B20CI0301

Programming in Java Laboratory Manual

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CONTENTS

	and zero), with three try/catch blocks, and prints either the output of halfOf() or the caught Odd Exception.	
9	Implement a class named Fraction that represents fractions with numerator and denominator always stored reduced to lowest terms. If fraction is negative, the numerator will always be negative, and all operations leave results stored in lowest terms. Implement the addition, subtraction, multiplication and division operation for the Fraction class and also handle divide by zero using java exception handling mechanism.	50
10	Create a class Student that has instance variables as Name, Age, Address and access transmutation methods to access the instance variables along with display method to print the details of student. Next write a main() function that will create a collection of 10students and reverse the list. Print the details before and after reversing the collection.	61
11	Use generics to build a class Sort. Implement the bubble sort algorithm to sort an array of any type.	66
12	Write a generic method to count the number of elements in a collection that have a specific property (for example, odd integers, prime numbers, palindromes).	70
	List of additional experiments	74
Viva Voice		

I.GUIDELINES TO THE STUDENTS

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

II. LAB REQUIREMENTS

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

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

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

Simple Java Program:

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

Output: This is my first program in java

How to compile and run the above program

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

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

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

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

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

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

```
javac FirstJavaProgram.java
```

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

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

Set Path in Windows:

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

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

Note: Your jdk version may be different.

Set Path in Mac OS X

Open Terminal, type the following command and hit return.

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

Type the following command on terminal to confirm the path.

```
echo $JAVA_HOME
```

That's it.

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

```
java FirstJavaProgram
```

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

PROGRAM 1 - 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 '+' ← 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

```
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;
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
```

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 2 – Simple calculator using Switch case	
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

```
1.  public class CountVowelConsonant {
2.     public static void main(String[] args) {

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

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

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

9.        for(int i = 0; i < str.length(); i++) {
10.          //Checks whether a character is a vowel
11.          if(str.charAt(i) == 'a' || str.charAt(i) == 'e' || str.charAt(i) == 'i' || str.charAt(i) == 'o' ||
    str.charAt(i) == 'u') {
12.             //Increments the vowel counter
13.             vCount++;
14.          }
15.          //Checks whether a character is a consonant
16.          else if(str.charAt(i) >= 'a' && str.charAt(i)<='z') {
17.             //Increments the consonant counter
18.             cCount++;
19.          }
```

```
20.         }
21.         System.out.println("Number of vowels: " + vCount);
22.         System.out.println("Number of consonants: " + cCount);
23.     }
24. }
```

4	**Expected Results**

Number of vowels: 10

Number of consonants: 17

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 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

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

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
```

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 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**
1.	public class Remove{
2.	public static int removeDuplicateElements(int arr[], int n){
3.	if (n==0 || n==1){
4.	return n;
5.	}
6.	int[] temp = new int[n];
7.	int j = 0;
8.	for (int i=0; i<n-1; i++){
9.	if (arr[i] != arr[i+1]){
10.	temp[j++] = arr[i];
11.	}
12.	}
13.	temp[j++] = arr[n-1];
	// Changing original array
14.	for (int i=0; i<j; i++){
15.	arr[i] = temp[i];
16.	}
17.	return j;
18.	}
19.	public static void main (String[] args) {
20.	int arr[] = {10,20,20,30,30,40,50,50};
21.	int length = arr.length;
22.	length = removeDuplicateElements(arr, length);
	//printing array elements
23.	for (int i =0; i < length; i = i+2) {
24.	System.out.println(arr[i]);
25.	}
26.	}
27.	}
4	**Expected Results**

```
C:\shivu>javac Remove.java

C:\shivu>java Remove
10
30
50

C:\shivu>
```

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 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

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

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>
```

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

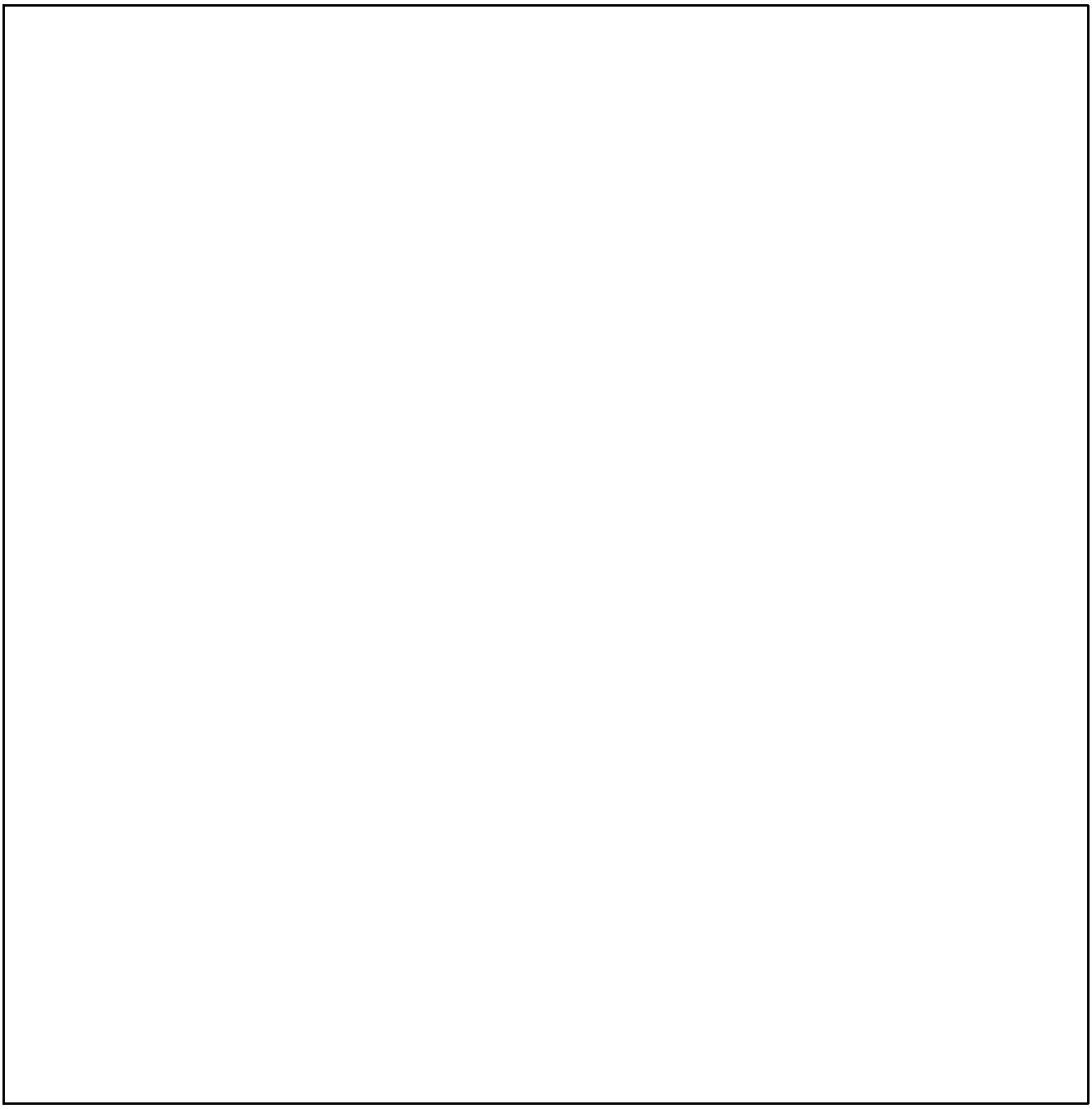
3.6	**Simulate the Errors**
3.6.1	**Syntax Error**

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 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 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.

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:

```
1. class Box
2. {
3. double width;
4. double height;
5. double depth;
6. // This is the constructor for Box.
7. Box(double w, double h, double d)
8. {
9. width = w;
10. height = h;
11. depth = d;
12. }
13. // compute and return volume
14. double volume()
15. {
16. return width * height * depth;
17. }
18. }
19. Class BoxDemo
20. {
21. public static void main(String args[])
22. {
23. // declare, allocate, and initialize Box objects
24. Box mybox1 = new Box(10, 20, 15);
25. Box mybox2 = new Box(3, 6, 9);
26. double vol;
```

```
27. // get volume of first box
28. vol = mybox1.volume();
29. System.out.println("Volume is " + vol);
30. // get volume of second box
31. vol = mybox2.volume();
32. System.out.println("Volume is " + vol);
33. }
34. }
```

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:

```
35. class Car
36. {
37. public Car()
38. {
39. System.out.println("Class Car");
40. }
41. public void vehicleType()
42. {
43. System.out.println("Vehicle Type: Car");
44. }
45. }
46. Class Maruti extends Car{
47. Public Maruti()
48. {
49. System.out.println("Class Maruti");
50. }
51. public void brand()
52. {
53. System.out.println("Brand: Maruti");
54. }
55. public void speed()
56. {
```

```
57. System.out.println("Max: 90Kmph");
58. }
59. }
60. public class Maruti800 extends Maruti
61. {
62. public Maruti800()
63. {
64. System.out.println("Maruti Model: 800");
65. }
66. public void speed()
67. {
68. System.out.println("Max: 80Kmph");
69. }
70. public static void main(String args[])
71. {
72. Maruti800 obj=new Maruti800();
73. obj.vehicleType();
74. obj.brand();
75. obj.speed();
76. }
77. }
```

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**.

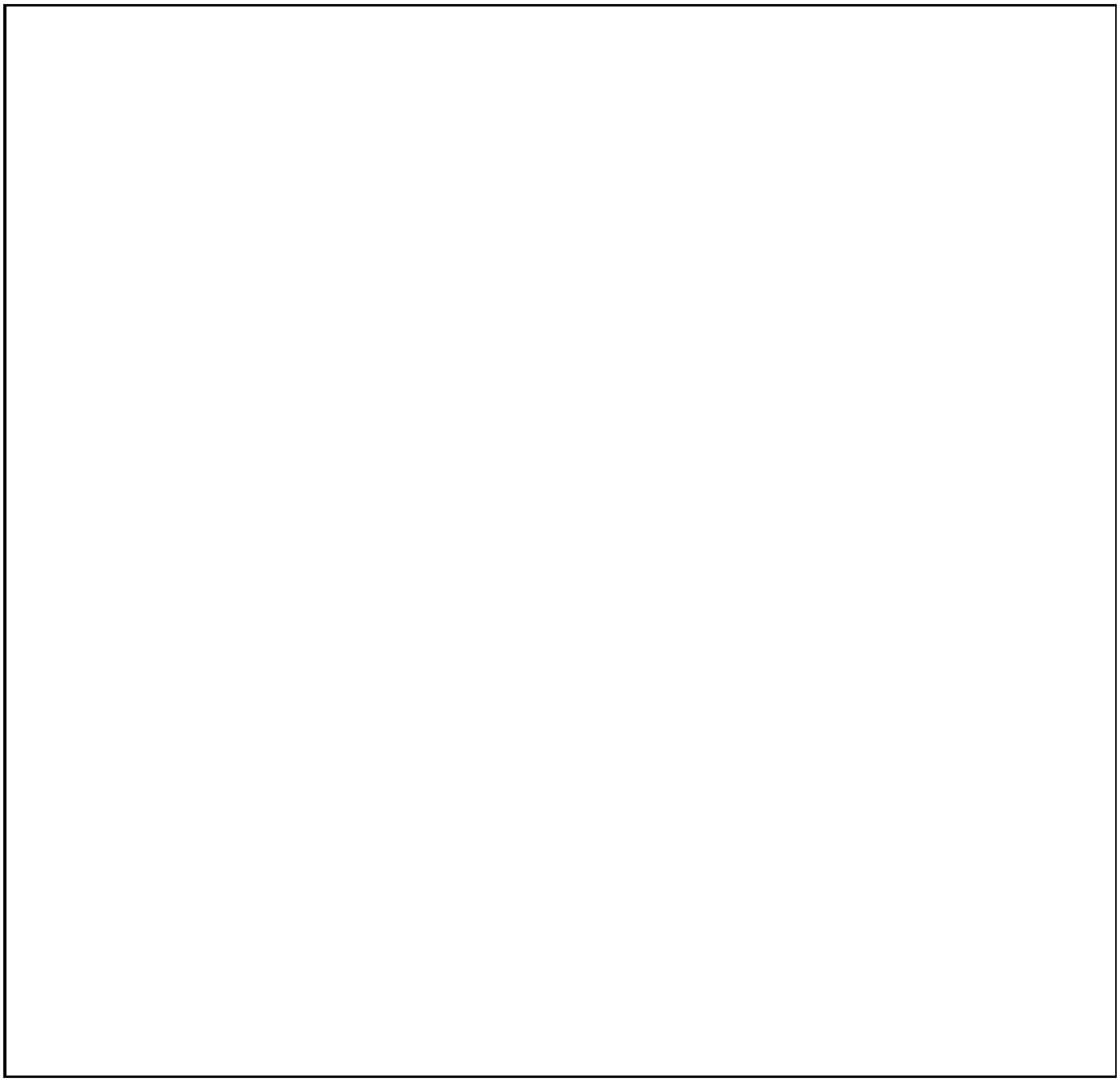
3.6	**Simulate the Errors**
3.6.1	**Syntax Error**

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

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

1	**Problem Statement**
•	
2	**Student Learning Outcomes**
•	
3	**Design of the Program**
3.1	**Description**
3.2	**Algorithm**
3.3	**Coding using C++ Language**

```
1. import java.lang.*;
2. abstract class Employee
3. {
4. String name;
5. double basic_sal;
6. double da;
7. double hra;
8. double it;
9. Employee(String n, double b)
10. {
11. name=n;
12. basic_sal=b;
13. da=basic_sal*0.7;
```

```
14. hra=basic_sal*0.3;
15. it=basic_sal*0.4;
16. }
17. abstract double gross_sal();
18. }
19. class Manager extends Employee
20. {
21. double inc;
22. Manager(String n, double b)
23. {
24. super(n,b);
25. }
26. double gross_sal()
27. {
28. inc=basic_sal*0.1;
29. double gross=basic_sal + da + hra - it + inc;
30. System.out.println("Employee Name : "+name);
31. System.out.println("Designation : Manager ");
32. System.out.println("Basic of Employee : "+basic_sal);
33. System.out.println("DA of Employee : "+da);
34. System.out.println("HRA of Employee : "+hra);
35. System.out.println("IT of Employee : "+it);
36. System.out.println("Annual Increment of Employee : "+inc);
37. return gross;
38. }
39. }
40. public class Technician extends Employee
41. {
42. double inc;
43. Technician(String n, double b)
44. {
```

```
45. super(n,b);
46. }
47. double gross_sal()
48. {
49. inc=basic_sal*0.15;
50. double gross=basic_sal + da + hra - it + inc;
51. System.out.println("Employee Name : "+name);
52. System.out.println("Designation : Technician ");
53. System.out.println("Basic of Employee : "+basic_sal);
54. System.out.println("DA of Employee : "+da);
55. System.out.println("HRA of Employee : "+hra);
56. System.out.println("IT of Employee : "+it);
57. System.out.println("Annual Increment of Employee : "+inc);
58. return gross;
59. }
60. public static void main(String args[])
61. {
62. Manager m = new Manager("anil",4000);
63. Technician t1 = new Technician("Ram",500);
64. Technician t2 = new Technician("Shyam",1000);
65. Technician t3 = new Technician("shiv",1500);
66. Technician t4 = new Technician("gopal",2000);
67. Employee e;
68. e=m;
69. System.out.println("Gross Salary " + e.gross_sal());
70. e=t1;
71. System.out.println("Gross Salary " + e.gross_sal());
72. e=t2;
73. System.out.println("Gross Salary " + e.gross_sal());
74. e=t3;
75. System.out.println("Gross Salary " + e.gross_sal());
```

```
76. e=t4;
77. System.out.println("Gross Salary " + e.gross_sal());
78. }
79. }
```

4	**Expected Results**

$javac Technician.java $java Technician Employee Name : anil
Designation : Manager
Basic of Employee : 4000.0
DA of Employee : 2800.0
HRA of Employee : 1200.0
IT of Employee : 1600.0
Annual Increment of Employee : 400.0
Gross Salary 6800.0
Employee Name : Ram
Designation : Technician
Basic of Employee : 500.0
DA of Employee : 350.0
HRA of Employee : 150.0
IT of Employee : 200.0
Annual Increment of Employee : 75.0
Gross Salary 875.0
Employee Name : Shyam
Designation : Technician
Basic of Employee : 1000.0
DA of Employee : 700.0
HRA of Employee : 300.0
IT of Employee : 400.0
Annual Increment of Employee : 150.0
Gross Salary 1750.0
Employee Name : shiv
Designation : Technician
Basic of Employee : 1500.0
DA of Employee : 1050.0
HRA of Employee : 450.0
IT of Employee : 600.0
Annual Increment of Employee : 225.0
Gross Salary 2625.0
Employee Name : gopal
Designation : Technician
Basic of Employee : 2000.0
DA of Employee : 1400.0

HRA of Employee : 600.0
IT of Employee : 800.0
Annual Increment of Employee : 300.0
Gross Salary 3500.0

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

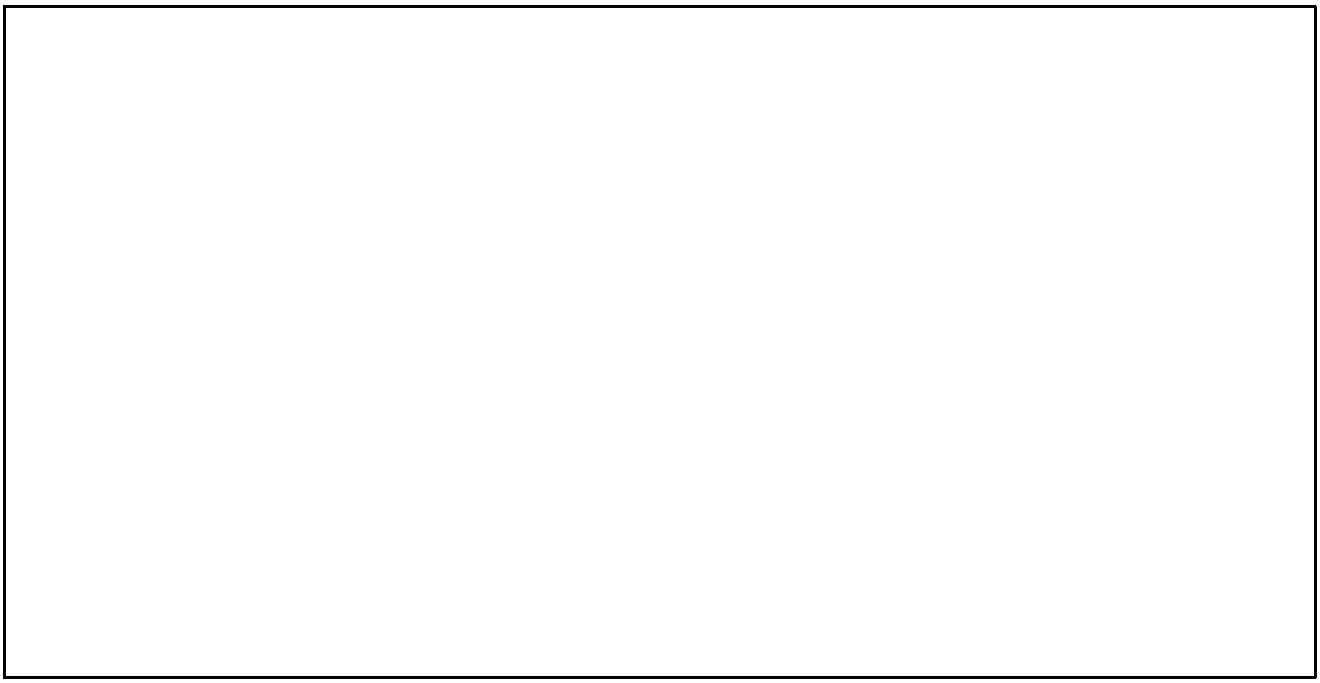
3.6	**Simulate the Errors**
3.6.1	**Syntax Error**

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

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.

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 divide 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**

```
  import java.util.*;
import java.lang.Exception;

import java.util.Scanner;

class OddException extends Exception  // Statement 1
{
   OddException()
   {
      super("Odd number exception");
   }
   OddException(String msg)
   {
      super(msg);
   }
}
```

```
class EvenOdd
{
        void halfOf(int num)
        {
                try
      {
        if(num%2 != 0)
           throw(new OddException());    // Statement 2
        else
        if (num == 0)
                throw(new OddException());
        else
           System.out.print("\n\t" + num + " is an even number and its half is:"+ (num/2));
      }
      catch(OddException Ex)
      {
        System.out.print("\n\tError : " + Ex.getMessage());
      }

      System.out.print("\n\tEnd of program");
   }
}
class EvenOddApp
{
   public static void main(String[] args)
   {
      int num;
     Scanner Sc = new Scanner(System.in);

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

      EvenOdd EO= new EvenOdd();
      EO.halfOf(num);
   }
    1.
```

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

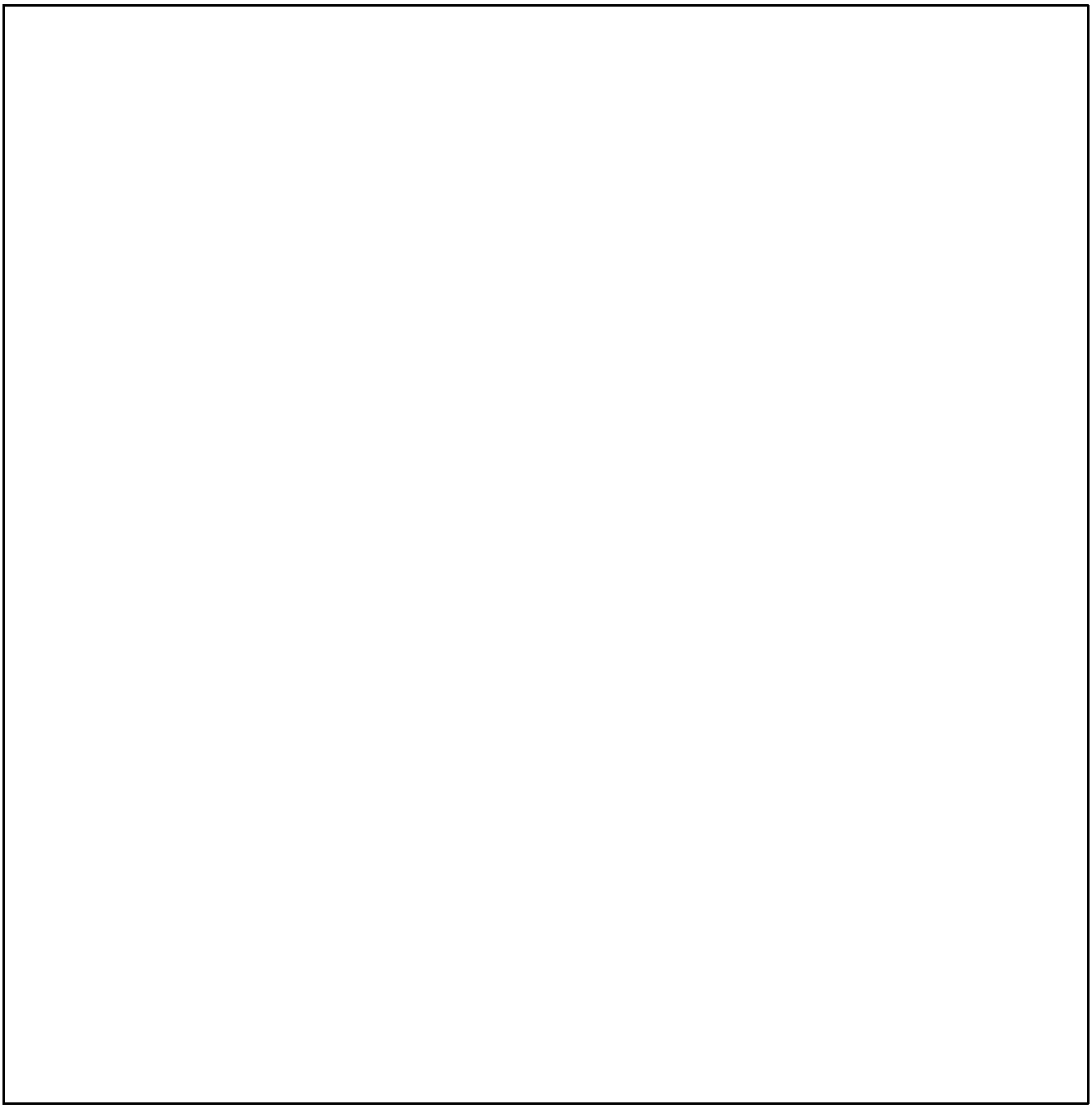
3.6	**Simulate the Errors**
3.6.1	**Syntax Error**

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 9

1	**Problem Statement**

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

2	**Student Learning Outcomes**

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

- .

3	**Design of the Program**
3.1	**Description**

3.2	**Algorithm**

Step 1:

3.3	**Coding using JAVA Language**

```
import java.util.Scanner;


public class Fraction
{
  // member variables
  private int numerator, denominator;  // stores the fraction data
    Scanner myInput = new Scanner( System.in );
```

```
public Fraction()
{
  numerator = denominator = 0;
}


public int getNumerator()
{
  return numerator;
}


public void setNumerator(int num)
{
  numerator=num;
}


public int getDenominator()
{
  return denominator;
}


public void setDenominator(int den)
{
  denominator=den;
}
```

```
public Fraction add(Fraction b)
{
  // check preconditions
  if ((denominator == 0) || (b.denominator == 0))
    throw new IllegalArgumentException("invalid denominator");
  // find lowest common denominator
  int common = lcd(denominator, b.denominator);
  // convert fractions to lcd
  Fraction commonA = new Fraction();
  Fraction commonB = new Fraction();
  commonA = convert(common);
  commonB = b.convert(common);
  // create new fraction to return as sum
  Fraction sum = new Fraction();
  // calculate sum
  sum.numerator = commonA.numerator + commonB.numerator;
  sum.denominator = common;
  // reduce the resulting fraction
  sum = sum.reduce();
  return sum;
}


public Fraction subtract(Fraction b)
{
  // check preconditions
  if ((denominator == 0) || (b.denominator == 0))
    throw new IllegalArgumentException("invalid denominator");
  // find lowest common denominator
  int common = lcd(denominator, b.denominator);
  // convert fractions to lcd
```

```
    Fraction commonA = new Fraction();
    Fraction commonB = new Fraction();
    commonA = convert(common);
    commonB = b.convert(common);
    // create new fraction to return as difference
    Fraction diff = new Fraction();
    // calculate difference
    diff.numerator = commonA.numerator - commonB.numerator;
    diff.denominator = common;
    // reduce the resulting fraction
    diff = diff.reduce();
    return diff;
  }


  public Fraction multiply(Fraction b)
  {
    // check preconditions
    if ((denominator == 0) || (b.denominator == 0))
      throw new IllegalArgumentException("invalid denominator");
    // create new fraction to return as product
    Fraction product = new Fraction();
    // calculate product
    product.numerator = numerator * b.numerator;
    product.denominator = denominator * b.denominator;
    // reduce the resulting fraction
    product = product.reduce();
    return product;
  }
```

```
public Fraction divide(Fraction b)
{
  // check preconditions
  if ((denominator == 0) || (b.numerator == 0))
    throw new IllegalArgumentException("invalid denominator");
  // create new fraction to return as result
  Fraction result = new Fraction();
  // calculate result
  result.numerator = numerator * b.denominator;
  result.denominator = denominator * b.numerator;
  // reduce the resulting fraction
  result = result.reduce();
  return result;
}


public void input()
{
  // prompt user to enter numerator
  System.out.print("Please enter an integer for numerator: ");
  // get user input
  numerator = myInput.nextInt();
  // prompt user to enter denominator in a loop to prevent
  // an invalid (zero) value for denominator
  do {
    System.out.print("Please enter a non-zero integer for denominator: ");
    denominator = myInput.nextInt();
    // make sure it is non-zero
    if (denominator == 0)
      System.out.println("Invalid value.  Please try again.");
  } while (denominator == 0);
```

```
}


public void output()
{
   System.out.print(this);
}


public String toString()
{
  String buffer = numerator + "/" + denominator;
  return buffer;
}


private int lcd(int denom1, int denom2)
{
  int factor = denom1;
  while ((denom1 % denom2) != 0)
    denom1 += factor;
  return denom1;
}


private int gcd(int denom1, int denom2)
{
  int factor = denom2;
  while (denom2 != 0) {
    factor = denom2;
    denom2 = denom1 % denom2;
```

```
      denom1 = factor;
    }
    return denom1;
}


private Fraction convert(int common)
{
    Fraction result = new Fraction();
    int factor = common / denominator;
    result.numerator = numerator * factor;
    result.denominator = common;
    return result;
}


private Fraction reduce()
{
    Fraction result = new Fraction();
    int common = 0;
    // get absolute values for numerator and denominator
    int num = Math.abs(numerator);
    int den = Math.abs(denominator);
    // figure out which is less, numerator or denominator
    if (num > den)
      common = gcd(num, den);
    else if (num < den)
      common = gcd(den, num);
    else  // if both are the same, don't need to call gcd
      common = num;
```

```
    // set result based on common factor derived from gcd
    result.numerator = numerator / common;
    result.denominator = denominator / common;
    return result;
  }


  public static void main(String args[])
  {
    Fraction f1 = new Fraction();   // local fraction objects
    Fraction f2 = new Fraction();   // used to test methods

    // one way to set up fractions is simply to hard-code some values
    f1.setNumerator(1);
    f1.setDenominator(3);
    f2.setNumerator(1);
    f2.setDenominator(6);

    // try some arithmetic on these fractions
    Fraction result = new Fraction();
    // test addition
    result = f1.add(f2);
    // one way to output results, using toString method directly
    System.out.println(f1 + " + " + f2 + " = " + result);
    // test addition going the other way - should be same result
    result = f2.add(f1);
    // output results
    System.out.println(f2 + " + " + f1 + " = " + result);
    System.out.println();

    // test subtraction
```

```
result = f1.subtract(f2);
// output results
System.out.println(f1 + " - " + f2 + " = " + result);
// test subtraction going the other way - should be different result
result = f2.subtract(f1);
// output results
System.out.println(f2 + " - " + f1 + " = " + result);

// another way to set up fractions is to get user input
System.out.println();
System.out.println("Fraction 1:");
f1.input();
System.out.println();
System.out.println("Fraction 2:");
f2.input();
System.out.println();

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

// another way to output results is to use the output method
// this uses the toString method indirectly
f1.output();
System.out.print(" * ");
f2.output();
System.out.print(" = ");
result.output();
System.out.println();

// test division
result = f1.divide(f2);
```

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

}
```

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 10	
1	**Problem Statement**
Create a class Student that has instance variables as Name, Age, Address and access transmutation methods to access the instance variables along with display method to print the details of student. Next write a main() function that will create a collection of 10students and reverse the list. Print the details before and after reversing the collection.	
2	**Student Learning Outcomes**
After successful completion of this lab, the student shall be able to •	
3	**Design of the Program**
3.1	**Description**
3.2	**Algorithm**
Step 1: Start **Step 2:**	
3.3	**Coding using JAVA Language**

```
import java.util.*;

class Student{
private int roll;
private String name;
private int marks;

public void setRoll(int roll){
this.roll = roll;
}
public int getRoll(){
return roll;
}
```

```
public void setName(String name){
this.name = name;
}
public String getName(){
return name;
}

public void setMarks(int marks){
this.marks = marks;
}
public int getMarks(){
return marks;
}
}

public class SortStudents
{
public static void main(String args[]){
List stu= new ArrayList();
Student st1= new Student();
st1.setRoll(101);
st1.setName("Amit");
st1.setMarks(56);
Student st2= new Student();
st2.setRoll(103);
st2.setName("Anil");
st2.setMarks(66);
Student st3= new Student();
st3.setRoll(103);
st3.setName("Ankit");
```

```
st3.setMarks(76);
stu.add(st1);
stu.add(st2);
stu.add(st3);
ListIterator listItr =(ListIterator)stu.listIterator();
while(listItr.hasNext()){
Student stud =(Student)listItr.next();
System.out.print(" "+stud.getRoll());
System.out.print(" "+stud.getName());
System.out.println(" "+stud.getMarks());
}

while(listItr.hasPrevious()){
Student stud =(Student)listItr.previous();
System.out.print(" "+stud.getRoll());
System.out.print(" "+stud.getName());
System.out.println(" "+stud.getMarks());
}

}
}
```

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

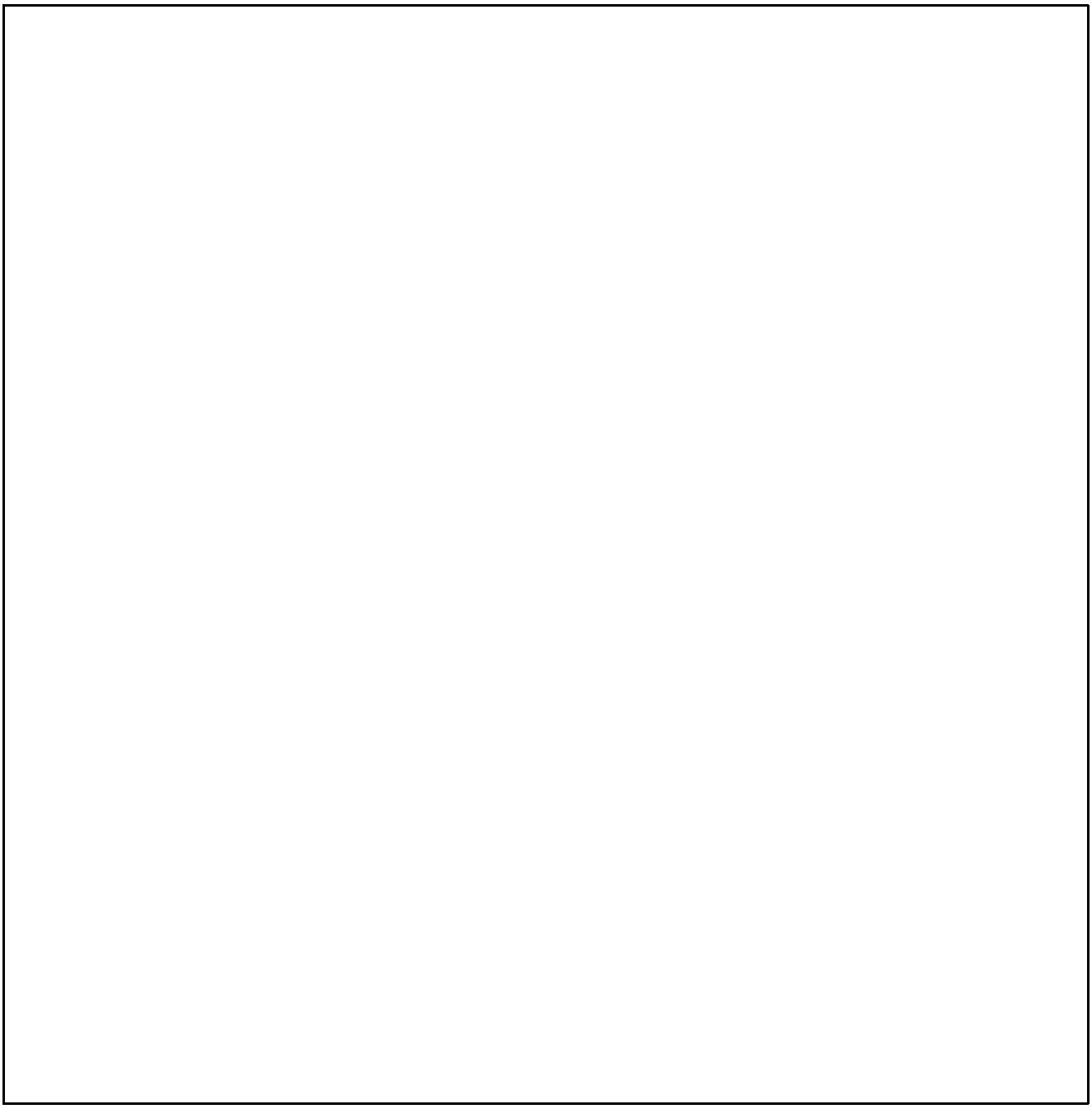
3.6	**Simulate the Errors**
3.6.1	**Syntax Error**

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 11	
1	**Problem Statement**
Use generics to build a class Sort. Implement the bubble sort algorithm to sort an array of any type.	
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:	
3.3	**Coding using JAVA Language**

```
import java.util.Arrays;

public class BubbleSortGeneric<T extends Comparable<? super T>> {
  T[] array;
  BubbleSortGeneric(T[] array){
    this.array = array;
  }

  private T[] bubbleSort(){
```

```
    for(int i = array.length; i > 1; i--){
      for(int j = 0; j < i - 1; j++){
        //if greater swap elements
        if(array[j].compareTo(array[j+1]) > 0){
          swapElements(j, array);
        }
      }
    }
    return array;
  }
  private void swapElements(int index, T[] arr){
    T temp = arr[index];
    arr[index] = arr[index+1];
    arr[index+1] = temp;
  }
  public static void main(String[] args) {
    Integer[] intArr = {47, 85, 62, 34, 7, 10, 92, 106, 2, 54};
    BubbleSortGeneric<Integer> bsg1 = new BubbleSortGeneric<Integer>(intArr);
    Integer[] sa1 = bsg1.bubbleSort();
    System.out.println("Sorted array- " + Arrays.toString(sa1));

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

1.

4	**Expected Results**

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

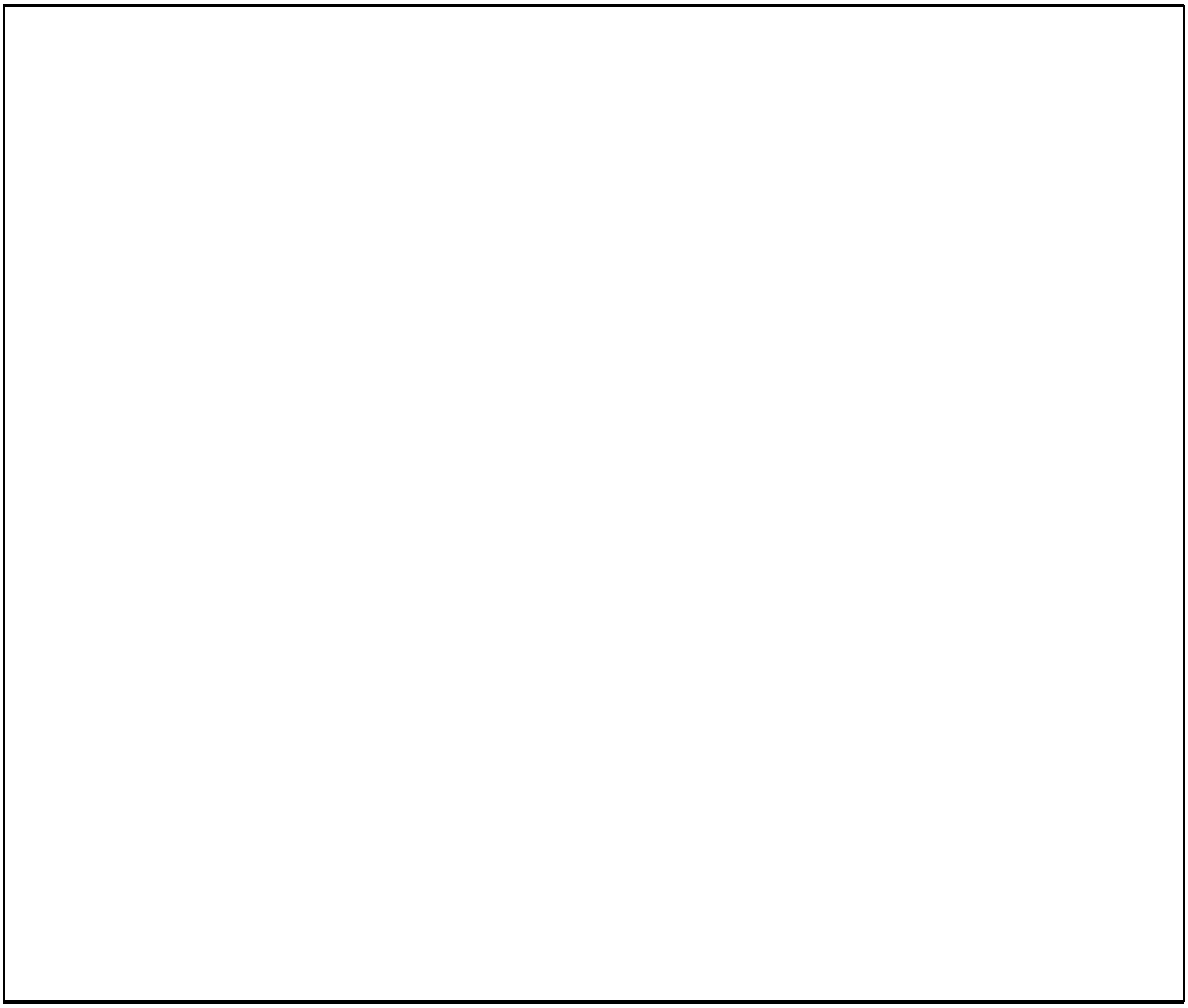
3.6	**Simulate the Errors**
3.6.1	**Syntax Error**

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 12	
1	**Problem Statement**
Write a generic method to count the number of elements in a collection that have a specific property (for example, odd integers, prime numbers, palindromes.	
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

```
import java.util.ArrayList;

public class UpperBoundWildcard {

   private static Double add(ArrayList<? extends Number> num) {
      double sum=0;
      double den =2.0;

      for(Number n:num)
      {
         if((n.doubleValue()%den)!=0)
         {   sum = sum+n.doubleValue();
            System.out.println(n.doubleValue());
         }

      }

      return sum;
   }

   public static void main(String[] args) {

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



   }

}
```

4	Expected Results

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

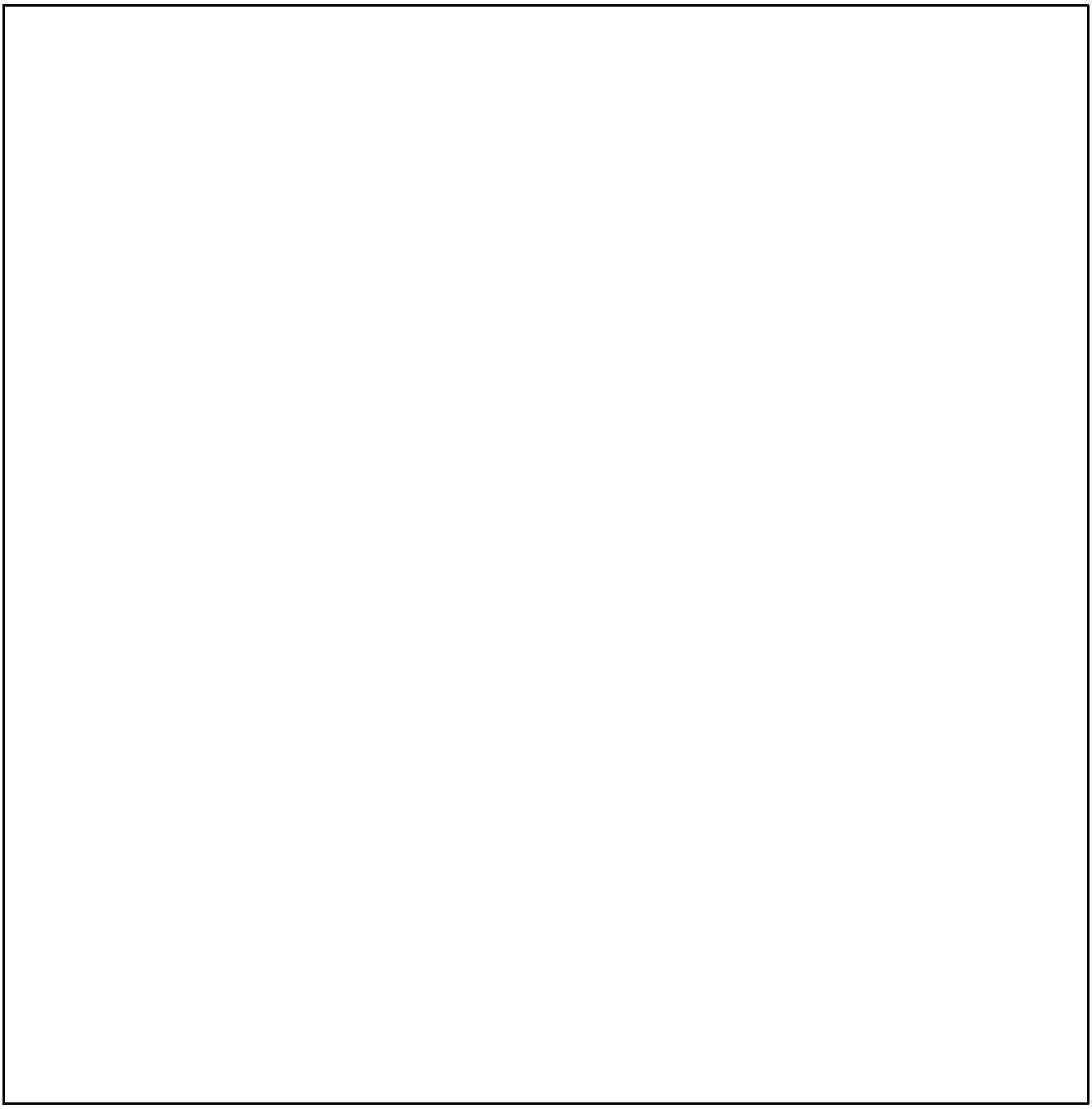
3.6	**Simulate the Errors**
3.6.1	**Syntax Error**

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**



List of Additional Experiments

1. Airline reservation system

One of the best ideas to start experimenting you hands-on Java projects for students is working on Airline reservation system. The proposed airline reservation system is a web-based Java project. It is a comprehensive passenger processing system that includes inventory, fares, e-ticket operations, and online transactions. The main features of the airline reservation system are:

- Reservation and cancellation of the airline tickets.
- Automation of airline system functions.
- Perform transaction management and routing functions.
- Offer quick responses to customers.
- Maintain passenger records and report on the daily business transactions.

This integrated airline reservation management application features an open architecture that encourages the addition of new systems and functionalities. This means that the app can be tweaked to keep up with the dynamic needs of the airline business. If you are looking for cool java projects to add to your resume, this is the one.

The VRS software suite incorporates four key modules, namely, user registration, login, reservation, and cancellation. This is one of the important java projects for beginners and the app allows for all communications to take place through a TCP/IP network protocol, thereby facilitating the usage of intranet and internet communications globally.

2. Course management system

This is an excellent Java projects for beginners. As the name suggests, this course management system is an online management software application designed for educational institutions. The primary goal of the project is to facilitate seamless interaction between students and instructors in schools, colleges, and universities concerning the submission of projects, assignments, thesis, and receiving feedback from instructors. This project has three interlinked modules:

- **Administrator module –** This module is designed exclusively for managing administrative functions like creating accounts for students and instructors, creating the curriculum, coding the subjects, managing the employees, payroll, and so on. Basically, this module lays the groundwork for the other two modules.
- **Students module –** This module is designed for the usage of students. They can log in to their accounts to view their coursework, submit their projects, get feedback from instructors, etc.
- **Instructor module –** This module is for the instructors who can log in to their accounts and check the projects submitted by the students, communicate with the students, and offer guidance to them.

As we mentioned earlier, this project aims to promote the sharing of information between qualified instructors and students via the Internet.

3. Data visualization software

Data visualization is a crucial element in the modern industry driven by Data Science, Business Intelligence, and Business Analytics. It refers to the visual representation of data, either in a graphical or pictorial format. This is an important java projects for beginners. This data

visualization project is all about providing an overview of the design and implementation techniques in data visualization. The objectives of this project are:

- To deliver precise and effective communication of the insights hidden in the data through appropriate graphical or pictorial representations.
- To offer relevant insights into complex datasets for conveying ideas effectively.
- To stimulate the viewer's attention and engagement while communicating accurate information.
- To be functional as well as aesthetically pleasing.

This data visualization software displays the node connectivity in networking in the form of data visualization. You can use a mouse or a trackpad to locate it at different locations. The best part about the project is that you can enhance and tweak the software features and functions according to your requirements. Mentioning Java projects can help your resume look much more interesting than others.

4. Electricity billing system

project is a modern version of the traditional electricity billing system. The main focus of this Java project is to computerize the electricity billing system to make it more seamless, accessible, and efficient. The software calculates the units consumed within a specified time duration and accordingly calculates the amount of money to be paid for those units. This is one of the excellent Java project ideas for beginners. The following features make the electricity billing system more service-oriented and straightforward:

- It features a high-performance speed along with accuracy.
- It allows for seamless data sharing between the electricity office and customers.
- It is protected by high-security measures and controls.
- It includes the necessary provisions for debugging.

Unlike the conventional billing system, this computerized software does not require a large number of human employees to handle and manage the process of bill generation. Once it is installed on the system, it will automatically calculate the units consumed and the bills from time to time and also provide the meter readings to each customer. You can continue to add new features in the system as and when user requirements change.

5. e-Healthcare management system

One of the best ideas to start experimenting you hands-on Java projects for students is working on e-Healthcare management system. The e-Healthcare management system is a web-based project that seeks to provide effective management of employee data and medical data of patients in hospitals and clinics.

Data mining techniques lies at the core of this project, which consists of 2 modules: an administration module and a client module. While the administration module is concerned with Medicare Management that includes healthcare departments, doctors, nurses, ward, and clerks, the client module is for patients.In many Ways business intelligence is revolutionizing healthcare.

The key features of the e-Healthcare management system are:

- It establishes a clear line of contact and communication between doctors and patients.
- It accurately analyses usage percentage of the hospital resources, including laboratory equipment, bed occupation ratio, administration, medicines, etc.
- It leverages the CRISP-DM (standard cross-industry process for data mining) creating an accurate and effective management system.

- It eliminates the problems of missing data and incorrect data.

Through these features, the e-Healthcare management system will help overcome the drawbacks and challenges of the existing healthcare management system. It will allow for the smooth management of hospital staff and quicken the process of delivery of healthcare services.

6. Library management system

This software project is implemented in Java using MS Access database design. It is designed for managing and maintaining libraries in any educational institution through an integrated computerized system. The library management software will allow librarians to operate more productively while handling the typical day-to-day tasks of a library.

In a traditional library management system, everything is done manually. All the library operations and records, including the number of books, genres of books, names of books, records of the students who've issued/returned books, etc., are all done via pen and paper. Naturally, this process requires a significant amount of time, effort, and even human resources. If you are looking for final year java projects, this is perfect for you.

The proposed project seeks to solve all the challenges associated with the traditional library management system. Since it stores and manages all the library records in a computerized database, it eliminates the need for manual record-keeping. The software includes different modules, each of which handles and manages specific library operations. Mentioning Java projects can help your resume look much more interesting than others.

By using this software application, librarians and students need not search the entire library to find a book. They can enter the name and author of the book, and the system will display the list of all the possible books available for that search keyword/phrase. This is one of the best features of this library management software.