VIDYA JYOTHI INSTITUTE OF TECHNOLOGY

(An Autonomous Institution)

(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)

B.Tech(CSE-DS) II Year / II Semester (R22)

LAB MANUAL

Name of the Faculty	**KISHORE K**
Department	**CSE(Data Science)**
Year & Semester	**B.Tech-II & II Sem**
Regulation	**R22**
Lab Name	**OBJECT ORIENTED PROGRAMMING THROUGH JAVA**

DEPARTMENT OF CSE (Data Science)

VIDYA JYOTHI INSTITUTE OF TECHNOLOGY

(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)

An Autonomous Institution

AZIZ NAGAR, C B POST, HYDERABAD-500075

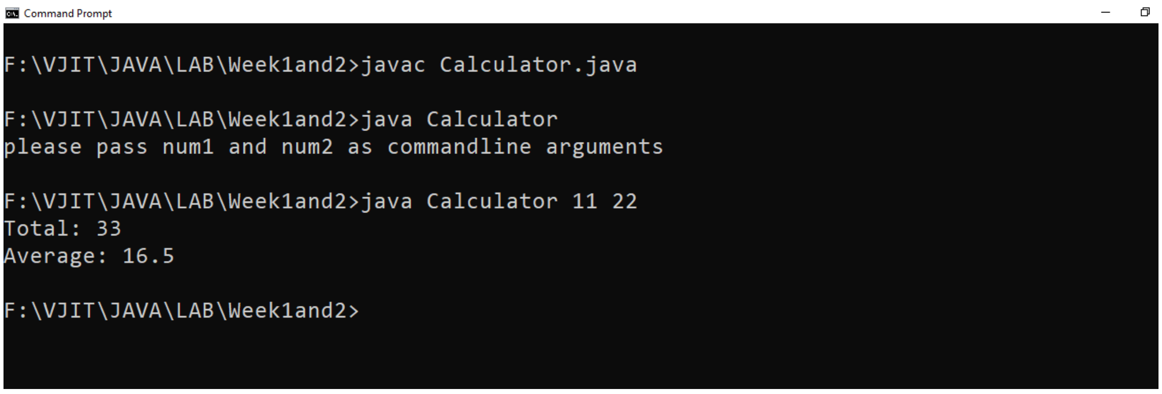
2024-2025

INDEX

1. Write a program to find total, average of given two numbers by **using method** with **command-line arguments, static data members**.

```
public class Calculator
{
    // Static data members
    static int num1, num2,total;
   static double average;
    // Function to add the numbers and return the total
       public static int addNumbers()
      {
           return num1 + num2;
       }
        // Function to calculate the average and return the average
          public static double calculateAverage()
        {
           return (num1 + num2) / 2.0;
        }
        // Function to display results
       public static void displayResults()
      {
        System.out.println("Total: " + total);
        System.out.println("Average: " + average);
      }
    public static void main(String args[])
   {
        if (args.length != 2)
        {
     System.out.println("please pass num1 and num2 as commandline arguments");
            return;
        }
        num1 = Integer.parseInt(args[0]);
        num2 = Integer.parseInt(args[1]);
        total = addNumbers();
        average = calculateAverage();
        displayResults();
    }
}
```

Output:

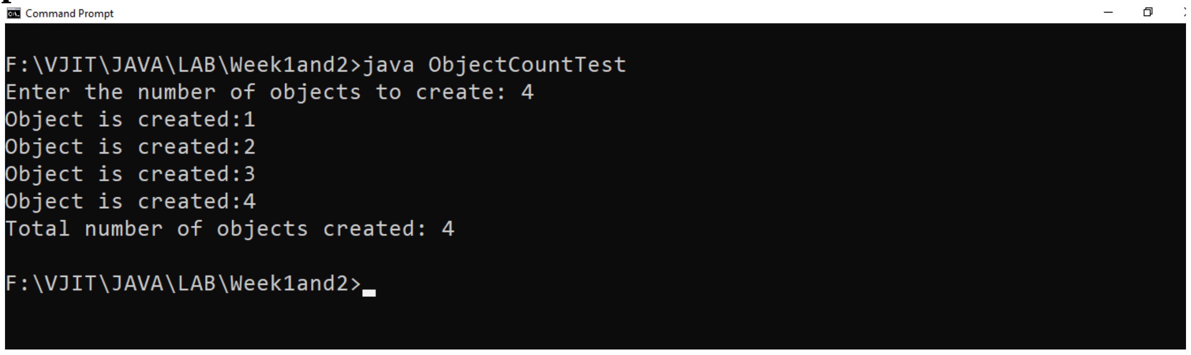


```
F:\VJIT\JAVA\LAB\Week1and2>javac Calculator.java

F:\VJIT\JAVA\LAB\Week1and2>java Calculator
please pass num1 and num2 as commandline arguments

F:\VJIT\JAVA\LAB\Week1and2>java Calculator 11 22
Total: 33
Average: 16.5

F:\VJIT\JAVA\LAB\Week1and2>
```

2. Write a program to illustrate **class** and **objects**.

```
import java.util.Scanner;
class ObjectCounter
{
   // Static variable to keep track of the number of objects
     private static int count = 0;
    public ObjectCounter()
   {
     // Constructor increments the object count when an object is created
     count++;
     System.out.println("Object is created:"+count);
   }
   public static int getObjectCount()
   {
    // Static method to get the current count of objects &  return count
      return count;
   }
}
public class ObjectCountTest
{
      public static void main(String[] args)
       {
       Scanner sc = new Scanner(System.in);
       // Prompt user for the number of objects to create
       System.out.print("Enter the number of objects to create: ");
       int numObjects = sc.nextInt();
       // Loop to create the specified number of objects
       for (int i = 0; i < numObjects; i++)
        {
          new ObjectCounter(); // Creating an object will increment the count
          }
       // Display the total count of objects created
System.out.println("Total number of objects created: " + ObjectCounter.getObjectCount());
       // Close the scanner
       sc.close();
   }
}
```

Output:

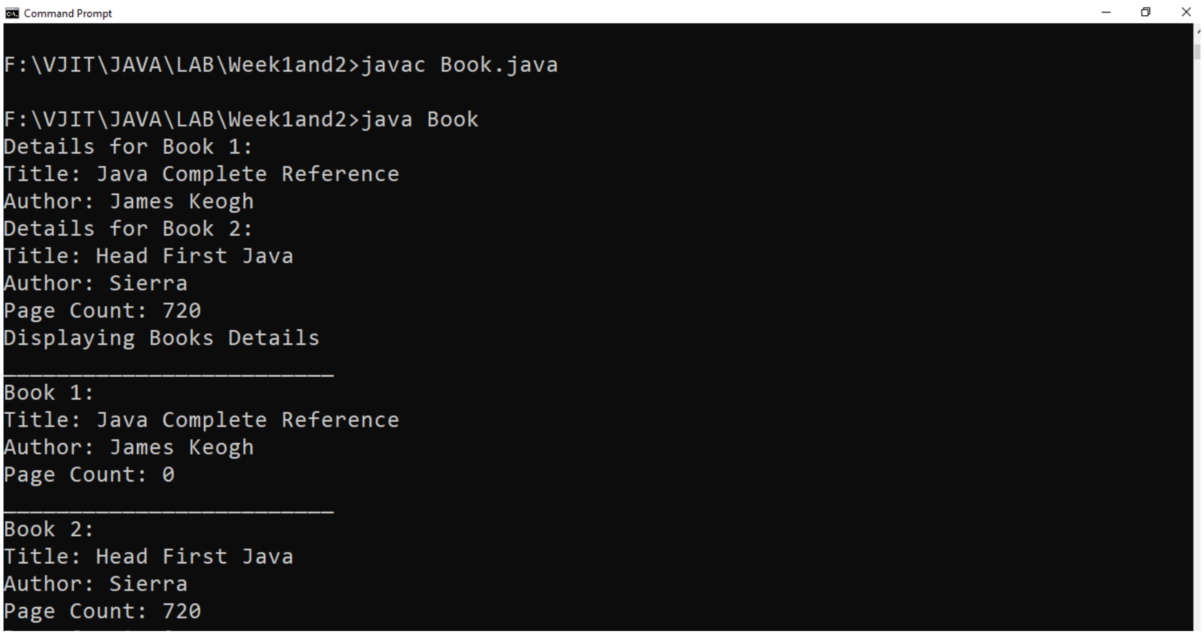


```
F:\VJIT\JAVA\LAB\Week1and2>java ObjectCountTest
Enter the number of objects to create: 4
Object is created:1
Object is created:2
Object is created:3
Object is created:4
Total number of objects created: 4

F:\VJIT\JAVA\LAB\Week1and2>
```

3.1 Write a program to illustrate **Constructor Overloading**.

```
import java.util.Scanner;
public class Book
{
 // Attributes of the Book class
    private String title;
    private String author;
    private int pageCount;

    // Default constructor
    public Book()
   {
        this.title = "Unknown Title";
        this.author = "Unknown Author";
        this.pageCount = 666;
    }
    // Constructor with title and author parameters
    public Book(String title, String author)
  {
        this.title = title;
        this.author = author;
        //this.pageCount = 0;  // Default page count
    }
    // Constructor with title, author, and pageCount parameters
    public Book(String title, String author, int pageCount)
    {
        this.title = title;
        this.author = author;
        this.pageCount = pageCount;
      }
    // Method to display book details
    public void displayBookDetails()
    {
        System.out.println("Title: " + title);
        System.out.println("Author: " + author);
        System.out.println("Page Count: " + pageCount);
    }
    public static void main(String[] args)
   {
        Scanner scanner = new Scanner(System.in);

      // Taking user inputs for Book 1
        System.out.println("Details for Book 1:");
        System.out.print("Title: ");
        String title1 = scanner.nextLine();
        System.out.print("Author: ");
        String author1 = scanner.nextLine();

// Creating Book 1 using the constructor with title and author parameters
        Book book1 = new Book();

 // Taking user inputs for Book 2
        System.out.println("Details for Book 2:");
        System.out.print("Title: ");
        String title2 = scanner.nextLine();
        System.out.print("Author: ");
        String author2 = scanner.nextLine();
        System.out.print("Page Count: ");
        int pageCount2 = scanner.nextInt();

 // Creating Book 2 using the constructor with all parameters
        Book book2 = new Book(title2, author2, pageCount2);

 // Displaying details of each book
          System.out.println("Displaying Books Details");
          System.out.println("___________________________");
        System.out.println("Book 1:");
        book1.displayBookDetails();
        System.out.println("___________________________");
        System.out.println("Book 2:");
        book2.displayBookDetails();
        // Close the scanner
        scanner.close();
    }
}
```

Output:



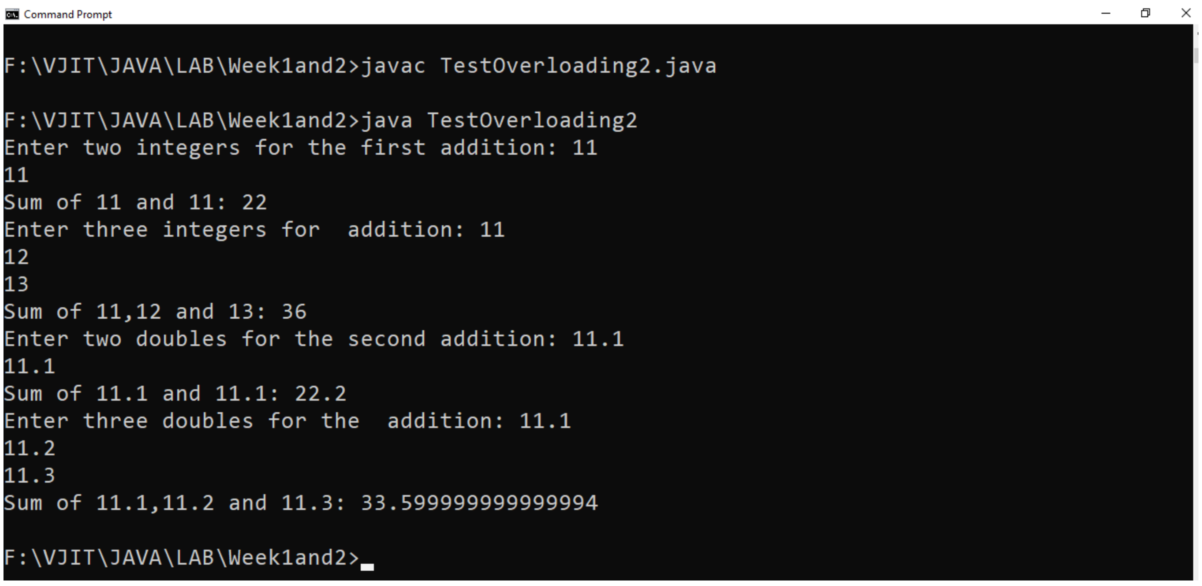
```
F:\VJIT\JAVA\LAB\Week1and2>javac Book.java

F:\VJIT\JAVA\LAB\Week1and2>java Book
Details for Book 1:
Title: Java Complete Reference
Author: James Keogh
Details for Book 2:
Title: Head First Java
Author: Sierra
Page Count: 720
Displaying Books Details
_________________________
Book 1:
Title: Java Complete Reference
Author: James Keogh
Page Count: 0
_________________________
Book 2:
Title: Head First Java
Author: Sierra
Page Count: 720
```

3.2 Write a program to illustrate **method overloading**.

```
import java.util.Scanner;

public class TestOverloading2
{
   public static void main(String[] args)
   {
      Scanner scanner = new Scanner(System.in);
       System.out.print("Enter two integers for the first addition: ");
          int num1 = scanner.nextInt();
          int num2 = scanner.nextInt();
          int intSum2 = sum(num1, num2); // Sum of two integers
         System.out.println("Sum of "+num1+" and "+num2+": "+intSum2);
         System.out.print("Enter three integers for  addition: ");
           int num3 = scanner.nextInt();
          int num4 = scanner.nextInt();
          int num5 = scanner.nextInt();
          int intSum3 = sum(num3, num4, num5); // Sum of three integers
 System.out.println("Sum of "+num3+","+num4+" and "+num5+": "+intSum3);
      System.out.print("Enter two doubles for the second addition: ");
     double num6 = scanner.nextDouble();
      double num7 = scanner.nextDouble();
         double doubleSum2 = sum(num6, num7); // Sum of two doubles
 System.out.println("Sum of "+num6+" and "+num7+": "+doubleSum2);
          System.out.print("Enter three doubles for the  addition: ");
         double num8 = scanner.nextDouble();
      double num9 = scanner.nextDouble();
      double num10 = scanner.nextDouble();
 double doubleSum3 = sum(num8, num9, num10); // Sum of three doubles
System.out.println("Sum of "+num8+","+num9+" and "+num10+": "+doubleSum3);
      scanner.close();
   }
  // Method to sum two integers
   public static int sum(int a, int b)
  {
      return a + b;
   }
```

```
    // Method to sum three integers
    public static int sum(int a, int b, int c)
    {
       return a + b + c;
    }

    // Method to sum two doubles
    public static double sum(double a, double b)
     {
       return a + b;
    }

    // Method to sum three doubles
    public static double sum(double a, double b, double c)
   {
       return a + b + c;
    }
}
```

Output:



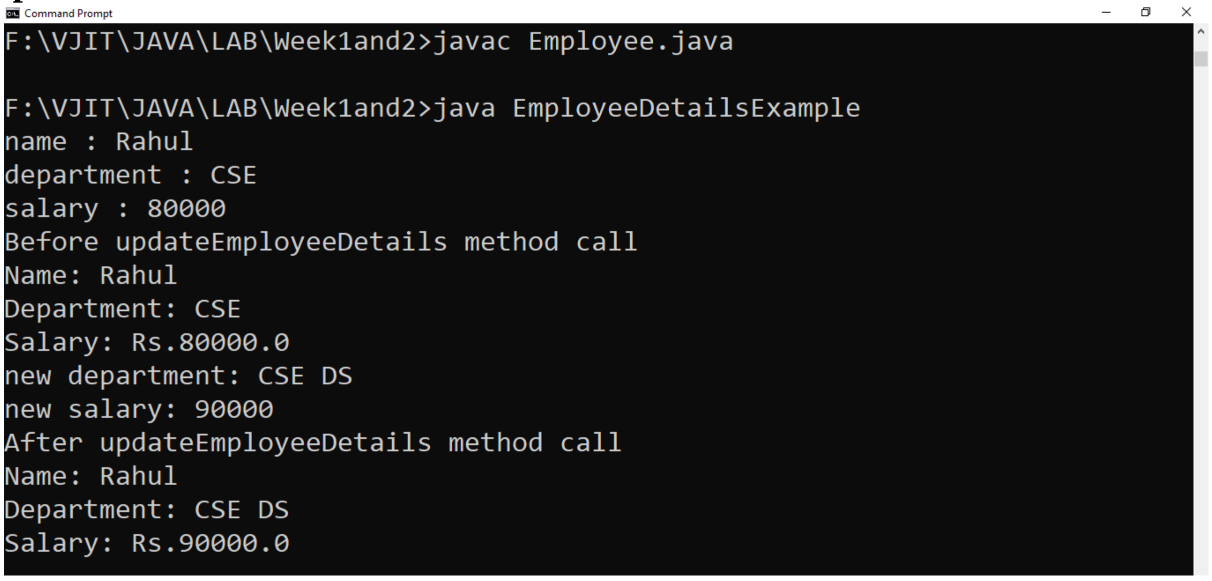
```
F:\VJIT\JAVA\LAB\Week1and2>javac TestOverloading2.java

F:\VJIT\JAVA\LAB\Week1and2>java TestOverloading2
Enter two integers for the first addition: 11
11
Sum of 11 and 11: 22
Enter three integers for  addition: 11
12
13
Sum of 11,12 and 13: 36
Enter two doubles for the second addition: 11.1
11.1
Sum of 11.1 and 11.1: 22.2
Enter three doubles for the  addition: 11.1
11.2
11.3
Sum of 11.1,11.2 and 11.3: 33.599999999999994

F:\VJIT\JAVA\LAB\Week1and2>
```

4. Write a program to illustrate **Parameter Passing** using objects.

```
import java.util.Scanner;
// A class representing an Employee with name, department, and salary.
public class Employee
 {
   String name;
   String department;
   double salary;

   // Constructor to initialize Employee object
    Employee(String name, String department, double salary)
   {
       this.name = name;
       this.department = department;
       this.salary = salary;
    }

   // Method to display the Employee details on separate lines
   void display()
  {
       System.out.println("Name: " + name);
       System.out.println("Department: " + department);
       System.out.println("Salary: Rs." + salary);
    }

   // Method to update the Employee's department
   void updateDepartment(String newDepartment)
  {
       this.department = newDepartment;
    }

   // Method to update the Employee's salary
   void updateSalary(double newSalary)
  {
       this.salary = newSalary;
    }
}
```

```
class EmployeeDetailsExample
 {
  // A method that accepts an Employee object and modifies its details
   public static void updateEmployeeDetails(Employee employee, String
newDepartment, double newSalary)
 {
       // Modifying the Employee object's fields
        // Changing the department
       employee.updateDepartment(newDepartment);
       // Changing the salary
           employee.updateSalary(newSalary);
    }
   public static void main(String[] args)
  {
       Scanner scanner = new Scanner(System.in);
       System.out.print("name : ");
       String name = scanner.nextLine();
           System.out.print("department : ");
       String department = scanner.nextLine();
           System.out.print("salary : ");
       double salary = scanner.nextDouble();
       // Create an Employee object using user input
       Employee employee = new Employee(name, department, salary);
       // Display the initial details of the employee
       System.out.println("Before updateEmployeeDetails method call");
       employee.display();
       scanner.nextLine();
       System.out.print("new department: ");
       String newDepartment = scanner.nextLine();
           System.out.print("new salary: ");
       double newSalary = scanner.nextDouble();
       // Update the Employee details with new department and salary
       updateEmployeeDetails(employee, newDepartment, newSalary);
       // Display the details of the employee after modification
       System.out.println("After updateEmployeeDetails method call");
       employee.display();
           scanner.close();
    }
}
```

Output:

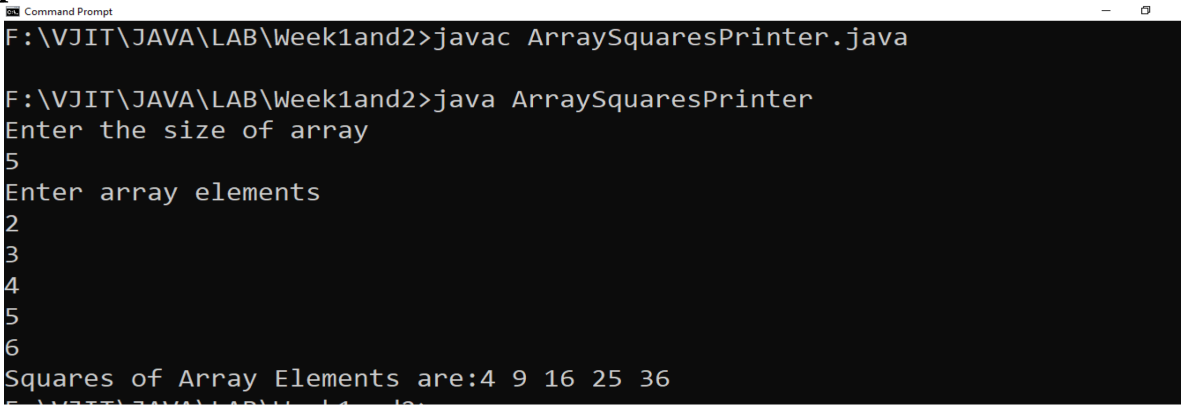


```
F:\VJIT\JAVA\LAB\Week1and2>javac Employee.java

F:\VJIT\JAVA\LAB\Week1and2>java EmployeeDetailsExample
name : Rahul
department : CSE
salary : 80000
Before updateEmployeeDetails method call
Name: Rahul
Department: CSE
Salary: Rs.80000.0
new department: CSE DS
new salary: 90000
After updateEmployeeDetails method call
Name: Rahul
Department: CSE DS
Salary: Rs.90000.0
```

5. Write a program to illustrate **Array Manipulation.**

```
import java.util.Scanner;
public class ArraySquaresPrinter
 {
   public static void main(String[] args)
   {
          // Create a Scanner object for input
      Scanner sc = new Scanner(System.in);
      // Read the size of the array
         System.out.println("Enter the size of array");
         int N = sc.nextInt();
      // Initialize an array to store the elements
      int[] arr = new int[N];
      System.out.println("Enter array elements");
      // Read the array elements
      for (int i = 0; i < N; i++)
     {
         arr[i] = sc.nextInt();
      }
      // Print the squares of the array elements
         System.out.print("Squares of Array Elements are:");
      for (int i = 0; i < N; i++)
      {
         System.out.print(arr[i] * arr[i] + " ");
      }
      // Close the scanner
      sc.close();
   }
}
```

Output:



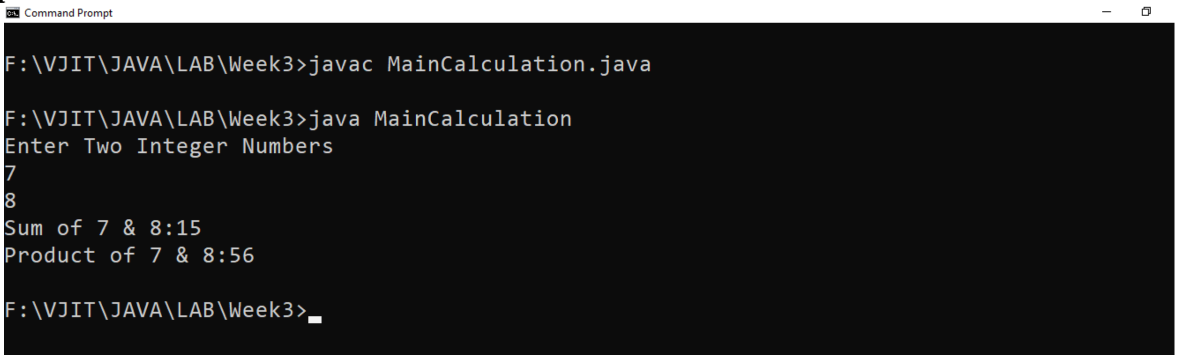
```
F:\VJIT\JAVA\LAB\Week1and2>javac ArraySquaresPrinter.java

F:\VJIT\JAVA\LAB\Week1and2>java ArraySquaresPrinter
Enter the size of array
5
Enter array elements
2
3
4
5
6
Squares of Array Elements are:4 9 16 25 36
```

6.1 Addition and Multiplication using **Inheritance** by creating a simple calculator application that performs addition and multiplication operations.

```
import java.util.Scanner;
class Calculation
{
    protected int num1, num2;
    // Constructor to initialize the two integers
    public Calculation(int num1, int num2)
   {
        this.num1 = num1;
        this.num2 = num2;
    }
    // Method to perform addition and return the result
    public int addition()
      {
        return num1 + num2;
        }
}
class My_Calculation extends Calculation
{
    // Constructor that calls the parent constructor
    public My_Calculation(int num1, int num2)
      {
        super(num1, num2);
    }
    // Method to perform multiplication and return the result
    public int multiplication()
   {
        return num1 * num2;
    }
}
```

```
public class MainCalculation
{
   public static void main(String[] args)
     {
      Scanner scanner = new Scanner(System.in);
      System.out.println("Enter Two Integer Numbers");
      int num1 = scanner.nextInt();
      int num2 = scanner.nextInt();

      My_Calculation myCalculation = new My_Calculation(num1, num2);
      int sum = myCalculation.addition();
      int product = myCalculation.multiplication();
      System.out.println("Sum of "+num1+" & "+num2+":"+sum);
      System.out.println("Product of "+num1+" & "+num2+":"+product);
      scanner.close();
   }
}
```

Output:



```
F:\VJIT\JAVA\LAB\Week3>javac MainCalculation.java

F:\VJIT\JAVA\LAB\Week3>java MainCalculation
Enter Two Integer Numbers
7
8
Sum of 7 & 8:15
Product of 7 & 8:56

F:\VJIT\JAVA\LAB\Week3>
```

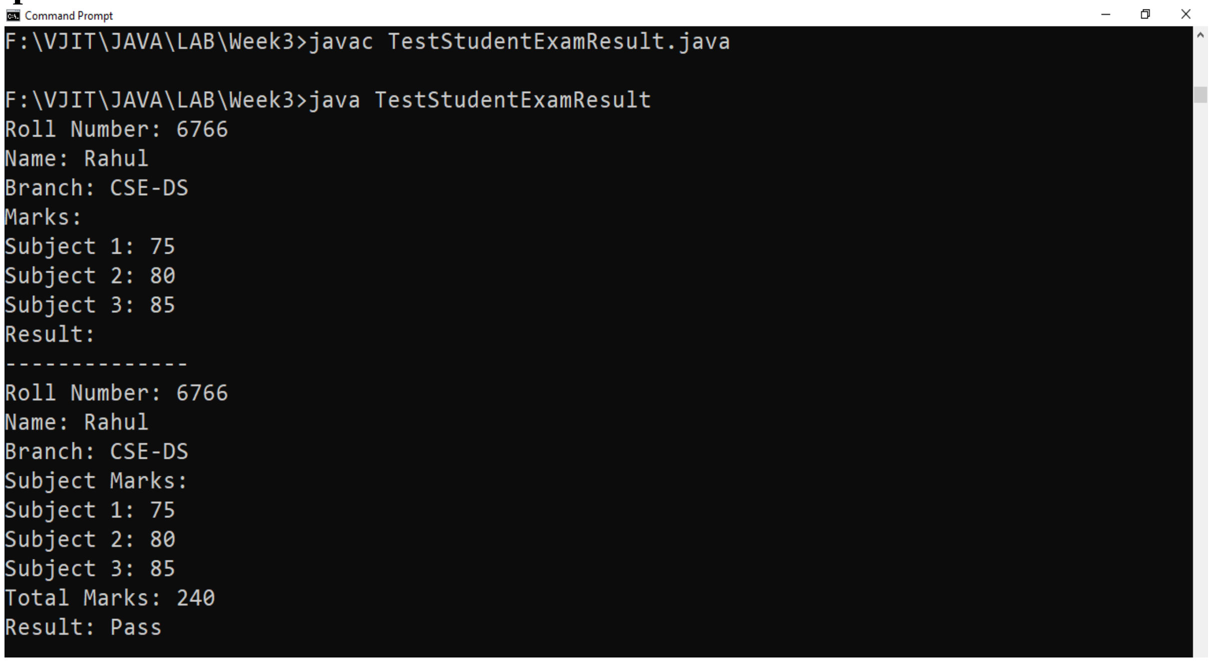
6.2 Write a Java program to model a **multi-level inheritance** scenario involving classes Student, Exam, and Result.

```
import java.util.Scanner;
class Student
{
    protected int rollNumber;
    protected String name;
    protected String branch;
        public Student(int rollNumber, String name, String branch)
        {
        this.rollNumber = rollNumber;
        this.name = name;
        this.branch = branch;
       }
         public void displayStudentDetails(
        {
        System.out.println("Roll Number: " + rollNumber);
        System.out.println("Name: " + name);
        System.out.println("Branch: " + branch);
       }
}
class Exam extends Student
{
   protected int[] subjectMarks;
   // Constructor to initialize student and subject marks
        public Exam(int rollNumber, String name, String branch, int[] subjectMarks)
        {
        super(rollNumber, name, branch);
        this.subjectMarks = subjectMarks;
       }
         public void displayExamDetails()
        {
        displayStudentDetails();
        System.out.println("Subject Marks:");
           for (int i = 0; i < subjectMarks.length; i++)
           {
            System.out.println("Subject " + (i+1) + ": " + subjectMarks[i]);
        }
    }
}
```

```
class Result extends Exam
{
    private int totalMarks;
    private String result;

    // Constructor to initialize student details, subject marks, and calculate the result
       public Result(int rollNumber, String name, String branch, int[] subjectMarks)
        {
        super(rollNumber, name, branch, subjectMarks);
        calculateResult();
        }
    // Method to calculate total marks and determine the result
        private void calculateResult()
        {
        totalMarks = 0;
            for (int marks : subjectMarks)
            {
            totalMarks += marks;
            }
        // Determine result based on total marks
        result = totalMarks >= 150 ? "Pass" : "Fail"; // Assuming 150 is the passing criteria

       }

       public void displayResult()
        {
        System.out.println("Result:");
        System.out.println("--------------");
        displayExamDetails();
        System.out.println("Total Marks: " + totalMarks);
        System.out.println("Result: " + result);
    }
}
```

```
public class TestStudentExamResult
{
        public static void main(String[] args)
        {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Roll Number: ");
        int rollNumber = scanner.nextInt();
        scanner.nextLine();
        System.out.print("Name: ");
        String name = scanner.nextLine();
        System.out.print("Branch: ");
        String branch = scanner.nextLine();
        int[] subjectMarks = new int[3];
        System.out.println("Marks:");
        for (int i = 0; i < 3; i++)
        {
           System.out.print("Subject " + (i+1) + ": ");
           subjectMarks[i] = scanner.nextInt();
        }
        Result studentResult = new Result(rollNumber, name, branch, subjectMarks);
        studentResult.displayResult();
        scanner.close();
     }
}
```

Output:



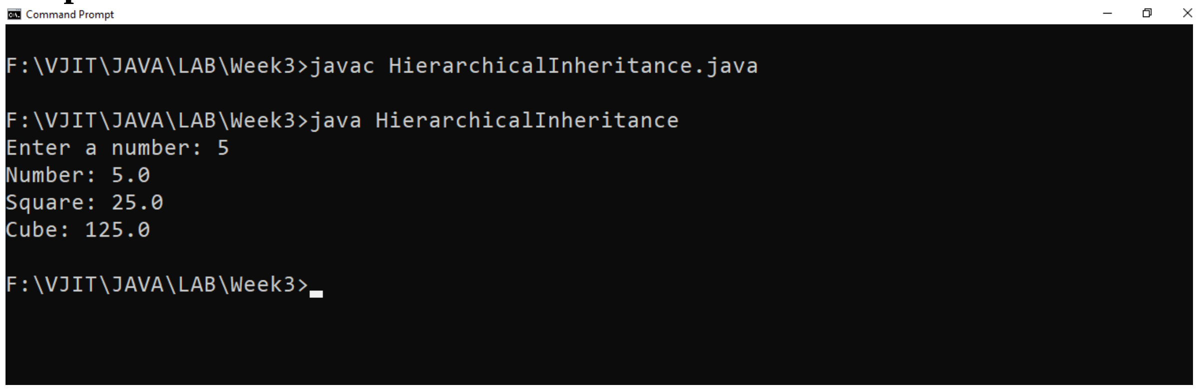
```
F:\VJIT\JAVA\LAB\Week3>javac TestStudentExamResult.java

F:\VJIT\JAVA\LAB\Week3>java TestStudentExamResult
Roll Number: 6766
Name: Rahul
Branch: CSE-DS
Marks:
Subject 1: 75
Subject 2: 80
Subject 3: 85
Result:
--------------
Roll Number: 6766
Name: Rahul
Branch: CSE-DS
Subject Marks:
Subject 1: 75
Subject 2: 80
Subject 3: 85
Total Marks: 240
Result: Pass
```

6.3 Write a Java program to demonstrate **hierarchical inheritance** involving a base class Number and two derived classes Square and Cube.

```
import java.util.Scanner;

// Base class Number
class Number
{
     double number;
     // Constructor to initialize the number
      public Number(double number)
     {
        this.number = number;
     }
}

// Derived class 1 - Square
class Square extends Number
{
    // Constructor to initialize the number for Square
    public Square(double number)
     {
        super(number);  // Call the base class constructor
     }

    // Method to calculate and return the square
    public double calculateSquare()
     {
        return number * number;
     }
}
// Derived class 2 - Cube
class Cube extends Number
{
    // Constructor to initialize the number for Cube
    public Cube(double number)
     {
        super(number);  // Call the base class constructor
     }
    // Method to calculate and return the cube
    public double calculateCube()
     {
        return number * number * number;
     }
}
```

```
// Main class
public class HierarchicalInheritance
{
   public static void main(String[] args)
    {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Enter a number: ");
      double inputNumber = scanner.nextDouble();
     // Create objects of derived classes
      Square squareObject = new Square(inputNumber);
      Cube cubeObject = new Cube(inputNumber);
      // Display the number, square, and cube
      System.out.println("Number: " + inputNumber);
      System.out.println("Square: " + squareObject.calculateSquare());
      System.out.println("Cube: " + cubeObject.calculateCube());

      scanner.close();

   }
}
```

Output:



```
F:\VJIT\JAVA\LAB\Week3>javac HierarchicalInheritance.java

F:\VJIT\JAVA\LAB\Week3>java HierarchicalInheritance
Enter a number: 5
Number: 5.0
Square: 25.0
Cube: 125.0

F:\VJIT\JAVA\LAB\Week3>
```

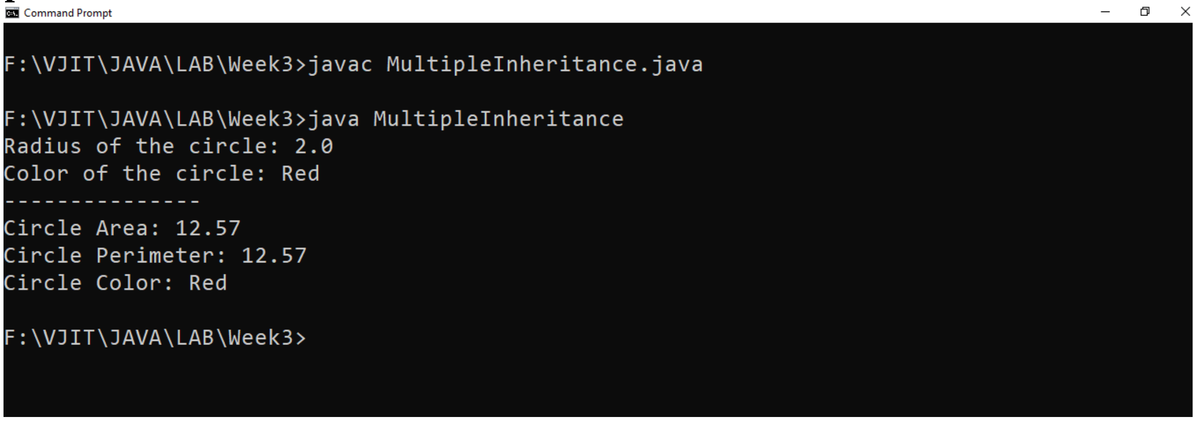
6.4 Write a Java Program to demonstrate Interface concept **(Implementation Multiple inheritance using interface),**

Implement necessary methods to calculate the Area of circle, Perimeter of the circle and getColor method to return the color and the main method has been provided.

```
import java.util.*;
interface Shape
{
   double calculateArea();
   double calculatePerimeter();
}
interface Color
{
   String getColor();
}
class Circle implements Shape, Color
{
   private double radius;
   private String color;
   // Constructor to initialize radius and color
     public Circle(double radius, String color)
     {
      this.radius = radius;
      this.color = color;
     }
   // Implementing calculateArea method from Shape interface
   @Override
     public double calculateArea()
     {
      return Math.PI * radius * radius;  // Area of a circle = π * r²
   }
   // Implementing calculatePerimeter method from Shape interface
   @Override
   public double calculatePerimeter()
     {
      return 2 * Math.PI * radius;  // Perimeter (circumference) of a circle = 2 * π * r
   }
   // Implementing getColor method from Color interface
   @Override
     public String getColor()
     {
      return color;
      }
}
```

```
public class MultipleInheritance
{
      public static void main(String[] args)
      {
       Scanner scanner = new Scanner(System.in);
       System.out.print("Radius of the circle: ");
       double radius = scanner.nextDouble();
       scanner.nextLine();
       System.out.print("Color of the circle: ");
       String color = scanner.nextLine();
       Circle circle = new Circle(radius, color);
       System.out.println("---------------");
       // Formatting double values to 2 decimal places
       System.out.printf("Circle Area: %.2f%n", circle.calculateArea());
       System.out.printf("Circle Perimeter: %.2f%n", circle.calculatePerimeter());
       System.out.println("Circle Color: " + circle.getColor());

       scanner.close();
    }
}
```

Output:



```
F:\VJIT\JAVA\LAB\Week3>javac MultipleInheritance.java

F:\VJIT\JAVA\LAB\Week3>java MultipleInheritance
Radius of the circle: 2.0
Color of the circle: Red
---------------
Circle Area: 12.57
Circle Perimeter: 12.57
Circle Color: Red

F:\VJIT\JAVA\LAB\Week3>
```

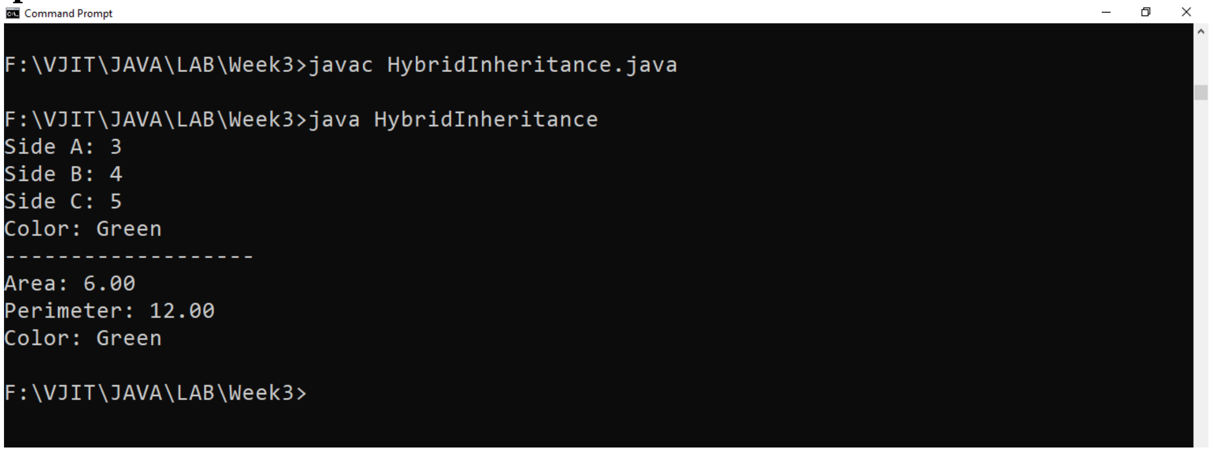
6.5 Write a Java program to demonstrate the Interface concept by implementing **Hybrid Inheritance** using interfaces.

Create a Triangle class that implements the Shape and Color interfaces. Implement the necessary methods to calculate the Area of the triangle, the Perimeter of the triangle, and return the Color of the triangle.

```
import java.util.Scanner;

interface Shape
{
   double calculateArea();      // Method to calculate area
   double calculatePerimeter(); // Method to calculate perimeter
}
interface Color
{
   String getColor();           // Method to get the color of the shape
}
class Triangle implements Shape, Color
{
   private double sideA;
   private double sideB;
   private double sideC;
   private String color;
   // Constructor to initialize the sides of the triangle and its color
    public Triangle(double sideA, double sideB, double sideC, String color)
    {
      this.sideA = sideA;
      this.sideB = sideB;
      this.sideC = sideC;
      this.color = color;
    }
 // Implementing calculateArea method from Shape interface using Heron's formula
   @Override
   public double calculateArea()
    {
      double s = (sideA + sideB + sideC) / 2;  // Semi-perimeter

      // Area calculation using Heron's formula
      return Math.sqrt(s * (s - sideA) * (s - sideB) * (s - sideC));
   }
```

```
    // Implementing calculatePerimeter method from Shape interface
    @Override
    public double calculatePerimeter()
     {
       return sideA + sideB + sideC; // Perimeter = sum of all sides
      }
    // Implementing getColor method from Color interface
    @Override
    public String getColor()
     {
       return color; // Return the color of the triangle
      }
   }
public class HybridInheritance
{
     public static void main(String[] args)
     {
       Scanner scanner = new Scanner(System.in);
       // Input: sides of the triangle and its color
       System.out.print("Side A: ");
       double sideA = scanner.nextDouble();
       System.out.print("Side B: ");
       double sideB = scanner.nextDouble();
       System.out.print("Side C: ");
       double sideC = scanner.nextDouble();
        scanner.nextLine();  // Consume newline left-over
       System.out.print("Color: ");
       String color = scanner.nextLine();
       // Create Triangle object
       Triangle triangle = new Triangle(sideA, sideB, sideC, color);
       System.out.println("-------------------");
       // Output the area, perimeter, and color of the triangle
       System.out.printf("Area: %.2f\n", triangle.calculateArea());
       System.out.printf("Perimeter: %.2f\n", triangle.calculatePerimeter());
       System.out.println("Color: " + triangle.getColor());
       scanner.close();
     }
}
```

Output:



```
F:\VJIT\JAVA\LAB\Week3>javac HybridInheritance.java

F:\VJIT\JAVA\LAB\Week3>java HybridInheritance
Side A: 3
Side B: 4
Side C: 5
Color: Green
-------------------
Area: 6.00
Perimeter: 12.00
Color: Green

F:\VJIT\JAVA\LAB\Week3>
```

7. Employee - Method Overriding

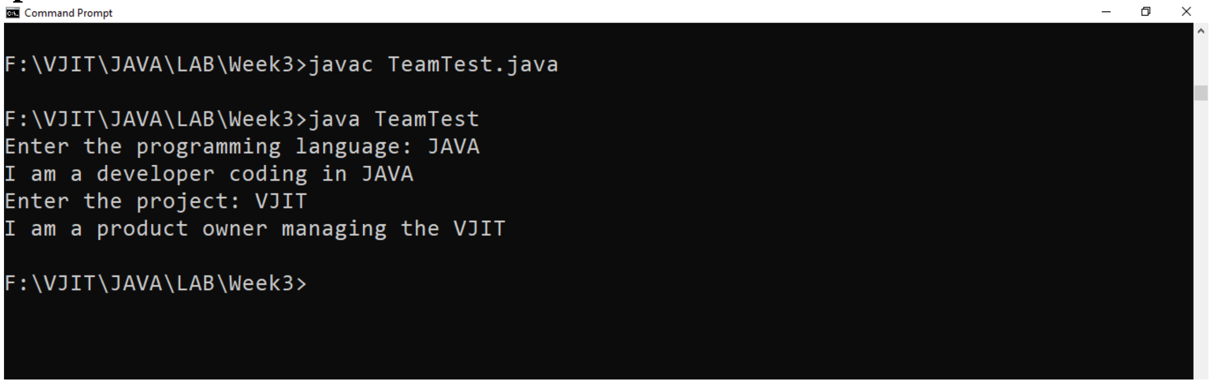
You are developing a program to simulate different roles in a software development team. Design a class named TeamMember with a method performTask() that prints "Performing a task as a team member".

Now, create two subclasses, Developer and ProductOwner, which extend the TeamMember class's functionality by overriding the performTask() method.

```
import java.util.Scanner;

class TeamMember
{
    public void performTask()
    {
      System.out.println("Performing a task as a team member");
    }
}
class Developer extends TeamMember
{
     @Override
     public void performTask()
    {
      Scanner scanner = new Scanner(System.in);
      // Prompt for the programming language
      System.out.print("Enter the programming language: ");
      String programmingLanguage = scanner.nextLine();
      // Output message indicating the developer and their programming language
      System.out.println("I am a developer coding in " + programmingLanguage);
    }

}
```

```
class ProductOwner extends TeamMember
{
     @Override
    public void performTask()
     {
        Scanner scanner = new Scanner(System.in);

        // Prompt for the project being managed
        System.out.print("Enter the project: ");
        String project = scanner.nextLine();

        // Output message indicating the product owner and their project
        System.out.println("I am a product owner managing the " + project);
    }
}
public class TeamTest
{
    public static void main(String[] args)
     {
        // Creating an instance of Developer
        Developer developer = new Developer();
        developer.performTask();

        // Creating an instance of ProductOwner
        ProductOwner productOwner = new ProductOwner();
        productOwner.performTask();
    }
}
```

Output:



```
F:\VJIT\JAVA\LAB\Week3>javac TeamTest.java

F:\VJIT\JAVA\LAB\Week3>java TeamTest
Enter the programming language: JAVA
I am a developer coding in JAVA
Enter the project: VJIT
I am a product owner managing the VJIT

F:\VJIT\JAVA\LAB\Week3>
```

8. Dynamic Method Dispatch

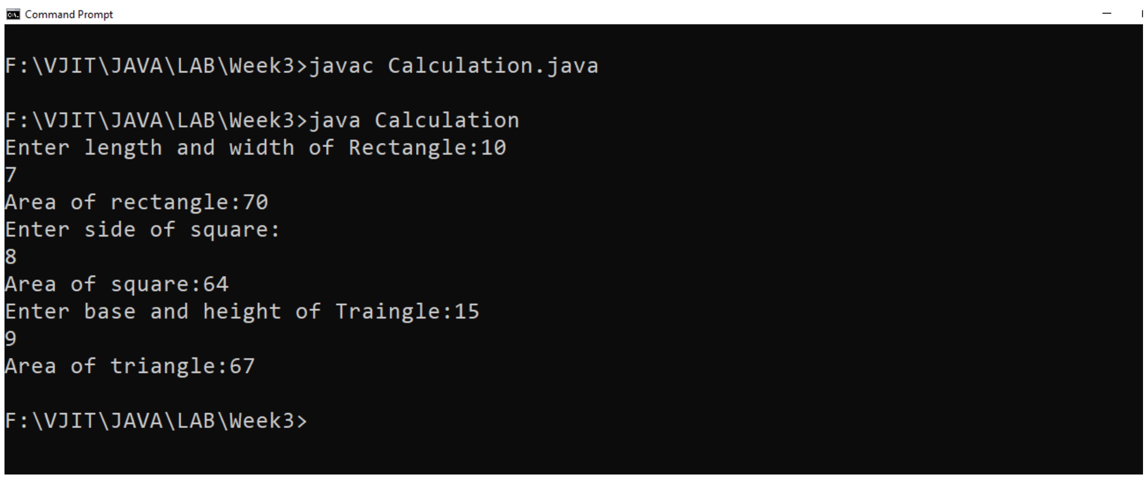
Implement runtime polymorphism using the dynamic method dispatch for the scenario given:

Create three classes Rectangle (Parent), Square(Child of Rectangle), and Triangle (Child of Rectangle). These classes contain a method "area" to calculate the area of the shape using appropriate expressions.

```
import java.util.Scanner;

class Rectangle
{

    public void area()
    {
     Scanner sc = new Scanner(System.in);
     System.out.println("Enter length and width of Rectangle:");
     double length = sc.nextDouble();
     double width = sc.nextDouble();
     int area = (int) Math.floor(length * width);
     System.out.println("Area of rectangle:" + area);
     }
}

class Square extends Rectangle
{

    public void area()
    {
     Scanner sc = new Scanner(System.in);
     System.out.println("Enter side of square:");
     int side = sc.nextInt();
     int area = side * side;
     System.out.println("Area of square:" + area);
    }
}
```

```
class Triangle extends Rectangle
{

   public void area()
    {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter base and height of Traingle:");
      double base = sc.nextDouble();
      double height = sc.nextDouble();
      int area =(int)Math.floor(0.5 * base * height);
      System.out.println("Area of triangle:" + area);
   }
}

class Calculation
{
   public static void main(String args[])
    {
      Rectangle r = new Rectangle();
      r.area();
      r = new Square();
      r.area();
      r = new Triangle();
      r.area();
   }
}
```

Output:



```
F:\VJIT\JAVA\LAB\Week3>javac Calculation.java

F:\VJIT\JAVA\LAB\Week3>java Calculation
Enter length and width of Rectangle:10
7
Area of rectangle:70
Enter side of square:
8
Area of square:64
Enter base and height of Traingle:15
9
Area of triangle:67

F:\VJIT\JAVA\LAB\Week3>
```

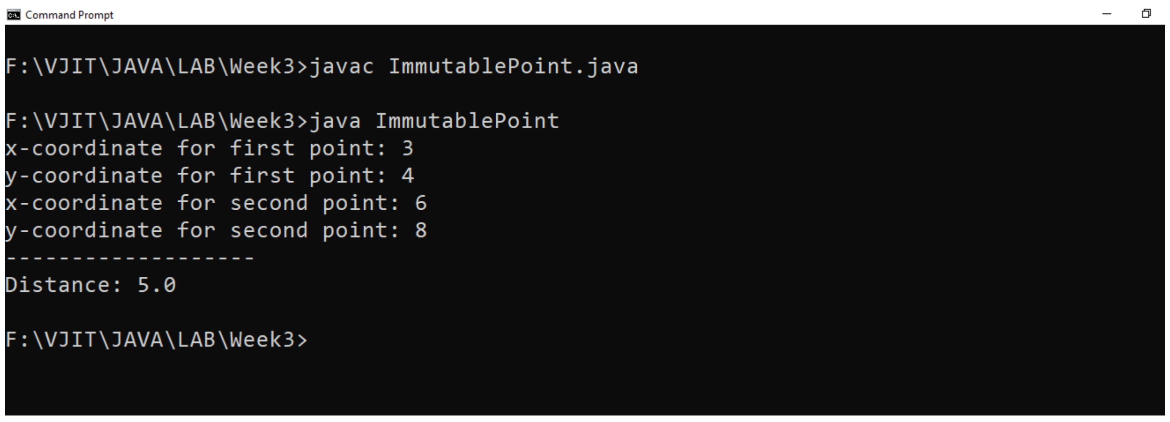
9. Write a Java program that creates **a final class** called ImmutablePoint to represent a point in a 2D plane. The class should have two final instance variables x and y to store the coordinates of the point.

Implement the following:

1. A constructor ImmutablePoint(double x, double y) that initializes the x and y coordinates.
2. Getter methods getX() and getY() to retrieve the values of x and y, respectively.
3. A method calculateDistance(ImmutablePoint other) that takes another ImmutablePoint object as input and calculates the Euclidean distance between the current point and the given point using the formula sqrt((x2 - x1)^2 + (y2 - y1)^2).

```
import java.util.Scanner;

public final class ImmutablePoint

{
    // Final instance variables to represent the coordinates of the point
    private final double x;
    private final double y;
    // Constructor to initialize the coordinates
   public ImmutablePoint(double x, double y)
   {
       this.x = x;
       this.y = y;
    }

    // Getter methods to retrieve the coordinates
   public double getX()
   {
       return x;
    }
    public double getY()
    {
       return y;
    }

   // Method to calculate the distance between two points
    public double calculateDistance(ImmutablePoint other)
   {
       double dx = other.getX() - this.x;
       double dy = other.getY() - this.y;
       return Math.sqrt(dx * dx + dy * dy); // Euclidean distance formula
    }
    public static void main(String[] args)
    {
      Scanner scanner = new Scanner(System.in);
      // Input coordinates for the first point
      System.out.print("x-coordinate for first point: ");
      double x1 = scanner.nextDouble();
      System.out.print("y-coordinate for first point: ");
      double y1 = scanner.nextDouble();
      ImmutablePoint point1 = new ImmutablePoint(x1, y1);
      // Input coordinates for the second point
      System.out.print("x-coordinate for second point: ");
      double x2 = scanner.nextDouble();
      System.out.print("y-coordinate for second point: ");
      double y2 = scanner.nextDouble();
      ImmutablePoint point2 = new ImmutablePoint(x2, y2);
      // Close the scanner
      scanner.close();
      // Calculate and display the distance between the two points
      double distance = point1.calculateDistance(point2);
      System.out.println("-------------------");
     System.out.println("Distance: " + distance);
   }
}
```

Output:



```
F:\VJIT\JAVA\LAB\Week3>javac ImmutablePoint.java

F:\VJIT\JAVA\LAB\Week3>java ImmutablePoint
x-coordinate for first point: 3
y-coordinate for first point: 4
x-coordinate for second point: 6
y-coordinate for second point: 8
-------------------
Distance: 5.0

F:\VJIT\JAVA\LAB\Week3>
```

10.Demonstrate a usage of Package

Write a Java program that generates and prints the first 'n' prime numbers using a single package named primepackage. Please follow the instructions below:

- Create a package named **primepackage**.
- Inside the primepackage package, create a class named **PrimeGenerator**.
- Inside the PrimeGenerator class, implement a static method **generatePrimes** that takes an integer 'n' as input and returns an array of the first 'n' prime numbers.

PrimeGenerator.java

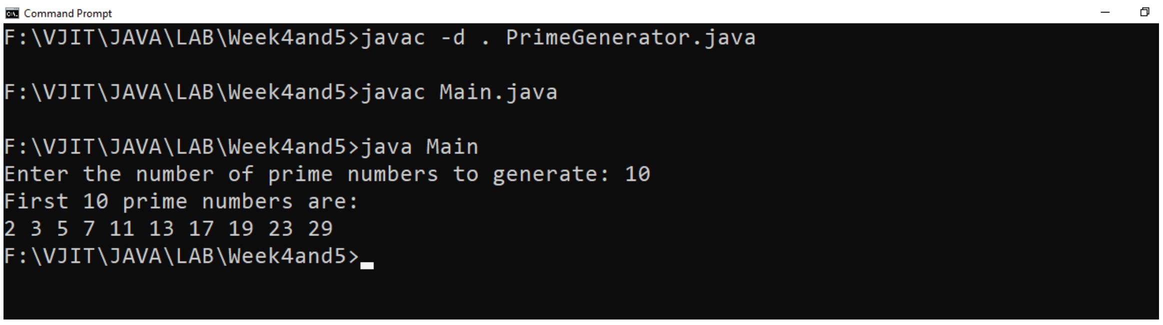
```
package primepackage;
public class PrimeGenerator
{
   // Static method to generate the first 'n' prime numbers
   public static int[] generatePrimes(int n)
   {
      // An array to store the first 'n' prime numbers
      int[] primes = new int[n];
      int count = 0; // To count the number of primes found
      int number = 2; // Starting point (first prime number)

      while (count < n)
     {
         if (isPrime(number))
          {
            primes[count] = number;
            count++;
         }
         number++;
      }
      return primes;
   }
```

```
    // Helper method to check if a number is prime
     private static boolean isPrime(int num)
    {
        if (num <= 1)
            {
           return false; // Numbers less than or equal to 1 are not prime
        }
        for (int i = 2; i <= Math.sqrt(num); i++)
        {
           if (num % i == 0)
           {
              return false; // Divisible by i, hence not prime
           }
        }
        return true; // Prime number
     }
  }
```

Main.java

```
import primepackage.PrimeGenerator;
import java.util.*;
public class Main
{
   public static void main(String[] args)
    {
      Scanner scanner = new Scanner(System.in);
      // Taking input from the user for the number of primes to generate
      System.out.print("Enter the number of prime numbers to generate: ");
      int n = scanner.nextInt();
      // Generate the first 'n' primes using the generatePrimes method
      int[] primes = PrimeGenerator.generatePrimes(n);
      // Printing the generated prime numbers
      System.out.println("First " + n + " prime numbers are:");
      for (int prime : primes)
    {
         System.out.print(prime + " ");
      }

      scanner.close();
   }
}
```

Output:



```
F:\VJIT\JAVA\LAB\Week4and5>javac -d . PrimeGenerator.java

F:\VJIT\JAVA\LAB\Week4and5>javac Main.java

F:\VJIT\JAVA\LAB\Week4and5>java Main
Enter the number of prime numbers to generate: 10
First 10 prime numbers are:
2 3 5 7 11 13 17 19 23 29
F:\VJIT\JAVA\LAB\Week4and5>
```

11.Multiple Catch blocks using command line arguments

Write a Java Program to illustrate Multiple Catch blocks using command line argument. Refer to the sample test case provided to write the code.

Input Format:

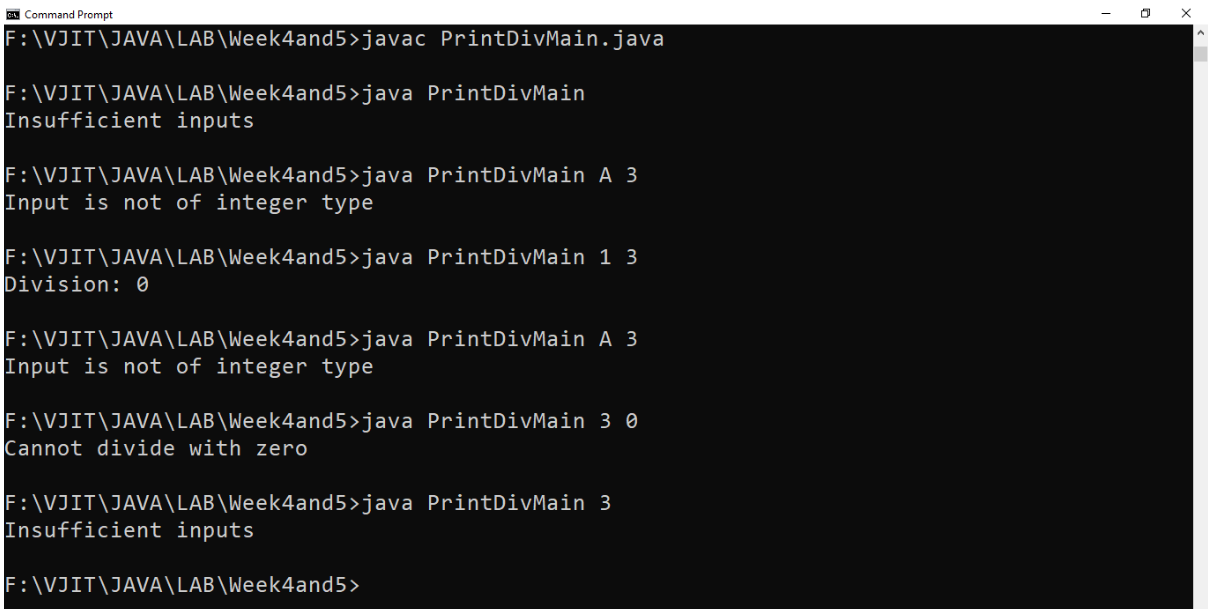
The program takes two command-line arguments:

- The first argument is the numerator (an integer).
- The second argument is the denominator (an integer).

```
public class PrintDivMain
{
                  public static void main(String[] args)
                  {
                     try
                     {
                     //two command-line arguments converted into integer values
                     // Check if the command-line arguments are valid
                             if (args.length < 2)
                             {
                                     System.out.println("Insufficient inputs");
                                     return; // Exit if there are not enough arguments
                             }
                       // Convert the command-line arguments to integers
                             int numerator = Integer.parseInt(args[0]);
                             int denominator = Integer.parseInt(args[1]);
                             try
                             {
                             //perform division operation and catch exception
                                      if (denominator == 0)
                                      {
                                        // Throw an exception if dividing by zero
                                          throw new ArithmeticException();
                                    }
                              // Perform the division and print the result
                                int result = numerator / denominator;
                               System.out.println("Division: " + result);

                             }
```

```
                    catch(ArithmeticException e)
                    {
                      // Handle division by zero
                       System.out.println("Cannot divide with zero");
                     }
            }
            catch(NumberFormatException e)
            {
               // Handle invalid input (non-integer type)
               System.out.println("Input is not of integer type");
            }
            catch(Exception e)
            {
               // Catch any other unexpected exceptions
               System.out.println("An unexpected error occurred");
            }
        }
    }
```

Output:



```
F:\VJIT\JAVA\LAB\Week4and5>javac PrintDivMain.java

F:\VJIT\JAVA\LAB\Week4and5>java PrintDivMain
Insufficient inputs

F:\VJIT\JAVA\LAB\Week4and5>java PrintDivMain A 3
Input is not of integer type

F:\VJIT\JAVA\LAB\Week4and5>java PrintDivMain 1 3
Division: 0

F:\VJIT\JAVA\LAB\Week4and5>java PrintDivMain A 3
Input is not of integer type

F:\VJIT\JAVA\LAB\Week4and5>java PrintDivMain 3 0
Cannot divide with zero

F:\VJIT\JAVA\LAB\Week4and5>java PrintDivMain 3
Insufficient inputs

F:\VJIT\JAVA\LAB\Week4and5>
```

12. User Defined Exceptions

Write a Java program that checks if a person is eligible to vote based on their age.

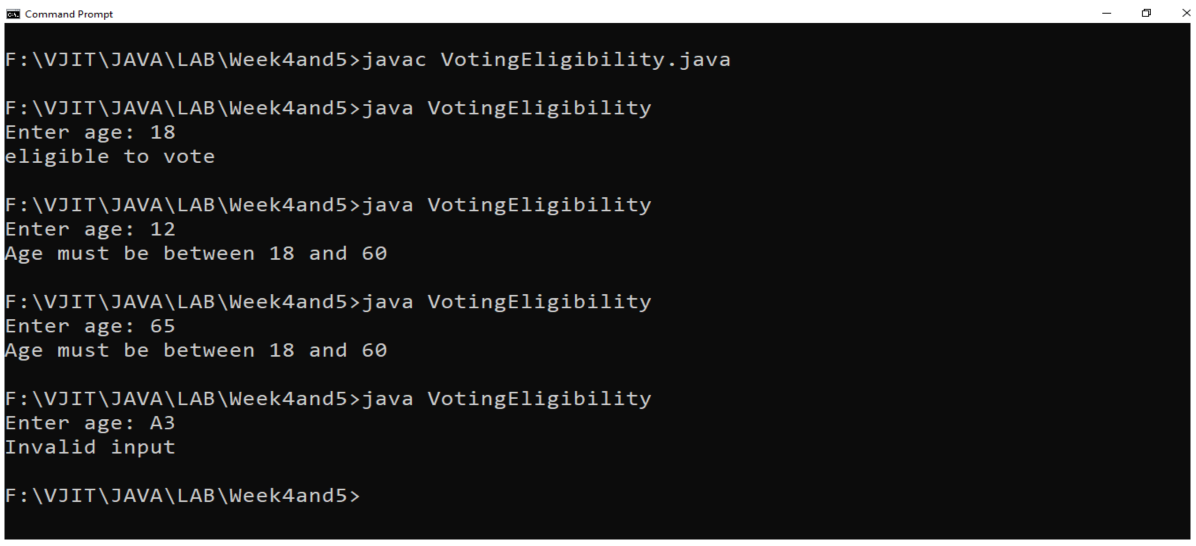
- If the age is below 18 or above 60, throw a custom exception with the message: "Age must be between 18 and 60"
- If the age is 18 or above, but below 60, print "eligible to vote"
- Handle any invalid input gracefully (e.g., non-integer inputs).

You need to create a custom exception class named **VotingEligibilityException** to handle the age validation.

```
import java.util.Scanner;

class VotingEligibilityException extends Exception
{
    // Constructor to pass the error message to the base Exception class
    public VotingEligibilityException(String message)
     {
       super(message);
    }
}

public class VotingEligibility
{
    public static void main(String[] args)
     {
       Scanner scanner = new Scanner(System.in);
       try
      {
         // Prompt user to enter age
          System.out.print("Enter age: ");
          int age = scanner.nextInt(); // Read user input

          // Check if the age is in the valid voting range
          if (age < 18 || age > 60)
         {
           throw new VotingEligibilityException("Age must be between 18 and 60");
          }
        else
        {
             System.out.println("eligible to vote");
          }
       }
```

```
            catch (VotingEligibilityException e)
            {
                // Handle custom exception (if age is outside valid range)
                System.out.println(e.getMessage());
             }
             catch (Exception e)
             {
               // Handle invalid input (e.g., non-integer input)
                System.out.println("Invalid input");
             }
            finally
            {
                // Close the scanner
                scanner.close();
             }
        }
    }
```

Output:



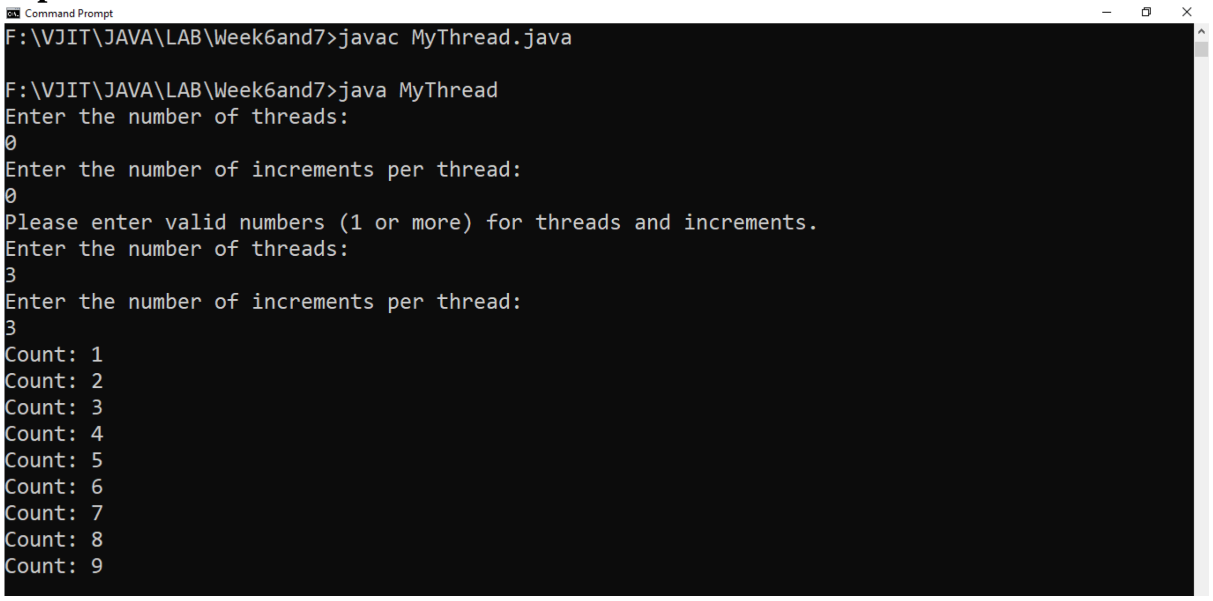
```
F:\VJIT\JAVA\LAB\Week4and5>javac VotingEligibility.java

F:\VJIT\JAVA\LAB\Week4and5>java VotingEligibility
Enter age: 18
eligible to vote

F:\VJIT\JAVA\LAB\Week4and5>java VotingEligibility
Enter age: 12
Age must be between 18 and 60

F:\VJIT\JAVA\LAB\Week4and5>java VotingEligibility
Enter age: 65
Age must be between 18 and 60

F:\VJIT\JAVA\LAB\Week4and5>java VotingEligibility
Enter age: A3
Invalid input

F:\VJIT\JAVA\LAB\Week4and5>
```

13. Write a program to illustrate **Multithreading** and **Multitasking**.

```
import java.util.Scanner;
class MyThread extends Thread
{
    private SharedResource resource;
    private int numIncrements;

   // Constructor to initialize shared resource and number of increments
   public MyThread(SharedResource resource, int numIncrements)
    {
      this.resource = resource;
      this.numIncrements = numIncrements;
   }

   @Override
   public void run()
    {
      // Perform the specified number of increments
      for (int i = 0; i < numIncrements; i++)
     {
         resource.increment();
      }
}
    public static void main(String[] args)
    {
      Scanner scanner = new Scanner(System.in);

      int numThreads = 0;
      int numIncrements = 0;
      boolean validInput = false;

      while (!validInput)
      {
         try
        {
            System.out.println("Enter the number of threads:");
            numThreads = Integer.parseInt(scanner.nextLine());
            System.out.println("Enter the number of increments per thread:");
```

```
                numIncrements = Integer.parseInt(scanner.nextLine());
                if (numThreads >= 1 && numIncrements >= 1)
                {
                    validInput = true;
                }
                else
                {
System.out.println("Please enter valid numbers (1 or more) for threads and increments.");
                }
            }
            catch (NumberFormatException e)
            {
                System.out.println("Please enter valid numbers.");
            }
        }
        SharedResource resource = new SharedResource();
        MyThread[] threads = new MyThread[numThreads];
        for (int i = 0; i < numThreads; i++)
        {
            threads[i] = new MyThread(resource, numIncrements);
            threads[i].start();
        }

        scanner.close();
    }
}

class SharedResource
{
    private int count = 0;
    public void increment()
    {
        synchronized (this)
        {
            count++;
            System.out.println("Count: " + count);
        }
    }
}
```

Output:

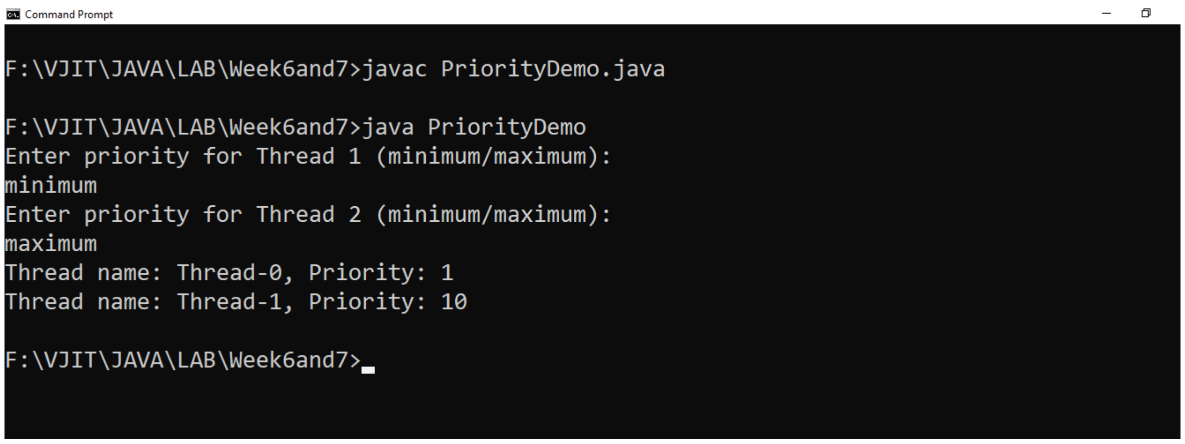


```
F:\VJIT\JAVA\LAB\Week6and7>javac MyThread.java

F:\VJIT\JAVA\LAB\Week6and7>java MyThread
Enter the number of threads:
0
Enter the number of increments per thread:
0
Please enter valid numbers (1 or more) for threads and increments.
Enter the number of threads:
3
Enter the number of increments per thread:
3
Count: 1
Count: 2
Count: 3
Count: 4
Count: 5
Count: 6
Count: 7
Count: 8
Count: 9
```

14. Write a program to illustrate thread priorities.

```
import java.util.Scanner;

class PriorityDemo extends Thread
{
  @Override
    public void run()
   {
        // Print thread name and its priority
                System.out.println("Thread name: " +
         Thread.currentThread().getName() + ", Priority: " +
         Thread.currentThread().getPriority());
    }

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

        PriorityDemo thread1 = new PriorityDemo();
        PriorityDemo thread2 = new PriorityDemo();

   System.out.println("Enter priority for Thread 1 (minimum/maximum):");
        String priorityInput1 = scanner.nextLine().toLowerCase();
   System.out.println("Enter priority for Thread 2 (minimum/maximum):");
        String priorityInput2 = scanner.nextLine().toLowerCase();

        int priority1 = getPriorityValue(priorityInput1);
        int priority2 = getPriorityValue(priorityInput2);

        thread1.setPriority(priority1);
        thread2.setPriority(priority2);

        thread1.start();
        thread2.start();

        scanner.close();
    }
    private static int **getPriorityValue**(String input)
   {
        if (input.equals("minimum"))
        {
           return Thread.MIN_PRIORITY;
        }
        else if (input.equals("maximum"))
        {
           return Thread.MAX_PRIORITY;
        }
         else
         {
           System.out.println("Invalid input. Setting priority to default.");
           return Thread.NORM_PRIORITY;
        }
    }
}
```

Output:



```
F:\VJIT\JAVA\LAB\Week6and7>javac PriorityDemo.java

F:\VJIT\JAVA\LAB\Week6and7>java PriorityDemo
Enter priority for Thread 1 (minimum/maximum):
minimum
Enter priority for Thread 2 (minimum/maximum):
maximum
Thread name: Thread-0, Priority: 1
Thread name: Thread-1, Priority: 10

F:\VJIT\JAVA\LAB\Week6and7>
```

15. 1 Write a program to illustrate Synchronization

Write a Java program using synchronized threads, which demonstrates producer consumer concept.

The **Producer-Consumer** problem is a classic synchronization problem in computer science. It involves **two types** of threads:

- **Producer**: This thread generates data (e.g., items) and places it in a shared resource (like a buffer).
- **Consumer**: This thread takes data (items) from the shared resource for processing or consumption.

The problem revolves around coordinating these threads such that:

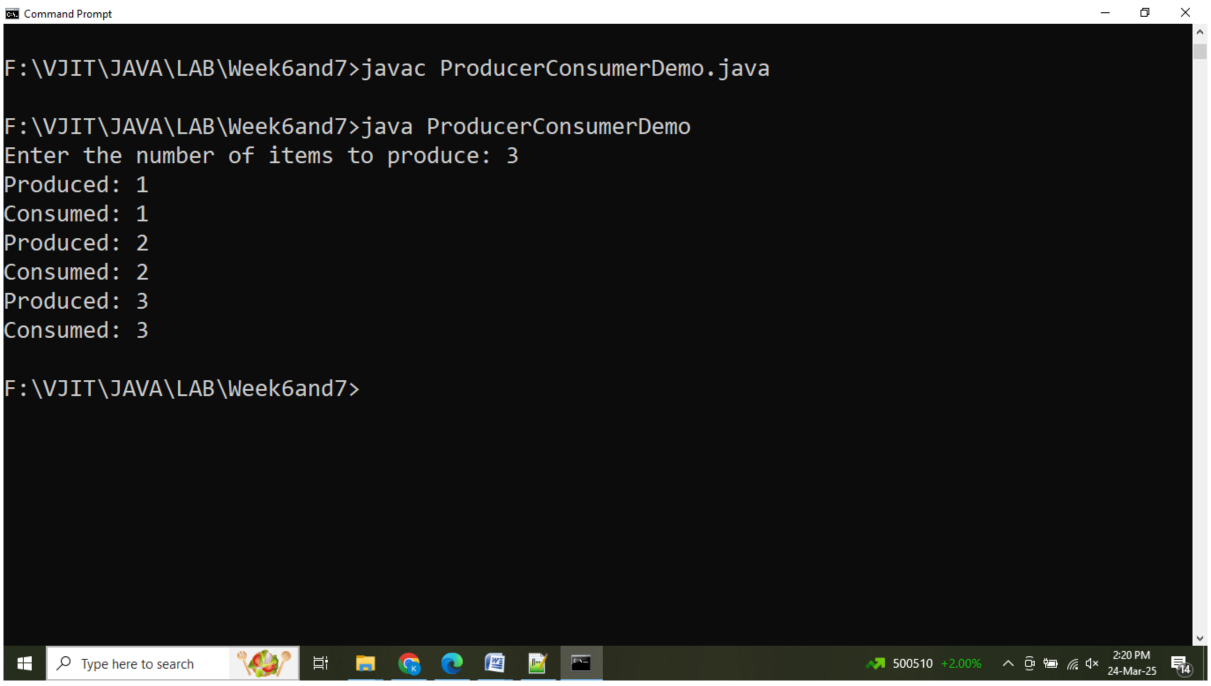
- The producer does not produce when the buffer is full.
- The consumer does not consume when the buffer is empty.
- Both threads operate safely without corrupting the shared resource.

To achieve synchronization in Java, we can use the synchronized keyword along with wait and notify methods to control the interaction between the threads.

```
import java.util.LinkedList;
import java.util.Scanner;

class SharedBuffer
{
    private LinkedList<Integer> buffer = new LinkedList<>();
    private int capacity;
    // Constructor to set the buffer capacity
    public SharedBuffer(int capacity)
   {
        this.capacity = capacity;
    }
    // Produce an item and add it to the buffer
  public synchronized void produce(int item) throws InterruptedException
  {
        while (buffer.size() == capacity)
       {
            wait(); // Wait if the buffer is full
        }
        buffer.add(item);
        System.out.println("Produced: " + item);
        notify(); // Notify the consumer that an item is available
    }
    // Consume an item from the buffer
    public synchronized void consume() throws InterruptedException
    {
        while (buffer.isEmpty())
        {
            wait(); // Wait if the buffer is empty
        }
        int item = buffer.removeFirst();
        System.out.println("Consumed: " + item);
        notify(); // Notify the producer that space is available
    }
}
class Producer extends Thread
{
    private SharedBuffer buffer;
    private int itemsToProduce;
    public Producer(SharedBuffer buffer, int itemsToProduce)
    {
        this.buffer = buffer;
        this.itemsToProduce = itemsToProduce;
    }

    @Override
    public void run()
    {
        try
        {
            for (int i = 1; i <= itemsToProduce; i++)
            {
                buffer.produce(i);
                Thread.sleep(100); // Simulating some delay in producing
            }
        }
        catch (InterruptedException e)
        {
            Thread.currentThread().interrupt();
        }
    }
}
```

```
class Consumer extends Thread
{
   private SharedBuffer buffer;

   public Consumer(SharedBuffer buffer)
  {
      this.buffer = buffer;
   }

   @Override
   public void run()
  {
      try
         {
         // Continue consuming until the producer has finished producing
         while (true)
          {
            buffer.consume();
            Thread.sleep(100); // Simulating some delay in consuming
         }
      }
     catch (InterruptedException e)
     {
         Thread.currentThread().interrupt();
      }
   }
}

public class ProducerConsumerDemo
{
 public static void main(String[] args)
 {
      Scanner scanner = new Scanner(System.in);
      // Get user input for the number of items to produce
      System.out.print("Enter the number of items to produce: ");
      int itemsToProduce = scanner.nextInt();
      // Create a shared buffer
      SharedBuffer buffer = new SharedBuffer(itemsToProduce);
      // Create producer and consumer threads
      Producer producer = new Producer(buffer, itemsToProduce);
      Consumer consumer = new Consumer(buffer);
      // Start the threads
      producer.start();
      consumer.start();
// Wait for producer to finish producing before terminating the consumer
        try
        {
            producer.join();  // Wait for producer to finish
        }
            catch (InterruptedException e)
            {
            Thread.currentThread().interrupt();
        }
       // Interrupt the consumer to gracefully stop the thread after production
        consumer.interrupt();
        // Close the scanner
        scanner.close();
    }
}
```

Output:



```
F:\VJIT\JAVA\LAB\Week6and7>javac ProducerConsumerDemo.java

F:\VJIT\JAVA\LAB\Week6and7>java ProducerConsumerDemo
Enter the number of items to produce: 3
Produced: 1
Consumed: 1
Produced: 2
Consumed: 2
Produced: 3
Consumed: 3

F:\VJIT\JAVA\LAB\Week6and7>
```

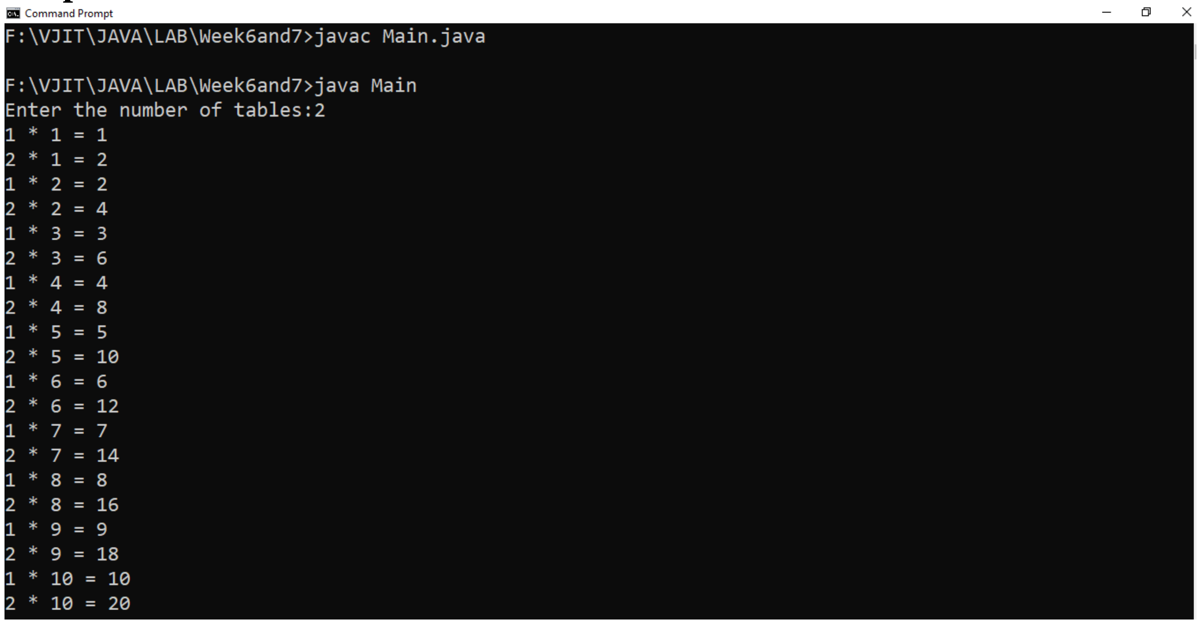
15.2 Printing tables using **synchronization** and threads

Create a Java program that utilizes multi-threading to generate multiplication tables.

```
import java.util.Scanner;

class TablePrinter implements Runnable
{
    private int tableNumber;

    public TablePrinter(int tableNumber)
   {
        this.tableNumber = tableNumber;
    }

    // Implement the run method to generate the multiplication table
    @Override
    public void run()
   {
        try
         {
            // Generate multiplication table for the assigned tableNumber
            for (int i = 1; i <= 10; i++)
           {
             System.out.println(tableNumber + " * " + i + " = " + (tableNumber * i));
               // Adding a small delay (100ms) for better visualization
               Thread.sleep(100);
            }
        }
            catch (InterruptedException e)
            {
            System.err.println("Thread was interrupted: " + e.getMessage());
        }
    }

}
```

```
public class Main
{
   public static void main(String[] args)
  {
       Scanner scanner = new Scanner(System.in);
       System.out.print("Enter the number of tables:");
       int numTables = scanner.nextInt();
       Thread[] threads = new Thread[numTables];

            // Create and start threads for each table
       for (int i = 1; i <= numTables; i++)
       {
          // Create a TablePrinter for each table (1 to numTables)
          TablePrinter tablePrinter = new TablePrinter(i);
          // Create a new thread for each TablePrinter instance
          threads[i - 1] = new Thread(tablePrinter);
          threads[i - 1].start(); // Start the thread
       }

       // Wait for all threads to finish using join()
       try
      {
          for (Thread thread : threads)
         {
           // Ensure the main thread waits for each worker thread to complete
           thread.join();
          }
       }
           catch (InterruptedException e)
           {
          System.err.println("Main thread was interrupted: " + e.getMessage());
       }

       // Close the scanner
       scanner.close();

   }
}
```

Output:

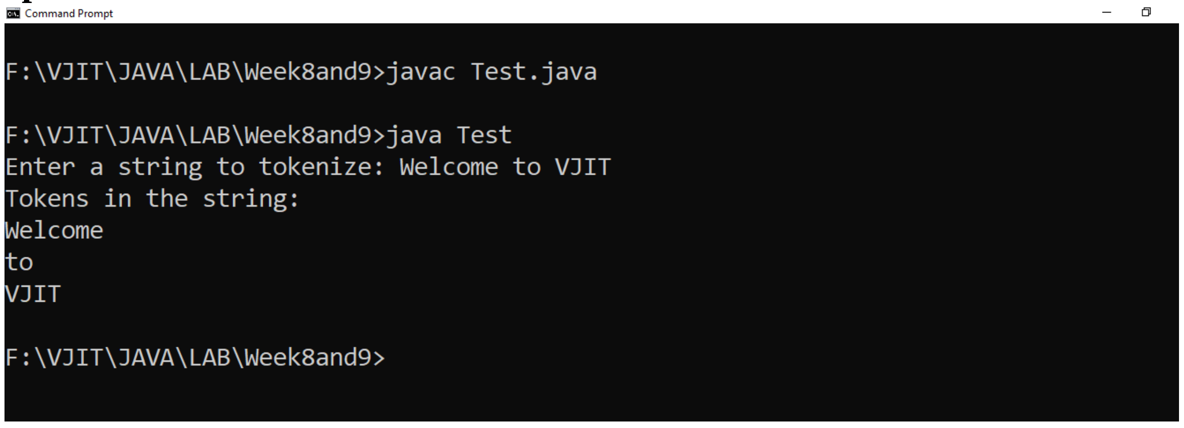


```
F:\VJIT\JAVA\LAB\Week6and7>javac Main.java

F:\VJIT\JAVA\LAB\Week6and7>java Main
Enter the number of tables:2
1 * 1 = 1
2 * 1 = 2
1 * 2 = 2
2 * 2 = 4
1 * 3 = 3
2 * 3 = 6
1 * 4 = 4
2 * 4 = 8
1 * 5 = 5
2 * 5 = 10
1 * 6 = 6
2 * 6 = 12
1 * 7 = 7
2 * 7 = 14
1 * 8 = 8
2 * 8 = 16
1 * 9 = 9
2 * 9 = 18
1 * 10 = 10
2 * 10 = 20
```

16. Write a program to implement **StringTokenizer**.

```
import java.util.StringTokenizer;
import java.util.Scanner;
public class Test
{
 public static void main(String[] args)
 {
       // Create Scanner object to read input from the user
       Scanner scanner = new Scanner(System.in);

       // Prompt the user to enter a string
       System.out.print("Enter a string to tokenize: ");
        // Read the entire line
       String input = scanner.nextLine();
       // Create StringTokenizer object to split the string by space (" ")
       StringTokenizer tokenizer = new StringTokenizer(input);
       // Loop through the tokens and print each one
       System.out.println("Tokens in the string:");
       while (tokenizer.hasMoreTokens())
       {
          // Get the next token and print it
           System.out.println(tokenizer.nextToken());
       }
       scanner.close();
    }
}
```

Output:

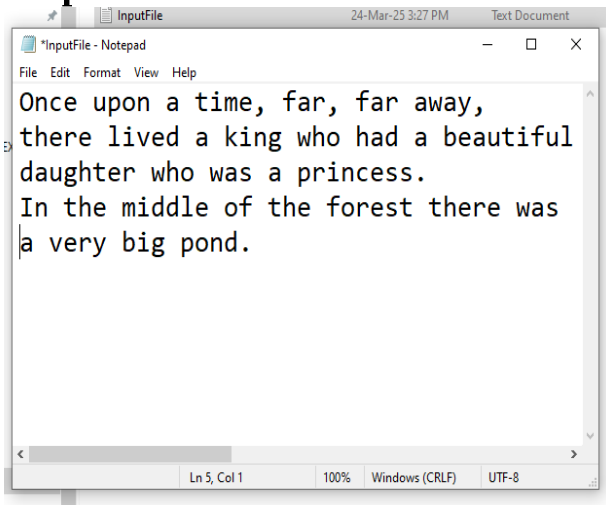


```
F:\VJIT\JAVA\LAB\Week8and9>javac Test.java

F:\VJIT\JAVA\LAB\Week8and9>java Test
Enter a string to tokenize: Welcome to VJIT
Tokens in the string:
Welcome
to
VJIT

F:\VJIT\JAVA\LAB\Week8and9>
```

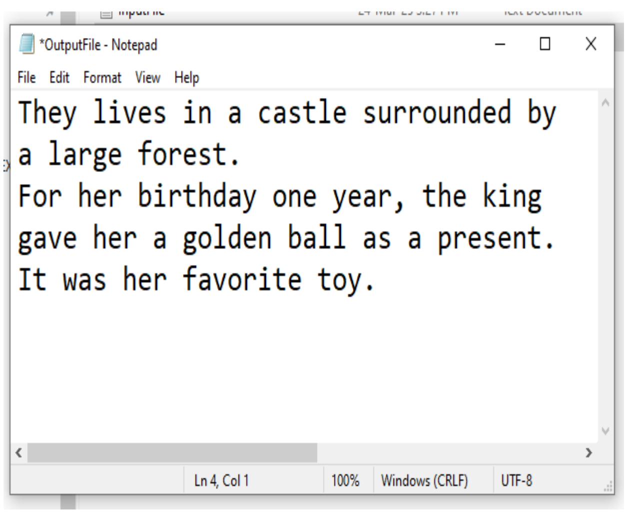
17. Write a program to read one line at a time, and write it to another file.

```
import java.io.*;
import java.util.*;
public class FileCopy
{
 public static void copyData(File source, File destination) throws IOException
 {
      // Create input and output streams
try (BufferedReader reader = new BufferedReader(new FileReader(source));
      BufferedWriter writer = new BufferedWriter(new FileWriter(destination)))
 {
          String line;
          // Read each line from the source file and write it to the destination file
          while ((line = reader.readLine()) != null)
          {
             writer.write(line);
             writer.newLine(); // Add a newline after each line
          }
           System.out.println("Successfully copied contents are:");
      }
    }
 public static void main(String[] args) throws Exception
 {
         Scanner sc = new Scanner(System.in);
          System.out.print("Name of source file: ");
          String file1 = sc.nextLine();
          File s = new File(file1);
          System.out.print("Name of destination file: ");
          String file2 = sc.nextLine();
          File d = new File(file2);
          sc.close();
          copyData(s, d);
          Scanner outFile = new Scanner(new File(file2));
          while (outFile.hasNextLine())
          {
            System.out.println(outFile.nextLine());
          }
          sc.close();
 }
}
```

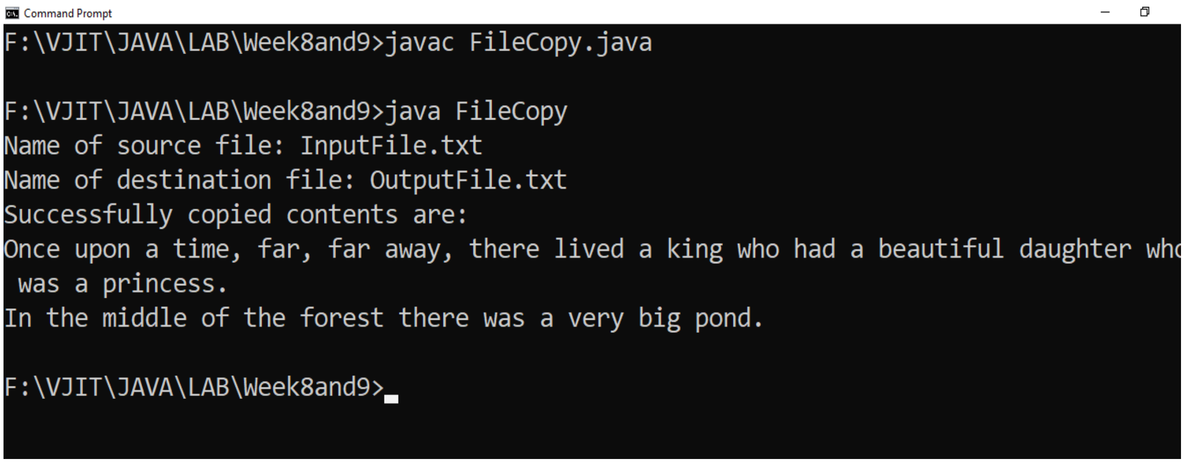
Output:



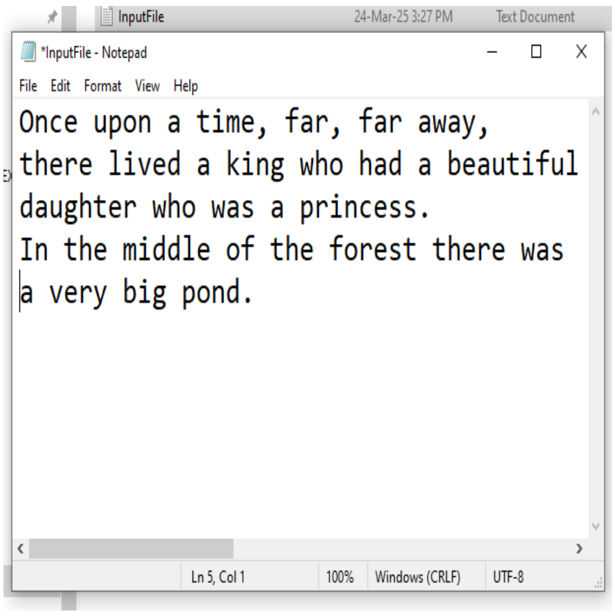
```
Once upon a time, far, far away,
there lived a king who had a beautiful
daughter who was a princess.
In the middle of the forest there was
a very big pond.
```



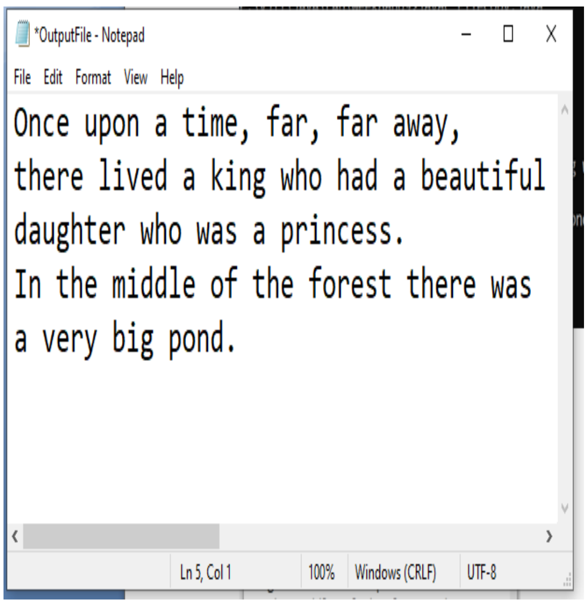
```
They lives in a castle surrounded by
a large forest.
For her birthday one year, the king
gave her a golden ball as a present.
It was her favorite toy.
```



```
F:\VJIT\JAVA\LAB\Week8and9>javac FileCopy.java

F:\VJIT\JAVA\LAB\Week8and9>java FileCopy
Name of source file: InputFile.txt
Name of destination file: OutputFile.txt
Successfully copied contents are:
Once upon a time, far, far away, there lived a king who had a beautiful daughter who
 was a princess.
In the middle of the forest there was a very big pond.

F:\VJIT\JAVA\LAB\Week8and9>_
```



```
Once upon a time, far, far away,
there lived a king who had a beautiful
daughter who was a princess.
In the middle of the forest there was
a very big pond.
```

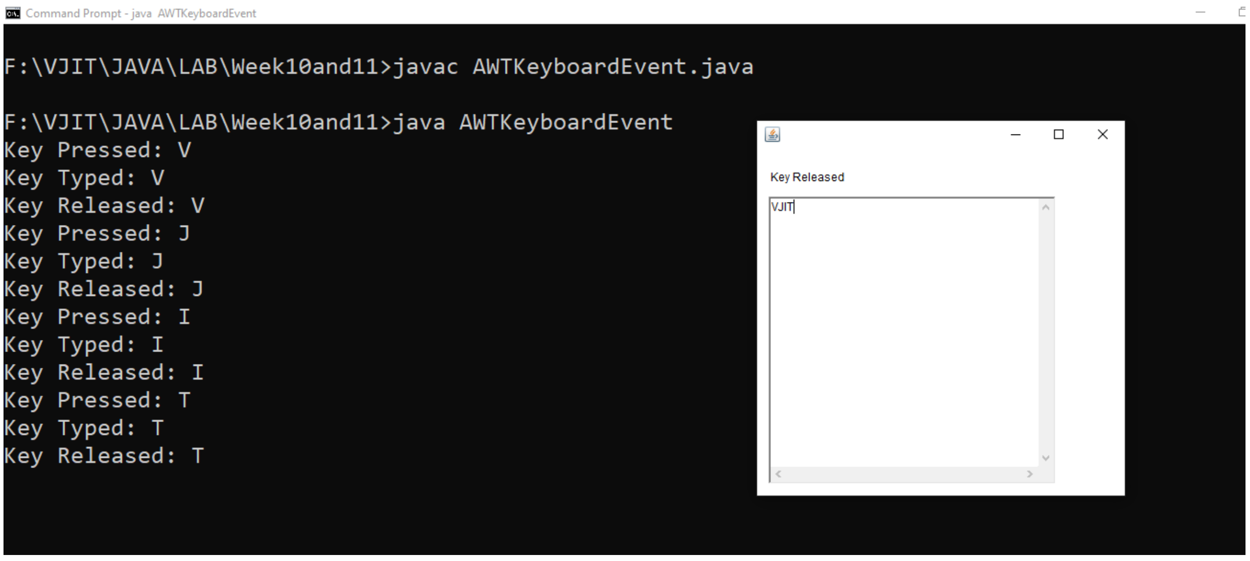


```
Once upon a time, far, far away,
there lived a king who had a beautiful
daughter who was a princess.
In the middle of the forest there was
a very big pond.
```

18.1 Write a program to illustrate **Event Handling** for **Keyboard** Events

```
// importing awt libraries
import java.awt.*;
import java.awt.event.*;

public class AWTKeyboardEvent extends Frame implements KeyListener
{
      Label l;
      AWTKeyboardEvent()
      {
         // creating the label
         l = new Label();
       // setting the location of the label in frame
             l.setBounds (20, 50, 100, 20);
       // creating the text area
            TextArea area = new TextArea();
       // setting the location of text area
             area.setBounds (20, 80, 300, 300);
       // adding the KeyListener to the text area
             area.addKeyListener(this);
       // adding the label and text area to the frame
             add(l);
            add(area);
       // setting the size, layout and visibility of frame
            setSize (400, 400);
            setLayout (null);
            setVisible (true);
       addWindowListener(new WindowAdapter()
      {
          public void windowClosing(WindowEvent we)
         {
             System.exit(0);  // Close the application
          }
       });
    }
// overriding the keyPressed() method of KeyListener interface
    public void keyPressed (KeyEvent e)
   {
        l.setText ("Key Pressed");
        int keyCode = e.getKeyCode();
        System.out.println("Key Pressed: " + KeyEvent.getKeyText(keyCode));

    }

// overriding the keyReleased() method of KeyListener interface
    public void keyReleased (KeyEvent e)
      {
        l.setText ("Key Released");
        int keyCode = e.getKeyCode();
        System.out.println("Key Released: " + KeyEvent.getKeyText(keyCode));

    }

// overriding the keyTyped() method of KeyListener interface
    public void keyTyped (KeyEvent e)
   {
        l.setText ("Key Typed");
          char keyChar = e.getKeyChar();
        System.out.println("Key Typed: " + keyChar);
    }

  // main method
    public static void main(String[] args)
    {
        new AWTKeyboardEvent();
    }
}
```

Output:

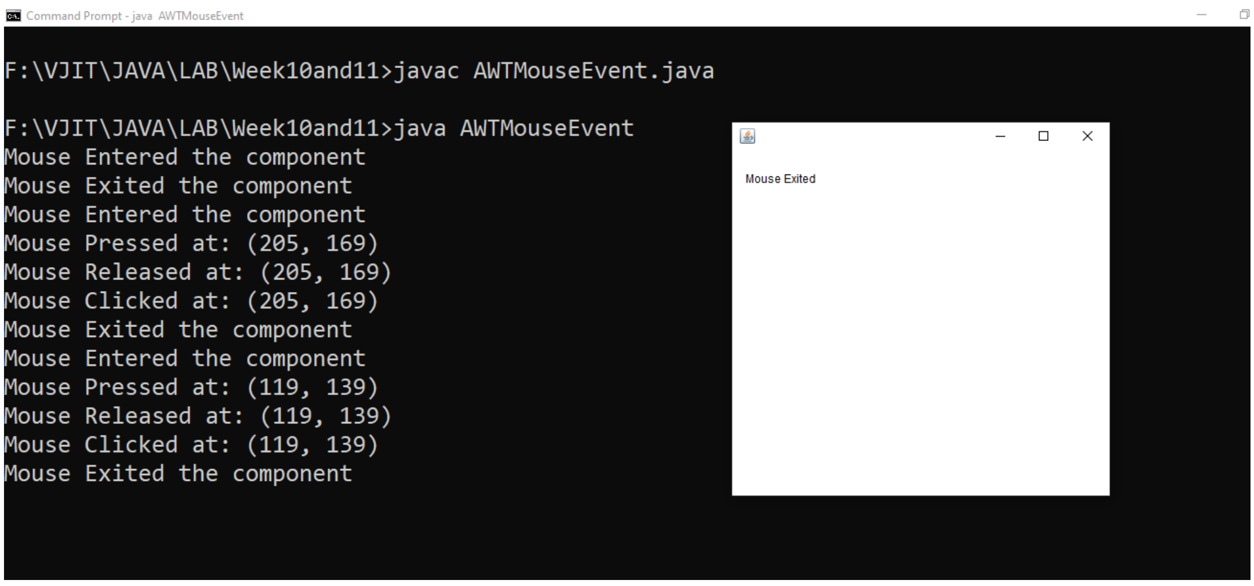


```
F:\VJIT\JAVA\LAB\Week10and11>javac AWTKeyboardEvent.java

F:\VJIT\JAVA\LAB\Week10and11>java AWTKeyboardEvent
Key Pressed: V
Key Typed: V
Key Released: V
Key Pressed: J
Key Typed: J
Key Released: J
Key Pressed: I
Key Typed: I
Key Released: I
Key Pressed: T
Key Typed: T
Key Released: T
```

18.2 Write a program to illustrate **Event Handling** for **Mouse** Events

```
import java.awt.*;
import java.awt.event.*;
public class AWTMouseEvent extends Frame implements MouseListener
{
   Label l;
   AWTMouseEvent()
    {
      addMouseListener(this);

      l=new Label();
      l.setBounds(20,50,100,20);
      add(l);
      setSize(300,300);
      setLayout(null);
      setVisible(true);
      addWindowListener(new WindowAdapter()
      {
         public void windowClosing(WindowEvent we)
        {
            System.exit(0);  // Close the application
         }
      });
    }
    public void mouseEntered(MouseEvent e)
   {
      l.setText("Mouse Entered");
      System.out.println("Mouse Entered the component");
    }
    public void mouseExited(MouseEvent e)
   {
      l.setText("Mouse Exited");
          System.out.println("Mouse Exited the component");
    }
   public void mousePressed(MouseEvent e)
   {
      l.setText("Mouse Pressed");
      System.out.println("Mouse Pressed at: (" + e.getX() + ", " + e.getY() + ")");
   }
   public void mouseReleased(MouseEvent e)
   {
      l.setText("Mouse Released");
      System.out.println("Mouse Released at: (" + e.getX() + ", " + e.getY() + ")");
   }
   public void mouseClicked(MouseEvent e)
   {
      l.setText("Mouse Clicked");
      System.out.println("Mouse Clicked at: (" + e.getX() + ", " + e.getY() + ")");
   }

   public static void main(String[] args)
   {
      new AWTMouseEvent();
   }
}
```

Output:



```
F:\VJIT\JAVA\LAB\Week10and11>javac AWTMouseEvent.java

F:\VJIT\JAVA\LAB\Week10and11>java AWTMouseEvent
Mouse Entered the component
Mouse Exited the component
Mouse Entered the component
Mouse Pressed at: (205, 169)
Mouse Released at: (205, 169)
Mouse Clicked at: (205, 169)
Mouse Exited the component
Mouse Entered the component
Mouse Pressed at: (119, 139)
Mouse Released at: (119, 139)
Mouse Clicked at: (119, 139)
Mouse Exited the component
```

19. Write a program to illustrate **applet life cycle** and parameter passing.

```
import java.applet.Applet;
import java.awt.*;
import java.util.Date;

/*
<applet code="AppletLifeCycle" width=300 height=200>
   <param name="greeting" value="Welcome to Java Applet">
   <param name="author" value="VJIT">
</applet>
*/

public class AppletLifeCycle extends Applet
{

   // Instance variables to store parameter values
   private String greetingMessage;
   private String author;

   // Called when the applet is first loaded into memory
   @Override
   public void init()
   {
      // Retrieve parameters from the applet tag
      greetingMessage = getParameter("greeting");  // Get greeting message
      author = getParameter("author");  // Get author name

      if (greetingMessage == null)
      {
         greetingMessage = "Hello, this is a default greeting!";
      }
      if (author == null)
      {
         author = "Unknown Author";
      }

      System.out.println("Applet Initialized");
   }
```

```
    // Called when the applet is started or brought to the foreground
    @Override
    public void start()
     {
        System.out.println("Applet Started");
    }

    // Called to display the applet on the screen
    @Override
    public void paint(Graphics g)
     {
        // Display greeting message and author name on the applet
        g.drawString(greetingMessage, 20, 50);
        g.drawString("Author: " + author, 20, 70);

        // Display current date and time
        g.drawString("Current Date: " + new Date(), 20, 90);
    }

    // Called when the applet is stopped (i.e., the user navigates away)
    @Override
    public void stop()
     {
        System.out.println("Applet Stopped");
    }

    // Called when the applet is destroyed
    @Override
    public void destroy()
     {
        System.out.println("Applet Destroyed");
    }
}
```

Output:

20. Write a program to develop a **Calculator** application using **AWT**

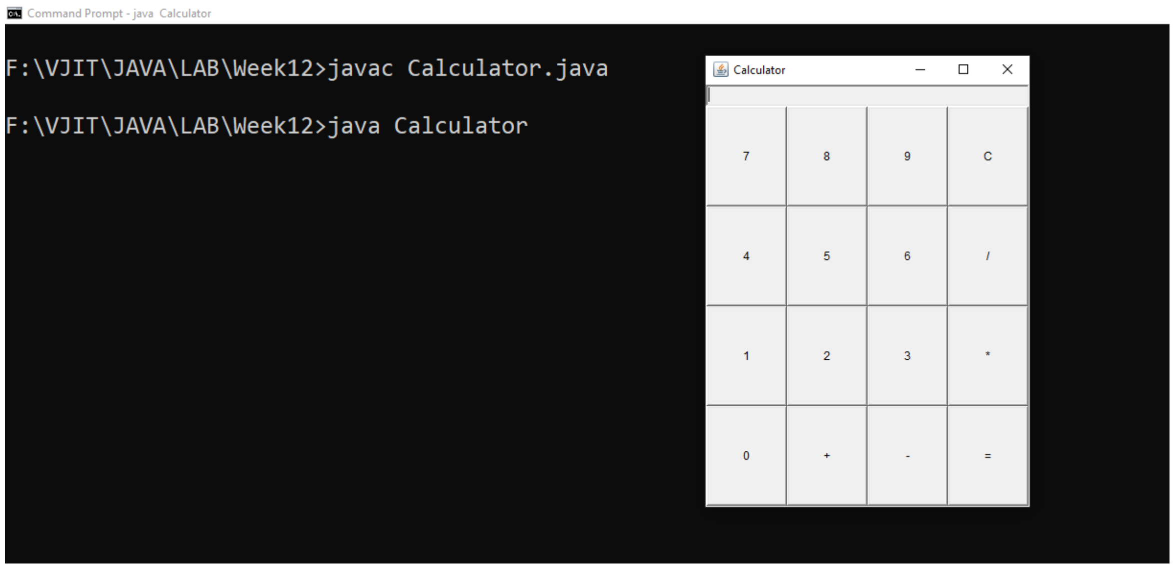
```
import java.awt.*;
import java.awt.event.*;
public class Calculator extends Frame implements ActionListener
{
    // Declare all components
    TextField display;
    Button add, subtract, multiply, divide, equals, clear;
    double num1, num2, result;
    char operator;
    // Constructor to set up the GUI
    public Calculator()
    {
        // Set title and layout
        setTitle("Calculator");
        setSize(300, 400);
        setLayout(new BorderLayout());

        // Create display (TextField)
        display = new TextField();
        display.setEditable(false);
        add(display, BorderLayout.NORTH);

        // Panel to hold number buttons
        Panel panel = new Panel();
        panel.setLayout(new GridLayout(4, 4)); // 4 rows, 4 columns

// Create Number buttons
    Button b0 = new Button("0");
    Button b1 = new Button("1");
    Button b2 = new Button("2");
    Button b3 = new Button("3");
    Button b4 = new Button("4");
    Button b5 = new Button("5");
    Button b6 = new Button("6");
    Button b7 = new Button("7");
    Button b8 = new Button("8");
    Button b9 = new Button("9");
```

```
// Create operator buttons
        add = new Button("+");
        subtract = new Button("-");
        multiply = new Button("*");
         divide = new Button("/");
        equals = new Button("=");
        clear = new Button("C");

// Add action listeners to the Number buttons
      b0.addActionListener(this);
      b1.addActionListener(this);
      b2.addActionListener(this);
       b3.addActionListener(this);
      b4.addActionListener(this);
      b5.addActionListener(this);
       b6.addActionListener(this);
      b7.addActionListener(this);
      b8.addActionListener(this);
      b9.addActionListener(this);

        // Add action listeners to the operator buttons
        add.addActionListener(this);
        subtract.addActionListener(this);
        multiply.addActionListener(this);
        divide.addActionListener(this);
        equals.addActionListener(this);
        clear.addActionListener(this);

        // Add operator buttons to the panel (in GridLayout)
           panel.add(b7);
           panel.add(b8);
           panel.add(b9);
          panel.add(clear);
          panel.add(b4);
           panel.add(b5);
           panel.add(b6);
           panel.add(divide);
           panel.add(b1);
           panel.add(b2);
           panel.add(b3);
           panel.add(multiply);
           panel.add(b0);
            panel.add(add);
             panel.add(subtract);
             panel.add(equals);

        // Add the panel to the frame
        add(panel, BorderLayout.CENTER);

        // Window closing event
        addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent we)
          {
              System.exit(0);
            }
        });

        // Set the frame visible
        setVisible(true);
     }

     // ActionListener method to handle button clicks
     public void actionPerformed(ActionEvent e)
     {
        String command = e.getActionCommand();
        if ((command.charAt(0) >= '0' && command.charAt(0) <= '9'))
       {
            // If a number is pressed, add it to the display
            display.setText(display.getText() + command);
        }
          else if (command.charAt(0) == 'C')
          {
            // Clear the display when 'C' is pressed
            display.setText("");
        }
```

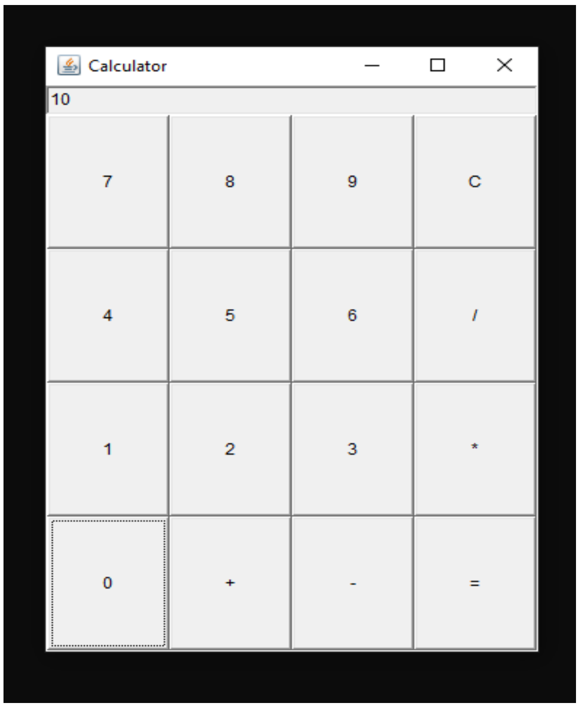
```
        else if (command.charAt(0) == '=')
        {
          // When '=' is pressed, calculate the result
          num2 = Double.parseDouble(display.getText());
          switch (operator)
        {
             case '+':
                          result = num1 + num2;
                          break;
             case '-':
                           result = num1 - num2;
                           break;
             case '*':
                            result = num1 * num2;
                            break;
             case '/':
                             if (num2 != 0)
                             {
                                result = num1 / num2;
                             }
                             else
                             {
                                display.setText("Error");
                                return;
                             }
                             break;
          }
          display.setText(String.valueOf(result));
          num1 = result; // Store result for further calculation
       }
        else
        {
          // When an operator is pressed, store the number and operator
          if (!display.getText().isEmpty())
        {
             num1 = Double.parseDouble(display.getText());
             operator = command.charAt(0);
             display.setText("");
          }
       }
    }
```

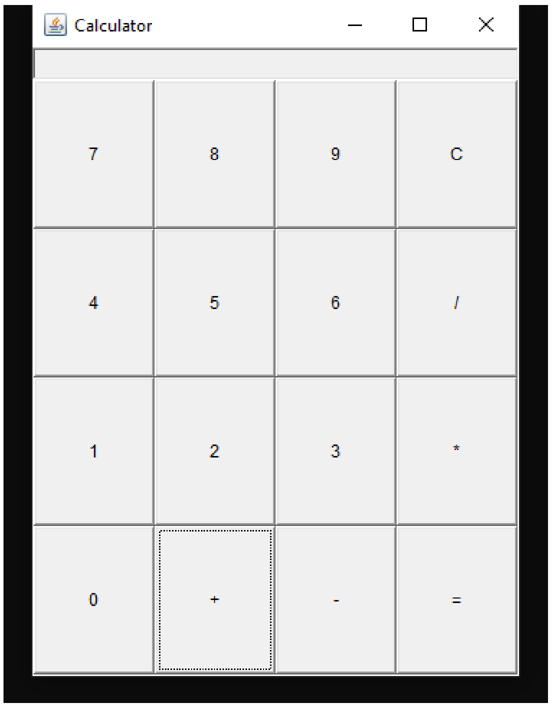
```
        public static void main(String[] args)
       {
           // Create an instance of the Calculator class
           new Calculator();
        }
    }
```

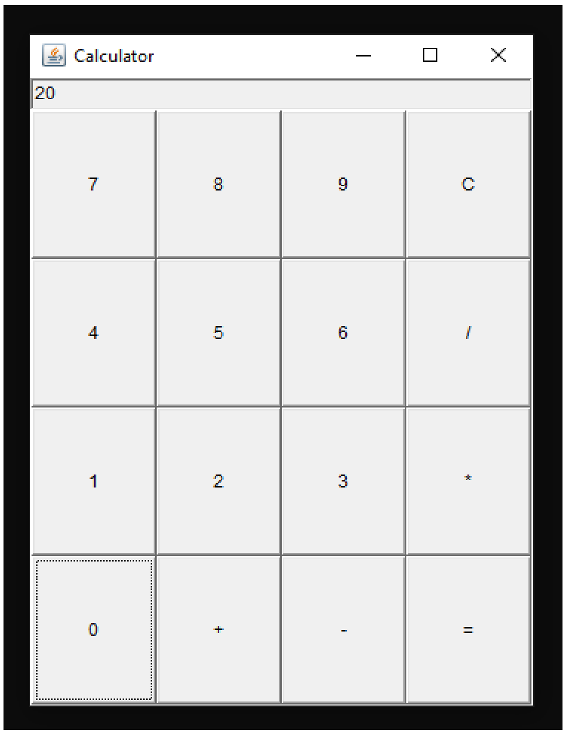
Output:

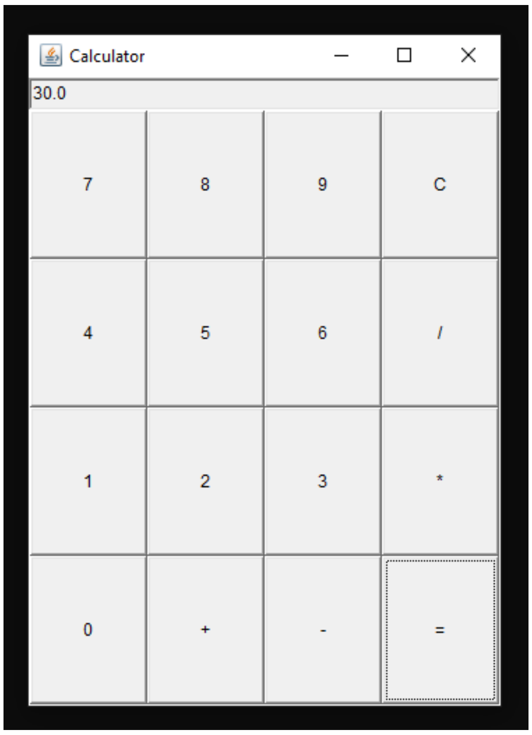


```
Command Prompt - java  Calculator

F:\VJIT\JAVA\LAB\Week12>javac Calculator.java

F:\VJIT\JAVA\LAB\Week12>java Calculator
```





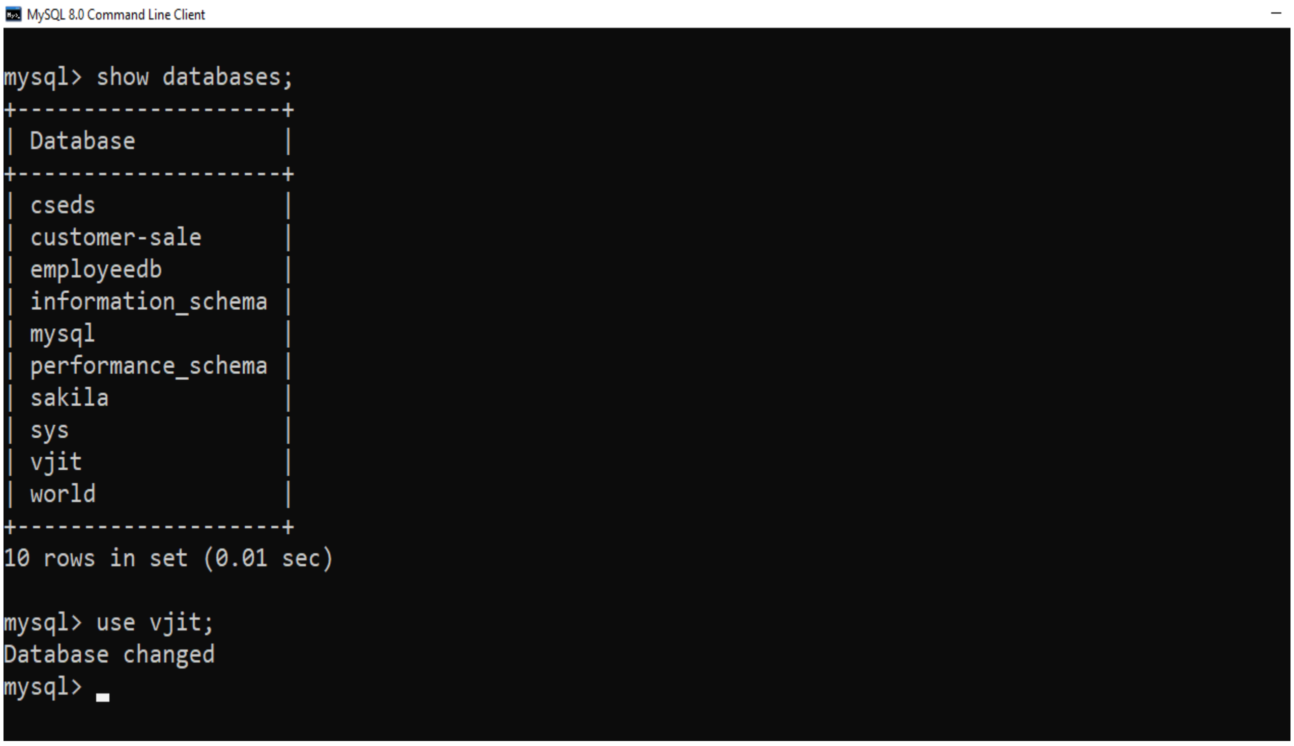

Calculator
20
7
8
9
C
4
5
6
/
1
2
3
*
0
+
-
=


Calculator
30.0
7
8
9
C
4
5
6
/
1
2
3
*
0
+
-
=

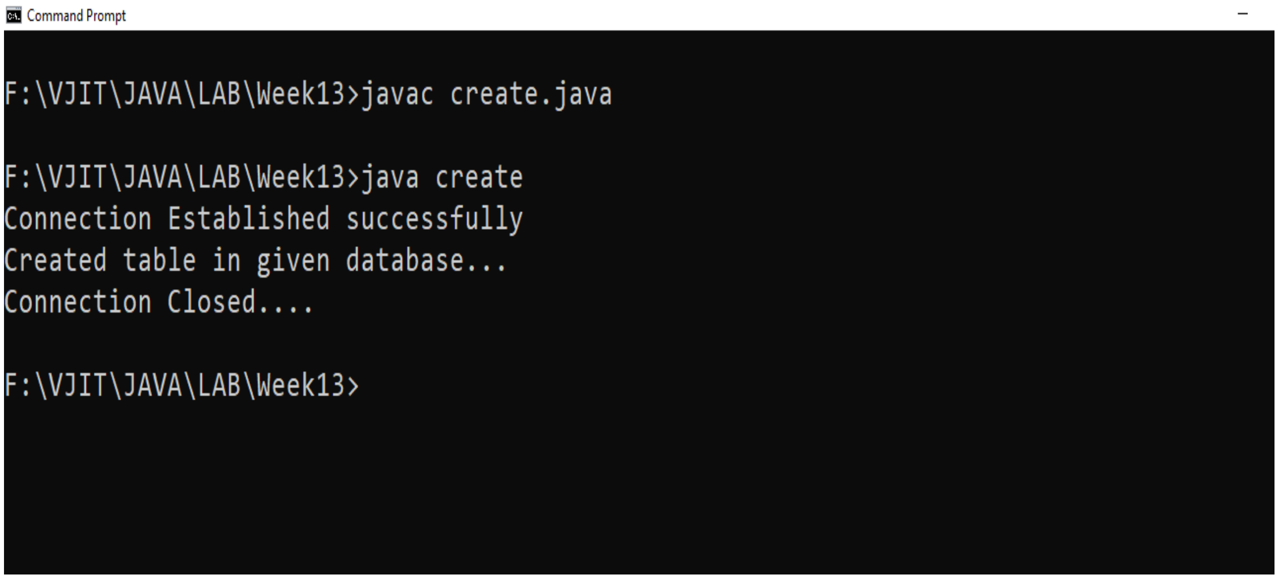
21.1 Java Program to **CREATE** Database Table in MySql Server

```
// This code is for establishing connection with MySQL database and retrieving data
// from database through JDBC
/*
 *1. import --->java.sql
 *2. load and register the driver ---> com.jdbc.
 *3. create connection
 *4. create a statement
 *5. execute the query
 *6. process the results
 *7. close
 */

import java.io.*;
import java.sql.*;

class create
{
 public static void main(String[] args) throws ClassNotFoundException,SQLException
   {
       String url= "jdbc:mysql://localhost:3306/vjit"; // Database details
       String username = "root"; // MySQL credentials
       String password = "vjit";
       Class.forName("com.mysql.cj.jdbc.Driver"); // Driver name
       Connection con = DriverManager.getConnection(url, username, password);
       System.out.println("Connection Established successfully");
       Statement st = con.createStatement();
      String sql = "CREATE TABLE STUDENTS(Roll_No INTEGER NOT
      NULL,First_Name VARCHAR(255),Last_Name VARCHAR(255),Age
      INTEGER,PRIMARY KEY ( Roll_No ))";
       st.execute(sql);
       System.out.println("Created table in given database...");
       st.close(); // close statement
       con.close(); // close connection
       System.out.println("Connection Closed....");
   }
}
```

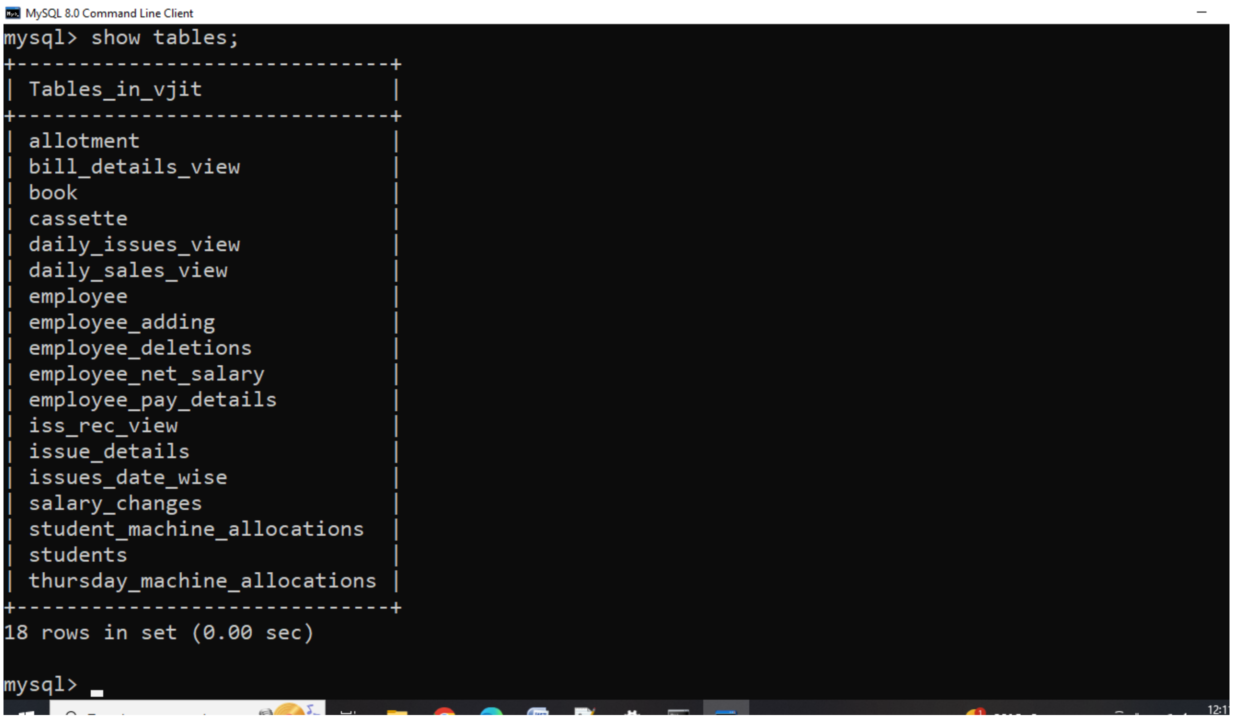
Output:



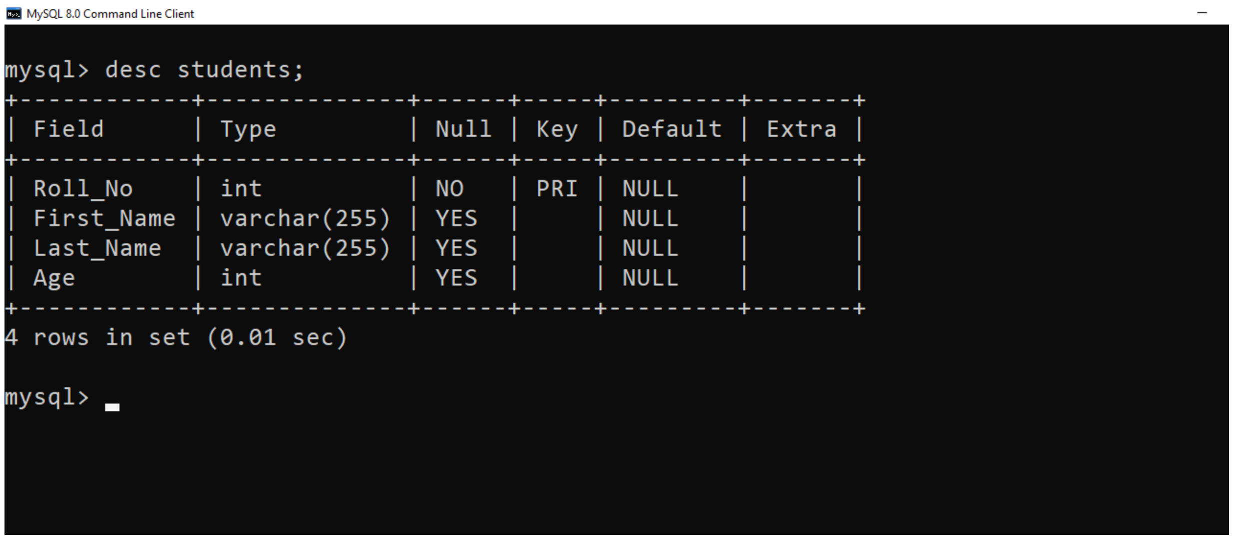
```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| cseds              |
| customer-sale      |
| employeedb         |
| information_schema |
| mysql              |
| performance_schema |
| sakila             |
| sys                |
| vjit               |
| world              |
+--------------------+
10 rows in set (0.01 sec)

mysql> use vjit;
Database changed
mysql>
```



```
F:\VJIT\JAVA\LAB\Week13>javac create.java

F:\VJIT\JAVA\LAB\Week13>java create
Connection Established successfully
Created table in given database...
Connection Closed....

F:\VJIT\JAVA\LAB\Week13>
```



```
mysql> show tables;
+------------------------------+
| Tables_in_vjit               |
+------------------------------+
| allotment                    |
| bill_details_view            |
| book                         |
| cassette                     |
| daily_issues_view            |
| daily_sales_view             |
| employee                     |
| employee_adding              |
| employee_deletions           |
| employee_net_salary          |
| employee_pay_details         |
| iss_rec_view                 |
| issue_details                |
| issues_date_wise             |
| salary_changes               |
| student_machine_allocations  |
| students                     |
| thursday_machine_allocations |
+------------------------------+
18 rows in set (0.00 sec)

mysql> _
```

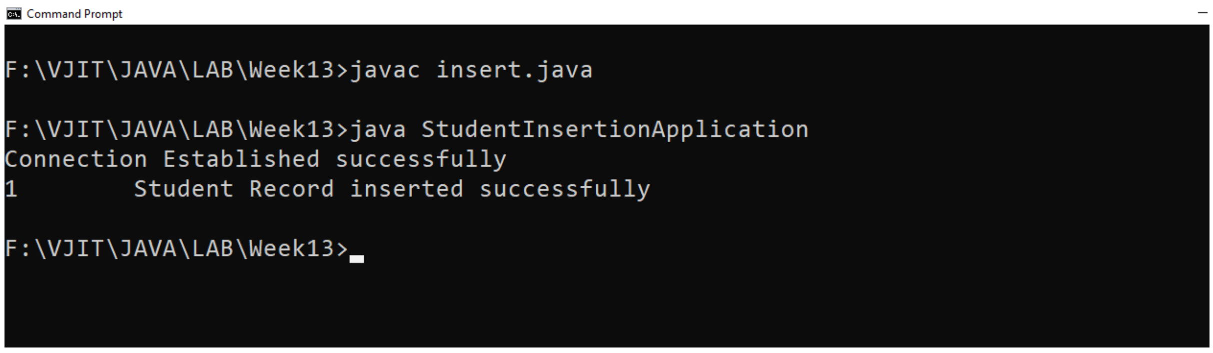


```
mysql> desc students;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| Roll_No    | int          | NO   | PRI | NULL    |       |
| First_Name | varchar(255) | YES  |     | NULL    |       |
| Last_Name  | varchar(255) | YES  |     | NULL    |       |
| Age        | int          | YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
4 rows in set (0.01 sec)

mysql> _
```

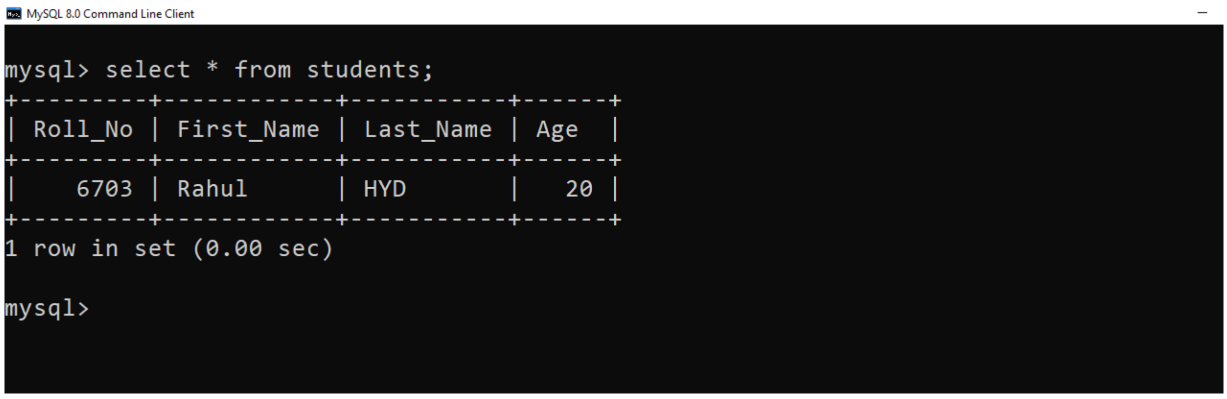
21.2 Java Program to **INSERT** Record into Database Table in MySql Server

```
import java.sql.*;
class StudentInsertionApplication
{
public static void main(String[] args) throws ClassNotFoundException,SQLException
 {
     String url= "jdbc:mysql://localhost:3306/vjit"; // Database details
        String username = "root"; // MySQL credentials
        String password = "vjit";
        Class.forName("com.mysql.cj.jdbc.Driver"); // Driver name
        Connection con = DriverManager.getConnection(url, username, password);
        System.out.println("Connection Established successfully");
                   Statement st = con.createStatement();
        int c = st.executeUpdate("insert into students values(6703,'Rahul','HYD',20)");
        System.out.println(c + "\t Student Record inserted successfully");
        st.close();
        con.close();
    }
}
```

Output:



```
F:\VJIT\JAVA\LAB\Week13>javac insert.java

F:\VJIT\JAVA\LAB\Week13>java StudentInsertionApplication
Connection Established successfully
1        Student Record inserted successfully

F:\VJIT\JAVA\LAB\Week13>_
```

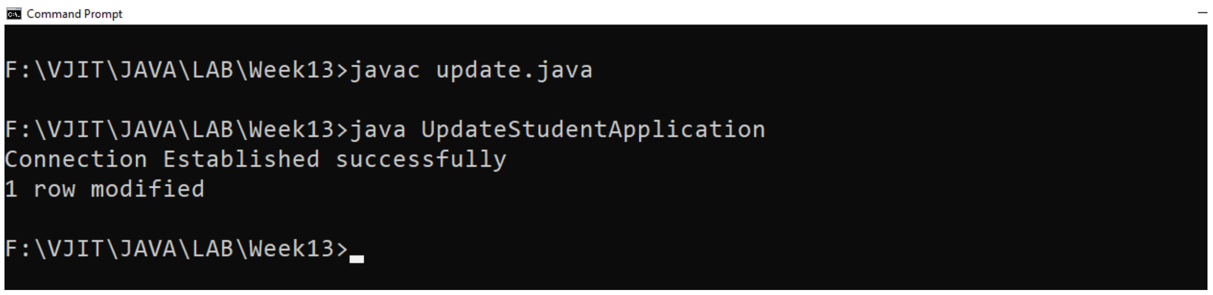


```
mysql> select * from students;
+---------+------------+-----------+------+
| Roll_No | First_Name | Last_Name | Age  |
+---------+------------+-----------+------+
|    6703 | Rahul      | HYD       |   20 |
+---------+------------+-----------+------+
1 row in set (0.00 sec)

mysql>
```

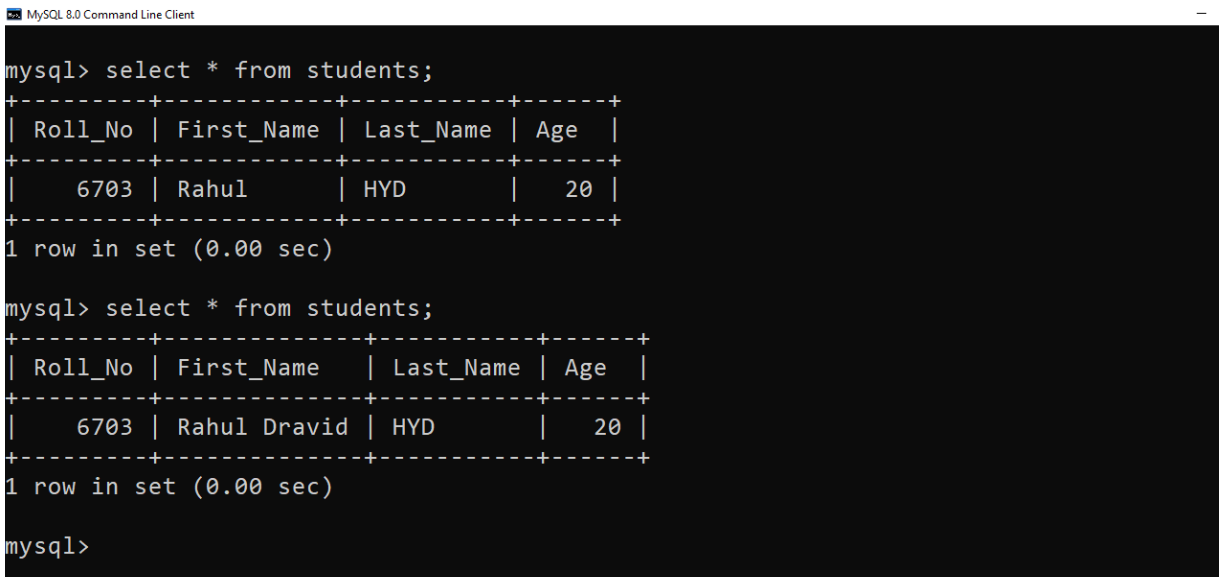
21.3 Java Program to **UPDATE** Record into Database Table in MySql Server

```
 import java.sql.*;
 class UpdateStudentApplication
 {
public static void main (String[]args) throws ClassNotFoundException, SQLException
  {
        String url= "jdbc:mysql://localhost:3306/vjit"; // Database details
        String username = "root"; // MySQL credentials
        String password = "vjit";
        Class.forName("com.mysql.cj.jdbc.Driver"); // Driver name
        Connection con = DriverManager.getConnection(url, username, password);
        System.out.println("Connection Established successfully");
                Statement st = con.createStatement();
  int rows = st.executeUpdate("update students set First_Name = 'Rahul Dravid' where
 Roll_No=6703");
        System.out.println(rows + " row modified");
        st.close();
        con.close();
     }
 }
```

Output:



```
F:\VJIT\JAVA\LAB\Week13>javac update.java

F:\VJIT\JAVA\LAB\Week13>java UpdateStudentApplication
Connection Established successfully
1 row modified

F:\VJIT\JAVA\LAB\Week13>_
```

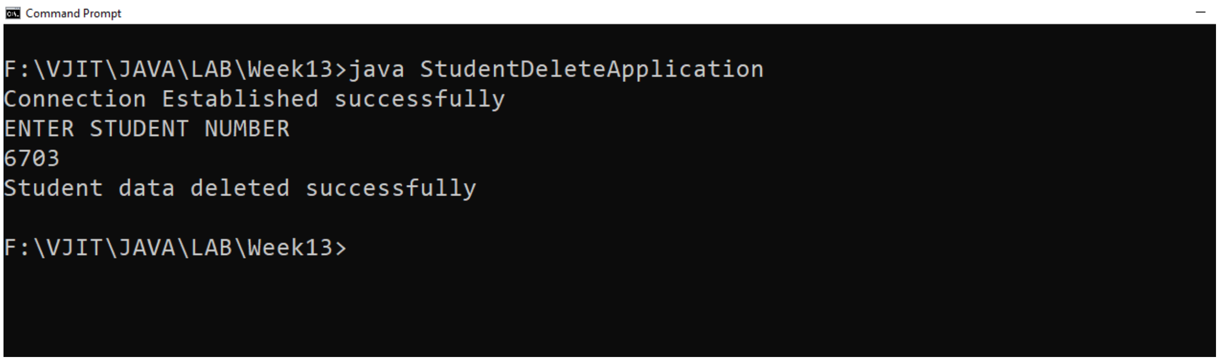


```
mysql> select * from students;
+---------+------------+-----------+------+
| Roll_No | First_Name | Last_Name | Age  |
+---------+------------+-----------+------+
|    6703 | Rahul      | HYD       |   20 |
+---------+------------+-----------+------+
1 row in set (0.00 sec)

mysql> select * from students;
+---------+--------------+-----------+------+
| Roll_No | First_Name   | Last_Name | Age  |
+---------+--------------+-----------+------+
|    6703 | Rahul Dravid | HYD       |   20 |
+---------+--------------+-----------+------+
1 row in set (0.00 sec)

mysql>
```

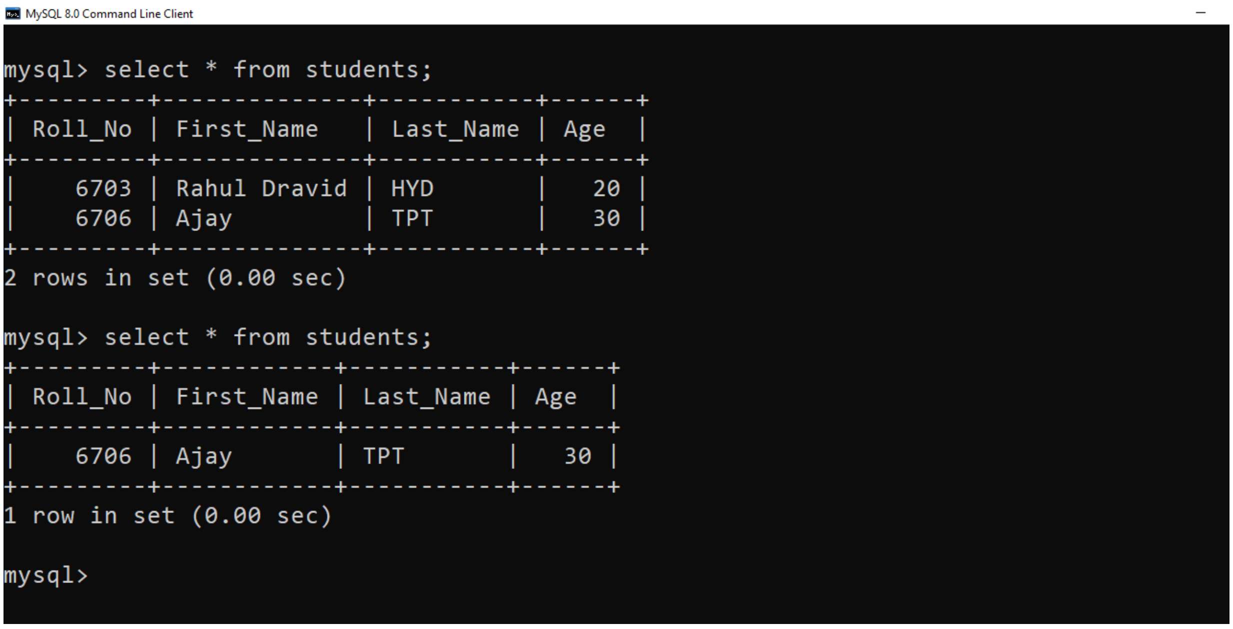
21.4 Java Program to **DELETE** Record from Database Table in MySql Server

```
 import java.sql.*;
 import java.util.*;
 class StudentDeleteApplication
 {
public static void main (String[]args) throws ClassNotFoundException, SQLException
  {
        String url= "jdbc:mysql://localhost:3306/vjit"; // Database details
        String username = "root"; // MySQL credentials
        String password = "vjit";
        Class.forName("com.mysql.cj.jdbc.Driver"); // Driver name
        Connection con = DriverManager.getConnection(url, username, password);
        System.out.println("Connection Established successfully");
                Statement st = con.createStatement();
                Scanner sc = new Scanner(System.in);
        System.out.println("ENTER STUDENT NUMBER");
        int rno = sc.nextInt();

        int c = st.executeUpdate("delete from students where Roll_No =" + rno);
        if (c == 0)
           System.out.println("Student data does not exist");
        else
           System.out.println("Student data deleted successfully");
        st.close();
        con.close();
    }
 }
```

Output:



```
F:\VJIT\JAVA\LAB\Week13>java StudentDeleteApplication
Connection Established successfully
ENTER STUDENT NUMBER
6703
Student data deleted successfully

F:\VJIT\JAVA\LAB\Week13>
```

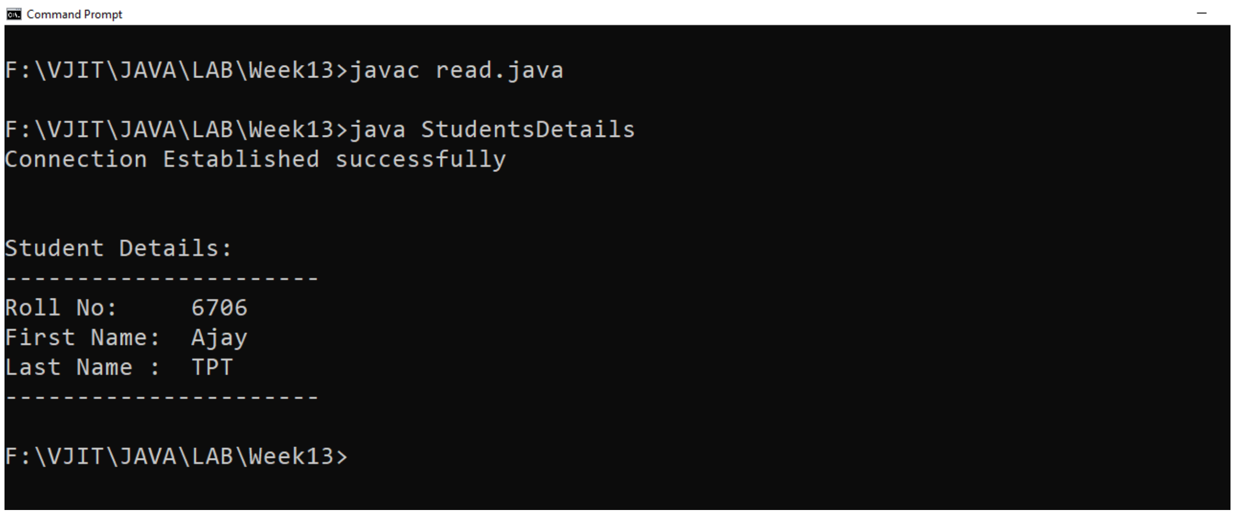


```
mysql> select * from students;
+---------+--------------+-----------+------+
| Roll_No | First_Name   | Last_Name | Age  |
+---------+--------------+-----------+------+
|    6703 | Rahul Dravid | HYD       |   20 |
|    6706 | Ajay         | TPT       |   30 |
+---------+--------------+-----------+------+
2 rows in set (0.00 sec)

mysql> select * from students;
+---------+------------+-----------+------+
| Roll_No | First_Name | Last_Name | Age  |
+---------+------------+-----------+------+
|    6706 | Ajay       | TPT       |   30 |
+---------+------------+-----------+------+
1 row in set (0.00 sec)

mysql>
```

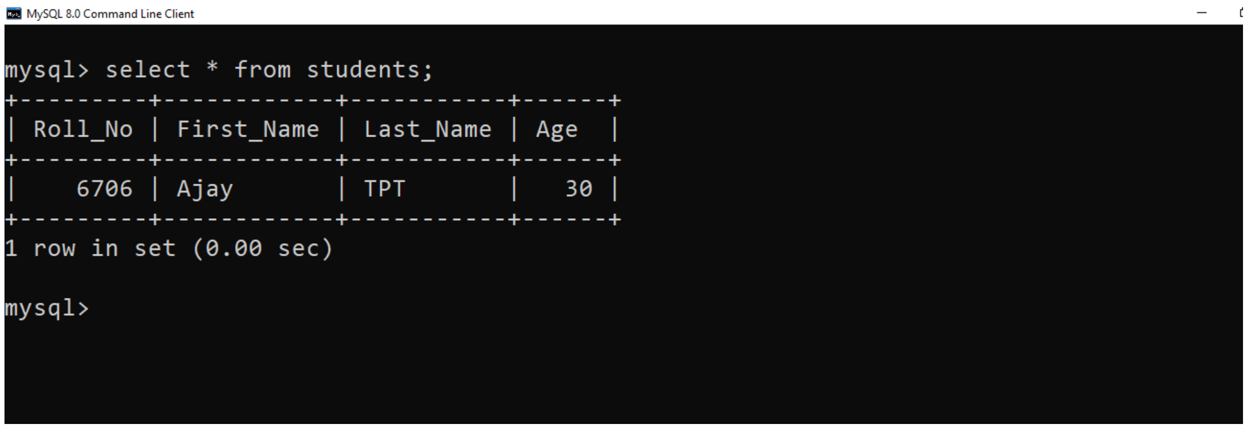
21.5 Java Program to **SELECT** Records from Database Table in MySql Server

```
import java.sql.*;
import java.util.*;
class StudentsDetails
{
public static void main (String[]args) throws ClassNotFoundException, SQLException
 {
        String url= "jdbc:mysql://localhost:3306/vjit"; // Database details
        String username = "root"; // MySQL credentials
        String password = "vjit";
        Class.forName("com.mysql.cj.jdbc.Driver"); // Driver name
        Connection con = DriverManager.getConnection(url, username, password);
        System.out.println("Connection Established successfully");
                Statement st = con.createStatement();
                System.out.println("\n\nStudent Details: ");
                System.out.println("----------------------");
                ResultSet rs = st.executeQuery("select * from students");
        while(rs.next())
        {
             System.out.println("Roll No:     " + rs.getInt(1));
            System.out.println("First Name:  " + rs.getString(2));
            System.out.println("Last Name :  " + rs.getString(3));
            System.out.println("----------------------");
        }
       rs.close();
        st.close();
        con.close();
    }
}
```

Output:



```
F:\VJIT\JAVA\LAB\Week13>javac read.java

F:\VJIT\JAVA\LAB\Week13>java StudentsDetails
Connection Established successfully


Student Details:
----------------------
Roll No:     6706
First Name:  Ajay
Last Name :  TPT
----------------------

F:\VJIT\JAVA\LAB\Week13>
```



```
mysql> select * from students;
+---------+------------+-----------+------+
| Roll_No | First_Name | Last_Name | Age  |
+---------+------------+-----------+------+
|    6706 | Ajay       | TPT       |   30 |
+---------+------------+-----------+------+
1 row in set (0.00 sec)

mysql>
```