



REVA
UNIVERSITY

JAVA Programming Laboratory Manual

Second Year

Batch 2023-2027

**SCHOOL OF COMPUTING
AND
INFORMATION TECHNOLOGY**

NAME	
SRN	
SEM	
SECTION	

BRANCH	
ACADEMIC YEAR	

CONTENTS

S.NO	PROGRAMS	Page No.
I	Course description	3
II	Course outcomes	3
III	Course objectives	3
IV	Guidelines to students	3
V	Lab requirements	4

PROGRAMS TO BE PRACTISED

1.	<p>Write a java program to create a console application that allows the user to choose an arithmetic operation able to provide his choice of operands for the same. Display the appropriate output. Note: Use switch statement to perform the operations.</p> <p>Aim: Implementation of Simple Calculator</p>	
2	<p>A String is a collection of characters, a given string can be a combination of vowels and consonants. Develop a java program to count the number of vowels, special characters, numbers, white spaces and consonants in a string.</p> <p>Aim: Count Vowels, Consonants, special characters, numbers, white spaces in a given string.</p>	
3.	<p>Using a one dimensional array, read an array of integer elements and perform the following operations.</p> <p>a. Copy all elements from one array to another.</p> <p>b. Remove duplicate elements from array and print only even position of array.</p>	

4.	<p>Develop a JAVA program to write an application to create student database to input name, SRN and college name, where college name should be declared as static variable and perform the following tasks.</p> <p>Create a class with static variable.</p> <p>Insert some values into the members of the class including static member and display the values of the members.</p> <p>Change the value of static variable and display the updated values of the members.</p> <p>Aim: Implement Static Variable for a Student database</p>	
5.	<p>Write a Java program that implements the below functionality using constructor overloading.</p> <p>Define a Student class with the attributes name (String), rollNumber (int), marks (double). Implement three constructors with different parameters:</p> <ol style="list-style-type: none"> 1. A constructor that initializes only the name, with default values (rollNumber = 0 (or "Not Assigned"), marks = 0.0. 2. A constructor that initializes the name and roll number, with default marks = 0.0. 3. A constructor that initializes the name, roll number, and marks. <p>Implement a method displayStudentDetails() that prints the student's information. In the main method, create three student objects using different constructors and display their details</p>	
6	<p>A child inherits the features of parents and also develops its own personality as it grows up in the society, over a period of time. This situation can be represented by the concept of multilevel inheritance in java programming. Apply the same concept in the car manufacturing scenario in your own terms.</p> <p>Aim: Construct multilevel inheritance</p>	
7	<p>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</p>	

	<p>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.</p> <p>Aim: Develop multiple inheritance using Interfaces</p>	
8	Write a JAVA Program to handle multiple exceptions using Nested Try block	
9	Develop a Banking System that allows users to withdraw money from their bank accounts. However, if the user tries to withdraw an amount greater than the available balance, the system should throw a custom exception called InsufficientBalanceException.	
10.	Write a program to create a simple generic class, where T is a type parameter that will be replaced by a real type, when an object of type Gen class is created.	
11.	Write a Java program that implements a multi-thread application that has three threads. First thread generates a random integer for every 1 second; second thread computes the square of the number and prints; third thread will print the value of cube of the number.	
Viva Voice		

I. COURSE DESCRIPTION

This laboratory course supplements the material taught in the theory course programming with JAVA. The objective of this course is to get hands-on experience in JAVA programming and implementing the techniques learnt in the theory course. Laboratory exercises will normally be conducted using Windows Operating system. The students will be exposed to basic syntax of classes, objects and data structure operations using JAVA.

II. LAB OBJECTIVES

The objectives of this lab course are to make students to;

1. Practice basic and fundamental object oriented programming concepts.
2. Use constructors and destructors to solve any real world problems.
3. Write and practice java program that contains classes and objects.
4. Write and practice java program that implements data structure operations.

III. LAB OUTCOMES

After successful completion of this lab course students shall be able to:

1. Design and develop java programs for solving simple problems.
2. Compile and debug programs in java language.
3. Design and develop programs using all OOP concepts
4. Develop complex applications to address needs of society, industry and others.

IV. GUIDELINES TO THE STUDENTS

1. The students should have the basic knowledge of the C, JAVA language.
2. This course is inter-related to theory taught in the class, please don't miss both the class as well as labs.

V. 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 1 GB RAM and 250 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 - Implementation of Simple Calculator

1 Problem Statement

Develop a calculator that allows you to easily handle all the calculations necessary for everyday life with a single application. Write a JAVA program using switch statement to design a basic calculator that performs basic operations.

2 Student Learning Outcomes

After successful completion of this lab, the student's shall be able to

- Identify basic operations that is performed using basic calculator.
- Apply switch statement to perform the calculations efficiently

3 Design of the Program

3.1 Description

In this Program we are making a simple calculator that performs addition, subtraction, multiplication and division based on the user input. The program takes the value of both the numbers (entered by user) and then user is asked to enter the operation (+, -, * and /), based on the input program performs the selected operation on the entered numbers using switch case.

3.2 Algorithm

Step 1: START

Step 2: Read the first num, second num and operator

Step 3: Perform required operation based on operator entered in STEP 2.

switch(operator)

case '+' \leftarrow output = num1 + num2; go to STEP 4

case '-' \leftarrow output = num1 – num2; go to STEP 4

case '*' \leftarrow output = num1 * num2; go to STEP 4

case '/' \leftarrow output = num1 / num2; go to STEP 4

default \leftarrow print "You have entered wrong operator"

END switch

Step 4: print num1,operator,num2,output

Step 5: END

3.3 Coding using JAVA Language

```
1. import java.util.Scanner;
2. public class JavaExample {
3.     public static void main(String[] args) {
4.         double num1, num2;
5.         Scanner scanner = new Scanner(System.in);
6.         System.out.print("Enter first number:");
7.         num1 = scanner.nextDouble();
8.         System.out.print("Enter second number:");
9.         num2 = scanner.nextDouble();
10.        System.out.print("Enter an operator (+, -, *, /): ");
11.        char operator = scanner.next().charAt(0);
12.        scanner.close();
13.        double output=0.0;
14.        switch(operator)
15.        {
16.        case '+':  output = num1 + num2;    break;
17.        case '-':  output = num1 - num2;    break;
18.        case '*':  output = num1 * num2;    break;
19.        case '/':  output = num1 / num2;    break;
20.        default:  System.out.printf("You have entered wrong operator");    return;
21.        }
22.        System.out.println(num1+" "+operator+" "+num2+": "+output);
23.    }
24. }
```

3.4 OUTPUT

Enter first number:40

Enter second number:4

Enter an operator (+, -, *, /): /

40.0 / 4.0: 10.0

Enter first number:4

Enter second number:8

Enter an operator (+, -, *, /): -

4.0 / 8.0: -4.0

Enter first number:2

Enter second number:3

Enter an operator (+, -, *, /): +

2.0 / 3.0: 5.0

Enter first number:6

Enter second number:2

Enter an operator (+, -, *, /): *

6.0 / 2.0: 12.0

Enter first number:7

Enter second number:4

Enter an operator (+, -, *, /): %

You have entered wrong operator

PROGRAM 2 – counting vowels, consonants, special characters, white spaces and numbers	
1	Problem Statement
<p>A String is a collection of characters, a given string can be a combination of vowels and consonants. Develop a java program to count the number of vowels, special characters, numbers, white spaces and consonants in a string.</p>	
2	Student Learning Outcomes
<p>After successful completion of this lab, the student shall be able to</p> <ul style="list-style-type: none"> • Convert uppercase characters to lowercase characters • Compare the characters in a string with all the vowels 	
3	Design of the Program
3.1	Description
<p>In this program, our task is to count the total number of vowels and consonants present in the given string.</p> <p>As we know that, the characters a, e, i, o, u are known as vowels in the English alphabet. Any character other than that is known as the consonant.</p> <p>To solve this problem, First of all, we need to convert every upper-case character in the string to lower-case so that the comparisons can be done with the lower-case vowels only not upper-case vowels, i.e.(A, E, I, O, U). Then, we have to traverse the string using a for or while loop and match each character with all the vowels, i.e., a, e, i, o, u. If the match is found, increase the value of count by 1 otherwise continue with the normal flow of the program.</p>	
3.2	Algorithm
<p>Step 1: START</p> <p>Step 2: SET vCount =0, cCount =0</p> <p>Step 3: DEFINE string str = "This is a really simple sentence".</p> <p>Step 4: CONVERT str to lowercase</p> <p>Step 5: SET i =0.</p> <p>Step 6: REPEAT STEP 6 to STEP 8 UNTIL i<str.length()</p>	

Step 7: IF any character of str matches with any vowel then

vCount = vCount + 1.

Step 8: IF any character excepting vowels lies BETWEEN a and z then

cCount = cCount +=1.

Step 9: IF any character str matches with digit then

d+=1

Step 10: IF any character str matches with space

space= + 1

Step 11: IF any character str matches with special character

Sc+=1

PRINT vCount.

Step 11: PRINT cCount, vCount, digit, space, special character

Step 12: END

3.3 Coding using JAVA Language

```
1. import java.util.*;
2. public class CountC
3. {
4.     public static void main(String[] args)
5.     {
6.         Scanner s=new Scanner(System.in);
7.         System.out.println("Enter the word to count Vowels and Alphabets in it");
8.         String str = s.nextLine();
9.         //Counter variable to store the count of vowels, consonant, digits, spaces, special characters
10.        int vCount = 0, cCount = 0,d=0,spaces=0,sc=0;
11.        //Converting entire string to lower case to reduce the comparisons
12.        str = str.toLowerCase();
13.        for(int i = 0; i < str.length(); i++) {
14.            char ch = str.charAt(i);
15.            if(ch == 'a' || ch== 'e' || ch== 'i' || ch == 'o' || ch == 'u') {
16.                vCount++;
17.            }
```

```

18.         else if(ch >= 'a' && ch<='z') {
19.             cCount++;
20.         }
21.         else if(ch>= '0' && ch <= '9'){
22.             d++;
23.         }
24.         else if(ch== ' '){
25.             spaces++;
26.         }
27.         else {
28.             sc++;
29.         }
30.     }
31.     System.out.println("Number of vowels: " + vCount);
32.     System.out.println("Number of consonants: " + cCount);
33.     System.out.println("Number of digits: " + d);
34.     System.out.println("Number of spaces: " + spaces);
35.     System.out.println("Number of spec: " + sc);
36.     }
37.     }

```

4	Expected Results
----------	-------------------------

```

F:\JAVA>javac CountC.java
F:\JAVA>java CountC
Enter the word to count Vowels and Alphabets in it
reva uni 2034 *
Number of vowels: 4
Number of consonants: 3
Number of digits: 4
Number of spaces: 3
Number of spec: 1

```

PROGRAM 3A - Copy Elements of One Array to Another	
1	Problem Statement
Develop a JAVA Program to copy the elements of One Array to Another.	
2	Student Learning Outcomes
After successful completion of this lab, the student shall be able to <ul style="list-style-type: none"> • Demonstrate how the data can be stored in contiguous memory locations using arrays. • Initialize an array and copy the elements of one array to another array. 	
3	Design of the Program
3.1	Description
<p>Arrays:</p> <p>An array, in the context of Java, is a dynamically-created object that serves as a container to hold constant number of values of the same type. By declaring an array, memory space is allocated for values of a particular type. At the time of creation, the length of the array must be specified and remains constant.</p> <p>To access an array element, the numerical index (a non-negative value) corresponding to the location of that element must be used. The first index value in the array is zero, thus, the index with value four is used to access the fifth element in the array. An array element that is also an array is known as a subarray. Arrays could have one or two dimensions. We initialize an array and copy the elements of one array to another array.</p>	
3.2	Algorithm
<p>Step 1: START</p> <p>Step 2: INITIALIZE arr1[] $\leftarrow \{1, 2, 3, 4, 5\}$</p> <p>Step 3: CREATE arr2[] of size arr1[].</p> <p>Step 4: COPY elements of arr1[] to arr2[]</p> <p>Step 5: REPEAT STEP 6 UNTIL (i<arr1.length)</p> <p>Step 6: arr2[i] \leftarrow arr1[i]</p> <p>Step 7: DISPLAY elements of arr1[].</p> <p>Step 8: REPEAT STEP 9 UNTIL (i<arr1.length)</p>	

Step 9: PRINT arr1[i]

Step 10: DISPLAY elements of arr2[].

Step 11: REPEAT STEP 12 UNTIL (i<arr2.length)

Step 12: PRINT arr2[i].

Step 13: END

3.3 Coding using JAVA Language

```
1. public class CopyArray {
2.     public static void main(String[] args) {
3.         //Initialize array
4.         int [] arr1 = new int [] {1, 2, 3, 4, 5};
5.         //Create another array arr2 with size of arr1
6.         int arr2[] = new int[arr1.length];
7.         //Copying all elements of one array into another
8.         for (int i = 0; i < arr1.length; i++) {
9.             arr2[i] = arr1[i];
10.        }
11.        //Displaying elements of array arr1
12.        System.out.println("Elements of original array: ");
13.        for (int i = 0; i < arr1.length; i++) {
14.            System.out.print(arr1[i] + " ");
15.        }
16.        System.out.println();
17.        //Displaying elements of array arr2
18.        System.out.println("Elements of new array: ");
19.        for (int i = 0; i < arr2.length; i++) {
20.            System.out.print(arr2[i] + " ");
21.        }
22.    }
23. }
```

4	Expected Results
<pre> C:\>cd shivu C:\shivu>javac CopyArray.java C:\shivu>java CopyArray Elements of original array: 1 2 3 4 5 Elements of new array: 1 2 3 4 5 </pre>	

PROGRAM 3B - Removing Duplicate Elements from Array	
1	Problem Statement
Develop a JAVA Program to remove the duplicate elements from the array and print only the elements present in even position of the array.	
2	Student Learning Outcomes
<p>After successful completion of this lab, the student shall be able to</p> <ul style="list-style-type: none"> • Remove the duplicate elements present in the array. • Display the elements present only in the even position in the array. 	
3	Design of the Program
3.1	Description
<ul style="list-style-type: none"> • In the array we can have duplicate elements i.e. an element can be repeated more than one time in the array. We can remove the duplicate elements from the array and only print the array by displaying the element only once. • Create a new array temp with same size as original array. • Iterate over array starting from index location 0. • Match current element with next element indexes until mismatch is found. • Add element to temp and make current element to element which was mismatched. • Continue the iteration. • Using for loop print only the elements present in even position. 	
3.2	Algorithm

Step 1: START

Step 2: Return, if array is empty or contains a single element.

Step 3: Start traversing elements.

Step 4: If current element is not equal to next element then store that current element.

Step 5: Store the last element as whether it is unique or repeated, it hasn't stored previously.

Step 6: Modify original array.

Step 7: removing the duplicates and returns new size of array.

Step 8: Print the elements present in even position of the array.

Step 9: STOP

3.3

Coding using JAVA Language

```
import java.util.*;
class p3b
{
public static void main(String[] args) {
    int A[]=new int[10];
    int B[]=new int [10];
    int n, i, j, k = 0;
    Scanner s=new Scanner(System.in);
    System.out.println("Enter size of array : ");
    n=s.nextInt();
    System.out.println( "Enter elements of array : ");
    for (i = 0; i < n; i++)
        A[i]=s.nextInt();
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < k; j++)
        {
            if (A[i] == B[j])
                break;
```

```

    }
    if (j == k)
    {
        B[k] = A[i];
        k++;
    }
}
System.out.println( " elements after deletion : ");
for (i = 0; i < k; i++){
    A[i]=B[i];
    System.out.println( A[i] );
}
}
}

```

4	Expected Results
----------	-------------------------

```

C:\Jlab>java p3b
Enter size of array :
5
Enter elements of array :
12
13
13
14
15
elements after deletion :
12
13
14
15

```

OR

```
//deletion of duplicate elements using single array
import java.util.*;
class p3b
{
public static void main(String[] args) {
    int a[]=new int[10];
    int n, i, j, k = 0;
    Scanner s=new Scanner(System.in);
    System.out.println("Enter size of array : ");
    n=s.nextInt();
    System.out.println( "Enter elements of array : ");
    for (i = 0; i < n; i++)
        a[i]=s.nextInt();
    for(i=0;i<n;i++){
        for(j = i+1; j < n; j++){
            if(a[i] == a[j]){
                for(k = j; k < n; k++){
                    a[k] = a[k+1];
                }
                j--;
                n--;
            }
        }
    }
    System.out.println( " elements after deletion : ");
    for (i = 0; i < n; i++){
        System.out.println( a[i] );
    }
}
```

Expected output

```
D:\lks>javac p3n.java
```

```
D:\lks>java p3n
```

```
Enter size of array :
```

```
5
```

```
Enter elements of array :
```

```
11
```

```
12
```

```
23
```

```
11
```

```
12
```

```
Entered element are:
```

```
11
```

```
12
```

```
23
```

```
11
```

```
12
```

```
After deleting the duplicate element the Array is:
```

```
11
```

```
12
```

```
23
```

```
D:\lks>
```

PROGRAM 4 : Student Database	
1	Problem Statement
Develop a JAVA program to write an application to create student database to input name, SRN and college name, where college name should be declared as static variable	
2	Student Learning Outcomes
<p>After successful completion of this lab, the student shall be able to</p> <ul style="list-style-type: none"> • Create a class with static variable. • Insert some values into the members of the class including static member and display the values of the members. • Change the value of static variable and display the updated values of the members. 	
3	Design of the Program
3.1	Description
<p>Static Variables:</p> <p>When a variable is declared as static, then a single copy of variable is created and shared among all objects at class level. Static variables are, essentially, global variables. All instances of the class share the same static variable.</p> <p>Important points for static variables:</p> <ul style="list-style-type: none"> • We can create static variables at class-level only. • Static block and static variables are executed in order they are present in a program. • It is a variable which belongs to the class and not to object(instance). • Static variables are initialized only once, at the start of the execution. These variables will be initialized first, before the initialization of any instance variables. • A single copy to be shared by all instances of the class. • A static variable can be accessed directly by the class name and doesn't need any object. 	
3.2	Algorithm
<p>Step 1: START</p> <p>Step 2: Create a class Student with members SRN, name and static variable collegeName.</p> <p>Step 3: Initialize some values to the members of the class Student including static variable collegeName.</p>	

Step 4: Display the members of the class Student.

Step 5: Change the value of the static variable collegeName.

Step 6: Display the updated values of the members of class Student.

Step 7: STOP

3.3 Coding using JAVA Language

```
1. class Student
2. {
3.   String SRN;
4.   String name;
5.   static String collegeName; //static variable
6.   public static void main(String[] args)
7.   {
8.       //create 3 object which will share collegeName value
9.       Student s1= new Student();
10.      Student s2= new Student();
11.      Student s3= new Student();
12.      //assign value to static variable collegeName
13.      Student.collegeName="REVA UNIVERSITY";
14.      //assign values to instance variables
15.      s1.SRN="R18CS001";
16.      s1.name="stud1";
17.      s2.SRN="R18CS002";
18.      s2.name="stud2";
19.      s3.SRN="R18CS003";
20.      s3.name="stud3";
21.      //Print the values of the objects
22.      System.out.println("S1 SRN.= "+s1.SRN+" S1 Name= "+s1.name+" S1 College Name= "+s1.collegeName );
23.      System.out.println("S2 SRN.= "+s2.SRN+" S2 Name= "+s2.name+" S2 College Name= "+s2.collegeName );
```

```

20. System.out.println("S3 SRN.= "+s3.SRN+" S3 Name= "+s3.name+" S3 College Name=
    "+s3.collegeName );

    //if one object change the value of static variable then it will reflect into all objects
21. s2.collegeName="REVA";
22. s2.name="JAMES";

    //Print the values of the objects after change
23. System.out.println("S1 SRN.= "+s1.SRN+" S1 Name= "+s1.name+" S1 College Name=
    "+s1.collegeName );
24. System.out.println("S2 SRN.= "+s2.SRN+" S2 Name= "+s2.name+" S2 College Name=
    "+s2.collegeName );
25. System.out.println("S3 SRN.= "+s3.SRN+" S3 Name= "+s3.name+" S3 College Name=
    "+s3.collegeName );
26. }
27. }

```

4	Expected Results
----------	-------------------------

```

C:\shivu>javac Student.java

C:\shivu>java Student
S1 SRN.= R18CS001 S1 Name= stud1 S1 College Name= REVA UNIVERSITY
S2 SRN.= R18CS002 S2 Name= stud2 S2 College Name= REVA UNIVERSITY
S3 SRN.= R18CS003 S3 Name= stud3 S3 College Name= REVA UNIVERSITY
S1 SRN.= R18CS001 S1 Name= stud1 S1 College Name= REVA
S2 SRN.= R18CS002 S2 Name= JAMES S2 College Name= REVA
S3 SRN.= R18CS003 S3 Name= stud3 S3 College Name= REVA

C:\shivu>_

```

Student data base using array of objects

```
import java.util.Scanner;

class student
{
    int roll_no;
    String name;
    static String cname="REVA";
    Scanner s=new Scanner(System.in);
    void read()
    {
        name=s.nextLine();
        roll_no=s.nextInt();
    }
    void display()
    {
        System.out.print("Student roll number=" + roll_no);
        System.out.print("Student name=" + name);
        System.out.print(" College name=" + cname);
    }
    public static void main(String args[])
    {
        Scanner s=new Scanner(System.in);
        System.out.println("Enter no of students");
        int n=s.nextInt();
        Student1 a[]=new Student1[n];
        for(int i=0;i<n;i++) {
            a[i]=new Student1();
            System.out.println("Enter name and srn of student " + (i+1));
            a[i].read();
        }
        for(int i=0;i<n;i++)
```



```
{  
System.out.println("\ndetails of " + (i+1));  
a[i].display();  
}  
}  
}
```

Expected Results

D:\lks>javac student.java

D:\lks>java student

Enter no of students

4

Enter name and srn of student 1

aryan

1

Enter name and srn of student 2

bharath

2

Enter name and srn of student 3

chanakya

3

Enter name and srn of student 4

Druva

4

details of 1

Student roll number=1

Student name=aryan

College name=REVA

details of 2

Student roll number=2

Student name=bharath

College name=REVA

details of 3

Student roll number=3

Student name=chanakya

College name=REVA

details of 4

Student roll number=4

Student name=Druva

College name=REVA

D:\lks>

PROGRAM 5 – To calculate volume of a box	
1	Problem Statement
	<p>Write a Java program that implements the below functionality using constructor overloading.</p> <p>Define a Student class with the attributes name (String), rollNumber (int), marks (double).</p> <p>Implement three constructors with different parameters:</p> <ol style="list-style-type: none"> 1. A constructor that initializes only the name, with default values (rollNumber = 0 (or "Not Assigned"), marks = 0.0. 2. A constructor that initializes the name and roll number, with default marks = 0.0. 3. A constructor that initializes the name, roll number, and marks. <p>Implement a method displayStudentDetails() that prints the student's information. In the main method, create three student objects using different constructors and display their details</p>
2	Student Learning Outcomes
	<p>After successful completion of this lab, the student's shall be able to</p> <ul style="list-style-type: none"> • Identify the classes, data and function members in each class • Implement the construction overloading
3	Design of the Program
3.1	Description
	<p>Parameterised constructor:</p> <p>Syntax:</p> <pre> class(keyword) class_name(user-defined) { Data_type variable name; Return_type function_name(parameters); } </pre>
3.2	Algorithm
	<p>Algorithm: Student Class</p> <p>Step 1 : START</p> <p>Step 2 : Define a class named Student and private attributes name, rollNumber, marks</p>

Step 3 : Create a constructor that takes a string parameter name.

Step 4 : Create a constructor that takes two parameters: name (string) and rollNumber (integer).

Step 5: Create a constructor that takes three parameters: name (string), rollNumber (integer), and marks (double).

Step 6 : Create a method named displayStudentDetails.

Step 7: Create objects of the Student class using different constructors.

Step 8: Call the displayStudentDetails method for each object to print their details.

Step 9: Stop

3.3 Coding using java Language

Program Description:

```
class Student {  
  
    private String name;  
    private int rollNumber;  
    private double marks;  
  
    // Constructor 1: Only Name (Default rollNumber = 0, Default marks = 0.0)  
    public Student(String name) {  
        this.name = name;  
        this.rollNumber = 0;  
        this.marks = 0.0;  
        System.out.println("Student created: " + name + " (Roll: Not Assigned, Marks: 0.0)");  
    }  
  
    // Constructor 2: Name and Roll Number  
    public Student(String name, int rollNumber) {  
        this.name = name;  
        this.rollNumber = rollNumber;  
    }  
}
```

```

    this.marks = 0.0; // Default marks
    System.out.println("Student created: " + name + " (Roll: " + rollNumber + ", Marks: 0.0)");
}

// Constructor 3: Name, Roll Number, and Marks
public Student(String name, int rollNumber, double marks) {
    this.name = name;
    this.rollNumber = rollNumber;
    this.marks = marks;
    System.out.println("Student created: " + name + " (Roll: " + rollNumber + ", Marks: " + marks + ")");
}

// Display student details
public void displayStudentDetails() {
    System.out.println("\nStudent Name: " + name);
    System.out.println("Roll Number: " + (rollNumber == 0 ? "Not Assigned" : rollNumber));
    System.out.println("Marks: " + marks);
}

public static void main(String[] args) {
    // Creating objects using different constructors
    Student s1 = new Student("Alice");
    s1.displayStudentDetails();

    Student s2 = new Student("Bob", 101);
    s2.displayStudentDetails();

    Student s3 = new Student("Charlie", 102, 85.5);
    s3.displayStudentDetails();
}
}

```

3.4	Expected Results
<p>F:\JAVA>javac Student.java</p> <p>F:\JAVA>java Student</p> <p>Student created: Alice (Roll: Not Assigned, Marks: 0.0)</p> <p>Student Name: Alice</p> <p>Roll Number: Not Assigned</p> <p>Marks: 0.0</p> <p>Student created: Bob (Roll: 101, Marks: 0.0)</p> <p>Student Name: Bob</p> <p>Roll Number: 101</p> <p>Marks: 0.0</p> <p>Student created: Charlie (Roll: 102, Marks: 85.5)</p> <p>Student Name: Charlie</p> <p>Roll Number: 102</p> <p>Marks: 85.5</p> <p>Output:</p> <p>Volume is 3000.0</p> <p>Volume is 162.0</p>	
3.5	Implementation Phase : Execute the Program Compile, remove syntax errors, if any, generate object code and make note of the same.

PROGRAM 6 – Multilevel Inheritance	
1	Problem Statement
<p>A child inherits the features of parents and also develops its own personality as it grows up in the society, over a period of time. This situation can be represented by the concept of multilevel inheritance in java programming. Apply the same concept in the car manufacturing scenario in your own terms.</p> <p>Note: Student should identify the classes, data and function members in each class and write the program.</p>	
2	Student Learning Outcomes
<p>After successful completion of this lab, the student's shall be able to</p> <ul style="list-style-type: none"> • Identify the classes, data and function members in each class • Implement multilevel inheritance concept • Use variables & functions of parent class in child class • Understand how to reuse the existing code and increase the program efficiency 	
3	Design of the Program
3.1	Description
<p>Multilevel inheritance:</p> <p>A class represents a real world entity. A class' properties can be used by another class by inheriting them using the concept of multilevel inheritance. This feature in java helps to reduce the size of the code and increases the reusability thus improving the efficiency of programs significantly.</p> <p>Syntax:</p> <pre> class(keyword) class_name-1(user-defined) { } Class class_name-2: extends class_name-1 { } </pre>	
3.2	Algorithm
<p>Step 1 : START</p> <p>Step 2 : declare 3 classes that represents a real world entity: car, ferrari and ferrariSF90</p>	

Step 3 : declares variables of all the classes: car,ferrari and ferrariSF90

Step 4 : Inherit properties of class car to ferrari (use keyword extends)

Step 5 : Inherit properties of class ferrari to ferrariSF90(use keyword extends)

Step 6 :Use/invoke the functions and variables of respective parent classes through derived class:maruthi800 to prove the concept of multi level inheritance

Step 7 : STOP

3.3 Coding using java Language

Program Description:

```
1. class Car
2. {
3.   public Car()
4.   {
5.     System.out.println("Class Car");
6.   }
7.   public void vehicleType()
8.   {
9.     System.out.println("Vehicle Type: Car");
10.  }
11. }
12. class ferrari extends Car{
13.   public ferrari()
14.   {
15.     System.out.println("Class ferrari");
16.   }
17.   public void brand()
18.   {
19.     System.out.println("Brand: ferrari");
20.   }
21.   public void speed()
22.   {
```



```

23. System.out.println("Max: 200Kmph");
24. }
25. }
26. public class ferrariSF90 extends ferrari
27. {
28. public ferrariSF90()
29. {
30. System.out.println("ferrariSF90");
31. }
32. public void speed()
33. {
34. System.out.println("Max:400Kmph");
35. }
36. public static void main(String args[])
37. {
38. ferrariSF90 obj=new ferrariSF90();
39. obj.vehicleType();
40. obj.brand();
41. obj.speed();
42. }
43. }

```

3.4 Expected Results

F:\JAVA>javac ferrariSF90.java

F:\JAVA>java ferrariSF90

Class Car

Class ferrari

ferrariSF90

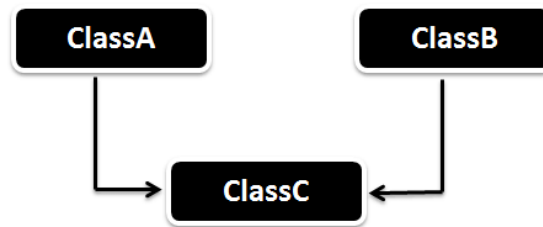
Vehicle Type: Car

Brand: ferrari

Max:400Kmph

PROGRAM 7 - Multiple Inheritance using Interface	
1	Problem Statement
<p>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.</p> <p>Aim: Develop multiple inheritance using Interfaces</p>	
2	Student Learning Outcomes
<p>After successful completion of this lab, the student shall be able to</p> <ul style="list-style-type: none"> • Understand why multiple inheritance is not supported in java • Understand what is interface • Able to implement a interface concepts in a program. 	
3	Design of the Program
3.1	Description
<p>Multiple <u>Inheritances</u> in Java is nothing but one class extending more than one class. <u>Java</u> does <i>not</i> have this capability. As the designers considered that multiple inheritance will to be too complex to manage, but indirectly you can achieve Multiple <u>Inheritance in Java</u> using <u>Interfaces</u>.</p> <p>Multiple Inheritances is a feature of object oriented concept, where a class can inherit properties of more than one parent class. The problem occurs when there exist methods with same signature in both the super classes and subclass. On calling the method, the compiler cannot determine which class method to be called and even on calling which class method gets the priority. Java does not support multiple inheritances of classes.</p> <p>As in Java we can implement more than one interface we achieve the same effect using interfaces.</p> <p><u>Flow Diagram</u></p>	

Conceptually Multiple Inheritance has to be like the below diagram, ClassA and ClassB both inherited by ClassC. Since it is not supported we will changing the ClassA to InterfaceA and ClassB to InterfaceB.



3.2 Algorithm

Step 1: **Start**

Step 2: Define `AnnualIncrement` and `Payroll` interfaces with `applyIncrement()` and `calculateSalary()`.

Step 3: Create an **abstract class** `Employee` with:

- Attributes: `name`, `id`, `basicSalary`, `grossSalary`, `netSalary`.
- Method `calculateSalary()`:
 - Compute **DA (70%)**, **HRA (30%)**, **Gross Salary**, **Income Tax (40%)**, and **Net Salary**.
- Abstract method `applyIncrement()`.

Step 4: Create **Manager class** (extends `Employee`):

- Implements `applyIncrement()`: **Increase basic salary by 10%**.

Step 5: Create **Technician class** (extends `Employee`):

- Implements `applyIncrement()`: **Increase basic salary by 15%**.

Step 6: In `main()` method:

- Create **1 Manager and 4 Technicians**.
- Store them in an `Employee` array (Polymorphism).
- Loop through each employee:
 - Call `applyIncrement()`.
 - Call `calculateSalary()`.
 - Call `display()`.

• **End**

3.3	Coding using C++ Language
	<pre> 1. interface AnnualIncrement { 2. void applyIncrement(); 3. } 4. 5. // Interface for Payroll Calculation 6. interface Payroll { 7. void calculateSalary(); 8. } 9. 10. // Abstract Employee Class 11. abstract class Employee implements AnnualIncrement, Payroll { 12. String name; 13. int id; 14. double basicSalary; 15. double grossSalary; 16. double netSalary; 17. 18. // Constructor 19. public Employee(String name, int id, double basicSalary) { 20. this.name = name; 21. this.id = id; 22. this.basicSalary = basicSalary; 23. } 24. 25. // Method to calculate gross salary 26. public void calculateSalary() { 27. double da = 0.7 * basicSalary; // 70% Dearness Allowance 28. double hra = 0.3 * basicSalary; // 30% House Rent Allowance 29. grossSalary = basicSalary + da + hra; 30. double incomeTax = 0.4 * grossSalary; // 40% Income Tax </pre>

```
31.     netSalary = grossSalary - incomeTax;
32. }
33.
34. // Abstract method for annual increment
35. public abstract void applyIncrement();
36.
37. // Display Employee Details
38. public void display() {
39.     System.out.println("\nEmployee ID: " + id);
40.     System.out.println("Name: " + name);
41.     System.out.println("Basic Salary: $" + basicSalary);
42.     System.out.println("Gross Salary: $" + grossSalary);
43.     System.out.println("Net Salary (After Tax): $" + netSalary);
44. }
45. }
46.
47. // Manager Class (inherits from Employee)
48. class Manager extends Employee {
49.     public Manager(String name, int id, double basicSalary) {
50.         super(name, id, basicSalary);
51.     }
52.
53.     // Overriding applyIncrement method
54.     public void applyIncrement() {
55.         basicSalary += 0.10 * basicSalary; // 10% increment for Manager
56.     }
57. }
58.
59. // Technician Class (inherits from Employee)
60. class Technician extends Employee {
61.     public Technician(String name, int id, double basicSalary) {
```

```

62.     super(name, id, basicSalary);
63. }
64.
65. // Overriding applyIncrement method
66. public void applyIncrement() {
67.     basicSalary += 0.15 * basicSalary; // 15% increment for Technicians
68. }
69. }
70.
71. // Main Class
72. public class XYZPayroll {
73.     public static void main(String[] args) {
74.         // Creating Manager and Technicians
75.         Employee manager = new Manager("Alice Johnson", 101, 50000);
76.         Employee tech1 = new Technician("Bob Smith", 102, 30000);
77.         Employee tech2 = new Technician("Charlie Brown", 103, 32000);
78.         Employee tech3 = new Technician("David Wilson", 104, 31000);
79.         Employee tech4 = new Technician("Eve Davis", 105, 29000);
80.
81.         // Storing employees in an array (Demonstrating Polymorphism)
82.         Employee[] employees = {manager, tech1, tech2, tech3, tech4};
83.
84.         // Apply increments and calculate salaries
85.         for (Employee emp : employees) {
86.             emp.applyIncrement(); // Polymorphic behavior
87.             emp.calculateSalary();
88.             emp.display();
89.         }
90.     }
91. }

```

4

Expected Results

```
F:\JAVA>javac XYZPayroll.java
```

```
F:\JAVA>java XYZPayroll
```

Employee ID: 101

Name: Alice Johnson

Basic Salary: \$55000.0

Gross Salary: \$110000.0

Net Salary (After Tax): \$66000.0

Employee ID: 102

Name: Bob Smith

Basic Salary: \$34500.0

Gross Salary: \$69000.0

Net Salary (After Tax): \$41400.0

Employee ID: 103

Name: Charlie Brown

Basic Salary: \$36800.0

Gross Salary: \$73600.0

Net Salary (After Tax): \$44160.0

Employee ID: 104

Name: David Wilson

Basic Salary: \$35650.0

Gross Salary: \$71300.0

Net Salary (After Tax): \$42780.0

Employee ID: 105

Name: Eve Davis

Basic Salary: \$33350.0

Gross Salary: \$66700.0

Net Salary (After Tax): \$40020.0

PROGRAM 8 - Exception handling	
1	Problem Statement Write a JAVA Program to handle multiple exceptions using Nested Try block
2	Student Learning Outcomes After successful completion of this lab, the student shall be able to <ul style="list-style-type: none"> • Understand what is exception • What are the types of exception • Why to use nested try catch block for excetion handling • Able to write a programto handle multiple exception using nested try catch block
3	Design of the Program
3.1	Description <p>Exception Handling in Java is one of the powerful <i>mechanism to handle the runtime errors</i> so that normal flow of the application can be maintained. Exception is an abnormal condition. In Java, an exception is an event that disrupts the normal flow of the program. It is an object which is thrown at runtime.</p> <p>Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, RemoteException, etc.</p> <p>Advantage of Exception Handling</p> <p>The core advantage of exception handling is to maintain the normal flow of the application. An exception normally disrupts the normal flow of the application that is why we use exception handling.</p> <p>The try block within a try block is known as nested try block in java.</p> <p>Why use nested try block?</p> <p>Sometimes a situation may arise where a part of a block may cause one error and the entire block itself may cause another error. In such cases, exception handlers have to be nested.</p> <p>Syntax:</p> <pre> try { statement 1; statement 2; try </pre>


```

        {
            statement 1;
            statement 2;
        }
        catch(Exception e)
        {
        }
    }
    catch(Exception e)
    {
    }
}

```

3.2 Algorithm

Step1: Create class Excep

Step 2: Create main

Step 3: Create try block with in try create another try catch block which handles deiveide be error exception

Step 4: Create one more try block to handle Array out of bound exception with in same try block

Step 5: Create a cathch block for the outer try block

Step 6: close the Class

Step 7: stop

3.3 Coding using C++ Language

```

1. class Excep{
2.     public static void main(String args[]){
3.         try{
4.             try{
5.                 System.out.println("going to divide");
6.                 int b =39/0;
7.             }catch(ArithmeticException e){System.out.println(e);}
8.         }try{
9.             int a[]=new int[5];
10.            a[5]=4;
11.        }catch(ArrayIndexOutOfBoundsException e){System.out.println(e);}

```

```
12. System.out.println("other statement");  
13. }catch(Exception e){System.out.println("handeled");}  
14. System.out.println("normal flow..");  
15. }  
16. }
```

4	Expected Results
----------	-------------------------

```
C:\lks>javac Excep.java  
  
C:\lks>java Excep  
going to divide  
java.lang.ArithmeticException: / by zero  
java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 5  
other statement  
normal flow..  
  
C:\lks>_
```

PROGRAM 9 - Custom Exceptions	
1	Problem Statement
<p>Develop a Banking System that allows users to withdraw money from their bank accounts. However, if the user tries to withdraw an amount greater than the available balance, the system should throw a custom exception called <code>InsufficientBalanceException</code>.</p>	
2	Student Learning Outcomes
<p>After successful completion of this lab, the student shall be able to</p> <ul style="list-style-type: none"> Identify the difference between in-built exception classes and user defined exception classes. Implement their own custom exception classes to handle various exceptions generated by the program. 	
3	Design of the Program
3.1	Description
<p>Java provides us facility to create our own exceptions which are basically derived classes of <code>Exception</code>. For example <code>InvalidAgeException</code> extends <code>Exception</code> class. In this program we are creating our own exception class such as <code>CustomExcep</code> to check the validity of a voting person based on his/her age criteria. If the person is not eligible to vote an exception will be thrown <code>CustomExcep</code> class.</p>	
3.2	Algorithm
<p>Step 1: Start</p> <p>Step 2 : Create a custom exception class <code>InsufficientBalanceException</code> extending the built-in <code>Exception</code> class.</p> <p>Step 3: Create a <code>CustomExcep</code> class</p> <p>Step 4: Create a <code>BankAccount</code> class with private attributes <code>accountHolder</code> and <code>balance</code>.</p> <p>Step 5: Define a constructor for the <code>BankAccount</code> class that takes <code>accountHolder</code> and <code>balance</code> parameters and initializes the corresponding attributes.</p> <p>Step 6: Define a <code>withdraw</code> method that takes a <code>amount</code> parameter and throws an <code>InsufficientBalanceException</code> if the withdrawal amount exceeds the current balance.</p> <p>Step 7: Check if the withdrawal amount is greater than the current balance.</p> <p>Step 8: If true, throw an <code>InsufficientBalanceException</code> with a custom error message.</p>	

Step 9: If false, subtract the withdrawal amount from the current balance and print a success message.

Step 10: . Define a displayAccountDetails method that prints the account holder's name and current balance.

Step 11: Test the program

Step 12: End

3.3 Coding using JAVA Language

```
1. // Custom Exception Class
2. class InsufficientBalanceException extends Exception {
3.     public InsufficientBalanceException(String message) {
4.         super(message);
5.     }
6. }
7.
8. // BankAccount Class
9. class BankAccount {
10.     private String accountHolder;
11.     private double balance;
12.
13.     // Constructor
14.     public BankAccount(String accountHolder, double balance) {
15.         this.accountHolder = accountHolder;
16.         this.balance = balance;
17.     }
18.
19.     // Method to withdraw money
20.     public void withdraw(double amount) throws InsufficientBalanceException {
21.         if (amount > balance) {
22.             throw new InsufficientBalanceException("Insufficient balance! Your current
                balance is: $" + balance);
23.         }
```

```

24.     balance -= amount;
25.     System.out.println("Withdrawal successful! Remaining balance: $" + balance);
26. }
27.
28. // Display account details
29. public void displayAccountDetails() {
30.     System.out.println("\nAccount Holder: " + accountHolder);
31.     System.out.println("Account Balance: $" + balance);
32. }
33.
34. public static void main(String[] args) {
35.     BankAccount account = new BankAccount("Alice", 1000);
36.
37.     account.displayAccountDetails();
38.
39.     try {
40.         account.withdraw(500); // Successful withdrawal
41.         account.withdraw(600); // Should throw custom exception
42.     } catch (InsufficientBalanceException e) {
43.         System.out.println("Exception: " + e.getMessage());
44.     }
45. }
46. }

```

4	Expected Results
----------	-------------------------

F:\JAVA>javac BankAccount.java

F:\JAVA>java BankAccount

Account Holder: Alice

Account Balance: \$1000.0

Withdrawal successful! Remaining balance: \$500.0

Exception: Insufficient balance! Your current balance is: \$500.0

PROGRAM 10 – java program on generic class	
1	Problem Statement
Write a program to create a simple generic class, where T is a type parameter that will be replaced by a real type, when an object of type Gen class is created.	
2	Student Learning Outcomes
<p>After successful completion of this lab, the student shall be able to</p> <ul style="list-style-type: none"> • Illustrate the use of generic methods and classes. • Implement small real world applications using generic methods and classes. 	
3	Design of the Program
3.1	Description
<p>A class is generic if it declares one or more type variables. These type variables are known as the type parameters of the class. A programmer can set any object; and can expect any return value type from get method since all java types are subtypes of Object class.</p> <p>Generic types are instantiated to form parameterized types by providing actual type arguments that replace the formal type parameters. A class like Gen<T> is a generic type, that has a type parameter T. Instantiations, such as Gen<Integer> or a Gen<String>, are called parameterized types, and String and Integer are the respective actual type arguments.</p> <p>getClass returns a Class object that represents the object's class. getName then returns the name of that class as a string. So for example "hello".getClass().getName() will return "java.lang.String" and new ArrayList<String>().getClass().getName() will return java.util.ArrayList.</p>	
3.2	Algorithm
<p>Step 1: Start</p> <p>Step 2: Create a generic class Gen<T></p> <p>Step 3: Create a constructor to assign the value and getob() to return the value of an object.</p> <p>Step 4: Create a class GenDemo to demonstrate generic class Gen<T></p> <p>Step 5: Declare an object that takes integer as its argument that displays the type of argument through getClass().getname()</p> <p>Step 6: Print the value of an object through getob().</p> <p>Step 7: End</p>	

3.3 Coding using JAVA Language

```
1. // A simple generic class.
2. // Here, T is a type parameter that
3. // will be replaced by a real type
4. // when an object of type Gen is created.
5. class Gen<T> {
6.     T ob; // declare an object of type T
7.     // Pass the constructor a reference to
8.     // an object of type T.
9.     Gen(T o) {
10.         ob = o;
11.     }
12.     // Return ob.
13.     T getob() {
14.         return ob;
15.     }
16.     // Show type of T.
17.     void showType() {
18.         System.out.println("Type of T is " +ob.getClass().getName());
19.     }
20. }

21. // Demonstrate the generic class.
22. class GenDemo {
23.     public static void main(String args[]) {
24.         // Create a Gen reference for Integers.
25.         Gen<Integer> iOb;
26.         // Create a Gen<Integer> object and assign its
27.         // reference to iOb. Notice the use of autoboxing
28.         // to encapsulate the value 88 within an Integer object.
```

```

29. iOb = new Gen<Integer>(88);
30. // Show the type of data used by iOb.
31. iOb.showType();
32. // Get the value in iOb. Notice that
33. // no cast is needed.
34. int v = iOb.getob();
35. System.out.println("value: " + v);
36. System.out.println();
37. // Create a Gen object for Strings.
38. Gen<String> strOb = new Gen<String>("Generics Test");
39. // Show the type of data used by strOb.
40. strOb.showType();

41. // Get the value of strOb. Again, notice
42. // that no cast is needed.
43. String str = strOb.getob();
44. System.out.println("value: " + str);
45. }
}

```

4	Expected Results
----------	-------------------------

Type of T is java.lang.Integer
value: 88
Type of T is java.lang.String
value: Generics Test

PROGRAM 11 – Java program for multi-thread application	
	Problem Statement
Write a Java program that implements a multi-thread application that hash tree threads. First thread generates a random integer for every 1 second; second thread computes the square of the number and prints; third thread will print the value of cube of the number.	
2	Student Learning Outcomes
After successful completion of this lab, the student shall be able to <ul style="list-style-type: none"> • Illustrate the use of multithreads • Implement small real world applications multithreading 	
3	Design of the Program
3.1	Description
This Java program demonstrates the concept of multithreading by generating a random number and performing square and cube calculations using multiple threads. The program consists of three main classes: <code>first</code> , <code>second</code> , and <code>third</code> . The <code>first</code> class extends the <code>Thread</code> class and generates a random number in each iteration. For every generated number, two additional threads are created: one for calculating the square (<code>second</code> class) and another for calculating the cube (<code>third</code> class). The <code>second</code> and <code>third</code> classes implement the <code>Runnable</code> interface, which allows them to execute in separate threads. The <code>run()</code> method in each class defines the specific task of the thread. The <code>Thread.sleep(1000)</code> method introduces a delay of 1 second between iterations to simulate real-time execution. This program highlights the advantages of multithreading, such as parallel execution and efficient CPU utilization , by allowing multiple computations to run simultaneously.	
3.2	Algorithm
<i>Step 1: Start</i> 1.1. Create the Multithread class with the main method. 1.2. Create an object of the <code>first</code> class and call its <code>start()</code> method to begin execution. <i>Step 2: Define the first Class (Extends Thread)</i> 2.1. Inside the <code>run()</code> method: 2.1.1. Create an object of the <code>Random</code> class. 2.1.2. Run a loop 5 times to generate random numbers.	

- 2.1.3. Generate a random number between 0 and 99.
- 2.1.4. Print the generated random number.
- 2.1.5. Create a new thread for the second class and start it.
- 2.1.6. Create a new thread for the third class and start it.
- 2.1.7. Pause execution for 1 second using Thread.sleep(1000).
- 2.2. Handle exceptions using a try-catch block.

Step 3: Define the second Class (Implements Runnable)

- 3.1. Store the received random number in an instance variable.
- 3.2. Inside the run() method:
 - 3.2.1. Compute the square of the number.
 - 3.2.2. Print the square of the number.

Step 4: Define the third Class (Implements Runnable)

- 4.1. Store the received random number in an instance variable.
- 4.2. Inside the run() method:
 - 4.2.1. Compute the cube of the number.
 - 4.2.2. Print the cube of the number.

Step 5: Execute the Program

- 5.1. The first thread generates a number and starts two new threads for second and third.
- 5.2. The second thread calculates and prints the square of the number.
- 5.3. The third thread calculates and prints the cube of the number.
- 5.4. Repeat this process 5 times for different random numbers.

Step 6: End

- 6.1. Program execution completes after five iterations.

3.3	Coding using JAVA Language
------------	-----------------------------------

```
import java.util.*;

public class Multithread
{
    public static void main(String args[])
    {
        first a=new first();
        a.start();
    }
}
```

```

class first extends Thread
{
    public void run()
    {
        int num=0;
        Random r=new Random();
        try
        {
            for(int i=0;i<5;i++)
            {
                num=r.nextInt(100);
                System.out.println("first thread generated number is"+num);

                Thread t2=new Thread (new second(num));
                t2.start();

                Thread t3=new Thread(new third(num));
                t3.start();
                Thread.sleep(1000);
            }
        }
        catch(Exception e)
        {
            System.out.println(e.getMessage());
        }
    }
}

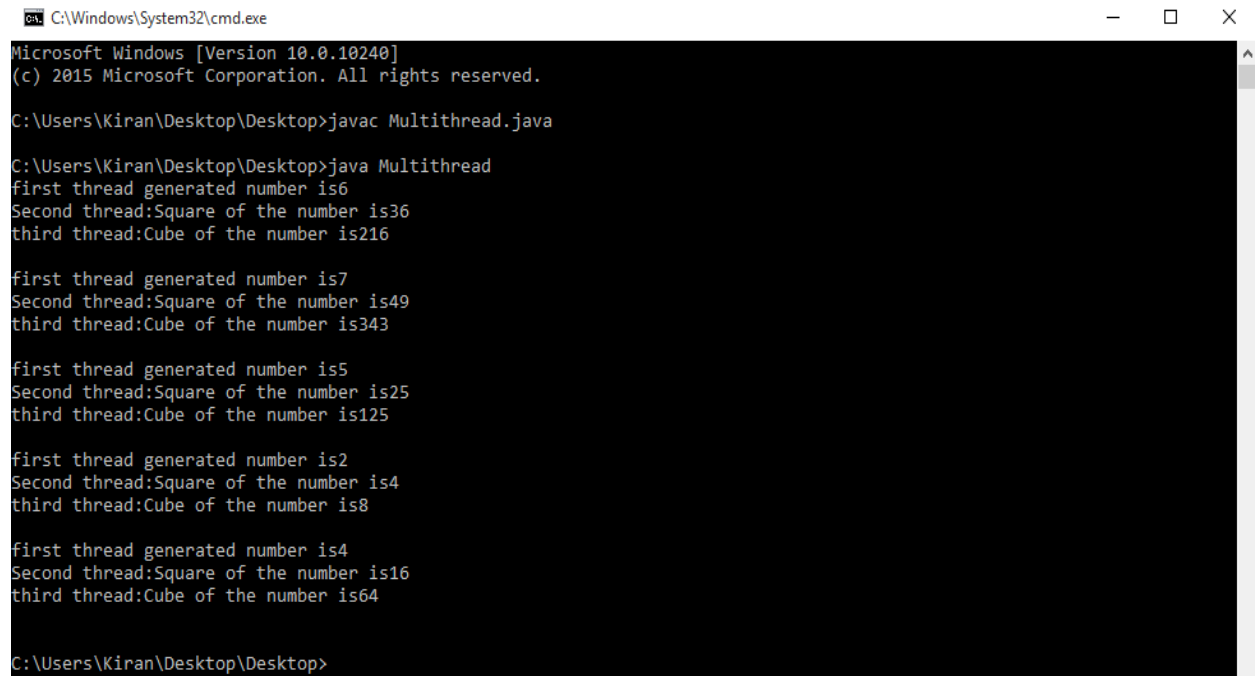
class second implements Runnable
{
    public int x;
    public second (int x)
    {
        this.x=x;
    }
    public void run()
    {
        System.out.println("Second thread:Square of the number is" + (x*x));
    }
}

class third implements Runnable
{
    public int x;
    public third(int x)
    {
        this.x=x;
    }
}

```

```
    }  
    public void run()  
    {  
        System.out.println("third thread:Cube of the number is" + (x*x*x) + "\n");  
    }  
}
```

4 Expected Results



```
C:\Windows\System32\cmd.exe  
Microsoft Windows [Version 10.0.10240]  
(c) 2015 Microsoft Corporation. All rights reserved.  
  
C:\Users\Kiran\Desktop\Desktop>javac Multithread.java  
  
C:\Users\Kiran\Desktop\Desktop>java Multithread  
first thread generated number is6  
Second thread:Square of the number is36  
third thread:Cube of the number is216  
  
first thread generated number is7  
Second thread:Square of the number is49  
third thread:Cube of the number is343  
  
first thread generated number is5  
Second thread:Square of the number is25  
third thread:Cube of the number is125  
  
first thread generated number is2  
Second thread:Square of the number is4  
third thread:Cube of the number is8  
  
first thread generated number is4  
Second thread:Square of the number is16  
third thread:Cube of the number is64  
  
C:\Users\Kiran\Desktop\Desktop>
```