DEPSTAR

Devang Patel Institute of Advance Technology and Research

(A Constitute Institute of CHARUSAT)

Certificate

This is to certify that

Mr./~~Mrs.~~ Maharshi K Patel

of DEPSTAR CSE2 (A) *Class,*

ID. No. 23DCS080 *has satisfactorily completed*

his/ her term work in CSE 201 *for*

the ending in NOV 2024 /2025

Date : 17/10/24

[illegible] 17/10/24

Sign. of Faculty

[illegible]

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

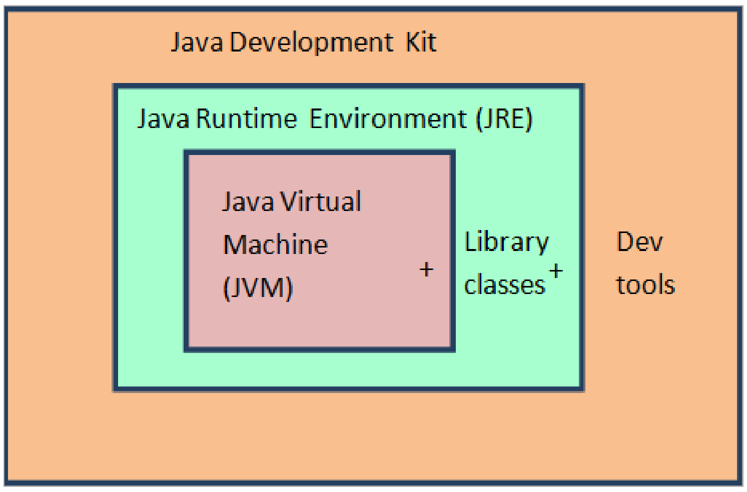
Subject Name: JAVA PROGRAMMING

Semester: 3

Subject Code: CSE201

Academic year:2024-25

PART-I Data Types, Variables, String, Control Statements, Operators, Arrays

No.	Aim of the Practical
	SET-1
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 Blue and Console Programming.  Java Development Kit Java Runtime Environment (JRE) Java Virtual Machine (JVM) + Library classes + Dev tools

1.JDK (Java Development Kit) is a Kit that provides the environment to develop and execute(run) the Java program. JDK is a kit (or package) that includes two things

• Development Tools (to provide an environment to develop your java programs)

• JRE (to execute your java program).

2. JRE (Java Runtime Environment) is an installation package that provides an environment to only run (not develop) the java program (or application) onto your machine. JRE is only used by those who only want to run Java programs that are end-users of your system.

3. JVM (Java Virtual Machine) is a very important part of both JDK and JRE because it is contained or inbuilt in both. Whatever Java program you run using JRE or JDK goes into JVM and JVM is responsible for executing the java program line by line, hence it is also known as an interpreter.
JVM (Java Virtual Machine) acts as a run-time engine to run Java applications. JVM is the one that actually calls the main method present in a java code. JVM is a part of JRE (Java Runtime Environment).

Java applications are called WORA (Write Once Run Anywhere). This means a programmer can develop Java code on one system and can expect it to run on any other Java-enabled system without any adjustment. This is all possible because of JVM.

When we compile a *.java* file, *.class* files (contains byte-code) with the same class names present in *.java* file are generated by the Java compiler.

Javadoc: Javadoc tool is a document generator tool in Java programming language for generating standard documentation in HTML format. It generates API documentation. It parses the declarations ad documentation in a set of source file describing classes, methods, constructors, and fields.
Before using Javadoc tool, you must include Javadoc comments /**………………. */ providing information about classes, methods, and constructors, etc. For creating a good and understandable document API for any java file you must write better comments for every class, method, constructor.

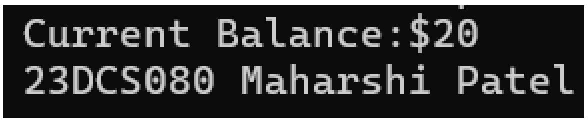
Command Line Argument: Java command-line argument is an argument i.e. passed at the time of running the Java program. In the command line, the arguments passed from the console can be received in the java program and they can be used as input. The users can pass the arguments during the execution bypassing the command-line arguments inside the main () method.

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.*;
class Bank{
public static void main(String args[]){
int a=20;
System.out.println("Current Balance:$"+a);
System.out.println("23DCS080 Maharshi Patel");
}
}
```

OUTPUT:



```
Current Balance:$20
23DCS080 Maharshi Patel
```

CONCLUSION:

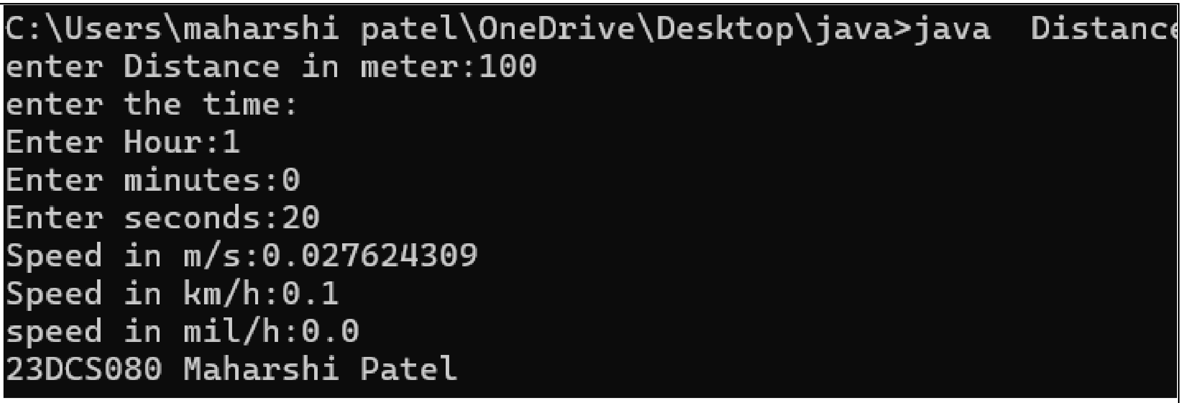
This program demonstrates the basic concept of storing and displaying data in Java, showcasing the use of variables and output statements.

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, kilometres per hour and miles per hour (hint:1 mile = 1609 meters).

PROGRAM CODE:

```
import java.util.*;
class Distance{
public static void main(String args[]){
Scanner sc=new Scanner(System.in);
System.out.print("enter Distance in meter:");
int d=sc.nextInt();
System.out.println("enter the time:");
System.out.print("Enter Hour:");
int hr=sc.nextInt();
System.out.print("Enter minutes:");
int min=sc.nextInt();
System.out.print("Enter seconds:");
int sec=sc.nextInt();
int time=(hr*3600)+(min*60)+sec;
float v=(float)d/time;
System.out.println("Speed in m/s:"+v);
time=hr+(min/60)+(sec/3600);
 v=(float)(d*0.001)/time;
System.out.println("Speed in km/h:"+v);
v=(float)(d/1609)/time;
System.out.println("speed in mil/h:"+v);
System.out.println("23DCS080 Maharshi Patel");

}
}
```

OUTPUT:



```
C:\Users\maharshi patel\OneDrive\Desktop\java>java  Distanc
enter Distance in meter:100
enter the time:
Enter Hour:1
Enter minutes:0
Enter seconds:20
Speed in m/s:0.027624309
Speed in km/h:0.1
speed in mil/h:0.0
23DCS080 Maharshi Patel
```

CONCLUSION:

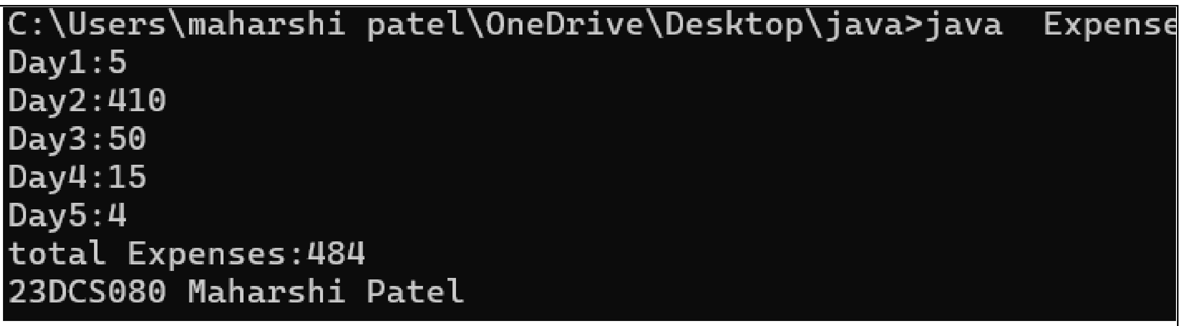
This program demonstrates the ability to take user input, perform calculations, and display results in different units, showcasing fundamental programming concepts and unit conversions.

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 CODE:

```
import java.util.*;
class Expense{
public static void main(String args[]){
int day[]=new int[30];
Scanner sc=new Scanner(System.in);
int sum=0;
for(int i=0;i<5;i++){
System.out.print("Day"+(i+1)+":");
int a=sc.nextInt();
sum+=a;
}
System.out.println("total Expenses:"+sum);
System.out.println("23DCS080 Maharshi Patel ");
}
}
```

OUTPUT:



```
C:\Users\maharshi patel\OneDrive\Desktop\java>java  Expense
Day1:5
Day2:410
Day3:50
Day4:15
Day5:4
total Expenses:484
23DCS080 Maharshi Patel
```

<u>CONCLUSION:</u>

This program demonstrates the use of arrays and loops to collect and calculate the sum of daily expenses, providing a simple yet effective way to track monthly expenditures.

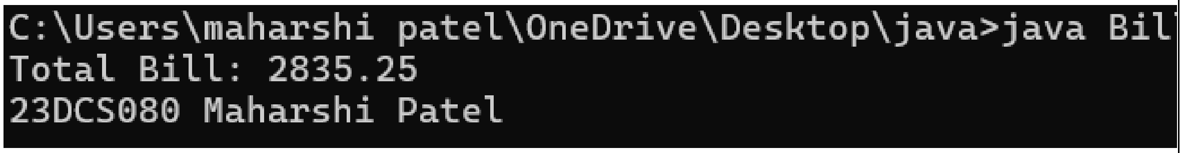
5. An electric appliance shop assigns code 1 to motor,2 to fan,3 to tube and 4 for wires. All other items have code 5 or more. While selling the goods, a sales tax of 8% to motor,12% to fan,5% to tube light,7.5% to wires and 3% for all other items is charged. A list containing the product code and price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

<u>PROGRAM CODE:</u>

```java
class Bill{
   public static void main(String[] args) {

       int[] codes = {1, 2, 3, 4, 5};
```

```
        double[] prices = {1000, 500, 200, 150, 800};


        double totalBill = 0;
        for (int i = 0; i < codes.length; i++) {
            totalBill += calculateBill(codes[i], prices[i]);
        }


        System.out.println("Total Bill: " + totalBill);
System.out.println("23DCS080 Maharshi Patel");

    }

    public static double calculateBill(int code, double price) {
        double bill = 0;
        switch (code) {
            case 1:
                bill = price + (price * 0.08); // 8% tax for motor
                break;
            case 2:
                bill = price + (price * 0.12); // 12% tax for fan
                break;
            case 3:
                bill = price + (price * 0.05); // 5% tax for tube light
                break;
            case 4:
                bill = price + (price * 0.075); // 7.5% tax for wires
                break;
            default:
                bill = price + (price * 0.03); // 3% tax for all other items
                break;
        }
        return bill;
    }
}
```

OUTPUT:



```
C:\Users\maharshi patel\OneDrive\Desktop\java>java Bil
Total Bill: 2835.25
23DCS080 Maharshi Patel
```

CONCLUSION:

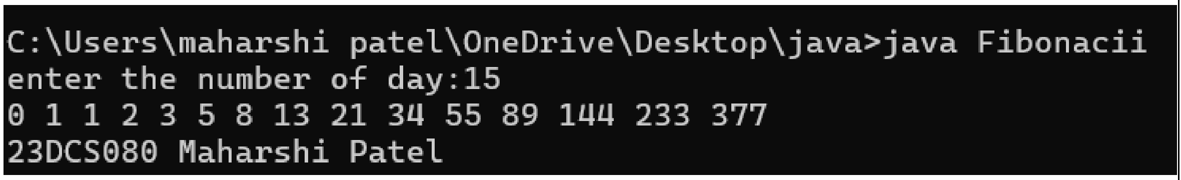
This program demonstrates the use of a switch statement to apply different tax rates based on product codes, calculating the total bill by iterating through the arrays of codes and prices. It showcases a practical application of conditional statements in Java programming.

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 CODE:

```
import java.util.*;
class Fibonacii{
public static void main(String args[]){
Scanner sc=new Scanner(System.in);

int n;
System.out.print("enter the number of day:");
n=sc.nextInt();
int n1=0,n2=1,n3;
System.out.print(n1+" "+n2);
for(int i=2;i<n;++i){
  n3=n1+n2;
System.out.print(" "+n3);
n1=n2;
n2=n3;
}
System.out.println(" ");
System.out.println("23DCS080 Maharshi Patel ");
}}
```

OUTPUT:



```
C:\Users\maharshi patel\OneDrive\Desktop\java>java Fibonacii
enter the number of day:15
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
23DCS080 Maharshi Patel
```

CONCLUSION:

This program demonstrates the use of a loop to generate the Fibonacci series, calculating the exercise duration for each day based on the user-input number of days. It showcases a practical application of mathematical concepts in Java programming, providing a fun and interactive way to generate an exercise routine.

SET-2

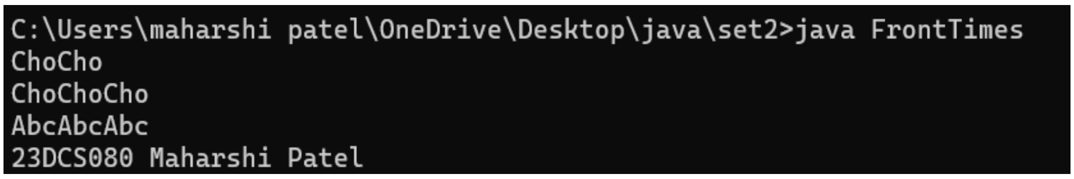
7. Given a string and a non-negative int n, we'll say that the front of the string is the first 3 chars, or whatever is there if the string is less than length 3. Return n copies of the front;

front_times('Chocolate', 2) → 'ChoCho'
front_times('Chocolate', 3) → 'ChoChoCho'
front_times('Abc', 3) → 'AbcAbcAbc'

PROGRAM CODE:

```
import java.util.*;
class FrontTimes{
 public static void main(String[] args) {
        System.out.println(frontTimes("Chocolate", 2));
        System.out.println(frontTimes("Chocolate", 3));
        System.out.println(frontTimes("Abc", 3));
System.out.println("23DCS080 Maharshi Patel ");


    }
    public static String frontTimes(String str, int n) {
        String front = str.substring(0, Math.min(3, str.length()));String
        result = "";
        for (int i = 0; i < n; i++) {
            result += front;
        }
        return result;
    }
}
```

OUTPUT:



```
C:\Users\maharshi patel\OneDrive\Desktop\java\set2>java FrontTimes
ChoCho
ChoChoCho
AbcAbcAbc
23DCS080 Maharshi Patel
```

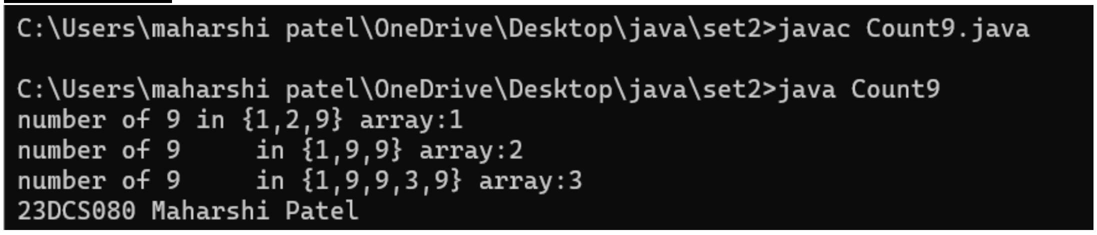
CONCLUSION:

To solve the problem of generating n copies of the front part of a stringin Java, use the substring function to extract the first three characters (or the entire string if it's shorter). Then, repeat this segment n times.

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

PROGRAM CODE:

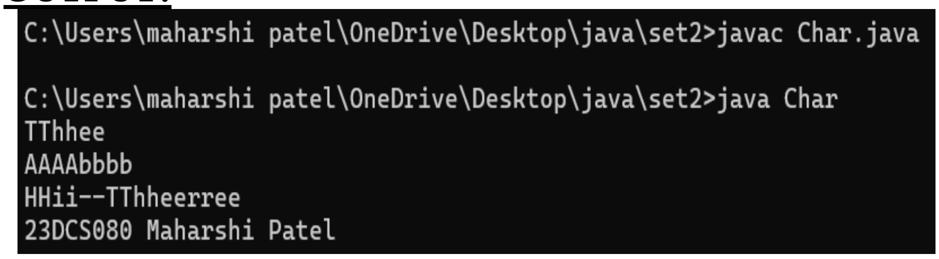
```
import java.util.*;
class Count9{
public static int array_count9(int[] nums)
 { int count = 0;
for (int num : nums) { if (num == 9) {
count++;
}
}
return count;
}
public static void main(String[] args)
 { int[] nums1 = {1, 2, 9};
int[] nums2 = {1, 9, 9};
int[] nums3 = {1, 9, 9, 3, 9};
System.out.println("number of 9 in {1,2,9} array:"+ array_count9(nums1));
```

	// Output: 1
	System.out.println("number of 9 in {1,9,9} array:"+ array_count9(nums2)); // Output: 2
	System.out.println("number of 9 in {1,9,9,3,9} array:"+array_count9(nums3)); // Output: 3
	System.out.println("23DCS080 Maharshi Patel ");
	}
	}
	OUTPUT:
	
	C:\Users\maharshi patel\OneDrive\Desktop\java\set2>javac Count9.java
	
	C:\Users\maharshi patel\OneDrive\Desktop\java\set2>java Count9
	number of 9 in {1,2,9} array:1
	number of 9 in {1,9,9} array:2
	number of 9 in {1,9,9,3,9} array:3
	23DCS080 Maharshi Patel
	CONCLUSION:
	In this practical we learn how to use function argument,use the count varible And print number of int in the array is same.

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 CODE:
	import java.util.*;
	class Char{
	public static String double_char(String str) {
	String result = "";
	for (int i = 0; i < str.length(); i++) {
	result += str.substring(i, i + 1) + str.substring(i, i + 1);
	}
	return result;
	}
	
	public static void main(String[] args) {
	System.out.println(double_char("The"));
	System.out.println(double_char("AAbb"));
	System.out.println(double_char("Hi-There"));
	System.out.println("23DCS080 Maharshi Patel ");

```
}
}
```

OUTPUT:



```
C:\Users\maharshi patel\OneDrive\Desktop\java\set2>javac Char.java

C:\Users\maharshi patel\OneDrive\Desktop\java\set2>java Char
TThhee
AAAAbbbb
HHii--TThheerree
23DCS080 Maharshi Patel
```

CONCLUSION:

To double every character in a string in Java, iterate through each character, appending it twice to a new string. This method ensures each character is duplicated, creating a new string with repeated characters. Itprovides an efficient and straightforward solution for string manipulation tasks.

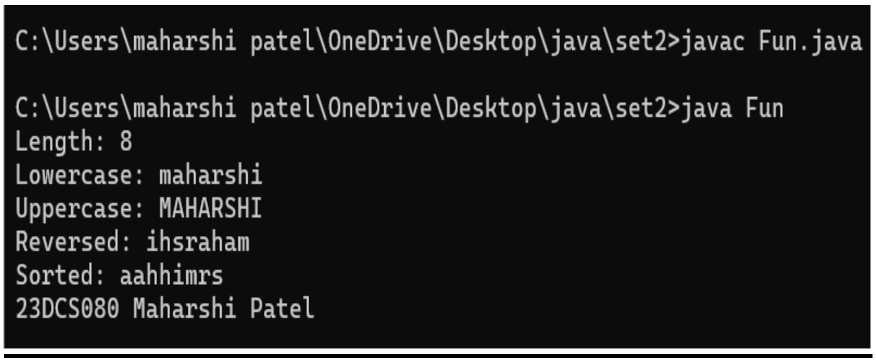
10 Perform following functionalities of the string:

- Find Length of the String
- Lowercase of the String
- Uppercase of the String
- Reverse String, Sort the string

PROGRAM CODE:

```
        import java.util.*;
        class Fun{
        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();
}
public static String sortString(String str) { char chars[] =
str.toCharArray(); Arrays.sort(chars);
return new String(chars);
}
public static void main(String args[]){ String str =
"maharshi";

int length = findLength(str); System.out.println("Length:
" + length);

String lowercase = toLowercase(str);
System.out.println("Lowercase: " + lowercase);

String uppercase = toUppercase(str);
System.out.println("Uppercase: " + uppercase);

String reversed = reverseString(str);
System.out.println("Reversed: " + reversed);

String sorted = sortString(str);
System.out.println("Sorted: " + sorted);
System.out.println("23DCS080 Maharshi Patel ");
}
}
```

OUTPUT:



```
C:\Users\maharshi patel\OneDrive\Desktop\java\set2>javac Fun.java

C:\Users\maharshi patel\OneDrive\Desktop\java\set2>java Fun
Length: 8
Lowercase: maharshi
Uppercase: MAHARSHI
Reversed: ihsraham
Sorted: aahhimrs
23DCS080 Maharshi Patel
```

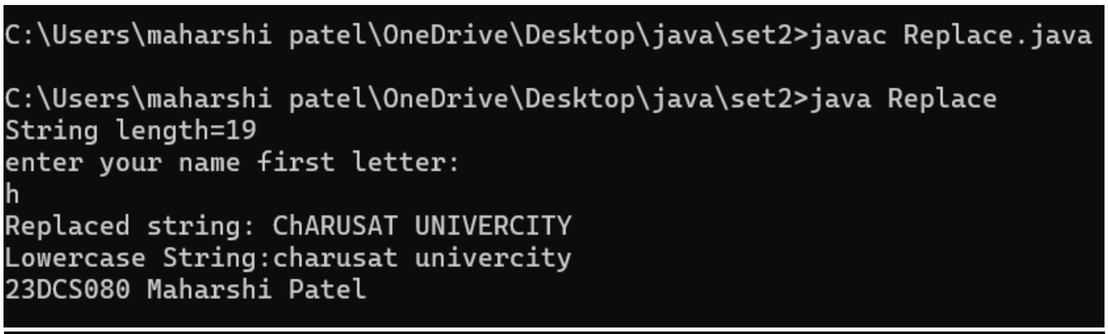
11. Perform following Functionalities of the string: "CHARUSAT UNIVERSITY"

- ● Find length
- ● Replace 'H' by 'FIRST LATTER OF YOUR NAME'
- ● Convert all character in lowercase

PROGRAM CODE:

```
import java.util.*;
class Replace{
public static void main(String args[])
{ String str="CHARUSAT UNIVERCITY";
int length=str.length();
 System.out.println("String length="+length);
Scanner sc=new Scanner(System.in);
System.out.println("enter your name first letter:");
char n;
n=sc.next().charAt(0);
String replacedStr = str.replace('H', n);
System.out.println("Replaced string: " + replacedStr);
 String lowerCaseStr = replacedStr.toLowerCase();
System.out.println("Lowercase String:"+lowerCaseStr);
System.out.println("23DCS080 Maharshi Patel ");

 }
}
```

OUTPUT:



```
C:\Users\maharshi patel\OneDrive\Desktop\java\set2>javac Replace.java

C:\Users\maharshi patel\OneDrive\Desktop\java\set2>java Replace
String length=19
enter your name first letter:
h
Replaced string: ChARUSAT UNIVERCITY
Lowercase String:charusat univercity
23DCS080 Maharshi Patel
```

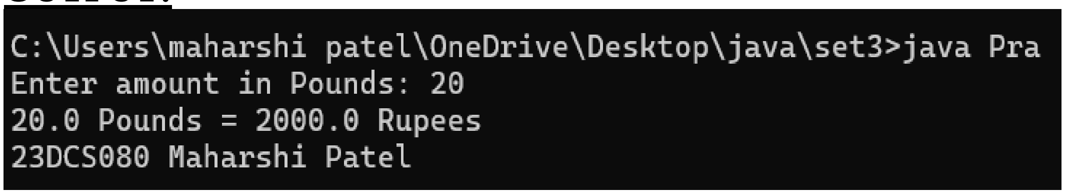
SET-3

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.**

<u>PROGRAM CODE:</u>

```
import java.util.*;

public class Pra {
    public static void main(String[] args) {
        if (args.length == 1) {
                                        // Command-line argument
            double amount = Double.parseDouble(args[0]);
            double rupees = amount*100;
            System.out.println(amount + " Pounds = " + rupees + " Rupees");
        }
 else {
                                        // Interactive mode
            Scanner scanner = new Scanner(System.in);
            System.out.print("Enter amount in Pounds: ");
            double amount = scanner.nextDouble();
            double rupees = amount*100;
            System.out.println(amount + " Pounds = " + rupees + " Rupees");
        }
System.out.println("23DCS080 Maharshi Patel ");


    }


    }
```

<u>OUTPUT:</u>



```
C:\Users\maharshi patel\OneDrive\Desktop\java\set3>java Pra
Enter amount in Pounds: 20
20.0 Pounds = 2000.0 Rupees
23DCS080 Maharshi Patel
```

CONCLUSION:

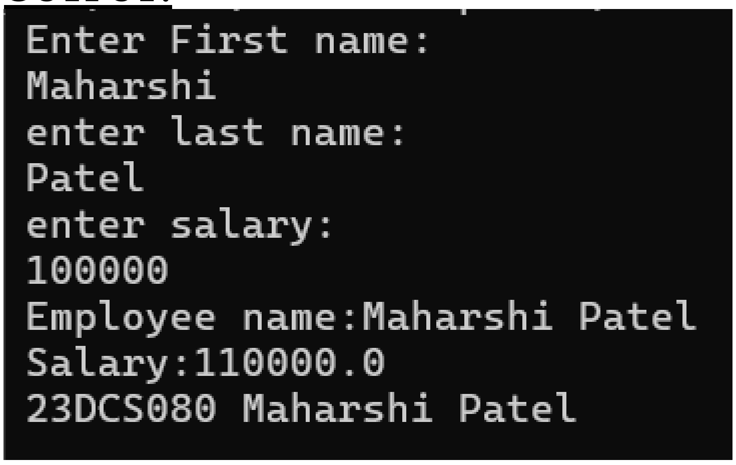
The Java program converts Pounds to Rupees at a fixed rate of 1 Pound = 100 Rupees, handling both command-line arguments and interactive user inputs effectively, ensuring user-friendly currency conversion.

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:

```
import java.util.*;
class Employee{
 Scanner sc=new Scanner(System.in);

 String fs=" ";
 String ls=" ";
 double sal;
Employee(String f,String l,double s){
fs=f;
ls=l;
sal=s;
}
void setfs(){
 fs=sc.nextLine();
}

void setls(){
ls=sc.nextLine();
}

void setsal(){
sal=sc.nextDouble();
if(sal<0){
sal=0.0;
}
else{
sal=sal+(sal*0.10);
}
}

String getfs(){
return fs;
}
String getls(){
return ls;
}
double getsal(){
return sal;
}
}
class EmpT{
public static void main(String args[]){
 Employee E1=new Employee("","",0.0);
System.out.println("Enter First name:");
E1.setfs();
System.out.println("enter last name:");
E1.setls();
System.out.println("enter salary:");
E1.setsal();

System.out.println("Employee name:"+E1.getfs()+" "+E1.getls());
System.out.println("Salary:"+E1.getsal());
System.out.println("23DCS080 Maharshi Patel ");
}
}
```

OUTPUT:



```
Enter First name:
Maharshi
enter last name:
Patel
enter salary:
100000
Employee name:Maharshi Patel
Salary:110000.0
23DCS080 Maharshi Patel
```

CONCLUSION:

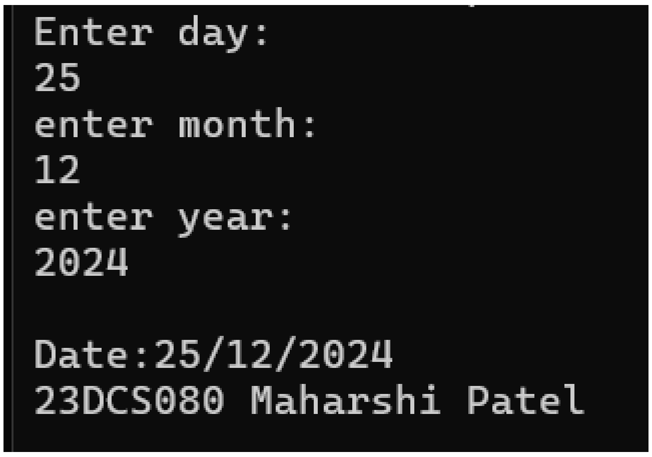
The Employee class initializes and updates employee data through setters and getters. The EmpT class demonstrates capturing employee details and adjusting salaries, showcasing the class's functionality effectively.

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 CODE:

```
import java.util.*;
class Date{
 Scanner sc=new Scanner(System.in);

 int d;
 int m;
 int y;
Date(int day, int month ,int year){
d=day;
m=month;
y=year;
}
void setd(){
 d=sc.nextInt();
if(d>31){
```

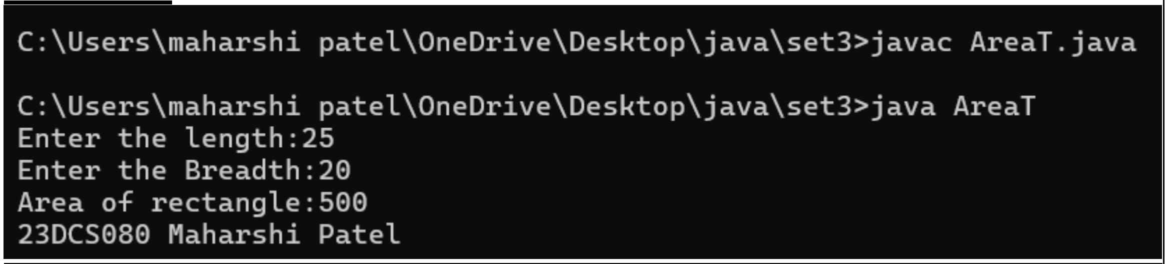
```
System.out.println("enter right day");
}
}

void setm(){
m=sc.nextInt();
if(m>12){
System.out.println("enter right month");
}

}

void sety(){
y=sc.nextInt();
}

int getd(){
return d;
}
int getlm(){
return m;
}
int gety(){
return y;
}
void displayDate(){

System.out.println("\nDate:"+d+"/"+m+"/"+y);
}
}
class DateTest{
public static void main(String args[]){
 Date D=new Date(0,0,0);
System.out.println("Enter day:");
D.setd();
System.out.println("enter month:");
D.setm();
System.out.println("enter year:");
D.sety();
D.displayDate();
System.out.println("23DCS080 Maharshi Patel ");
```

	}
	}
	
	OUTPUT:
	
	
	```
	Enter day:
	25
	enter month:
	12
	enter year:
	2024

	Date:25/12/2024
	23DCS080 Maharshi Patel
	```
	
	CONCLUSION:
	The Date class successfully manages date information with proper setters, getters, and a display method. The DateTest application effectively demonstrates the class's ability to handle and display date information.

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 CODE:
	import java.util.*;
	class Area{
	int l, b;
	 Scanner sc=new Scanner(System.in);
	
	Area(){
	}
	Area(int length, int breadth){
	b=breadth;
	l=length;
	}
	void setl(){

```
l=sc.nextInt();
}
void setb(){
b=sc.nextInt();
}

void returnArea(){
System.out.println(l*b);
}
}

class AreaT{

public static void main(String args[]){
 Area a=new Area(0,0);
System.out.print("Enter the length:");
a.setl();
System.out.print("Enter the Breadth:");
a.setb();
System.out.print("Area of rectangle:");
a.returnArea();
System.out.println("23DCS080 Maharshi Patel ");
}
}
```

OUTPUT:



```
C:\Users\maharshi patel\OneDrive\Desktop\java\set3>javac AreaT.java

C:\Users\maharshi patel\OneDrive\Desktop\java\set3>java AreaT
Enter the length:25
Enter the Breadth:20
Area of rectangle:500
23DCS080 Maharshi Patel
```

CONCLUSION:

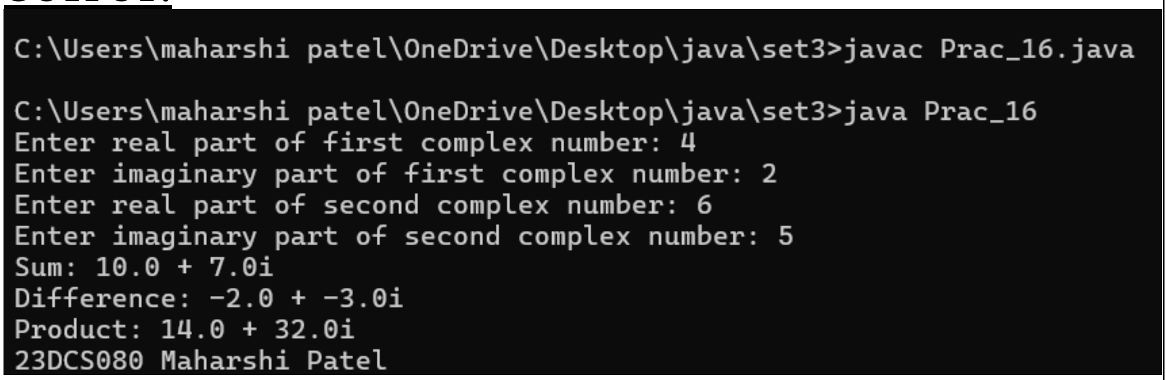
The program effectively calculates and prints the area of a rectangle using the Area class, which initializes length and breadth via constructor parameters and provides a method to return the calculated area.

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 CODE:

```
import java.util.Scanner;

class Complex {
    private double real;
    private double imag;

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

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

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

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

    public String toString() {
        return real + " + " + imag + "i";
    }
}

public class Prac_16 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter real part of first complex number: ");
        double real1 = scanner.nextDouble();
        System.out.print("Enter imaginary part of first complex number: ");
        double imag1 = scanner.nextDouble();

        System.out.print("Enter real part of second complex number: ");
        double real2 = scanner.nextDouble();
        System.out.print("Enter imaginary part of second complex number: ");
        double imag2 = scanner.nextDouble();

        Complex num1 = new Complex(real1, imag1);
        Complex num2 = new Complex(real2, imag2);

        Complex sum = num1.add(num2);
        Complex difference = num1.subtract(num2);
        Complex product = num1.multiply(num2);

        System.out.println("Sum: " + sum);
        System.out.println("Difference: " + difference);
        System.out.println("Product: " + product);
System.out.println("23DCS080 Maharshi Patel ");
    }
 }
```

OUTPUT:



```
C:\Users\maharshi patel\OneDrive\Desktop\java\set3>javac Prac_16.java

C:\Users\maharshi patel\OneDrive\Desktop\java\set3>java Prac_16
Enter real part of first complex number: 4
Enter imaginary part of first complex number: 2
Enter real part of second complex number: 6
Enter imaginary part of second complex number: 5
Sum: 10.0 + 7.0i
Difference: -2.0 + -3.0i
Product: 14.0 + 32.0i
23DCS080 Maharshi Patel
```

CONCLUSION:

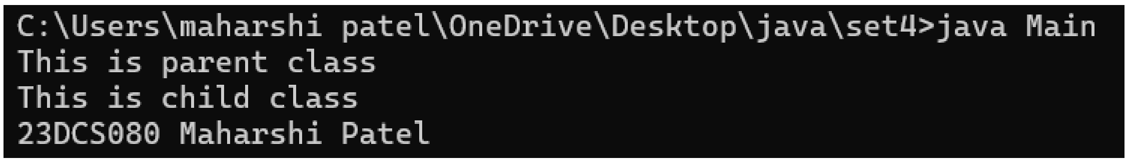
The Complex class correctly performs addition, subtraction, and multiplication of two complex numbers, with methods for each operation. User input for real and imaginary parts demonstrates the class's functionality.

	SET-4
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.**

PROGRAM CODE:

```
class Parent {

    void displayParent() {
        System.out.println("This is parent class");
    }
}

class Child extends Parent {

    void displayChild() {
        System.out.println("This is child class");
    }
}


public class Main {
    public static void main(String[] args) {

        Parent parentObject = new Parent();

        parentObject.displayParent();


        Child childObject = new Child();

        childObject.displayChild();
System.out.println("23DCS080 Maharshi Patel ");


    }
}
```

OUTPUT:



```
C:\Users\maharshi patel\OneDrive\Desktop\java\set4>java Main
This is parent class
This is child class
23DCS080 Maharshi Patel
```

CONCLUSION:

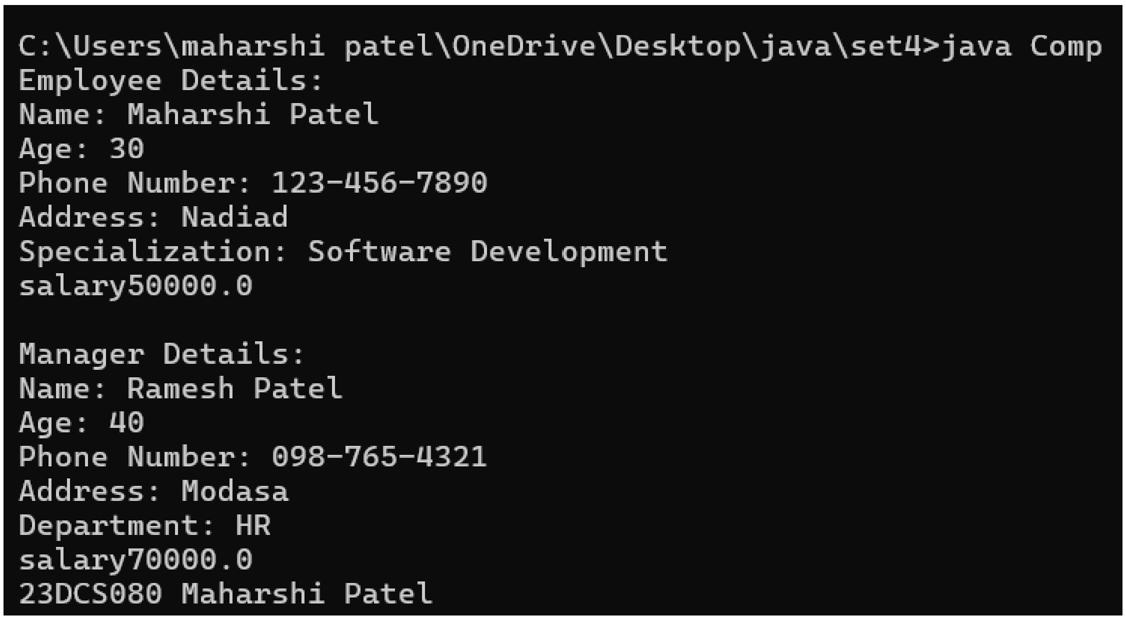
To inherit the class , we use the “extends” keyword .

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.

PROGRAM CODE:

```
class Member{
    String name;
    int age;
    String phonenumber;
    String address;
    double salary;

    void printsalary(){
        System.err.println("salary" + salary);
    }

}

class Employee extends Member{
    String specialization;

}

class Manager extends Member{

    String department;

}
public class Comp {
    public static void main(String[] args) {
        Employee employee = new Employee();

        // employee

        employee.name = "Maharshi Patel";
        employee.age = 30;
        employee.phonenumber = "123-456-7890";
        employee.address = "Nadiad";
        employee.salary = 50000;
        employee.specialization = "Software Development";

        // Print employee's details
        System.out.println("Employee Details:");
        System.out.println("Name: " + employee.name);
        System.out.println("Age: " + employee.age);
        System.out.println("Phone Number: " + employee.phonenumber);
        System.out.println("Address: " + employee.address);
        System.out.println("Specialization: " + employee.specialization);
        employee.printsalary();


        Manager manager = new Manager();

        manager.name = "Ramesh Patel";
        manager.age = 40;
        manager.phonenumber = "098-765-4321";
        manager.address = "Modasa";
        manager.salary = 70000;
        manager.department = "HR";
        //  manager
        System.out.println("\nManager Details:");
        System.out.println("Name: " + manager.name);
        System.out.println("Age: " + manager.age);
        System.out.println("Phone Number: " + manager.phonenumber);
        System.out.println("Address: " + manager.address);
        System.out.println("Department: " + manager.department);
        manager.printsalary();

        System.out.println("23DCS080 Maharshi Patel ");
    }

    }
```

OUTPUT:



```
C:\Users\maharshi patel\OneDrive\Desktop\java\set4>java Comp
Employee Details:
Name: Maharshi Patel
Age: 30
Phone Number: 123-456-7890
Address: Nadiad
Specialization: Software Development
salary50000.0

Manager Details:
Name: Ramesh Patel
Age: 40
Phone Number: 098-765-4321
Address: Modasa
Department: HR
salary70000.0
23DCS080 Maharshi Patel
```

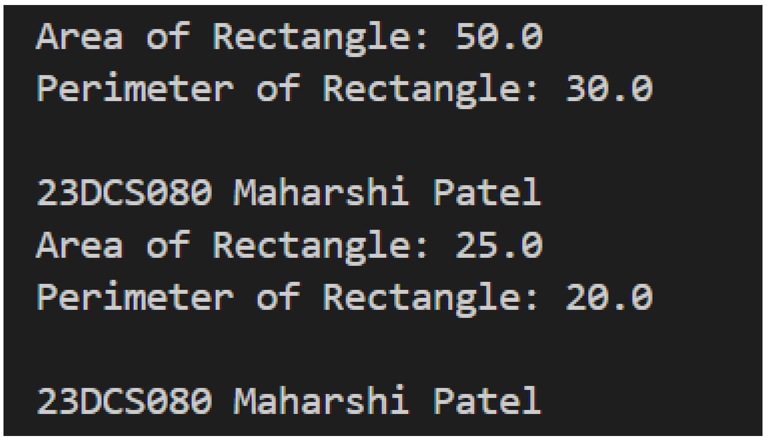
CONCLUSION:

This example demonstrates how to create a base class with common data members and methods, and how to extend this class to create more specific classes while adding unique attributes.

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 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.**

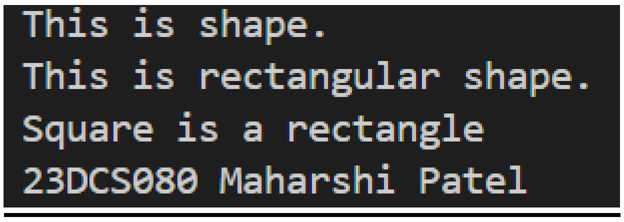
PROGRAM CODE:

```
class Rectangle {
    double length, breadth;

    // Constructor
    Rectangle(double length, double breadth) {
        this.length = length;
        this.breadth = breadth;
    }

    // Method to print area
    void printArea() {
        System.out.println("Area of Rectangle: " + (length * breadth));
    }

    // Method to print perimeter
    void printPerimeter() {
        System.out.println("Perimeter of Rectangle: " + 2 * (length +
breadth));
    }
}

// Class Square inheriting Rectangle
class Square extends Rectangle {
    // Constructor
    Square(double side) {
        super(side, side);
    }
}
public class P19 {
    public static void main(String[] args) {
        // Array of objects
        Rectangle[] rectangles = new Rectangle[2];
```

	```
	        // Creating objects
	        rectangles[0] = new Rectangle(5, 10);
	        rectangles[1] = new Square(5);

	        // Printing area and perimeter
	        for (Rectangle rectangle : rectangles) {
	            rectangle.printArea();
	            rectangle.printPerimeter();
	            System.out.println();
	System.out.println("23DCS080 Maharshi Patel ");

	        }
	    }
	}
	```
	OUTPUT:
	
	```
	Area of Rectangle: 50.0
	Perimeter of Rectangle: 30.0

	23DCS080 Maharshi Patel
	Area of Rectangle: 25.0
	Perimeter of Rectangle: 20.0

	23DCS080 Maharshi Patel
	```

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.**
	PROGRAM CODE:
	```
	class Shape{
	    public static void printShape(){
	```

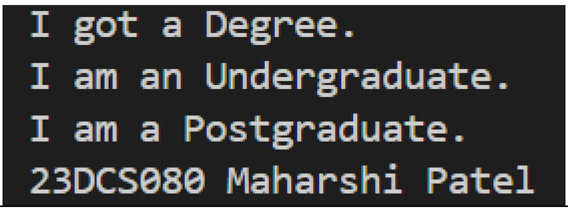
```
        System.out.println("This is shape.");
    }

    public void printDetails() {
        // TODO Auto-generated method stub
        throw new UnsupportedOperationException("Unimplemented
method 'printDetails'");
    }
}
class Rectangle extends Shape{
    public static void printRectangle(){
        System.out.println("This is rectangular shape.");
    }
}

class Circle extends Shape{
    public static void printCircle(){
        System.out.println("This is circular shape.");
    }
}
class Square extends Rectangle{
    public static void printSquare(){
        System.out.println("Square is a rectangle");
    }
}

public class P20 {
    public static void main(String[] args) {
        Square s=new Square();
        s.printShape();
        s.printRectangle();
        s.printSquare();

    }

}
```

	OUTPUT:  `This is shape.` `This is rectangular shape.` `Square is a rectangle` `23DCS080 Maharshi Patel`

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.**
	 PROGRAM CODE:
	 `class Degree{`
	 `    void getDegree(){`
	 `        System.out.println("I got a degree.");`
	 `    }`
	 `}`
	 `class Undergraduate extends Degree{`
	 `    void getDegree(){`
	 `        System.out.println("I am an Undergraduate.");`
	 `    }`
	 `}`
	 `class Postgraduate extends Degree{`
	 `    void getDegree(){`
	 `        System.out.println("I am a Postgraduate.");`
	 `    }`
	 `}`
	 `public class P21 {`
	 `    public static void main(String[] args) {`
	 `        Degree D=new Degree();`
	 `        Undergraduate UG=new Undergraduate();`
	 `        Postgraduate PG=new Postgraduate();`
	
	 `        D.getDegree();`

```
        UG.getDegree();
        PG.getDegree();
System.out.println("23DCS080 Maharshi Patel ");

    }
}
```

<u>OUTPUT:</u>



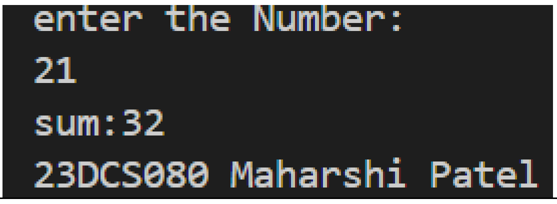
```
I got a Degree.
I am an Undergraduate.
I am a Postgraduate.
23DCS080 Maharshi Patel
```

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.

<u>PROGRAM CODE:</u>

```
import java.util.*;
interface AdvancedArithmetic {
    int divisor_sum(int n);

}
class calledMyCalculator implements AdvancedArithmetic{
    public int divisor_sum(int n){
        int sum=0;
        int sqrt =(int) Math.sqrt(n);
        for(int i=1;i<=sqrt;i++){
            if(n%i==0){
                sum+=i;
                if(i!=n/i){
                    sum +=n/i;
                }
            }
        }
        return sum;
    }
}

class P22{
    static Scanner sc=new Scanner(System.in);
    public static void main(String args[]){
     calledMyCalculator cmc=new calledMyCalculator();
     System.out.println("enter the Number:");
     int x=sc.nextInt();

     System.out.println("sum:"+cmc.divisor_sum(x));
System.out.println("23DCS080 Maharshi Patel ");


    }
}
```

OUTPUT:



```
enter the Number:
21
sum:32
23DCS080 Maharshi Patel
```

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.**

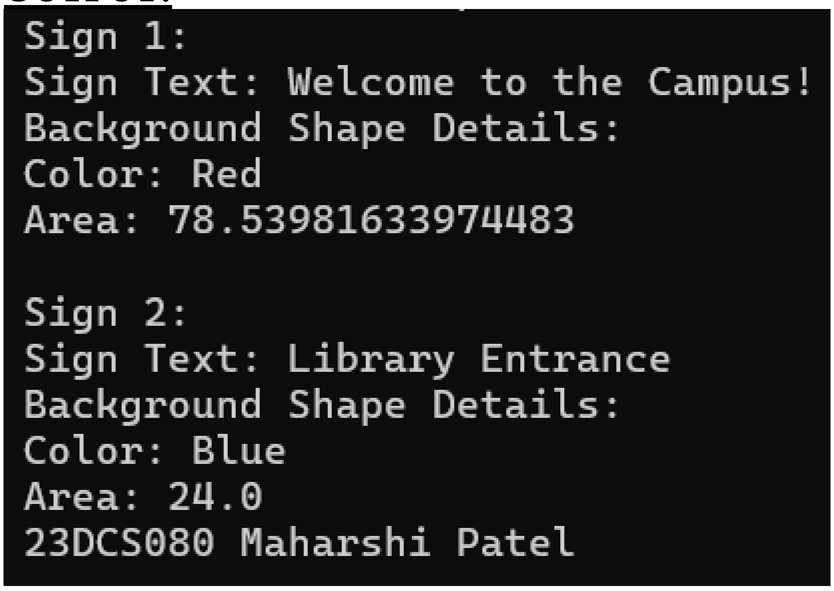
PROGRAM CODE:

```
// Shape.java
interface Shape {
    String getColor();
    double getArea();

    default void printDetails() {
        System.out.println("Color: " + getColor());
        System.out.println("Area: " + getArea());
    }
}

// Circle.java
 class Circle implements Shape {
    private double radius;
    private String color;

    public Circle(double radius, String color) {
        this.radius = radius;
        this.color = color;
    }

    @Override
    public String getColor() {
        return color;
    }

    @Override
    public double getArea() {
        return Math.PI * radius * radius;
    }
}
```

```
// Rectangle.java
 class Rectangle implements Shape {
    private double length;
    private double width;
    private String color;

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

    @Override
    public String getColor() {
        return color;
    }

    @Override
    public double getArea() {
        return length * width;
    }
}

// Sign.java
 class Sign {
    private Shape backgroundShape;
    private String text;

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

    public void display() {
        System.out.println("Sign Text: " + text);
        System.out.println("Background Shape Details:");
        backgroundShape.printDetails();
    }
}
```

```
// Main.java
public class Per_23 {
    public static void main(String[] args) {
        Shape circle = new Circle(5, "Red");
        Shape rectangle = new Rectangle(4, 6, "Blue");

        Sign sign1 = new Sign(circle, "Welcome to the Campus!");
        Sign sign2 = new Sign(rectangle, "Library Entrance");

        System.out.println("Sign 1:");
        sign1.display();

        System.out.println("\nSign 2:");
        sign2.display();
System.out.println("23DCS080 Maharshi Patel ");

    }
}
```

OUTPUT:



```
Sign 1:
Sign Text: Welcome to the Campus!
Background Shape Details:
Color: Red
Area: 78.53981633974483

Sign 2:
Sign Text: Library Entrance
Background Shape Details:
Color: Blue
Area: 24.0
23DCS080 Maharshi Patel
```

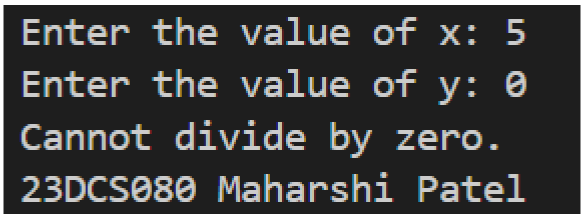
SET 5

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 CODE:

```
import java.util.*;

public class Per_24 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        try {
            System.out.print("Enter the value of x: ");
            int x = Integer.parseInt(scanner.nextLine());
            System.out.print("Enter the value of y: ");
            int y = Integer.parseInt(scanner.nextLine());

            // Perform the division
            int result = x / y;

            System.out.println("The result of " + x + " / " + y + " = " + result);
        }
        catch (NumberFormatException e) {
            System.out.println("Input is not a valid integer.");
        }
        catch (ArithmeticException e) {
            System.out.println("Cannot divide by zero.");
        }
        finally {
            // Close the scanner to prevent resource leakage
            scanner.close();
        }
System.out.println("23DCS080 Maharshi Patel ");

    }
}
```

OUTPUT:



```
Enter the value of x: 5
Enter the value of y: 0
Cannot divide by zero.
23DCS080 Maharshi Patel
```

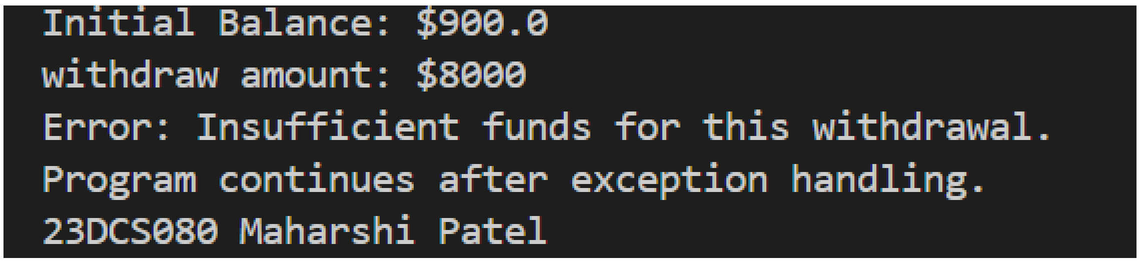
CONCLUSION:

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

PROGRAM CODE:

```
class BankAccount {
    private double balance;

    public BankAccount(double initialBalance) {
        this.balance = initialBalance;
    }

    public void withdraw(double amount) {
        if (amount > balance) {
            throw new RuntimeException("Insufficient funds for this
withdrawal.");
        }
        balance -= amount;
        System.out.println("Successfully withdrew: $" + amount);
    }

    public double getBalance() {
        return balance;
    }
}

public class Per_25 {
    public static void main(String[] args) {
        try {
            BankAccount account = new BankAccount(900);

            System.out.println("Initial Balance: $" + account.getBalance());
           System.out.println("withdraw amount: $600");

            account.withdraw(8000);
            System.out.println("Aveleble Balance: $" + account.getBalance());

        } catch (RuntimeException e) {

            System.out.println("Error: " + e.getMessage());
        }

       System.out.println("Program continues after exception handling.");
System.out.println("23DCS080 Maharshi Patel ");

    }

}
```

OUTPUT:



```
Initial Balance: $900.0
withdraw amount: $8000
Error: Insufficient funds for this withdrawal.
Program continues after exception handling.
23DCS080 Maharshi Patel
```

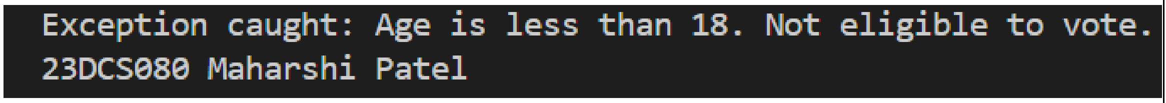
CONCLUSION:

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 CODE:

```
class InvalidAgeException extends Exception {
    public InvalidAgeException(String message) {
        super(message);
    }
}

public class Per_26 {
        public static void checkAge(int age) throws InvalidAgeException {
        if (age < 18) {
            throw new InvalidAgeException("Age is less than 18. Not eligible to
vote.");
        } else {
            System.out.println("You are eligible to vote.");
        }
    }

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

            checkAge(16);
        }
        catch (InvalidAgeException e) {
            System.out.println("Exception caught: " + e.getMessage());
        }
System.out.println("23DCS080 Maharshi Patel ");

    }
}
```

OUTPUT:



```
Exception caught: Age is less than 18. Not eligible to vote.
23DCS080 Maharshi Patel
```

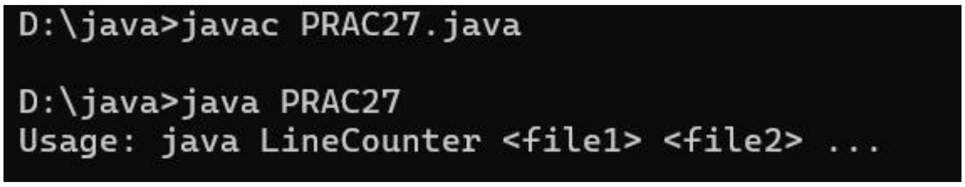
CONCLUSION:

By this experiment I learnt how to implement checked exception and unchecked exception.

	SET 6
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.

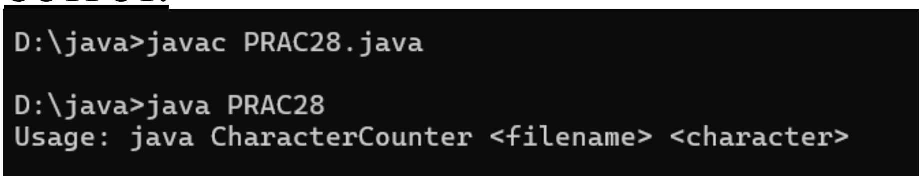
PROGRAM CODE:

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

public class PRAC27 {
    public static void main(String[] args) {
        if (args.length == 0) {
            System.out.println("Usage: java LineCounter <file1> <file2> ...");
            return;
        }

        for (String filename : args) {
            try {
                int lineCount = countLines(filename);
                System.out.println(filename + ": " + lineCount + " lines");
            } catch (IOException e) {
                System.err.println("Error reading file: " + filename);
            }
        }
    }

    private static int countLines(String filename) throws IOException {
        try (BufferedReader reader = new BufferedReader(new FileReader(filename))) {
            int lines = 0;
            while (reader.readLine() != null) {
                lines++;
            }
            return lines;
        }
    }
}
```

	OUTPUT:  `D:\java>javac PRAC27.java` `D:\java>java PRAC27` `Usage: java LineCounter <file1> <file2> ...` **CONCLUSION:** This Java program, PRAC27, reads multiple file names from command-line arguments and counts the number of lines in each file. It uses a BufferedReader to read each file and handle any IOException that may occur. The results are printed to the console, displaying the line count for each specified file.
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.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class PRAC28 {
    public static void main(String[] args) {
        if (args.length != 2) {
            System.out.println("Usage: java CharacterCounter <filename>
<character>");
            return;
        }

        String filename = args[0];
        char targetChar = args[1].charAt(0);

        try {
            int charCount = countCharacterOccurrences(filename, targetChar);
            System.out.println("Occurrences of '" + targetChar + "' in " + filename
+ ": " + charCount);
        } catch (IOException e) {
            System.err.println("Error reading file: " + filename);
        }
    }

    private static int countCharacterOccurrences(String filename, char targetChar) throws IOException {
        try (BufferedReader reader = new BufferedReader(new FileReader(filename))) {
            int count = 0;
            int currentChar;
            while ((currentChar = reader.read()) != -1) {
                if (currentChar == targetChar) {
                    count++;
                }
            }
            return count;
        }
    }
}
```

OUTPUT:



```
D:\java>javac PRAC28.java

D:\java>java PRAC28
Usage: java CharacterCounter <filename> <character>
```

CONCLUSION:

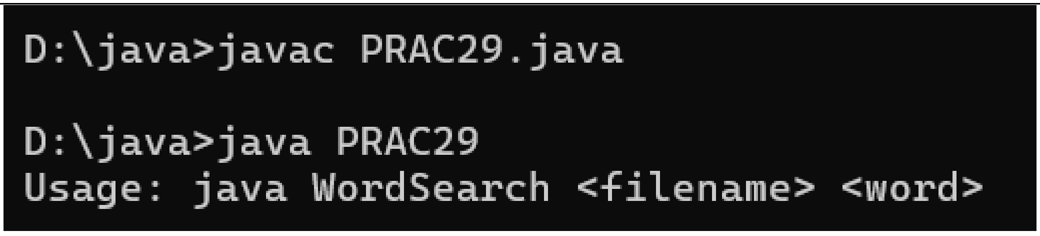
The PRAC28 Java program counts the occurrences of a specified character in a given file, using command-line arguments for the file name and target character. It reads the file character-by-character with a BufferedReader and handles any IOException that might occur. The result displays the total count of the target character in the specified file.

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

PROGRAM CODE:

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

public class PRAC29 {
    public static void main(String[] args) {
        if (args.length != 2) {
            System.out.println("Usage: java WordSearch <filename> <word>");
            return;
        }

        String filename = args[0];
        String targetWord = args[1];

        try {
            boolean found = searchWord(filename, targetWord);
            if (found) {
                System.out.println("Word '" + targetWord + "' found in " +
filename);
            } else {
                System.out.println("Word '" + targetWord + "' not found in " +
filename);
            }
        } catch (IOException e) {
            System.err.println("Error reading file: " + filename);
        }
    }

    private static boolean searchWord(String filename, String targetWord)
throws IOException {
        try (BufferedReader reader = new BufferedReader(new
FileReader(filename))) {
            String line;
            while ((line = reader.readLine()) != null) {
                if (line.contains(targetWord)) {
                    return true;
                }
            }
            return false;
        }
    }
}
```

OUTPUT:



```
D:\java>javac PRAC29.java

D:\java>java PRAC29
Usage: java WordSearch <filename> <word>
```

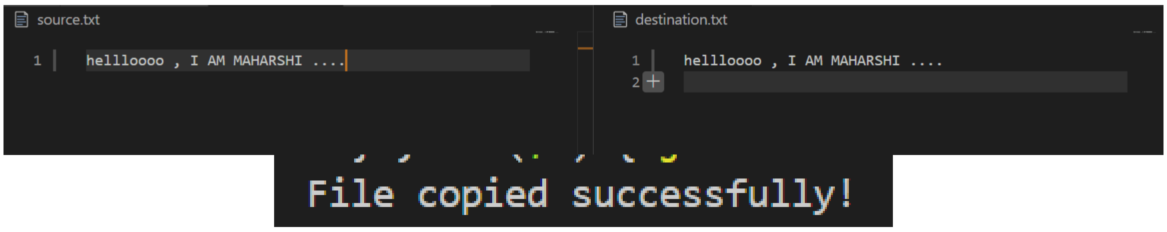
CONCLUSION:
The PRAC29 Java program searches for a specified word in a given file using command-line arguments for the file name and target word. It reads the file line-by-line with a BufferedReader and checks if any line contains the word. The result indicates whether the word was found or not in the specified file.

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 CODE:

```
import java.io.*;

public class PRAC30{
    public static void main(String[] args) {
        String sourceFile = "source.txt";
        String destinationFile = "destination.txt";

        try (BufferedReader reader = new BufferedReader(new
FileReader(sourceFile));
             BufferedWriter writer = new BufferedWriter(new
FileWriter(destinationFile))) {

            String line;
            while ((line = reader.readLine()) != null) {
                writer.write(line);
                writer.newLine();
            }

            System.out.println("File copied successfully!");

        } catch (IOException e) {
            System.out.println("An error occurred: " + e.getMessage());
        }
    }
}
```

OUTPUT:



```
helllоооо , I AM MAHARSHI ....
```

```
helllоооо , I AM MAHARSHI ....
```

```
File copied successfully!
```

CONCLUSION:

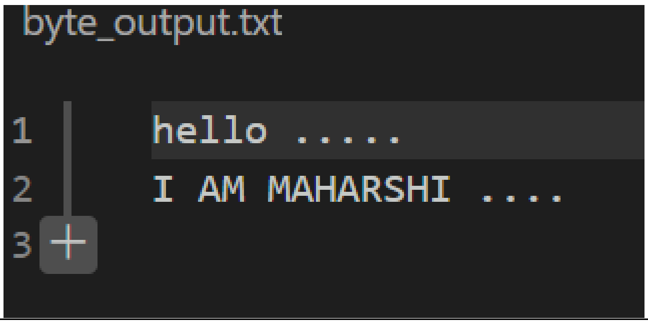
- BufferedReader: Reads the data from the source file in an efficient manner by buffering characters.
- BufferedWriter: Writes the data to the destination file and buffers it for efficient output.
- If the destination file (destination.txt) does not exist, it is automatically created by FileWriter.

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.

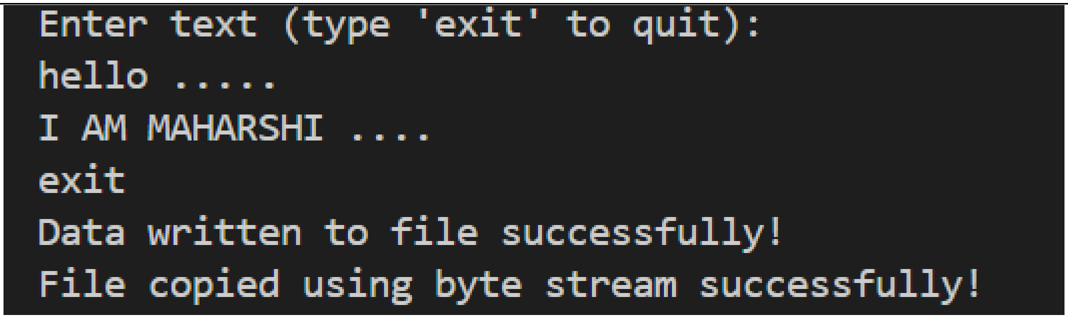
PROGRAM CODE:

```
import java.io.*;

public class PRAC31 {
    public static void main(String[] args) {
        String filePath = "output.txt";

        try (BufferedReader consoleReader = new BufferedReader(new
InputStreamReader(System.in));
             BufferedWriter fileWriter = new BufferedWriter(new
FileWriter(filePath))) {

            System.out.println("Enter text (type 'exit' to quit):");

            String line;
            while (!(line = consoleReader.readLine()).equalsIgnoreCase("exit"))
{
                fileWriter.write(line);
                fileWriter.newLine();
            }

            System.out.println("Data written to file successfully!");

        } catch (IOException e) {
            System.out.println("An error occurred: " + e.getMessage());
        }

        try (FileInputStream fis = new FileInputStream(filePath);
             FileOutputStream fos = new FileOutputStream("byte_output.txt")) {

            int byteData;
            while ((byteData = fis.read()) != -1) {
                fos.write(byteData);
            }

            System.out.println("File copied using byte stream successfully!");

        } catch (IOException e) {
            System.out.println("An error occurred: " + e.getMessage());
        }
    }
}
```

OUTPUT:


byte_output.txt

```
hello .....
I AM MAHARSHI ....
```



```
Enter text (type 'exit' to quit):
hello .....
I AM MAHARSHI ....
exit
Data written to file successfully!
File copied using byte stream successfully!
```

CONCLUSION:

This program reads lines of text input from the console and writes them to a file named output.txt. It continues to read input until the user types "exit."

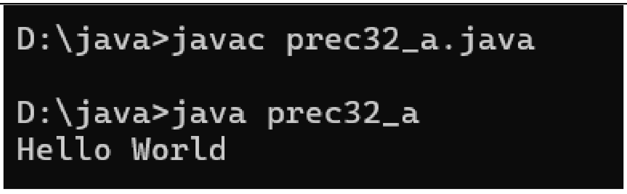
SET 7

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

PROGRAM CODE:

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

public class prec32_a {
    public static void main(String[] args) {
        MyThread t1 = new MyThread();
        t1.start();
    }
}
```

OUTPUT:

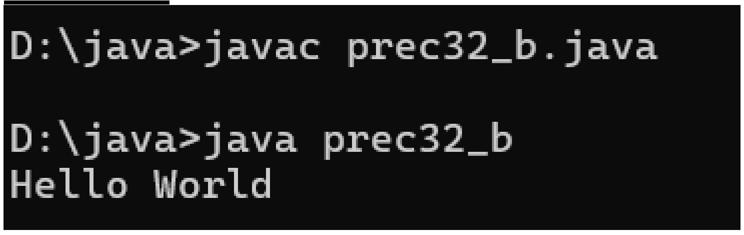


```
D:\java>javac prec32_a.java

D:\java>java prec32_a
Hello World
```

PROGRAM CODE:

```
class MyRunnable implements Runnable {
    public void run() {
        System.out.println("Hello World");
    }
}

public class prec32_b {
    public static void main(String[] args) {
        MyRunnable m1 =  new MyRunnable();
        Thread t1 = new Thread(m1);
        t1.start();
    }

}
```

OUTPUT:



```
D:\java>javac prec32_b.java

D:\java>java prec32_b
Hello World
```

CONCLUSION:
In both examples, threads are used to execute the run() method. In the first case, MyThread directly extends Thread and overrides its run() method, allowing the thread to be started using t1.start(). In the second example, MyRunnable implements the Runnable interface, and its run() method is passed to a Thread object. Both approaches are valid for multithreading in Java, but implementing Runnable is preferred in most cases since it allows the class to extend other classes as well.

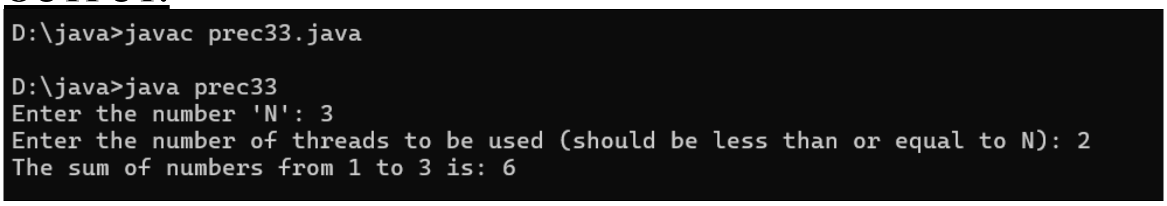
33. Write a program which takes N and number of threads as an argument. Program should distribute the task of summation of N numbers amongst number of threads and final result to be displayed on the console.

PROGRAM CODE:

```
import java.util.Scanner;

class MyThread extends Thread {
    int start, end;
    static int sum = 0;

    // Constructor to initialize the range for each thread
    MyThread(int start, int end) {
        this.start = start;
        this.end = end;
    }

    // Synchronized method to safely add the sum from each thread
    static void addSum(int partialSum) {
        sum += partialSum;
    }

    // The task that each thread will perform
    public void run() {
        int partialSum = 0;
        for (int i = start; i <= end; i++) {
            partialSum += i;
        }
        addSum(partialSum);  // Add the partial sum from this thread to the total
sum
    }
}

public class prec33 {
    public static void main(String[] args) {
        int N, numThreads;
        Scanner s = new Scanner(System.in);

        // Getting input for the number N and number of threads
        System.out.print("Enter the number 'N': ");
        N = s.nextInt();
        System.out.print("Enter the number of threads to be used (should be less
than or equal to N): ");
```

```
        numThreads = s.nextInt();

        // Array to hold the thread objects
        MyThread[] threads = new MyThread[numThreads];

        // Distribute the range of numbers across threads
        int range = N / numThreads;
        int start = 1, end;

        for (int i = 0; i < numThreads; i++) {
            end = (i == numThreads - 1) ? N : start + range - 1; // Ensure last thread
handles the remainder
            threads[i] = new MyThread(start, end); // Create thread with the range
of numbers
            threads[i].start();  // Start the thread
            start = end + 1;
        }

        try {
            for (int i = 0; i < numThreads; i++) {
                threads[i].join();
            }
        } catch (InterruptedException e) {
            System.out.println("Thread interrupted.");
        }

         System.out.println("The sum of numbers from 1 to " + N + " is: " +
MyThread.sum);
    }
}
```

OUTPUT:



```
D:\java>javac prec33.java

D:\java>java prec33
Enter the number 'N': 3
Enter the number of threads to be used (should be less than or equal to N): 2
The sum of numbers from 1 to 3 is: 6
```

CONCLUSION:

In this program, multiple threads are used to calculate the sum of numbers from 1 to N. Each thread is responsible for a portion of the range, and the

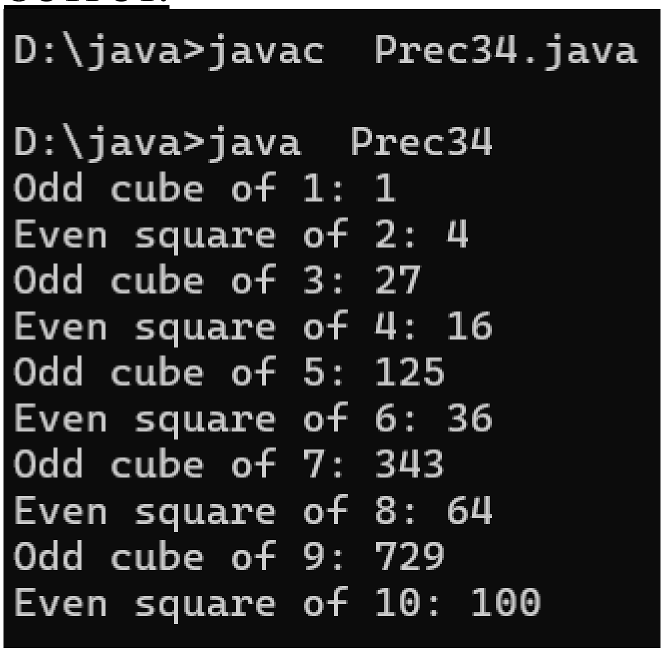
	results are combined using a synchronized method to ensure thread safety. The use of Thread.join() ensures that the main program waits for all threads to complete before displaying the final sum. This approach speeds up the summation by distributing the task across multiple threads.
34.	Write a java 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 CODE:

```
class even extends Thread {
   int n;

   public even(int n) {
      this.n = n;
   }

   public void run() {
      System.out.println(n * n);
   }
}

class odd extends Thread {
   int n;

   public odd(int n) {
      this.n = n;
   }

   public void run() {
      System.out.println(n * n * n);
   }
}

public class prec34 {
   public static void main(String[] args) {
      even e;
      odd o;
```

```
        for (int i = 1; i <= 10; i++) {
            e = new even(i);
            o = new odd(i);
            try {
                if (i % 2 == 0) {
                    e.start();
                } else {
                    o.start();
                }
                Thread.sleep(1000);
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        }
    }
}
```

OUTPUT:



```
D:\java>javac  Prec34.java

D:\java>java  Prec34
Odd cube of 1: 1
Even square of 2: 4
Odd cube of 3: 27
Even square of 4: 16
Odd cube of 5: 125
Even square of 6: 36
Odd cube of 7: 343
Even square of 8: 64
Odd cube of 9: 729
Even square of 10: 100
```

CONCLUSION:

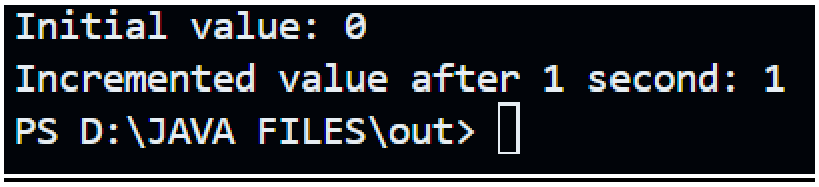
In this program, separate threads calculate the square of even numbers and the cube of odd numbers from 1 to 10. The synchronized block ensures that the output is printed without interruption, and a delay of 1 second between thread executions is added for clarity. This demonstrates multithreading for handling different tasks based on even or odd numbers.

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 CODE:

```
class IncrementThread extends Thread {
    private int number;

    public IncrementThread(int number) {
        this.number = number;
    }

    @Override
    public void run() {
        try {
            Thread.sleep(1000);
            number++;
            System.out.println("Incremented value after 1 second: " + number);
        } catch (InterruptedException e) {
            System.out.println("Thread was interrupted");
        }
    }
}

public class IncrementValue {
    public static void main(String[] args) {
        int initialValue = 0;
        System.out.println("Initial value: " + initialValue);
        IncrementThread incrementThread = new IncrementThread(initialValue);
        incrementThread.start();
    }
}
```

OUTPUT:

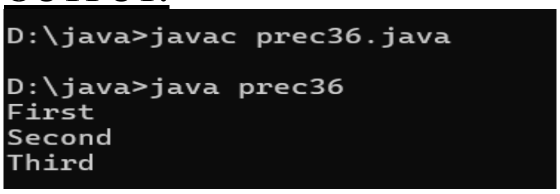


```
Initial value: 0
Incremented value after 1 second: 1
PS D:\JAVA FILES\out>
```

CONCLUSION:
This program creates a thread that increments a variable's value by one after a one-second delay using the sleep() method. The initial value is printed first, followed by the updated value. It demonstrates how multithreading and delays can be handled in Java using the Thread class and sleep() function.

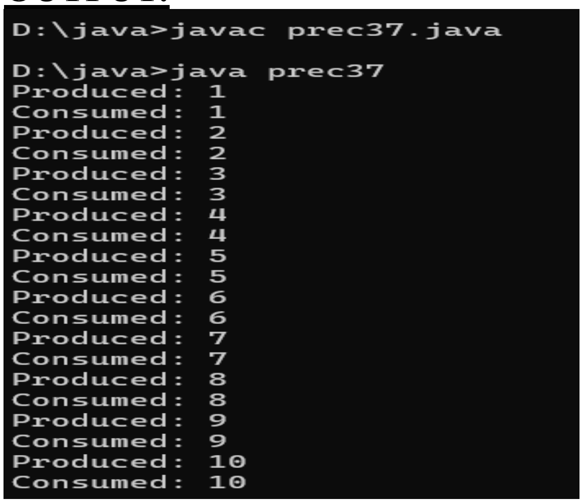
36. Write a program to create three threads ‘FIRST’,‘SECOND’, ‘THIRD’. Set the priority of the ‘FIRST’ thread to 3, the ‘SECOND’ thread to 5(default) and the
‘THIRD’ thread to 7.

PROGRAM CODE:
```
class first extends Thread {
    public void run() {
        System.out.println("First");
    }
}

class Second extends Thread {
    public void run() {
        System.out.println("Second");
    }
}

class Third extends Thread {
    public void run() {
        System.out.println("Third");
    }
}

public class prec36 {
    public static void main(String[] args) {
        first f = new first();

        Second s = new Second();
        Third t = new Third();

        f.setPriority(8);
        s.setPriority(4);
        t.setPriority(6);
```

	```
	        f.start();
	        s.start();
	        t.start();
	    }
	}
	```
	OUTPUT:
	
	```
	D:\java>javac prec36.java

	D:\java>java prec36
	First
	Second
	Third
	```
	CONCLUSION:
	In this program, three threads (first, Second, and Third) are created and assigned different priorities. The thread priorities influence the likelihood of execution but do not guarantee the order in which they will run. This demonstrates how thread priorities can be set, though actual execution order depends on the system's thread scheduling.

37.	Write a program to solve producer-consumer problem using thread synchronization.
	
	PROGRAM CODE:
	```
	class Produce extends Thread {
	    int n;
	    boolean produced = false;

	    Produce(int n) {
	        this.n = n;
	    }

	    public synchronized void run() {
	        for (int i = 1; i <= n; i++) {
	            while (produced) {  // Wait if something is already produced
	                try {
	                    wait();
	                } catch (InterruptedException e) {
	                    Thread.currentThread().interrupt();
	                }
	            }
	            System.out.println("Produced: " + i);
	```

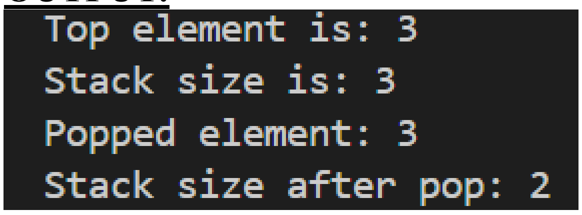
```
            produced = true;
            notify();  // Notify the consumer that production is done
        }
    }
}

class Consume extends Thread {
    int n;
    Produce producer;

    Consume(int n, Produce producer) {
        this.n = n;
        this.producer = producer;
    }
 public synchronized void run() {
        for (int i = 1; i <= n; i++) {
            synchronized (producer) {
                while (!producer.produced) {  // Wait until something is produced
                    try {
                        producer.wait();
                    } catch (InterruptedException e) {
                        Thread.currentThread().interrupt();
                    }
                }
                System.out.println("Consumed: " + i);
                producer.produced = false;
                producer.notify();  // Notify the producer to produce more
            }
        }
    }
}
public class prec37 {
    public static void main(String[] args) {
        int n = 10;
Produce p = new Produce(n);
        Consume c = new Consume(n, p);
 p.start();
        c.start();
    }
}
```

	OUTPUT: 
	```
	D:\java>javac prec37.java

	D:\java>java prec37
	Produced: 1
	Consumed: 1
	Produced: 2
	Consumed: 2
	Produced: 3
	Consumed: 3
	Produced: 4
	Consumed: 4
	Produced: 5
	Consumed: 5
	Produced: 6
	Consumed: 6
	Produced: 7
	Consumed: 7
	Produced: 8
	Consumed: 8
	Produced: 9
	Consumed: 9
	Produced: 10
	Consumed: 10
	```
	 CONCLUSION: **In this program, two threads—Produce and Consume—work together to produce and consume items sequentially. The producer thread creates items, and the consumer thread waits until an item is produced before consuming it. Synchronization and wait/notify methods are used to ensure proper coordination between the two threads, allowing them to work in an alternating fashion.**

	SET 8
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 this stack. **PROGRAM CODE:**

```
import java.util.ArrayList;

public class Per_38 {
    private ArrayList<Object> list = new ArrayList<>();

    public boolean isEmpty() {
        return list.isEmpty();
    }

    public int getSize() {
        return list.size();
    }

    public Object peek() {
        if (!isEmpty()) {
            return list.get(list.size() - 1);
        }
        return null;
    }

    public Object pop() {
        if (!isEmpty()) {
            return list.remove(list.size() - 1);
        }
        return null;
    }

    public void push(Object o) {
        list.add(o);
    }

    public static void main(String[] args) {
        MyStack stack = new MyStack();
        stack.push(1);
        stack.push(2);
        stack.push(3);

        System.out.println("Top element is: " + stack.peek());
        System.out.println("Stack size is: " + stack.getSize());
        System.out.println("Popped element: " + stack.pop());
        System.out.println("Stack size after pop: " + stack.getSize());

        System.out.println("23DCS080 Maharshi Patel");
    }
}
```

OUTPUT:



```
Top element is: 3
Stack size is: 3
Popped element: 3
Stack size after pop: 2
```

CONCLUSION:

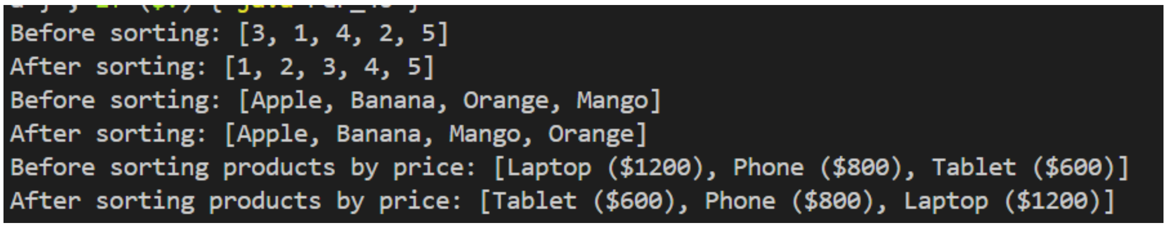
The program defines a simple stack implementation using ArrayList. It allows basic stack operations such as push(), pop(), and peek(), while also checking if the stack is empty and retrieving its size. The main method demonstrates these operations by adding and removing elements, as well as showing the top element and the size of the stack before and after popping an element.

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.Arrays;

public class Per_40 {
    public static <T extends Comparable<T>> void sort(T[] array) {
        Arrays.sort(array);
    }

    public static void main(String[] args) {
        Integer[] intArray = {3, 1, 4, 2, 5};
        System.out.println("Before sorting: " + Arrays.toString(intArray));
        sort(intArray);
        System.out.println("After sorting: " + Arrays.toString(intArray));

        String[] stringArray = {"Apple", "Banana", "Orange", "Mango"};
        System.out.println("Before sorting: " + Arrays.toString(stringArray));
        sort(stringArray);
        System.out.println("After sorting: " + Arrays.toString(stringArray));

        Product[] productArray = {
            new Product("Laptop", 1200),
            new Product("Phone", 800),
            new Product("Tablet", 600)
        };
                System.out.println("Before sorting products by price: " +
Arrays.toString(productArray));
        sort(productArray);
                System.out.println("After sorting products by price: " +
Arrays.toString(productArray));
    }
}

class Product implements Comparable<Product> {
    private String name;
    private int price;

    public Product(String name, int price) {
        this.name = name;
        this.price = price;
    }

    @Override
    public int compareTo(Product other) {
        return Integer.compare(this.price, other.price);
    }
    @Override
    public String toString() {
        return name + " ($" + price + ")";
    }
}
```

OUTPUT:



```
Before sorting: [3, 1, 4, 2, 5]
After sorting: [1, 2, 3, 4, 5]
Before sorting: [Apple, Banana, Orange, Mango]
After sorting: [Apple, Banana, Mango, Orange]
Before sorting products by price: [Laptop ($1200), Phone ($800), Tablet ($600)]
After sorting products by price: [Tablet ($600), Phone ($800), Laptop ($1200)]
```

CONCLUSION:

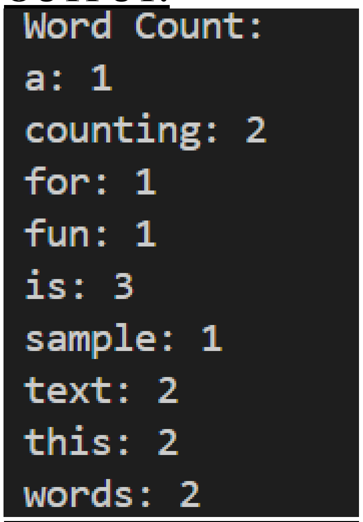
This program demonstrates a generic sort method that can sort arrays of any type that implements the Comparable interface, such as Integer, String, and custom Product objects. It sorts an integer array, a string array, and an array of Product objects by price using the Arrays.sort() method. The program effectively highlights how different types can be sorted in a consistent manner.

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 CODE:

```
import java.util.*;

public class Per_40 {

    public static void main(String[] args) {
         String text = "This is a sample text. This text is for counting words.
Counting words is fun.";
        text = text.replaceAll("[^a-zA-Z ]", "").toLowerCase();
        String[] words = text.split("\\s+");
        Map<String, Integer> wordCountMap = new HashMap<>();

        for (String word : words) {
            if (!word.isEmpty()) {
                wordCountMap.put(word, wordCountMap.getOrDefault(word, 0) +
1);
            }
        }

        Set<String> uniqueWords = new TreeSet<>(wordCountMap.keySet());
        System.out.println("Word Count:");
        for (String word : uniqueWords) {
            System.out.println(word + ": " + wordCountMap.get(word));
        }
    }
}
```

OUTPUT:

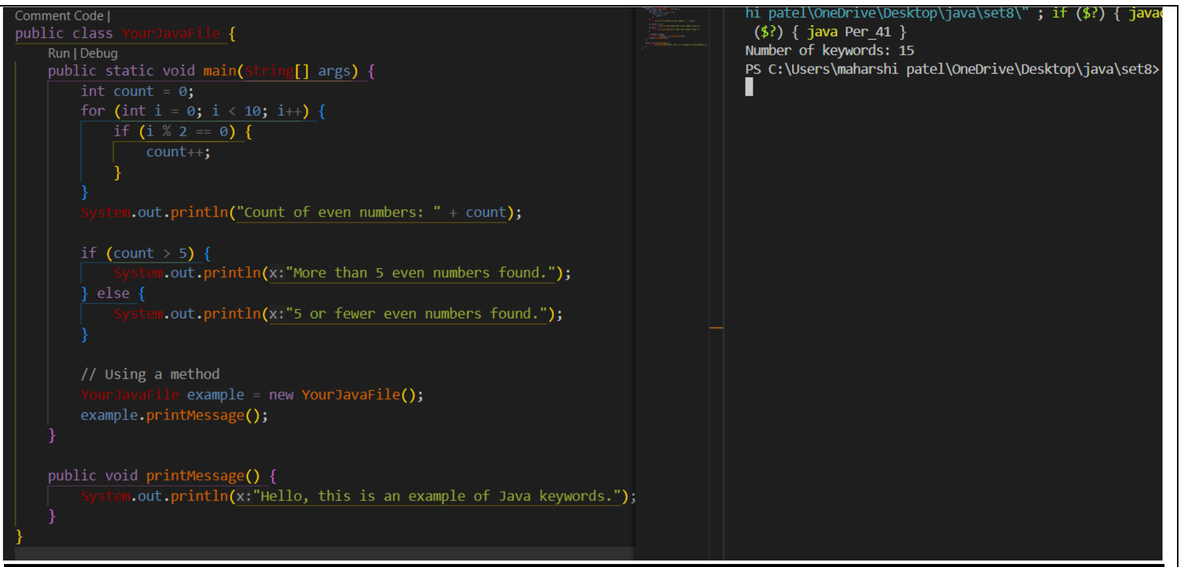


```
Word Count:
a: 1
counting: 2
for: 1
fun: 1
is: 3
sample: 1
text: 2
this: 2
words: 2
```

	CONCLUSION: This program processes a text to count the frequency of each word. It removes punctuation, converts the text to lowercase, and uses a HashMap to store the word counts. The unique words are then displayed in sorted order, along with their respective frequencies, using a TreeSet for sorted output.
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 CODE:**

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

public class Per_41 {
    public static void main(String[] args) {
        String filePath = "YourJavaFile.java";
        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);
        }

        int keywordCount = 0;
        try (BufferedReader br = new BufferedReader(new FileReader(filePath)))
{
            String line;
            while ((line = br.readLine()) != null) {
                String[] words = line.split("\\W+");
                for (String word : words) {
                    if (keywords.contains(word)) {
                        keywordCount++;
                    }
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }

        System.out.println("Number of keywords: " + keywordCount);
    }
}
```

OUTPUT:



```
Comment Code |
public class YourJavaFile {
    Run | Debug
    public static void main(String[] args) {
        int count = 0;
        for (int i = 0; i < 10; i++) {
            if (i % 2 == 0) {
                count++;
            }
        }
        System.out.println("Count of even numbers: " + count);

        if (count > 5) {
            System.out.println(x:"More than 5 even numbers found.");
        } else {
            System.out.println(x:"5 or fewer even numbers found.");
        }

        // Using a method
        YourJavaFile example = new YourJavaFile();
        example.printMessage();
    }

    public void printMessage() {
        System.out.println(x:"Hello, this is an example of Java keywords.");
    }
}
```

```
hi patel\OneDrive\Desktop\java\set8\" ; if ($?) { javac
 ($?) { java Per_41 }
Number of keywords: 15
PS C:\Users\maharshi patel\OneDrive\Desktop\java\set8>
```

CONCLUSION:

This program reads a Java source file, identifies and counts the occurrences of Java keywords using a predefined list stored in a HashSet for fast lookup. It processes the file line by line, splitting it into words, and increments the keyword count whenever a match is found. The total number of Java keywords is then displayed at the end.