new Scanner(System.in);
    i=sc.nextDouble();
    Circle circle = new Circle("Circle",+ i);
    System.out.println("Area of circle: " + circle.calculateArea());
    System.out.println("Perimeter of circle: " +
circle.calculatePerimeter());

    System.out.println("Enter sides of Triangle j, k, l:");
    j=sc.nextDouble();
    k=sc.nextDouble();
    l=sc.nextDouble();
    Tringle triangle = new Tringle("Triangle", +j , + k, + l);
    System.out.println("Area of triangle: " + triangle.calculateArea());
    System.out.println("Perimeter of Triangle:"+
triangle.calculatePerimeter());
    }
}
```

Program 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

```
interface Resizable {
        void resizeWidth(int width);
        void resizeHeight(int height);
    }
    class Rectangle implements Resizable {
        private int width;
        private int height;
        public Rectangle(int width, int height) {
            this.width = width;
            this.height = height;
        }
        @Override
        public void resizeWidth(int width) {
            this.width = width;
        }
        @Override
        public void resizeHeight(int height) {
            this.height = height;
        }
        public int getWidth() {
            return width;
        }
        public int getHeight() {
            return height;
        }
    }


public class Resizable_main {
    public static void main(String[] args) {
        Rectangle rectangle = new Rectangle(10, 20);
        System.out.println("Original width: " + rectangle.getWidth());
        System.out.println("Original height: " + rectangle.getHeight());
        rectangle.resizeWidth(30);
        rectangle.resizeHeight(40);
        System.out.println("New width: " + rectangle.getWidth());
        System.out.println("New height: " + rectangle.getHeight());
    }
}
```

Program 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.

```
public class Outer_Inner {
        public void display() {
            System.out.println("Outer class display method:");
            System.out.println("In java first outer class will
execute....");
        }
              class Inner {
            public void display() {
                System.out.println("Inner class display method:");
                System.out.println("After outer class inner class will
execute....");
            }
        }
    }

public class Outer_Inner_main {
        public static void main(String[] args) {
             Outer_Inner outer = new Outer_Inner();
            outer.display(); // Outer class display method
            Outer_Inner.Inner inner = outer.new Inner();
            inner.display(); // Inner class display method
        }
    }
```

Program 9

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

```
public class Custom_Exception {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter two numbers: ");
        int a = scanner.nextInt();
        int b = scanner.nextInt();
        try {
            if (b == 0) {
                throw new ArithmeticException("Division by zero is not
allowed!");
            }
            float result = a / b;
            System.out.println("Result: " + result);
        } catch (ArithmeticException e) {
            System.out.println(e.getMessage());
        } finally {
            scanner.close();

        }
    }
}
```

Program 10

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

```
package mypack_10;

interface  Mypackage_interface {
            void display();
        }

public class MyClass implements Mypackage_interface {
    public void display() {
        System.out.println("Hello!");
        System.out.println("Hi...");
        System.out.println("You -)");
        System.out.println("Yes You..");
        System.out.println("Hello, You only.");
        System.out.println("this is the result-)-)-)");
    }

}
```

```
import mypack_10.MyClass;

public class Mypack_main {
        public void display() {
                System.out.println("Yes its working");
        }
        public static void main(String[] args) {
        MyClass obj = new MyClass();
        Mypack_main obj1 = new Mypack_main();

        obj.display();
        obj1.display();
    }
}
```

Program 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).

```
public class Thread_Example implements Runnable {

                private String name;
                    public Thread_Example(String name) {
                   this.name = name;
                }
                @Override
                public void run() {
                   System.out.println("Thread started: " + name);
                   try {
                      Thread.sleep(500);
                   } catch (InterruptedException e) {
                      e.printStackTrace();
                   }
                   System.out.println("Thread ended: " + name);
                }
                public static void main(String[] args) {
                    Thread_Example runnableExample1 = new
                            Thread_Example("Thread-1");
                    Thread_Example runnableExample2 = new
                            Thread_Example("Thread-2");
                    Thread_Example runnableExample3 = new
                            Thread_Example("Thread-3");
                  Thread thread1 = new Thread(runnableExample1);
                  Thread thread2 = new Thread(runnableExample2);
                  Thread thread3 = new Thread(runnableExample3);
                  thread1.start();
                  thread2.start();
                  thread3.start();
                }
        }
```

Program 12

Develop a program to create a class MyThread in this class a constructor, call the base class constructor, using super and start the thread. The run method of the class starts after this. It can be observed that both main thread and created child thread are executed concurrently.

```
public class MyThread extends Thread {
        public MyThread() {
            super();
            System.out.println("Child Thread:");
        }
        @Override
        public void run() {
            try {
                for (int i = 0; i < 5; i++) {
                    System.out.println("Child Thread: " + i);
                    Thread.sleep(500);
                }
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        public static void main(String[] args) {
            MyThread t1 = new MyThread();
            t1.start();
            try {
                for (int i = 0; i < 5; i++) {
                    System.out.println("Main Thread: " + i);
                    Thread.sleep(500);
                }
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
```