



**Devang Patel Institute of
Advance Technology and Research**
(A Constitue Institute of CHARUSAT)

Certificate

This is to certify that

~~Mr./~~ Mrs. Pranchi Sanghvi

of CSE-2 Class,

ID. No. 23DCS113 has satisfactorily completed

~~his/~~ her term work in Java Programming for

the ending in Nov 2024/2025

Date : 17/10/24

Sign. of Faculty

Head of Department

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

**Subject : JAVA PROGRAMMING Semester: 3 Subject Code: CSE201 Academic Year
:2024-25**

Course Outcome (COs):

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

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

Bloom's Taxonomy:

Level 1- Remembering

Level 2- Understanding

Level 3- Applying

Level 4- Analyzing

Level 5- Evaluating

Level 6- Creating

CSE201- JAVA PROGRAMMING PRACTICAL LIST



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	<p>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.</p> <p>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.</p>	1	1, 2	2,3
5	<p>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.</p>	1	1, 2	2
6	Create a Java program that prompts the user to enter the	1	1, 2	2,3,4

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

	<p>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.</p> <p>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.</p>			
PART-II Strings				

7	<p>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'</p> <p>front_times('Chocolate', 3) → 'ChoChoCho'</p> <p>front_times('Abc', 3) → 'AbcAbcAbc'</p>	1	1, 2	2,3,4
8	<p>Given an array of ints, return the number of 9's in the array. array_count9([1, 2, 9]) → 1</p> <p>array_count9([1, 9, 9]) → 2</p> <p>array_count9([1, 9, 9, 3, 9]) → 3</p> <p>Supplementary Experiment:</p> <p>1. Write a Java program to replace each substring of a given string that matches the given regular expression with the given replacement.</p> <p>Sample string : "The quick brown fox jumps over the lazy dog."</p> <p>In the above string replace all the fox with cat.</p>	1	1, 2	2,3
9	<p>Given a string, return a string where for every char in the original, there are two chars.</p> <p>double_char('The') → 'TThhee'</p> <p>double_char('AAbb') → 'AAAAbbbb'</p> <p>double_char('Hi-There') → 'HHii--TThheerree'</p>	1	1, 2	2
10	<p>Perform following functionalities of the string:</p> <ul style="list-style-type: none"> ● 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	<p>Perform following Functionalities of the string: "CHARUSAT UNIVERSITY"</p> <ul style="list-style-type: none"> • Find length • Replace 'H' by 'FIRST LATTER OF YOUR NAME' • Convert all character in lowercase <p>Supplementary Experiment:</p> <p>1. Write a Java program to count and print all duplicates in the input string.</p> <p>Sample Output: The given string is: resource The duplicate characters and counts are: e appears 2 times r appears 2 times</p>	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

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

15	<p>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.</p> <p>Supplementary Experiment:</p> <p>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]</p>	1	1, 2	3
16	<p>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.</p>	1	1, 2	2,3
PART-IV Inheritance, Interface, Package				
17	<p>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</p>	1	1, 2, 3	3
18	<p>Create a class named 'Member' having the following members: Data members</p> <ul style="list-style-type: none"> 1 - Name 2 - Age 3 - Phone number 4 - Address 5 - Salary <p>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.</p>	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
----	--	---	-----	---



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

	<p>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.</p> <p>Supplementary Experiment:</p> <p>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]</p>			
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	<p>Write a java that implements an interface AdvancedArithmetic which contains a method signature <code>int divisor_sum(int n)</code>. You need to write a class called <code>MyCalculator</code> which implements the interface. <code>divisorSum</code> function just takes an integer as input and return the sum of all its divisors.</p> <p>For example, divisors of 6 are 1, 2, 3 and 6, so <code>divisor_sum</code> should return 12. The value of <code>n</code> will be at most 1000.</p> <p>Supplementary Experiment:</p> <p>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,</p>	2	2,3	2,3
----	--	---	-----	-----



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

	calculate interest, and view balances. SavingsAccount and CurrentAccount should implement the Account interface and have their own unique methods. [L:A]			
23	<p>Assume you want to capture shapes, which can be either circles (with a radius and 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.</p>	2	2,3	6
PART-V Exception Handling				
24	<p>Write a java program which takes two integers <code>x</code> & <code>y</code> as input, you have to compute <code>x/y</code>. If <code>x</code> and <code>y</code> are not integers or if <code>y</code> is zero, exception will occur and you have to report it.</p>	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



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

	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



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

	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: 3rd****Subject Code: CSE201****Academic year: 2024-25****Part - 1**

No.	Aim of the Practical
1.	<p>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.</p> <p><u>ANSWER:</u></p> <p>Java Installation Steps</p> <ol style="list-style-type: none">1. Download JDK: Visit Oracle's JDK download page and select the version for your OS.2. Install JDK: Run the installer and follow instructions.3. Set Environment Variables: Add JDK's bin directory to your system PATH.4. Verify Installation: Use <code>java -version</code> and <code>javac -version</code> in Command Prompt. <p>Object-Oriented Concepts</p> <ul style="list-style-type: none">- Class: Blueprint for objects.- Object: Instance of a class.- Inheritance, Polymorphism, Encapsulation, Abstraction

1. Inheritance: Inheritance is a mechanism in object-oriented programming that allows a class to inherit properties and behaviors (methods) from another class. It promotes code reuse and establishes a parent-child relationship between classes.
2. Polymorphism: Polymorphism allows objects of different classes to be treated as objects of a common superclass. It enables a single function or method to work in different ways based on the object it is acting upon.
3. Abstraction: Abstraction is the concept of hiding complex implementation details and showing only the necessary features of an object. It helps in reducing complexity by allowing the user to interact with the object at a higher level.
4. Encapsulation: Encapsulation is the practice of bundling the data (attributes) and methods (functions) that operate on the data into a single unit, or class. It restricts direct access to some of an object's components, which can prevent the accidental modification of data.

Comparison with Other Languages

- C++: Manual memory management.
- Python: Slower, dynamically typed.
- C#: Tied to Microsoft ecosystem.

Java Components

- JDK: Development kit.
- JRE: Runtime environment.
- JVM: Executes Java bytecode.
- Javadoc: Generates documentation.
- Command Line Arguments: Pass configuration info to `main(String[] args)`.

IDEs and Console Programming

- Eclipse, NetBeans, BlueJ: Popular IDEs for Java development.
- Console Programming: Compile with `javac` and run with `java`.

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:

```
class bank {  
    public static void main (String[] args) {  
  
        int a=20;  
        System.out.println("Current balance = $" + a);  
        System.out.println("23DCS113 Pranchi  
Sanghvi");  
    }  
};
```

OUTPUT:

```
C:\Users\pranc\OneDrive\Desktop\College\Java>java bank  
Current balance = $20  
23DCS113 Pranchi Sanghvi
```

CONCLUSION:

In this example, we created a basic Java program that simulates a banking application by storing a user's account balance in a variable and then displaying it. This demonstrates the fundamental concept of variable storage and output in Java, which is essential for developing more complex banking systems.

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

PROGRAM CODE:

```
import java.util.*;  
  
class pract3 {  
    public static void main(String[] args) {
```



```

Scanner sc=new Scanner (System.in);
System.out.print("Enter the distance in meters : ");
float distance = sc.nextFloat();
System.out.print("Enter the time (in hours) : ");
int hours = sc.nextInt();
System.out.print("Enter the time (in minutes) : ");
int minutes = sc.nextInt();
System.out.print("Enter the time (in seconds) : ");
int seconds = sc.nextInt();

int totalSeconds;
totalSeconds = seconds+(minutes*60)+(hours*3600);
float speed1 = distance / totalSeconds;
float speed2 = (distance / 1000) / (totalSeconds / 3600.0f);
float speed3 = (distance / 1609) / (totalSeconds / 3600.0f);

System.out.println("Speed in meters/second : " + speed1);
System.out.println("Speed in kilometers/hour : " + speed2);
System.out.println("Speed in miles/hour : " + speed3);
System.out.println("23DCS113 Pranchi Sanghvi");
}
};

```

OUTPUT:

```

C:\Users\pranc\OneDrive\Desktop\College\Java>java pract3
Enter the distance in meters : 55000
Enter the time (in hours) : 3
Enter the time (in minutes) : 5
Enter the time (in seconds) : 35
Speed in meters/second : 4.93938
Speed in kilometers/hour : 17.781769
Speed in miles/hour : 11.051442
23DCS113 Pranchi Sanghvi

```

CONCLUSION:

In this program, we used Java's Scanner class to take user input for distance in meters and time in hours, minutes, and seconds. We then converted the time to total seconds to simplify the calculations. The program calculates and displays the speed in three different units: meters per second, kilometers per hour, and miles per hour. This example demonstrates how to handle user input, perform unit conversions, and output results in Java.

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.Scanner;
```

```
class pract4 {
```

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

```
        Scanner sc = new Scanner(System.in);
```

```
        System.out.print("Enter the number of days in a month : ");
```

```
        int NumberOfDays = sc.nextInt();
```

```
        float DailyExpense[] = new float[NumberOfDays];
```

```
        for (int i = 0; i < NumberOfDays; i++) {
```

```
            System.out.print("Enter your daily expense for day " + (i + 1) + " : ");
```

```
            DailyExpense[i] = sc.nextFloat();
```

```
        }
```

```
        float TotalExpense = 0;
```

```
for (int i = 0; i < NumberOfDays; i++) {  
    TotalExpense += DailyExpense[i];  
}  
  
System.out.println("Total Expense of the month is " + TotalExpense);  
System.out.println("23DCS113 Pranchi Sanghvi");  
}  
};
```

OUTPUT:

```
C:\Users\pranc\OneDrive\Desktop\College\Java>java pract4  
Enter the number of days in a month : 30  
Enter your daily expense for day 1 : 0  
Enter your daily expense for day 2 : 0  
Enter your daily expense for day 3 : 60  
Enter your daily expense for day 4 : 120  
Enter your daily expense for day 5 : 300  
Enter your daily expense for day 6 : 0  
Enter your daily expense for day 7 : 825  
Enter your daily expense for day 8 : 70  
Enter your daily expense for day 9 : 0  
Enter your daily expense for day 10 : 10  
Enter your daily expense for day 11 : 450  
Enter your daily expense for day 12 : 0  
Enter your daily expense for day 13 : 250  
Enter your daily expense for day 14 : 0  
Enter your daily expense for day 15 : 0  
Enter your daily expense for day 16 : 0  
Enter your daily expense for day 17 : 300  
Enter your daily expense for day 18 : 0  
Enter your daily expense for day 19 : 0  
Enter your daily expense for day 20 : 100  
Enter your daily expense for day 21 : 0  
Enter your daily expense for day 22 : 0  
Enter your daily expense for day 23 : 500  
Enter your daily expense for day 24 : 0  
Enter your daily expense for day 25 : 0  
Enter your daily expense for day 26 : 60  
Enter your daily expense for day 27 : 25  
Enter your daily expense for day 28 : 0  
Enter your daily expense for day 29 : 140  
Enter your daily expense for day 30 : 0  
Total Expense of the month is 3210.0  
23DCS113 Pranchi Sanghvi
```

CONCLUSION:

This Java program provides a simple yet effective way to track and calculate monthly expenses based on daily inputs from the user. It demonstrates how to use arrays to store data, take user input through a loop, and perform a summation of array elements.

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.

PROGRAM CODE:

```
import java.util.*;

class pract5 {
    public static void main(String[] args) {
        int[] productcode = {1,2,3,4,5};
        double[] prices = {100.0,150.0,200.0,250.0,300.0};
        int i;
        double totalprice = 0.0;

        System.out.println("Motor : 100.0");
        System.out.println("Fan : 150.0");
        System.out.println("Tube : 200.0");
        System.out.println("Wire : 250.0");
        System.out.println("Others : 300.0");

        for(i=0;i<productcode.length;i++) {
            int code = productcode[i];
            double price = prices[i];
            double tax = 0.0;
```

```
switch(code) {  
    case 1:  
        tax = 0.08;  
        break;  
    case 2:  
        tax = 0.12;  
        break;  
    case 3:  
        tax = 0.05;  
        break;  
    case 4:  
        tax = 0.075;  
        break;  
    case 5:  
        tax = 0.03;  
        break;  
}  
  
double taxamount = price * tax;  
double totalamount = price + taxamount;  
totalprice += totalamount;  
}  
  
System.out.println("Total Bill is " + totalprice);  
System.out.println("23DCS113 Pranchi Sanghvi");  
}  
};
```

OUTPUT:

```
C:\Users\pranc\OneDrive\Desktop\College\Java>java pract5
Motor : 100.0
Fan : 150.0
Tube : 200.0
Wire : 250.0
Others : 300.0
Total Bill is 1063.75
23DCS113 Pranchi Sanghvi
```

CONCLUSION:

This practical exercise reinforced the fundamentals of array handling, control structures, and basic arithmetic operations in Java.

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 pract6 {
    public static void main(String[] args) {
        Scanner sc=new Scanner (System.in);
        int n,t1=0,t2=1;
        System.out.print("Enter the number of days(n) to generate your exercise routine : ");
        n=sc.nextInt();

        for(int i=1;i<=n;++i) {
            System.out.print(t1 + ",");
            int sum=t1+t2;
            t1=t2;
            t2=sum;
        }
    }
}
```

```

    }
    System.out.println("");
    System.out.println("23DCS113 Pranchi Sanghvi");
    }
};

```

OUTPUT:

```

C:\Users\pranc\OneDrive\Desktop\College\Java>java pract6
Enter the number of days(n) to generate your exercise routine : 5
0,1,1,2,3,
23DCS113 Pranchi Sanghvi

```

CONCLUSION:

This program provides an easy way to generate an exercise routine based on the Fibonacci series, which is often used to progressively increases the workout durations. By following this routine, you can ensure a gradual increase in your exercise duration, which helps you improve endurance and overall fitness. The Fibonacci series starts with 0 and 1, and each subsequent term is the sum of the previous two terms. This pattern creates a natural progressive routine, making it a useful tool for designing exercise plans.

PART – 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.lang.*;

class pract7 {

```



```
public static String front_times(String str, int n) {  
    String result = "";  
    int i;  
    if(str.length()>3) {  
        String s1 = str.substring(0,3);  
        for(i=0;i<n;i++) {  
            result = result + s1;  
        }  
    }  
    else  
    {  
        String s1 = str;  
        for(i=0;i<n;i++) {  
            result = result + s1;  
        }  
    }  
    return result;  
}
```

```
public static void main(String[] args) {  
    System.out.println(front_times("Chocolate",2));  
    System.out.println(front_times("Chocolate",3));  
    System.out.println(front_times("Abc",3));  
    System.out.println("23DCS113 Pranchi Sanghvi");  
    }  
};
```

OUTPUT:

```
C:\Users\pranc\OneDrive\Desktop\College\Java>java pract7
ChoCho
ChoChoCho
AbcAbcAbc
23DCS113 Pranchi Sanghvi
```

CONCLUSION:

This Java program prompts the user to enter two integers, n and m, then prints the first 3 characters of the string "chocolate" n times and m times consecutively. Additionally, it prints the first 3 characters of the string "Abc" m times. The program separates different outputs with dashes and concludes by displaying a footer message. The use of loops ensures repetitive printing based on user input, demonstrating basic input-output operations and string manipulation in Java.

8. Given an array of ints, return the number of 9's in the array.

array_count9([1, 2, 9]) → 1

array_count9([1, 9, 9]) → 2

array_count9([1, 9, 9, 3, 9]) → 3

PROGRAM CODE:

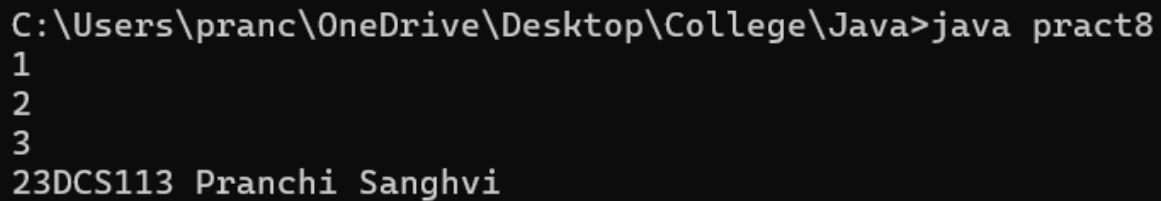
```
import java.lang.*;

class pract8 {
    public static int array_count9(int[] arr) {
        int length = arr.length;
        int i;
        int count = 0;

        for (i = 0; i < length; i++) {
            if (arr[i] == 9) {
                count++;
            }
        }
    }
}
```

```
}  
}  
return count;  
}  
  
public static void main(String[] args) {  
    System.out.println(array_count9(new int[]{1, 2, 9}));  
    System.out.println(array_count9(new int[]{1, 9, 9}));  
    System.out.println(array_count9(new int[]{1, 9, 9, 3, 9}));  
    System.out.println("23DCS113 Pranchi Sanghvi");  
}  
};
```

OUTPUT:



```
C:\Users\pranc\OneDrive\Desktop\College\Java>java pract8  
1  
2  
3  
23DCS113 Pranchi Sanghvi
```

CONCLUSION:

This Java program defines a method `array_count9` that calculates and returns the number of times the integer `9` appears in an input integer array. It then demonstrates the functionality by applying the method to three different arrays (arr1, arr2, arr3) and prints the results. The program effectively showcases basic array manipulation and function usage in Java.

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.lang.*;
```

```
class pract9 {
```

```
    public static String double_char(String str) {
```

```
        String result = "";
```

```
        int len = str.length();
```

```
        int i;
```

```
        for(i=0;i<len;i++) {
```

```
            result = result + str.charAt(i) + str.charAt(i);
```

```
        }
```

```
        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("23DCS113 Pranchi Sanghvi");
```

```
    }
```

```
};
```

OUTPUT:

```
C:\Users\pranc\OneDrive\Desktop\College\Java>java pract9
TThhee
AAAAbbbb
HHii--TThheerree
23DCS113 Pranchi Sanghvi
```

CONCLUSION:

This Java program defines a method `doubleChar` within the class `pract9`, which takes a string `str` as input and doubles each character in the string. It uses a `StringBuilder` to efficiently build the resultant string by appending each character twice in a loop. The `main` method demonstrates the functionality of `doubleChar` by creating an instance of `pract9`, calling the method with different input strings ("The", "AAbb", "Hi-There"), and printing the transformed strings.

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.Scanner;
import java.util.Arrays;

class pract10 {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the string : ");
        String str = sc.nextLine();
```

```
int length = str.length();
System.out.println("The length of the string is " + length);

String lower = str.toLowerCase();
System.out.println("String in lower case is " + lower);

String upper = str.toUpperCase();
System.out.println("String in upper case is " + upper);

String reverse = "";
for (int i = str.length() - 1; i >= 0; i--) {
    reverse = reverse + str.charAt(i);
}
System.out.println("Reverse of the string is " + reverse);

char[] charArray = str.toCharArray();
Arrays.sort(charArray);
String sortedString = new String(charArray);
System.out.println("The sorted string is " + sortedString);

System.out.println("23DCS113 Pranchi Sanghvi");
}
};
```

OUTPUT:

```
C:\Users\pranc\OneDrive\Desktop\College\Java>java pract10
Enter the string : Pranchi
The length of the string is 7
String in lower case is pranchi
String in upper case is PRANCHI
Reverse of the string is ihcnarP
The sorted string is Pachinr
23DCS113 Pranchi Sanghvi
```

CONCLUSION:

The Java program provided demonstrates several string manipulation techniques. It calculates the length of a string, converts it to lowercase and uppercase, reverses the string, and sorts its characters alphabetically. These operations showcase fundamental string handling capabilities in Java using built-in methods and loops. Additionally, the program concludes by printing a fixed message. Overall, it serves as a basic example of how to perform common string operations and utilize array manipulation functions in Java.

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

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

PROGRAM CODE:

```
import java.util.Scanner;
import java.util.Arrays;

class pract11 {
    public static void main(String args[]) {
        String str = "CHARUSAT UNIVERSITY";
        int length = str.length();
        System.out.println("The length of the string : " + length);
```



```
String replacedstr = str.replace('H','P');
System.out.println("Replaced String : " + replacedstr);

String lowerstr = str.toLowerCase();
System.out.println("The string to lower case : " + lowerstr);

System.out.println("23DCS113 Pranchi Sanghvi");
}
};
```

OUTPUT:

```
C:\Users\pranc\OneDrive\Desktop\College\Java>java pract11
The length of the string : 19
Replaced String : CPARUSAT UNIVERSITY
The string to lower case : charusat university
23DCS113 Pranchi Sanghvi
```

CONCLUSION:

The Java code performs various string manipulations on the input "CHARUSAT UNIVERSITY". It first calculates and prints the length of the string. Then, it replaces the character 'H' with 'P' and prints the modified string. Next, the code converts the entire string to lowercase and prints the result. These operations demonstrate basic string handling techniques in Java.

PART - 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.

PROGRAM CODE:

a) import java.util.Scanner;

```
class pract121 {  
    public static void main(String[] args) {  
        float pound;  
        float rupees;  
        Scanner sc = new Scanner(System.in);  
        System.out.print("Enter the amount in pound : ");  
        pound = sc.nextFloat();  
        rupees = pound * 100;  
        System.out.println("The amount in rupees : " + rupees);  
  
        System.out.println("23DCS113 Pranchi Sanghvi");  
    }  
};
```

b) import java.util.*;

```
class pract122 {  
    public static void main(String args[]) {  
        float rupees;  
        System.out.print("Enter the amount in pound : ");  
        float pound = Float.parseFloat(args[0]);  
        rupees = pound * 100;  
        System.out.println("The amount in rupees : " + rupees);  
  
        System.out.println("23DCS113 Pranchi Sanghvi");  
    }  
};
```

OUTPUT:

```
C:\Users\pranc\OneDrive\Desktop\College\Java>java pract121
Enter the amount in pound : 55
The amount in rupees : 5500.0
23DCS113 Pranchi Sanghvi
```

a) User input

CONCLUSION:

This Java program converts Pounds to Rupees using a conversion rate of 1 Pound = 100 Rupees. It accepts the amount either from command-line arguments or interactively from the user. The result is then printed to the console.

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.Scanner;

class pract13 {
    String fname;
    String lname;
    double salary;

    pract13() {
        fname = "";
        lname = "";
        salary = 0.0;
    }
}
```

```

    }

    pract13(String fname, String lname, double salary) {
        this.fname = fname;
        this.lname = lname;
        this.salary = salary;
    }

    void get() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter your First Name : ");
        fname = sc.nextLine();
        System.out.print("Enter your Last Name : ");
        lname = sc.nextLine();
        System.out.print("Enter your monthly salary : ");
        salary = sc.nextDouble();

        if (salary < 0) {
            salary = 0.0;
            System.out.println("Your salary is " + salary);
        }
        else {
            double year = salary * 12;
            System.out.println("Your yearly salary is " + year);
            double raise = salary * 0.1;
            System.out.println("Your salary raise is " + raise);
        }
    }
}

```

```

public static void main(String[] args) {
    pract13 p = new pract13();
    p.get();
    System.out.println("23DCS113 Pranchi Sanghvi");
}
};

```

OUTPUT:

```

C:\Users\pranc\OneDrive\Desktop\College\Java>java pract13
Enter your First Name : Pranchi
Enter your Last Name : Sanghvi
Enter your monthly salary : 55000
Your yearly salary is 660000.0
Your salary raise is 5500.0
23DCS113 Pranchi Sanghvi

```

CONCLUSION:

This Java program defines an Employee class with methods to calculate yearly salary, give a raise, and display employee details. In the main method, two employees are created, their details and yearly salaries are displayed, and then their salaries are increased by 10%. The updated yearly salaries are printed.

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:

```

class pract14 {
    private int month;
    private int day;

```

```
private int year;

public int getmonth()
{
    return month;
}

public int getday()
{
    return day;
}

public int getyear()
{
    return year;
}

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

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

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

```

    }

    public pract14(int day, int month, int year)
    {
        this.day = day;
        this.month = month;
        this.year = year;
    }

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

    public static void main(String[] args) {
        pract14 myDate = new pract14(1, 4, 2024);
        myDate.display();
        System.out.println("23DCS113 Pranchi Sanghvi");
    }
};

```

OUTPUT:

```

C:\Users\pranc\OneDrive\Desktop\College\Java>java pract14
1/4/2024
23DCS113 Pranchi Sanghvi

```

CONCLUSION:

It defines a Date class with private fields for day, month, and year. There's also a displayDate() method that prints the date in the format "day/month/year".

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 {

    private int length;
    private int breadth;

    Area(int l, int b) {
        length = l;
        breadth = b;
    }

    int returnArea() {
        return length * breadth;
    }
}

public class pract15 {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the length of the rectangle : ");
        int length = scanner.nextInt();

        System.out.print("Enter the breadth of the rectangle : ");
        int breadth = scanner.nextInt();

        Area rectangle = new Area(length, breadth);
```

```
System.out.println("Area of the rectangle : " + rectangle.returnArea());  
System.out.println("23DCS113 Pranchi Sanghvi");  
}  
};
```

OUTPUT:

```
C:\Users\pranc\OneDrive\Desktop\College\Java>java pract15  
Enter the length of the rectangle : 55  
Enter the breadth of the rectangle : 40  
Area of the rectangle : 2200  
23DCS113 Pranchi Sanghvi
```

CONCLUSION:

The code defines an Area class with private fields for length and breadth. It calculates the area of a rectangle using the formula length * breadth. The user is prompted to input the length and breadth of the rectangle. The program then creates an instance of the Area class and computes the 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:

```
class pract16 {  
    private double real;  
    private double img;  
  
    public pract16() {  
        real = 0;  
        img = 0;  
    }  
}
```

```
public pract16(double real, double img) {  
    this.real = real;  
    this.img = img;  
}  
  
public pract16(pract16 c1) {  
    this.real = c1.real;  
    this.img = c1.img;  
}  
  
public double getReal() {  
    return real;  
}  
  
public double getImg() {  
    return img;  
}  
  
public void setReal(double real) {  
    this.real = real;  
}  
  
public void setImg(double img) {  
    this.img = img;  
}  
  
public void display() {  
    System.out.println(real + " + " + img + "i");  
}
```

```

public pract16 add(pract16 other) {
    return new pract16(this.real + other.real, this.img + other.img);
}

public pract16 subtract(pract16 other) {
    return new pract16(this.real - other.real, this.img - other.img);
}

public pract16 multiply(pract16 other) {
    double realPart = this.real * other.real - this.img * other.img;
    double imgPart = this.real * other.img + this.img * other.real;
    return new pract16(realPart, imgPart);
}

public pract16 divide(pract16 other) {
    double denominator = other.real * other.real + other.img * other.img;
    double realPart = (this.real * other.real + this.img * other.img) / denominator;
    double imgPart = (this.img * other.real - this.real * other.img) / denominator;
    return new pract16(realPart, imgPart);
}

public static void main(String[] args) {
    pract16 c1 = new pract16(3.0, 4.0);
    System.out.print("c1 : ");
    c1.display();

    pract16 c2 = new pract16(c1);
    System.out.print("c2 : ");
    c2.display();
}

```

```
pract16 c3 = new pract16(1.0, 2.0);  
System.out.print("c3 : ");  
c3.display();  
  
pract16 sum = c1.add(c3);  
System.out.print("Sum : ");  
sum.display();  
  
pract16 difference = c1.subtract(c3);  
System.out.print("Difference : ");  
difference.display();  
  
pract16 product = c1.multiply(c3);  
System.out.print("Product : ");  
product.display();  
  
pract16 quotient = c1.divide(c3);  
System.out.print("Quotient : ");  
quotient.display();  
System.out.println("23DCS113 Pranchi Sanghvi");  
}  
};
```

OUTPUT:

```
C:\Users\pranc\OneDrive\Desktop\College\Java>java pract16
c1 : 3.0 + 4.0i
c2 : 3.0 + 4.0i
c3 : 1.0 + 2.0i
Sum : 4.0 + 6.0i
Difference : 2.0 + 2.0i
Product : -5.0 + 10.0i
Quotient : 2.2 + -0.4i
23DCS113 Pranchi Sanghvi
```

CONCLUSION:

The program defines a Complex class that represents complex numbers. It has methods for calculating the sum, difference, and product of two complex numbers. The user is prompted to input the real and imaginary parts of two complex numbers. The program creates instances of the Complex class for both input numbers and computes the requested operations.

PART - 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:

```
import java.util.*;

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 pract17 {
    public static void main(String[] args) {
        Parent parentObj = new Parent();
        parentObj.displayParent();
        Child childObj = new Child();
        childObj.displayChild();
        childObj.displayParent();
        System.out.println("23DCS113 Pranchi Sanghvi");
    }
}

```

OUTPUT:

```

C:\Users\PRANCHI\OneDrive\Desktop\College\Java>java pract17
This is parent class
This is child class
This is parent class
23DCS113 Pranchi Sanghvi

```

CONCLUSION:

This code demonstrates basic inheritance in Java, where the ChildClass inherits from the ParentClass. The ChildClass can call both its own method printChild() and the inherited method printParent() from the ParentClass. The code outputs messages from both classes and showcases polymorphism by calling methods of both parent and child classes.

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:

```
import java.util.*;

public class pract18 {
    public static void main(String[] args) {
        Employee E = new Employee();
        E.name = "Pranchi Sanghvi";
        E.age = 19;
        E.mobile_number = 1551558625;
        E.Address = "Rajkot,India";
        E.Salary = 50000;
        E.Specialization = "Software Development";

        Manager M = new Manager();
        M.name = "Disha Patel";
        M.age = 18;
        M.mobile_number = 1551558625;
        M.Address = "Nadiad,India";
        M.Salary = 50000;
        M.Department = "Software Development";

        System.out.println("_____");
        E.displayEmployeeDetails();
        System.out.println("_____");
        M.displayManagerDetails();
    }
}
```



```
}
```

```
class Member {
```

```
    String name;
```

```
    int age;
```

```
    int mobile_number;
```

```
    String Address;
```

```
    double Salary;
```

```
    void printSalary() {
```

```
        System.out.println("Salary: " + Salary);
```

```
    }
```

```
}
```

```
class Employee extends Member {
```

```
    String Specialization;
```

```
    void displayEmployeeDetails() {
```

```
        System.out.println("Employee Details:");
```

```
        System.out.println("Name: " + name);
```

```
        System.out.println("Age: " + age);
```

```
        System.out.println("Phone Number: " + mobile_number);
```

```
        System.out.println("Address: " + Address);
```

```
        System.out.println("Specialization: " + Specialization);
```

```
        printSalary();
```

```
    }
```

```
}
```

```
class Manager extends Member {
```

```
    String Department;
```

```
    void displayManagerDetails() {
```

```
        System.out.println("Manager Details:");
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("Phone Number: " + mobile_number);
        System.out.println("Address: " + Address);
        System.out.println("Specialization: " + Department);
        printSalary();
    }
}
```

OUTPUT:

```
C:\Users\PRANCHI\OneDrive\Desktop\College\Java>java pract18
-----
Employee Details:
Name: Pranchi Sanghvi
Age: 19
Phone Number: 1551558625
Address: Rajkot,India
Specialization: Software Development
Salary: 50000.0
-----
Manager Details:
Name: Disha Patel
Age: 18
Phone Number: 1551558625
Address: Nadiad,India
Specialization: Software Development
Salary: 50000.0
```

CONCLUSION:

This code demonstrates inheritance in Java where both Employee and Manager classes inherit common attributes and methods from the Member class. It captures and displays details specific to both employees and managers, including specialization for employees and department for managers. The use of inheritance allows for shared functionality, such as printing salaries, while enabling unique attributes for each subclass.

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:

```
public class pract19 {  
    public static void main(String[] args) {  
        Rectangle[] rectangles = new Rectangle[2];  
        rectangles[0] = new Rectangle(4, 11);  
        rectangles[1] = new Square(6);  
  
        for (Rectangle shape : rectangles) {  
            System.out.println("Area : " + shape.area());  
            System.out.println("Perimeter : " + shape.perimeter());  
        }  
  
        System.out.println("23DCS113 Pranchi Sanghvi");  
    }  
}  
  
class Rectangle {  
    int length;  
    int breadth;  
  
    Rectangle(int l, int b) {  
        length = l;  
        breadth = b;  
    }  
    int area() {  
        return length * breadth;  
    }  
  
    int perimeter() {  
        return 2 * (length + breadth);  
    }  
}
```

```

    }
}

class Square extends Rectangle {
    Square(int s) {
        super(s, s);
    }
}

```

OUTPUT:

```

C:\Users\PRANCHI\OneDrive\Desktop\College\Java>java pract19
Area : 44
Perimeter : 30
Area : 36
Perimeter : 24
23DCS113 Pranchi Sanghvi

```

CONCLUSION:

This code gives us the concept of inheritance and polymorphism in Java. The Square class inherits from the Rectangle class, as a square is a special case of a rectangle where the length and breadth are equal. The array of Rectangle objects includes both rectangles and squares, and through polymorphism, the program calculates and displays the area and perimeter for each shape using their respective implementations.

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 void printShape() {
        System.out.println("This is a shape.");
    }
}

class Rectangle extends Shape {

```

```

    public void printRectangle() {
        System.out.println("This is a rectangular shape.");
    }
}

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

class Square extends Rectangle {
    public void printSquare() {
        System.out.println("Square is a rectangle.");
    }
}

public class pract20 {
    public static void main(String[] args) {
        Square mySquare = new Square();
        mySquare.printShape();
        mySquare.printRectangle();
        System.out.println("23DCS113 Pranchi Sanghvi");
    }
}

```

OUTPUT:

```

C:\Users\PRANCHI\OneDrive\Desktop\College\Java>java pract20
This is a shape.
This is a rectangular shape.
23DCS113 Pranchi Sanghvi

```

CONCLUSION:

This code demonstrates multi-level inheritance in Java. The Square class extends Rectangle, which in turn extends the base class Shape. The Square class inherits methods from both Shape and Rectangle, and it has its own method print4(). The code showcases how different classes represent shapes, with Square being treated as a specific type of rectangle.

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 {  
    public void getDegree() {  
        System.out.println("I got a degree.");  
    }  
}  
  
class Undergraduate extends Degree {  
  
    public void getDegree() {  
        System.out.println("I am an Undergraduate.");  
    }  
}  
  
class Postgraduate extends Degree {  
  
    public void getDegree() {  
        System.out.println("I am a Postgraduate.");  
    }  
}  
  
public class pract21 {  
    public static void main(String[] args) {  
        Degree degree = new Degree();  
        Undergraduate undergrad = new Undergraduate();
```

```

Postgraduate postgrad = new Postgraduate();

degree.getDegree();
undergrad.getDegree();
postgrad.getDegree();

System.out.println("23DCS113 Pranchi Sanghvi");
}
}

```

OUTPUT:

```

C:\Users\PRANCHI\OneDrive\Desktop\College\Java>java pract21
I got a degree.
I am an Undergraduate.
I am a Postgraduate.
23DCS113 Pranchi Sanghvi

```

CONCLUSION:

This code demonstrates method overriding in Java, where the Undergraduate and Postgraduate classes override the getDegree() method of the Degree class to provide specific outputs. Each class prints a message related to the type of degree it represents. This showcases how subclasses can customize inherited behavior by overriding methods from a parent class.

22. Write a java that implements an interface AdvancedArithmetic which contains a method signature `int divisor_sum(int n)`. You need to write a class called MyCalculator which implements the interface. `divisor_sum` 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 atmost 1000.

PROGRAM CODE:

```

interface AdvancedArithmetic {
    int divisor_sum(int n);
}

```

```
class MyCalculator implements AdvancedArithmetic {  
    public int divisor_sum(int n) {  
        int sum = 0;  
        for (int i = 1; i <= n; i++) {  
            if (n % i == 0) {  
                sum += i;  
            }  
        }  
        return sum;  
    }  
}  
  
public class pract22 {  
    public static void main(String[] args) {  
        MyCalculator my_calculator = new MyCalculator();  
        System.out.print("I implemented : ");  
        int n = 6;  
        System.out.println(my_calculator.divisor_sum(n));  
        System.out.println("23DCS113 Pranchi Sanghvi");  
    }  
}
```

OUTPUT:

```
C:\Users\PRANCHI\OneDrive\Desktop\College\Java>java pract22  
I implemented : 12  
23DCS113 Pranchi Sanghvi
```


CONCLUSION:

This code demonstrates the implementation of an interface in Java. The AdvanceArithmetic interface defines a method divisor_sum(int n), which is implemented by the MyCalculator class. The program calculates and returns the sum of all divisors of a given number, showcasing how interfaces can be used to define a contract for classes to implement specific functionalities.

23. Assume you want to capture shapes, which can be either circles (with a radius and 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:

```
import java.util.*;

interface Shape {
    double area();

    default void displayDetails() {
        System.out.println("This is a shape.");
    }
}

class Circle implements Shape {
    private double radius;
    private String color;

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

```

        this.color = color;
    }

    public double area() {
        return Math.PI * radius * radius;
    }

    public void displayDetails() {
        System.out.println("This is a circle with radius " + radius + " and color " + color);
    }
}

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;
    }

    public double area() {
        return length * width;
    }

    public void displayDetails() {
        System.out.println("This is a rectangle with length " + length + ", width " + width +
            ", and color " + color);
    }
}

```

```

    }
}

class Sign {
    private Shape shape;
    private String text;

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

    public void displaySign() {
        shape.displayDetails();
        System.out.println("Text : " + text);
    }
}

public class pract23 {
    public static void main(String[] args) {

        Circle circle = new Circle(5.0, "Red");
        Rectangle rectangle = new Rectangle(4.0, 6.0, "Blue");

        Sign circleSign = new Sign(circle, "Hello, World!");
        Sign rectangleSign = new Sign(rectangle, "Welcome to the Campus Center!");

        circleSign.displaySign();
        rectangleSign.displaySign();
    }
}

```

```

        System.out.println("23DCS113 Pranchi Sanghvi");
    }
}

```

OUTPUT:

```

C:\Users\PRANCHI\OneDrive\Desktop\College\Java>java pract23
This is a circle with radius 5.0 and color Red
Text : Hello, World!
This is a rectangle with length 4.0, width 6.0, and color Blue
Text : Welcome to the Campus Center!
23DCS113 Pranchi Sanghvi

```

CONCLUSION:

This code demonstrates the use of interfaces, composition, and default methods in Java. The shape interface defines the methods color() and area(), and a default method info(), which are implemented by the Circle and Rectangle classes. The sign class uses composition to associate a shape with text, and its display() method prints both the sign's text and shape information.

PART-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.Scanner;

public class pract24 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        try {
            System.out.print("Enter the value of X : ");
            int x = sc.nextInt();

```

```

        System.out.print("Enter the value of Y : ");
        int y = sc.nextInt();
        int result = x / y;
        System.out.println("Result : " + result);
    } catch (Exception e) {
        if (e instanceof ArithmeticException) {
            System.out.println("Error : Divison by 0 is not possible.");
        } else {
            System.out.println("Please Enter valid integer.");
        }
    }
    System.out.println("23DCS113 Pranchi Sanghvi");
}
}

```

OUTPUT:

```

C:\Users\PRANCHI\OneDrive\Desktop\College\Java>java pract24
Enter the value of X : 5
Enter the value of Y : 4
Result : 1
23DCS113 Pranchi Sanghvi

```

```

C:\Users\PRANCHI\OneDrive\Desktop\College\Java>java pract24
Enter the value of X : 3
Enter the value of Y : 0
Error : Divison by 0 is not possible.
23DCS113 Pranchi Sanghvi

```

CONCLUSION:

In conclusion, this Java program effectively handles the division of two integers while managing potential exceptions such as invalid input types and division by zero. By utilizing `InputMismatchException` and `ArithmeticException`, the program ensures that errors are caught and reported, preventing crashes and providing informative feedback to the user. This approach enhances the program's robustness and ensures smooth execution even in the face of unexpected input.

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

PROGRAM CODE:

```
public class pract25 {  
    public static void main(String[] args) {  
        try {  
            int[] numbers = { 1, 2, 3, 4, 5 };  
  
            System.out.println(numbers[5]);  
  
        } catch (ArrayIndexOutOfBoundsException e) {  
            System.out.println("Error: Array index out of bounds!");  
            System.out.println("Exception Message: " + e.getMessage());  
        }  
        System.out.println("23DCS113 Pranchi Sanghvi");  
    }  
}
```

OUTPUT:

```
C:\Users\PRANCHI\OneDrive\Desktop\College\Java>java pract25  
Error: Array index out of bounds!  
Exception Message: Index 5 out of bounds for length 5  
23DCS113 Pranchi Sanghvi
```

CONCLUSION:

This program demonstrates how to manually throw and handle exceptions in Java. By using a try-catch block, the program ensures that thrown exceptions are caught, preventing abnormal termination. This structured error handling mechanism allows developers to manage unexpected scenarios effectively, ensuring that the program continues to operate smoothly and provides meaningful error messages to users.

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 MyException extends Exception {
    public MyException() {
        System.out.println("Exception created by user");
    }
}

public class pract26 {

    static void checkValue(int value) throws MyException {
        if (value > 10) {
            throw new MyException();
        }
        System.out.println("Value acceptable : " + value);
    }

    public static void main(String[] args) {
        try {
            checkValue(5);
            checkValue(15);

        } catch (MyException e) {
            System.out.println("Caught exception : This is a checked exception.");
        }
        System.out.println("23DCS113 Pranchi Sanghvi");
    }
}
```

OUTPUT:

```
C:\Users\PRANCHI\OneDrive\Desktop\College\Java>java pract26
Value acceptable : 5
Exception created by user
Caught exception : This is a checked exception.
23DCS113 Pranchi Sanghvi
```

CONCLUSION:

This program highlights the difference between checked and unchecked exceptions in Java. Checked exceptions must be declared or handled at compile-time, ensuring potential errors are addressed, while unchecked exceptions occur at runtime and don't need to be explicitly handled, often arising from logical or programming errors. 40

PART-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 Thread1 extends Thread{ // by extending Thread class
public void run(){
System.out.println("Hello world");
}
}
class Thread2 implements Runnable{ //by using Runnable interface.
public void run(){
System.out.println("Hello world 1");
}
}
public class P32 {
public static void main(String[] args) {
Thread1 thread = new Thread1();
thread.start();
Thread2 obj2 = new Thread2();
Thread t1 = new Thread(obj2);
t1.start();
}
}
```


OUTPUT:

```
Hello world
Hello world 1
```

CONCLUSION:

This program demonstrates two approaches to creating threads in Java: extending the Thread class and implementing the Runnable interface. Both methods effectively print "Hello World," showcasing the flexibility of Java's concurrency model.

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 SumTask implements Runnable {
    private int start;
    private int end;
    private static int totalSum = 0;
    public SumTask(int start, int end) {
        this.start = start;
        this.end = end;
    }
    public void run() {
        int partialSum = 0;
        for (int i = start; i <= end; i++) {
            partialSum += i;
        }
        synchronized (SumTask.class) {
            totalSum += partialSum;
        }
    }
    public static int getTotalSum() {
        return totalSum;
    }
}
```

```
public class P33 {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
        System.out.print("Enter N: ");  
        int N = scanner.nextInt();  
        System.out.print("Enter number of threads: ");  
        int numThreads = scanner.nextInt();  
        Thread[] threads = new Thread[numThreads];  
        int range = N / numThreads;  
        int remainder = N % numThreads;  
        int start = 1;  
        for (int i = 0; i < numThreads; i++) {  
            int end = start + range - 1;  
            if (i == numThreads - 1) {  
                end += remainder;  
            }  
            threads[i] = new Thread(new SumTask(start, end));  
            threads[i].start();  
            start = end + 1;  
        }  
        for (Thread thread : threads) {  
            try {  
                thread.join();  
            } catch (InterruptedException e) {  
                e.printStackTrace();  
            }  
        }  
        System.out.println("Total Sum: " + SumTask.getTotalSum());  
    }  
}
```

OUTPUT:

```
Enter N: 100  
Enter number of threads: 5  
Total Sum: 5050
```

CONCLUSION:

This program effectively demonstrates how to utilize multiple threads in Java to perform a summation task concurrently. By distributing the workload among threads, it showcases improved efficiency in computation, making it a practical example of multithreading in action.

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:

```
import java.util.Random;
class RandomNumberGenerator extends Thread {
private final Object lock;
public RandomNumberGenerator(Object lock) {
this.lock = lock;
}
public void run() {
Random random = new Random();
while (true) {
int number = random.nextInt(100);
synchronized (lock) {
P34.lastNumber = number;
lock.notifyAll();
System.out.println("Generated: " + number);
try {
Thread.sleep(1000);
} catch (InterruptedException e) {
e.printStackTrace();
} } } } }
class EvenNumberProcessor extends Thread {
private final Object lock;
public EvenNumberProcessor(Object lock) {
this.lock = lock;
```

```

}
public void run() {
while (true) {
synchronized (lock) {
try {
lock.wait();
} catch (InterruptedException e) {
e.printStackTrace();
}
if (P34.lastNumber % 2 == 0) {
int square = P34.lastNumber * P34.lastNumber;
System.out.println("Square: " + square);
} } } } }
class OddNumberProcessor extends Thread {
private final Object lock;
public OddNumberProcessor(Object lock) {
this.lock = lock;
}
public void run() {
while (true) {
synchronized (lock) {
try {
lock.wait();
} catch (InterruptedException e) {
e.printStackTrace();
}
if (P34.lastNumber % 2 != 0) {
int cube = P34.lastNumber * P34.lastNumber * P34.lastNumber;
System.out.println("Cube: " + cube);
} } } } }
public class P34 {
public static int lastNumber;
public static void main(String[] args) {
Object lock = new Object();
RandomNumberGenerator generator = new RandomNumberGenerator(lock);
EvenNumberProcessor evenProcessor = new EvenNumberProcessor(lock);
OddNumberProcessor oddProcessor = new OddNumberProcessor(lock);

```

```
generator.start();
evenProcessor.start();
oddProcessor.start();
}}
```

OUTPUT:

```
Generated: 43
Cube: 79507
Generated: 85
Cube: 614125
Generated: 8
Square: 64
Generated: 93
Cube: 804357
Generated: 11
Cube: 1331
Generated: 63
Cube: 250047
Generated: 80
Square: 6400
```

CONCLUSION:

This program effectively demonstrates a multi-threaded application where one thread generates random integers, while two other threads process these integers based on their parity. It highlights the use of synchronization in Java to safely share data among threads, showcasing how concurrency can be leveraged for efficient task distribution.

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:

```
public class P35 extends Thread {
private int value = 0;
public void run() {
while (true) {
value++;
System.out.println("Value: " + value);
try {
Thread.sleep(1000);
} catch (InterruptedException e) {
e.printStackTrace();
}
```

```

    } } }
    public static void main(String[] args) {
    P35 incrementer = new P35();
    incrementer.start();
    } }

```

OUTPUT:

```

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

```

CONCLUSION:

This program effectively demonstrates the use of a thread to increment a variable every second. It utilizes the sleep() method to create a delay between increments, showcasing basic thread functionality in Java.

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 MyThread extends Thread {
    MyThread(String name) {
        super(name);
    }
    public void run() {
        System.out.println(getName() + " is running with priority " + getPriority());
    } }
public class P36 {
    public static void main(String[] args) {
        MyThread first = new MyThread("FIRST");
        MyThread second = new MyThread("SECOND");

```

```

MyThread third = new MyThread("THIRD");
first.setPriority(3);
first.start();
second.setPriority(Thread.NORM_PRIORITY);
second.start();
third.setPriority(7);
third.start();
} }

```

OUTPUT:

```

FIRST is running with priority 3
THIRD is running with priority 7
SECOND is running with priority 5

```

CONCLUSION:

This program demonstrates thread creation and priority setting in Java by extending the Thread class. Each thread prints its name and priority when executed. Different priority levels (3, 5, 7) are set using setPriority(), showcasing the influence of priority on execution order. However, actual execution may vary due to the system's thread scheduling.

37.

Write a program to solve producer-consumer problem using thread synchronization.

PROGRAM CODE:

```

class Buffer {
private int data;
private boolean isEmpty = true;
public synchronized void produce(int value) {
while (!isEmpty) {
try {
wait();
} catch (InterruptedException e) {
e.printStackTrace();
} }
data = value;
isEmpty = false;
}
}

```

```

System.out.println("Produced: " + data);
notify();
}
public synchronized void consume() {
while (isEmpty) {
try {
wait();
} catch (InterruptedException e) {
e.printStackTrace();
} }
System.out.println("Consumed: " + data);
isEmpty = true;
notify();
} }
class Producer extends Thread {
private Buffer buffer;
public Producer(Buffer buffer) {
this.buffer = buffer;
}
public void run() {
for (int i = 1; i <= 5; i++) {
buffer.produce(i); // Produce values from 1 to 5
try {
Thread.sleep(1000);
} catch (InterruptedException e) {
e.printStackTrace();
} } } }
class Consumer extends Thread {
private Buffer buffer;
public Consumer(Buffer buffer) {
this.buffer = buffer;
}
public void run() {
for (int i = 1; i <= 5; i++) {
buffer.consume();
try {
Thread.sleep(1500);

```



```
} catch (InterruptedException e) {  
e.printStackTrace();  
} } } }  
public class P37 {  
public static void main(String[] args) {  
Buffer buffer = new Buffer();  
Producer producer = new Producer(buffer);  
Consumer consumer = new Consumer(buffer);  
producer.start();  
consumer.start();  
} }
```

OUTPUT:

```
Produced: 1  
Consumed: 1  
Produced: 2  
Consumed: 2  
Produced: 3  
Consumed: 3  
Produced: 4  
Consumed: 4  
Produced: 5  
Consumed: 5
```

CONCLUSION:

This program demonstrates producer-consumer synchronization in Java using the wait() and notify() methods. The producer thread generates data, while the consumer thread consumes it, both synchronized to avoid race conditions. The use of wait() and notify() ensures proper coordination between the threads, allowing for controlled data production and consumption.

PART-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;
class MyStack {
private ArrayList<Object> list = new ArrayList<>();
public boolean isEmpty() {
return list.isEmpty();
}
public int getSize() {
return list.size();
}
public Object peek() {
if (isEmpty()) {
return "Stack is empty";
}
return list.get(list.size() - 1);
}
public Object pop() {
if (isEmpty()) {
return "Stack is empty";
}
return list.remove(list.size() - 1);
}
public void push(Object o) {
list.add(o);
} }
public class P38 {
```

```

public static void main(String[] args) {
    MyStack stack = new MyStack();
    stack.push(10);
    stack.push(20);
    stack.push(30);
    System.out.println("Top element is: " + stack.peek());
    System.out.println("Popped element: " + stack.pop());
    System.out.println("Popped element: " + stack.pop());
    System.out.println("Is stack empty ? " + stack.isEmpty());
    System.out.println("Current stack size: " + stack.getSize());
    System.out.println("Top element now: " + stack.peek());
} }

```

OUTPUT:

```

Top element is: 30
Popped element: 30
Popped element: 20
Is stack empty ? false
Current stack size: 1
Top element now: 10

```

CONCLUSION:

This program demonstrates the implementation of a custom stack using the ArrayList class in Java. It provides functionalities to push, pop, peek, check if the stack is empty, and get the current size of the stack. The program effectively showcases how to manage a dynamic collection of elements while adhering to stack principles.

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 P39 {
    public static <T extends Comparable<T>> void sortArray(T[] array) {
        Arrays.sort(array);
    }
    public static void main(String[] args) {
        Integer[] numbers = {5, 3, 9, 1, 7};
        System.out.println("Before sorting (Integers): " + Arrays.toString(numbers));
        sortArray(numbers);
        System.out.println("After sorting (Integers): " + Arrays.toString(numbers));
        String[] names = {"John", "Alice", "Bob", "David"};
        System.out.println("\nBefore sorting (Strings): " + Arrays.toString(names));
        sortArray(names);
        System.out.println("After sorting (Strings): " + Arrays.toString(names));
        Product[] products = {
            new Product("Laptop", 1000),
            new Product("Phone", 800),
            new Product("Tablet", 600),
            new Product("Smartwatch", 200)
        };
        System.out.println("\nBefore sorting (Products by price): ");
        for (Product p : products) {
            System.out.println(p);
        }
        sortArray(products);
        System.out.println("\nAfter sorting (Products by price): ");
        for (Product p : products) {
            System.out.println(p);
        } }
    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 this.price - other.price;
}
@Override
public String toString() {
return name + ": $" + price;
} }
```

OUTPUT:

```
Before sorting (Integers): [5, 3, 9, 1, 7]
After sorting (Integers): [1, 3, 5, 7, 9]

Before sorting (Strings): [John, Alice, Bob, David]
After sorting (Strings): [Alice, Bob, David, John]

Before sorting (Products by price):
Laptop: $1000
Phone: $800
Tablet: $600
Smartwatch: $200

After sorting (Products by price):
Smartwatch: $200
Tablet: $600
Phone: $800
Laptop: $1000
```

CONCLUSION:

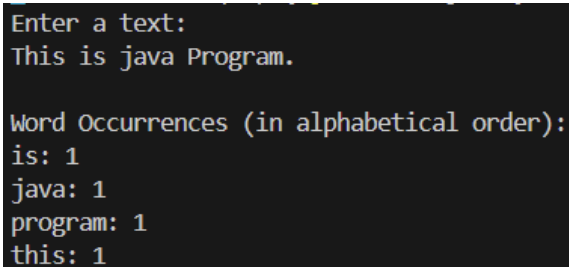
This program demonstrates the use of generics in Java to create a versatile sorting method for arrays of different types. By implementing the Comparable interface in the Product class, it enables sorting of custom objects based on specific criteria, such as price. The output shows the effective sorting of integers, strings, and products, highlighting the flexibility and reusability of the generic sorting method.

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 P40 {
    public static void main(String[] args) {
        Map<String, Integer> wordMap = new TreeMap<>();
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a text:");
        String text = scanner.nextLine();
        String[] words = text.toLowerCase().split("\\W+");
        for (String word : words) {
            if (!word.isEmpty()) {
                wordMap.put(word, wordMap.getOrDefault(word, 0) + 1);
            }
        }
        System.out.println("\nWord Occurrences (in alphabetical order):");
        Set<Map.Entry<String, Integer>> entrySet = wordMap.entrySet();
        for (Map.Entry<String, Integer> entry : entrySet) {
            System.out.println(entry.getKey() + ": " + entry.getValue());
        }
    }
}
```

OUTPUT:A screenshot of a terminal window showing the program's output. The user enters the text "This is java Program.". The program then displays the word occurrences in alphabetical order: "is: 1", "java: 1", "program: 1", and "this: 1".

```
Enter a text:
This is java Program.

Word Occurrences (in alphabetical order):
is: 1
java: 1
program: 1
this: 1
```

CONCLUSION:

This program demonstrates how to count and display the occurrences of words in a given text using Java's Map and Set classes. The words are stored in a TreeMap, ensuring that they are presented in alphabetical order. The use of getOrDefault() simplifies the counting process, showcasing efficient word frequency analysis.

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.*;
import java.util.*;
public class P41 {
private static final HashSet<String> keywords = new HashSet<>();
static {
String[] keywordArray = {
"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", "package", "private", "protected", "public",
"return", "short", "static", "strictfp", "super", "switch", "synchronized", "this",
"throw", "throws", "transient", "try", "void", "volatile", "while"
};
for (String keyword : keywordArray) {
keywords.add(keyword);
} }
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
System.out.print("Enter the path of the Java source file: ");
String filePath = scanner.nextLine();
try {
File file = new File(filePath);
Scanner fileScanner = new Scanner(file);
int keywordCount = 0;
while (fileScanner.hasNext()) {
String word = fileScanner.next();
if (keywords.contains(word)) {
keywordCount++;
} }
System.out.println("Number of Java keywords in the file: " + keywordCount);
fileScanner.close();
} catch (FileNotFoundException e) {
```

```
System.out.println("File not found: " + filePath);  
} } }
```

OUTPUT:

```
Enter the path of the Java source file: P40.java  
Number of Java keywords in the file: 11
```

CONCLUSION:

This program demonstrates the use of a HashSet to efficiently count Java keywords in a source file. By reading each word from the file and checking for its presence in the set of keywords, it showcases how to utilize collections for rapid lookups. The result is the total number of keywords, providing a simple yet effective tool for analyzing Java code.