Parul University

FACULTY OF ENGINEERING AND TECHNOLOGY
BACHELOR OF TECHNOLOGY

OBJECT ORIENTED PROGRAMMING (303105205)

III SEMESTER

Computer Science & Engineering Department

Laboratory Manual

Session:2024-25

OBJECT ORIENTED PROGRAMMING PRACTICAL BOOK

COMPUTER SCIENCE &ENGINEERING DEPARTMENT

PREFACE

It gives us immense pleasure to present the first edition of the **OBJECT ORIENTED PROGRAMMING** Practical Book for the B.Tech . 3rd **semester** students for **PARUL UNIVERSITY**.

The **OOP** theory and laboratory courses at PARUL UNIVERSITY, WAGHODIA, VADODARA are designed in such a way that students develop the basic understanding of the subject in the theory classes and then try their hands on the experiments to realize the various implementations of problems learnt during the theoretical sessions. The main objective of the **OOP** laboratory course is: Learning **OOP** through Experimentations. All the experiments are designed to illustrate various problems in different areas of **OOP** and also to expose the students to various uses.

The objective of this **OOP** Practical Book is to provide a comprehensive source for all the experiments included in the **OOP** laboratory course. It explains all the aspects related to every experiment such as: basic underlying concept and how to analyze a problem. It also gives sufficient information on how to interpret and discuss the obtained results.

We acknowledge the authors and publishers of all the books which we have consulted while developing this Practical book. Hopefully this **OOP** Practical Book will serve the purpose for which it has been developed.

INSTRUCTIONS TO STUDENTS

1. The main objective of the **OOP** laboratory is: Learning through the Experimentation. All the experiments are designed to illustrate various problems in different areas of **OOP** and also to expose the students to various problems and their uses.
2. Be prompt in arriving to the laboratory and always come well prepared for the practical.
3. Every student should have his/her individual copy of the **OOP** Practical Book.
4. Every student have to prepare the notebooks specifically reserved for the **OOP** practical work: " **OOP** Practical Book"
5. Every student has to necessarily bring his/her **OOP** Practical Book, **OOP** Practical Class Notebook and **OOP** Practical Final Notebook.
6. Finally find the output of the experiments along with the problem and note results in the **OOP** Practical Notebook.
7. The grades for the **OOP** practical course work will be awarded based on our performance in the laboratory, regularity, recording of experiments in the **OOP** Practical Final Notebook, lab quiz, regular viva-voce and end-term examination.

CERTIFICATE

This is to certify that

Mr./Ms...

.... with enrolment no. ....

has successfully completed his/her laboratory experiments in the **OBJECT ORIENTED PROGRAMMING (303105205)** from the department of **COMPUTER SCIENCE & ENGINEERING** during the academic year **2024-2025**.

Date of Submission:.........................   Staff In charge:...........................

Head Of Department:..

TABLE OF CONTENT

Sr. No.	Experiment Title	Page No.		Date of Start	Date of Completion	Sign	Marks (out of 10)
		From	To				
24.	Write a java program that implements a multi-thread application that has three threads. First thread generates random integer every 1 second and if the value is even, second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of cube of the number.						
25.	Write a program to create dynamic array using ArrayList class and the print the contents of the array object.						
26.	Write programs to implement add, search and remove operation on ArrayList object.						

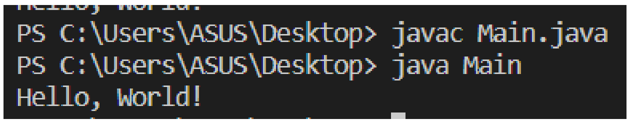
Practical-1

Aim:- Write a program to display Hello World message in console window.

Code:-

```
// This is the declaration of the Main class, which is the entry point of the program.
public class Main {
   // This is the main method, where the program starts execution.
   public static void main(String[] args) {
      // This line prints the message "Hello, World!" to the console.
      System.out.println("Hello, World!");
   }
}
```

Output:-



```
PS C:\Users\ASUS\Desktop> javac Main.java
PS C:\Users\ASUS\Desktop> java Main
Hello, World!
```

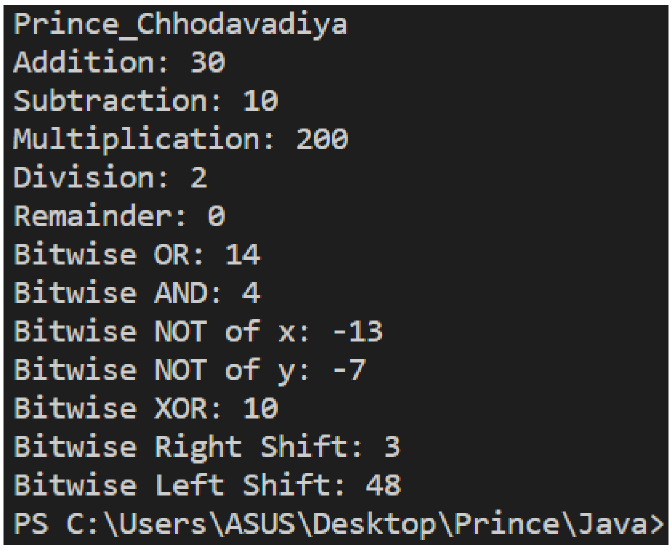
Practical-2

Aim:- Write a program to perform arithmetic and bitwise operations in a single source program without object creation.

Code:-

```
// This is the declaration of the arith_bitwise class.
public class arith_bitwise
{
  // This is the main method, where the program starts execution.
  public static void main(String[] args)
  {
    // This line prints "Prince_Chhodavadiya" to the console.
    System.out.println("Prince_Chhodavadiya");

    // Declare and initialize two integer variables, 'a' and 'b'.
    int a = 20;
    int b = 10;

    // Perform addition, subtraction, multiplication, division, and modulo operations on 'a' and 'b'.
    int A = a + b;
    System.out.println("Addition: " + A);
    int S = a - b;
    System.out.println("Subtraction: " + S);
    int M = a * b;
    System.out.println("Multiplication: " + M);
    int D = a / b;
    System.out.println("Division: " + D);
    int R = a % b;
    System.out.println("Remainder: " + R);

    // Bitwise Operations
    int x = 12; // 1100 in Binary
    int y = 6;  // 0110 in Binary

    // Perform bitwise OR, bitwise AND, bitwise NOT, bitwise XOR,
    // bitwise right shift, and bitwise left shift operations on 'x' and 'y'.
    int Or = x | y;
    System.out.println("Bitwise OR: " + Or);
    int And = x & y;
    System.out.println("Bitwise AND: " + And);
    int NotX = ~x;
    System.out.println("Bitwise NOT of x: " + NotX);
    int NotY = ~y;
    System.out.println("Bitwise NOT of y: " + NotY);
    int Xor = x ^ y;
    System.out.println("Bitwise XOR: " + Xor);
    int rightShift = x >> 2;
    System.out.println("Bitwise Right Shift: " + rightShift);
    int leftShift = x << 2;
    System.out.println("Bitwise Left Shift: " + leftShift);
  }
}
```

Output:-



```
Prince_Chhodavadiya
Addition: 30
Subtraction: 10
Multiplication: 200
Division: 2
Remainder: 0
Bitwise OR: 14
Bitwise AND: 4
Bitwise NOT of x: -13
Bitwise NOT of y: -7
Bitwise XOR: 10
Bitwise Right Shift: 3
Bitwise Left Shift: 48
PS C:\Users\ASUS\Desktop\Prince\Java>
```

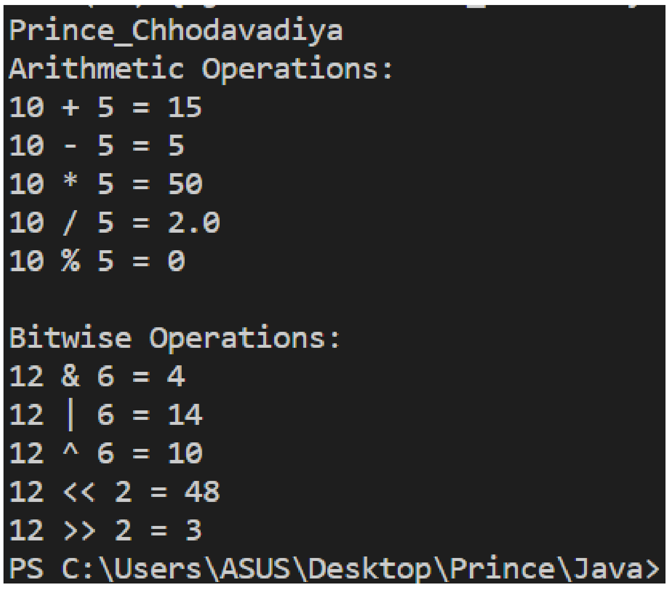
Practical-3

Aim:- Write a program to perform arithmetic and bitwise operations by creating individual methods and classes than create an object to execute the individual methods of each operation.

Code:-

```
// This class contains methods for performing arithmetic operations.
class ArithmeticOperations {
   public int add(int a, int b) {
      return a + b;
   }

   public int subtract(int a, int b) {
      return a - b;
   }

   public int multiply(int a, int b) {
      return a * b;
   }

   public double divide(int a, int b) {
      return (double) a / b;
   }

   public int modulus(int a, int b) {
      return a % b;
   }
}

// This class contains methods for performing bitwise operations.
class BitwiseOperations {
   public int bitwiseAnd(int x, int y) {
      return x & y;
   }

   public int bitwiseOr(int x, int y) {
      return x | y;
   }

   public int bitwiseXor(int x, int y) {
      return x ^ y;
   }

   public int bitwiseLeftShift(int x, int n) {
      return x << n;
   }

   public int bitwiseRightShift(int x, int n) {
      return x >> n;
   }
}
// This is the main class that contains the program's entry point.
public class Arithmetic_Bitwise {
   public static void main(String[] args) {
      // Print a message to the console.
      System.out.println("Prince_Chhodavadiya");

      // Initialize two integer variables, 'a' and 'b'.
      int a = 10;
      int b = 5;

      // Create instances of the ArithmeticOperations and BitwiseOperations classes.
      ArithmeticOperations arithmetic = new ArithmeticOperations();
      BitwiseOperations bitwise = new BitwiseOperations();

      // Perform arithmetic operations and store the results in variables.
      int addition = arithmetic.add(a, b);
      int subtraction = arithmetic.subtract(a, b);
      int multiplication = arithmetic.multiply(a, b);
      double division = arithmetic.divide(a, b);
      int modulus = arithmetic.modulus(a, b);

      // Initialize two integer variables, 'x' and 'y'.
      int x = 12;
      int y = 6;

      // Perform bitwise operations and store the results in variables.
      int bitwiseAnd = bitwise.bitwiseAnd(x, y);
      int bitwiseOr = bitwise.bitwiseOr(x, y);
      int bitwiseXor = bitwise.bitwiseXor(x, y);
      int bitwiseLeftShift = bitwise.bitwiseLeftShift(x, 2);
      int bitwiseRightShift = bitwise.bitwiseRightShift(x, 2);

      // Print the results of arithmetic operations.
      System.out.println("Arithmetic Operations:");
      System.out.println(a + " + " + b + " = " + addition);
      System.out.println(a + " - " + b + " = " + subtraction);
      System.out.println(a + " * " + b + " = " + multiplication);
      System.out.println(a + " / " + b + " = " + division);
      System.out.println(a + " % " + b + " = " + modulus);

      // Print the results of bitwise operations.
      System.out.println("\nBitwise Operations:");
      System.out.println(x + " & " + y + " = " + bitwiseAnd);
      System.out.println(x + " | " + y + " = " + bitwiseOr);
      System.out.println(x + " ^ " + y + " = " + bitwiseXor);
      System.out.println(x + " << 2 = " + bitwiseLeftShift);
      System.out.println(x + " >> 2 = " + bitwiseRightShift);
   }
}
```

Output:-



```
Prince_Chhodavadiya
Arithmetic Operations:
10 + 5 = 15
10 - 5 = 5
10 * 5 = 50
10 / 5 = 2.0
10 % 5 = 0

Bitwise Operations:
12 & 6 = 4
12 | 6 = 14
12 ^ 6 = 10
12 << 2 = 48
12 >> 2 = 3
PS C:\Users\ASUS\Desktop\Prince\Java>
```

Practical-4

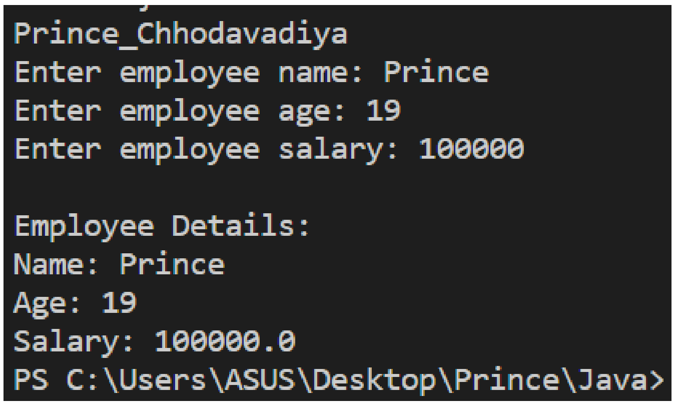
Aim:- Write a java program to display the employee details using Scanner class.

Code:-

```
// Import the Scanner class from the java.util package to read user input.
import java.util.Scanner;

// Create a class named Employee to represent an employee's information.
class Employee {
   // Declare instance variables for name, age, and salary.
   String name;
   int age;
   double salary;

   // Method to input employee details.
   public void inputDetails() {
      // Create a Scanner object to read user input.
      Scanner scanner = new Scanner(System.in);

      // Prompt the user to enter the employee's name and read it as a string.
      System.out.print("Enter employee name: ");
      name = scanner.nextLine();

      // Prompt the user to enter the employee's age and read it as an integer.
      System.out.print("Enter employee age: ");
      age = scanner.nextInt();

      // Prompt the user to enter the employee's salary and read it as a double.
      System.out.print("Enter employee salary: ");
      salary = scanner.nextDouble();
   }

   // Method to display employee details.
   public void displayDetails() {
      // Print a header for the employee details.
      System.out.println("\nEmployee Details:");

      // Print the employee's name, age, and salary.
      System.out.println("Name: " + name);
      System.out.println("Age: " + age);
      System.out.println("Salary: " + salary);
   }
}

// Main class that contains the program's entry point.
public class Main {
   public static void main(String[] args) {
      // Print a message to the console.
      System.out.println("Prince_Chhodavadiya");
```

```
        // Create an instance of the Employee class to represent an employee.
        Employee employee = new Employee();

        // Call the inputDetails method to input the employee's information.
        employee.inputDetails();

        // Call the displayDetails method to display the employee's information.
        employee.displayDetails();
    }
}
```

Output:-



```
Prince_Chhodavadiya
Enter employee name: Prince
Enter employee age: 19
Enter employee salary: 100000

Employee Details:
Name: Prince
Age: 19
Salary: 100000.0
PS C:\Users\ASUS\Desktop\Prince\Java>
```

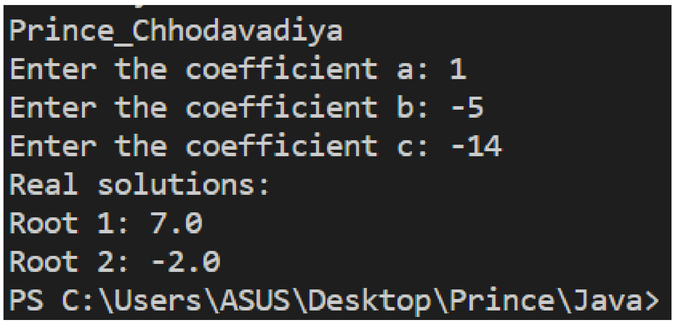
Practical-5

Aim:- Write a Java program that prints all real solutions to the quadratic equation ax2 + bx+c = 0. Read in a, b, c and use the quadratic formula. If the discriminate b2-4ac is negative, display a message stating that there are no real solutions?

Code:-

```java
// Import the Scanner class from the java.util package to read user input.
import java.util.Scanner;

// Main class that contains the program's entry point.
public class Main {
  public static void main(String[] args) {
    // Print a message to the console.
    System.out.println("Prince_Chhodavadiya");
    // Create a Scanner object to read user input.
    Scanner scanner = new Scanner(System.in);

    // Prompt the user to enter coefficients a, b, and c for a quadratic equation.
    System.out.print("Enter the coefficient a: ");
    double a = scanner.nextDouble();

    System.out.print("Enter the coefficient b: ");
    double b = scanner.nextDouble();

    System.out.print("Enter the coefficient c: ");
    double c = scanner.nextDouble();

    // Calculate the discriminant of the quadratic equation.
    double discriminant = b * b - 4 * a * c;

    // Check the discriminant to determine the type of solutions.
    if (discriminant > 0) {
      // Calculate and display real solutions if the discriminant is positive.
      double root1 = (-b + Math.sqrt(discriminant)) / (2 * a);
      double root2 = (-b - Math.sqrt(discriminant)) / (2 * a);

      System.out.println("Real solutions:");
      System.out.println("Root 1: " + root1);
      System.out.println("Root 2: " + root2);
    } else if (discriminant == 0) {
      // Calculate and display a real solution if the discriminant is zero.
      double root = -b / (2 * a);
      System.out.println("Real solution:");
      System.out.println("Root: " + root);
    } else {
      // Display a message indicating no real solutions if the discriminant is negative.
      System.out.println("No real solutions.");
    }
  }
}
```

Output:-



```
Prince_Chhodavadiya
Enter the coefficient a: 1
Enter the coefficient b: -5
Enter the coefficient c: -14
Real solutions:
Root 1: 7.0
Root 2: -2.0
PS C:\Users\ASUS\Desktop\Prince\Java>
```

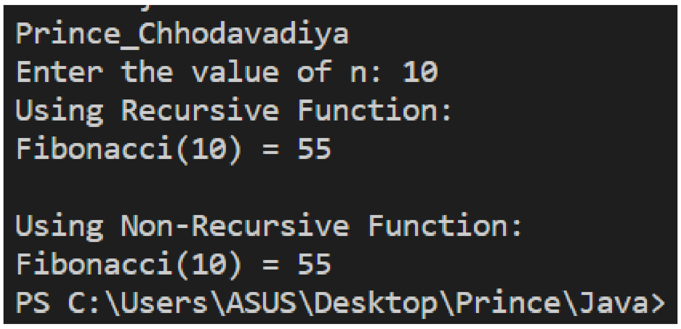
Practical-6

Aim:- The Fibonacci sequence is defined by the following rule. The first 2 values in the sequence are 1, 1. Every subsequent value is the sum of the 2 values preceding it. Write a Java program that uses both recursive and non- recursive functions to print the nth value of the Fibonacci sequence?

Code:-

```
// Import the Scanner class from the java.util package to read user input.
import java.util.Scanner;

// Main class that contains the program's entry point.
public class Main {
    // Recursive function to calculate the nth Fibonacci number
    public static int recursiveFibonacci(int n) {
        // Base case: If n is 0 or 1, return n.
        if (n <= 1) {
            return n;
        }
        // Recursive step: Calculate Fibonacci(n-1) + Fibonacci(n-2).
        return recursiveFibonacci(n - 1) + recursiveFibonacci(n - 2);
    }

    // Non-recursive function to calculate the nth Fibonacci number
    public static int nonRecursiveFibonacci(int n) {
        // Base case: If n is 0 or 1, return n.
        if (n <= 1) {
            return n;
        }

        // Initialize variables to store the previous two Fibonacci numbers.
        int prev1 = 1;
        int prev2 = 1;
        int fib = 0;

        // Calculate the nth Fibonacci number iteratively.
        for (int i = 2; i < n; i++) {
            fib = prev1 + prev2;
            prev1 = prev2;
            prev2 = fib;
        }

        return fib;
    }

    public static void main(String[] args) {
        // Print a message to the console.
        System.out.println("Prince_Chhodavadiya");
        // Create a Scanner object to read user input.
        Scanner scanner = new Scanner(System.in);

        // Prompt the user to enter the value of n.
        System.out.print("Enter the value of n: ");
        int n = scanner.nextInt();

        // Check if n is non-positive and prompt the user to enter a positive integer.
        if (n <= 0) {
            System.out.println("Please enter a positive integer.");
            return;
        }

        // Calculate the nth Fibonacci number using both recursive and non-recursive methods.
        int recursiveResult = recursiveFibonacci(n);
        int nonRecursiveResult = nonRecursiveFibonacci(n);

        // Display the results using both methods.
        System.out.println("Using Recursive Function:");
        System.out.println("Fibonacci(" + n + ") = " + recursiveResult);

        System.out.println("\nUsing Non-Recursive Function:");
        System.out.println("Fibonacci(" + n + ") = " + nonRecursiveResult);
    }
}
```

<u>Output</u>:-



```
Prince_Chhodavadiya
Enter the value of n: 10
Using Recursive Function:
Fibonacci(10) = 55

Using Non-Recursive Function:
Fibonacci(10) = 55
PS C:\Users\ASUS\Desktop\Prince\Java>
```

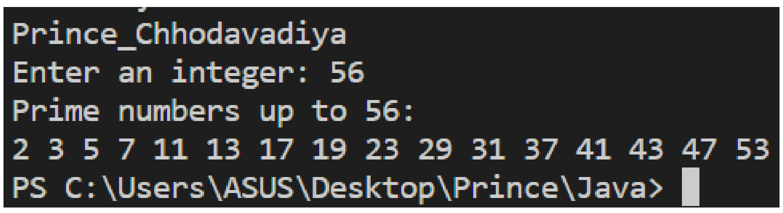
Practical-7

Aim:- Write a Java program that prompts the user for an integer and then prints out all the prime numbers up to that Integer?

Code:-

```
// Import the Scanner class from the java.util package to read user input.
import java.util.Scanner;

// Main class that contains the program's entry point.
public class Main {
   // Function to check if a number is prime
   public static boolean isPrime(int num) {
      // Check if the number is less than or equal to 1; such numbers are not prime.
      if (num <= 1) {
         return false;
      }
      // Check if the number is 2 or 3; they are prime.
      if (num <= 3) {
         return true;
      }
      // Check if the number is divisible by 2 or 3; if yes, it's not prime.
      if (num % 2 == 0 || num % 3 == 0) {
         return false;
      }

      // Check for prime numbers using the 6k +/- 1 rule.
      for (int i = 5; i * i <= num; i += 6) {
         if (num % i == 0 || num % (i + 2) == 0) {
            return false;
         }
      }
      return true;
   }

   public static void main(String[] args) {
      // Print a message to the console.
      System.out.println("Prince_Chhodavadiya");

      // Create a Scanner object to read user input.
      Scanner scanner = new Scanner(System.in);

      // Prompt the user to enter an integer.
      System.out.print("Enter an integer: ");
      int n = scanner.nextInt();

      // Display prime numbers up to the entered integer.
      System.out.println("Prime numbers up to " + n + ":");
      for (int i = 2; i <= n; i++) {
         if (isPrime(i)) {
            System.out.print(i + " ");
            }
        }
    }
}
```

Output:-



```
Prince_Chhodavadiya
Enter an integer: 56
Prime numbers up to 56:
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53
PS C:\Users\ASUS\Desktop\Prince\Java>
```

Practical-8

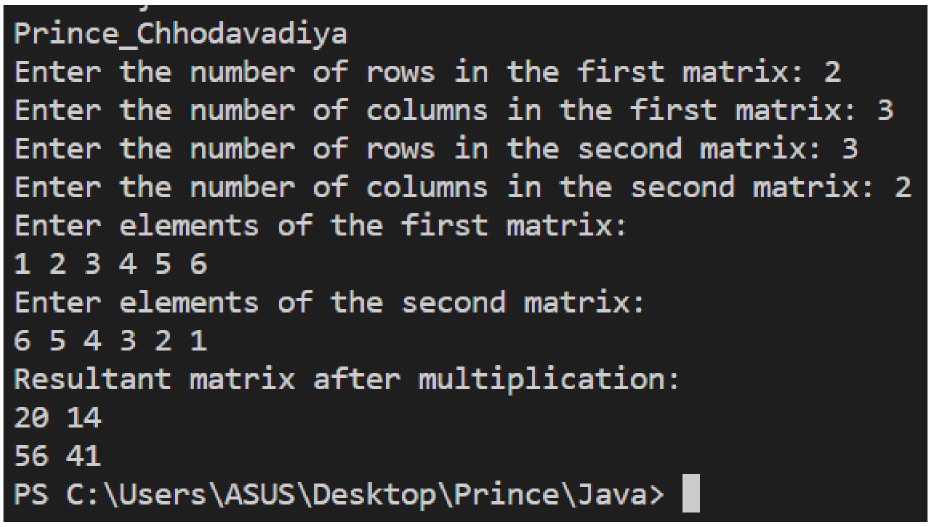
Aim:- Write a Java program to multiply two given matrices?

Code:-

```
// Import the Scanner class from the java.util package to read user input.
import java.util.Scanner;

// Main class that contains the program's entry point.
public class Main {
   public static void main(String[] args) {
      // Print a message to the console.
      System.out.println("Prince_Chhodavadiya");

      // Create a Scanner object to read user input.
      Scanner scanner = new Scanner(System.in);

      // Prompt the user to enter the dimensions of the first and second matrices.
      System.out.print("Enter the number of rows in the first matrix: ");
      int rowsA = scanner.nextInt();
      System.out.print("Enter the number of columns in the first matrix: ");
      int colsA = scanner.nextInt();

      System.out.print("Enter the number of rows in the second matrix: ");
      int rowsB = scanner.nextInt();
      System.out.print("Enter the number of columns in the second matrix: ");
      int colsB = scanner.nextInt();

      // Check if matrix multiplication is possible by comparing dimensions.
      if (colsA != rowsB) {
         System.out.println("Matrix multiplication is not possible.");
         return;
      }

      // Create arrays to store the first and second matrices.
      int[][] matrixA = new int[rowsA][colsA];
      int[][] matrixB = new int[rowsB][colsB];

      // Prompt the user to enter elements of the first matrix.
      System.out.println("Enter elements of the first matrix:");
      for (int i = 0; i < rowsA; i++) {
         for (int j = 0; j < colsA; j++) {
            matrixA[i][j] = scanner.nextInt();
         }
      }

      // Prompt the user to enter elements of the second matrix.
      System.out.println("Enter elements of the second matrix:");
      for (int i = 0; i < rowsB; i++) {
         for (int j = 0; j < colsB; j++) {
            matrixB[i][j] = scanner.nextInt();
         }
```

```
        }

        // Call the multiplyMatrices function to perform matrix multiplication.
        int[][] resultMatrix = multiplyMatrices(matrixA, matrixB, rowsA, colsA, colsB);

        // Display the resultant matrix after multiplication.
        System.out.println("Resultant matrix after multiplication:");
        for (int i = 0; i < rowsA; i++) {
            for (int j = 0; j < colsB; j++) {
                System.out.print(resultMatrix[i][j] + " ");
            }
            System.out.println();
        }
    }

    // Function to multiply two matrices.
    public static int[][] multiplyMatrices(int[][] A, int[][] B, int rowsA, int colsA, int colsB) {
        int[][] result = new int[rowsA][colsB];

        for (int i = 0; i < rowsA; i++) {
            for (int j = 0; j < colsB; j++) {
                result[i][j] = 0;
                for (int k = 0; k < colsA; k++) {
                    result[i][j] += A[i][k] * B[k][j];
                }
            }
        }

        return result;
    }
}
```

Output:-



```
Prince_Chhodavadiya
Enter the number of rows in the first matrix: 2
Enter the number of columns in the first matrix: 3
Enter the number of rows in the second matrix: 3
Enter the number of columns in the second matrix: 2
Enter elements of the first matrix:
1 2 3 4 5 6
Enter elements of the second matrix:
6 5 4 3 2 1
Resultant matrix after multiplication:
20 14
56 41
PS C:\Users\ASUS\Desktop\Prince\Java>
```

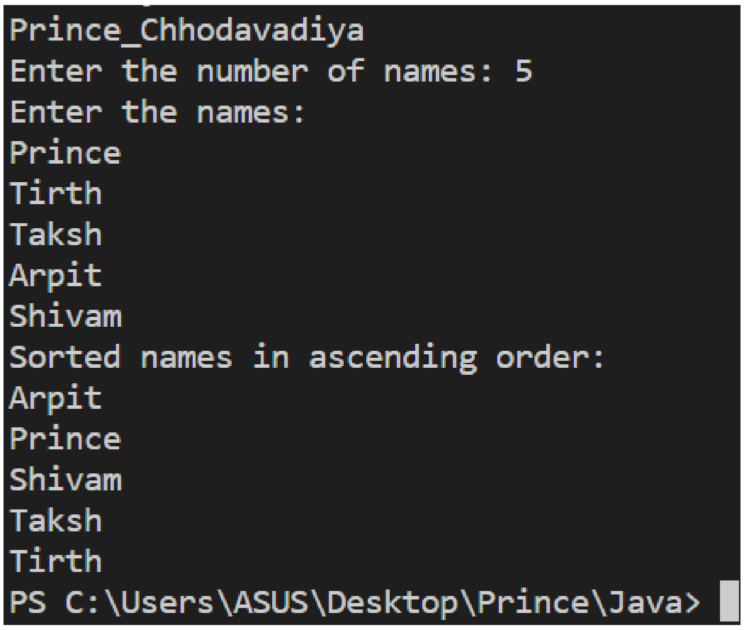
Practical-9

Aim:- Write a Java program for sorting a given list of names in ascending order?

Code:-

```
// Import the necessary classes.
import java.util.Arrays;
import java.util.Scanner;

// Main class that contains the program's entry point.
public class Main {
   public static void main(String[] args) {
      // Print a message to the console.
      System.out.println("Prince_Chhodavadiya");

      // Create a Scanner object to read user input.
      Scanner scanner = new Scanner(System.in);

      // Prompt the user to enter the number of names.
      System.out.print("Enter the number of names: ");
      int numNames = scanner.nextInt();
      scanner.nextLine(); // Consume the newline character

      // Create an array to store the names based on the user's input.
      String[] names = new String[numNames];

      // Prompt the user to enter the names one by one.
      System.out.println("Enter the names:");
      for (int i = 0; i < numNames; i++) {
         names[i] = scanner.nextLine();
      }

      // Sort the names in ascending order using the Arrays.sort method.
      Arrays.sort(names);

      // Display the sorted names in ascending order.
      System.out.println("Sorted names in ascending order:");
      for (String name : names) {
         System.out.println(name);
      }
   }
}
```

Output:-



```
Prince_Chhodavadiya
Enter the number of names: 5
Enter the names:
Prince
Tirth
Taksh
Arpit
Shivam
Sorted names in ascending order:
Arpit
Prince
Shivam
Taksh
Tirth
PS C:\Users\ASUS\Desktop\Prince\Java>
```

Practical-10

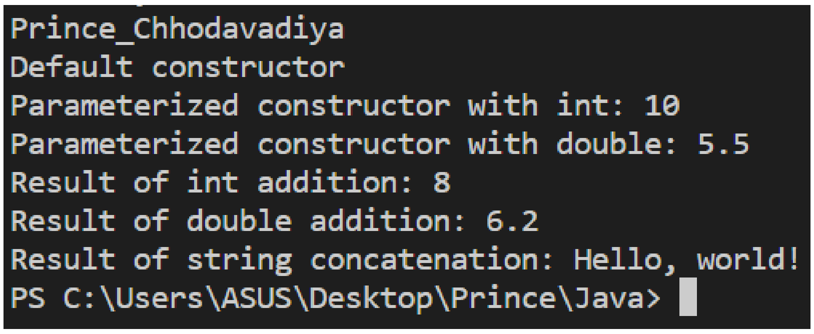
Aim:- Write a java program for Method overloading and Constructor overloading.

Code:-

```
// Create a class named Overloading to demonstrate overloading.
class Overloading {
  // Constructor overloading

  // Default constructor
  public Overloading() {
    System.out.println("Default constructor");
  }

  // Parameterized constructor with an integer argument
  public Overloading(int x) {
    System.out.println("Parameterized constructor with int: " + x);
  }

  // Parameterized constructor with a double argument
  public Overloading(double y) {
    System.out.println("Parameterized constructor with double: " + y);
  }

  // Method overloading

  // Method to add two integers
  public int add(int a, int b) {
    return a + b;
  }

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

  // Method to concatenate two strings
  public String add(String s1, String s2) {
    return s1 + s2;
  }
}

// Main class that contains the program's entry point.
public class Main {
  public static void main(String[] args) {
    // Print a message to the console.
    System.out.println("Prince_Chhodavadiya");

    // Create instances of the Overloading class with different constructors.
    Overloading obj = new Overloading();       // Default constructor
```

```
        Overloading objInt = new Overloading(10);      // Parameterized constructor with int
Overloading objDouble = new Overloading(5.5); // Parameterized constructor with double

        // Call overloaded methods to perform additions and string concatenation.
        int result1 = obj.add(5, 3);
        double result2 = obj.add(2.5, 3.7);
        String result3 = obj.add("Hello, ", "world!");

        // Display the results of method overloading.
        System.out.println("Result of int addition: " + result1);
        System.out.println("Result of double addition: " + result2);
        System.out.println("Result of string concatenation: " + result3);
    }
}
```

<u>Output</u>:-



```
Prince_Chhodavadiya
Default constructor
Parameterized constructor with int: 10
Parameterized constructor with double: 5.5
Result of int addition: 8
Result of double addition: 6.2
Result of string concatenation: Hello, world!
PS C:\Users\ASUS\Desktop\Prince\Java>
```

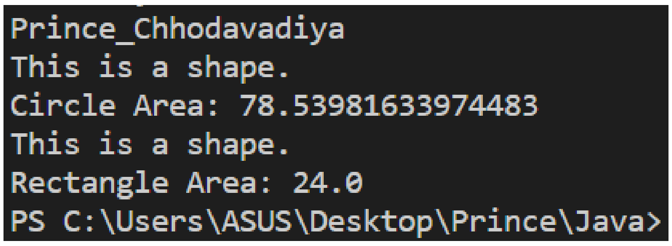
Practical-11

Aim:- **Write a java program to represent Abstract class with example.**

Code:-

```
// Abstract class
abstract class Shape {
   // Abstract method (no implementation)
   abstract double calculateArea();

   // Concrete method
   void display() {
      System.out.println("This is a shape.");
   }
}

// Concrete subclass (Circle)
class Circle extends Shape {
   double radius;

   // Constructor to initialize the circle with a radius
   Circle(double radius) {
      this.radius = radius;
   }

   // Override the abstract method to calculate the area of the circle
   @Override
   double calculateArea() {
      return Math.PI * radius * radius;
   }
}

// Concrete subclass (Rectangle)
class Rectangle extends Shape {
   double length;
   double width;

   // Constructor to initialize the rectangle with length and width
   Rectangle(double length, double width) {
      this.length = length;
      this.width = width;
   }

   // Override the abstract method to calculate the area of the rectangle
   @Override
   double calculateArea() {
      return length * width;
   }
}

// Main class that contains the program's entry point
public class Main {
   public static void main(String[] args) {
      // Print a message to the console
      System.out.println("Prince_Chhodavadiya");

      // Create instances of Circle and Rectangle
      Circle circle = new Circle(5.0);
      Rectangle rectangle = new Rectangle(4.0, 6.0);

      // Call the display method and calculateArea method for Circle
      circle.display();
      System.out.println("Circle Area: " + circle.calculateArea());

      // Call the display method and calculateArea method for Rectangle
      rectangle.display();
      System.out.println("Rectangle Area: " + rectangle.calculateArea());
   }
}
```

<u>Output</u>:-



```
Prince_Chhodavadiya
This is a shape.
Circle Area: 78.53981633974483
This is a shape.
Rectangle Area: 24.0
PS C:\Users\ASUS\Desktop\Prince\Java>
```

Practical-12

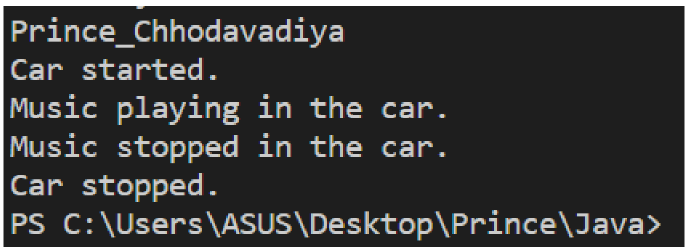
Aim:- Write a java program to implement multiple Inheritances.

Code:-

```
// Interface for a vehicle
interface Vehicle {
  void start(); // Method to start the vehicle
  void stop();  // Method to stop the vehicle
}

// Interface for a music player
interface MusicPlayer {
  void playMusic(); // Method to play music
  void stopMusic();  // Method to stop music
}

// Class representing a Car
class Car implements Vehicle, MusicPlayer {
  // Implementing the start method from the Vehicle interface
  @Override
  public void start() {
    System.out.println("Car started.");
  }

  // Implementing the stop method from the Vehicle interface
  @Override
  public void stop() {
    System.out.println("Car stopped.");
  }

  // Implementing the playMusic method from the MusicPlayer interface
  @Override
  public void playMusic() {
    System.out.println("Music playing in the car.");
  }

  // Implementing the stopMusic method from the MusicPlayer interface
  @Override
  public void stopMusic() {
    System.out.println("Music stopped in the car.");
  }
}

// Main class that contains the program's entry point
public class Main {
  public static void main(String[] args) {
    // Print a message to the console
    System.out.println("Prince_Chhodavadiya");

    // Create an instance of the Car class
    Car car = new Car();
```

```
        // Call various methods to demonstrate vehicle and music player functionality
        car.start();
        car.playMusic();
        car.stopMusic();
        car.stop();
    }
}
```

Output:-



```
Prince_Chhodavadiya
Car started.
Music playing in the car.
Music stopped in the car.
Car stopped.
PS C:\Users\ASUS\Desktop\Prince\Java>
```

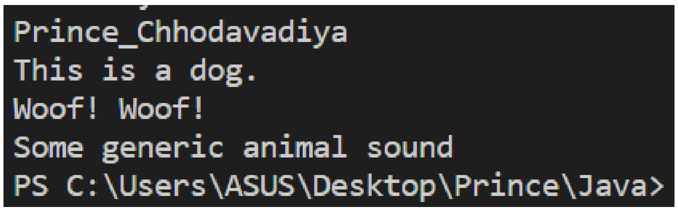
Practical-13

Aim:- Write a java program to demonstrate method overriding and super keyword.

Code:-

```java
// Base class representing an Animal
class Animal {
  // Method to make a generic animal sound
  void makeSound() {
    System.out.println("Some generic animal sound");
  }
}

// Subclass Dog extends Animal
class Dog extends Animal {
  // Override the makeSound method to provide a specific sound for dogs
  @Override
  void makeSound() {
    System.out.println("Woof! Woof!");
  }

  // Additional method specific to the Dog class
  void display() {
    System.out.println("This is a dog.");
  }

  // Method to demonstrate calling the superclass's makeSound method using 'super'
  void makeSuperSound() {
    super.makeSound(); // Calling the superclass's method using 'super'
  }
}
// Main class that contains the program's entry point
public class Main {
  public static void main(String[] args) {
    // Print a message to the console
    System.out.println("Prince_Chhodavadiya");

    // Create an instance of the Dog class
    Dog dog = new Dog();

    // Call the display method of the Dog class
    dog.display(); // Call subclass's method

    // Call the overridden makeSound method of the Dog class
    dog.makeSound(); // Call overridden method in subclass

    // Call the makeSuperSound method to demonstrate calling the superclass's method
    dog.makeSuperSound(); // Call superclass's method using 'super'
  }
}
```

<u>Output</u>:-



```
Prince_Chhodavadiya
This is a dog.
Woof! Woof!
Some generic animal sound
PS C:\Users\ASUS\Desktop\Prince\Java>
```

Practical-14

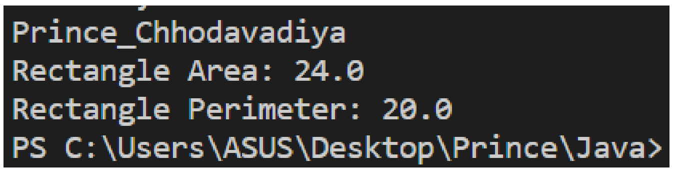
Aim:- Write a java program to implement Interface using extends keyword.

Code:-

```
// Interface for a shape
interface Shape {
   double calculateArea();     // Method to calculate the area of the shape
   double calculatePerimeter(); // Method to calculate the perimeter of the shape
}

// Class representing a rectangle that implements the Shape interface
class Rectangle implements Shape {
   double length;
   double width;

   // Constructor to initialize the rectangle with length and width
   Rectangle(double length, double width) {
      this.length = length;
      this.width = width;
   }

   // Implementing the calculateArea method from the Shape interface
   @Override
   public double calculateArea() {
      return length * width;
   }

   // Implementing the calculatePerimeter method from the Shape interface
   @Override
   public double calculatePerimeter() {
      return 2 * (length + width);
   }
}

// Main class that contains the program's entry point
public class Main {
   public static void main(String[] args) {
      // Print a message to the console
      System.out.println("Prince_Chhodavadiya");

      // Create an instance of the Rectangle class with specified dimensions
      Rectangle rectangle = new Rectangle(4.0, 6.0);

      // Calculate the area and perimeter of the rectangle
      double area = rectangle.calculateArea();
      double perimeter = rectangle.calculatePerimeter();

      // Display the calculated area and perimeter
      System.out.println("Rectangle Area: " + area);
```

```
        System.out.println("Rectangle Perimeter: " + perimeter);
    }
}
```

<u>Output</u>:-



```
Prince_Chhodavadiya
Rectangle Area: 24.0
Rectangle Perimeter: 20.0
PS C:\Users\ASUS\Desktop\Prince\Java>
```

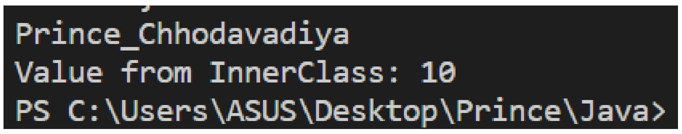
Practical-15

Aim:- Write a java program to create inner classes.

Code:-

```java
// OuterClass represents the outer class
class OuterClass {
   private int outerValue = 10; // Private member variable of the outer class

   // InnerClass represents the inner class within OuterClass
   class InnerClass {
      // Method to display the value of outerValue from the outer class
      void display() {
         System.out.println("Value from InnerClass: " + outerValue);
      }
   }

   // Method in the outer class to create and use the inner class
   void useInnerClass() {
      InnerClass inner = new InnerClass(); // Create an instance of the inner class
      inner.display(); // Call the display method of the inner class
   }
}

// Main class that contains the program's entry point
public class Main {
   public static void main(String[] args) {
      // Print a message to the console
      System.out.println("Prince_Chhodavadiya");

      // Create an instance of the OuterClass
      OuterClass outer = new OuterClass();

      // Call the useInnerClass method to demonstrate using the inner class
      outer.useInnerClass();
   }
}
```

Output:-



```
Prince_Chhodavadiya
Value from InnerClass: 10
PS C:\Users\ASUS\Desktop\Prince\Java>
```

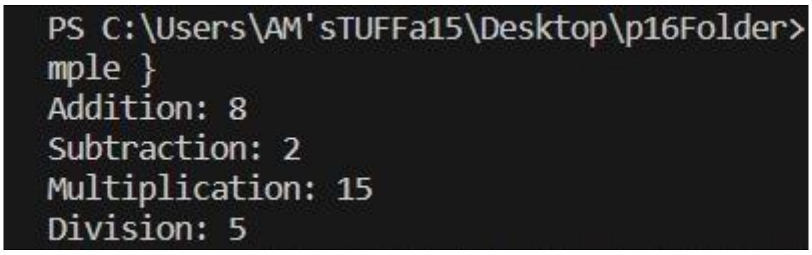
Practical-16

Aim:- Write a java program to create user defined package.

Code:-

```
/* Let's say we want to create a package named "com.example" and include a class called "Calculator"
within it.
Create a new directory structure for your package. In your project folder, create a folder named "com"
and within it, create another folder named "example".
Create the "Calculator.java" file and place it inside the "com/example" folder.
com/example/Calculator.java  */


package com.example;
public class Calculator {
   public int add(int a, int b) {
      return a + b;
   }
   public int subtract(int a, int b) {
      return a - b;
   }
   public int multiply(int a, int b) {
      return a * b;
   }
   public int divide(int a, int b) {
      if (b != 0) {
         return a / b;
      } else {
         throw new ArithmeticException("Cannot divide by zero!");
      }
   }
}


// Now, create another file outside the "com" folder to access the Calculator class from the user-
defined package.


import com.example.Calculator;
public class PackageExample {
   public static void main(String[] args) {
      Calculator calculator = new Calculator();
      int result = calculator.add(5, 3);
      System.out.println("Addition: " + result);
      result = calculator.subtract(5, 3);
      System.out.println("Subtraction: " + result);
      result = calculator.multiply(5, 3);
      System.out.println("Multiplication: " + result);
      result = calculator.divide(10, 2);
      System.out.println("Division: " + result);
   }
}
```

Output:-



```
PS C:\Users\AM'sTUFFa15\Desktop\p16Folder>
mple }
Addition: 8
Subtraction: 2
Multiplication: 15
Division: 5
```

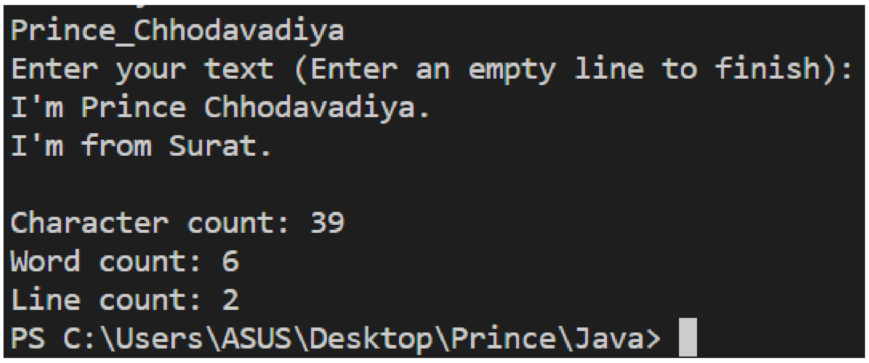
Practical-17

Aim:- Write a Java program that displays the number of characters, lines and words in a text?

Code:-

```
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    // Print a message to the console
    System.out.println("Prince_Chhodavadiya");

    // Create a Scanner object to read user input
    Scanner scanner = new Scanner(System.in);

    // Prompt the user to enter text with instructions
    System.out.println("Enter your text (Enter an empty line to finish):");

    int charCount = 0; // Variable to count characters
    int wordCount = 0;  // Variable to count words
    int lineCount = 0;  // Variable to count lines

    // Continue reading lines of text until an empty line is entered
    while (true) {
      String line = scanner.nextLine();

      if (line.isEmpty()) {
        break;  // Exit the loop when an empty line is entered
      }

      // Count characters by adding the length of the line
      charCount += line.length();

      // Split the line into words using space as a delimiter and count words
      String[] words = line.split("\\s+");
      wordCount += words.length;

      // Increment the line count for each line read
      lineCount++;
    }

    // Display the counts of characters, words, and lines
    System.out.println("Character count: " + charCount);
    System.out.println("Word count: " + wordCount);
    System.out.println("Line count: " + lineCount);
  }
}
```

Output:-



```
Prince_Chhodavadiya
Enter your text (Enter an empty line to finish):
I'm Prince Chhodavadiya.
I'm from Surat.

Character count: 39
Word count: 6
Line count: 2
PS C:\Users\ASUS\Desktop\Prince\Java>
```

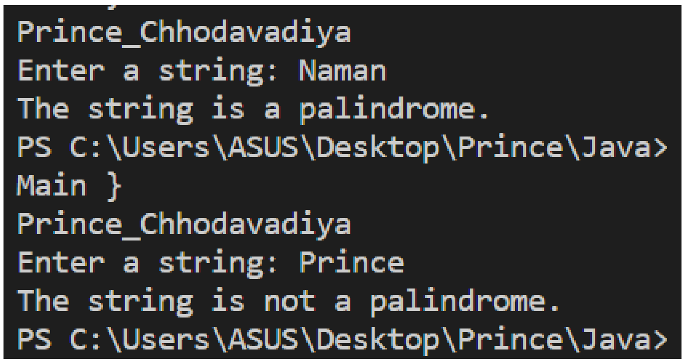
Practical-18

Aim:- Write a Java program that checks whether a given string is a palindrome or not. Ex: MADAM is a palindrome?

Code:-

```java
import java.util.Scanner;

public class Main {
   public static void main(String[] args) {
      // Print a message to the console
      System.out.println("Prince_Chhodavadiya");

      // Create a Scanner object to read user input
      Scanner scanner = new Scanner(System.in);

      // Prompt the user to enter a string
      System.out.print("Enter a string: ");
      String input = scanner.nextLine();

      // Check if the input string is a palindrome and display the result
      if (isPalindrome(input)) {
         System.out.println("The string is a palindrome.");
      } else {
         System.out.println("The string is not a palindrome.");
      }
   }

   // Function to check if a string is a palindrome
   public static boolean isPalindrome(String str) {
      str = str.toLowerCase(); // Convert the string to lowercase for case-insensitive comparison

      int left = 0;
      int right = str.length() - 1;

      // Compare characters from the beginning and end of the string
      while (left < right) {
         if (str.charAt(left) != str.charAt(right)) {
            return false; // If characters do not match, it's not a palindrome
         }
         left++;
         right--;
      }

      return true; // If the loop completes without finding a mismatch, it's a palindrome
   }
}
```

<u>Output</u>:-



```
Prince_Chhodavadiya
Enter a string: Naman
The string is a palindrome.
PS C:\Users\ASUS\Desktop\Prince\Java>
Main }
Prince_Chhodavadiya
Enter a string: Prince
The string is not a palindrome.
PS C:\Users\ASUS\Desktop\Prince\Java>
```

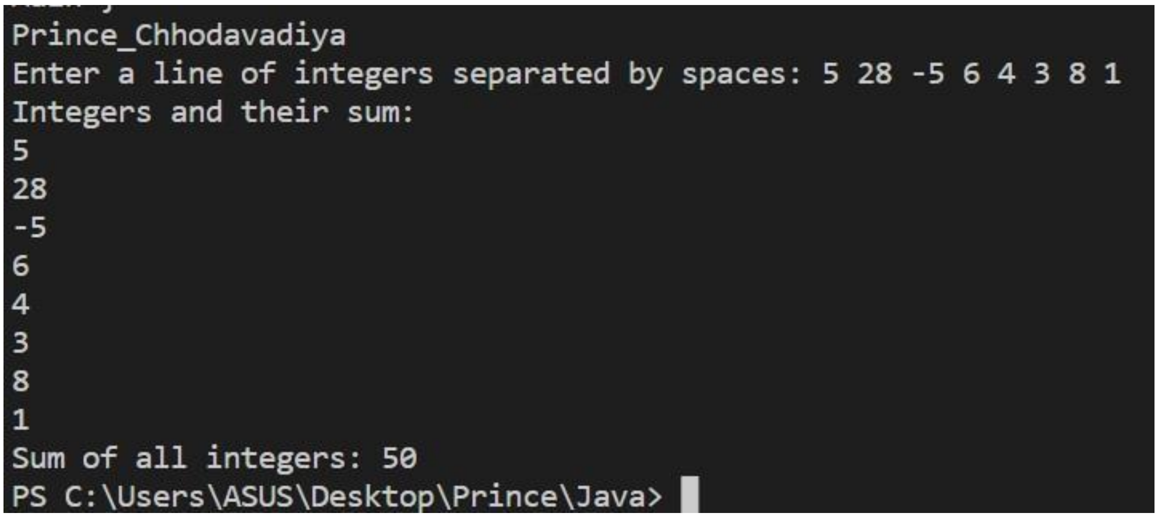
Practical-19

Aim:- Write a Java program that reads a line of integers and then displays each integer and the sum of all integers. (Use StringTokenizer class)?

Code:-

```java
import java.util.Scanner;
import java.util.StringTokenizer;

public class Main {
   public static void main(String[] args) {
      // Print a message to the console
      System.out.println("Prince_Chhodavadiya");

      // Create a Scanner object to read user input
      Scanner scanner = new Scanner(System.in);

      // Prompt the user to enter a line of integers separated by spaces
      System.out.print("Enter a line of integers separated by spaces: ");
      String inputLine = scanner.nextLine();

      // Create a StringTokenizer to tokenize the input string
      StringTokenizer tokenizer = new StringTokenizer(inputLine);
      int sum = 0; // Initialize the sum to zero

      // Display a message indicating the integers and their sum
      System.out.println("Integers and their sum:");

      // Iterate through the tokens (integers) in the input string
      while (tokenizer.hasMoreTokens()) {
         String token = tokenizer.nextToken(); // Get the next token (string)
         int number = Integer.parseInt(token); // Parse the token to an integer

         System.out.println(number); // Print the integer
         sum += number; // Add the integer to the sum
      }

      // Display the sum of all integers in the input
      System.out.println("Sum of all integers: " + sum);
   }
}
```

Output:-



```
Prince_Chhodavadiya
Enter a line of integers separated by spaces: 5 28 -5 6 4 3 8 1
Integers and their sum:
5
28
-5
6
4
3
8
1
Sum of all integers: 50
PS C:\Users\ASUS\Desktop\Prince\Java>
```

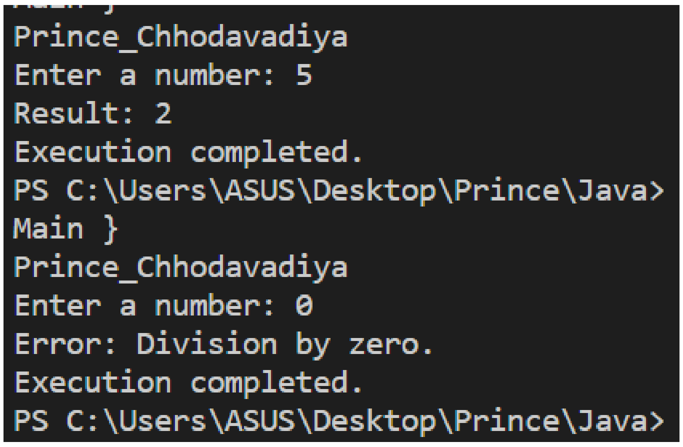
Practical-20

Aim:- Write a java program for creating single try block with multiple catch blocks.

Code:-

```
import java.util.InputMismatchException;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        // Print a message to the console
        System.out.println("Prince_Chhodavadiya");

        // Create a Scanner object to read user input
        Scanner scanner = new Scanner(System.in);

        try {
            // Prompt the user to enter a number
            System.out.print("Enter a number: ");
            int number = scanner.nextInt(); // Read an integer from the user

            // Perform a division operation that may throw an ArithmeticException if 'number' is 0
            int result = 10 / number;

            // Display the result of the division
            System.out.println("Result: " + result);
        } catch (InputMismatchException e) {
            // Handle the case where the user enters invalid input (non-integer)
            System.out.println("Invalid input! Please enter a valid number.");
        } catch (ArithmeticException e) {
            // Handle the case where division by zero occurs
            System.out.println("Error: Division by zero.");
        } catch (Exception e) {
            // Handle any other unexpected exceptions
            System.out.println("An unexpected error occurred.");
        } finally {
            // This block is executed regardless of whether an exception was thrown
            System.out.println("Execution completed.");
            scanner.close(); // Close the scanner to prevent resource leakage
        }
    }
}
```

Output:-



```
Prince_Chhodavadiya
Enter a number: 5
Result: 2
Execution completed.
PS C:\Users\ASUS\Desktop\Prince\Java>
Main }
Prince_Chhodavadiya
Enter a number: 0
Error: Division by zero.
Execution completed.
PS C:\Users\ASUS\Desktop\Prince\Java>
```

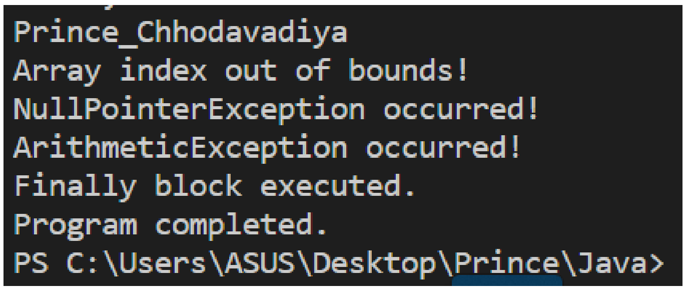
Practical-21

Aim:- Write a program for multiple try blocks and multiple catch blocks including finally.

Code:-

```
public class Main {
  public static void main(String[] args) {
    // Print a message to the console
    System.out.println("Prince_Chhodavadiya");

    try {
      // Attempt to access an element outside the bounds of the array
      int[] numbers = { 1, 2, 3 };
      System.out.println(numbers[5]); // This will throw an ArrayIndexOutOfBoundsException
    } catch (ArrayIndexOutOfBoundsException e) {
      // Handle the specific exception for array index out of bounds
      System.out.println("Array index out of bounds!");
    }

    try {
      // Attempt to access the length of a null string
      String str = null;
      System.out.println(str.length()); // This will throw a NullPointerException
    } catch (NullPointerException e) {
      // Handle the specific exception for a null pointer
      System.out.println("NullPointerException occurred!");
    }

    try {
      // Attempt to perform division by zero
      int result = 10 / 0; // This will throw an ArithmeticException
      System.out.println("Result: " + result);
    } catch (ArithmeticException e) {
      // Handle the specific exception for arithmetic errors
      System.out.println("ArithmeticException occurred!");
    } finally {
      // This block is executed regardless of whether an exception was thrown
      System.out.println("Finally block executed.");
    }

    // Display a message indicating that the program has completed
    System.out.println("Program completed.");
  }
}
```

Output:-



```
Prince_Chhodavadiya
Array index out of bounds!
NullPointerException occurred!
ArithmeticException occurred!
Finally block executed.
Program completed.
PS C:\Users\ASUS\Desktop\Prince\Java>
```

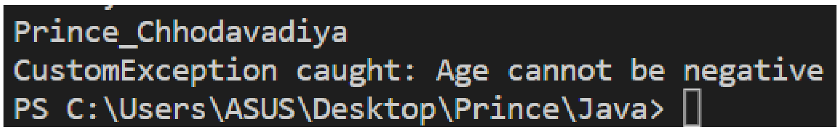
Practical-22

Aim:- Write a program to create user defined exception.

Code:-

```
// Custom exception class that extends Exception
class CustomException extends Exception {
   public CustomException(String message) {
      super(message);
   }
}

public class Main {
   public static void main(String[] args) {
      // Print a message to the console
      System.out.println("Prince_Chhodavadiya");

      try {
         // Attempt to set a negative age
         int age = -5;
         if (age < 0) {
            // Throw a CustomException with a custom error message
            throw new CustomException("Age cannot be negative");
         }
         System.out.println("Age: " + age);
      } catch (CustomException e) {
         // Catch and handle the CustomException
         System.out.println("CustomException caught: " + e.getMessage());
      }
   }
}
```

Output:-



```
Prince_Chhodavadiya
CustomException caught: Age cannot be negative
PS C:\Users\ASUS\Desktop\Prince\Java> 
```

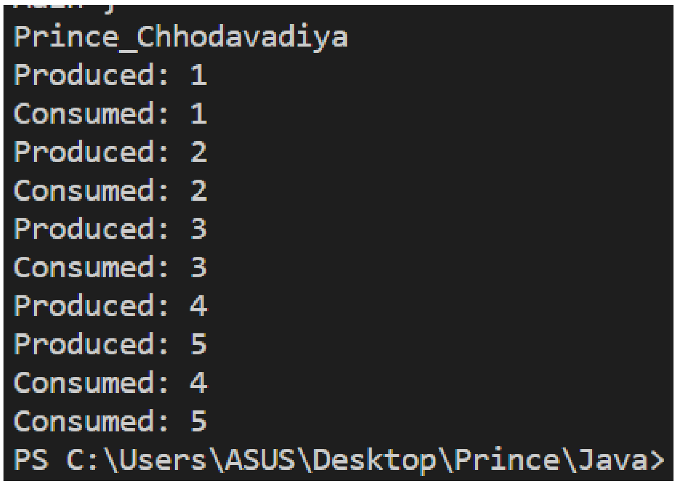
Practical-23

Aim:- Write a java program for producer and consumer problem using Threads.

Code:-

```java
import java.util.LinkedList;
import java.util.Queue;

// A class representing a buffer for producing and consuming items
class Buffer {
    private Queue<Integer> queue;
    private int maxSize;

    public Buffer(int maxSize) {
        this.maxSize = maxSize;
        this.queue = new LinkedList<>();
    }

    // Method for producing an item and adding it to the buffer
    public synchronized void produce(int item) throws InterruptedException {
        while (queue.size() == maxSize) {
            wait(); // Wait if the buffer is full
        }
        queue.add(item); // Add the item to the buffer
        System.out.println("Produced: " + item);
        notify(); // Notify waiting consumers that an item is available
    }

    // Method for consuming an item from the buffer
    public synchronized int consume() throws InterruptedException {
        while (queue.isEmpty()) {
            wait(); // Wait if the buffer is empty
        }
        int item = queue.poll(); // Remove and retrieve the item from the buffer
        System.out.println("Consumed: " + item);
        notify(); // Notify waiting producers that space is available
        return item;
    }
}

// A producer thread that adds items to the buffer
class Producer implements Runnable {
    private Buffer buffer;

    public Producer(Buffer buffer) {
        this.buffer = buffer;
    }

    @Override
    public void run() {
        try {
            for (int i = 1; i <= 5; i++) {
                buffer.produce(i); // Produce an item and add it to the buffer
                Thread.sleep(1000); // Sleep for 1 second
            }
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
    }
}

// A consumer thread that removes items from the buffer
class Consumer implements Runnable {
    private Buffer buffer;

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

    @Override
    public void run() {
        try {
            for (int i = 1; i <= 5; i++) {
                int item = buffer.consume(); // Consume an item from the buffer
                Thread.sleep(1500); // Sleep for 1.5 seconds
            }
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
    }
}

public class Main {
    public static void main(String[] args) {
        // Print a message to the console
        System.out.println("Prince_Chhodavadiya");

        // Create a buffer with a maximum size of 3
        Buffer buffer = new Buffer(3);

        // Create producer and consumer threads
        Thread producerThread = new Thread(new Producer(buffer));
        Thread consumerThread = new Thread(new Consumer(buffer));

        // Start the producer and consumer threads
        producerThread.start();
        consumerThread.start();
    }
}
```

Output:-



```
Prince_Chhodavadiya
Produced: 1
Consumed: 1
Produced: 2
Consumed: 2
Produced: 3
Consumed: 3
Produced: 4
Produced: 5
Consumed: 4
Consumed: 5
PS C:\Users\ASUS\Desktop\Prince\Java>
```

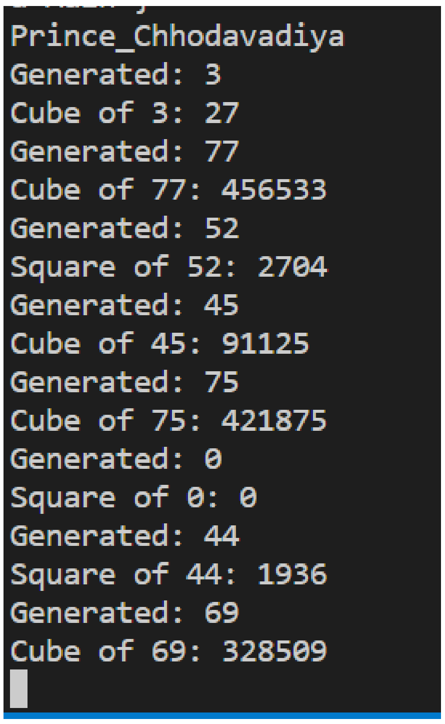
Practical-24

Aim:- Write a java program that implements a multi-thread application that has three threads. First thread generates random integer every 1 second and if the value is even, second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of cube of the number.

Code:-

```
import java.util.Random;

// A Runnable class that generates random numbers
class RandomNumberGenerator implements Runnable {
    @Override
    public void run() {
        Random random = new Random();

        while (true) {
            int number = random.nextInt(100); // Generate a random integer between 0 and 99
            System.out.println("Generated: " + number);

            // Depending on whether the number is even or odd, create and start threads for square or cube
calculation
            if (number % 2 == 0) {
                Thread t2 = new Thread(new SquareCalculator(number));
                t2.start(); // Start the square calculator thread
            } else {
                Thread t3 = new Thread(new CubeCalculator(number));
                t3.start(); // Start the cube calculator thread
            }

            try {
                Thread.sleep(1000); // Sleep for 1 second
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        }
    }
}

// A Runnable class that calculates the square of a number
class SquareCalculator implements Runnable {
    private int number;

    public SquareCalculator(int number) {
        this.number = number;
    }

    @Override
    public void run() {
        int square = number * number;
        System.out.println("Square of " + number + ": " + square);
    }
}

// A Runnable class that calculates the cube of a number
class CubeCalculator implements Runnable {
    private int number;

    public CubeCalculator(int number) {
        this.number = number;
    }

    @Override
    public void run() {
        int cube = number * number * number;
        System.out.println("Cube of " + number + ": " + cube);
    }
}

public class Main {
    public static void main(String[] args) {
        // Print a message to the console
        System.out.println("Prince_Chhodavadiya");

        // Create and start the random number generator thread
        Thread t1 = new Thread(new RandomNumberGenerator());
        t1.start();
    }
}
```

Output:-



```
Prince_Chhodavadiya
Generated: 3
Cube of 3: 27
Generated: 77
Cube of 77: 456533
Generated: 52
Square of 52: 2704
Generated: 45
Cube of 45: 91125
Generated: 75
Cube of 75: 421875
Generated: 0
Square of 0: 0
Generated: 44
Square of 44: 1936
Generated: 69
Cube of 69: 328509
```

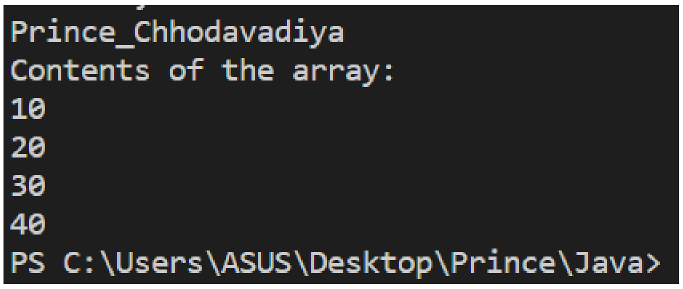
Practical-25

Aim:- Write a program to create dynamic array using ArrayList class and the print the contents of the array object.

Code:-

```
import java.util.ArrayList;

public class Main {
   public static void main(String[] args) {
      // Print a message to the console
      System.out.println("Prince_Chhodavadiya");

      // Create an ArrayList of integers called dynamicArray
      ArrayList<Integer> dynamicArray = new ArrayList<>();

      // Add elements to the ArrayList
      dynamicArray.add(10);
      dynamicArray.add(20);
      dynamicArray.add(30);
      dynamicArray.add(40);

      // Print the contents of the array
      System.out.println("Contents of the array:");
      for (int i = 0; i < dynamicArray.size(); i++) {
         System.out.println(dynamicArray.get(i)); // Get and print the element at index 'i'
      }
   }
}
```

Output:-



```
Prince_Chhodavadiya
Contents of the array:
10
20
30
40
PS C:\Users\ASUS\Desktop\Prince\Java>
```

Practical-26

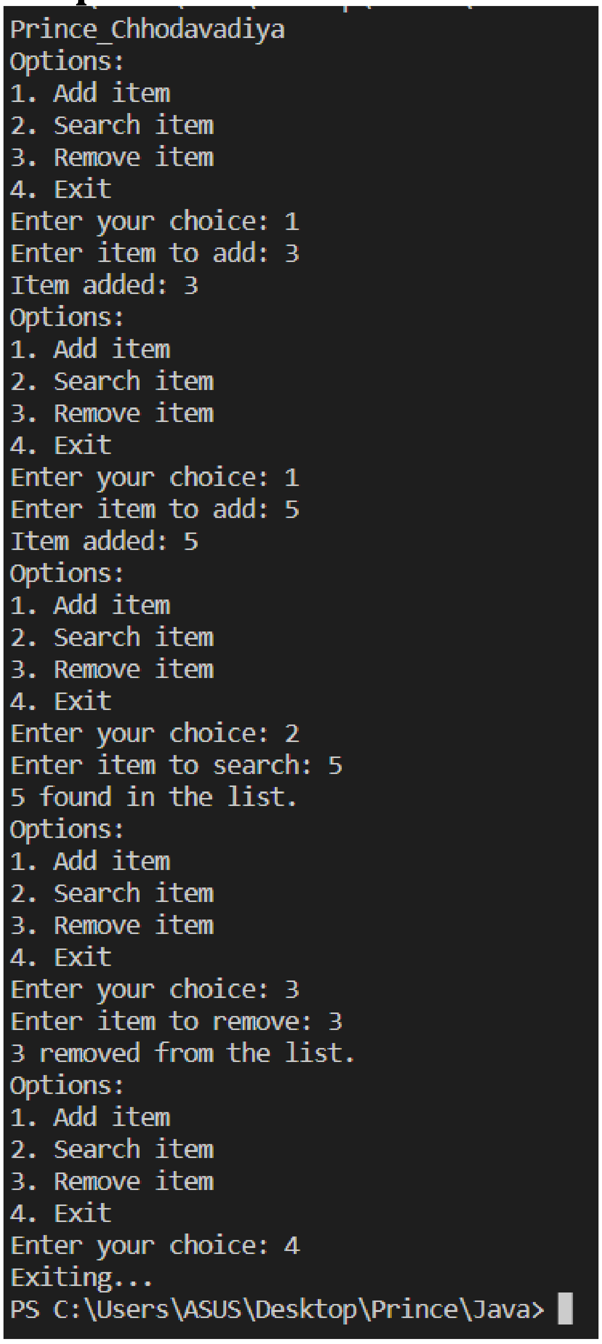
Aim:- Write programs to implement add, search and remove operation on ArrayList object.

Code:-

```
import java.util.ArrayList;
import java.util.Scanner;

public class Main {
   public static void main(String[] args) {
      System.out.println("Prince_Chhodavadiya");

      // Create an ArrayList to store items
      ArrayList<String> items = new ArrayList<>();

      // Create a Scanner for user input
      Scanner scanner = new Scanner(System.in);

      while (true) {
         System.out.println("Options:");
         System.out.println("1. Add item");
         System.out.println("2. Search item");
         System.out.println("3. Remove item");
         System.out.println("4. Exit");
         System.out.print("Enter your choice: ");

         // Read the user's choice
         int choice = scanner.nextInt();
         scanner.nextLine(); // Consume the newline character

         switch (choice) {
            case 1:
               System.out.print("Enter item to add: ");
               String newItem = scanner.nextLine();
               items.add(newItem);
               System.out.println("Item added: " + newItem);
               break;
            case 2:
               System.out.print("Enter item to search: ");
               String searchItem = scanner.nextLine();

               // Check if the item exists in the list
               if (items.contains(searchItem)) {
                  System.out.println(searchItem + " found in the list.");
               } else {
                  System.out.println(searchItem + " not found in the list.");
               }
               break;
            case 3:
               System.out.print("Enter item to remove: ");
```

```
                String removeItem = scanner.nextLine();

                // Check if the item exists in the list and remove it
                if (items.contains(removeItem)) {
                    items.remove(removeItem);
                    System.out.println(removeItem + " removed from the list.");
                } else {
                    System.out.println(removeItem + " not found in the list.");
                }
                break;
            case 4:
                System.out.println("Exiting...");
                scanner.close(); // Close the scanner
                return;
            default:
                System.out.println("Invalid choice. Please choose again.");
        }
    }
  }
}
```

Output:-



```
Prince_Chhodavadiya
Options:
1. Add item
2. Search item
3. Remove item
4. Exit
Enter your choice: 1
Enter item to add: 3
Item added: 3
Options:
1. Add item
2. Search item
3. Remove item
4. Exit
Enter your choice: 1
Enter item to add: 5
Item added: 5
Options:
1. Add item
2. Search item
3. Remove item
4. Exit
Enter your choice: 2
Enter item to search: 5
5 found in the list.
Options:
1. Add item
2. Search item
3. Remove item
4. Exit
Enter your choice: 3
Enter item to remove: 3
3 removed from the list.
Options:
1. Add item
2. Search item
3. Remove item
4. Exit
Enter your choice: 4
Exiting...
PS C:\Users\ASUS\Desktop\Prince\Java>
```