/* PROGRAM TO FIND SUM AND PRODUCT OF N NUMBERS */

```
import java.util.Scanner;
public class SumAndProduct {
   public static void main(String[] args) {
// Initialize sum and product
      int sum = 0;
      long product = 1; // Use long to handle larger products
      Scanner scanner = new Scanner(System.in);
      // Asking for the number of inputs
      System.out.print("Enter the number of elements: ");
      int n = scanner.nextInt();
      // Loop to get n numbers from the user
      System.out.println("Enter " + n + " numbers:");
      for (int i = 0; i < n; i++) {
         int number = scanner.nextInt();
         sum += number; // Add to sum
         product *= number; // Multiply for product
      }
      // Display the results
      System.out.println("Sum: " + sum);
      System.out.println("Product: " + product);
      // Close the scanner
      scanner.close();
   }
}
```

OUTPUT

C:\Users\Admin>D:

D:\>cd PS

D:\PS> javac SumAndProduct.java

D:\PS> java SumAndProduct

Enter the number of elements: 3

Enter 3 numbers:

2

4

6

Sum: 12

Product: 48

Explanation:

- The number of elements is 3.
- The numbers entered are 2, 4, and 6.
- Sum: 2 + 4 + 6 = 12
- Product: 2 * 4 * 6 = 48

/* PROGRAM TO FIND SUM OF INDIVIDUAL DIGITS */

```
import java.util.Scanner;
public class SumOfDigits {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      // Ask the user for a number
      System.out.print("Enter a number: ");
      int number = scanner.nextInt();
      // Initialize sum
      int sum = 0;
      // Loop to extract and sum digits
      while (number != 0) {
         sum += number % 10; // Add the last digit to the sum
         number /= 10;      // Remove the last digit
      }
      // Display the result
      System.out.println("Sum of digits: " + sum);
      // Close the scanner
      scanner.close();
   }
}
```

OUTPUT

D:\PS> javac SumOfDigits.java

D:\PS> java SumOfDigits

Enter a number: 4567

Sum of digits: 22

Explanation:

- First digit: 7
- Second digit: 6
- Third digit: 5
- Fourth digit: 4
- 4 + 5 + 6 + 7 = 22

/*PROGRAM TO PRINT NO. OF EVENS AND NO. OF ODDS FROM 1 TO N */

```
import java.util.Scanner;
public class CountEvenOdd {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      // Ask the user for the upper limit
      System.out.print("Enter a number N: ");
      int N = scanner.nextInt();
      int evenCount = 0;
      int oddCount = 0;
      // Loop to count even and odd numbers
      for (int i = 1; i <= N; i++) {
         if (i % 2 == 0) {
            evenCount++; // Increment even count
         } else {
            oddCount++;  // Increment odd count
         }
      }
      // Display the results
      System.out.println("Number of even numbers from 1 to " + N + ": " + evenCount);
      System.out.println("Number of odd numbers from 1 to " + N + ": " + oddCount);
      // Close the scanner
      scanner.close();
   }
}
```

OUTPUT

D:\PS> javac CountEvenOdd.java

D:\PS> java CountEvenOdd

Enter a number N: 7

Number of even numbers from 1 to 7: 3

Number of odd numbers from 1 to 7: 4

Explanation:

- Numbers from 1 to 7: 1, 2, 3, 4, 5, 6, 7
- Even numbers: 2, 4, 6
- Odd numbers: 1, 3, 5, 7

/*PROGRAM TO PRINT EVENODDNUMBERS*/

```
import java.util.Scanner;
public class EvenOddNumbers {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      // Ask the user for the upper limit
      System.out.print("Enter a number N: ");
      int N = scanner.nextInt();
      System.out.println("Even numbers from 1 to " + N + ":");
      for (int i = 1; i <= N; i++) {
         if (i % 2 == 0) {
            System.out.print(i + " "); // Print even number
         }
      }
      System.out.println("\nOdd numbers from 1 to " + N + ":");
      for (int i = 1; i <= N; i++) {
         if (i % 2 != 0) {
            System.out.print(i + " "); // Print odd number
         }
      }
      // Close the scanner
      scanner.close();
   }
}
```

OUTPUT

D:\PS> javac EvenOddNumbers.java

D:\PS> java EvenOddNumbers

Enter a number N: 10

Even numbers from 1 to 10:

2 4 6 8 10

Odd numbers from 1 to 10:

1 3 5 7 9

/*PROGRAM TO FIND FIBONACCI SEQUENCE */

```
import java.util.Scanner;
public class Fibonacci {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // Ask the user for the number of terms to display
        System.out.print("Enter the number of terms in Fibonacci sequence: ");
        int n = scanner.nextInt();
        // Initializing the first two terms of Fibonacci sequence
        int first = 0, second = 1;
        System.out.println("Fibonacci sequence:");
        // Loop to print Fibonacci sequence up to n terms
        for (int i = 1; i <= n; i++) {
            System.out.print(first + " ");  // Print the current term
            // Calculate the next term
            int nextTerm = first + second;
            first = second;  // Update first
            second = nextTerm;  // Update second
        }
        // Close the scanner
        scanner.close();
    }
}
```

OUTPUT

D:\PS> javac Fibonacci.java

D:\PS> java Fibonacci

Enter the number of terms in Fibonacci sequence: 10

Fibonacci sequence:

0 1 1 2 3 5 8 13 21 34

/*PROGRAM TO FIND PRIME NUMBERS FROM 1 TO N */

```
import java.util.Scanner;
public class PrimeNumbers {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // Ask the user for the upper limit
        System.out.print("Enter a number N: ");
        int N = scanner.nextInt();
        System.out.println("Prime numbers from 1 to " + N + ":");
        for (int num = 2; num <= N; num++) {
            boolean isPrime = true;
            // Check if num is prime
            for (int i = 2; i <= Math.sqrt(num); i++) {
                if (num % i == 0) {
                    isPrime = false;
                    break;
                }  }
            // Print the prime number
            if (isPrime) {
                System.out.print(num + " ");
            }
        }
        // Close the scanner
        scanner.close();
    }
}
```

OUTPUT

D:\PS> javac PrimeNumbers.java

D:\PS> java PrimeNumbers

Enter a number N: 20

Prime numbers from 1 to 20:

2 3 5 7 11 13 17 19

Explanation:

The prime numbers between 1 and 20 are:

- **2** (divisible only by 1 and 2)
- **3** (divisible only by 1 and 3)
- **5** (divisible only by 1 and 5)
- **7** (divisible only by 1 and 7)
- **11** (divisible only by 1 and 11)
- **13** (divisible only by 1 and 13)
- **17** (divisible only by 1 and 17)
- **19** (divisible only by 1 and 19)

/*PROGRAM TO PRINT ARMSTRONG NUMBERS FROM 1 TO 1000 */

```
public class ArmstrongNumbers {
    public static void main(String[] args) {
        System.out.println("Armstrong numbers from 1 to 1000:");
        for (int num = 1; num <= 1000; num++) {
            int originalNum = num;
            int sum = 0;
            int digits = String.valueOf(num).length();
            // Calculate the sum of the digits raised to the power of the number of digits
            while (originalNum != 0) {
                int digit = originalNum % 10;
                sum += Math.pow(digit, digits);
                originalNum /= 10;
            }
            // Check if the number is an Armstrong number
            if (sum == num) {
                System.out.print(num + " ");
            }
        }
    }
}
```

OUTPUT

D:\PS> javac ArmstrongNumbers.java

D:\PS> java ArmstrongNumbers

Armstrong numbers from 1 to 1000:

1 153 370 371 407

Explanation:

1: 11=11^1 = 111=1

153: 13+53+33=1531^3 + 5^3 + 3^3 = 15313+53+33=153

370: 33+73+03=3703^3 + 7^3 + 0^3 = 37033+73+03=370

371: 33+73+13=3713^3 + 7^3 + 1^3 = 37133+73+13=371

407: 43+03+73=4074^3 + 0^3 + 7^3 = 40743+03+73=407

/*PROGRAM TO PERFORM MATRIX ADDITION */

```
import java.util.Scanner;
public class MatrixAddition {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      // Input matrix dimensions
      System.out.println("Enter the number of rows and columns for the matrices: ");
      int rows = scanner.nextInt();
      int cols = scanner.nextInt();
      int[][] matrix1 = new int[rows][cols];
      int[][] matrix2 = new int[rows][cols];
      int[][] resultAdd = new int[rows][cols];
      // Input elements for matrix 1
      System.out.println("Enter elements for matrix 1:");
      for (int i = 0; i < rows; i++) {
         for (int j = 0; j < cols; j++) {
            matrix1[i][j] = scanner.nextInt();
         }
      }
      // Input elements for matrix 2
      System.out.println("Enter elements for matrix 2:");
      for (int i = 0; i < rows; i++) {
         for (int j = 0; j < cols; j++) {
            matrix2[i][j] = scanner.nextInt();
         }
      }
```

```
        // Matrix addition
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                resultAdd[i][j] = matrix1[i][j] + matrix2[i][j];
            }
        }
        // Display result of addition
        System.out.println("Result of Matrix 1 + Matrix 2:");
        displayMatrix(resultAdd);
        scanner.close();
    }
    // Method to display the matrix
    public static void displayMatrix(int[][] matrix) {
        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[i].length; j++) {
                System.out.print(matrix[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

OUTPUT

D:\PS> javac MatrixAddition.java

D:\PS> java MatrixAddition

Enter the number of rows and columns for the matrices:

2 3

Enter elements for matrix 1:

1 2 3

4 5 6

Enter elements for matrix 2:

7 8 9

10 11 12

Result of Matrix 1 + Matrix 2:

8 10 12

14 16 18

/*PROGRAM TO PERFORM MATRIX MULTIPLICATION*/

```
import java.util.Scanner;
public class MatrixMultiplication {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // Input matrix dimensions for matrix 1
        System.out.println("Enter the number of rows and columns for matrix 1: ");
        int rows1 = scanner.nextInt();
        int cols1 = scanner.nextInt();
        // Input matrix dimensions for matrix 2
        System.out.println("Enter the number of rows and columns for matrix 2: ");
        int rows2 = scanner.nextInt();
        int cols2 = scanner.nextInt();
        // Check if multiplication is possible
        if (cols1 != rows2) {
            System.out.println("Matrix multiplication is not possible. Number of columns in
matrix 1 must equal number of rows in matrix 2.");
            return;
        }
        int[][] matrix1 = new int[rows1][cols1];
        int[][] matrix2 = new int[rows2][cols2];
        int[][] resultMul = new int[rows1][cols2];
        // Input elements for matrix 1
        System.out.println("Enter elements for matrix 1:");
        for (int i = 0; i < rows1; i++) {
            for (int j = 0; j < cols1; j++) {
                matrix1[i][j] = scanner.nextInt();
```

```
}
    }
    // Input elements for matrix 2
    System.out.println("Enter elements for matrix 2:");
    for (int i = 0; i < rows2; i++) {
      for (int j = 0; j < cols2; j++) {
        matrix2[i][j] = scanner.nextInt();
      }
    }
    // Matrix multiplication
    for (int i = 0; i < rows1; i++) {
      for (int j = 0; j < cols2; j++) {
        resultMul[i][j] = 0;
        for (int k = 0; k < cols1; k++) {
          resultMul[i][j] += matrix1[i][k] * matrix2[k][j];
        }
      }
    }
    // Display result of multiplication
    System.out.println("Result of Matrix 1 * Matrix 2:");
    displayMatrix(resultMul);
    scanner.close();
  }
  // Method to display the matrix
  public static void displayMatrix(int[][] matrix) {
    for (int i = 0; i < matrix.length; i++) {
```

```
  for (int j = 0; j < matrix[i].length; j++) {
  System.out.print(matrix[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

OUTPUT

D:\PS> javac MatrixMultiplication.java

D:\PS> java MatrixMultiplication

Enter the number of rows and columns for matrix 1:

2 3

Enter the number of rows and columns for matrix 2:

3 2

Enter elements for matrix 1:

1 2 3

4 5 6

Enter elements for matrix 2:

7 8

9 10

11 12

Result of Matrix 1 * Matrix 2:

58 64

139 154

Explanation:

Matrix 1 (2x3):

1 2 3

4 5 6

Matrix 2 (3x2):

7 8

9 10

11 12

The result of multiplying these matrices is:

The element at position (0,0) is calculated as:

(1*7) + (2*9) + (3*11) = 7 + 18 + 33 = 58

The element at position (0,1) is calculated as:

(1*8) + (2*10) + (3*12) = 8 + 20 + 36 = 64

The element at position (1,0) is calculated as:

(4*7) + (5*9) + (6*11) = 28 + 45 + 66 = 139

The element at position (1,1) is calculated as:

(4*8) + (5*10) + (6*12) = 32 + 50 + 72 = 154

So, the final result is:

58 64

139 154

/* PROGRAM TO FIND LARGEST AND SMALLEST NUMBERS IN LIST */

```
public class FindLargestAndSmallest {
    public static void main(String[] args) {
        int[] numbers = {5, 2, 8, -1, 6, 3, 7};
        // Initialize smallest and largest with the first element of the array
        int smallest = numbers[0], largest = numbers[0];
        // Loop through the array to find the smallest and largest
        for (int num : numbers) {
            if (num < smallest) smallest = num;
            if (num > largest) largest = num;
        }
        // Output the results
        System.out.println("Smallest number: " + smallest);
        System.out.println("Largest number: " + largest);
    }
}
```

OUTPUT

D:\PS> javac FindLargestAndSmallest.java

D:\PS> java FindLargestAndSmallest

Smallest number: -1

Largest number: 8

/*PROGRAM TO CHECK GIVEN STRING IS PALINDROME OR NOT */

```
import java.util.*;
class PN{
public static void main(String args[]){
Scanner os =new Scanner(System.in);
System.out.println("Enter the string");
String str=os.nextLine();
StringBuffer str1= new StringBuffer(str);
str1.reverse();
String rev=str1.toString();
if(rev.compareTo(str)==0)
System.out.println("the given string"+str+ "is palindrome");
else
System.out.println("the given string"+str+ "is not a palindrome");
}
}
```

OUTPUT

D:\PS> javac PN.java

D:\PS> java PN

For madam (a palindrome):

Enter the string

madam

the given string madam is palindrome

For hello (not a palindrome):

Enter the string

hello

the given string hello is not a palindrome

/* PROGRAM TO SORT THE NAMES IN ASCENDING ORDER */

```
import java.util.Arrays;
import java.util.Collections;
public class SortNames {
    public static void main(String[] args) {
        // List of names to be sorted
        String[] names = {"John", "Alice", "Bob", "Charlie", "Diana"};
        // Sorting in ascending order
        Arrays.sort(names);
        System.out.println("Names in ascending order:");
        for (String name : names) {
            System.out.println(name);
        }
        // Sorting in descending order
        Arrays.sort(names, Collections.reverseOrder());
        System.out.println("\nNames in descending order:");
        for (String name : names) {
            System.out.println(name);
        }
    }
}
```

OUTPUT

D:\PS> javac SortNames.java

D:\PS> java SortNames

Names in ascending order:

Alice

Bob

Charlie

Diana

John

Names in descending order:

John

Diana

Charlie

Bob

Alice

/* INSERTING THE SUBSTRING IN TO GIVEN MAIN STRING FROM A GIVEN POSITION. DELTING 'N' CHARACTERS FROM A GIVEN POSITION IN A STRING. */

```
import java.util.*;
class SubString {
   public static void main(String args[]) {
      Scanner os = new Scanner(System.in);
      // Input: Main string
      System.out.println("Enter the main string:");
      String str = os.nextLine();
      // Convert to StringBuffer for modification
      StringBuffer str1 = new StringBuffer(str);
      // Input: Position to insert the substring
      System.out.println("Enter the position to insert the substring:");
      int n = os.nextInt();
      // Consume the newline character left by nextInt()
      os.nextLine();
      // Input: Substring to insert
      System.out.println("Enter the substring to insert:");
      String str3 = os.nextLine();
      // Insert the substring into the main string at the specified position
      StringBuffer newstring = str1.insert(n, str3);
      System.out.println("After inserting the substring, the new String is: " + newstring);
      // Input: Start index for deletion
      System.out.println("Enter the Start Index to delete:");
      int a = os.nextInt();
      // Input: End index for deletion
      System.out.println("Enter the End Index to delete:");
      int b = os.nextInt();
      // Delete the substring from the start index to the end index
      System.out.println("After deletion, the string is: " + newstring.delete(a, b));
      // Close scanner
      os.close();
   }
}
```

OUTPUT

D:\PS> javac SubString.java

D:\PS> java SubString

Enter the main string:

HelloWorld

Enter the position to insert the substring:

5

Enter the substring to insert:

Java

After inserting the substring, the new String is: HelloJavaWorld

Enter the Start Index to delete:

0

Enter the End Index to delete:

5

After deletion, the string is: JavaWorld

//CONCATENATION OF TWO STRINGS & COMPARISION OF TWO STRINGS

```
import java.util.Scanner;
public class StringOperations {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // Input: First string
        System.out.print("Enter the first string: ");
        String str1 = scanner.nextLine();
        // Input: Second string
        System.out.print("Enter the second string: ");
        String str2 = scanner.nextLine();
        // Concatenating the two strings
        String concatenatedString = str1 + str2;
        System.out.println("Concatenated String: " + concatenatedString);
        // String comparison (case-sensitive)
        if (str1.equals(str2)) {
            System.out.println("The two strings are equal (case-sensitive).");
        } else {
            System.out.println("The two strings are not equal (case-sensitive).");
        }
        // String comparison (case-insensitive)
        if (str1.equalsIgnoreCase(str2)) {
            System.out.println("The two strings are equal (case-insensitive).");
        } else {
            System.out.println("The two strings are not equal (case-insensitive).");
        }
```

```
        // Close the scanner
        scanner.close();
    }
}
```

OUTPUT

D:\PS> javac StringOperations.java

D:\PS> java StringOperations

1.

Enter the first string: Hello

Enter the second string: world

Concatenated String: Helloworld

The two strings are not equal (case-sensitive).

The two strings are not equal (case-insensitive).

2.

Enter the first string: hello

Enter the second string: Hello

Concatenated String: helloHello

The two strings are not equal (case-sensitive).

The two strings are equal (case-insensitive).

/*PROGRAM TO ILLUSTRATE CONSTRUCTOR OVERLOADING.*/

```
class Rectangle {
   double length, breadth;
   // Constructor with two parameters
   Rectangle(double l, double b) {
      length = l;
      breadth = b;
   }
   // Constructor with one parameter (for creating a square)
   Rectangle(double a) {
      length = a;
      breadth = a;
   }
   // Method to calculate the area
   double Area() {
      return length * breadth;
   }
}
public class Const_Overloading {
   public static void main(String args[]) {
      double area1, area2;
      // Rectangle with two parameters
      Rectangle r1 = new Rectangle(7.5, 8.5);
      area1 = r1.Area();
      System.out.println("The area of Rectangle using Two Parameters=" + area1);
      // Rectangle with one parameter (square)
```

```
        Rectangle rec2 = new Rectangle(8.5);
        area2 = rec2.Area();
        System.out.println("The area of Rectangle using One Parameter=" + area2);
    }
}
```

OUTPUT

D:\PS> javac Const_Overloading.java

D:\PS> java Const_Overloading

The area of Rectangle using Two Parameters= 63.75

The area of Rectangle using One Parameter= 72.25

/*PROGRAM TO ILLUSTRATE METHOD OVERLOADING.*/

```
class Rectangle {
   double length, breadth;
   // Method to calculate the area of a rectangle
   double Area(double l, double b) {
      length = l;
      breadth = b;
      return length * breadth;
   }
   // Method to calculate the area of a square
   double Area(double a) {
      length = a;
      breadth = a;
      return length * breadth;
   }
}
public class Overloading {
   public static void main(String args[]) {
      double area;
      // Create an instance of the Rectangle class
      Rectangle rec = new Rectangle();
      // Call the method for calculating area of rectangle
      area = rec.Area(7.5, 4.5);
      System.out.print("Area of Rectangle is=" + area);
      System.out.println();
```

```
// Call the method for calculating area of square
      area = rec.Area(8.5);
      System.out.print("Area of Square is=" + area);
   }
}
```

OUTPUT

D:\PS> javac Overloading.java

D:\PS> java Overloading

Area of Rectangle is= 33.75

Area of Square is= 72.25

/* PROGRAM TO ILLUSTRATE THE CONCEPT OF SINGLE INHERITANCE */

```
// Class representing a Rectangle
class ARect {
   double length, breadth;
   // Constructor to initialize length and breadth
   ARect(double x, double y) {
      length = x;
      breadth = y;
   }
   // Method to calculate the area of the rectangle
   double area() {
      return (length * breadth);
   }
}
// Class representing a Cube (or Cuboid) that extends ARect
class VCube extends ARect {
   double height;
   // Constructor to initialize length, breadth, and height
   VCube(double x, double y, double z) {
      super(x, y);  // Calling the constructor of ARect
      height = z;
   }
   // Method to calculate the volume of the cube
   double volume() {
      return area() * height;  // Volume = Area * Height
   }
```

```
}
// Main class to execute the program
public class SingleInheritance {
   public static void main(String args[]) {
      // Creating an object of VCube with dimensions 10, 20, and 30
      VCube v1 = new VCube(10, 20, 30);
      // Calculating the area of the rectangle
      double Rect_Area = v1.area();
      // Calculating the volume of the cube
      double Cube_Vol = v1.volume();
      // Printing the results
      System.out.println("Area of the Rectangle: " + Rect_Area);
      System.out.println("Volume of the Cube: " + Cube_Vol);
   }
}
```

OUTPUT

D:\PS> javac SingleInheritance.java

D:\PS> java SingleInheritance

Area of the Rectangle: 200.0

Volume of the Cube: 6000.0

/* PROGRAM TO ILLUSTRATE THE CONCEPT OF MULTILEVEL INHERITANCE */

```
// Base class representing an Employee
class Employee {
   int eno;  // Employee number
   double bsal;  // Basic salary
   // Constructor to initialize employee number and basic salary
   Employee(int e, double b) {
      eno = e;
      bsal = b;
   }
   // Method to calculate HRA (House Rent Allowance)
   double Hra() {
      return (bsal * 0.08);  // HRA is 8% of basic salary
   }
   // Method to calculate DA (Dearness Allowance)
   double Da() {
      return (bsal * 0.11);  // DA is 11% of basic salary
   }
   // Method to calculate PF (Provident Fund)
   double Pf() {
      return (bsal * 0.07);  // PF is 7% of basic salary
   }
   // Method to display employee details
   void display() {
      System.out.println("Emp number=" + eno);
      System.out.println("Basic salary=" + bsal);
```

```
 }
}
// Class to calculate the Gross salary which inherits from Employee
class Gsalary extends Employee {
   // Constructor to initialize the Employee part
   Gsalary(int e, double b) {
      super(e, b);
   }
   // Method to calculate Gross Salary
   double gSalary() {
      return (bsal + Hra() + Da());  // Gross salary = basic salary + HRA + DA
   }
}
// Class to calculate Net salary which inherits from Gsalary
class Nsalary extends Gsalary {
   // Constructor to initialize the Employee and Gsalary part
   Nsalary(int e, double b) {
      super(e, b);
   }
   // Method to calculate Net Salary
   double nSalary() {
      return (gSalary() - Pf());  // Net salary = Gross salary - PF
   }
}
// Main class to run the program
public class Multilevel {
```

```
    public static void main(String args[]) {
   // Declare variables to hold HRA, DA, PF, Gross Salary, and Net Salary
       double hra, da, pf, gs, ns;
       // Create an instance of Nsalary with employee number 10 and basic salary 2000
       Nsalary n1 = new Nsalary(10, 2000);
       // Calculate allowances and salaries
       hra = n1.Hra();
       da = n1.Da();
       pf = n1.Pf();
       gs = n1.gSalary();
       ns = n1.nSalary();
       // Display employee details
       n1.display();
       // Print the calculated values
       System.out.println("Hra=" + hra);
       System.out.println("Da=" + da);
       System.out.println("Pf=" + pf);
       System.out.println("Gross salary=" + gs);
       System.out.println("Net salary=" + ns);
    }
}
```

OUTPUT

D:\PS> javac Multilevel.java

D:\PS> java Multilevel

Emp number=10

Basic salary=2000.0

Hra=160.0

Da=220.0

Pf=140.0

Gross salary=2380.0

Net salary=2240.0

Explanation:

Given the **employee number** (eno) is 10 and **basic salary** (bsal) is 2000, let's calculate each component:

1. **HRA** (8% of basic salary):

   HRA=2000×0.08=160HRA = 2000 \times 0.08 = 160HRA=2000×0.08=160

2. **DA** (11% of basic salary):

   DA=2000×0.11=220DA = 2000 \times 0.11 = 220DA=2000×0.11=220

3. **PF** (7% of basic salary):

   PF=2000×0.07=140PF = 2000 \times 0.07 = 140PF=2000×0.07=140

4. **Gross Salary** (Basic salary + HRA + DA):

   GS=2000+160+220=2380GS = 2000 + 160 + 220 = 2380GS=2000+160+220=2380

5. **Net Salary** (Gross salary - PF):

   NS=2380−140=2240NS = 2380 - 140 = 2240NS=2380−140=2240

/* PROGRAM TO ILLUSTRATE THE CONCEPT OF HIERARCHICAL INHERITANCE */

```
// Base class representing a Rectangle
class Rectangle {
    double length, breadth;
    // Constructor to initialize length and breadth of the rectangle
    Rectangle(double l, double b) {
        length = l;
        breadth = b;
    }
    // Method to calculate the area of the rectangle
    double rectangleArea() {
        return length * breadth;
    }
}
// Class representing a Triangle that inherits from Rectangle
class Triangle extends Rectangle {
    // Constructor to initialize length and breadth for the triangle
    Triangle(double l, double b) {
        super(l, b);  // Call the Rectangle constructor
    }
    // Method to calculate the area of the triangle
    double triangleArea() {
        return 0.5 * rectangleArea();  // Area of triangle = 0.5 * area of rectangle
    }
}
// Class representing a Cuboid that inherits from Rectangle
```

```
class Volume extends Rectangle {
   double height;
   // Constructor to initialize length, breadth, and height for the cuboid
   Volume(double l, double b, double h) {
      super(l, b);  // Call the Rectangle constructor
      height = h;
   }
   // Method to calculate the volume of the cuboid
   double cuboidVolume() {
      return rectangleArea() * height;  // Volume of cuboid = area of rectangle * height
   }
}
// Main class to execute the program
public class Hierarchical {
   public static void main(String args[]) {
      // Declare variables to hold areas and volume
      double rectangleArea, triangleArea, cuboidVolume;
      // Create instances of Triangle and Volume with given dimensions
      Triangle obj1 = new Triangle(1.2, 2.3);  // Rectangle with length=1.2, breadth=2.3 for triangle
      Volume obj2 = new Volume(1.2, 2.3, 3.4);  // Rectangle with length=1.2, breadth=2.3, height=3.4 for cuboid
      // Calculate areas and volume
      rectangleArea = obj1.rectangleArea();
      triangleArea = obj1.triangleArea();
      cuboidVolume = obj2.cuboidVolume();
      // Output the results
```

```
        System.out.println("Area of Rectangle = " + rectangleArea);
        System.out.println("Area of Triangle = " + triangleArea);
        System.out.println("Volume of Cuboid = " + cuboidVolume);
    }
}
```

OUTPUT

D:\PS> javac Hierarchical.java

D:\PS> java Hierarchical

Area of Rectangle = 2.76

Area of Triangle = 1.38

Volume of Cuboid = 9.384

/*PROGRAM TO ILLUSTRATE THE CONCEPT OVERRIDING */

```
// Base class representing a Square
class Square {
    int a;  // Side length of the square
    // Constructor to initialize the side of the square
    Square(int side) {
        a = side;
    }
    // Method to calculate the area of the square
    void area() {
        int areaSquare = a * a;
        System.out.println("Area of the Square is : " + areaSquare);
    }
}
// Derived class representing a Cube (which is a 3D square)
class Cube extends Square {
    // Constructor to initialize the side of the cube (same as the square)
    Cube(int s) {
        super(s);  // Call the constructor of the Square class
    }
    // Overriding the area method to calculate the volume of the cube
    void volume() {
        int volumeCube = a * a * a;  // Volume of cube = side^3
        System.out.println("Volume of the Cube is : " + volumeCube);
    }
}
```

```
// Main class to test the program
class Overriding {
    public static void main(String args[]) {
        // Create an object of Cube with side length 4
        Cube CubeObj = new Cube(4);
        // Calling the volume method of Cube
        CubeObj.volume();
    }
}
```

OUTPUT

D:\PS> javac Overriding.java

D:\PS> java Overriding

Volume of the Cube is : 64

/* PROGRAM TO ILLUSTRATE THE CONCEPT OF INTERFACE */

```
import java.lang.*;
// Interface Deposit with method interest
interface Deposit {
   double interest();
}
// Class Simple implements Deposit interface
class Simple implements Deposit {
   double sim, stot, p, t, r; // Principal, Time, Rate
   // Constructor for Simple Interest
   Simple(double a, double b, double c) {
      p = a; // Principal
      t = b; // Time in years
      r = c; // Rate of interest
   }
   // Method to calculate simple interest
   public double interest() {
      sim = (p * t * r) / 100; // Simple Interest formula
      System.out.println("Simple Interest: " + sim);
      stot = sim + p; // Total amount = Simple Interest + Principal
      return stot;
   }
}
// Class Compound extends Simple and implements Deposit interface
class Compound extends Simple {
   double com, ctot; // Compound Interest and Total amount
```

```
    // Constructor for Compound Interest
    Compound(double a, double d, double q) {
        super(a, d, q); // Call Simple class constructor
    }
    // Method to calculate compound interest
    public double interest() {
        // Compound Interest formula: A = P * (1 + r/100)^t
        com = p * Math.pow((1 + r / 100), t) - p; // Compound Interest
        System.out.println("Compound Interest: " + com);
        ctot = p + com; // Total amount = Principal + Compound Interest
        return ctot;
    }
}
// Main class to demonstrate Simple and Compound Interest calculation
class Interfacedemo {
    public static void main(String args[]) throws Exception {
        // Creating Simple interest object
        Simple p = new Simple(10000, 2, 3); // Principal = 10000, Time = 2 years, Rate =
3%
        double a = p.interest(); // Calculate simple interest
        System.out.println("Principal with Simple Interest: " + a);
        // Creating Compound interest object
        Compound q = new Compound(10000, 2, 3); // Principal = 10000, Time = 2 years,
Rate = 3%
        double b = q.interest(); // Calculate compound interest
        System.out.println("Principal with Compound Interest: " + b);
```

```
        }
}
```

OUTPUT

D:\PS> javac Interfacedemo.java

D:\PS> java Interfacedemo

Simple Interest: 600.0

Principal with Simple Interest: 10600.0

Compound Interest: 609.0

Principal with Compound Interest: 10609.0

/* PROGRAM TO ILLUSTRATE THE CONCEPT OF ABSTRACT CLASS */

```
// Abstract class shape with an abstract method sides
abstract class Shape {
   // Abstract method to be implemented by subclasses
   abstract void sides();
}
// Class Trapezoid extends Shape and implements the sides method
class Trapezoid extends Shape {
   void sides() {
      System.out.println("Trapezoid sides = 5");
   }
}
// Class Triangle extends Shape and implements the sides method
class Triangle extends Shape {
   void sides() {
      System.out.println("Triangle sides = 3");
   }
}
// Class Hexagon extends Shape and implements the sides method
class Hexagon extends Shape {
   void sides() {
      System.out.println("Hexagon sides = 6");
   }
}
public class Adtt {
   public static void main(String args[]) {
```

```
// Create objects of each shape class
        Shape s1 = new Trapezoid();
        Shape c1 = new Triangle();
        Shape h1 = new Hexagon();
        // Call the sides method for each object
        s1.sides();
        c1.sides();
        h1.sides();
    }
}
```

OUTPUT

D:\PS> javac Adtt.java

D:\PS> java Adtt

Trapezoid sides = 5

Triangle sides = 3

Hexagon sides = 6

/*PROGRAM TO CREATE AND IMPORT PACKAGES*/

```
  package p1;
  public class Rectangle {
     public static double area(double l, double b) {
        return l * b;
     }
}
package p1;
public class Square {
   public static double area(double m) {
      return m * m;
   }
}
package p1;
public class Triangle {
   public static double area(double a, double b) {
      return 0.5 * a * b;
   }
}
import p1.Rectangle;
import p1.Triangle;
import p1.Square;
public class pdemo {
   public static void main(String[] args) {
      // Calculate areas of different shapes
      double a = Rectangle.area(5.2, 3.2);  // Rectangle area
```

```
        double b = Triangle.area(5.0, 3.0);  // Triangle area
        double c = Square.area(5.2);        // Square area
        // Print the areas
        System.out.println("Area of rectangle: " + a);
        System.out.println("Area of triangle: " + b);
        System.out.println("Area of square: " + c);
    }
}
```

OUTPUT

D:\PS> javac -d . p1/Rectangle.java p1/Square.java p1/Triangle.java p1/pdemo.java

D:\PS> java p1.pdemo

Area of rectangle: 16.64

Area of triangle: 7.5

Area of square: 27.04

/* PROGRAM TO ILLUSTRATE PREDEFINED EXCEPTION*/

```
class Excep3 {
    public static void main(String args[]) {
        try {
            int d = args.length;
            int a = 42 / d; // May cause ArithmeticException
            int c[] = {1};
            c[5] = 40; // May cause ArrayIndexOutOfBoundsException
        } catch (ArithmeticException e) {
            System.out.println("Division by zero error occurred");
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Please check that index is out of Bound");
        } catch (Exception e) { // Generic catch block to handle any other exceptions
            System.out.println("An unexpected error occurred: " + e);
        }
        System.out.println("Last Stmt in the program");
    }
}
```

OUTPUT

D:\PS> javac Excep3.java

D:\PS> java Excep3 // With no arguments

Division by zero error occurred

Last Stmt in the program

D:\PS> java Excep3 // One or more arguments

Please check that index is out of Bound

Last Stmt in the program

/* PROGRAM TO ILLUSTRATE USER DEFINED EXCEPTION*/

```
import java.io.*;
class MyException extends Exception {
   MyException(String s) {
      super(s);
   }
}
class ThrowDemo {
   public static void main(String args[]) {
      // Corrected arrays
      int accno[] = {100, 101, 102, 103, 104}; // Only 5 elements, indices 0 to 4
      String name[] = {"A", "B", "C", "D", "E"}; // Changed char array to String for customer names
      double bal[] = {10000, 20000, 15000, 999, 25000}; // Balances array
      try {
         System.out.println("ACCNO\tCUSTOMER\tBALANCE");
         for (int i = 0; i < accno.length; i++) { // Loop until accno.length
            System.out.println(accno[i] + "\t" + name[i] + "\t" + bal[i]);
            if (bal[i] < 1000) { // Check if balance is less than 1000
               MyException me = new MyException("Balance less for account " + accno[i] + ": " + bal[i]);
               throw me; // Throw custom exception
            }
         }
      } catch (MyException e) {
         System.out.println(e.toString()); // Print the exception message
      }
```

```
        }
}
```

OUTPUT

D:\PS> javac ThrowDemo.java

D:\PS> java ThrowDemo

ACCNO CUSTOMER BALANCE

100 A 10000.0

101 B 20000.0

102 C 15000.0

103 D 999.0

Balance less for account 103: 999.0

/* CREATING MULTIPLE THREADS BY EXTENDING THREAD CLASS */

```
class A extends Thread {
   public void run() {
      try {
         for (int i = 1; i <= 5; i++) {
            System.out.println("From Thread A: " + i);
            Thread.sleep(100); // Sleep for 100 milliseconds
         }
      } catch (InterruptedException e) {
         System.out.println(e);
      }
   }
}
class B extends Thread {
   public void run() {
      try {
         for (int i = 1; i <= 5; i++) {
            System.out.println("From Thread B: " + i);
            Thread.sleep(100); // Sleep for 100 milliseconds
         }
      } catch (InterruptedException e) {
         System.out.println(e);
      }
   }
}
```

```
public class MultiThreadExample {
   public static void main(String args[]) {
      A t1 = new A(); // Create thread t1 of class A
      B t2 = new B(); // Create thread t2 of class B
      t1.start(); // Start thread t1
      t2.start(); // Start thread t2
   }
}
```

OUTPUT

D:\PS> javac MultiThreadExample.java

D:\PS> java MultiThreadExample

Case 1:

From Thread A: 1

From Thread B: 1

From Thread A: 2

From Thread B: 2

From Thread A: 3

From Thread B: 3

From Thread A: 4

From Thread B: 4

From Thread A: 5

From Thread B: 5

Case 2:

From Thread A: 1

From Thread A: 2

From Thread B: 1

From Thread B: 2

From Thread A: 3

From Thread B: 3

From Thread A: 4

From Thread B: 4

From Thread A: 5

From Thread B: 5

/* CREATING MULTIPLE THREADS BY IMPLEMENTING RUNNABLE INTERFACE*/

```
class A implements Runnable {
   public void run() {
      try {
         for (int i = 1; i <= 5; i++) {
            System.out.println("From Thread A: " + i);
            Thread.sleep(100); // Sleep for 100 milliseconds
         }
      } catch (InterruptedException e) {
         System.out.println(e);
      }
   }
}
class B implements Runnable {
   public void run() {
      try {
         for (int i = 1; i <= 5; i++) {
            System.out.println("From Thread B: " + i);
            Thread.sleep(100); // Sleep for 100 milliseconds
         }
      } catch (InterruptedException e) {
         System.out.println(e);
      }
   }
}
```

```
public class MultiThread_Runnable {
  public static void main(String args[]) {
      A t1 = new A();  // Create an object of class A (implements Runnable)
      B t2 = new B();  // Create an object of class B (implements Runnable)
      // Create Thread objects, passing the Runnable tasks as arguments
      Thread th1 = new Thread(t1);
      Thread th2 = new Thread(t2);
      // Start the threads
      th1.start();
      th2.start();
  }
}
```

OUTPUT

D:\PS> javac MultiThread_Runnable.java

D:\PS> java MultiThread_Runnable

Case 1:

From Thread A: 1

From Thread B: 1

From Thread A: 2

From Thread B: 2

From Thread A: 3

From Thread B: 3

From Thread A: 4

From Thread B: 4

From Thread A: 5

From Thread B: 5

Case 2:

From Thread A: 1

From Thread A: 2

From Thread B: 1

From Thread A: 3

From Thread B: 2

From Thread A: 4

From Thread B: 3

From Thread A: 5

From Thread B: 4

From Thread B: 5

/* FREQUENCY COUNT OF WORDS USING STRINGTOKENIZER*/

```
import java.util.StringTokenizer;
class StDemo {
    public static void main(String args[]) {
        String str = "This is a java program"; // Input string
        StringTokenizer st = new StringTokenizer(str); // Tokenize based on spaces
        int sum = 0; // Initialize word count
        while (st.hasMoreTokens()) {
            st.nextToken(); // Get the next word (token)
            sum++; // Increment word count
        }
        System.out.println("The given sentence is: " + str);
        System.out.println("Number of words in the sentence is: " + sum);
    }
}
```

OUTPUT

D:\PS> javac StDemo.java

D:\PS> java StDemo

The given sentence is: This is a java program

Number of words in the sentence is: 4

/* PROGRAM TO ILLUSTRATE Vector CLASS*/

```
import java.util.*;
class VectorDemo{
public static void main(String args[]){
Vector<Integer> V=new Vector<Integer>();
int a[]={22,20,10,40,56,8};
for(int i=0;i<a.length;i++){
V.add(a[i]);
}
System.out.println("vector elements:");
for(int i=0;i<V.size();i++){
System.out.println(V.get(i));
}
System.out.println("Retrieving elements using ListIterator Interface:");
ListIterator Lit=V.listIterator();
System.out.println("IN Forward Direction:");
while(Lit.hasNext())
System.out.print(Lit.next()+"\t");
System.out.println("\n In Backward Direction:");
while(Lit.hasPrevious())
System.out.print(Lit.previous()+"\t");
}
}
```

OUTPUT

D:\PS> javac VectorDemo.java

D:\PS> java VectorDemo

vector elements:

22

20

10

40

56

8

Retrieving elements using ListIterator Interface:

IN Forward Direction:

22 20 10 40 56 8

In Backward Direction:

8 56 40 10 20 22

/* PROGRAM TO CREATE HASHSET FOR STORING AND RETRIEVING ELEMENTS*/

```
import java.util.*;
class HSDemo{
public static void main(String args[]){
HashSet<String> hs=new HashSet<String>();
hs.add("INDIA");
hs.add("AMERICA");
hs.add("japan");
hs.add("china");
hs.add("AMERICA");
System.out.println("Hashset="+hs);
Iterator it=hs.iterator();
System.out.println("Retrieving elements using iterator interface");
while(it.hasNext())
{
String s=(String)it.next();
System.out.println(s);
}
}
}
```

OUTPUT

D:\PS> javac HSDemo.java

D:\PS> java HSDemo

Hashset=[INDIA, AMERICA, japan, china]

Retrieving elements using iterator interface

INDIA

AMERICA

japan

china

//PROGRAM TO CREATE STACK FOR STORING AND RETRIEVING ELEMENTS

```
import java.util.Stack;
import java.util.Scanner;
public class StackExample {
    public static void main(String[] args) {
        // Create a Stack of strings
        Stack<String> stack = new Stack<>();
        // Create a Scanner object for user input
        Scanner scanner = new Scanner(System.in);
        // Ask user how many elements they want to push onto the stack
        System.out.print("How many elements do you want to push onto the stack? ");
        int n = scanner.nextInt();
        scanner.nextLine(); // Consume the leftover newline
        // Push elements onto the Stack
        for (int i = 0; i < n; i++) {
            System.out.print("Enter element " + (i + 1) + ": ");
            String element = scanner.nextLine();
            stack.push(element); // Push element onto the stack
        }
        // Display the elements in the Stack using traditional for loop
        System.out.println("\nThe elements in the Stack are:");
        for (int i = 0; i < stack.size(); i++) {
            System.out.println(stack.get(i)); // Print each element by index
        }
        // Ask user for the element to pop (remove) from the stack
        System.out.print("\nDo you want to pop an element from the stack? (yes/no): ");
```

```
String response = scanner.nextLine();
    if (response.equalsIgnoreCase("yes") && !stack.isEmpty()) {
        String poppedElement = stack.pop(); // Pop the top element from the stack
       System.out.println("Element '" + poppedElement + "' has been popped.");
    } else {
        System.out.println("No element popped (either no or empty stack).");
    }
    // Display the elements in the Stack after popping using traditional for loop
    System.out.println("\nThe elements in the Stack after popping:");
    for (int i = 0; i < stack.size(); i++) {
        System.out.println(stack.get(i)); // Print each remaining element by index
    }
    // Ask user for the index to replace an element
    System.out.print("\nDo you want to change an element in the stack? (yes/no): ");
    response = scanner.nextLine();
    if (response.equalsIgnoreCase("yes") && !stack.isEmpty()) {
        System.out.print("Enter the index of the element you want to change: ");
        int indexToChange = scanner.nextInt();
        scanner.nextLine(); // Consume the leftover newline
        // Check if the index is valid
        if (indexToChange >= 0 && indexToChange < stack.size()) {
            System.out.print("Enter the new value for the element at index " +
indexToChange + ": ");
            String newValue = scanner.nextLine();
            stack.set(indexToChange, newValue); // Replace element at the specified
index
```

```
    System.out.println("Element at index " + indexToChange + " has been changed to: " +
  newValue);
        } else {
                System.out.println("Invalid index. No changes made.");
            }
        }
    // Display the elements in the Stack after change using traditional for loop
        System.out.println("\nThe elements in the Stack after modification:");
        for (int i = 0; i < stack.size(); i++) {
            System.out.println(stack.get(i)); // Print each element by index
        }
        // Close the scanner
        scanner.close();
    }
}
```

OUTPUT

D:\PS> javac StackExample.java

D:\PS> java StackExample

How many elements do you want to push onto the stack? 3

Enter element 1: Apple

Enter element 2: Banana

Enter element 3: Cherry

The elements in the Stack are:

Apple

Banana

Cherry

Do you want to pop an element from the stack? (yes/no): yes

Element 'Cherry' has been popped.

The elements in the Stack after popping:

Apple

Banana

Do you want to change an element in the stack? (yes/no): yes

Enter the index of the element you want to change: 1

Enter the new value for the element at index 1: Orange

Element at index 1 has been changed to: Orange

The elements in the Stack after modification:

Apple

Orange

//PROGRAM TO CREATE LINKEDLIST FOR STORING AND RETRIEVING ELEMENTS

```
import java.util.LinkedList;
import java.util.Scanner;
public class LinkedListExample {
    public static void main(String[] args) {
        // Create a LinkedList of strings
        LinkedList<String> list = new LinkedList<>();
        // Create a Scanner object for user input
        Scanner scanner = new Scanner(System.in);
        // Ask user how many elements they want to add
        System.out.print("How many elements do you want to add to the LinkedList? ");
        int n = scanner.nextInt();
        scanner.nextLine(); // Consume the leftover newline
        // Store elements in the LinkedList
        for (int i = 0; i < n; i++) {
            System.out.print("Enter element " + (i + 1) + ": ");
            String element = scanner.nextLine();
            list.add(element); // Add element to the LinkedList
        }
        // Display the elements in the LinkedList using traditional for loop
        System.out.println("\nThe elements in the LinkedList are:");
        for (int i = 0; i < list.size(); i++) {
            System.out.println(list.get(i)); // Print each element by index
        }
        // Ask user for the element to remove
        System.out.print("\nEnter the element you want to remove: ");
```

```
String elementToRemove = scanner.nextLine();
    // Remove the element if it exists
    if (list.contains(elementToRemove)) {
        list.remove(elementToRemove); // Removes the first occurrence of the element
        System.out.println("Element '" + elementToRemove + "' has been removed.");
    } else {
        System.out.println("Element '" + elementToRemove + "' not found in the list.");
    }
    // Display the elements in the LinkedList after removal using traditional for loop
    System.out.println("\nThe elements in the LinkedList after removal:");
    for (int i = 0; i < list.size(); i++) {
        System.out.println(list.get(i)); // Print each element by index
    }
    // Ask user for index to update an element
    System.out.print("\nEnter the index of the element you want to change: ");
    int indexToChange = scanner.nextInt();
    scanner.nextLine(); // Consume the leftover newline
    // Check if index is valid
    if (indexToChange >= 0 && indexToChange < list.size()) {
        System.out.print("Enter the new value for the element at index " +
indexToChange + ": ");
        String newValue = scanner.nextLine();
        list.set(indexToChange, newValue); // Change the element at the specified index
        System.out.println("Element at index " + indexToChange + " has been changed
to: " + newValue);
    } else {
        System.out.println("Invalid index. No changes made.");
```

```
    }
        // Display the elements in the LinkedList after change using traditional for loop
        System.out.println("\nThe elements in the LinkedList after modification:");
        for (int i = 0; i < list.size(); i++) {
            System.out.println(list.get(i)); // Print each element by index
        }
        // Close the scanner
        scanner.close();
    }
}
```

OUTPUT

D:\PS> javac LinkedListExample.java

D:\PS> java LinkedListExample

How many elements do you want to add to the LinkedList? 3

Enter element 1: Apple

Enter element 2: Banana

Enter element 3: Cherry

The elements in the LinkedList are:

Apple

Banana

Cherry

Enter the element you want to remove: Banana

Element 'Banana' has been removed.

The elements in the LinkedList after removal:

Apple

Cherry

Enter the index of the element you want to change: 1

Enter the new value for the element at index 1: Orange

Element at index 1 has been changed to: Orange

The elements in the LinkedList after modification:

Apple

Orange

/* APPLET PROGRAM TO DISPLAY A SIMPLE MESSAGE*/

```
import java.awt.*;
import java.applet.Applet;
/* <applet code="App.class" width=200 height=100>
</applet> */
public class App extends Applet {
    public void paint(Graphics g) {
        // Draw the string "Hello world" at coordinates (10, 10)
        g.drawString("Hello world", 10, 10);
    }
}
```

HTML Code for Running the Applet (with appletviewer):

```
<html>
  <body>
    <applet code="App.class" width="200" height="100">
    </applet>
  </body>
</html>
```

OUTPUT

D:\PS> javac App.java

D:\PS> appletviewer AppletTest.html