B20EF0306

Programming in Java Laboratory Manual

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CONTENTS

	Ignore Case function as a parameter to the sort function using method reference. Print the sorted array.	
7	XYZ technologies is firm that has 5employees with 1manager, and 4 technicians. XYZ wants to digitize its payroll system, the following requirements: Dearness Allowance is 70% of basic for all employees. House Rent Allowance is 30% of basic for all employees. Income Tax is 40% of gross salary for all employees. The annual increments to the employees are to be given of the following criteria: -Manager 10% of the basic salary, and Technicians 15% of basic. Develop the pay roll for XYZ. Implement a class hierarchy using inheritance, where Employee is an abstract class and Manager and Technician are derived from Employee. Demonstrate a polymorphic behavior for giving the annual increments.	16
8	Define a new Exception class named Odd Exception. Create a new class named Even Odd. Write a method called halfOf(), which takes an int as parameter and throws an Odd Exception if the int is odd or zero, otherwise returns (int / 2). Write a main method that calls halfOf() three times (once each with an even int, an odd int,	19

	and zero), with three try/catch blocks, and prints either the output of halfOf() or the caught Odd Exception.	
9	Implement a class named Fraction that represents fractions with numerator and denominator always stored reduced to lowest terms. If fraction is negative, the numerator will always be negative, and all operations leave results stored in lowest terms. Implement the addition, subtraction, multiplication and division operation for the Fraction class and also handle divide by zero using java exception handling mechanism.	21
10	Create a class Student that has instance variables as Name, Age, Address and access transmutation methods to access the instance variables along with display method to print the details of student. Next write a main() function that will create a collection of 10students and reverse the list. Print the details before and after reversing the collection.	27
11	Use generics to build a class Sort. Implement the bubble sort algorithm to sort an array of any type.	29
12	Write a generic method to count the number of elements in a collection that have a specific property (for example, odd integers, prime numbers, palindromes).	30
	List of additional experiments	74
Viva Voice		

I.GUIDELINES TO THE STUDENTS

1. The students should have the basic knowledge of the C, JAVA language.
2. The students should have the basic knowledge of OOPs concepts.
3. This course is inter-related to theory taught in the class, please don’t miss both the class as well as labs.
4. The students have to maintain the lab observation and record book properly.
5. The students have to maintain the lab neatly.

II. LAB REQUIREMENTS

Following are the required hardware and software for this lab, which is available in the laboratory.

Hardware: Desktop system or Virtual machine in a cloud with OS installed. Presently in the Lab, Pentium IV Processor having 4 GB RAM and 500 GB Hard Disk is available.

Software: JDK(Java Development Kit) of recent version (JDK-11), Eclipse IDE, NetBeans and many more.

Simple Java Program:

```
public class FirstJavaProgram {
  public static void main(String[] args){
    System.out.println("This is my first program in java");
  }//End of main
}//End of FirstJavaProgram Class
```

Output: This is my first program in java

How to compile and run the above program

Prerequisite: You need to have java installed on your system.

Step 1: Open a text editor, like Notepad on windows and TextEdit on Mac. Copy the above program and paste it in the text editor.

You can also use IDE like Eclipse to run the java program. The following steps are used to run the program using text editor and command prompt (or terminal).

Step 2: Save the file as **FirstJavaProgram.java**. We should always name the file same as the public classname. In our program, the public class name is FirstJavaProgram, that's why our file name should be **FirstJavaProgram.java**.

Step 3: In this step, we will compile the program. For this, open **command prompt (cmd) on Windows**, if you are **Mac OS then open Terminal**.

To compile the program, type the following command and hit enter.

```
javac FirstJavaProgram.java
```

You may get this error when you try to compile the program: "**javac' is not recognized as an internal or external command, operable program or batch file**". This error occurs when the java path is not set in your system

If you get this error then you first need to set the path before compilation.

Set Path in Windows:

Open command prompt (cmd), go to the place where you have installed java on your system and locate the bin directory, copy the complete path and write it in the command like this.

```
set path=C:\Program Files\Java\jdk1.8.0_121\bin
```

Note: Your jdk version may be different.

Set Path in Mac OS X

Open Terminal, type the following command and hit return.

```
export JAVA_HOME=/Library/Java/Home
```

Type the following command on terminal to confirm the path.

```
echo $JAVA_HOME
```

That's it.

The steps above are for setting up the path temporary which means when you close the command prompt or terminal, the path settings will be lost and you will have to set the path again next time you use it. **Step 4:** After compilation the .java file gets translated into the .class file(byte code). Now we can run the program. To run the program, type the following command and hit enter:

```
java FirstJavaProgram

```

Note that you should not append the .java extension to the file name while running the program.

PROGRAM 1

Problem Statement:

The sieve of Eratosthenes is one of the most efficient ways to find all primes smaller than n when n is smaller than 10 million. Given a number n, use JAVA to print all primes smaller than or equal to n.

Expected Skill:

Understanding conditional operators and statements.

Code:

```
import java.util.*;
class prime
{
public static void main(String args[])
{
Scanner sc=new Scanner(System.in);
  System.out.println("Enter the range");
  int n=sc.nextInt();
  int a[]=new int[n];
  for(int i=1;i<=n;i++)
   {
     a[i-1]=i;
   }
  for(int i=1;i<n;i++)
    {  if(a[i]!=0)
      {for(int j=i+a[i];j<n;j+=a[i])
       {
         a[j]=0;
       }}
    }
  for(int i=0;i<n;i++)
    {
      if(a[i]!=0)
      {
        System.out.print(a[i]+"\t");
}
}
}
}
```

Expected Output:

```
Enter the range
20
1       2       3       5       7       11      13      17      19
```

PROGRAM 2

Problem Statement:

The Gauss-Jordan method is also known as Gauss-Jordan elimination method is very useful in solving a linear system of equations. It is a technique in which a system of linear equations is resolved by the means of matrices. Develop a JAVA program to solve a given set of linear equations.

Expected Skill:

Creating an array and performing some operations on array.

Code:

```
import java.util.Scanner;
public class DemoTranslation {

   public static int n;

        public static void convertToDiagonal(float[][] a, int n) {
                float ratio;
                for(int i = 0; i< n; i++) {
                        for(int j = 0; j < n; j++) {
                                if(j != i) {
                                        ratio = a[j][i] / a[i][i];
                                        for(int k = 0; k < n + 1; k++) {
                                                a[j][k] = a[j][k] - ratio * a[i][k];
                                        }
                                        System.out.println("Intermediate forms:");
                                        for(int x = 0; x < n; x++) {
                                                for(int y = 0; y < n + 1; y++) {
                                                        System.out.printf("%f ", a[x][y]);
                                                }
                                                System.out.println();
                                        }
                                        System.out.println();
                                }
                        }
                }
        }

        public static void printUnknowns(float[][] a, int n) {
                System.out.println("Values of unknowns are:");
                for(int i = 0; i< n; i++) {
                        System.out.printf("Value of Variable %d=%f\n", i, a[i][n] / a[i][i]);
                }
        }

        public static void main(String[] args) {
                int i, j, k, x, y;
                float ratio;
                System.out.println("Enter no of Unknowns");
                n = STDIN_SCANNER.nextInt();
                float[][] a = new float[n][n + 1];
                System.out.println("Enter the Augmented Matrix");
                for(int i2 = 0; i2 < n; i2++) {
```

```
                        for(int j2 = 0; j2 < n + 1; j2++) {
                                a[i2][j2] = STDIN_SCANNER.nextFloat();
                        }
                }
                convertToDiagonal(a, n);
                printUnknowns(a, n);
        }
        public final static Scanner STDIN_SCANNER = new Scanner(System.in);
}
```

Expected Output:

```
Enter  no of unknows:
3
Enter the Augmented matrix
2  1  1  10
3  2  3  18
1  4  9  16
Intermediate forms:
2.001.00 1.00 10.00
0.00 0.50 1.50 3.00
1.00 4.00 9.00 16.00
Intermediate forms:
2.00 1.00 1.00 10.00
0.00 0.50 1.50 3.00
0.003.50 8.50 11.00
Intermediate forms:
2.00 0.00 -2.00 4.00
0.00 0.50 1.50 3.00
0.00 3.50 8.50 11.00
Intermediate forms:
2.00 0.00 -2.00 4.00
0.00 0.50 1.50 3.00
0.00 0.00 -2.00 -10.00
Intermediate forms:
2.00 0.00 0.00 14.00
0.00 0.50 1.50 3.00
0.00 0.00 -2.00 -10.00
Intermediate forms:
2.00 0.00 0.00 14.00
0.00 0.50 0.00 -4.50
0.00 0.00 -2.00 -10.00

Values of unknowns are:
Value of Variable 0=7.000000
Value of Variable 1=-9.000000
Value of Variable 2=5.000000
```

PROGRAM 3

Problem Statement:

To compute a square root of any positive number a, start with an initial guess x=x1 for √a; then calculate successive approximations x2,x3...,√a using the formula:

$$x_i = \frac{x_{i-1} + (a / x_{i-1})}{2}, i = 2, 3, \ldots$$

Develop a JAVA application that implements the above SQRT function to compute the square root of any positive

Expected Skill:

Understanding conditional statements (if, if..else, etc)

Code:

```
import java.util.Scanner;
public class SquareRoot {
   public static void main(String[] args) {
      System.out.println("Enter the number:");
      Scanner sc = new Scanner(System.in);
      double a = sc.nextDouble();
      double epsilon = 1.0e-15;    // relative error tolerance ,1.0e-15 = 1.0 x 10-15 = 0.0000000000000010
      double x = a;                // estimate of the square root of c
// repeatedly apply Newton update step until desired precision is achieved
      while (Math.abs(x - a/x) > epsilon*x)
      {
         x = (x + a/x ) / 2.0;
      }
// print out the estimate of the square root of c
      System.out.println(x);
   }
}
```

Expected Output:

```
Enter the number:
81
9.0
```

PROGRAM 4

Problem Statement:

Model a lamp as a Java object. Make a Lamp class. This will contain atleast one instance variable which will be of type Boolean and will hold the state of the lamp: i.e., whether it is on or off. In addition, add methods to do the following things: switch the light on and off, and check its current state, i.e., whether it is on or off. Maintain proper encapsulation mechanism.

Next, write a launcher class with a main() method to carry out the following tasks:

- create a lamp object;
- turn it on and off;
- print the lamp's on/off status to the console.

Expected Skill:

Object and class creation and its usage

Code:

```
class Lamp {

	boolean lampOn = false;

	/**
	 * Switch the light on and off.
	 */
	void toggleLamp() {
		lampOn = !lampOn;
	}

	/**
	 * Check the status of the lamp.
	 */
	boolean getLampState() {
		return lampOn;
	}

	/**
	 * Print out the lamp's status to the console -- this is just for
	 * convenience.
	 */
	void reportStatus() {
		if (getLampState())
			System.out.println("The lamp is on.");
		else
			System.out.println("The lamp is off.");
	}

	public static void main(String[] args) {
		Lamp l = new Lamp();

		// turn the lamp on
		l.toggleLamp();
		l.reportStatus();
```

```
                // turn it off again
                l.toggleLamp();
                l.reportStatus();
        }
}
```

Expected Output:

```
The lamp is on.
The lamp is off.
The lamp is off.
The lamp is on.
```

PROGRAM 5

Problem Statement:

Given the following functional interface:

```
interface MathOperation {
int operation(int a, int b);
}
```

Develop an application that would implement the above interface using lambda expressions as to perform the addition, subtraction, multiplication and division operations.

Expected Skill:

Creation of interfaces and its usage.

Code:

```
import java.util.*;
interface Arithmetic {
        int operation(int a, int b);
}
public class LambdaExpression {
        public static void main(String[] args) {
                int x,y;
                System.out.println("Enter 2 operands:");
                Scanner in=new Scanner(System.in);
                x=in.nextInt();
                y=in.nextInt();
                // Addition using Lambda expression
                Arithmetic addition = (int a, int b) -> (a + b);
                // Arithmetic addition = (int a, int b) -> {return a + b;};
                System.out.println("Addition = " + addition.operation(x, y));

                // Subtraction using Lambda expression
                Arithmetic subtraction = (int a, int b) -> (a - b);
                // Arithmetic addition = (int a, int b) -> {return a - b;};

                System.out.println("Subtraction = " + subtraction.operation(x, y));

                // Multiplication using Lambda expression
                Arithmetic multiplication = (int a, int b) -> (a * b);
                // Arithmetic addition = (int a, int b) -> {return a * b;};
                System.out.println("Multiplication = " + multiplication.operation(x, y));

                // Division using Lambda expression
                Arithmetic division = (int a, int b) -> (a / b);
                // Arithmetic addition = (int a, int b) -> {return a * b;};
                System.out.println("Division = " + division.operation(x, y));
                }
}
```

Expected Output:

```
Enter 2 operands: 6 3
Addition = 9
Subtraction = 3
Multiplication = 18
Division = 2
```

PROGRAM 6

Problem Statement:

The String class in JAVA has a static method compare To Ignore Case, which compares two strings and the Arrays class has a static sort method. Build a JAVA program that creates an array of strings, use the sort function from Arrays class to sort the strings by passing the compare To Ignore Case function as a parameter to the sort function using method reference. Print the sorted array.

Expected Skill:

Creation of string class and its usage

Code:

```
import java.util.Arrays;

public class javapracticecode {
  public static void main(String args[]) {
String[] countries = { "norway ", "Norway", "canada", "australia", "zimbawe", "brazil", "india", "srilanka",
"England", "Netherland" };
System.out.println("Array before sorting:"+Arrays.toString(countries));
 int size = countries.length;
    for (int i = 0; i < size - 1; i++) {
      for (int j = i + 1; j < countries.length; j++) {

        if (countries[i].compareToIgnoreCase(countries[j]) > 0) {
          String temp = countries[i];
          countries[i] = countries[j];
          countries[j] = temp;
        }
      }
    }

    System.out.println("Array after sorting:"+Arrays.toString(countries));
}
}
```

Expected Output:

Array before sorting:[norway , Norway, canada, australia, zimbawe, brazil, india, srilanka, England, Netherland]

Array after sorting:[australia, brazil, canada, England, india, Netherland, Norway, norway , srilanka, zimbawe]

PROGRAM 7

Problem Statement:

XYZ technologies is firm that has 5 employees with 1 manager, and 4 technicians. XYZ wants to digitize its payroll system, the following requirements: Dearness Allowance is 70% of basic for all employees. House Rent Allowance is 30% of basic for all employees. Income Tax is 40% of gross salary for all employees. The annual increments to the employees are to be given of the following criteria: -Manager 10% of the basic salary, and Technicians 15% of basic. Develop the pay roll for XYZ. Implement a class hierarchy using inheritance, where Employee is an abstract class and Manager and Technician are derived from Employee. Demonstrate a polymorphic behavior for giving the annual increments.

Expected Skill:

Creation multiple inheritance and its usage

Code:

```
import java.lang.*;
abstract class Employee
{
        String name;
        double basic_sal;
        double da;
        double hra;
        double it;

        Employee(String n, double b)
        {
        name=n;
        basic_sal=b;
        da=basic_sal*0.7;
        hra=basic_sal*0.3;
        it=basic_sal*0.4;
        }

        abstract double gross_sal();
}

class Manager extends Employee
{
        double inc;
        Manager(String n, double b)
        {
        super(n,b);
        }

        double gross_sal()
        {
        inc=basic_sal*0.1;
        double gross=basic_sal + da + hra - it + inc;
        System.out.println("Employee Name : "+name);
        System.out.println("Designation : Manager ");
        System.out.println("Basic of Employee : "+basic_sal);
        System.out.println("DA of Employee : "+da);
```

```
        System.out.println("HRA of Employee : "+hra);
        System.out.println("IT of Employee : "+it);
        System.out.println("Annual Increment of Employee : "+inc);
        return gross;
}
}

public class Technician extends Employee
{
        double inc;
        Technician(String n, double b)
        {
        super(n,b);
        }
        double gross_sal()
        {
        inc=basic_sal*0.15;
        double gross=basic_sal + da + hra - it + inc;
        System.out.println("Employee Name : "+name);
        System.out.println("Designation : Technician ");
        System.out.println("Basic of Employee : "+basic_sal);
        System.out.println("DA of Employee : "+da);
        System.out.println("HRA of Employee : "+hra);
        System.out.println("IT of Employee : "+it);
        System.out.println("Annual Increment of Employee : "+inc);
        return gross;
        }

        public static void main(String args[])
        {
        Manager m = new Manager("anil",4000);
        Technician t1 = new Technician("Ram",500);
        Employee e;
        e=m;
        System.out.println("Gross Salary " + e.gross_sal());
         // System.out.println("Gross Salary " + m.gross_sal());
        e=t1;
        System.out.println("Gross Salary " + e.gross_sal());
        // System.out.println("Gross Salary " + t1.gross_sal());
        }
}
```

Expected Output:

```
Employee Name : anil
Designation : Manager

Basic of Employee : 4000.0
DA of Employee : 2800.0
HRA of Employee : 1200.0
IT of Employee : 1600.0
```

Annual Increment of Employee : 400.0
Gross Salary 6800.0

Employee Name : Ram
Designation : Technician
Basic of Employee : 500.0
DA of Employee : 350.0
HRA of Employee : 150.0
IT of Employee : 200.0
Annual Increment of Employee : 75.0
Gross Salary 875.0

PROGRAM 8

Problem Statement:

Define a new Exception class named Odd Exception. Create a new class named Even Odd. Write a method called halfOf(), which takes an int as parameter and throws an Odd Exception if the int is odd or zero, otherwise returns (int / 2). Write a main method that calls halfOf() three times (once each with an even int, an odd int, and zero), with three try/catch blocks, and prints either the output of halfOf() or the caught Odd Exception.

Expected Skill:

Creation of exception class and its usage

Code:

```
 import java.lang.Exception;
 import java.util.Scanner;
 class OddException extends Exception
 {
    OddException()
    {
       super("Odd number exception");
    }
    OddException(String msg)
    {
       super(msg);
    }
 }
class EvenOdd
{
          void halfOf(int num)
          {
          try
          {
          if(num%2 != 0)
             throw(new OddException());    // Statement 2
          else
          if (num == 0)
                    throw(new OddException());
          else
                              System.out.print("\n\t" + num + " is an even number and its half is:"+ (num/2));
           }
          catch(OddException Ex)
          {
          System.out.print("\n\tError : " + Ex.getMessage());
          }
          System.out.print("\n\tEnd of program");
          }
}
class EvenOddApp
{
   public static void main(String[] args)
   {
      int num;
     Scanner Sc = new Scanner(System.in);
```

```
        System.out.print("\n\tEnter any number : ");
        num = Integer.parseInt(Sc.nextLine());

        EvenOdd EO= new EvenOdd();
        EO.halfOf(num);
    }
}
```

Expected Output:

```
Enter any number : 50
50 is an even number and its half is:25
End of program

Enter any number : 5
Error : Odd number exception
End of program
```

PROGRAM 9

Problem Statement:

Implement a class named Fraction that represents fractions with numerator and denominator always stored reduced to lowest terms. If fraction is negative, the numerator will always be negative, and all operations leave results stored in lowest terms. Implement the addition, subtraction, multiplication and division operation for the Fraction class and also handle divide by zero using java exception handling mechanism.

Expected Skill:

Creation of exception class and its usage

Code:

```
 import java.util.Scanner;
 public class Fraction
 {
    // member variables
    private int numerator, denominator; // stores the fraction data
    Scanner myInput = new Scanner( System.in );
public Fraction()
{
  numerator = denominator = 0;
}
public int getNumerator()
{
  return numerator;
}
public void setNumerator(int num)
{
  numerator=num;
}
public int getDenominator()
{
  return denominator;
}
public void setDenominator(int den)
{
  denominator=den;
}
public Fraction add(Fraction b)
{
  // check preconditions
  if ((denominator == 0) || (b.denominator == 0))
     throw new IllegalArgumentException("invalid denominator"); // find
  lowest common denominator

  int common = lcd(denominator, b.denominator);

  // convert fractions to lcd
  Fraction commonA = new Fraction();
  Fraction commonB = new Fraction();
  commonA = convert(common);
  commonB = b.convert(common);
```

```
  // create new fraction to return as sum Fraction
  sum = new Fraction();

  // calculate sum
  sum.numerator = commonA.numerator + commonB.numerator;
  sum.denominator = common;

  // reduce the resulting fraction sum =
  sum.reduce();

  return sum;
}
public Fraction subtract(Fraction b)
{
  // check preconditions

  if ((denominator == 0) || (b.denominator == 0))

    throw new IllegalArgumentException("invalid denominator"); // find
  lowest common denominator

  int common = lcd(denominator, b.denominator);

  // convert fractions to lcd
  Fraction commonA = new Fraction();
  Fraction commonB = new Fraction();
  commonA = convert(common);
  commonB = b.convert(common);

  // create new fraction to return as difference Fraction
  diff = new Fraction();

  // calculate difference
  diff.numerator = commonA.numerator - commonB.numerator;
  diff.denominator = common;

  // reduce the resulting fraction diff =
  diff.reduce();
  return diff;

}
public Fraction multiply(Fraction b)
{

  // check preconditions
  if ((denominator == 0) || (b.denominator == 0))

    throw new IllegalArgumentException("invalid denominator");

  // create new fraction to return as product Fraction
```

```
  product = new Fraction();
  //
  // calculate product
  product.numerator = numerator * b.numerator;
  product.denominator = denominator * b.denominator;

  // reduce the resulting fraction product =
  product.reduce(); return product;

}
public Fraction divide(Fraction b)

{
  // check preconditions

  if ((denominator == 0) || (b.numerator == 0))

    throw new IllegalArgumentException("invalid denominator");

  // create new fraction to return as result Fraction
  result = new Fraction();

  // calculate result
  result.numerator = numerator * b.denominator;
  result.denominator = denominator * b.numerator;

//    reduce the resulting fraction result =
result.reduce(); return result;
}
public void input()
{
  // prompt user to enter numerator
  System.out.print("Please enter an integer for numerator: "); // get user
  input
  numerator = myInput.nextInt();

  // prompt user to enter denominator in a loop to prevent
  // an invalid (zero) value for denominator

  do {

    System.out.print("Please enter a non-zero integer for denominator: ");
    denominator = myInput.nextInt();

    // make sure it is non-zero if
    (denominator == 0)

      System.out.println("Invalid value. Please try again."); } while
  (denominator == 0);
}
```

```
public void output()
{
    System.out.print(this);
}
public String toString()
{
   String buffer = numerator + "/" + denominator;
   return buffer;
}
private int lcd(int denom1, int denom2)
{
   int factor = denom1;
   while ((denom1 % denom2) != 0)
      denom1 += factor;
   return denom1;
}
private int gcd(int denom1, int denom2)
{
            int factor = denom2;
            while (denom2 != 0) {
               factor = denom2;
denom2 = denom1 % denom2;
      denom1 = factor;
   }
   return denom1;
   }
private Fraction convert(int common)
{
   Fraction result = new Fraction();
   int factor = common / denominator;
   result.numerator = numerator * factor;
   result.denominator = common;
   return result;
}
private Fraction reduce()
{
   Fraction result = new Fraction();
   int common = 0;
   // get absolute values for numerator and denominator int num =
   Math.abs(numerator);

   int den = Math.abs(denominator);

   // figure out which is less, numerator or denominator if (num >
   den)

      common = gcd(num, den); else if
   (num < den)

      common = gcd(den, num);
```

```
  else // if both are the same, don't need to call gcd common =
     num;
  // set result based on common factor derived from gcd
  result.numerator = numerator / common; result.denominator =
  denominator / common; return result;

}
public static void main(String args[])
{
 Fraction f1 = new Fraction();
 Fraction f2 = new Fraction();
  // one way to set up fractions is simply to hard-code some values
  f1.setNumerator(1);
  f1.setDenominator(3);
  f2.setNumerator(1);
  f2.setDenominator(6);


  // try some arithmetic on these fractions
  // test addition result =
  f1.add(f2);

  // one way to output results, using toString method directly
  System.out.println(f1 + " + " + f2 + " = " + result);

  // test addition going the other way - should be same result result =
  f2.add(f1);

  // output results

  System.out.println(f2 + " + " + f1 + " = " + result);
  System.out.println();


  // test subtraction
  result = f1.subtract(f2);

  // output results

  System.out.println(f1 + " - " + f2 + " = " + result);

  // test subtraction going the other way - should be different result result =
  f2.subtract(f1);

  // output results

  System.out.println(f2 + " - " + f1 + " = " + result);
```

```
  // another way to set up fractions is to get user input
  System.out.println(); System.out.println("Fraction 1:");

  f1.input(); System.out.println();
  System.out.println("Fraction 2:"); f2.input();
  System.out.println();



  // test multiplication
  result = f1.multiply(f2);


  // another way to output results is to use the output method
  // this uses the toString method indirectly

  f1.output();
  System.out.print(" * ");
  f2.output();
  System.out.print(" = ");
  result.output();
  System.out.println();

  // test division

  result = f1.divide(f2);
      // output results f1.output();
      System.out.print(" / ");
      f2.output(); System.out.print("
      = "); result.output();
      System.out.println();

    }
}
```

PROGRAM 10

Problem Statement:

Create a class Student that has instance variables as Name, Age, Address and access transmutation methods to access the instance variables along with display method to print the details of student. Next write a main() function that will create a collection of 10 students and reverse the list. Print the details before and after reversing the collection.

Expected Skill:

Object and class creation and its usage

Code:

```
import java.util.*;
class Student    {
private int roll;
private String name;
private int marks;
public void setRoll(int roll)   {
this.roll = roll;
}
public int getRoll(){
return roll;
}
public void setName(String name){
this.name = name;
}
public String getName(){
return name;
}
public void setMarks(int marks){
this.marks = marks;
}
public int getMarks(){
return marks;
}
}

public class SortStudents
{
public static void main(String args[])  {
List stu= new ArrayList();
```

```
Student st1= new Student();
st1.setRoll(101);
st1.setName("Amit");
st1.setMarks(56);
Student st2= new Student();
st2.setRoll(103);
st2.setName("Anil");
st2.setMarks(66);
Student st3= new Student();
st3.setRoll(103);
st3.setName("Ankit");
st3.setMarks(76);
stu.add(st1);
stu.add(st2);
stu.add(st3);
ListIterator listItr =(ListIterator)stu.listIterator();
while(listItr.hasNext())   {
Student stud =(Student)listItr.next();
System.out.print(" "+stud.getRoll());
System.out.print(" "+stud.getName());
System.out.println(" "+stud.getMarks());
}
while(listItr.hasPrevious()) {
Student stud =(Student)listItr.previous();
System.out.print(" "+stud.getRoll());
System.out.print(" "+stud.getName());
System.out.println(" "+stud.getMarks());
}
}
}
```

PROGRAM 11

Problem Statement:

Use generics to build a class Sort. Implement the bubble sort algorithm to sort an array of any type.

Expected Skill:

Creation of generics class and its usage

Code:

```
import java.util.Arrays;
public class BubbleSortGeneric<T extends Comparable<? super T>> {

  T[] array;
 BubbleSortGeneric(T[] array){
   this.array = array;
 }

 private T[] bubbleSort(){
   for(int i = array.length; i > 1; i--){
     for(int j = 0; j < i - 1; j++){
       //if greater swap elements
       if(array[j].compareTo(array[j+1]) > 0){
         swapElements(j, array);
       }
     }
   }
   return array;
 }
 private void swapElements(int index, T[] arr){
   T temp = arr[index];

   arr[index] = arr[index+1];
   arr[index+1] = temp;
 }
 public static void main(String[] args) {
   Integer[] intArr = {47, 85, 62, 34, 7, 10, 92, 106, 2, 54};

   BubbleSortGeneric<Integer> bsg1 = new BubbleSortGeneric<Integer>(intArr);
   Integer[] sa1 = bsg1.bubbleSort();
   System.out.println("Sorted array- " + Arrays.toString(sa1));

   String[] strArr = {"Earl", "Robert", "Asha", "Arthur"};
   BubbleSortGeneric<String> bsg2 = new BubbleSortGeneric<>(strArr);
   String[] sa2 = bsg2.bubbleSort();
   System.out.println("Sorted array- " + Arrays.toString(sa2));
 }
}
```

PROGRAM 12

Problem Statement:

Write a generic method to count the number of elements in a collection that have a specific property (for example, odd integers, prime numbers, palindromes).

Expected Skill:

Creation of generics class and its usage

Code:

```
import java.util.ArrayList;

public class UpperBoundWildcard {

   private static Double add(ArrayList<? extends Number> num) { double
      sum=0;

      double den =2.0;

      for(Number n:num)

      {
         if((n.doubleValue()%den)!=0)

         { sum = sum+n.doubleValue();
            System.out.println(n.doubleValue());
         }

      }

      return sum;
   }

   public static void main(String[] args) {

      ArrayList<Integer> l1=new ArrayList<Integer>();
      l1.add(10);
      l1.add(20);
      l1.add(21);
      l1.add(35);
      l1.add(42);
      l1.add(55);
      System.out.println("displaying the sum= "+add(l1));

   }

}
```