



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

# Certificate

*This is to certify that*

*Mr./Mrs.* Patel Veer K

*of* Depstar CSE - 2 *Class,*

*ID. No.* 23 DCS 097 *has satisfactorily completed*

*his/ her term work in* Java Programming - CSE 201 *for*

*the ending in* November 2024 / 2025

*Date :* 16/10/24

*Amaravat*

*Sign. of Faculty*

*G*

*Head of Department*

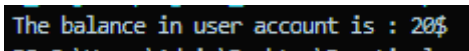
**CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY****DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY &  
RESEARCH**

Department of Computer Science &amp; Engineering

**Subject Name: Java Programming****Semester: III****Subject Code: CSE201****Academic year: 2024-25****Part-1**

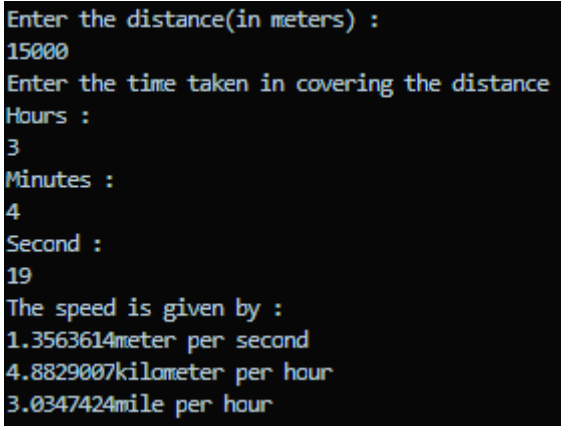
<b>No.</b>	<b>Aim</b>
<b>1.</b>	<p>Demonstration of installation steps of Java, Introduction to Object Oriented Concepts, comparison of Java with other object-oriented programming languages. Introduction to JDK, JRE, JVM, Javadoc, command line argument. Introduction to Eclipse or NetBeans IDE, or BlueJ and Console Programming.</p> <p>1. Installation of Java</p> <p>Steps to install Java Development Kit (JDK):</p> <p><input type="checkbox"/> Download JDK:</p> <ul style="list-style-type: none"><li>- Go to the Oracle JDK download page: [Oracle JDK Downloads] (<a href="https://www.oracle.com/java/technologies/javase-downloads.html">https://www.oracle.com/java/technologies/javase-downloads.html</a>).</li><li>- Select the appropriate JDK version for your operating system (Windows, macOS, Linux).</li><li>- Download the installer package (.exe for Windows, .dmg for macOS, .tar.gz for Linux).</li></ul> <p><input type="checkbox"/> Install JDK:</p> <ul style="list-style-type: none"><li>- Windows: Double-click the downloaded .exe file and follow the installation instructions.</li><li>- macOS: Double-click the downloaded .dmg file, then drag and drop the JDK package icon to the Applications folder.</li></ul>

	<ul style="list-style-type: none"> <li>- Linux: Extract the downloaded .tar.gz file to a directory and follow the instructions in the README file for installation.</li> </ul> <p>□ Set JAVA_HOME (Optional):</p> <ul style="list-style-type: none"> <li>- Windows: Set the JAVA_HOME environment variable to the JDK installation directory.</li> <li>- macOS/Linux: Add the JDK bin directory to your PATH and set JAVA_HOME in your shell profile (e.g., ~/.bash_profile, ~/.bashrc).</li> </ul> <p>□ Verify Installation:</p> <ul style="list-style-type: none"> <li>- Open a terminal or command prompt.</li> <li>- Type java -version and javac -version to verify that Java runtime and compiler are installed correctly.</li> </ul> <h2>2. Introduction to Object-Oriented Concepts</h2> <p>Object-oriented programming (OOP) revolves around the concept of objects, which are instances of classes. Key principles include:</p> <ul style="list-style-type: none"> <li>- Classes and Objects: Classes define the blueprint for objects.</li> <li>- Encapsulation: Bundling data (attributes) and methods (functions) that operate on the data within a single unit (class).</li> <li>- Inheritance: Mechanism where a new class (derived or child class) is created from an existing class (base or parent class).</li> <li>- Polymorphism: Ability of different objects to be treated as instances of the same class through method overriding and overloading.</li> </ul> <h2>3. Comparison of Java with Other Object-Oriented Programming Languages</h2> <p>Java is often compared with languages like C++, C#, and Python in terms of syntax, features, and application domains. Key points of comparison include:</p> <ul style="list-style-type: none"> <li>- Syntax: Java has a C-style syntax with similarities to C++.</li> <li>- Memory Management: Java uses automatic garbage collection, unlike C++ which requires manual memory management.</li> <li>- Platform Independence: Java programs are compiled into bytecode, which can run on any JVM, making it platform-independent.</li> <li>- Libraries: Java has a rich standard library (Java API) comparable to those in C++ and C#.</li> <li>- Community and Ecosystem: Java has a large developer community and extensive third-party libraries and frameworks.</li> </ul> <h2>4. Introduction to JDK, JRE, JVM, Javadoc, Command Line Arguments</h2>
--	---

	<ul style="list-style-type: none"> <li>- JDK (Java Development Kit): Includes tools for developing and running Java programs, including JRE and development tools such as javac (Java compiler).</li> <li>- JRE (Java Runtime Environment): Includes JVM (Java Virtual Machine) and libraries required to run Java applications, but does not include development tools.</li> <li>- JVM (Java Virtual Machine): Executes Java bytecode and provides a runtime environment for Java programs.</li> <li>- Javadoc: Tool for generating API documentation from Java source code comments.</li> <li>- Command Line Arguments: Parameters passed to a Java program when it is invoked from the command line.</li> </ul> <p>5. Introduction to Eclipse or NetBeans IDE (Integrated Development Environment)</p> <ul style="list-style-type: none"> <li>- Eclipse : A widely used open-source IDE for Java development, also supports other programming languages through plugins. Features include code editing, debugging, and version control integration.</li> <li>- NetBeans: Another popular open-source IDE primarily for Java development, with features similar to Eclipse.</li> </ul> <p>6. Introduction to BlueJ and Console Programming</p> <ul style="list-style-type: none"> <li>- BlueJ : A lightweight IDE specifically designed for teaching and learning Java programming, providing a simplified interface and visualization tools for object-oriented concepts.</li> <li>- Console Programming : Refers to writing Java programs that interact with users via text-based input and output through the console (command line interface).</li> </ul>
2.	<p>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.</p> <p><b>Program code:</b></p> <pre>public class prac2 {     public static void main(String[] args) {         int balance=20;         System.out.println("The balance in user account is : "+balance+"\$");     } }</pre> <p><b>Output:</b></p>  <p><b>Conclusion:</b></p> <p>From this practical , we can learn how to store the value in the variable and print it on</p>

	the console screen.
3.	<p>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).</p> <p><b>Program code:</b></p> <pre>import java.util.Scanner;  public class prac3 {      public static void main(String[] args) {          float distanceM;         float km, mile, thrs, tsec, mps, kph, Mph;         float hours, min, sec;          Scanner sc = new Scanner(System.in);          System.out.println("Enter the distance(in meters) :");         distanceM = sc.nextFloat();          System.out.println("Enter the time taken in covering the distance\nHours :");         hours = sc.nextFloat();          System.out.println("Minutes : ");         min = sc.nextFloat();          System.out.println("Second : ");         sec = sc.nextFloat();          thrs = hours + (min / 60) + (sec / 3600);         tsec = sec + (min * 60) + (hours * 3600);          km = distanceM / 1000;         mile = distanceM / 1609;          mps = distanceM / tsec;         kph = km / thrs;         Mph = mile / thrs;          System.out.println("The speed is given by : ");</pre>



	<pre> System.out.println(mps + "meter per second"); System.out.println(kph + "kilometer per hour"); System.out.println(Mph + "mile per hour"); sc.close(); }  } </pre> <p><b>Output:</b></p>  <pre> Enter the distance(in meters) : 15000 Enter the time taken in covering the distance Hours : 3 Minutes : 4 Second : 19 The speed is given by : 1.3563614meter per second 4.8829007kilometer per hour 3.0347424mile per hour </pre> <p><b>Conclusion:</b> From this practical, we learn that how we can perform the arithmetic operation on the variable and perform the given task as per our requirement.</p>
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><b>Program code:</b></p> <pre> import java.util.Scanner;  public class prac4 {     public static void main(String[] args) {         float totalExpense=0;         float array[]= new float[30];         Scanner sc= new Scanner(System.in);         for (int index = 0; index &lt; array.length; index++) {             array[index]=sc.nextFloat();             System.out.println("Expense of day "+(index+1)+" is "+array[index]);             totalExpense+=array[index];         }         System.out.println("The Total Expense is "+totalExpense);     } } </pre>

}

**Output:**

```

12
Expense of day 1 is 12.0
23
Expense of day 2 is 23.0
2
Expense of day 3 is 2.0
12
Expense of day 4 is 12.0
1
Expense of day 5 is 1.0
212
Expense of day 6 is 212.0
12
Expense of day 7 is 12.0
12
Expense of day 8 is 12.0
1
Expense of day 9 is 1.0
21
Expense of day 10 is 21.0
2
Expense of day 11 is 2.0
121
Expense of day 12 is 121.0
2
Expense of day 13 is 2.0
1
Expense of day 14 is 1.0
2
Expense of day 15 is 2.0

```

```

Expense of day 16 is 12.0
2
Expense of day 17 is 2.0
2
Expense of day 18 is 2.0
2
Expense of day 19 is 2.0
2
Expense of day 20 is 2.0
2
Expense of day 21 is 2.0
2
Expense of day 22 is 2.0
2
Expense of day 23 is 2.0
2
Expense of day 25 is 2.0
2
Expense of day 26 is 2.0
2
Expense of day 27 is 2.0
2
Expense of day 28 is 2.0
2
Expense of day 29 is 2.0
2
Expense of day 30 is 2.0
The Total Expense is 476.0

```

**Conclusion:**

From this practical, we conclude that how we can store our daily expenses in the array and calculate the total expenses.

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:**

```

public class prac5 {
    public static void main(String[] args) {

        float price[] = { 100, 200, 300, 400, 500 };
        int code[] = { 1, 2, 3, 4, 5 };
        float totalBill = 0;

        for (int i = 0; i < code.length; i++) {
            switch (code[i]) {

```

```
case 1:
    System.out.println("The price of motor is :"+price[i]);
    totalBill += price[i] + (0.08 * price[i]);
    break;

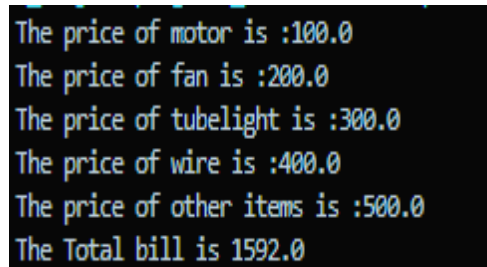
case 2:
    System.out.println("The price of fan is :"+price[i]);
    totalBill += price[i] + (0.12 * price[i]);
    break;

case 3:
    System.out.println("The price of tubelight is :"+price[i]);
    totalBill += price[i] + (0.05 * price[i]);
    break;

case 4:
    System.out.println("The price of wire is :"+price[i]);
    totalBill += price[i] + (0.075 * price[i]);
    break;

case 5:
    System.out.println("The price of other items is :"+price[i]);
    totalBill += price[i] + (0.03 * price[i]);
    break;

default:
    break;
}
}
System.out.println("The Total bill is "+totalBill);
}
```

**Output:**

```
The price of motor is :100.0
The price of fan is :200.0
The price of tubelight is :300.0
The price of wire is :400.0
The price of other items is :500.0
The Total bill is 1592.0
```

**Conclusion:**

From this practical , we learn about the array declaration , initialisation and the accessing the value from the array.



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.Scanner;

public class prac6 {
    public static void main(String[] args)
    {
        double total=1;
        double a=0,b=1;
        Scanner sc =new Scanner(System.in);

        System.out.println("Enter the number of terms :");
        int n = sc.nextInt();
        System.out.println("The fibonacci series is : \n"+a+"\n"+b);

        for (int i = 2; i < n; i++) {
            double nextterm=a+b;
            a=b;
            b=nextterm;
            total+=b;
            System.out.println(b);
        }
        System.out.println("the total of fibonnaci series is : " + (long)total );
        sc.close();
    }
}
```

**Output:**

```

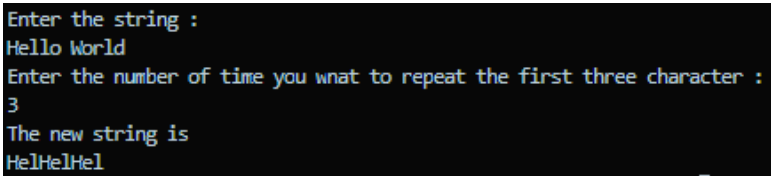
Enter the number of terms : 46368.0
50 75025.0
The fibonacci series is : 121393.0
0.0 196418.0
1.0 317811.0
1.0 514229.0
2.0 832040.0
3.0 1346269.0
5.0 2178309.0
8.0 3524578.0
13.0 5702887.0
21.0 9227465.0
34.0 1.4930352E7
55.0 2.4157817E7
89.0 3.9088169E7
144.0 6.3245986E7
233.0 1.02334155E8
377.0 1.65580141E8
610.0 2.67914296E8
987.0 2.67914296E8
1597.0 4.33494437E8
2584.0 7.01408733E8
4181.0 1.13490317E9
6765.0 1.836311903E9
10946.0 2.971215073E9
17711.0 4.807526976E9
28657.0 7.778742049E9
46368.0 the total of fibonnaci series is : 20365011073

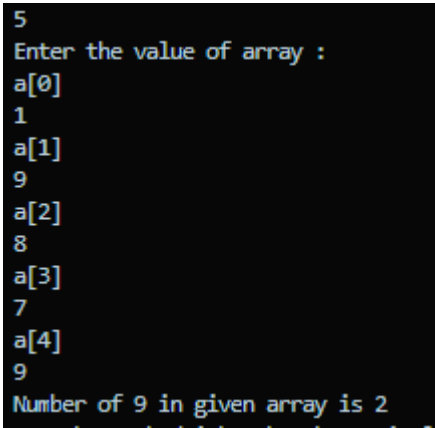
```

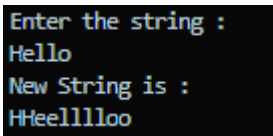
### Conclusion:

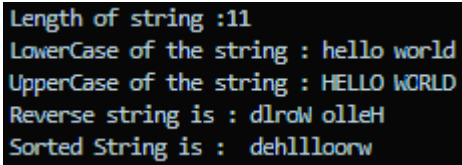
From this practical, we conclude that how we can perform action using the loops .

## Part-2

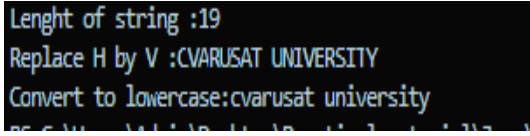
No.	Aim
7.	<p>Given a string and a non-negative int n, we'll say that the front of the string is the first 3 chars, or whatever is there if the string is less than length 3. Return n copies of the front; front_times('Chocolate', 2) → 'ChoCho' front_times('Chocolate', 3) → 'ChoChoCho' front_times('Abc', 3) → 'AbcAbcAbc'.</p> <p><b>Program code:</b></p> <pre>import java.util.*;  public class prac7 {     static String front_times(String a,int b){          String newstr="";         if(a.length()&lt;=3)         {             for (int i = 0; i &lt; b; i++) {                 newstr+=a;             }         }         else{             for (int i = 0; i &lt; b; i++) {                 newstr+=a.substring(0, 3);             }         }         return newstr;     }     public static void main(String[] args) {         String a;         Scanner sc =new Scanner(System.in);         System.out.println("Enter the string :");         a=sc.nextLine();         System.out.println("Enter the number of time you want to repeat the first three character :");         int n=sc.nextInt();         String newstr=front_times(a,n);          System.out.println("The new string is \n"+newstr);         sc.close();     } }</pre> <p><b>Output:</b></p>  <pre>Enter the string : Hello World Enter the number of time you want to repeat the first three character : 3 The new string is HelHelHel</pre>

	<p><b>Conclusion:</b></p> <p>From this practical , we cam to know how we can pass the string as the argument of the function and use it in our function as per our requirement .</p>
8.	<p>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</p> <p><b>Program code:</b></p> <pre>import java.util.Scanner;  public class prac8 {     static int count9(int a[]) {         int count = 0;         for (int i = 0; i &lt; a.length; i++) {             if (a[i] == 9)                 count++;         }         return count;     }     public static void main(String[] args) {         int size;         Scanner sc=new Scanner(System.in);         size=sc.nextInt();         int array[]=new int[size];         System.out.println("Enter the value of array :");         for (int i=0;i&lt;size;i++) {             System.out.println("a["+i+"]");             array[i]=sc.nextInt();         }         System.out.println("Number of 9 in given array is "+count9(array));         sc.close();     } }</pre> <p><b>Output:</b></p>  <p>The screenshot shows the execution of the program. It prompts the user to enter the value of the array. The user enters 5, and the program prompts for each element of the array from a[0] to a[4]. The user enters the values 1, 9, 8, 7, and 9 respectively. The final output is "Number of 9 in given array is 2".</p> <p><b>Conclusion:</b></p> <p>From this cam to know we can pass the array as the argument of the function and</p>

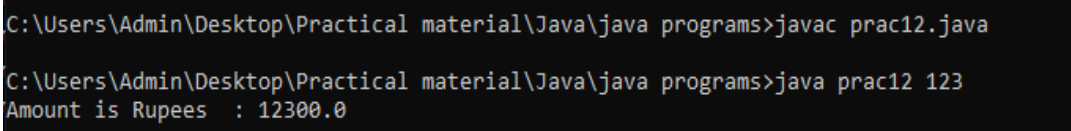
	deal with the different functionality with them.
<b>9.</b>	<p>Given a string, return a string where for every char in the original, there are two chars. <code>double_char('The') → 'TThhee'</code> <code>double_char('AAbb') → 'AAAAbbbb'</code> <code>double_char('Hi-There') → 'HHii--TThheerree'</code></p> <p><b>Program code:</b></p> <pre>import java.util.Scanner;  public class prac9 {     static String double_char(String str)     {         String newstr="";         for (int i = 0; i &lt; str.length(); i++) {             newstr+= str.charAt(i);             newstr+= str.charAt(i);         }          return newstr;     }      public static void main(String[] args) {         String str;         Scanner sc=new Scanner(System.in);         System.out.println("Enter the string :");         str=sc.nextLine();         System.out.println("New String is :\n"+double_char(str));         sc.close();     } }</pre> <p><b>Output:</b></p>  <p><b>Conclusion:</b></p> <p>From this practical , we came to know about how we can return the string type of data by using any function.</p>
<b>10.</b>	<p>Perform following functionalities of the string:</p> <ul style="list-style-type: none"> <li>Find Length of the String</li> <li>Lowercase of the String</li> <li>Uppercase of the String</li> <li>Reverse String</li> <li>Sort the string</li> </ul> <p><b>Program code:</b></p> <pre>import java.lang.reflect.Array; import java.util.Arrays;</pre>

	<pre> public class prac10 {     public static void main(String[] args) {         String str ="Hello World";         System.out.println("Length of string :"+str.length());         System.out.println("LowerCase of the string : "+str.toLowerCase());         System.out.println("UpperCase of the string : "+str.toUpperCase());          char arr[]=str.toCharArray();         char newarr[]=new char[str.length()];         for (int i = (arr.length-1),j=0; i &gt;=0; i--) {             newarr[j] = arr[i];             j++;         }          System.out.println("Reverse string is : "+new String(newarr));          String str1=str.toLowerCase();         newarr=str1.toCharArray();         Arrays.sort(newarr);          System.out.println("Sorted String is : "+new String(newarr));      } } </pre> <p><b>Output:</b></p>  <p><b>Conclusion:</b> From this we learn the different function available in the string class and how we can use it in our programming task.</p>
<p><b>11.</b></p>	<p>Perform following Functionalities of the string: “CHARUSAT UNIVERSITY” Find length Replace ‘H’ by ‘FIRST LATTER OF YOUR NAME’ Convert all character in lowercase</p> <p><b>Program code:</b></p> <pre> public class prac11 {     public static void main(String[] args) {         String str="CHARUSAT UNIVERSITY";         System.out.println("Lenght of string :"+str.length());         System.out.println("Replace H by V :"+(str=str.replace('H','V')));         System.out.println("Convert to lowercase:"+str.toLowerCase());     } } </pre>



	<pre>} }</pre> <p><b>Output:</b></p>  <p><b>Conclusion:</b></p> <p>From this we learn the different function available in the string class and how we can use it in our programming task.</p>
--	---

### Part III

No.	Aim
12.	<p>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.</p> <p><b>Program code:</b></p> <pre>public class prac12{     static float converToRs(float pound){         return pound*100;     }     public static void main(String[] args) {         float pound;         //System.out.println("Enter the amount in pound : ");         //Scanner sc=new Scanner(System.in);         pound=Float.parseFloat(args[0]);         System.out.println("Amount is Rupees : "+converToRs(pound));     } }</pre> <p><b>Output:</b></p>  <p><b>Conclusion:</b></p> <p>From this practical , we can learn how to the inline argument from the user while running the program in the terminal</p>
13.	<p>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.</p> <p><b>Program code:</b></p> <pre>import java.util.Scanner;  class Employee{     String Fname,Lname;     double sal,ysal;</pre>

```
public Employee() {
}

Employee(String f, String l, double s)
{
    Fname=f;
    Lname=l;
    if(s<0)
    {
        sal=0;
    }
    else {
        sal=s;
    }
    ysal=sal*12;
}
Scanner sc=new Scanner(System.in);
void set_fname()
{
    System.out.println("Enter First Name : ");
    Fname=sc.nextLine();
}
void set_lname()
{
    System.out.println("Enter Last Name : ");
    Lname=sc.nextLine();
}
void set_sal()
{
    System.out.println("Enter Salary : ");
    sal=sc.nextDouble();
    ysal=sal*12;
}

void get_fname()
{
    System.out.println("First name : "+Fname);
}
void get_lname()
{
    System.out.println("Last name : "+Lname);
}
void get_sal()
{
    System.out.println("Salary : "+sal);
}
void get_ysal(){

    System.out.println("Yearly salary is : "+ysal);
}
```

```
}  
  
class prac13 {  
    public static void main(String[] args) {  
        Employee e1,e2;  
        e1= new Employee();  
        e2= new Employee("Veer","Patel",50000);  
        e1.set_fname();  
        e1.set_lname();  
        e1.set_sal();  
  
        System.out.println("\nDetails of Employee 1 :");  
        e1.get_fname();  
        e1.get_lname();  
        e1.get_sal();  
  
        e1.get_ysal();  
        e1.ysal+=e1.ysal*0.1;  
        System.out.println("Salary of employee 1 after the 10% increment : ");  
        e1.get_ysal();  
  
        System.out.println("\nDetails of Employee 2 :");  
        e2.get_fname();  
        e2.get_lname();  
        e2.get_sal();  
        e2.get_ysal();  
  
        e2.ysal+=e2.ysal*0.1;  
        System.out.println("Salary of employee 2 after the 10% increment : ");  
  
        e2.get_ysal();  
    }  
}
```

**Output:**

	<pre> Enter First Name : veer Enter Last Name : patel Enter Salary : 100000  Details of Employee 1 : First name : veer Last name : patel Salary : 100000.0 Yearly salary is : 1200000.0 Salary of employee 1 after the 10% increment : Yearly salary is : 1320000.0  Details of Employee 2 : First name : Veer Last name : Patel Salary : 50000.0 Yearly salary is : 600000.0 Salary of employee 2 after the 10% increment : Yearly salary is : 660000.0 </pre> <p><b>Conclusion:</b></p> <p>From this practical , we can learn how to initialise the value of the given class's object using different types of constructor and access its value using different method.</p>
14.	<p>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.</p> <p><b>Program code:</b></p> <pre> class Date{     int day,month,year;      Date()     {         day=1;         month=1;         year=2001;     }     Date(int day,int month, int year){         this.day=day;         this.month=month;         this.year=year;     }      public void setDay(int day) {         this.day = day;     } </pre>

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

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

public int getDay() {
    return day;
}

public int getMonth() {
    return month;
}

public int getYear() {
    return year;
}

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

}

public class prac14 {
    public static void main(String[] args) {
        Date d1=new Date();
        Date d2=new Date(21, 10, 2005);
        Date d3=new Date();

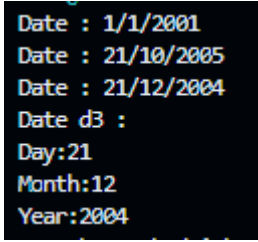
        d3.setDay(21);
        d3.setMonth(12);
        d3.setYear(2004);

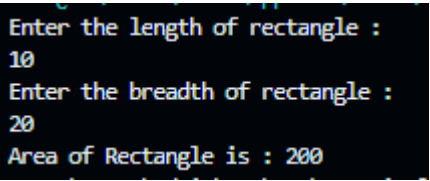
        d1.displayDate();
        d2.displayDate();
        d3.displayDate();

        System.out.println("Date d3 :");
        System.out.println("Day:"+d3.getDay());
        System.out.println("Month:"+d3.getMonth());
        System.out.println("Year:"+d3.getYear());

    }
}
```



	<p><b>Output:</b></p>  <p><b>Conclusion:</b></p> <p>From this practical , we can learn how we can use the different methods in the class using object as per our requirement.</p>
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><b>Program code:</b></p> <pre>import java.util.Scanner; import java.lang.Math;  class Area {      double length, breadth;      public Area() {         length = 1;         breadth = 1;     }      public Area(double length, double breadth) {         this.length = length;         this.breadth = breadth;     }      public double returnArea() {         return length * breadth;     } }  public class prac15 {     public static void main(String[] args) {         //Area a1=new Area();         double length , breadth;         Scanner sc=new Scanner(System.in) ;         System.out.println("Enter the length of rectangle :");         length=sc.nextDouble();         System.out.println("Enter the breadth of rectangle :");         breadth=sc.nextDouble();</pre>

	<pre> Area a2= new Area(length,breadth);  //System.out.println("Area of Rectangle 1 is : "+a1.returnArea()); System.out.println("Area of Rectangle is : "+Math.round(a2.returnArea())); } } </pre> <p><b>Output:</b></p>  <p><b>Conclusion:</b></p> <p>From this practical , we can learn how we can work with the methods by passing some argument and returning some value as per our requirement.</p>
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> <p><b>Program code:</b></p> <pre> import java.util.*;  class Complex {      int real, img;      Complex() {         real = 0;         img = 0;     }      Complex(int r, int i) {         real = r;         img = i;     }      void display() {         if (img &gt;= 0) {             System.out.println(real + "+" + img + "i");         } else {             System.out.println(real + img + "i");         }     }      static void add (Complex c1, Complex c2){         Complex c =new Complex();         c.real=c1.real+c2.real;         c.img =c1.img+c2.img;     } } </pre>

```

        System.out.print("The Sum is : ");
        c.display();
    }
    static void sub (Complex c1, Complex c2){
        Complex c =new Complex();
        c.real=c1.real-c2.real;
        c.img =c1.img-c2.img;
        System.out.print("The difference is : " );
        c.display();
    }
    static void mul (Complex c1, Complex c2){
        Complex c =new Complex();
        c.real=(c1.real*c2.real)-(c1.img*c2.img);
        c.img =(c1.real*c2.img)+(c1.img*c2.real);
        System.out.print("The Product is : ");
        c.display();
    }
}

public class prac16 {
    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);
        int r1,r2,i1,i2;
        System.out.println("Enter the value of Complex number 1 :\nreal part:");
        r1=sc.nextInt();
        System.out.println("imaginary part : ");
        i1=sc.nextInt();
        System.out.println("Enter the value of Complex number :\nreal part:");
        r2=sc.nextInt();
        System.out.println("imaginary part : ");
        i2=sc.nextInt();

        Complex c1=new Complex(r1,i1);
        Complex c2=new Complex(r2,i2);

        Complex.add(c1,c2);
        Complex.sub(c1,c2);
        Complex.mul(c1,c2);

    }
}

```

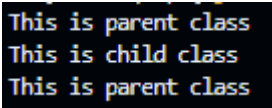
**Output:**

```
Enter the value of Complex number 1 :  
real part:  
2  
imaginary part :  
3  
Enter the value of Complex number :  
real part:  
5  
imaginary part :  
6  
The Sum is : 7+9i  
The difference is : -3-3i  
The Product is : -8+27i
```

**Conclusion:**

From this practical , we can learn how to use the static methods in class and passing and returning the object of the class.

## Part IV

No.	Aim
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> <p><b>Program code:</b></p> <pre> class parent{     void print_parent()     {         System.out.println("This is parent class");     } }  class child extends parent{     void print_child()     {         System.out.println("This is child class");     } }  public class prac17 {     public static void main(String[] args)     {         parent p1 =new parent();         child c1 = new child();         p1.print_parent();         c1.print_child();         c1.print_parent();     } } </pre> <p><b>Output:</b></p>  <p><b>Conclusion:</b></p> <p>From this practical , we can learn how use the concept of the inheritance and use the method of parent class in the child class.</p>
18	<p>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.</p>

**Program code:**

```
import java.math.BigInteger;
import java.util.Scanner;

class Member{
    String name,address;
    int age;
    BigInteger phone_number;
    float salary;
    Scanner sc=new Scanner(System.in);
    void set_data()
    {
        System.out.println("Enter the name :");
        name=sc.nextLine();
        System.out.println("Enter the Age :");
        age=sc.nextInt();
        System.out.println("Enter the Phone number :");
        phone_number=sc.nextBigInteger();
        sc.nextLine();
        System.out.println("Enter the Address :");
        address=sc.nextLine();
        System.out.println("Enter the Salary :");
        salary=sc.nextFloat();
    }

    void get_data(){
        System.out.println("Name : "+name);
        System.out.println("Age : "+age);
        System.out.println("Phone number : "+phone_number);
        System.out.println("Address : "+address);
    }

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

class Employee extends Member{
    String specialisation;
    @Override
    void set_data()
    {
        super.set_data();
        System.out.println("Enter the area of specialisation : ");
        sc.nextLine();
        specialisation=sc.nextLine();
    }
    @Override
    void get_data(){
```



```

        super.get_data();
        super.print_Salary();
        System.out.println("Specialisation : "+specialisation);
    }
}

class Manager extends Member{
    String department;
    @Override
    void set_data()
    {
        super.set_data();
        System.out.println("Enter the department : ");
        sc.nextLine();
        department=sc.nextLine();
    }
    @Override
    void get_data(){
        super.get_data();
        super.print_Salary();
        System.out.println("Department : "+department);
    }
}

public class prac18 {
    public static void main(String[] args) {
        Employee e1=new Employee();
        Manager m1=new Manager();
        System.out.println("Enter the data of Employee : ");
        e1.set_data();
        System.out.println("Enter the data of Manager : ");
        m1.set_data();

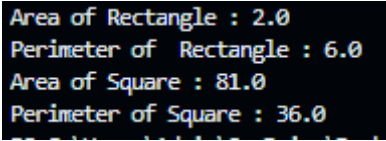
        System.out.println("\nEmployee details :\n");
        e1.get_data();

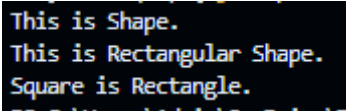
        System.out.println("\nManager Details :\n");
        m1.get_data();
    }
}

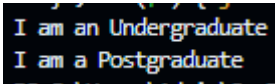
```

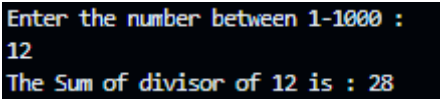
**Output:**

	<div style="display: flex; justify-content: space-between;"> <div style="width: 48%;"> <pre> Enter the data of Employee : Enter the name : Veer Enter the Age : 19 Enter the Phone number : 1231231231 Enter the Address : 12 abc bungalows nadiad gujarat Enter the Salary : 50000 Enter the area of specialisation : AIML Enter the data of Manager : Enter the name : veer2 Enter the Age : 20 Enter the Phone number : 3453453453 Enter the Address : 13 hello row house anand gujarat Enter the Salary : </pre> </div> <div style="width: 48%;"> <pre> Enter the Salary : 70000 Enter the department : Hr  Employee details :  Name : Veer Age : 19 Phone number : 1231231231 Address : 12 abc bungalows nadiad gujarat Salary : 50000.0 Specialisation : AIML  Manager Details :  Name : veer2 Age : 20 Phone number : 3453453453 Address : 13 hello row house anand gujarat Salary : 70000.0 Department : Hr </pre> </div> </div> <p><b>Conclusion:</b></p> <p>From this practical , we can learn how can call the method of the parent class in the child class and use the variables declared in the parent class in the child class.</p>
19	<p>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.</p> <p><b>Program code:</b></p> <pre> class Rectangle{     float length,breadth;     Rectangle()     {         length=0;         breadth=0;     }     Rectangle(float l, float b)     {         length=l;         breadth=b;     }      void area()     {         System.out.println("Area of Rectangle : "+length*breadth);     } } </pre>

	<pre>     }     void perimeter()     {         System.out.println("Perimeter of Rectangle : "+(2*(length+breadth)));     } }  class Square extends Rectangle{      Square() {         super();     }     Square(float s){         super(s,s);     }      void area()     {         System.out.println("Area of Square : "+length*breadth);     }     void perimeter()     {         System.out.println("Perimeter of Square : "+(2*(length+breadth)));     }  }  public class prac19 {     public static void main(String[] args) {         Rectangle r1[]=new Rectangle[2];         Square s1=new Square(9);         r1[0]=new Rectangle(1,2);         r1[0].area();         r1[0].perimeter();          s1.area();         s1.perimeter();     } } </pre> <p><b>Output:</b></p>  <pre> Area of Rectangle : 2.0 Perimeter of Rectangle : 6.0 Area of Square : 81.0 Perimeter of Square : 36.0 </pre> <p><b>Conclusion:</b></p> <p>From this practical , we can learn how to use the constructor of the parent in the child class using the super keyword.</p>
20	Create a class named 'Shape' with a method to print "This is This is shape". Then

	<p>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.</p> <p><b>Program code:</b></p> <pre> class Shape{     void print_shape(){         System.out.println("This is Shape.");     } }  class Rectangle extends Shape{     void print_rect(){         System.out.println("This is Rectangular Shape.");     } }  class Circle extends Shape{     void print_circle(){         System.out.println("This is Circluar Shape.");     } }  class Square extends Rectangle{     void print_square(){         System.out.println("Square is Rectangle.");     } }  public class prac20 {     public static void main(String[] args) {         Square sq=new Square();         sq.print_shape();         sq.print_rect();         sq.print_square();     } } </pre> <p><b>Output:</b></p>  <p><b>Conclusion:</b></p> <p>From this practical , we can learn that we can call the method of the parent and grandparent class using the object of the child class.</p>
21	<p>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</p>

	<p>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.</p> <p><b>Program code:</b></p> <pre> class Degree{     void getDegree(){         System.out.println("I got 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 prac21 {     public static void main(String[] args) {         Undergraduate std1=new Undergraduate();         Postgraduate std2=new Postgraduate();         std1.getDegree();         std2.getDegree();     } } </pre> <p><b>Output:</b></p>  <p><b>Conclusion:</b></p> <p>From this practical , we can learn that we can have the same method name for the different children class of the same parent class.</p>
22	<p>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. 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.</p> <p><b>Program code:</b></p> <pre> import java.util.Scanner;  interface AdvancedArithmetic {     int divisor_sum(int n); }  class MyCalculator implements AdvancedArithmetic { </pre>

	<pre> public int divisor_sum(int n){     int sum=0;     for(int i=1;i&lt;=n;i++){         if(n%i==0)             sum+=i;     }     return sum; } }  public class prac22 {     public static void main(String[] args) {         MyCalculator c1=new MyCalculator();         int n;         do{             System.out.println("Enter the number between 1-1000 :");             Scanner sc=new Scanner(System.in);             n=sc.nextInt();         }while(n&lt;1  n&gt;1000);          System.out.println("The Sum of divisor of "+n+" is : "+c1.divisor_sum(n));      } } </pre> <p><b>Output:</b></p>  <p><b>Conclusion:</b> From this practical , we can learn how we can use the interface instead of class only for the declaration of method and inherit it into the class.</p>
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> <p><b>Program code:</b></p> <pre> import java.util.Scanner;  interface Shape {     String getColour();     default void displayColour() {         System.out.println("Colour: " + getColour());     } } </pre>



```
interface Circle extends Shape {
    int getRadius();
    default void displayCircle() {
        System.out.println("Shape: Circle");
        System.out.println("Radius: " + getRadius());
        displayColour();
    }
}

interface Rectangle extends Shape {
    int getLength();
    int getWidth();

    void createRectangleSign(int length, int width, String colour, String text);
    default void displayRectangle() {
        System.out.println("Shape: Rectangle");
        System.out.println("Length: " + getLength());
        System.out.println("Width: " + getWidth());
        displayColour();
    }
}

class Sign implements Rectangle, Circle {
    private String colour;
    private String text;
    private int radius;
    private int length;
    private int width;

    public void createCircleSign(int radius, String colour, String text) {
        this.colour = colour;
        this.radius = radius;
        this.text = text;
    }

    public void createRectangleSign(int length, int width, String colour, String text) {
        this.colour = colour;
        this.length = length;
        this.width = width;
        this.text = text;
    }

    public String getColour() {
        return colour;
    }

    public int getRadius() {
        return radius;
    }
}
```

```
public int getLength() {
    return length;
}

public int getWidth() {
    return width;
}

public void displayCircleSign() {
    displayCircle();
    System.out.println("Text: " + text);
}

public void displayRectangleSign() {
    displayRectangle();
    System.out.println("Text: " + text);
}
}

public class prac23 {
    public static void main(String[] args) {
        Sign sign = new Sign();
        int choice, length, width, radius;
        String colour, text;
        Scanner sc = new Scanner(System.in);

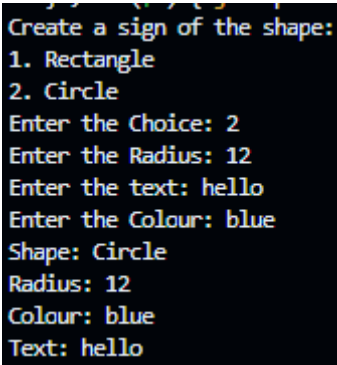
        System.out.println("Create a sign of the shape:");
        System.out.println("1. Rectangle");
        System.out.println("2. Circle");
        System.out.print("Enter the Choice: ");
        choice = sc.nextInt();
        sc.nextLine(); // Consume the newline

        switch (choice) {
            case 1:
                System.out.print("Enter the Length: ");
                length = sc.nextInt();
                System.out.print("Enter the Width: ");
                width = sc.nextInt();
                sc.nextLine(); // Consume the newline
                System.out.print("Enter the text: ");
                text = sc.nextLine();
                System.out.print("Enter the Colour: ");
                colour = sc.nextLine();
                sign.createRectangleSign(length, width, colour, text);
                sign.displayRectangleSign();
                break;

            case 2:
```

```
        System.out.print("Enter the Radius: ");
        radius = sc.nextInt();
        sc.nextLine(); // Consume the newline
        System.out.print("Enter the text: ");
        text = sc.nextLine();
        System.out.print("Enter the Colour: ");
        colour = sc.nextLine();
        sign.createCircleSign(radius, colour, text);
        sign.displayCircleSign();
        break;

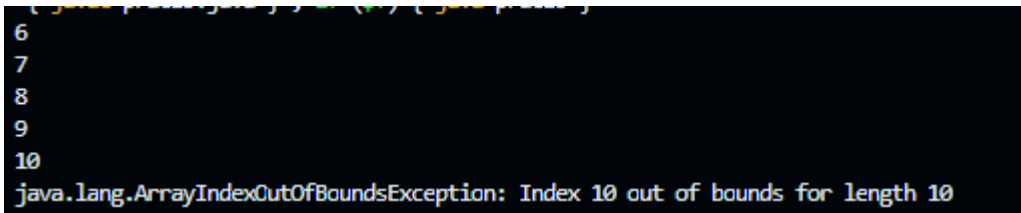
    default:
        System.out.println("Invalid choice.");
        break;
    }
}
```

**Output:**A screenshot of a terminal window showing the output of a Java program. The text is as follows:  
Create a sign of the shape:  
1. Rectangle  
2. Circle  
Enter the Choice: 2  
Enter the Radius: 12  
Enter the text: hello  
Enter the Colour: blue  
Shape: Circle  
Radius: 12  
Colour: blue  
Text: hello**Conclusion:**

From this practical , we can learn how we can inherit multiple interface from one interface and the class from multiple interface.

## Part V

No.	Aim
24	<p>Write a java program which takes two integers x &amp; 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.</p> <p><b>Program code:</b></p> <pre>import java.util.Scanner;  import java.util.InputMismatchException;  public class prac24 {      public static void main(String[] args) {         int x, y;         float ans;          try {             System.out.println("Enter the value of x :");             Scanner sc = new Scanner(System.in);             x = sc.nextInt();             System.out.println("Enter the value of y :");             y = sc.nextInt();             ans = x / y;             System.out.println(ans);         } catch (ArithmeticException e) {             System.out.println("The Value of y can't be zero!!!!");         } catch (InputMismatchException e2) {             System.out.println(e2);         } catch (Exception e3) {             System.out.println(e3);         }      } }</pre> <p><b>Output:</b></p> <div style="display: flex; justify-content: space-around; align-items: flex-start;"> <div style="background-color: black; color: white; padding: 5px; width: 40%;"> <pre>Enter the value of x : 2 Enter the value of y : 0 The Value of y can't be zero!!!!</pre> </div> <div style="background-color: black; color: white; padding: 5px; width: 40%;"> <pre>Enter the value of x : 3.46 java.util.InputMismatchException</pre> </div> </div> <div style="background-color: black; color: white; padding: 5px; width: 40%; margin-top: 10px;"> <pre>Enter the value of x : 12 Enter the value of y : 5 2.0</pre> </div> <p><b>Conclusion:</b></p> <p>From this practical , we can handle the exception occur during the runtime of the</p>

	code.
<b>25</b>	<p>Write a Java program that throws an exception and catch it using a try-catch block.</p> <p><b>Program code:</b></p> <pre>public class prac25 {     public static void main(String[] args) {         int array[]=new int[10];         for (int i = 0; i &lt; array.length; i++) {             array[i]=i+1;         }         try {             for (int i = 5; i &lt;15; i++) {                 System.out.println(array[i]);             }         } catch (ArrayIndexOutOfBoundsException e) {             System.out.println(e);         }     } }</pre> <p><b>Output:</b></p>  <p>The screenshot shows the output of the Java program. It displays the numbers 6, 7, 8, 9, and 10, followed by the exception message: java.lang.ArrayIndexOutOfBoundsException: Index 10 out of bounds for length 10.</p> <p><b>Conclusion:</b></p> <p>From this practical , we can learn how we can handle the exception using the try and catch in the java code.</p>
<b>26</b>	<p>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).</p> <p><b>Program code:</b></p> <pre>import java.util.Scanner;  class UserDefinedException extends Exception {     public UserDefinedException(String message) {         super(message);     } }  public class ExceptionHandling {     public static void main(String[] args) {         Scanner scanner = new Scanner(System.in);          // Checked exceptions         try {             System.out.print("Enter a number: ");             int number = scanner.nextInt(); </pre>

```

        if (number < 0) {
            throw new UserDefinedException("Number cannot be negative");
        }
        System.out.println("You entered: " + number);
    } catch (UserDefinedException e) {
        System.out.println("Caught UserDefinedException: " + e.getMessage());
    } catch (Exception e) {
        System.out.println("Caught Exception: " + e.getMessage());
    }
}

// Unchecked exceptions
try {
    int[] array = { 1, 2, 3 };
    System.out.println(array[4]); // ArrayIndexOutOfBoundsException
    int result = 10 / 0; // ArithmeticException
} catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("Caught ArrayIndexOutOfBoundsException: " +
e.getMessage());
} catch (ArithmeticException e) {
    System.out.println("Caught ArithmeticException: " + e.getMessage());
} finally {
    scanner.close();
}
}
}

```

**Output:**

```

Enter a number: -1
Caught UserDefinedException: Number cannot be negative
Caught ArrayIndexOutOfBoundsException: Index 4 out of bounds for length 3

```

**Conclusion:**

From this practical , we can learn can throw the user defined exception as per our requirement.

**PART-6**

<b>No.</b>	<b>Aim</b>
<b>27.</b>	<p>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.</p> <p><b>PROGRAM :</b></p> <pre>import java.io.BufferedReader; import java.io.FileReader; import java.io.IOException;  public class LineCounts {     public static void main(String[] args) {         for (String fileName : args) {             try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {                 int lineCount = 0;                 while (reader.readLine() != null) {                     lineCount++;                 }                 System.out.println(fileName + ": " + lineCount + " lines");             } catch (IOException e) {                 System.out.println("Error reading " + fileName + ": " + e.getMessage());             }         }     } }</pre> <p><b>OUTPUT:</b></p>  <p><b>CONCLUSION:</b> The Java program counts the lines in each text file given as command-line arguments, handling any read errors along the way. It outputs the results for each file, ensuring that it continues processing even if some files can't be read.`</p>
<b>28.</b>	<p>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.</p> <p><b>PROGRAM CODE:</b></p>

```

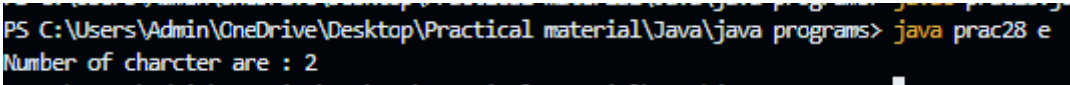
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

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

        char targetChar = args[0].charAt(0);
        String fileName = args[1];
        int count = 0;

        try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
            int ch;
            while ((ch = reader.read()) != -1) {
                if (ch == targetChar) {
                    count++;
                }
            }
            System.out.println("The character '" + targetChar + "' appears " + count + " times in "
+ fileName);
        } catch (IOException e) {
            System.out.println("Error reading " + fileName + ": " + e.getMessage());
        }
    }
}

```

**OUTPUT:**


```

PS C:\Users\Admin\OneDrive\Desktop\Practical material\Java\java programs> java prac28 e
Number of charcter are : 2

```

**CONCLUSION:**

The Java program successfully counts the occurrences of a specified character in a given file, providing the result in a clear format. It handles file read errors gracefully, ensuring robust performance even if issues arise during file access

- 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:**



```

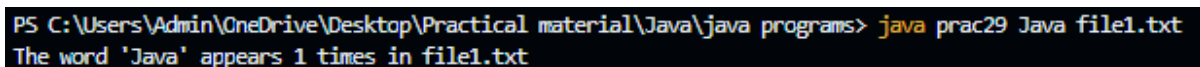
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

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

        String searchWord = args[0];
        String fileName = args[1];
        Integer count = 0;

        try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] words = line.split("\\W+");
                for (String word : words) {
                    if (word.equalsIgnoreCase(searchWord)) {
                        count++;
                    }
                }
            }
            System.out.println("The word '" + searchWord + "' appears " + count + " times in " +
fileName);
        } catch (IOException e) {
            System.out.println("Error reading " + fileName + ": " + e.getMessage());
        }
    }
}

```

**OUTPUT:**


```

PS C:\Users\Admin\OneDrive\Desktop\Practical material\Java\java programs> java prac29 Java file1.txt
The word 'Java' appears 1 times in file1.txt

```

**CONCLUSION:**

This Java program effectively searches for a specified word in a given file and counts its occurrences. It demonstrates the use of the Integer wrapper class to manage the count, showcasing how wrapper classes can be used for object manipulation in Java

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

**PROGRAM CODE:**

```

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

public class FileCopy {
    public static void main(String[] args) {
        if (args.length < 2) {
            System.out.println("Usage: java FileCopy <source file> <destination file>");
            return;
        }

        String sourceFile = args[0];
        String destinationFile = args[1];

        try (FileReader fr = new FileReader(sourceFile);
            FileWriter fw = new FileWriter(destinationFile)) {

            int ch;
            while ((ch = fr.read()) != -1) {
                fw.write(ch);
            }
            System.out.println("Data copied from " + sourceFile + " to " + destinationFile);
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}

```

**OUTPUT:**

```

PS C:\Users\Admin\OneDrive\Desktop\Practical material\Java\java programs> java prac30 file1.txt file2.txt
Data copied from file1.txt to file2.txt
PS C:\Users\Admin\OneDrive\Desktop\Practical material\Java\java programs> java prac30 file1.txt file3.txt
Data copied from file1.txt to file3.txt

```

**CONCLUSION:**

This Java program efficiently copies data from a source file to a destination file, automatically creating the destination file if it does not already exist. It handles any potential I/O exceptions during the process, ensuring robust performance.

- 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.

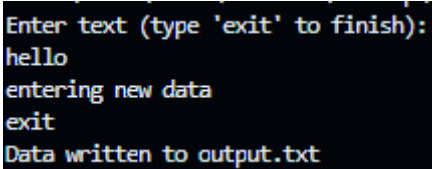
**PROGRAM CODE:**

```

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;

```

```
public class ConsoleToFile {  
    public static void main(String[] args) {  
        BufferedReader consoleReader = new BufferedReader(new  
InputStreamReader(System.in));  
        String fileName = "output.txt";  
  
        try (BufferedWriter fileWriter = new BufferedWriter(new FileWriter(fileName))) {  
            System.out.println("Enter text (type 'exit' to finish):");  
  
            String input;  
            while (!(input = consoleReader.readLine()).equalsIgnoreCase("exit")) {  
                fileWriter.write(input);  
                fileWriter.newLine();  
            }  
  
            System.out.println("Data written to " + fileName);  
        } catch (IOException e) {  
            System.out.println("Error: " + e.getMessage());  
        }  
    }  
}
```

**OUTPUT:**A screenshot of a terminal window with a black background and white text. The text shows the program's execution: a prompt 'Enter text (type \'exit\' to finish):', followed by the user input 'hello', then 'entering new data', then 'exit', and finally the program output 'Data written to output.txt'.

```
Enter text (type 'exit' to finish):  
hello  
entering new data  
exit  
Data written to output.txt
```

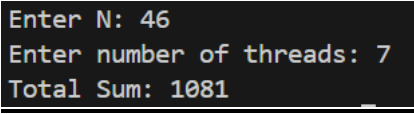
**CONCLUSION:**

This program effectively demonstrates the use of character streams via `BufferedReader` and `BufferedWriter` for reading console input and writing it to a file. It showcases how to handle text data efficiently while managing resources properly with try-with-resources

## Part-VII

No.	Aim
32.	<p>Write a program to create thread which display “Hello World” message. A. by extending Thread class B. by using Runnable interface.</p> <p><b>PROGRAM CODE:</b></p> <pre> public class HelloWorld {     static class HelloWorldThread extends Thread {         public void run() {             System.out.println("Hello World");         }     }      static class HelloWorldRunnable implements Runnable {         public void run() {             System.out.println("Hello World");         }     }      public static void main(String[] args) {         HelloWorldThread thread1 = new HelloWorldThread();         thread1.start();          Thread thread2 = new Thread(new HelloWorldRunnable());         thread2.start();     } } </pre> <p><b>OUTPUT:</b></p> <pre> Hello World Hello World </pre> <p><b>CONCLUSION:</b> This program demonstrates two approaches to creating threads in Java: extending the</p>

	Thread class and implementing the Runnable interface. Both methods effectively print "Hello World," showcasing the flexibility of Java's concurrency model.
<b>33.</b>	<p>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.</p> <p><b>PROGRAM CODE:</b></p> <pre> 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 &lt;= end; i++) {             partialSum += i;         }         synchronized (SumTask.class) {             totalSum += partialSum;         }     }      public static int getTotalSum() {         return totalSum;     } }  public class ThreadedSummation {     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 &lt; numThreads; i++) { </pre>

	<pre>         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()); } } </pre> <p><b><u>OUTPUT:</u></b></p>  <p><b>CONCLUSION:</b></p> <p>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</p>
34.	<p>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.</p> <p><b>PROGRAM CODE:</b></p> <pre> import java.util.Random;  class RandomNumberGenerator extends Thread {     private final Object lock;      public RandomNumberGenerator(Object lock) {         this.lock = lock;     } </pre>

```

public void run() {
    Random random = new Random();
    while (true) {
        int number = random.nextInt(100);
        synchronized (lock) {
            MultiThreadApplication.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 (MultiThreadApplication.lastNumber % 2 == 0) {
                    int square = MultiThreadApplication.lastNumber *
MultiThreadApplication.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 (MultiThreadApplication.lastNumber % 2 != 0) {
                int cube = MultiThreadApplication.lastNumber *
MultiThreadApplication.lastNumber * MultiThreadApplication.lastNumber;
                System.out.println("Cube: " + cube);
            }
        }
    }
}

public class MultiThreadApplication {
    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 IncrementVariable 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) {
        IncrementVariable incrementer = new IncrementVariable();
        incrementer.start();
    }
}
```

### OUTPUT:

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

### 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:

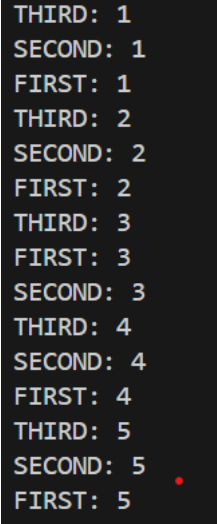
```
class MyThread extends Thread {
    public MyThread(String name) {
        super(name);
    }

    public void run() {
        for (int i = 1; i <= 5; i++) {
            System.out.println(getName() + ": " + i);
            try {
                Thread.sleep(500);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

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

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

        firstThread.start();
        secondThread.start();
```

	<pre>thirdThread.start(); } }</pre> <p><b>OUTPUT:</b></p>  <p><b>CONCLUSION:</b></p> <p>This program demonstrates thread creation and priority setting in Java. Each thread executes a simple loop, displaying its name and an iteration count, showcasing how thread priority can influence the execution order, although actual execution may vary due to the nature of thread scheduling.</p>
37.	<p>Write a program to solve producer-consumer problem using thread synchronization.</p> <p><b>PROGRAM CODE:</b></p> <pre>import java.util.LinkedList; import java.util.Queue;  class ProducerConsumer {     private final Queue&lt;Integer&gt; queue = new LinkedList&lt;&gt;();     private final int capacity = 5;      public void produce() throws InterruptedException {         int value = 0;         while (true) {             synchronized (this) {                 while (queue.size() == capacity) {                     wait();                 }                 queue.add(value);                 System.out.println("Produced: " + value);                 value++;                 notifyAll();             }         }     } }</pre>

```
        }
        Thread.sleep(1000);
    }
}

public void consume() throws InterruptedException {
    while (true) {
        synchronized (this) {
            while (queue.isEmpty()) {
                wait();
            }
            int value = queue.poll();
            System.out.println("Consumed: " + value);
            notifyAll();
        }
        Thread.sleep(1500);
    }
}
}

class Producer extends Thread {
    private final ProducerConsumer pc;

    public Producer(ProducerConsumer pc) {
        this.pc = pc;
    }

    public void run() {
        try {
            pc.produce();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}

class Consumer extends Thread {
    private final ProducerConsumer pc;

    public Consumer(ProducerConsumer pc) {
        this.pc = pc;
    }

    public void run() {
        try {
            pc.consume();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

```
}  
  
public class ProducerConsumerExample {  
    public static void main(String[] args) {  
        ProducerConsumer pc = new ProducerConsumer();  
        Producer producer = new Producer(pc);  
        Consumer consumer = new Consumer(pc);  
  
        producer.start();  
        consumer.start();  
    }  
}
```

**OUTPUT:**

```
Produced: 0  
Consumed: 0  
Produced: 1  
Produced: 2  
Consumed: 1  
Produced: 3  
Produced: 4  
Produced: 5  
Consumed: 2  
Produced: 6  
Consumed: 3  
Produced: 7  
Consumed: 4  
Produced: 8  
Produced: 9  
Consumed: 5
```

**CONCLUSION:**

This program effectively demonstrates the Producer-Consumer problem using thread synchronization in Java. The producer generates integers and adds them to a shared queue, while the consumer retrieves and consumes them. Synchronization ensures safe access to the shared resource, preventing data inconsistencies and race conditions.

**Part VIII**

<b>No.</b>	<b><i>Aim of the Practical</i></b>
38.	<p>Design a Custom Stack using ArrayList class, which implements following functionalities of stack. My Stack</p> <p>-list ArrayList&lt;Object&gt;: A list to store elements.</p> <p>+isEmpty: boolean: Returns true if this stack is empty.</p> <p>+getSize(): int: Returns number of elements in this stack.</p> <p>+peek(): Object: Returns top element in this stack without removing it.</p> <p>+pop(): Object: Returns and Removes the top elements in this stack.</p> <p>+push(o: object): Adds new element to the top of this stack.</p> <p><b><u>PROGRAM CODE:</u></b></p> <pre> import java.util.ArrayList;  class MyStack {     private ArrayList&lt;Object&gt; list = new ArrayList&lt;&gt;();      public boolean isEmpty() {         return list.isEmpty();     }      public int getSize() {         return list.size(); </pre>

```
}

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 code defines a custom stack implementation using an `ArrayList` and performs basic stack operations like push, pop, and peek. It also checks if the stack is empty and retrieves its size. The program demonstrates stack functionality with a simple usage example.

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:**

```

public class SortUtil {

    public static <T extends Comparable<T>> void sort(T[] array) {

        for (int i = 0; i < array.length - 1; i++) {

            for (int j = 0; j < array.length - i - 1; j++) {

                if (array[j].compareTo(array[j + 1]) > 0) {

```

```
        T temp = array[j];

        array[j] = array[j + 1];

        array[j + 1] = temp;

    }

}

}

}

}

public static void main(String[] args) {

    Product[] products = {

        new Product("Laptop", 1200),

        new Product("Phone", 800),

        new Product("Tablet", 600)

    };

    sort(products);

    for (Product product : products) {

        System.out.println(product.getName() + " - $" +

product.getPrice());

    }

}

}
```

```
class Product implements Comparable<Product> {  
    private String name;  
    private int price;  
  
    public Product(String name, int price) {  
        this.name = name;  
        this.price = price;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public int getPrice() {  
        return price;  
    }  
  
    @Override  
    public int compareTo(Product other) {  
        return Integer.compare(this.price, other.price);  
    }  
}
```

**OUTPUT:**

```
Tablet - $600  
Phone - $800  
Laptop - $1200
```

**CONCLUSION:**

This program demonstrates a generic bubble sort method that sorts an array of `Comparable` objects, specifically `Product` objects, based on their price. The `Product` class implements the `Comparable` interface, allowing the sorting to be based on the price attribute. After sorting, the products are displayed in ascending order of price.

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 WordCounter {  
    public static void main(String[] args) {  
        String text = "apple banana apple orange banana orange apple  
mango grape banana";  
  
        Map<String, Integer> wordCountMap = new TreeMap<>();  
        String[] words = text.split("\\s+");  
  
        for (String word : words) {  
            wordCountMap.put(word, wordCountMap.getOrDefault(word, 0)  
+ 1);  
        }  
  
        Set<Map.Entry<String, Integer>> entrySet =  
wordCountMap.entrySet();  
        for (Map.Entry<String, Integer> entry : entrySet) {  
            System.out.println(entry.getKey() + ": " + entry.getValue());  
        }  
    }  
}
```

**OUTPUT:**

```
apple: 3
banana: 3
grape: 1
mango: 1
orange: 2
```

### **CONCLUSION:**

This program counts the occurrences of each word in a given text and displays them in alphabetical order using a `TreeMap`. It demonstrates basic string manipulation, word counting, and sorting capabilities.

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: prac27.java  
Number of Java keywords in the file: 15
```

**CONCLUSION:**

This program counts the number of Java keywords in a source file by reading the file and checking each word against a predefined set of keywords stored in a `HashSet`. It demonstrates keyword detection using file handling and basic string comparison.