Programming with JAVA Lab Manual

B22CI0407

Second Year

4th Sem

SCHOOL OF COMPUTING

AND

INFORMATION TECHNOLOGY

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.	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. Aim: Implementation of Simple Calculator	
2	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 and consonants in a string. Aim: Count Vowels and Consonants in a given string.	
3.	Using a one-dimensional array, read an array of integer elements and perform the following operations. a. Copy all elements from one array to another. b. Remove duplicate elements from array and print only even position of array.	
4.	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. 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. Aim: Implement Static Variable for a Student database	
5.	Volume of a box to be computed using different features of a box: height, width, and depth. Write a generic java program that accepts the values of the features of a box during the construction of its object and calculate its volume and display the same. Note: Student should identify the classes, data and function members in each class and write the program. Aim: Implementation of constructor to calculate the volume of a box	
6	A child inherits the features of parents and 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. Aim: Construct multilevel inheritance	
7	XYZ technologies is a 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. The House Rent Allowance is 30% basic for all employees. Income Tax is 40% of gross salary for all employees. The annual increments for 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. Aim: Develop multiple inheritance using Interfaces	
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, and zero), with three try/catch blocks, and prints either the output of halfOf() or the caught Odd Exception. Aim: Demonstration for handling multiple exceptions and nested try blocks	
9	Implement Custom Exceptions.	
10.	Construct generic methods and generic class.	
Viva Voice		

S.NO	Exercise Programs
1	Programs on foreach loop
2	Programs on Strings
3	Programs on Arrays
4	Programs on Package creation and usage
5	Programs on Nested classes
6	Programs on Lambda expression
7	Programs on Anonymoues class
8	Programs on Abstract class
9	Programs on Concreate class
10	Programs on Enumerations
11	Programs on Exception handling
12	Programs on finally
13	Programs on type bounds
14	Programs on iterator, sets, map
15	Programs on super class

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 the needs of society, industry, and others.

IV.GUIDELINES TO THE STUDENTS

1. The students should have a 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.

The 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, the 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:

```java
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. Write the above program 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 students 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 '+' ← output = num1 + num2; go to STEP 4

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

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

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

default←print "You have entered wrong operator"

END switch

Step 4: print num1,operator,num2,output

Step 5: END

3.3	**Coding using JAVA Language**

```java
import java.util.Scanner;
public class JavaExample {
 public static void main(String[] args) {
double num1, num2;
Scanner sc = new Scanner(System.in);
System.out.print("Enter first number:");
 num1 = sc.nextDouble( );
System.out.print("Enter second number:");
num2 = sc.nextDouble( );
System.out.print("Enter an operator (+, -, *, /): ");
char operator = sc.next( ).charAt(0);
sc.close( );
double output;
switch(operator)
 {
case '+':  output = num1 + num2;     break;
case '-':  output = num1 – num2;     break;
 case '*': output = num1 * num2;    break;
case '/':   output = num1 / num2;     break;
 default:  System.out.printf("You have entered wrong operator");    return;
}
System.out.println(num1+" "+operator+" "+num2+": "+output);
 }
}
```

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 – Count Vowels and Consonants in a given string	
1	**Problem Statement**
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 and consonants in a string.	
2	**Student Learning Outcomes**
After successful completion of this lab, the student shall be able to - Convert uppercase characters to lowercase characters. - Compare the characters in a string with all the vowels	
3	**Design of the Program**
3.1	**Description**
In this program, our task is to count the total number of vowels and consonants present in the given string. 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. 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.	
3.2	**Algorithm**
Step 1: START **Step 2:** SET vCount =0, cCount =0 **Step 3:** DEFINE string str = "This is a really simple sentence". **Step 4:** CONVERT str to lowercase **Step 5:** SET i =0. **Step 6:** REPEAT STEP 6 to STEP 8 UNTIL i<str.length() **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:** i = i + 1 **Step 10:** PRINT vCount. **Step 11:** PRINT cCount. **Step 12:** END
3.3	**Coding using JAVA Language**

```java
public class CountVowelConsonant {

  public static void main(String[] args) {


    //Counter variable to store the count of vowels and consonant
    int vCount = 0, cCount = 0;


    //Declare a string
    String str = "This is a really simple sentence";


    //Converting entire string to lower case to reduce the comparisons
    str = str.toLowerCase( );


    for(int i = 0; i < str.length( ); i++) {
      //Checks whether a character is a vowel
      if(str.charAt(i) == 'a' || str.charAt(i) == 'e' || str.charAt(i) == 'i' || str.charAt(i) == 'o' || str.charAt(i) == 'u') {
        //Increments the vowel counter
        vCount++;
      }
      //Checks whether a character is a consonant
      else if(str.charAt(i) >= 'a' && str.charAt(i)<='z') {
        //Increments the consonant counter
        cCount++;
      }
```

```
        }
        System.out.println("Number of vowels: " + vCount);
        System.out.println("Number of consonants: " + cCount);
    }
}
```

4	**Expected Results**
Number of vowels: 10 Number of consonants: 17	

Number of vowels: 10

Number of consonants: 17

	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 • 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**	
Arrays: 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. 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.		
3.2	**Algorithm**	
Step 1: START **Step 2:** INITIALIZE arr1[] ← {1, 2, 3, 4, 5} **Step 3:** CREATE arr2[] of size arr1[]. **Step 4:** COPY elements of arr1[] to arr2[] **Step 5:** REPEAT STEP 6 UNTIL (i<arr1.length) **Step 6:** arr2[i] ← arr1[i] **Step 7:** DISPLAY elements of arr1[]. **Step 8:** REPEAT STEP 9 UNTIL (i<arr1.length) **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**
| . | ```java
public class CopyArray {
  public static void main(String[] args) {
        //Initialize array
     int [ ] arr1 = new int [] {1, 2, 3, 4, 5};
      //Create another array arr2 with size of arr1
     int arr2[] = new int[arr1.length];
     //Copying all elements of one array into another
     for (int i = 0; i < arr1.length; i++) {
        arr2[i] = arr1[i];
     }
      //Displaying elements of array arr1
     System.out.println("Elements of original array: ");
     for (int i = 0; i < arr1.length; i++) {
        System.out.print(arr1[i] + " ");
     }
     System.out.println( );
     //Displaying elements of array arr2
     System.out.println("Elements of new array: ");
     for (int i = 0; i < arr2.length; i++) {
       System.out.print(arr2[i] + " ");
     }
   }
}
``` |
| **4** | **Expected Results** |

```
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
```

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**
After successful completion of this lab, the student shall be able to - 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**
- 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**

```java
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

<div align="center">

OR

</div>

//deletion of duplicate elements using single array

```java
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 < k; 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**
	After successful completion of this lab, the student shall be able to • 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**
	Static Variables: 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. **Important points for static variables:** • 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**
	Step 1: START **Step 2:** Create a class Student with members SRN, name and static variable collegeName. **Step 3:** Initialize some values to the members of the class Student including static variable collegeName.

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**

```java
class Student
{
String SRN;
String name;
static String collegeName; //static variable
public static void main(String[] args)
{
    //create 3 object which will share collegeName value
Student s1= new Student();
Student s2= new Student();
Student s3= new Student();
    //assign value to static variable collegeName
Student.collegeName="REVA UNIVERSITY";
    //assign values to instance variables
s1.SRN="R18CS001";
s1.name="stud1";
s2.SRN="R18CS002";
s2.name="stud2";
s3.SRN="R18CS003";
s3.name="stud3";
    //Print the values of the objects
System.out.println("S1  SRN.= "+s1.SRN+" S1  Name= "+s1.name+" S1  College Name= "+s1.collegeName );
System.out.println("S2  SRN.= "+s2.SRN+" S2  Name= "+s2.name+" S2  College Name= "+s2.collegeName );
```

```
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
s2.collegeName="REVA";
s2.name="JAMES";
        //Print the values of the objects after change
System.out.println("S1  SRN.=  "+s1.SRN+"  S1  Name=  "+s1.name+"  S1  College  Name=
"+s1.collegeName );
System.out.println("S2  SRN.=  "+s2.SRN+"  S2  Name=  "+s2.name+"  S2  College  Name=
"+s2.collegeName );
System.out.println("S3  SRN.=  "+s3.SRN+"  S3  Name=  "+s3.name+"  S3  College  Name=
"+s3.collegeName );
 }
 }
```

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

```java
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

Volume of a box to be computed using different features of a box: height, width and depth. Write a java program that accepts the values of the features of a box during the construction of its object and calculate its volume and display the same.

Note: Student should identify the classes, data and function members in each class and write the program.

2	Student Learning Outcomes

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

- Identify the classes, data and function members in each class
- Implement the computation of volume of a box

3	Design of the Program

3.1	Description

Parameterised constructor:

A class represents a real world entityA Box's volume is calculated using its width, height & breadth. These parameters of a box are represented as variables of a class and are initialized via parameterized constructor. The parameterized constructor is called when an object is created and assigns the values to the variables of class.

Syntax:

  class(keyword) class_name(user-defined)

  {Data_type variable name;

   Return_type function_name(parameters);

   }

3.2	Algorithm

  Step 1 : START

  Step 2 : declare a class that represents box in real world

  Step 3 : declare variables of the entity box that represents height, width & depth

  Step 4 : Define a function that computes the volume: height*width*depth

  Step 5 : Display the output

  Step 6 : STOP

3.3	**Coding using java Language**

Program Description:

```java
class Box
{
double width;
double height;
double depth;
// This is the constructor for Box.
Box(double w, double h, double d)
{
width = w;
height = h;
depth = d;
}
// compute and return volume
double volume()
{
return width * height * depth;
}
}
classBoxDemo
{
public static void main(String args[])
{
// declare, allocate, and initialize Box objects
Box mybox1 = new Box(10, 20, 15);
Box mybox2 = new Box(3, 6, 9);
doublevol;
// get volume of first box
```

```
vol = mybox1.volume();

System.out.println("Volume is " + vol);

// get volume of second box

vol = mybox2.volume();

System.out.println("Volume is " + vol);

}

}
```

3.4	Expected Results

Output:

Volume is 3000.0

 Volume is 162.0

3.5	**Implementation Phase : Execute the Program** **Compile, remove syntax errors, if any, generate object code and make note of the same**.

3.6	**Simulate the Errors**
3.6.1	**Syntax Error**

Sl.No.	Name of the Error	Cause for the Error	Rectification

3.6.2	Logical Error

Sl.No.	Name of the Error	Cause for the Error	Rectification

4	Final Program and Results

	PROGRAM 6 – Multilevel Inheritance
1	**Problem Statement**

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.
Note: Student should identify the classes, data and function members in each class and write the program.

2	**Student Learning Outcomes**

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

- 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**

Multilevel inheritance:

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.

Syntax:

```
class(keyword) class_name-1(user-defined)
{
}
Class class_name-2: extends class_name-1
{
}
```

3.2	**Algorithm**

  Step 1 : START

  Step 2 : declare 3 classes that represents a real world entity: car, maruthi and maruthi800

Step 3 : declares variables of all the classes: car, maruthi and maruthi800.

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

Step 5 : Inherit properties of class maruthi to maruthi800(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:

```
class Car
{
public Car()
{
System.out.println("Class Car");
}
public void vehicleType()
{
System.out.println("Vehicle Type: Car");
}
}
Class Maruti extends Car{
publicMaruti()
{
System.out.println("Class Maruti");
}
public void brand()
{
System.out.println("Brand: Maruti");
}
public void speed()
{
```

```java
System.out.println("Max: 90Kmph");

}

}

public class Maruti800 extends Maruti

{

public Maruti800()

{

System.out.println("Maruti Model: 800");

}

public void speed()

{

System.out.println("Max: 80Kmph");

}

public static void main(String args[])

{

Maruti800 obj=new Maruti800();

obj.vehicleType();

obj.brand();

obj.speed();

}

}
```

3.4	**Expected Results**

ClassCar

ClassMaruti

MarutiModel: 800

VehicleType: Car

Brand: Maruti

Max: 80Kmph

3.5	**Implementation Phase : Execute the Program** **Compile, remove syntax errors, if any, generate object code and make note of the same**.

3.6	**Simulate the Errors**		
3.6.1	**Syntax Error**		

Sl.No.	Name of the Error	Cause for the Error	Rectification

3.6.2	**Logical Error**		

Sl.No.	Name of the Error	Cause for the Error	Rectification

4	**Final Program and Results**		

PROGRAM 7 - Multiple Inheritance using Interface	
1	**Problem Statement**
	Implement JAVA program on multiple inheritance using Interface. Here we have two interfaces **Car** and **Bus.** • **Car** interface has a attribute **speed** and a method defined **distanceTravelled**() • **Bus** interface has a attribute **distance** and method **speed**()
2	**Student Learning Outcomes**
	After successful completion of this lab, the student shall be able to • 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**
	Multiple Inheritances in Java is nothing but one class extending more than one class. Java does *not* have this capability. As the designers considered that multiple inheritance will to be too complex to manage, but indirectly you can achieve Multiple Inheritance in Java using Interfaces. 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. As in Java we can implement more than one interface we achieve the same effect using interfaces. ***Flow Diagram*** 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: Create interface Car with attribute speed and method distanceTravelled( )

Step 2: Create interface Bus with attribute distance and method speed( )

Step 3: create **Vehicle class** implements both interface **Car and Bus** and provides implementation

Step 4: write the definition of methods distanceTravelled ( ) and Speed( ) within Vechicle class

Step 5: create a object v1 of Vechicle class

Step 5: Invoke method distanceTravelled() of interface Car using the object v1

Step 6: Invoke method Speed() of interface Bus using the object v1

Step 7: Stop

3.3	Coding using C++ Language

```
interface Car
{
int  speed=60;
public void distanceTravelled();
}
interface Bus
{
int distance=100;
public void speed();
}
public class Vehicle  implements Car,Bus
{
int distanceTravelled;
int averageSpeed;
public void distanceTravelled()
```

```
{
distanceTravelled=speed*distance;
System.out.println("Total Distance Travelled is : "+distanceTravelled);
}
public void speed()
{
int averageSpeed=distanceTravelled/speed;
System.out.println("Average Speed maintained is : "+averageSpeed);
}
public static void main(String args[])
{
Vehicle v1=new Vehicle();
v1.distanceTravelled();
v1.speed();
}
}
```

4	**Expected Results**
3.5	**Implementation Phase : Execute the Program** **Compile, remove syntax errors, if any, generate object code and make note of the**

3.6	Simulate the Errors		

3.6.1	Syntax Error		

Sl.No.	Name of the Error	Cause for the Error	Rectification

3.6.2	Logical Error		

Sl.No.	Name of the Error	Cause for the Error	Rectification

4	Final Program and Results		

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 • 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

Exception Handling in Java is one of the powerful *mechanism to handle the runtime errors* 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.

Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, RemoteException, etc.

Advantage of Exception Handling

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.

The try block within a try block is known as nested try block in java.

Why use nested try block?

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.

Syntax:

```
try
  {
    statement 1;
   statement 2;
    try
```

```
        {
            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

```cpp
class Excep{

public static void main(String args[]){

try{

try{

System.out.println("going to divide");

int b =39/0;

}catch(ArithmeticException e){System.out.println(e);}

try{

int a[]=new int[5];

a[5]=4;

}catch(ArrayIndexOutOfBoundsException e){System.out.println(e);}
```

	System.out.println("other statement");
	}catch(Exception e){System.out.println("handeled");}
	System.out.println("normal flow..");
	}
	}
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>_
```

3.5	**Implementation Phase : Execute the Program** **Compile, remove syntax errors, if any, generate object code and make note of the**
3.6	**Simulate the Errors**
3.6.1	**Syntax Error**

Sl.No.	Name of the Error	Cause for the Error	Rectification

3.6.2	Logical Error		

Sl.No.	Name of the Error	Cause for the Error	Rectification

4	Final Program and Results

PROGRAM 9 - Custom Exceptions

1	**Problem Statement**

Implement a Java program, to check the validity of voting customer by providing customer's age as input using custom JAVA exception.

2	**Student Learning Outcomes**

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

- 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**

Java provides us facility to create our own exceptions which are basically derived classes of Exception. For example InvalidAgeException extends Exception class. In this program we are creating our own exception class such as CustomExcep 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 CustomExcep class.

3.2	**Algorithm**

Step 1: Start

Step 2 : Create InvalidAgeException class extends Exception

Step 3: Create a CustomExcep class

Step 4: if age is less than 18 throw an InvalidAgeException

Step 5: print "welcome to vote"

Step 6: main function to call validate function to check the validity of a person defined inside CustomExcep class

Step 7: End

3.3	Coding using JAVA Language

```java
class InvalidAgeException extends Exception{
    InvalidAgeException(String s){
super(s);
}
}
class CustomExcep{
static void validate(int age)throws InvalidAgeException{
if(age<18)
throw new InvalidAgeException("not valid");
else
System.out.println("welcome to vote");
}
public static void main(String args[]){
try{
validate(13);
}catch(Exception m){System.out.println("Exception occured: "+m);}
System.out.println("rest of the code...");
}
}
```

4	Expected Results

1. Input :Age=13

Exception occurred : InvalidAgeException: not valid

rest of the code……

2. Input :Age=19

Welcome to vote

rest of the code……

3.5	Implementation Phase : Execute the Program Compile, remove syntax errors, if any, generate object code and make note of the

3.6	Simulate the Errors
3.6.1	Syntax Error

Sl.No.	Name of the Error	Cause for the Error	Rectification

3.6.2	Logical Error

Sl.No.	Name of the Error	Cause for the Error	Rectification

4	Final Program and Results

PROGRAM 10 - Basic JAVA Program	
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**

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

- 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**

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.

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.

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.

3.2	**Algorithm**

Step 1: Start

Step 2: Create a generic class Gen<T>

Step 3: Create a constructor to assign the value and getob() to return the value of an object.

Step 4: Create a class GenDemo to demonstrate generic class Gen<T>

Step 5: Declare an object that takes integer as its argument that displays the type of argument through getclass().getname()

Step 6: Print the value of an object through getob().

Step 7: End

3.3	Coding using JAVA Language

```java
// A simple generic class.
// Here, T is a type parameter that
// will be replaced by a real type
// when an object of type Gen is created.
class Gen<T> {
T ob; // declare an object of type T
// Pass the constructor a reference to
// an object of type T.
Gen(T o) {
ob = o;
}
// Return ob.
T getob() {
return ob;
}
// Show type of T.
void showType() {
System.out.println("Type of T is " +ob.getClass().getName());
}
}

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

51

```
        iOb = new Gen<Integer>(88);

        // Show the type of data used by iOb.

        iOb.showType();

        // Get the value in iOb. Notice that

        // no cast is needed.

        int v = iOb.getob();

        System.out.println("value: " + v);

        System.out.println();

        // Create a Gen object for Strings.

        Gen<String> strOb = new Gen<String>("Generics Test");

        // Show the type of data used by strOb.

        strOb.showType();


        // Get the value of strOb. Again, notice

        // that no cast is needed.

        String str = strOb.getob();

        System.out.println("value: " + str);

    }

}
```

4	**Expected Results**
	Type of T is java.lang.Integer value: 88 Type of T is java.lang.String value: Generics Test
3.5	**Implementation Phase : Execute the Program** **Compile, remove syntax errors, if any, generate object code and make note of the**

3.6	**Simulate the Errors**		
3.6.1	**Syntax Error**		

Sl.No.	Name of the Error	Cause for the Error	Rectification

3.6.2	**Logical Error**		

Sl.No.	Name of the Error	Cause for the Error	Rectification

4	**Final Program and Results**