DEPSTAR

Devang Patel Institute of Advance Technology and Research

(A Constitute Institute of CHARUSAT)

Certificate

This is to certify that

Mr./Mrs. Kaushal Vora

of 3CSE-2 *Class,*

ID. No. 23DCS144 *has satisfactorily completed*

his/ her term work in Java Programming *for*

the ending in oct 2024 /2025

Date : 16/10/24

Sign. of Faculty

Head of Department

CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY (CHARUSAT)
DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY AND RESEARCH
DEPSTAR

Subject : JAVA PROGRAMMING **Semester:** 3

Subject Code: CSE201 **Academic Year :**2024-25

Course Outcome (COs):

At the end of the course, the students will be able to:

CO1	Comprehend Java Virtual Machine architecture and Java Programming Fundamentals.
CO2	Demonstrate basic problem-solving skills: analyzing problems, modelling a problem as a system of objects, creating algorithms, and implementing models and algorithms in an object-oriented computer language (classes, objects, methods with parameters)
CO3	Design applications involving Object Oriented Programming concepts such as inheritance, polymorphism, abstract classes and interfaces.
CO4	Build and test program using exception handling
CO5	Design and build multi-threaded Java Applications.
CO6	Build software using concepts such as files and collection frameworks.

Bloom's Taxonomy:

Level 1- Remembering
Level 2- Understanding
Level 3- Applying
Level 4- Analyzing
Level 5- Evaluating
Level 6- Creating

CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY (CHARUSAT)
DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY AND RESEARCH
DEPSTAR

Practical List

Sr No.	AIM	Hrs.	CO	Bloom's Taxonomy
PART-I Data Types, Variables, String, Control Statements, Operators, Arrays				
1	Demonstration of installation steps of Java, Introduction to Object Oriented Concepts, comparison of Java with other object-oriented programming languages. Introduction to JDK, JRE, JVM, Javadoc, command line argument. Introduction to Eclipse or NetBeans IDE,or BlueJ and Console Programming.	**2**	**1**	**1**
2	Imagine you are developing a simple banking application where you need to display the current balance of a user account. For simplicity, let's say the current balance is $20. Write a java program to store this balance in a variable and then display it to the user.	**1**	**1**	**2,3,4**
3	Write a program to take the user for a distance (in meters) and the time taken (as three numbers: hours, minutes, seconds), and display the speed, in meters per second, kilometers per hour and miles per hour (hint:1 mile = 1609 meters).	**1**	**1**	**2,3,4**
4	Imagine you are developing a budget tracking application. You need to calculate the total expenses for the month. Users will input their daily expenses, and the program should compute the sum of these expenses. Write a Java program to calculate the sum of elements in an array representing daily expenses. **Supplementary Experiment:** You are creating a library management system. The library has two separate lists of books for fiction and non-fiction. The system should merge these lists into a single list for inventory purposes. Write a Java program to merge two arrays.	**1**	**1, 2**	**2,3**
5	An electric appliance shop assigns code 1 to motor,2 to fan,3 to tube and 4 for wires. All other items have code 5 or more. While selling the goods, a sales tax of 8% to motor,12% to fan,5% to tube light,7.5% to wires and 3% for all other items is charged. A list containing the product code and price in two different arrays. Write a java program using switch statement to prepare the bill.	**1**	**1, 2**	**2**
6	Create a Java program that prompts the user to enter the	**1**	**1, 2**	**2,3,4**

	number of days (n) for which they want to generate their exercise routine. The program should then calculate and display the first n terms of the Fibonacci series, representing the exercise duration for each day. **Supplementary Experiment:** Imagine you are developing a classroom management system. You need to keep track of the grades of students in a class. After collecting the grades, you want to display each student's grade along with a message indicating if they have passed or failed. Let's assume the passing grade is 50.			
	PART-II Strings			
7	Given a string and a non-negative int n, we'll say that the front of the string is the first 3 chars, or whatever is there if the string is less than length 3. Return n copies of the front; front_times('Chocolate', 2) → 'ChoCho' front_times('Chocolate', 3) → 'ChoChoCho' front_times('Abc', 3) → 'AbcAbcAbc'	**1**	**1, 2**	**2,3,4**
8	Given an array of ints, return the number of 9's in the array. array_count9([1, 2, 9]) → 1 array_count9([1, 9, 9]) → 2 array_count9([1, 9, 9, 3, 9]) → 3 **Supplementary Experiment:** 1. Write a Java program to replace each substring of a given string that matches the given regular expression with the given replacement. Sample string : "The quick brown fox jumps over the lazy dog." **In the above string replace all the fox with cat.**	**1**	**1, 2**	**2,3**
9	Given a string, return a string where for every char in the original, there are two chars. double_char('The') → 'TThhee' double_char('AAbb') → 'AAAAbbbb' double_char('Hi-There') → 'HHii--TThheerree'	**1**	**1, 2**	**2**
10	Perform following functionalities of the string: ● Find Length of the String ● Lowercase of the String ● Uppercase of the String ● Reverse String	**1**	**1, 2**	**2,3,4**

	Sort the string			
11	Perform following Functionalities of the string: “CHARUSAT UNIVERSITY” ● Find length ● Replace ‘H’ by ‘FIRST LATTER OF YOUR NAME’ ● Convert all character in lowercase **Supplementary Experiment:** 1. Write a Java program to count and print all duplicates in the input string. Sample Output: The given string is: resource The duplicate characters and counts are: e appears 2 times r appears 2 times	**1**	**1, 2**	**4**
	PART-III Object Oriented Programming: Classes, Methods, Constructors			
12	Imagine you are developing a currency conversion tool for a travel agency. This tool should be able to convert an amount in Pounds to Rupees. For simplicity, we assume the conversion rate is fixed: 1 Pound = 100 Rupees. The tool should be able to take input both from command-line arguments and interactively from the user.	**1**	**2**	**3**
13	Create a class called Employee that includes three pieces of information as instance variables—a first name (type String), a last name (type String) and a monthly salary (double). Your class should have a constructor that initializes the three instance variables. Provide a set and a get method for each instance variable. If the monthly salary is not positive, set it to 0.0. Write a test application named EmployeeTest that demonstrates class Employee’s capabilities. Create two Employee objects and display each object’s yearly salary. Then give each Employee a 10% raise and display each Employee’s yearly salary again.	**2**	**1, 2**	**3**
14	Create a class called Date that includes three pieces of information as instance variables—a month (type int), a day (type int) and a year (type int). Your class should have a constructor that initializes the three instance variables and assumes that the values provided are correct. Provide a set and a get method for each instance variable. Provide a method displayDate that displays the month, day and year separated by forward slashes (/). Write a test application named DateTest that demonstrates class Date’s capabilities.	**2**	**1, 2**	**3**

15	Write a program to print the area of a rectangle by creating a class named 'Area' taking the values of its length and breadth as parameters of its constructor and having a method named 'returnArea' which returns the area of the rectangle. Length and breadth of rectangle are entered through keyboard. **Supplementary Experiment:** 1.Write a Java program to create a class called "Airplane" with a flight number, destination, and departure time attributes, and methods to check flight status and delay. [L:M]	**1**	**1, 2**	**3**
16	Print the sum, difference and product of two complex numbers by creating a class named 'Complex' with separate methods for each operation whose real and imaginary parts are entered by user.	**1**	**1, 2**	**2,3**
	PART-IV Inheritance, Interface, Package			
17	Create a class with a method that prints "This is parent class" and its subclass with another method that prints "This is child class". Now, create an object for each of the class and call 1 - method of parent class by object of parent	**1**	**1, 2, 3**	**3**
18	Create a class named 'Member' having the following members: Data members 1 - Name 2 - Age 3 - Phone number 4 - Address 5 – Salary It also has a method named 'printSalary' which prints the salary of the members. Two classes 'Employee' and 'Manager' inherits the 'Member' class. The 'Employee' and 'Manager' classes have data members 'specialization' and 'department' respectively. Now, assign name, age, phone number, address and salary to an employee and a manager by making an object of both of these classes and print the same.	**2**	**1, 2, 3**	**3**
19	Create a class named 'Rectangle' with two data members 'length' and 'breadth' and two methods to print the area and perimeter of the rectangle respectively. Its constructor having parameters for length and breadth is used to initialize length and breadth of the rectangle. Let class 'Square' inherit the 'Rectangle' class with its constructor having a parameter for its side (suppose s) calling the	**1**	**2,3**	**3**

	constructor of its parent class as 'super(s,s)'. Print the area and perimeter of a rectangle and a square. Also use array of objects. **Supplementary Experiment:** 1.Write a Java program to create a vehicle class hierarchy. The base class should be Vehicle, with subclasses Truck, Car and Motorcycle. Each subclass should have properties such as make, model, year, and fuel type. Implement methods for calculating fuel efficiency, distance traveled, and maximum speed. [L:A]			
20	Create a class named 'Shape' with a method to print "This is This is shape". Then create two other classes named 'Rectangle', 'Circle' inheriting the Shape class, both having a method to print "This is rectangular shape" and "This is circular shape" respectively. Create a subclass 'Square' of 'Rectangle' having a method to print "Square is a rectangle". Now call the method of 'Shape' and 'Rectangle' class by the object of 'Square' class.	**2**	**2,3**	**3**
21	Create a class 'Degree' having a method 'getDegree' that prints "I got a degree". It has two subclasses namely 'Undergraduate' and 'Postgraduate' each having a method with the same name that prints "I am an Undergraduate" and "I am a Postgraduate" respectively. Call the method by creating an object of each of the three classes.	**1**	**2,3**	**3**
22	Write a java that implements an interface AdvancedArithmetic which contains amethod signature int divisor_sum(int n). You need to write a class calledMyCalculator which implements the interface. divisorSum function just takes an integer as input and return the sum of all its divisors. For example, divisors of 6 are 1, 2, 3 and 6, so divisor_sum should return 12. The value of n will be at most 1000. **Supplementary Experiment:** 1.Write a Java programming to create a banking system with three classes - Bank, Account, SavingsAccount, and CurrentAccount. The bank should have a list of accounts and methods for adding them. Accounts should be an interface with methods to deposit, withdraw,	**2**	**2,3**	**2,3**

	calculate interest, and view balances. SavingsAccount and CurrentAccount should implement the Account interface and have their own unique methods. [L:A]			
23	Assume you want to capture shapes, which can be either circles (with a radiusand a color) or rectangles (with a length, width, and color). You also want to be able to create signs (to post in the campus center, for example), each of which has a shape (for the background of the sign) and the text (a String) to put on the sign. Create classes and interfaces for circles, rectangles, shapes, and signs. Write a program that illustrates the significance of interface default method.	**2**	**2,3**	**6**
PART-V Exception Handling				
24	Write a java program which takes two integers x & y as input, you have to compute x/y. If x and y are not integers or if y is zero, exception will occur and you have to report it.	**1**	**4**	**3**
25	Write a Java program that throws an exception and catch it using a try-catch block.	**1**	**4**	**3**
26	Write a java program to generate user defined exception using “throw” and “throws” keyword. Also Write a java that differentiates checked and unchecked exceptions. (Mention at least two checked and two unchecked exceptions in program). **Supplementary Experiment:** 1.Write a Java program that reads a list of integers from the user and throws an exception if any numbers are duplicates. [L:M]	**2**	**4**	**2,3**
	PART-VI File Handling & Streams			
27	Write a program that will count the number of lines in each file that is specified on the command line. Assume that the files are text files. Note that multiple files can be specified, as in "java Line Counts file1.txt file2.txt file3.txt". Write each file name, along with the number of lines in that file, to standard output. If an error occurs while trying to read from one of the files, you should print an error message for that file, but you should still process all the remaining files.	**1**	**4,6**	**3**
28	Write an example that counts the number of times a	**1**	**4,6**	**3**

	particular character, such as e, appears in a file. The character can be specified at the command line. You can use xanadu.txt as the input file.			
29	Write a Java Program to Search for a given word in a File. Also show use of Wrapper Class with an example.	**2**	**4,6**	**3**
30	Write a program to copy data from one file to another file. If the destination file does not exist, it is created automatically. **Supplementary Experiment:** 1.Write a Java program to sort a list of strings in alphabetical order, ascending and descending using streams.	**2**	**4,6**	**3**
31	Write a program to show use of character and byte stream. Also show use of BufferedReader/BufferedWriter to read console input and write them into a file.	**2**	**4,6**	**2,3**
	PART-VII Multithreading			
32	Write a program to create thread which display "Hello World" message. A. by extending Thread class B. by using Runnable interface.	**1**	**5,6**	**3**
33	Write a program which takes N and number of threads as an argument. Program should distribute the task of summation of N numbers amongst number of threads and final result to be displayed on the console.	**1**	**5,6**	**3**
34	Write a java program that implements a multi-thread application that has three threads. First thread generates random integer every 1 second and if the value is even, second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of cube of the number.	**2**	**5,6**	**3**
35	Write a program to increment the value of one variable by one and display it after one second using thread using sleep() method.	**2**	**5,6**	**2,3**
36	Write a program to create three threads 'FIRST', 'SECOND', 'THIRD'. Set the priority of the 'FIRST' thread to 3, the 'SECOND' thread to 5(default) and the 'THIRD' thread to 7.	**2**	**5,6**	**2,3**
37	Write a program to solve producer-consumer problem using thread synchronization.	**2**	**5,6**	**3**
	PART-VIII Collection Framework and Generic			
38	Design a Custom Stack using ArrayList class, which	**2**	**5**	**3**

	implements following functionalities of stack. My Stack -list ArrayList<Object>: A list to store elements. +isEmpty: boolean: Returns true if this stack is empty. +getSize(): int: Returns number of elements in this stack. +peek(): Object: Returns top element in this stack without removing it. +pop(): Object: Returns and Removes the top elements in this stack. +push(o: object): Adds new element to the top of this stack.			
39	Imagine you are developing an e-commerce application. The platform needs to sort lists of products based on different criteria, such as price, rating, or name. Each product object implements the Comparable interface to define the natural ordering. To ensure flexibility and reusability, you need a generic method that can sort any array of Comparable objects. Create a generic method in Java that sorts an array of Comparable objects. This method should be versatile enough to sort arrays of different types of objects (such as products, customers, or orders) as long as they implement the Comparable interface.	**2**	**5**	**6**
40	Write a program that counts the occurrences of words in a text and displays the words and their occurrences in alphabetical order of the words. Using Map and Set Classes.	**2**	**5**	**3**
41	Write a code which counts the number of the keywords in a Java source file. Store all the keywords in a HashSet and use the contains () method to test if a word is in the keyword set.	**2**	**5**	**2,3**

CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY

DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY & RESEARCH

Department of Computer Science & Engineering

Subject Name: Java Programming

Semester : 3
Subject Code: CSE201
Academic year: 2024-25

Part - 1

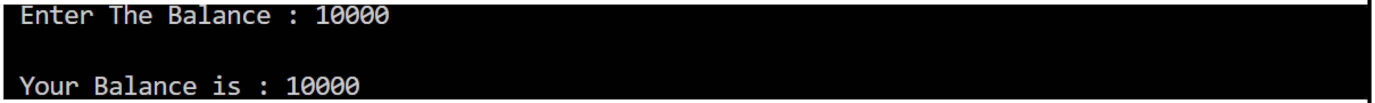
No.	Aim of the Practical
1.	Demonstration of installation steps of Java-Introduction to Object Oriented Concepts, comparison of Java with other object-oriented programming languages. Introduction to JDK, JRE, JVM, Javadoc, command line argument. Introduction to Eclipse or NetBeans IDE,or BlueJ and Console Programming. **<u>CONCLUSION:</u>** Installation of Java : Java SE 17.0. 2 (as of September, 14th 2021) Version : Java SE 11 (LTS) Download Link : https://www.oracle.com/in/java/technologies/javase-jdk11-downlods .html Official Website : https://java.com JDK : JDK contains To develop the Java programs and JRE to run the programs. It Contains compiler (javac.exe) , Java application launcher (java.exe), Applet Viewer. JRE : The JRE is required to run java applications.It combines the Java Virtual Machine , Platform core classes and supporting libraries. JVM : JVM is a virtual machine that enables a computer to run Java programs as well as program written in other languages and compiled to Java Bytecode. Byte code is intermediate representation of java source code.

2. Imagine you are developing a simple banking application where you need to display the current balance of a user account. For simplicity, let's say the current balance is $20. Write a java program to store this balance in a variable and then display it to the user.

PROGRAM CODE :

```
import java.util.Scanner;

class BankPrint{

public static void main(String[] args){

Scanner sc = new Scanner(System.in);

System.out.print(" Enter The Balance : ");

int balance = sc.nextInt();

System.out.println("\n Your Balance is

:"+balance);}
```

OUTPUT:



```
 Enter The Balance : 10000

 Your Balance is : 10000
```

CONCLUSION:

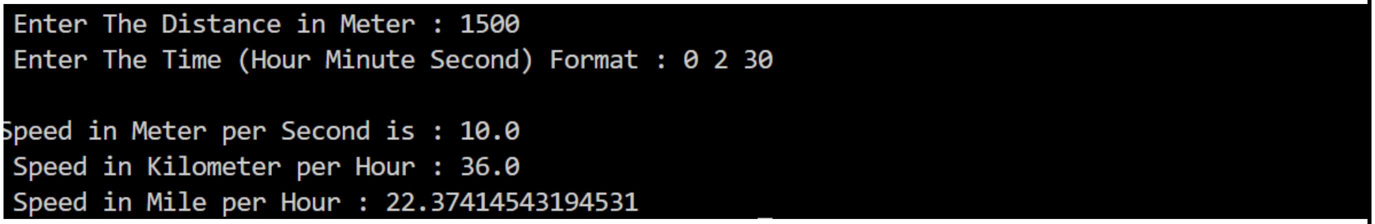
From this program I learned how to take input with the help of Scanner class Object and also taking input from command line argument. And printing The Amount that user inputs.

3. Write a program to take the user for a distance (in meters) and the time taken (as three numbers: hours, minutes, seconds), and display the speed, in meters per second, kilometers per hour and miles per hour (hint:1 mile = 1609 meters).

PROGRAM :

```
import java.util.Scanner;
class MeterPerSecond
{
public static void main(String[] args)
{
Scanner sc = new Scanner(System.in);
System.out.print(" Enter The Distance in Meter : ");
int distance = sc.nextInt();

System.out.print(" Enter The Time (Hour Minute Second) Format : ");
int hour = sc.nextInt();
int minute = sc.nextInt();
int second = sc.nextInt();

int total_second = (hour*3600) + (minute*60) + second ;

double mps = distance/total_second;
double kph = mps*3.6;
double mileph = kph/1.609;

System.out.println(" \nSpeed in Meter per Second is : "+mps);
System.out.println(" Speed in Kilometer per Hour : "+kph);
System.out.println(" Speed in Mile per Hour : "+mileph);
}
}
```

OUTPUT :



```
 Enter The Distance in Meter : 1500
 Enter The Time (Hour Minute Second) Format : 0 2 30

Speed in Meter per Second is : 10.0
 Speed in Kilometer per Hour : 36.0
 Speed in Mile per Hour : 22.37414543194531
```

CONCLUSION :

In this Program I have taken distance in meter and time in (hour minute second) format from user with the help of Scanner scale and calculated speed in m/s , km/h and mile/h.

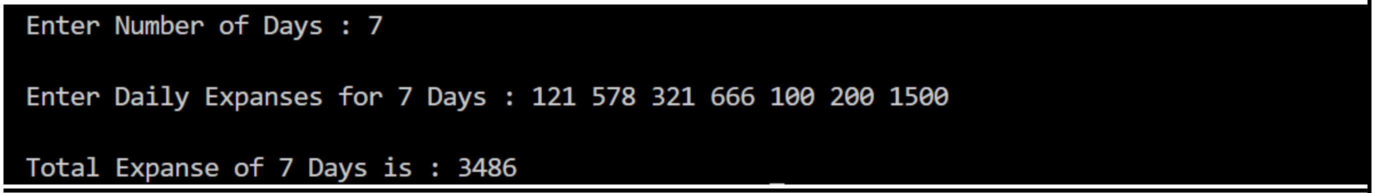
4.

Imagine you are developing a budget tracking application. You need to calculate the total expenses for the month. Users will input their daily expenses, and the program should compute the sum of these expenses. Write a Java program to calculate the sum of elements in an array representing daily expenses.

PROGRAM :

```
import java.util.Scanner;
public class sumofExpanse
{
public static void main(String[] args)
{
Scanner sc = new Scanner(System.in);
System.out.print("\n Enter Number of Days : ");
int days = sc.nextInt();

System.out.print("\n Enter Daily Expanses for "+days+" Days : ");

int[] Expanse = new int[days];
int total=0;
for(int i = 0 ; i<days ; i++)
{
Expanse[i] = sc.nextInt();
total+= Expanse[i];
}
System.out.println("\n Total Expanse of "+days+" Days is : "+total);
}
}
```

OUTPUT :



```
Enter Number of Days : 7

Enter Daily Expanses for 7 Days : 121 578 321 666 100 200 1500

Total Expanse of 7 Days is : 3486
```

CONCLUSION :

From this program I learned about how to store input in an Array and also how to add all the elements of Array in other variable.in this program I've taken an Expanse Array. User will input Number of Days for which he/she want to calculate Total Expanse.

5. An electric appliance shop assigns code 1 to motor,2 to fan,3 to tube and 4 for wires. All other items have code 5 or more. While selling the goods, a sales tax of 8% tomotor,12% to fan,5% to tube light,7.5% to wires and 3% for all other items is charged. A list containing the product code and price in two different arrays. Write a java program using switch statement to prepare the bill.

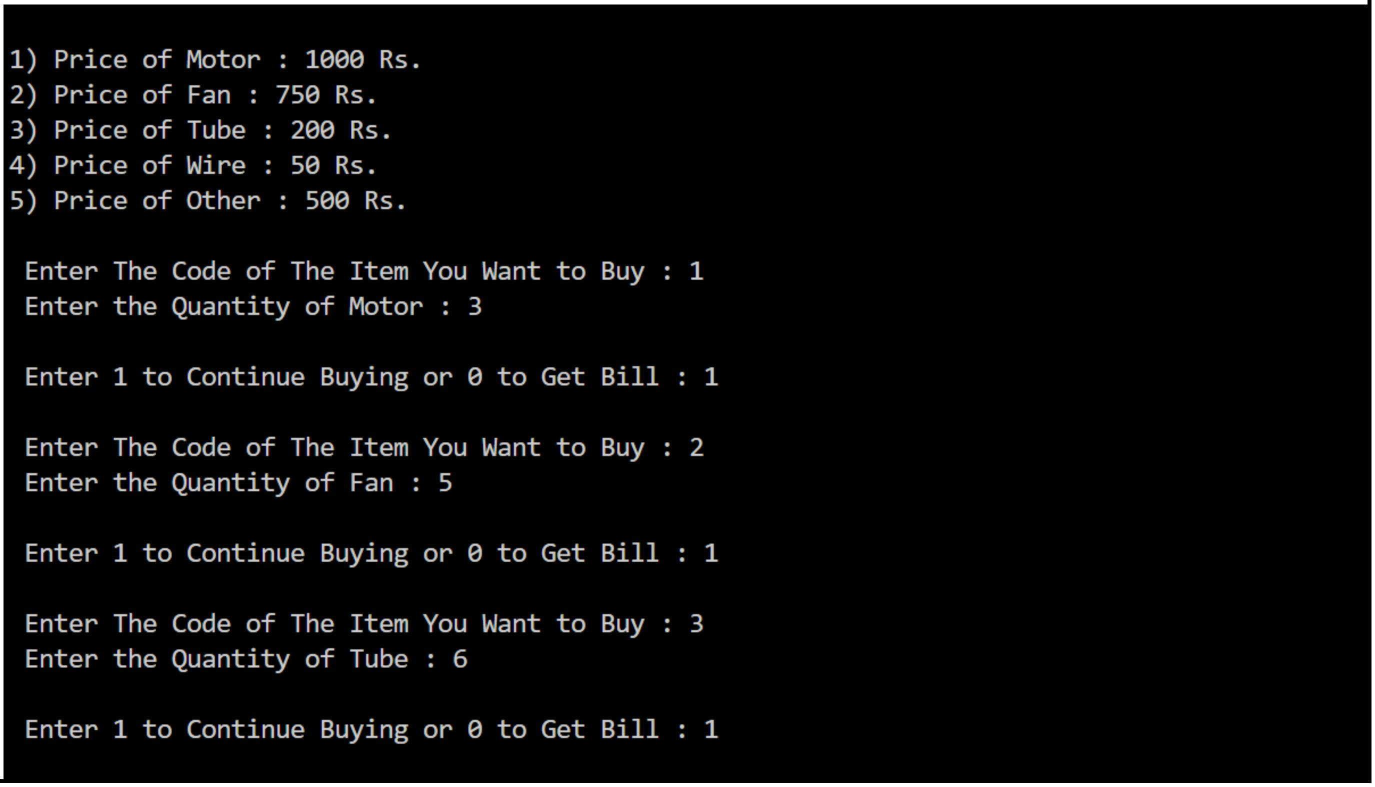
PROGRAM :

```
import java.util.Scanner;
public class Tax
{
public static void main(String[] args)
{
String[] code = {"Motor","Fan","Tube","Wire","Other"};
```

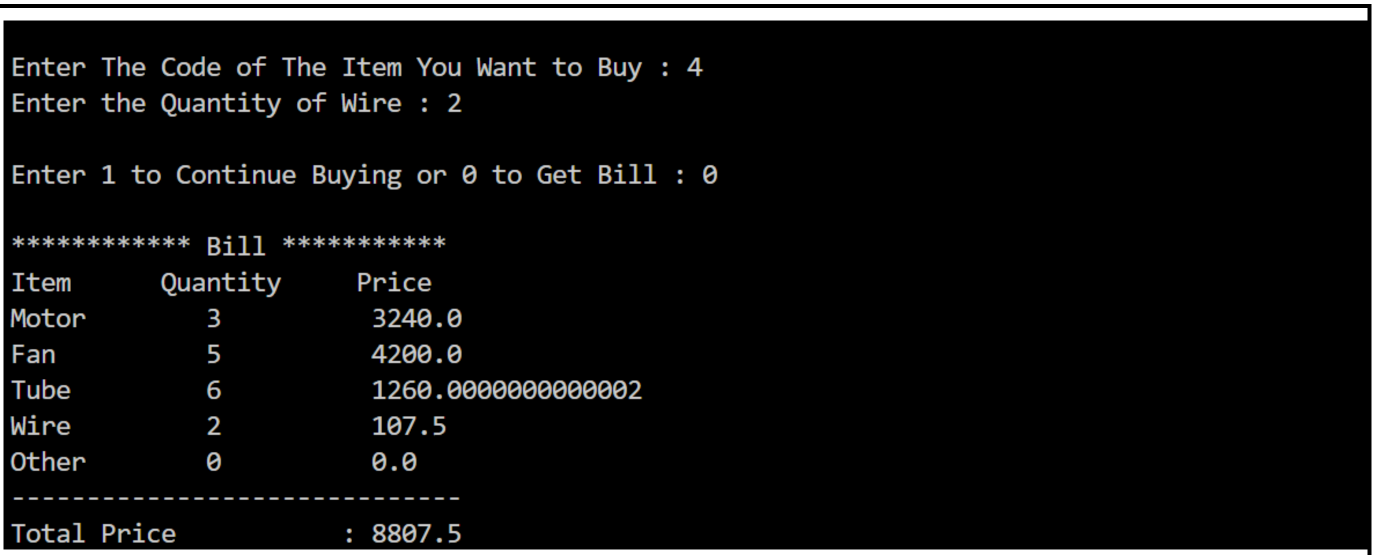
```
int[] price = {1000 , 750 , 200 , 50 , 500};
double[] tax = {1.08, 1.12 , 1.05 , 1.075 , 1.03};
int[] total_item = {0,0,0,0,0};

System.out.println();
for(int i = 0 ; i<code.length ; i++)
{
System.out.println((i+1)+") Price of "+code[i]+" : "+price[i]+" Rs.");
}
System.out.println();

int choice = 1;
Scanner sc = new Scanner(System.in);

while(choice == 1)
{
System.out.print(" Enter The Code of The Item You Want to Buy : ");
int code1 = sc.nextInt();

switch(code1)
{
case 1 :
System.out.print(" Enter the Quantity of Motor : ");
int tmotor = sc.nextInt();
total_item[code1-1]+=tmotor;
break;

case 2 :
System.out.print(" Enter the Quantity of Fan : ");
int tfan = sc.nextInt();
total_item[code1-1]+=tfan;
break;

case 3 :
System.out.print(" Enter the Quantity of Tube : ");
int ttube = sc.nextInt();
total_item[code1-1]+=ttube;
break;

case 4 :
System.out.print(" Enter the Quantity of Wire : ");
int twire = sc.nextInt();
total_item[code1-1]+=twire;
break;

case 5 :
System.out.print(" Enter the Quantity of Other Item : ");
int tother = sc.nextInt();
total_item[code1-1]+=tother;
break;

default :
System.out.println(" Wrong Input!!!");
}
System.out.print("\n Enter 1 to Continue Buying or 0 to Get Bill : ");
choice = sc.nextInt();
System.out.println();
}

double[] bill = new double[5];

for(int i = 0 ; i<bill.length ; i++)
```

```
{
bill[i]=total_item[i]*tax[i]*price[i];
}

System.out.println(" ************ Bill *********** ");
System.out.println(" Item      Quantity     Price");
System.out.println(" Motor       "+total_item[0]+"         "+bill[0]);
System.out.println(" Fan         "+total_item[1]+"         "+bill[1]);
System.out.println(" Tube        "+total_item[2]+"         "+bill[2]);
System.out.println(" Wire        "+total_item[3]+"         "+bill[3]);
System.out.println(" Other       "+total_item[4]+"         "+bill[4]);
System.out.println(" -----------------------------");
System.out.println(" Total Price          : "+(bill[0]+bill[1]+bill[2]+bill[3]+bill[4]));
}}
```

OUTPUT :



```
1) Price of Motor : 1000 Rs.
2) Price of Fan : 750 Rs.
3) Price of Tube : 200 Rs.
4) Price of Wire : 50 Rs.
5) Price of Other : 500 Rs.

 Enter The Code of The Item You Want to Buy : 1
 Enter the Quantity of Motor : 3

 Enter 1 to Continue Buying or 0 to Get Bill : 1

 Enter The Code of The Item You Want to Buy : 2
 Enter the Quantity of Fan : 5

 Enter 1 to Continue Buying or 0 to Get Bill : 1

 Enter The Code of The Item You Want to Buy : 3
 Enter the Quantity of Tube : 6

 Enter 1 to Continue Buying or 0 to Get Bill : 1
```



```
Enter The Code of The Item You Want to Buy : 4
Enter the Quantity of Wire : 2

Enter 1 to Continue Buying or 0 to Get Bill : 0

************ Bill ***********
Item      Quantity     Price
Motor        3          3240.0
Fan          5          4200.0
Tube         6          1260.0000000000002
Wire         2          107.5
Other        0          0.0
-----------------------------
Total Price           : 8807.5
```

CONCLUSION :

In this Program I learned about Switch Statement(i.e. it contains numbers of cases and

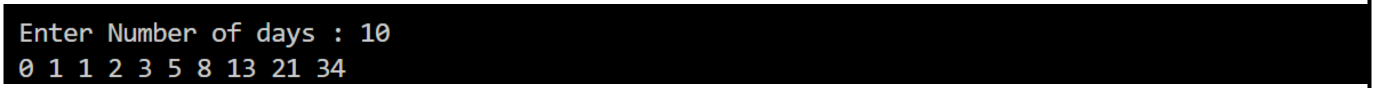
It will execute according to input given to switch if input of switch statement matches with any case then that case executes.

6. Create a Java program that prompts the user to enter the number of days (n) for which they want to generate their exercise routine. The program should then calculate and display the first n terms of the Fibonacci series, representing the exercise duration for each day.

PROGRAM :

```
import java.util.Scanner;
public class Fibo
{
public static void main(String[] args)
{
System.out.println();
System.out.print(" Enter Number of days : ");
Scanner sc = new Scanner(System.in);
int n = sc.nextInt();

int[] fibonacci = new int[n];
fibonacci[0] = 0; fibonacci[1] = 1;

for(int i = 2 ; i<n; i++)fibonacci[i] = fibonacci[i-1] + fibonacci[i-2];
for(int i = 0 ; i<n; i++)System.out.print(" "+fibonacci[i]);
System.out.println();
}
}
```

OUTPUT :



```
Enter Number of days : 10
0 1 1 2 3 5 8 13 21 34
```

CONCLUSION :

In this Fibonacci Series Program I’ve taken user input of how many numbers of Fibonacci series you want to print and then I calculate Fibonacci series by adding last 2 number of Fibonacci series (Example : Fibonacci(3) = Fibonacci(2) + Fibonacci(1)).

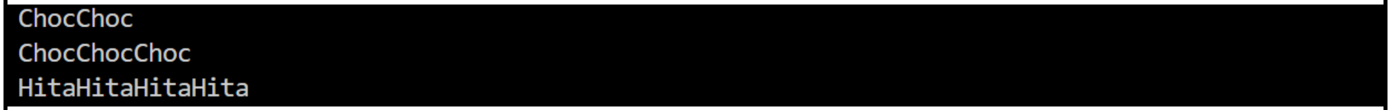
Part - 2

No.	Aim of the Practical
7.	Given a string and a non-negative int n, we'll say that the front of the string is the first 3 chars, or whatever is there if the string is less than length 3. Return n copies of the front; front_times('Chocolate', 2) → 'ChoCho' front_times('Chocolate', 3) → 'ChoChoCho' front_times('Abc', 3) → 'AbcAbcAbc'

PROGRAM :

```
public class ChoCho

{

public static void front_times(String a , int time)

{

int n = 4;

for(int i = 1 ; i<=time ; i++)

{

if(a.length()<n){System.out.println("      Entered

String is Small!! "); break;}

else{String                    sub                    =

a.substring(0,n);System.out.print(sub);}

}

System.out.println();

}
public static void main(String[] args){

String Example = "Hitarth";

int t = 4 ;

front_times("Chocolate",2);

front_times("Chocolate", 3);

front_times(Example,t);}}
```

OUTPUT :



```
ChocChoc
ChocChocChoc
HitaHitaHitaHita
```

CONCLUSION:

In this Java Program we have used the concept of substring and repeating that string for n times.

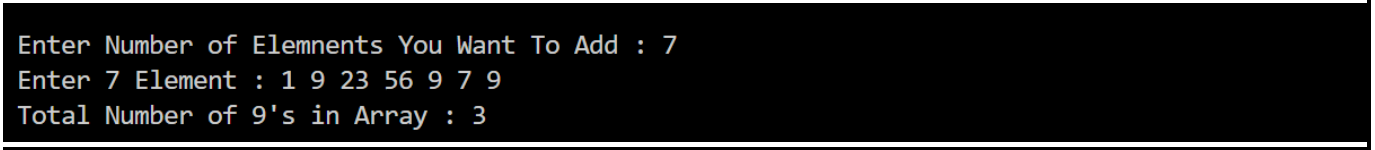
8. Given an array of ints, return the number of 9's in the array.
array_count9([1, 2, 9]) → 1
array_count9([1, 9, 9]) → 2
array_count9([1, 9, 9, 3, 9]) → 3

PROGRAM CODE :

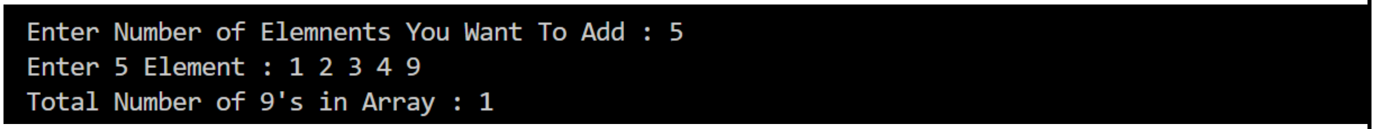
```
import java.util.Scanner;
public class set2Numberof9{
public static int array_count9(int[] array){
int count=0;
for(int i=0;i<array.length;i++) {
if(array[i]==9){ count++;} }
return count;}
public static void main(String[] args){
```

```
Scanner sc = new Scanner(System.in);
System.out.print("\n\n Enter Number of
Elemnents You Want To Add : ");
int n = sc.nextInt();
int[] arr = new int[n];
System.out.print(" Enter "+n+" Element : ");
for(int i = 0 ; i<n; i++) { arr[i]=sc.nextInt(); }
int Total9 = array_count9(arr);
System.out.println(" Total Number of 9's in
Array : "+Total9+"\n\n");}}
```

OUTPUT:



```
Enter Number of Elemnents You Want To Add : 7
Enter 7 Element : 1 9 23 56 9 7 9
Total Number of 9's in Array : 3
```



```
Enter Number of Elemnents You Want To Add : 5
Enter 5 Element : 1 2 3 4 9
Total Number of 9's in Array : 1
```

CONCLUSION:

In this Java Program we have applied the logic for counting the no. of two’s that we have entered in the string.

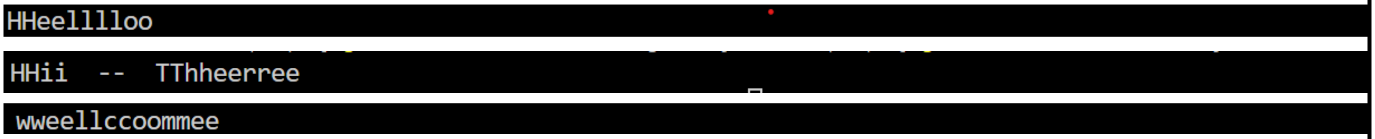
9. Given a string, return a string where for every char in the original, there are two chars.
double_char('The') → 'TThhee'
double_char('AAbb') → 'AAAAbbbb'
double_char('Hi-There') → 'HHii--TThheerree'

PROGRAM :

```
public class set2DoubleChar{
public static String DoublecHar(String input){
String output = "";
for (int i = 0; i < input.length(); i++) { output += input.charAt(i); output += input.charAt(i);}
return output;}
public static void main(String[] args){
```

```
String s = "Hello";
String output = DoublecHar(s);
System.out.println(output);}}
```

OUTPUT :



```
HHeellllloo
HHii  --  TThheerree
wweellccoommee
```

CONCLUSION :

In this java program we have applied logic to double every character of the string and print it.

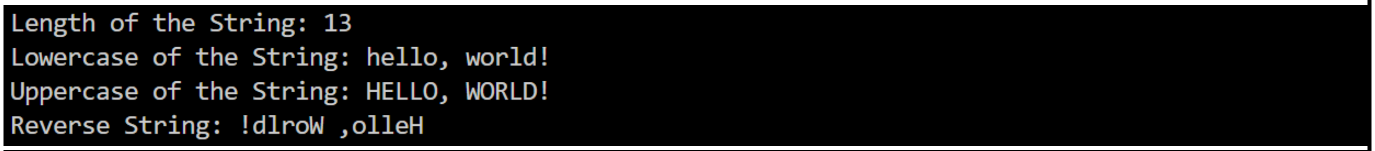
10. Perform following functionalities of the string:

1) Find Length of the String.
2) Lowercase of the String.
3) Uppercase of the String.
4) Reverse String.
5) Sort the String.

PROGRAM :

```
public class StringOperation {
public static void main(String[] args) {
String str = "Hello, World!";

System.out.println("Length of the String: " + findLength(str));

System.out.println("Lowercase of the String: " + toLowercase(str));

System.out.println("Uppercase of the String: " + toUppercase(str));

System.out.println("Reverse String: " + reverseString(str));
}

public static int findLength(String str) { return str.length(); }

public static String toLowercase(String str) { return str.toLowerCase(); }

public static String toUppercase(String str) { return str.toUpperCase(); }

public static String reverseString(String str)
{
StringBuilder sb = new StringBuilder(str);
return sb.reverse().toString();
}
}
```

OUTPUT :



```
Length of the String: 13
Lowercase of the String: hello, world!
Uppercase of the String: HELLO, WORLD!
Reverse String: !dlroW ,olleH
```

CONCLUSION :

In this java program we have applied logic to find the length of the string , convert the string to lowercase , convert to uppercase , and reverse the entered string.

11. Perform following Functionalities of the string: “CHARUSAT UNIVERSITY”

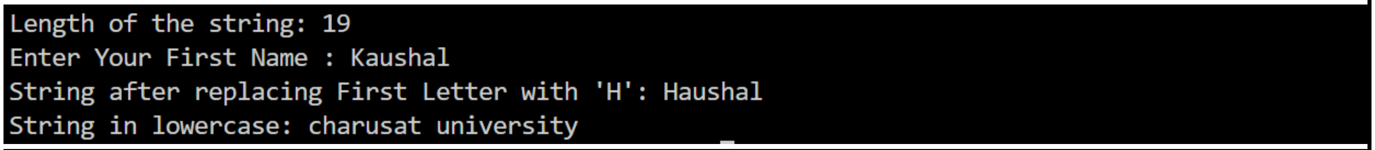
1) Find length
2) Replace ‘H’ by ‘FIRST LATTER OF YOUR NAME’
3) Convert all character in lowercase

PROGRAM :

```
import java.util.Scanner;
public class ReplaceH {
public static void main(String[] args){
```

```
String str = "CHARUSAT UNIVERSITY";

int length = str.length();
System.out.println("Length of the string: " + length);

Scanner sc = new Scanner(System.in);
System.out.print("Enter Your First Name : ");
String name = sc.next();
char first = name.charAt(0);

String replacedStr = name.replace(first, 'H');
System.out.println("String after replacing First Letter with 'H': " + replacedStr);

String lowerCaseStr = str.toLowerCase();
System.out.println("String in lowercase: " + lowerCaseStr);}}
```

OUTPUT :



```
Length of the string: 19
Enter Your First Name : Kaushal
String after replacing First Letter with 'H': Haushal
String in lowercase: charusat university
```

CONCLUSION :

In this java program we have applied the logic to replace the character of the entered string to some other character .

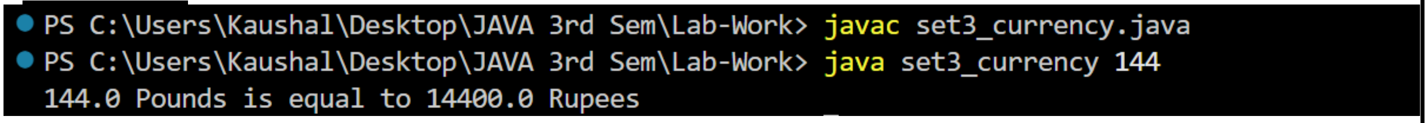
Part - 3

No.	Aim of the Practical
12.	Imagine you are developing a currency conversion tool for a travel agency. This tool should be able to convert an amount in Pounds to Rupees. For simplicity, we assume the conversion rate is fixed: 1 Pound = 100 Rupees. The tool should be able to take input both from command-line arguments and interactively from the user.

PROGRAM CODE :

```
public class set3_currency {
   public static void main(String[] args)
   {
      double amountInPounds = Double.parseDouble(args[0]);
      double amountInRupees = convertPoundsToRupees(amountInPounds);
      System.out.println(amountInPounds+" Pounds is equal to "+amountInRupees+" Rupees");
   }
   public static double convertPoundsToRupees(double amount) {
      return amount * 100;}}
```

OUTPUT:



```
● PS C:\Users\Kaushal\Desktop\JAVA 3rd Sem\Lab-Work> javac set3_currency.java
● PS C:\Users\Kaushal\Desktop\JAVA 3rd Sem\Lab-Work> java set3_currency 144
  144.0 Pounds is equal to 14400.0 Rupees
```

CONCLUSION:

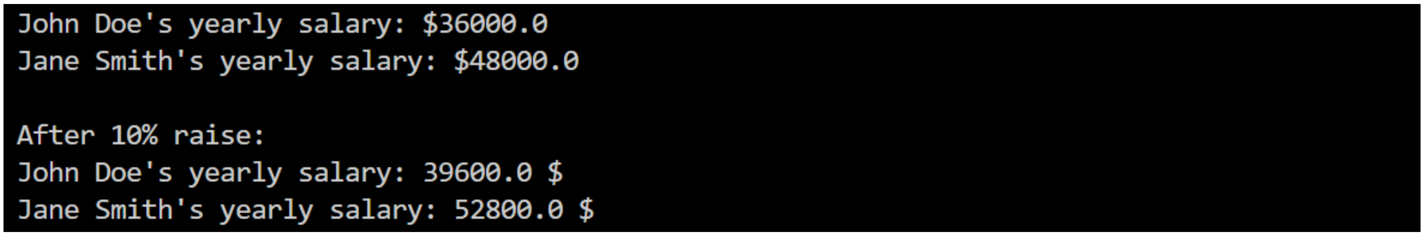
It converts a given amount to a different currency unit by multiplying it with a fixed conversion rate (100). The program utilizes command-line arguments for initial input and the Scanner class for runtime input from the user.

13. Create a class called Employee that includes three pieces of information as instance variables—a first name (type String), a last name (type String) and a monthly salary (double). Your class should have a constructor that initializes the three instance variables. Provide a set and a get method for each instance variable. If the monthly salary is not positive, set it to 0.0. Write a test application named EmployeeTest that demonstrates class Employee's capabilities. Create two Employee objects and display each object's yearly salary. Then give each Employee a 10% raise and display each Employee's yearly salary
again.

PROGRAM CODE:

```
public class set3_2
{
public static void main(String[] args) {
Employee emp1 = new Employee("John", "Doe", 3000);
Employee emp2 = new Employee("Jane", "Smith", 4000);

System.out.println(emp1.getFirstName() + " " + emp1.getLastName() + "'s yearly salary: $" +
emp1.getYearlySalary());
System.out.println(emp2.getFirstName() + " " + emp2.getLastName() + "'s yearly salary: $" +
emp2.getYearlySalary());

emp1.giveRaise(10);
emp2.giveRaise(10);

System.out.println("\nAfter 10% raise:");
System.out.println(emp1.getFirstName() + " " + emp1.getLastName() + "'s yearly salary: " +
emp1.getYearlySalary()+" $");
System.out.println(emp2.getFirstName() + " " + emp2.getLastName() + "'s yearly salary: " +
emp2.getYearlySalary()+" $");}}
class Employee
{
String firstName;
String lastName;
double monthlySalary;

public Employee(String firstName, String lastName, double monthlySalary) {
this.firstName = firstName;
this.lastName = lastName;
if (monthlySalary > 0.0) { this.monthlySalary = monthlySalary; }
else { this.monthlySalary = 0.0; }}

public String getFirstName() { return firstName; }

public void setFirstName(String firstName) { this.firstName = firstName; }

public String getLastName() { return lastName; }

public void setLastName(String lastName) { this.lastName = lastName; }

public double getMonthlySalary() { return monthlySalary; }

public void setMonthlySalary(double monthlySalary) {
if (monthlySalary > 0.0) { this.monthlySalary = monthlySalary; }
else { this.monthlySalary = 0.0; }}

public double getYearlySalary() { return monthlySalary * 12; }

public void giveRaise(double percentage)
{this.monthlySalary += this.monthlySalary * (percentage / 100);}}
```

<u>OUTPUT:</u>



```
John Doe's yearly salary: $36000.0
Jane Smith's yearly salary: $48000.0

After 10% raise:
John Doe's yearly salary: 39600.0 $
Jane Smith's yearly salary: 52800.0 $
```

<u>CONCLUSION:</u>

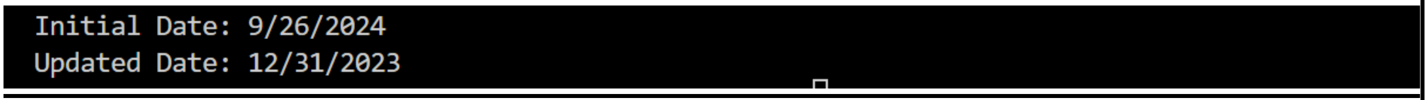
This `Employee` class in Java demonstrates encapsulation by using private fields for first name, last name, and salary, with a constructor for initialization and public methods for controlled access and modification, ensuring data hiding and integrity.

14. Create a class called Date that includes three pieces of information as instance variables—a month (type int), a day (type int) and a year (type int). Your class should have a constructor that initializes the three instance variables and assumes that the values provided are correct. Provide a set and a get method for each instance variable. Provide a method displayDate that displays the month, day and year separated by forward slashes (/). Write a test application named DateTest that demonstrates class Date's capabilities.

PROGRAM:

```
public class set3_3 {

static class Date {
private int month;
private int day;
private int year;

public Date(int month, int day, int year) {
this.month = month;
this.day = day;
this.year = year;
}

public int getMonth() {
return month;
}

public void setMonth(int month) {
this.month = month;
}

public int getDay() {
return day;
}

public void setDay(int day) {
this.day = day;
}

public int getYear() {
return year;
}

public void setYear(int year) {
this.year = year;
}

public void displayDate() {
System.out.println(month + "/" + day + "/" + year);
}
}

public static void main(String[] args) {
Date date = new Date(9, 26, 2024);

System.out.print("Initial Date: ");
date.displayDate();

date.setMonth(12);
date.setDay(31);
date.setYear(2023);

System.out.print("Updated Date: ");
date.displayDate();
}
}
```

OUTPUT:



```
Initial Date: 9/26/2024
Updated Date: 12/31/2023
```

CONCLUSION:

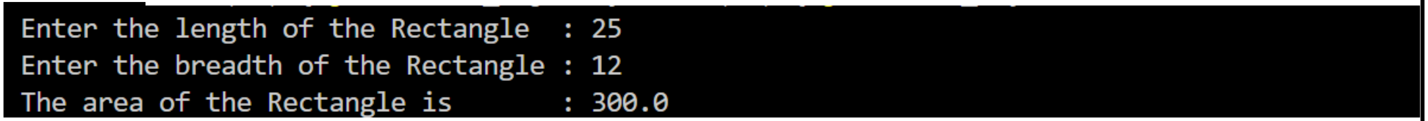
This Java program demonstrates OOP fundamentals by defining a Date class with a constructor for initialization and a method to display the date. The main method creates an instance of Date and calls displayDate.

15. Write a program to print the area of a rectangle by creating a class named 'Area' taking the values of its length and breadth as parameters of its constructor and having a method named 'returnArea' which returns the area of the rectangle. Length and breadth of rectangle are entered through keyboard.

PROGRAM:

```
import java.util.Scanner;
class Area {
private double length;
private double breadth;

public Area(double length, double breadth) {
this.length = length;
this.breadth = breadth;}

public double returnArea() { return length * breadth; }}
public class set3_4{
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);

System.out.print("Enter the length of the Rectangle  : ");
double length = scanner.nextDouble();

System.out.print("Enter the breadth of the Rectangle : ");
double breadth = scanner.nextDouble();

Area rectangle = new Area(length, breadth);

System.out.println("The area of the Rectangle is       : " + rectangle.returnArea());}}
```

OUTPUT:



```
Enter the length of the Rectangle  : 25
Enter the breadth of the Rectangle : 12
The area of the Rectangle is       : 300.0
```

CONCLUSION:

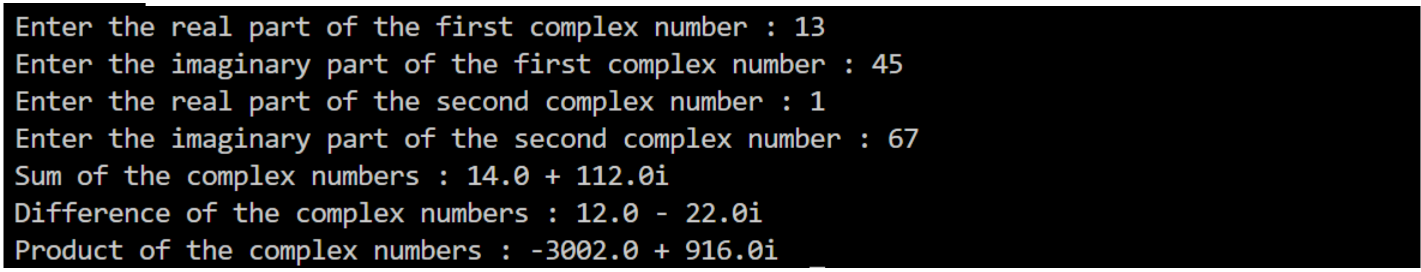
It includes user input handling with the Scanner class to dynamically receive length and breadth values. An Area class instance is created with these values, and its method returnArea() calculates and returns the area of a rectangle.

16. Print the sum, difference and product of two complex numbers by creating a class named ‘Complex’ with separate methods for each operation whose real and imaginary parts are entered by user.

PROGRAM:

```
import java.util.Scanner;
class Complex {
private double real;
private double imaginary;

public Complex(double real, double imaginary) {
this.real = real;
this.imaginary = imaginary;}

public Complex add(Complex other) {
double newReal = this.real + other.real;
double newImaginary = this.imaginary + other.imaginary;
return new Complex(newReal, newImaginary);}

public Complex subtract(Complex other) {
double newReal = this.real - other.real;
double newImaginary = this.imaginary - other.imaginary;
return new Complex(newReal, newImaginary);}

public Complex multiply(Complex other) {
double newReal = this.real * other.real - this.imaginary * other.imaginary;
double newImaginary = this.real * other.imaginary + this.imaginary * other.real;
return new Complex(newReal, newImaginary);}

public void display() {
if (imaginary >= 0)
System.out.println(real + " + " + imaginary + "i");
else System.out.println(real + " - " + (-imaginary) + "i");}}

public class set3_5_complex {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);

System.out.print("Enter the real part of the first complex number : ");
double real1 = scanner.nextDouble();
System.out.print("Enter the imaginary part of the first complex number : ");
double imaginary1 = scanner.nextDouble();

System.out.print("Enter the real part of the second complex number : ");
double real2 = scanner.nextDouble();
System.out.print("Enter the imaginary part of the second complex number : ");
double imaginary2 = scanner.nextDouble();

Complex complex1 = new Complex(real1, imaginary1);
Complex complex2 = new Complex(real2, imaginary2);

Complex sum        = complex1.add(complex2);
Complex difference = complex1.subtract(complex2);
Complex product    = complex1.multiply(complex2);

System.out.print("Sum of the complex numbers : ");
sum.display();

System.out.print("Difference of the complex numbers : ");
difference.display();

System.out.print("Product of the complex numbers : ");
product.display();}}
```

OUTPUT:



```
Enter the real part of the first complex number : 13
Enter the imaginary part of the first complex number : 45
Enter the real part of the second complex number : 1
Enter the imaginary part of the second complex number : 67
Sum of the complex numbers : 14.0 + 112.0i
Difference of the complex numbers : 12.0 - 22.0i
Product of the complex numbers : -3002.0 + 916.0i
```

CONCLUSION:

This Java program defines a `complex` class to handle complex numbers, including methods for setting real and imaginary parts and summing two complex numbers. The main method creates two `complex` objects, sets their values, and sums them, demonstrating basic OOP principles.

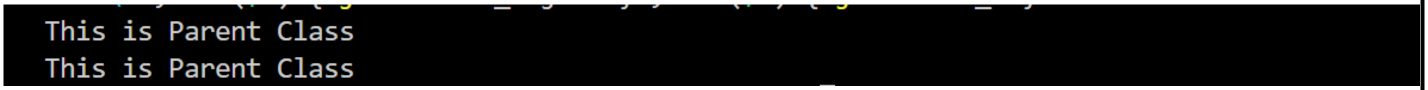
Part - 4

No.	Aim of the Practical
17.	Create a class with a method that prints "This is parentclass" and its subclass with another method that prints "This is child class". Now, create an object for each of theclass and call 1 - method of parent class by object of parent.

PROGRAM CODE:

```
class parent{
public void parentprint(){
   System.out.println(" This is Parent Class ");
}
}
class child extends parent{
public void childprint(){
   System.out.println(" This is Child Class");
}
}
public class set4_1 {
public static void main(String[] args) {
   parent p=new parent();
   child c=new child();
   p.parentprint();
   c.parentprint();
}
}
```

OUTPUT:



```
 This is Parent Class
 This is Parent Class
```

CONCLUSION:

By this experiment I learnt how to inherit classes in java .

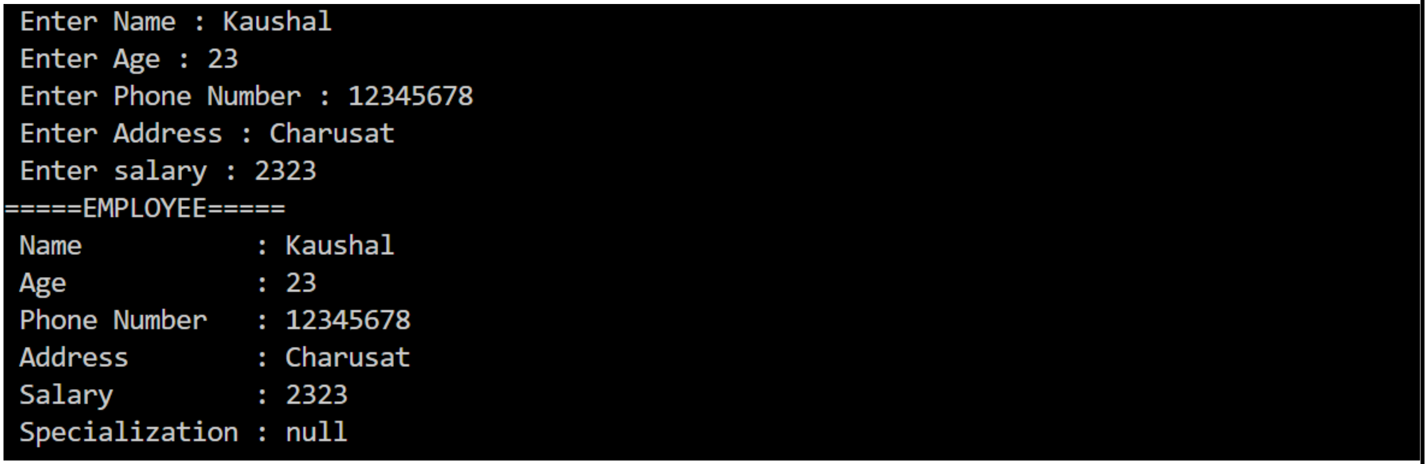
18. Create a class named 'Member' having the following
members: Data members
1 - Name
2 - Age
3 - Phone number
4 - Address
5 – Salary
It also has a method named 'printSalary' which prints thesalary of the members. Two classes 'Employee' and 'Manager' inherits the 'Member' class. The 'Employee' and'Manager' classes have data members 'specialization' and'department' respectively. Now, assign name, age, phonenumber, address and salary to an employee and a managerby making an object of both of these classes and print thesame.

PROGRAM CODE:

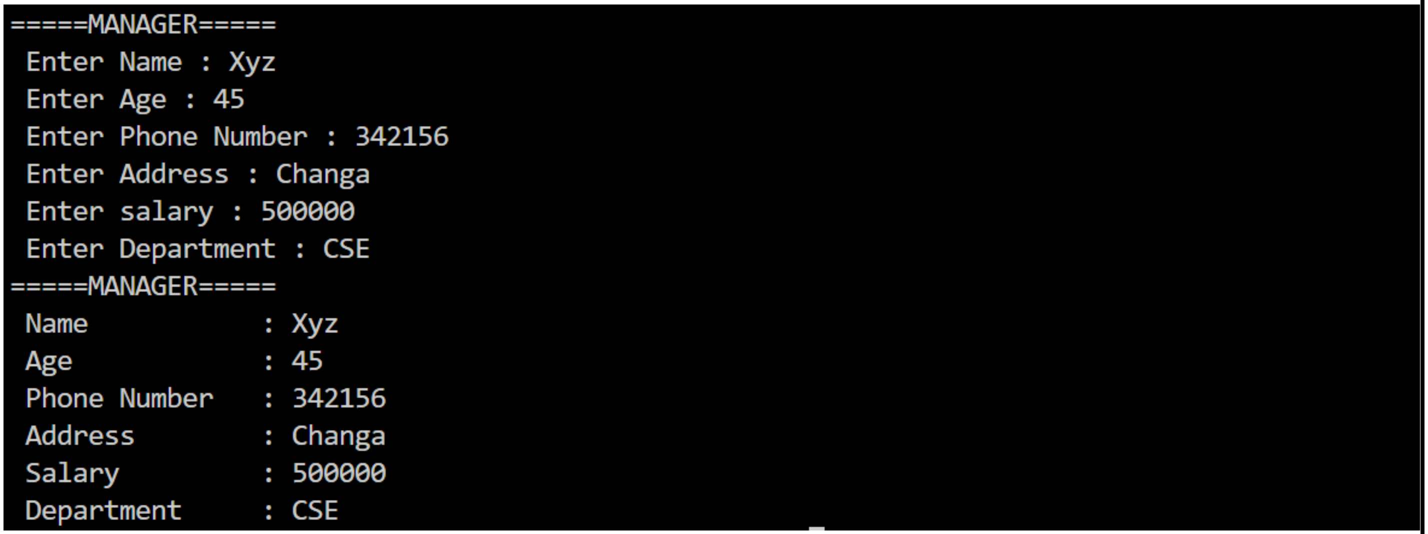
```
import java.util.Scanner;
class Member {

String name, address;
int age, salary;
int pn;

Scanner sc = new Scanner(System.in);

public void getinfo() {

System.out.print(" Enter Name : ");
name = sc.nextLine();
System.out.print(" Enter Age : ");
age = sc.nextInt();
sc.nextLine();
System.out.print(" Enter Phone Number : ");
pn = sc.nextInt();
sc.nextLine();
System.out.print(" Enter Address : ");
address = sc.nextLine();
System.out.print(" Enter salary : ");
salary = sc.nextInt();
}

public void printSalary() {
System.out.println(" Salary         : " + salary);
}

public  void print() {
System.out.println(" Name           : " + name);
System.out.println(" Age            : " + age);
System.out.println(" Phone Number   : " + pn);
System.out.println(" Address        : " + address);
printSalary();
}
}

class Employee extends Member {

String specialization;
Scanner sc = new Scanner(System.in);
public void getemp() {
System.out.println("=====EMPLOYEE=====");
getinfo();
System.out.print(" Enter Specialization : ");
specialization = sc.nextLine();
}

public void printemp(){
System.out.println("=====EMPLOYEE=====");
print();
System.out.println(" Specialization : "+specialization);
System.out.println();
}
}

class Manager extends Member {

String depart;

public void getmanager() {
System.out.println("=====MANAGER=====");
getinfo();
sc.nextLine();
System.out.print(" Enter Department : ");
depart = sc.nextLine();
}
public void printmanager(){
System.out.println("=====MANAGER=====");
print();
System.out.println(" Department     : "+depart);
}
}

public class set4_2 {

public static void main(String[] args) {
Manager m=new Manager();
```

```
Employee e=new Employee();
e.getinfo();
e.printemp();
m.getmanager();
m.printmanager();
}
}
```

OUTPUT:



```
 Enter Name : Kaushal
 Enter Age : 23
 Enter Phone Number : 12345678
 Enter Address : Charusat
 Enter salary : 2323
=====EMPLOYEE=====
 Name           : Kaushal
 Age            : 23
 Phone Number   : 12345678
 Address        : Charusat
 Salary         : 2323
 Specialization : null
```



```
=====MANAGER=====
 Enter Name : Xyz
 Enter Age : 45
 Enter Phone Number : 342156
 Enter Address : Changa
 Enter salary : 500000
 Enter Department : CSE
=====MANAGER=====
 Name           : Xyz
 Age            : 45
 Phone Number   : 342156
 Address        : Changa
 Salary         : 500000
 Department     : CSE
```

CONCLUSION:

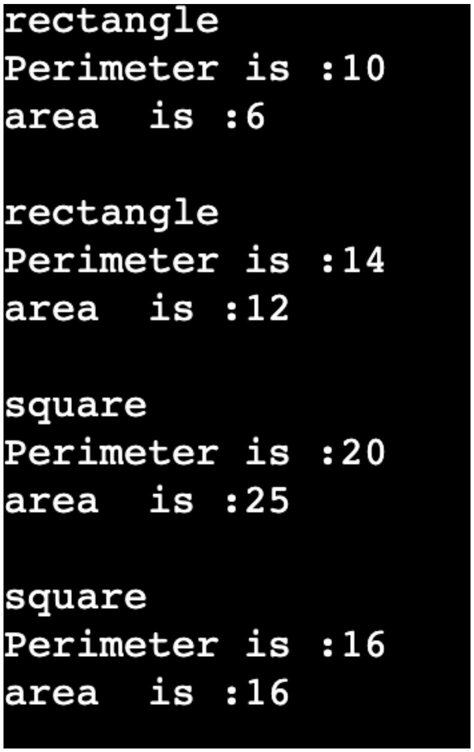
By this experiment I learnt how to do multilevel inheritance in java .

19. Create a class named 'Rectangle' with two data members 'length' and 'breadth' and two methods to print the area and perimeter of the rectangle respectively. Its constructor having parameters for length and breadth is used toinitialize length and breadth of the rectangle. Let class 'Square' inherit the 'Rectangle' class with its constructorhaving a parameter for its side (suppose s) calling the constructor of its parent class as 'super(s,s)'. Print the areaand perimeter of a rectangle and a square. Also use array of objects.

PROGRAM:

```
import java.util.*;

public class set4_3
{

int length;
int breadth;

set4_3(int length,int breadth)
{
this.length=length;
this.breadth=breadth;
}
void perimeter()
{

int c=2*(length+breadth);
System.out.print("Perimeter is :"+c);
System.out.println();

}
void area()
{
int a=length*breadth;
System.out.print("area  is :"+a);
System.out.println();
}
public static void main(String[]args)
{
set4_3 obj[]=new set4_3 [4];
obj[0]=new set4_3 (2,3);
obj[1]=new set4_3 (3,4);
obj[2]=new square(5);
obj[3]=new square(4);

for(set4_3  obj1:obj)
{
if(obj1 instanceof square){System.out.println("square");}
else{System.out.println("rectangle");}
obj1.perimeter();
obj1.area();
System.out.println();
}}}
class square extends set4_3
{
int side;
square(int side ) { super(side, side );}}
```

OUTPUT:



```
rectangle
Perimeter is :10
area  is :6

rectangle
Perimeter is :14
area  is :12

square
Perimeter is :20
area  is :25

square
Perimeter is :16
area  is :16
```

CONCLUSION:

By this experiment I learnt array of objects and inheritance in java .

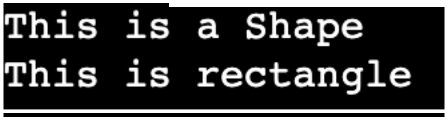
20. Create a class named 'Shape' with a method to print "Thisis This is shape". Then create two other classes named ‘Rectangle', 'Circle' inheriting the Shape class, bothhaving a method to print "This is rectangular shape" and"This is circular shape" respectively. Create a subclass'Square' of 'Rectangle' having a method to print "Squareis a rectangle". Now call the method of 'Shape' and'Rectangle' class by the object of 'Square' class.

PROGRAM:

```
import java.util.*;
public class set4_4 {
void shape()
{
System.out.println(" This is a Shape");
}

public static void main(String[]args)
{
square obj=new square();
obj.shape();
obj.printrectangle();
}
}

class rectangle extends set4_4
{
void printrectangle()
{
System.out.println(" This is rectangle");
}

}

class square extends rectangle  {
void printsquare()
{
System.out.println("rectangle is square");
}
}

class circle extends set4_4
{
```

```
void printcircle()
{
System.out.println("This is circle");
}
}
```

OUTPUT:



```
This is a Shape
This is rectangle
```

CONCLUSION:

By this experiment I learnt how to use multilevel inheritance in java.

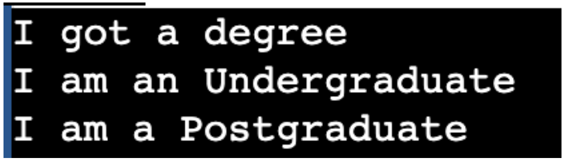
21. Create a class 'Degree' having a method 'getDegree' that prints "I got a degree". It has two subclasses namely' Undergraduate' and 'Postgraduate' each having a method with the same name that prints "I am an Undergraduate "and "I am a Postgraduate" respectively. Call the methodby creating an object of each of the three classes.

PROGRAM:

```
import java.util.*;

public class set4_5
{
void getdegree()
{
System.out.println("I got a degree");
}

public static void main(String[]args)
{
set4_5 obj=new set4_5 ();
obj.getdegree();
Undergraduate obj1=new Undergraduate();
obj1.getdegree();
Postgraduate obj2 =new Postgraduate();
obj2.getdegree();
}
}

class Undergraduate extends set4_5
{
void getdegree()
{
```

```
System.out.println("I am an Undergraduate");
}
}

class Postgraduate extends set4_5
{
void getdegree()
{
System.out.println("I am a Postgraduate");
}
}
```

OUTPUT:



```
I got a degree
I am an Undergraduate
I am a Postgraduate
```

CONCLUSION:

By this experiment I learnt how to call the method by the object of each class.

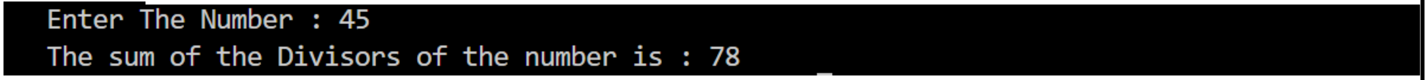
22. Write a java that implements an interfaceAdvancedArithmetic which contains a method signature int divisor_sum(int n). You need to write a classcalledMyCalculator which implements the interface.
divisorSum function just takes an integer as input and return the sum of all its divisors.
For example, divisors of 6 are 1, 2, 3 and 6, so divisor_sum should return 12. The value of n will be at most 1000.

PROGRAM:

```
import java.util.*;

interface AdvancedArithmetic { int divisor_sum(int n); }

class calledMyCalculator implements AdvancedArithmetic
{
public int  divisor_sum(int n)
{
int sum = 0;

for(int i = 1; i<= n; i++)
{ if(n % i == 0) { sum += i; } }
return sum;
}
}
public class set4_6
{
```

```
public static void main(String[] args)
{
Scanner sc = new Scanner(System.in);
calledMyCalculator s = new calledMyCalculator();
System.out.print(" Enter The Number : ");
int n = sc.nextInt();
System.out.println(" The sum of the Divisors of the number is : " + s.divisor_sum(n));
}
}
```

OUTPUT:



```
Enter The Number : 45
The sum of the Divisors of the number is : 78
```

CONCLUSION:

The code implements the AdvancedArithmetic interface, with MyCalculator calculating the sum of divisors of a number. In the main() method, the user inputs a number, and the program prints the sum of its divisors.

23. Assume you want to capture shapes, which can be eithercircles (with a radiusand a color) or rectangles (with alength, width, and color). You also want to be able tocreate signs (to post in the campus center, for example),each of which has a shape (for the background of the sign)and the text (a String) to put on the sign. Create classesand interfaces for circles, rectangles, shapes, and signs.Write a program that illustrates the significance ofinterface default method.

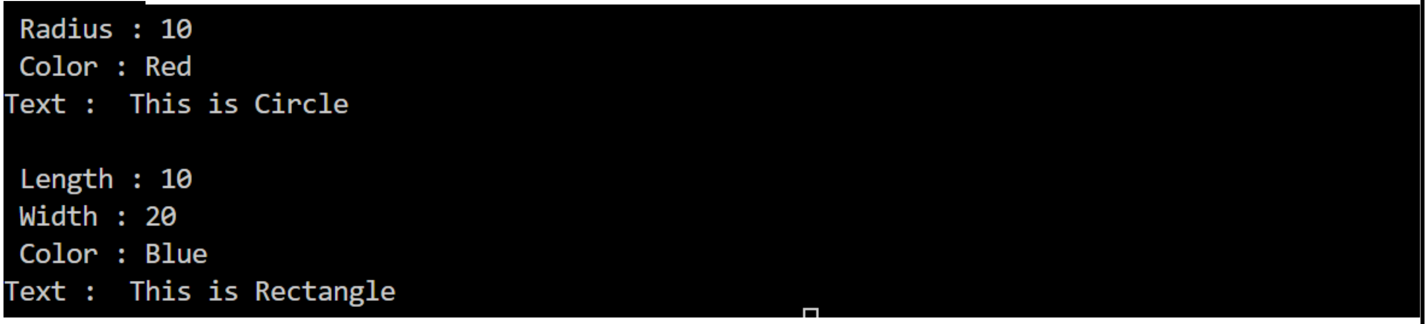
PROGRAM:

```
interface Shape { void print(); }
class Circle implements Shape {
   int radius;
   String color;
   Circle(int radius, String color)
   {
      this.radius = radius;
      this.color = color;
   }

   public void print()
   {
      System.out.println("\n Radius : " + radius + "\n Color : " + color);
   }
}
```

```
class Rectangle implements Shape
{
    int length;
    int width;
    String color;

    Rectangle(int length, int width, String color)
    {
        this.length = length;
        this.width = width;
        this.color = color;
    }

    public void print()
    {
        System.out.println("\n Length : " + length + "\n Width : " + width + "\n Color : " + color);
    }
}

class Sign
{
    Shape s;
    String text;

    Sign(Shape s, String text)
    {
        this.s = s;
        this.text = text;
    }

    void print()
    {
        s.print();
        System.out.println("Text : " + text);
    }
}
public class set4_7
{
    public static void main(String[] args)
    {
        Circle c = new Circle(10, "Red");
        Rectangle r = new Rectangle(10, 20, "Blue");
        Sign s = new Sign(c, " This is Circle");

        s.print();
        Sign s1 = new Sign(r, " This is Rectangle");
        s1.print();
```

```
  }
}
```

OUTPUT:



```
 Radius : 10
 Color : Red
Text :  This is Circle

 Length : 10
 Width : 20
 Color : Blue
Text :  This is Rectangle
```

CONCLUSION:

This program showcases the use of interfaces and polymorphism in Java by defining a Shape interface with methods to get color and calculate area. It demonstrates how different shapes like Circle and Rectangle implement this interface.

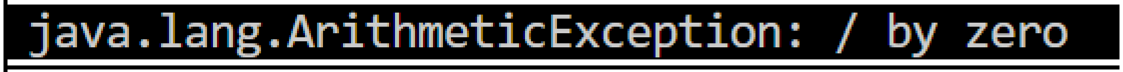
Part - 5

No.	Aim of the Practical
24.	Write a java program which takes two integers x & y as input, you have to compute x/y. If x and y are not integers or if y is zero, exception will occur and you have to report it.

PROGRAM :

```
public class ExceptionH
 {
 public static void main(String[] args)
 {
 int a=5 ; int b=0 ;
 try{ int c = a/b; }
 catch(Exception e) { System.out.println(e); }}}
```

OUTPUT :



```
java.lang.ArithmeticException: / by zero
```

CONCLUSION:

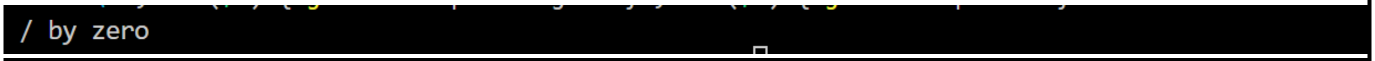
This Java program takes two integer inputs from the user and performs division, handling exceptions for invalid input and division by zero. It ensures the program doesn’t crash by providing appropriate error messages for these cases.

25. Write a Java program that throws an exception and catch it using a try-catch block.

PROGRAM CODE :

```
public class ExceptionH
{
public static void division() throws
ArithmeticException
{
int a = 10;
int b= 0;
int c = a/b;

}

public static void main(String[] args)
{
try{division();}
catch(ArithmeticException
e){System.out.println(e.getMessage());}
}
}
```

OUTPUT:



```
/ by zero
```

CONCLUSION:

This enhanced code includes comments, an additional finally block for code that should always be executed, and more descriptive error messages

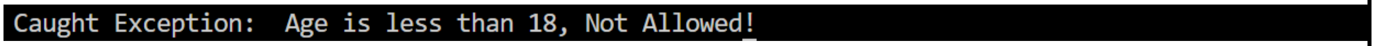
26. Write a java program to generate user defined exception using “throw” and “throws” keyword.

Also Write a java that differentiates checked and unchecked exceptions. (Mention at least two checked and two unchecked exceptions in program).

PROGRAM :

```
class CustomException extends Exception {
public CustomException(String message) {
super(message);}}
public class customExceptionDemo {
public static void checkAge(int age) throws CustomException {
if (age < 18) {
throw new CustomException(" Age is less than 18, Not Allowed!");
} else {System.out.println("Age is valid.");}}

public static void main(String[] args) {
try {
checkAge(15);
} catch (CustomException e) {
System.out.println("Caught Exception: " + e.getMessage());}}}
```

OUTPUT :



```
Caught Exception:  Age is less than 18, Not Allowed!
```

CONCLUSION :

The provided code demonstrates how to create and handle a custom exception in Java using the AgeException class. The checkAge method verifies whether a user is eligible for access based on their age. If the age is less than 18, it throws the AgeException with a custom error message. This exception is caught in the main method, where an appropriate message is printed. By utilizing the `throw` and `throws` keywords, the program effectively manages error conditions (age restrictions).

Part - 6

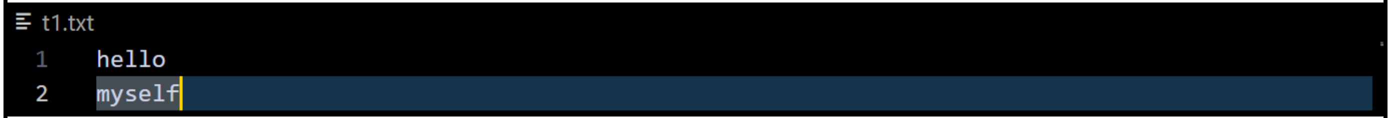
No.	Aim of the Practical
27.	Write a program that will count the number of lines in each file that is specified on the command line. Assume that the files are text files. Note that multiple files can be specified, as in "java Line Counts file1.txt file2.txt file3.txt". Write each file name, along with the number of lines in that file, to standard output. If an error occurs while trying to read from one of the files, you should print an error message for that file, but you should still process all the remaining files.

PROGRAM :

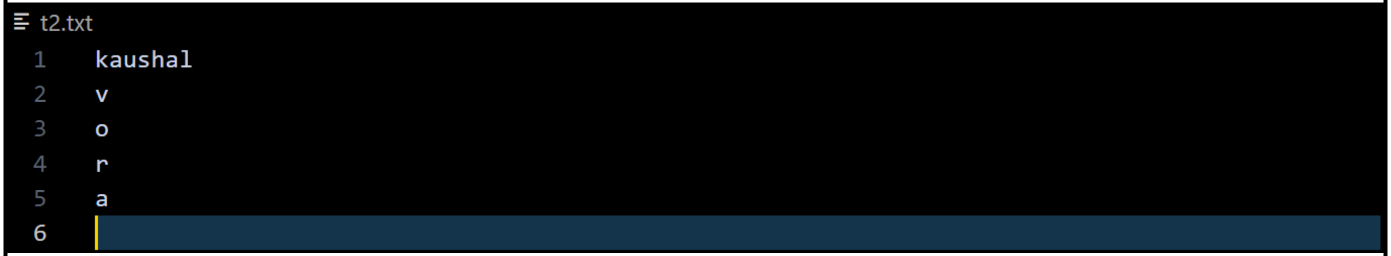
```
import java.io.FileReader;
import java.io.IOException;

public class set6_1 {

public static void main(String[] args) {
if (args.length == 0) {
System.out.println("Please specify at least one file.");
return;
}

for (String fileName : args) {
countLinesInFile(fileName);
}
}

private static void countLinesInFile(String fileName) {
int lineCount = 0;

try (FileReader fileReader = new FileReader(fileName))
{
int character;
while ((character = fileReader.read()) != -1)
{if (character == '\n') {lineCount++;}}

if (lineCount > 0 || character == -1) {
lineCount++;
}
System.out.println(fileName  +  ":  "  +  lineCount  +  "
lines");
} catch (IOException e) {
```

```
System.out.println("Error reading file " + fileName + ":
" + e.getMessage());
}
}
}
```

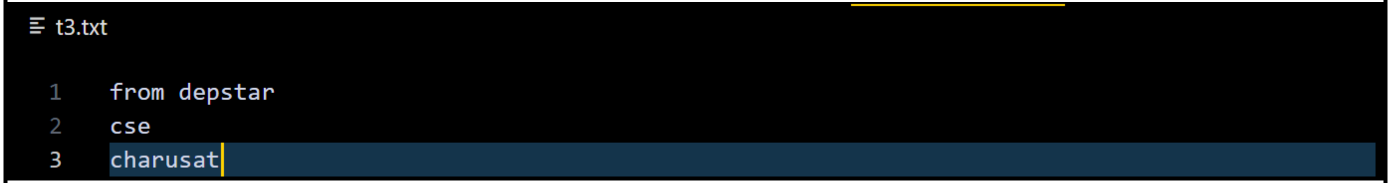
OUTPUT :



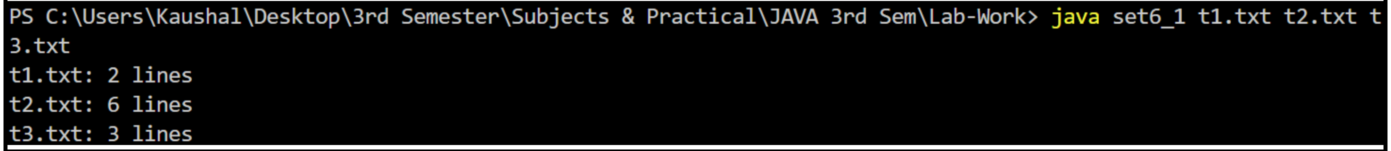
```
hello
myself
```



```
kaushal
v
o
r
a

```



```
from depstar
cse
charusat
```



```
PS C:\Users\Kaushal\Desktop\3rd Semester\Subjects & Practical\JAVA 3rd Sem\Lab-Work> java set6_1 t1.txt t2.txt t
3.txt
t1.txt: 2 lines
t2.txt: 6 lines
t3.txt: 3 lines
```

CONCLUSION:

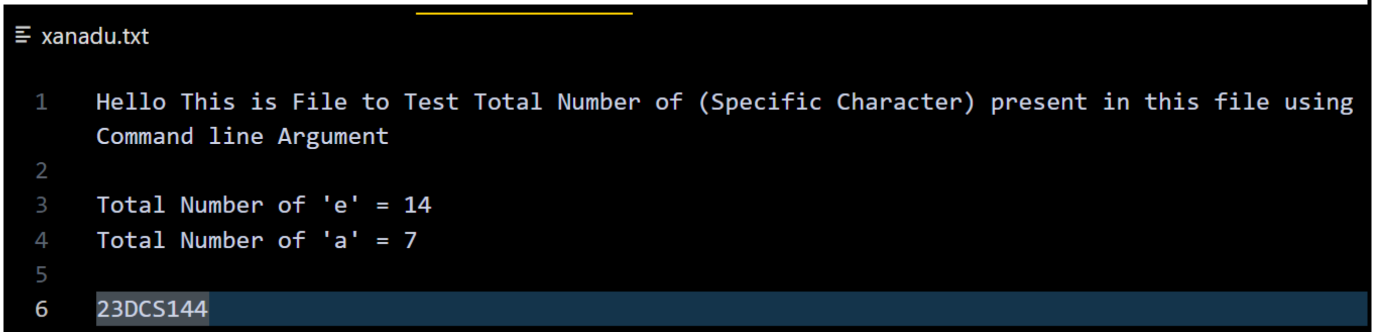
This program counts the number of lines in a file using Java. It reads each file specified in the command-line arguments or defaults to hello.txt if no arguments are provided. The program uses BufferedReader to read each line and increments a counter for each line read. It handles file reading errors gracefully using a try-with-resources block. The program prints the number of lines for each file processed. This showcases efficient file handling and error management in Java.

28. Write an example that counts the number of times a particular character, such as e, appears in a file. The character can be specified at the command line. You can use xanadu.txt as the input file.

PROGRAM CODE :

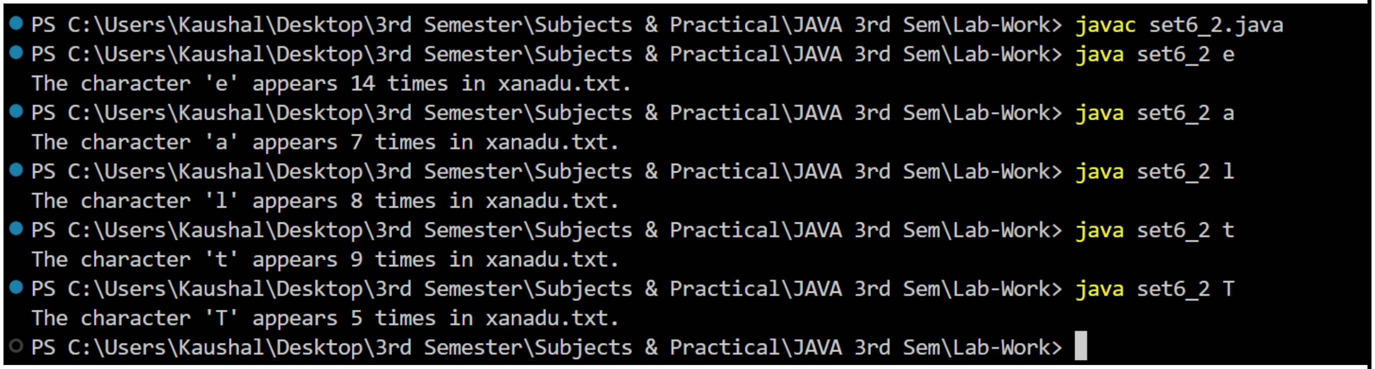
```
import java.io.FileReader;
import java.io.IOException;

public class set6_2 {

public static void main(String[] args) {
if (args.length != 1) {
System.out.println("Please specify a character to
count.");
return;
}

char targetChar = args[0].charAt(0);
String fileName = "xanadu.txt";

int count = countCharacterInFile(fileName, targetChar);
System.out.println("The character '" + targetChar + "'
appears " + count + " times in " + fileName + ".");
}

static int countCharacterInFile(String fileName, char
targetChar) {
int count = 0;

try (FileReader fileReader = new FileReader(fileName))
{
int character;
while ((character = fileReader.read()) != -1)
{if (character == targetChar) {count++;}}
}
catch (IOException e) {System.out.println("Error
reading file " + fileName + ": " + e.getMessage());}

return count;
}
}
```

OUTPUT:



xanadu.txt

```
Hello This is File to Test Total Number of (Specific Character) present in this file using
Command line Argument

Total Number of 'e' = 14
Total Number of 'a' = 7

23DCS144
```



```
PS C:\Users\Kaushal\Desktop\3rd Semester\Subjects & Practical\JAVA 3rd Sem\Lab-Work> javac set6_2.java
PS C:\Users\Kaushal\Desktop\3rd Semester\Subjects & Practical\JAVA 3rd Sem\Lab-Work> java set6_2 e
The character 'e' appears 14 times in xanadu.txt.
PS C:\Users\Kaushal\Desktop\3rd Semester\Subjects & Practical\JAVA 3rd Sem\Lab-Work> java set6_2 a
The character 'a' appears 7 times in xanadu.txt.
PS C:\Users\Kaushal\Desktop\3rd Semester\Subjects & Practical\JAVA 3rd Sem\Lab-Work> java set6_2 l
The character 'l' appears 8 times in xanadu.txt.
PS C:\Users\Kaushal\Desktop\3rd Semester\Subjects & Practical\JAVA 3rd Sem\Lab-Work> java set6_2 t
The character 't' appears 9 times in xanadu.txt.
PS C:\Users\Kaushal\Desktop\3rd Semester\Subjects & Practical\JAVA 3rd Sem\Lab-Work> java set6_2 T
The character 'T' appears 5 times in xanadu.txt.
PS C:\Users\Kaushal\Desktop\3rd Semester\Subjects & Practical\JAVA 3rd Sem\Lab-Work>
```

CONCLUSION:

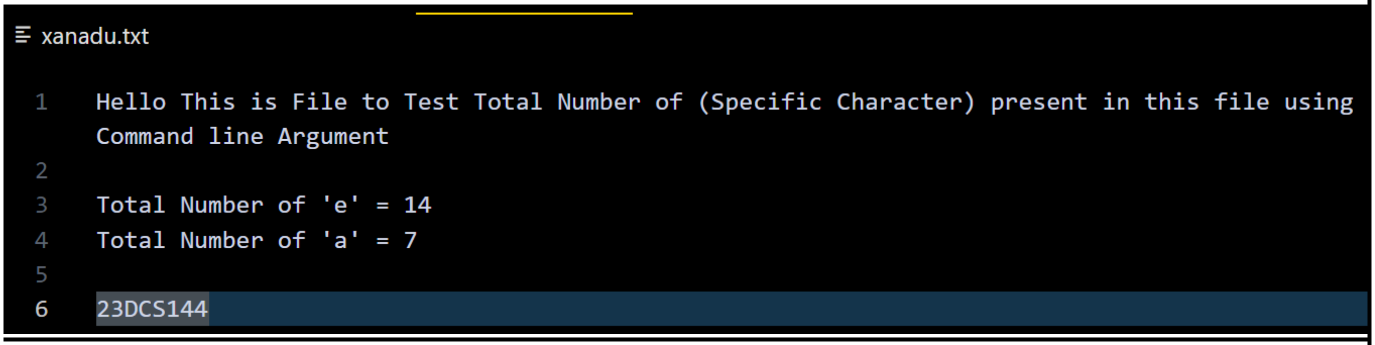
This program counts the occurrences of a specific character in a file using Java. It reads the file character by character with BufferedReader and compares each character to the target character. If they match, it increments a counter. The program handles file reading errors using a try-with-resources block to ensure the reader is closed properly. It also provides usage instructions if the required command-line arguments are not provided. This showcases efficient character processing and error management in Java.

29. Write a Java Program to Search for a given word in a File. Also show use of Wrapper Class with an example

PROGRAM :

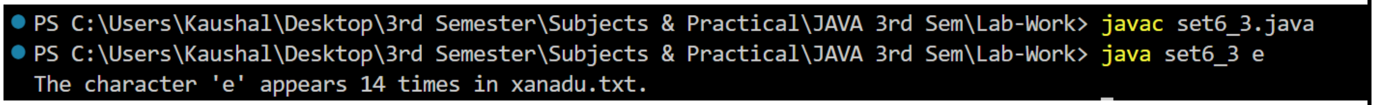
```
import java.io.FileReader;
import java.io.IOException;

public class set6_3 {

public static void main(String[] args) {
if (args.length != 1) {
System.out.println("Please specify a character to count.");
return;
}

char targetChar = args[0].charAt(0);
String fileName = "xanadu.txt";

int count = countCharacterInFile(fileName, targetChar);

Integer countWrapper = Integer.valueOf(count);
System.out.println("The character '" + targetChar + "' appears " + countWrapper + " times in " + fileName + ".");
}

private static int countCharacterInFile(String fileName, char targetChar) {
int count = 0;

try (FileReader fileReader = new FileReader(fileName)) {
int character;
while ((character = fileReader.read()) != -1) {
if (character == targetChar) {
count++;}}
} catch (IOException e) {
System.out.println("Error reading file " + fileName + ": " + e.getMessage());
}
return count;}}
```

OUTPUT :



xanadu.txt

```
Hello This is File to Test Total Number of (Specific Character) present in this file using Command line Argument

Total Number of 'e' = 14
Total Number of 'a' = 7

23DCS144
```



```
PS C:\Users\Kaushal\Desktop\3rd Semester\Subjects & Practical\JAVA 3rd Sem\Lab-Work> javac set6_3.java
PS C:\Users\Kaushal\Desktop\3rd Semester\Subjects & Practical\JAVA 3rd Sem\Lab-Work> java set6_3 e
The character 'e' appears 14 times in xanadu.txt.
```

CONCLUSION :

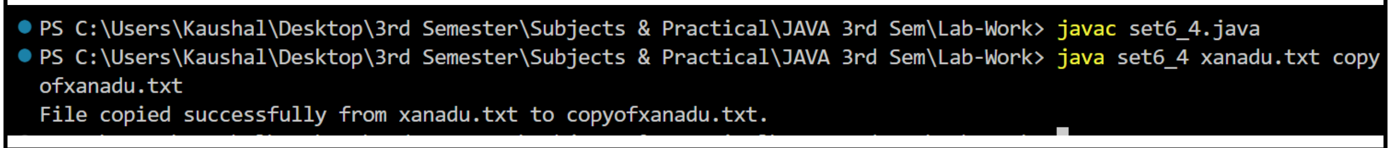
This program demonstrates how to count the occurrences of a specific word in a file using Java. It reads the file line by line with BufferedReader and splits each line into words. It then compares each word to the target word and increments a counter if they match. The program handles file reading errors gracefully using a try-with-resources block. It also provides usage instructions if the required command-line arguments are not provided. This showcases efficient text processing and error management in Java.

30. Write a program to copy data from one file to another file. If the destination file does not exist, it is created automatically.

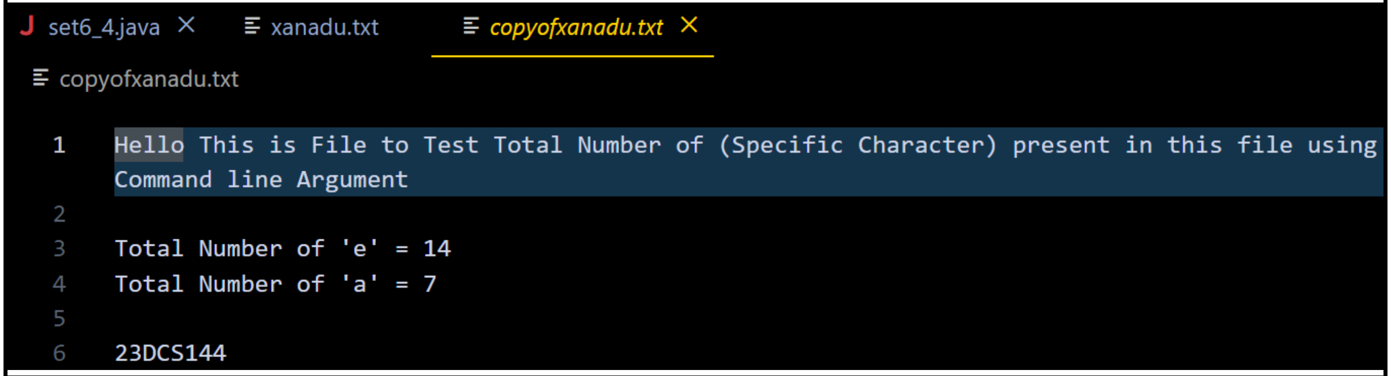
PROGRAM :

```
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class set6_4 {

public static void main(String[] args) {
if (args.length != 2) {
System.out.println("Usage: java Set6_4 <sourceFile> <destinationFile>");
return;
}

String sourceFileName = args[0];
String destinationFileName = args[1];

copyFile(sourceFileName, destinationFileName);
}

private static void copyFile(String sourceFileName, String destinationFileName) {
try (FileReader fileReader = new FileReader(sourceFileName);
FileWriter fileWriter = new FileWriter(destinationFileName)) {

int character;
while ((character = fileReader.read()) != -1) {
fileWriter.write(character);
}

System.out.println("File copied successfully from " + sourceFileName + " to " + destinationFileName + ".");
} catch (IOException e) {
System.out.println("Error during file copy: " + e.getMessage());
}
}
}
```

OUTPUT :



```
PS C:\Users\Kaushal\Desktop\3rd Semester\Subjects & Practical\JAVA 3rd Sem\Lab-Work> javac set6_4.java
PS C:\Users\Kaushal\Desktop\3rd Semester\Subjects & Practical\JAVA 3rd Sem\Lab-Work> java set6_4 xanadu.txt copy
ofxanadu.txt
File copied successfully from xanadu.txt to copyofxanadu.txt.
```



```
Hello This is File to Test Total Number of (Specific Character) present in this file using
Command line Argument

Total Number of 'e' = 14
Total Number of 'a' = 7

23DCS144
```

CONCLUSION :

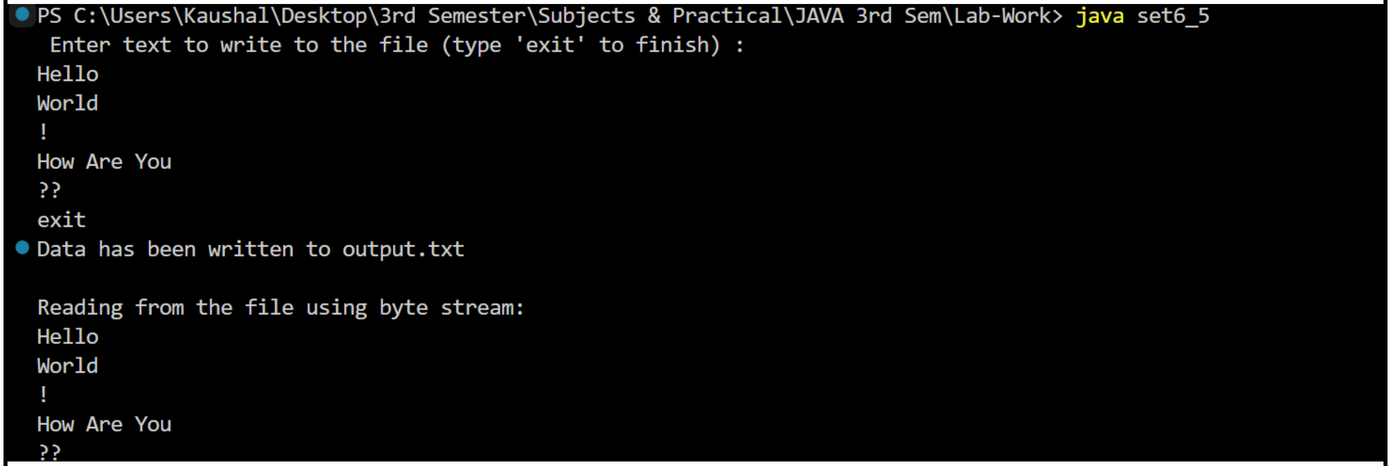
This program demonstrates how to copy data from one file to another using byte streams in Java. It reads from a source file and writes to a destination file, creating the destination file if it does not exist. The program uses FileInputStream to read bytes and FileOutputStream to write bytes. It handles errors using a try-with-resources block to ensure streams are closed properly. The program also provides usage instructions if the required command-line arguments are not provided. This showcases efficient file handling and error management in Java.

31. Write a program to show use of character and byte stream. Also show use of BufferedReader/BufferedWriter to read console input and write them into a file.

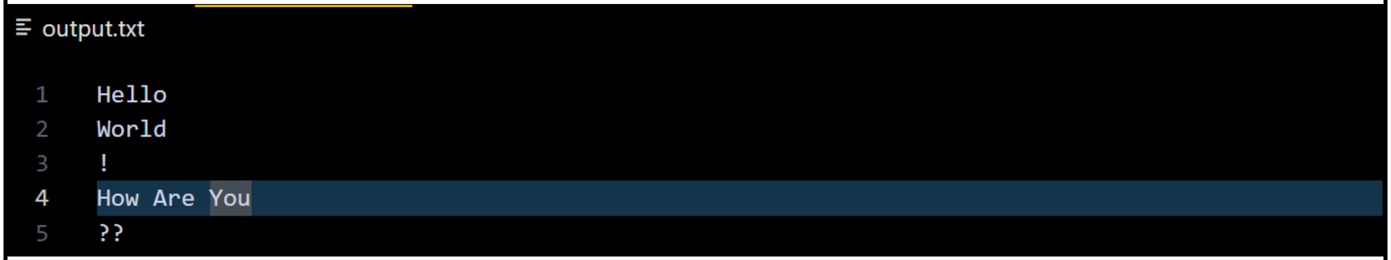
PROGRAM :

```
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class set6_4 {

public static void main(String[] args) {
if (args.length != 2) {
System.out.println("Usage: java Set6_4 <sourceFile> <destinationFile>");
return;}
String sourceFileName = args[0];
String destinationFileName = args[1];

copyFile(sourceFileName, destinationFileName);
}
static void copyFile(String sourceFileName, String destinationFileName) {
try (FileReader fileReader = new FileReader(sourceFileName);
FileWriter fileWriter = new FileWriter(destinationFileName)) {

int character;
while ((character = fileReader.read()) != -1) {
fileWriter.write(character);
}
System.out.println("File copied successfully from " + sourceFileName + " to " + destinationFileName + ".");
} catch (IOException e) {
System.out.println("Error during file copy: " + e.getMessage());}}}
```

OUTPUT :



```
PS C:\Users\Kaushal\Desktop\3rd Semester\Subjects & Practical\JAVA 3rd Sem\Lab-Work> java set6_5
 Enter text to write to the file (type 'exit' to finish) :
Hello
World
!
How Are You
??
exit
Data has been written to output.txt

Reading from the file using byte stream:
Hello
World
!
How Are You
??
```



output.txt

```
Hello
World
!
How Are You
??
```

CONCLUSION :

This program demonstrates the use of character and byte streams in Java. It reads from input.txt and writes to output_char.txt using character streams, and to output_byte.txt using byte streams. Additionally, it uses BufferedReader to read console input and BufferedWriter to write the input to console_output.txt. The program continues to read from the console until the user types "exit". This showcases efficient file handling and console interaction in Java.

Part – 7

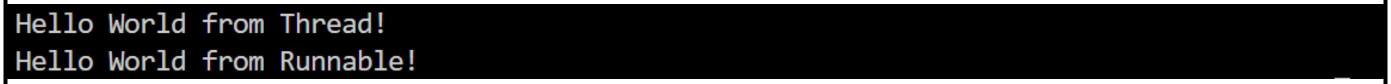
No.	Aim of the Practical

32. Write a program to create thread which display “Hello World” message. A. by extending Thread class B. by using Runnable interface.

PROGRAM :

```
class HelloWorldThread extends Thread {
public void run() {System.out.println("Hello World from
Thread!");}
}

class HelloWorldRunnable implements Runnable {
public void run() {System.out.println("Hello World from
Runnable!");}
}

public class set7_1 {
public static void main(String[] args) {
// A. Using the extended Thread class
HelloWorldThread thread1 = new HelloWorldThread();
thread1.start();

// B. Using the Runnable interface
Thread thread2 = new Thread(new
HelloWorldRunnable());
thread2.start();
}
}
```

OUTPUT :



```
Hello World from Thread!
Hello World from Runnable!
```

CONCLUSION:

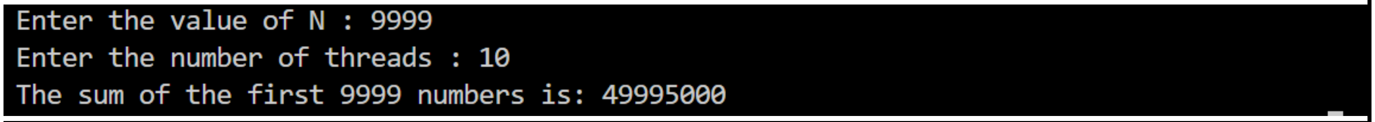
This code demonstrates two methods of creating threads in Java: by extending the Thread class and by implementing the Runnable interface. The run method prints a "Hello World" message.

33. Write a program which takes N and number of threads as an argument. Program should distribute the task of summation of N numbers amongst number of threads and final result to be displayed on the console.

PROGRAM CODE :

```
import java.util.Scanner;

class SumThread extends Thread {
int start;
int end;
int sum;

public SumThread(int start, int end) {
this.start = start;
this.end = end;
}

@Override
public void run() {
sum = 0;
for (int i = start; i <= end; i++) {sum += i;}
}

public int getSum() {return sum;}
}

public class Thread_sum {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
System.out.print("Enter the value of N : ");
int N = scanner.nextInt();

System.out.print("Enter the number of threads : ");
int numberOfThreads = scanner.nextInt();

int range = N / numberOfThreads;
SumThread[]          threads          =          new
SumThread[numberOfThreads];

for (int i = 0; i < numberOfThreads; i++) {
int start = i * range + 1;
int end = (i == numberOfThreads - 1) ? N : (i + 1) * range;
threads[i] = new SumThread(start, end);
threads[i].start();
}
int totalSum = 0;
try {
for (SumThread thread : threads) {
thread.join();
totalSum += thread.getSum();
}
} catch (InterruptedException e) {
System.out.println("Thread interrupted: " +
e.getMessage());
}

System.out.println("The sum of the first " + N + "
numbers is: " + totalSum);
scanner.close();
}
}
```

OUTPUT:



```
Enter the value of N : 9999
Enter the number of threads : 10
The sum of the first 9999 numbers is: 49995000
```

CONCLUSION:

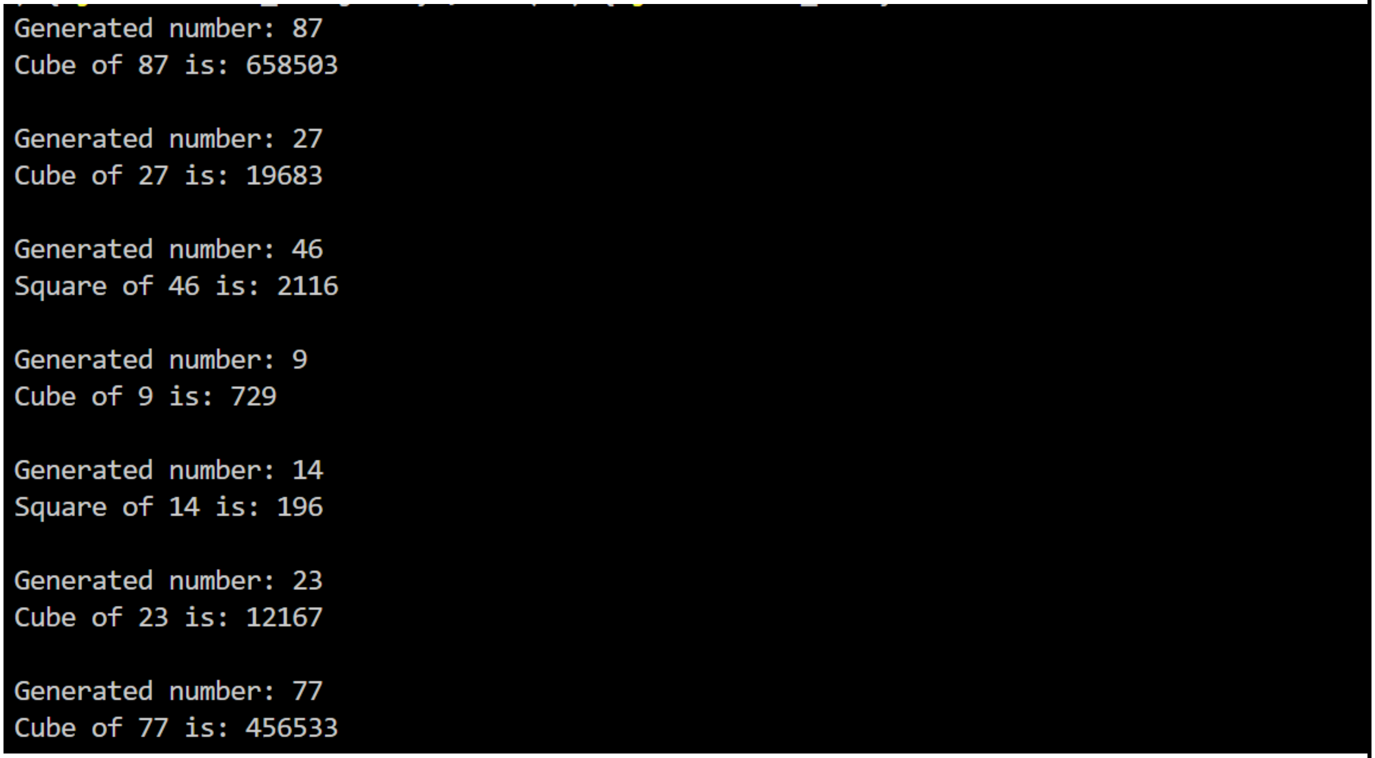
Each thread computes a partial sum, which is then combined to get the total sum. The join() method ensures that the main thread waits for all threads to complete before calculating the final result.

34. Write a java program that implements a multi-thread application that has three threads. First thread generates random integer every 1 second and if the value is even, second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of cube of the number.

PROGRAM :

```
import java.util.Random;
class ThreadGenerator extends Thread {
Random random = new Random();
EvenThread evenThread;
OddThread oddThread;

public ThreadGenerator(EvenThread evenThread, OddThread oddThread) {
this.evenThread = evenThread;
this.oddThread = oddThread;
}

@Override
public void run() {
for(int i = 0 ; i<10 ; i++)
{
int number = random.nextInt(100);
System.out.println("Generated number: " + number);

if (number % 2 == 0) {
evenThread.computeSquare(number);
} else {
oddThread.computeCube(number);
}

try {
Thread.sleep(1000);
} catch (InterruptedException e) {
e.printStackTrace();
}
}
}
}

class EvenThread extends Thread {
@Override
public void run() {}
public void computeSquare(int number) {
int square = number * number;
System.out.println("Square of " + number + " is: " + square+"\n");
}
}

class OddThread extends Thread {
@Override
public void run() {}
public void computeCube(int number) {
int cube = number * number * number;
System.out.println("Cube of " + number + " is: " + cube+"\n");
}
}

public class Thread_3rd
{
public static void main(String[] args)
{
EvenThread evenThread = new EvenThread();
OddThread oddThread = new OddThread();
ThreadGenerator threadGenerator = new ThreadGenerator(evenThread, oddThread);

evenThread.start();
oddThread.start();
threadGenerator.start();
}
}
```

OUTPUT :



```
Generated number: 87
Cube of 87 is: 658503

Generated number: 27
Cube of 27 is: 19683

Generated number: 46
Square of 46 is: 2116

Generated number: 9
Cube of 9 is: 729

Generated number: 14
Square of 14 is: 196

Generated number: 23
Cube of 23 is: 12167

Generated number: 77
Cube of 77 is: 456533
```

CONCLUSION :

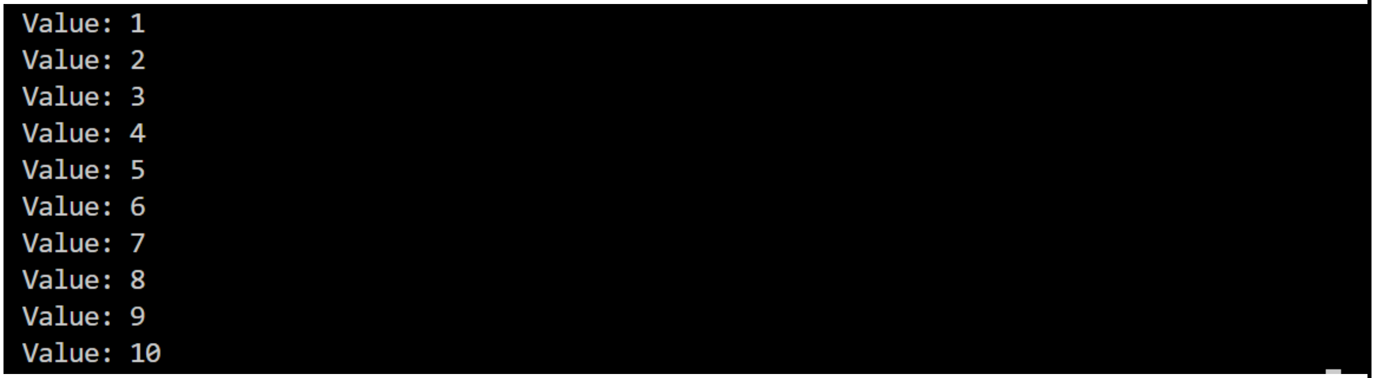
This code alternates between calculating the square and cube of numbers from 1 to 10 using two separate classes, Square and Cube. It runs the appropriate calculation based on whether the number is even or odd, with a one-second delay between each calculation. The program demonstrates basic thread usage and control flow, although it directly calls the run() method instead of starting a new thread.

35. Write a program to increment the value of one variable by one and display it after one second using thread using sleep() method.

PROGRAM :

```
class IncrementValueByThread implements Runnable {
int value = 0;
public void run() {
try {
for (int i = 0; i < 10; i++) {value++;
System.out.println("Value: " + value);
Thread.sleep(1000);}
} catch (InterruptedException e) {
System.out.println("Thread was interrupted.");}}}

public class Thread_35{
public static void main(String[] args) {
IncrementValueByThread incrementer = new IncrementValueByThread();
Thread thread = new Thread(incrementer);
thread.start();}}
```

OUTPUT :



```
Value: 1
Value: 2
Value: 3
Value: 4
Value: 5
Value: 6
Value: 7
Value: 8
Value: 9
Value: 10
```

CONCLUSION :

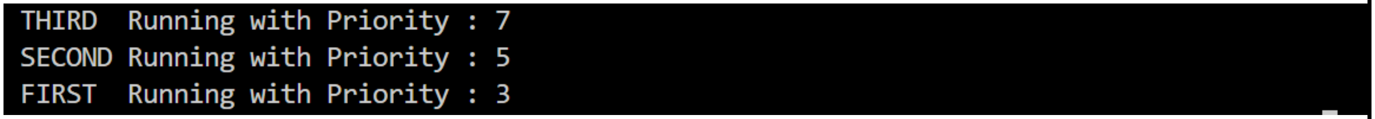
In conclusion, this program demonstrates how to manage timed delays in Java using `Thread.sleep()`, while also showing the importance of exception handling when using methods that can potentially interrupt normal execution.

36. Write a program to create three threads ‘FIRST’, ‘SECOND’, ‘THIRD’. Set the priority of the ‘FIRST’ thread to 3, the ‘SECOND’ thread to 5(default) and the ‘THIRD’ thread to 7.

PROGRAM :

```
class set7_5
{
public static void main(String[] args){
Thread firstThread  = new Thread(new MyThread(), "FIRST ");
Thread secondThread = new Thread(new MyThread(), "SECOND");
Thread thirdThread  = new Thread(new MyThread(), "THIRD ");

firstThread.setPriority(3);
secondThread.setPriority(Thread.NORM_PRIORITY);
thirdThread.setPriority(7);

firstThread.start();
secondThread.start();
thirdThread.start();}}

class MyThread implements Runnable{
public void run(){
System.out.println(Thread.currentThread().getName() + " Running with Priority : " + Thread.currentThread().getPriority());}}
```

OUTPUT :



```
THIRD  Running with Priority : 7
SECOND Running with Priority : 5
FIRST  Running with Priority : 3
```

CONCLUSION :

The program demonstrates creating multiple threads with different priorities using setPriority(). While thread priority influences scheduling, the actual execution order is not guaranteed and depends on the Java thread scheduler.

37. Write a program to create three threads ‘FIRST’, ‘SECOND’, ‘THIRD’. Set the priority of the ‘FIRST’ thread to 3, the ‘SECOND’ thread to 5(default) and the ‘THIRD’ thread to 7.

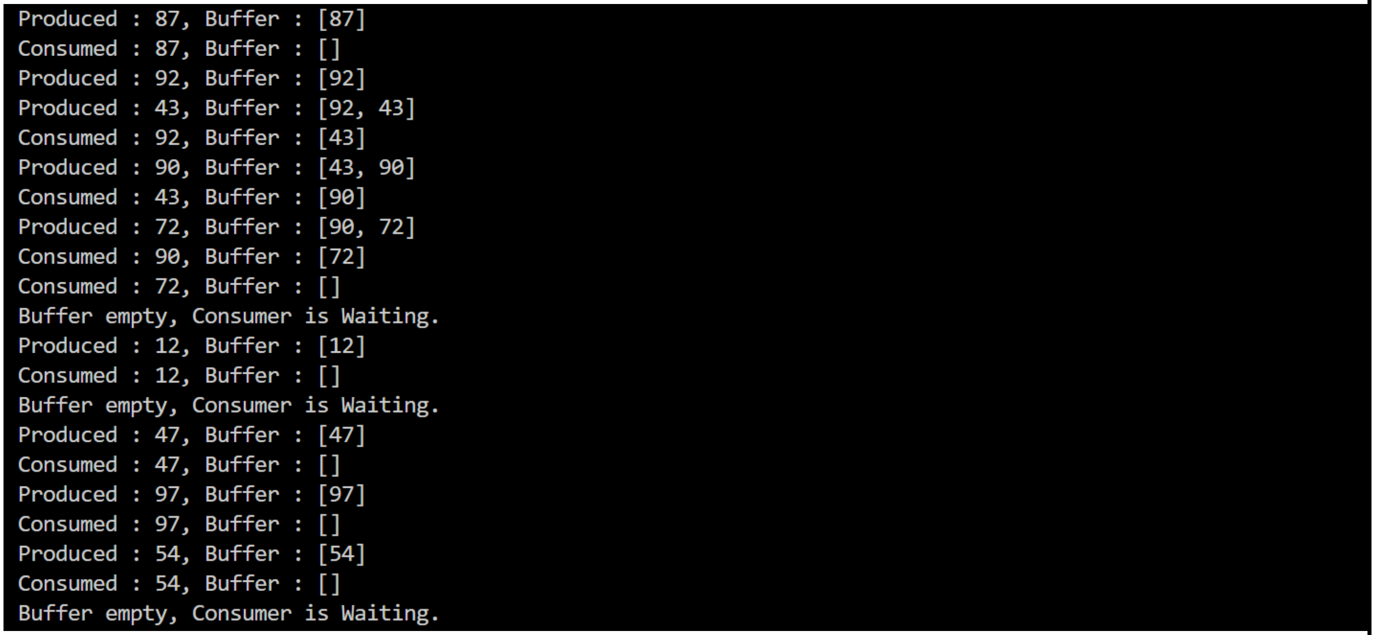
PROGRAM :

```
import java.util.LinkedList;

class Buffer {
LinkedList<Integer> buffer = new LinkedList<>();
int bufferSize;

public Buffer(int size) {bufferSize = size;}

public synchronized void produce(int item) throws InterruptedException {
while (buffer.size() >= bufferSize) {
System.out.println("Buffer full, producer is waiting.");
wait();
}
buffer.add(item);
System.out.println(" Produced : " + item + ", Buffer : " + buffer);
notify();
}

public synchronized int consume() throws InterruptedException {
while (buffer.isEmpty()) {
System.out.println(" Buffer empty, Consumer is Waiting.");
wait();
}
int item = buffer.removeFirst();
System.out.println(" Consumed : " + item + ", Buffer : " + buffer);
notify();
return item;
}
}

class Producer extends Thread {
```

```
Buffer buffer;

public Producer(Buffer buffer) {this.buffer = buffer;}

@Override
public void run() {
try {
while (true) {
int item = (int) (Math.random() * 100);
buffer.produce(item);
Thread.sleep((long) (Math.random() * 1000));
}
}
catch (InterruptedException e) {Thread.currentThread().interrupt();}
}
}

class Consumer extends Thread {
Buffer buffer;

public Consumer(Buffer buffer) {this.buffer = buffer;}

@Override
public void run() {
try {
while (true) {
buffer.consume();
Thread.sleep((long) (Math.random() * 1000));
}
}
catch (InterruptedException e) {Thread.currentThread().interrupt();}
}
}

public class set7_6 {
```

```
public static void main(String[] args) {
Buffer buffer = new Buffer(5);
Producer producer = new Producer(buffer);
Consumer consumer = new Consumer(buffer);

producer.start();
consumer.start();
}
}
```

OUTPUT :



```
Produced : 87, Buffer : [87]
Consumed : 87, Buffer : []
Produced : 92, Buffer : [92]
Produced : 43, Buffer : [92, 43]
Consumed : 92, Buffer : [43]
Produced : 90, Buffer : [43, 90]
Consumed : 43, Buffer : [90]
Produced : 72, Buffer : [90, 72]
Consumed : 90, Buffer : [72]
Consumed : 72, Buffer : []
Buffer empty, Consumer is Waiting.
Produced : 12, Buffer : [12]
Consumed : 12, Buffer : []
Buffer empty, Consumer is Waiting.
Produced : 47, Buffer : [47]
Consumed : 47, Buffer : []
Produced : 97, Buffer : [97]
Consumed : 97, Buffer : []
Produced : 54, Buffer : [54]
Consumed : 54, Buffer : []
Buffer empty, Consumer is Waiting.
```

CONCLUSION :

In conclusion, this program effectively demonstrates inter-thread communication and synchronization, ensuring smooth cooperation between the producer and consumer threads while preventing over-production and over-consumption. It solves the producer-consumer problem by using thread-safe techniques to manage shared resources.

Part – 8

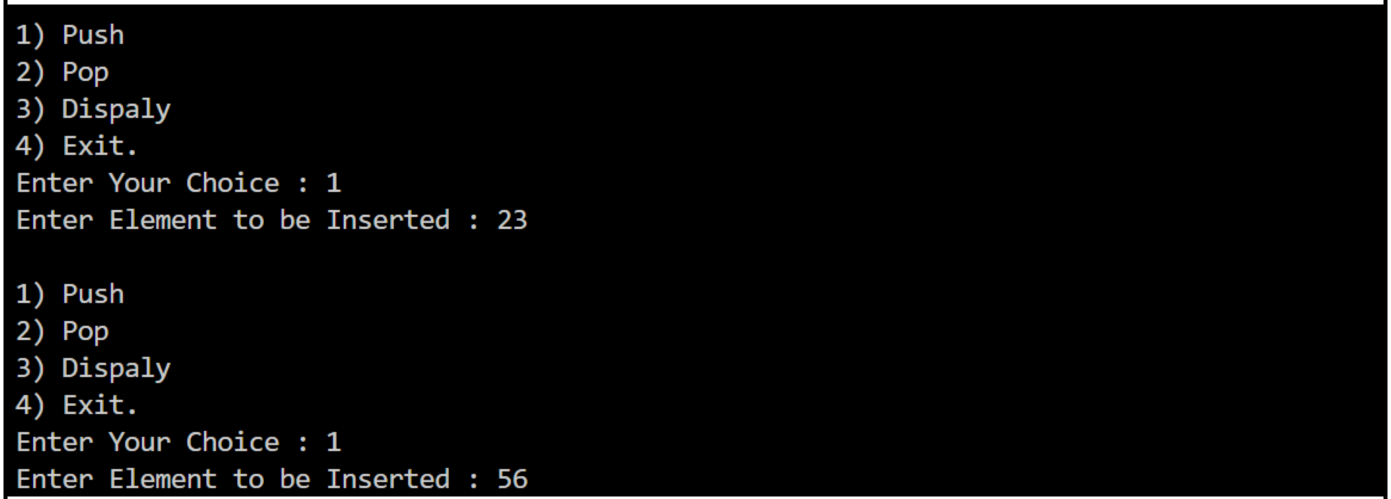
No.	Aim of the Practical
38.	Design a Custom Stack using ArrayList class, which implements following functionalities of stack. My Stack-list ArrayList<Object>: A list to store elements. +isEmpty: boolean: Returns true if this stack is empty. +getSize(): int: Returns number of elements in this stack. +peek(): Object: Returns top element in this stack without removing it. +pop(): Object: Returns and Removes the top elements in this stack. +push(o: object): Adds new element to the top of thisstack.

PROGRAM :

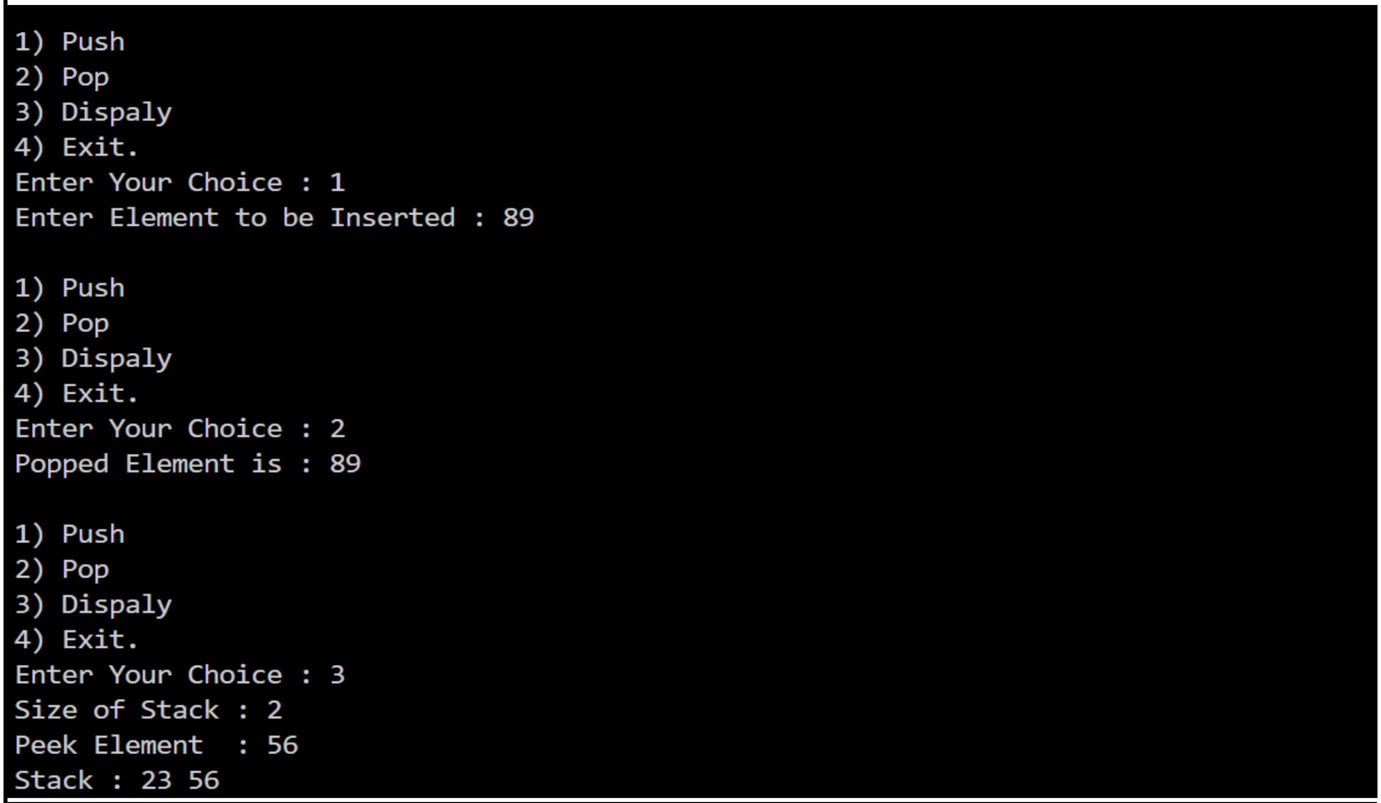
```
import java.util.Scanner;
import java.util.ArrayList;
import java.util.Stack;
class StAck{
static ArrayList<Integer> stack = new
ArrayList<Integer>(10);

public boolean isemptY(){ return stack.isEmpty();}

public static void push(Integer elem){ stack.add(elem); }

public static Integer pop(){
if (stack.isEmpty()){System.out.println(" Stack
Empty!!"); return -1;}
return stack.removeLast();}

public static int getsize(){ return stack.size(); }

public static int peek(){ return stack.get(getsize()-1);}

public static void display(){
if(!stack.isEmpty())
{
System.out.println(" Size of Stack : "+getsize());
System.out.println(" Peek Element  : "+peek());
System.out.print(" Stack : ");
for (int i = 0; i < stack.size(); i++)
{ System.out.print(stack.get(i)+ " "); }
System.out.println();}
else{ System.out.println(" Stack Empty!! "); }}}
public class set8_1 {
public static void main(String[] args){
```

```
Scanner sc = new Scanner(System.in);
OUTER:
while (true)
{
System.out.print("\n 1) Push  \n"+
" 2) Pop     \n"+
" 3) Dispaly \n"+
" 4) Exit.   \n"+
" Enter Your Choice : ");

int choice = sc.nextInt();

switch(choice){
case 1:
System.out.print(" Enter Element to be Inserted : ");
int elem = sc.nextInt();
StAck.push(elem);
break;

case 2:
int poP = StAck.pop();
if(poP != -1){System.out.println(" Popped Element is :
"+poP);}
break;

case 3:
StAck.display();
break;

case 4:
System.out.println(" Exit!!");
break OUTER;
default:
System.out.println(" Invalid Choice!!");
break;}}}}
```

OUTPUT :



```
1) Push
2) Pop
3) Dispaly
4) Exit.
Enter Your Choice : 1
Enter Element to be Inserted : 23

1) Push
2) Pop
3) Dispaly
4) Exit.
Enter Your Choice : 1
Enter Element to be Inserted : 56
```



```
1) Push
2) Pop
3) Dispaly
4) Exit.
Enter Your Choice : 1
Enter Element to be Inserted : 89

1) Push
2) Pop
3) Dispaly
4) Exit.
Enter Your Choice : 2
Popped Element is : 89

1) Push
2) Pop
3) Dispaly
4) Exit.
Enter Your Choice : 3
Size of Stack : 2
Peek Element  : 56
Stack : 23 56
```

CONCLUSION:

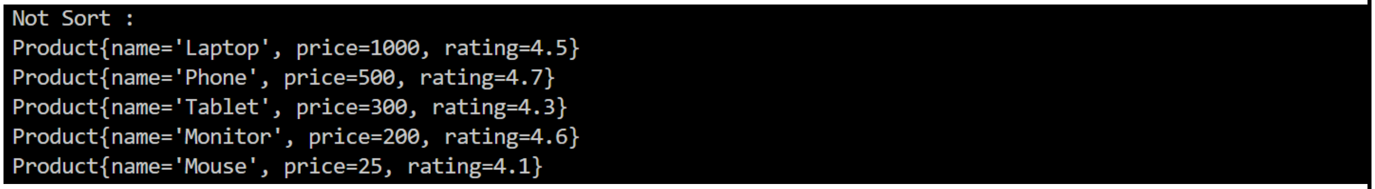
From this practical, I learned how to create a custom stack using the `ArrayList` class in Java. I implemented basic stack functionalities like checking if the stack is empty, getting the size, viewing the top element, and performing push and pop operations. This exercise helped me understand how to use an `ArrayList` to dynamically store elements and simulate a stack structure.

39. Imagine you are developing an e-commerce application. The platform needs to sort lists of products based on different criteria, such as price, rating, or name. Each product object implements the Comparable interface to define the natural ordering. To ensure flexibility and reusability, you need a generic method that can sort any array of Comparable objects. Create a generic method in Java that sorts an array of Comparable objects. This method should be versatile enough to sort arrays of different types of objects (such as products, customers, or orders) as long as they implement the Comparable interface.

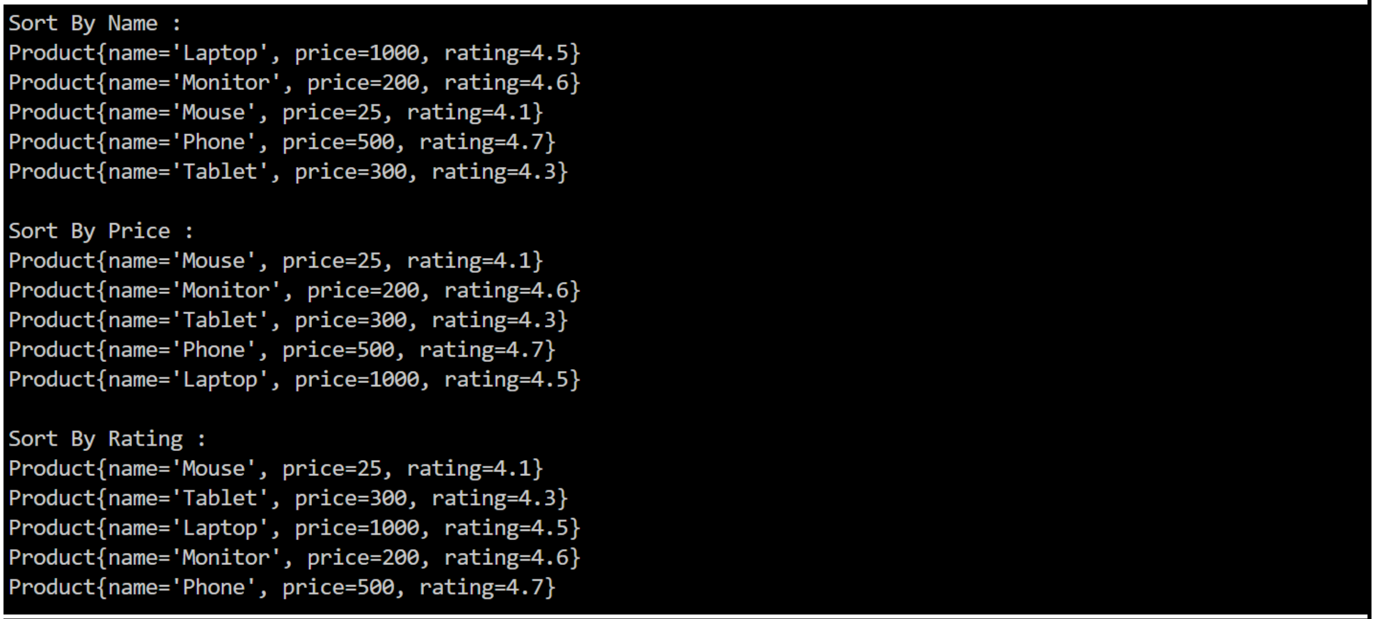
PROGRAM CODE :

```
import java.util.*;
class product
{
String name;
int price;
double rating;

product(String name , int price , double rating)
{
this.name = name;
this.price = price;
this.rating = rating;
}

public String toString() {
return " Product{name='" + name + "', price=" + price +
", rating=" + rating + "}";
}

public static product[] sort_name( product[] arr)
{
product[] sortedArr = Arrays.copyOf(arr, arr.length);
Arrays.sort(sortedArr,     Comparator.comparing(p     ->
p.name));
return sortedArr;
}

public static product[] sort_price( product[] arr)
{
product[] sortedArr = Arrays.copyOf(arr, arr.length);
Arrays.sort(sortedArr,   Comparator.comparingInt(p   ->
p.price));
return sortedArr;
}

public static product[] sort_rating( product[] arr)
{
product[] sortedArr = Arrays.copyOf(arr, arr.length);
Arrays.sort(sortedArr, Comparator.comparingDouble(p
-> p.rating));
return sortedArr;
}
}

public class set8_2 {
public static void main(String[] args)
{
product[] PRODUCTS = {
new product("Laptop", 1000, 4.5),
new product("Phone", 500, 4.7),
new product("Tablet", 300, 4.3),
new product("Monitor", 200, 4.6),
new product("Mouse", 25, 4.1)
};

product[] sortbyname =
product.sort_name(PRODUCTS);
product[] sortbyprice =
product.sort_price(PRODUCTS);
product[] sortbyrating =
product.sort_rating(PRODUCTS);

System.out.println(" Not Sort :");
for (product p : PRODUCTS) { System.out.println(p); }
System.out.println();

System.out.println(" Sort By Name :");
for (product q : sortbyname) { System.out.println(q); }
System.out.println();

System.out.println(" Sort By Price :");
for (product r : sortbyprice) { System.out.println(r); }
System.out.println();

System.out.println(" Sort By Rating :");
for (product s : sortbyrating) { System.out.println(s); }
System.out.println();
}
}
```

OUTPUT:



```
Not Sort :
Product{name='Laptop', price=1000, rating=4.5}
Product{name='Phone', price=500, rating=4.7}
Product{name='Tablet', price=300, rating=4.3}
Product{name='Monitor', price=200, rating=4.6}
Product{name='Mouse', price=25, rating=4.1}
```



```
Sort By Name :
Product{name='Laptop', price=1000, rating=4.5}
Product{name='Monitor', price=200, rating=4.6}
Product{name='Mouse', price=25, rating=4.1}
Product{name='Phone', price=500, rating=4.7}
Product{name='Tablet', price=300, rating=4.3}

Sort By Price :
Product{name='Mouse', price=25, rating=4.1}
Product{name='Monitor', price=200, rating=4.6}
Product{name='Tablet', price=300, rating=4.3}
Product{name='Phone', price=500, rating=4.7}
Product{name='Laptop', price=1000, rating=4.5}

Sort By Rating :
Product{name='Mouse', price=25, rating=4.1}
Product{name='Tablet', price=300, rating=4.3}
Product{name='Laptop', price=1000, rating=4.5}
Product{name='Monitor', price=200, rating=4.6}
Product{name='Phone', price=500, rating=4.7}
```

CONCLUSION:

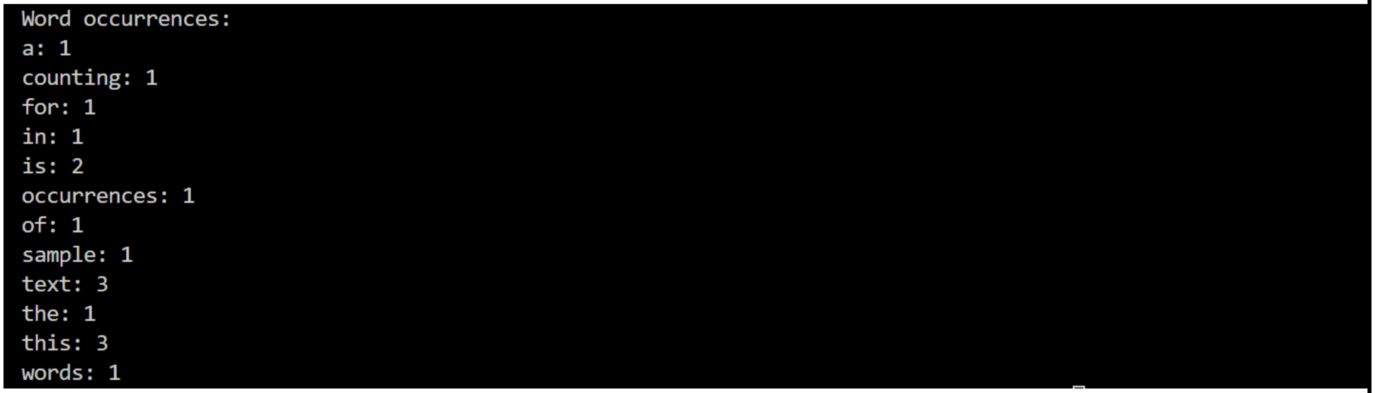
Through this practical, I gained insights into implementing a generic method in Java to sort arrays of objects that implement the `Comparable` interface. I learned how to ensure flexibility and reusability by enabling the method to sort various types of objects, such as products, customers, and orders, based on their natural ordering.

40. Write a program that counts the occurrences of words in a text and displays the words and their occurrences in alphabetical order of the words. Using Map and Set Classes.

PROGRAM :

```
import java.util.*;
public class set8_3 {
public static void main(String[] args) {
String text = "This is a sample text. This text is for counting the occurrences of words in this text.";
String normalizedText = text.toLowerCase().replaceAll("[^a-zA-Z0-9 ]", "");
String[] words = normalizedText.split("\\s+");
Map<String, Integer> wordCount = new HashMap<>();
for (String word : words) {
wordCount.put(word, wordCount.getOrDefault(word, 0) + 1);}
Set<String> sortedWords = new TreeSet<>(wordCount.keySet());
System.out.println("Word occurrences:");
for (String word : sortedWords) {
System.out.println(word + ": " + wordCount.get(word));}}}
```

OUTPUT :



```
Word occurrences:
a: 1
counting: 1
for: 1
in: 1
is: 2
occurrences: 1
of: 1
sample: 1
text: 3
the: 1
this: 3
words: 1
```

CONCLUSION :

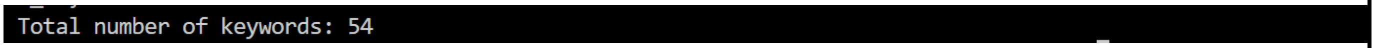
In this practical, I learned how to use Java's `Map` and `Set` classes to count and display the occurrences of words in a given text. I implemented a method that not only counts the occurrences but also sorts the words in alphabetical order. This exercise enhanced my understanding of utilizing collections to efficiently manage and manipulate data.

41. Write a code which counts the number of the keywords in a Java source file. Store all the keywords in a HashSet and use the contains () method to test if a word is in the keyword set.

PROGRAM :

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.HashSet;

public class set8_4 {
public static void main(String[] args) {
HashSet<String> keywords = new HashSet<>();
String[] javaKeywords = {
"abstract", "assert", "boolean", "break", "byte", "case", "catch", "char", "class",
"const", "continue", "default", "do", "double", "else", "enum", "extends", "final",
"finally", "float", "for", "goto", "if", "implements", "import", "instanceof", "int",
"interface", "long", "native", "new", "null", "package", "private", "protected",
"public", "return", "short", "static", "strictfp", "super", "switch", "synchronized",
"this", "throw", "throws", "transient", "try", "void", "volatile", "while"
};

for (String keyword : javaKeywords) {
keywords.add(keyword);
}



String filePath = "set8_1.java";

int keywordCount = countKeywordsInFile(filePath, keywords);

System.out.println("Total number of keywords: " + keywordCount);
}

public static int countKeywordsInFile(String filePath, HashSet<String> keywords) {
int count = 0;
try (BufferedReader br = new BufferedReader(new FileReader(filePath))) {
String line;
while ((line = br.readLine()) != null) {
String[] words = line.toLowerCase().split("\\W+"); // Split by non-word characters
for (String word : words) {
if (keywords.contains(word)) {
count++;
}
}
}
} catch (IOException e) {
e.printStackTrace();
}

return count;
}
}
```

OUTPUT :



```
Total number of keywords: 54
```

CONCLUSION :

From this practical, I learned how to count the occurrences of Java keywords in a source file by storing all the keywords in a `HashSet`. By using the `contains()` method, I was able to check whether a word is a keyword or not. This practical improved my skills in working with Java's collection framework, particularly using sets for fast lookups.