



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

## Certificate

*This is to certify that*

*Mr./Mrs. Kulathiya Aesha Hanikrushnabhai*

*of 3CSE1 Class,*

*ID. No. 23DCS043 has satisfactorily completed*

*his/ her term work in JAVA Programming -(CSE201) for*

*the ending in October 2024/2025*

*Date : 16/10/24*

*Sign. of Faculty*

*Head of Department*

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

**Department of Computer Science & Engineering**

**Subject Name: JAVA PROGRAMMING**

**Semester: 3**


**Subject Code: CSE201**

**Academic year:2024-25**

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

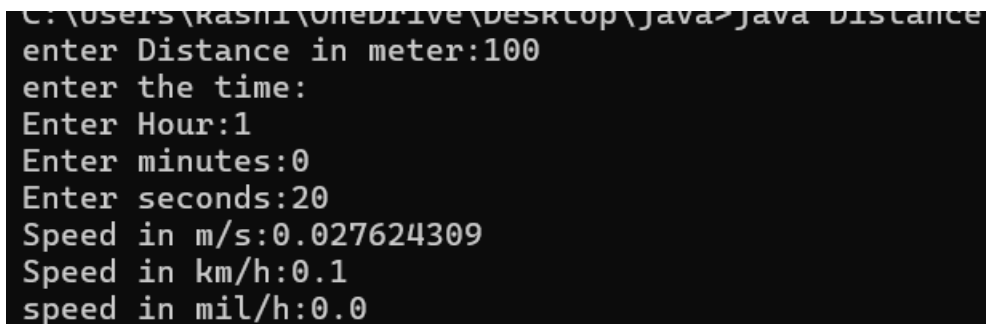
| No. | Aim of the Practical  |
|-----|---|
| 1.  | <p>Demonstration of installation steps of Java, Introduction to Object Oriented Concepts, comparison of Java with other object-oriented programming languages. Introduction to JDK, JRE, JVM, Javadoc, command line argument. Introduction to Eclipse or NetBeans IDE, or BlueJ and Console Programming.</p> <p><b>PROGRAM CODE:</b></p> <div data-bbox="592 1025 1198 1417" data-label="Diagram"> <pre> graph LR     subgraph JDK [Java Development Kit]         subgraph JRE [Java Runtime Environment (JRE)]             subgraph JVM [Java Virtual Machine (JVM)]             end             JRE --- LCL[Library classes +]         end         JRE --- DT[Dev tools]     end </pre> </div> <p>1. JDK (Java Development Kit) is a Kit that provides the environment to develop and execute(run) the Java program. JDK is a kit(or package) that includes two things Development Tools(to provide an environment to develop your java programs) JRE (to execute your java program).</p> <p>2. JRE (Java Runtime Environment) is an installation package that provides an environment only run(not develop) the java program(or application)onto your machine. JRE is only used those who only want to run Java programs that are end-users of your system.</p> <p>3. JVM (Java Virtual Machine) is a very important part of both JDK and JRE because it is contained or inbuilt in both. Whatever Java program you run using JRE or JDK goes into</p> |

|    |  |
|----|--|
|    | <p>JRE and JDK is responsible for executing the java program line by line, hence it is also known as an interpreter.</p> <p>JVM(Java Virtual Machine) acts as a run-time engine to run Java applications. JVM is the one that actually calls the main method present in a java code. JVM is a part of JRE(Java Runtime Environment).</p> <p>Java applications are called WORA (Write Once Run Anywhere). This means a programmer can develop Java code on one system and can expect it to run on any other Java-enabled system without any adjustment. This is all possible because of JVM.</p> <p>When we compile a <i>.java</i> file, <i>.class</i> files(contains byte-code) with the same class names present in <i>.java</i> file are generated by the Java compiler.</p> <p><b>4.javaDoc :</b> JavaDoc tool is a document generator tool in Java programming language for generating standard documentation in HTML format. It generates API documentation. It parses the declarations and documentation in a set of source file describing classes, methods, constructors, and fields.</p> <p>Before using JavaDoc tool, you must include JavaDoc comments <code>/**.....*/</code> providing information about classes, methods, and constructors, etc. For creating a good and understandable document API for any java file you must write better comments for every class, method, constructor.</p> <p><b>5.Command Line Argument :</b> Java command-line argument is an argument i.e. passed at the time of running the Java program. In the command line, the arguments passed from the console can be received in the java program and they can be used as input. The users can pass the arguments during the execution bypassing the command-line arguments inside the <code>main()</code> method.</p> |
| 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><u>PROGRAM CODE:</u></b></p>  |

|    |  |
|----|--|
|    | <pre>import java.util.*; class Bank{ public static void main(String args[]){ int a=20; System.out.println("Current Balance:\$"+a); } }</pre> <p><b><u>OUTPUT:</u></b></p>  <p><b><u>CONCLUSION:</u></b></p> <p>This program demonstrates the basic concept of storing and displaying data in Java, showcasing the use of variables and output statements.</p>  |
| 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><u>PROGRAM CODE:</u></b></p> <pre>import java.util.*; class Distance{ public static void main(String args[]){ Scanner sc=new Scanner(System.in); System.out.print("enter Distance in meter:"); int d=sc.nextInt(); System.out.println("enter the time:"); System.out.print("Enter Hour:"); int hr=sc.nextInt(); System.out.print("Enter minutes:"); int min=sc.nextInt(); System.out.print("Enter seconds:"); int sec=sc.nextInt(); int time=(hr*3600)+(min*60)+sec; float v=(float)d/time; System.out.println("Speed in m/s:"+v); time=hr+(min/60)+(sec/3600); v=(float)(d*0.001)/time; System.out.println("Speed in km/h:"+v); }</pre> |

```
v=(float)(d/1609)/time;
System.out.println("speed in mil/h:"+v);
}
}
```

### **OUTPUT:**



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


### **CONCLUSION:**

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

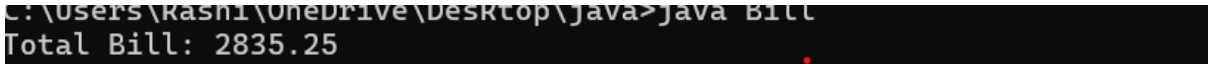
4. Imagine you are developing a budget tracking application. You need to calculate the total expenses for the month. Users will input their daily expenses, and the program should compute the sum of these expenses. Write a Java program to calculate the sum of elements in an array representing daily expenses.

### **PROGRAM CODE:**

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

|    |   |
|----|---|
|    | <pre>} </pre> <p><b><u>OUTPUT:</u></b></p>  <pre>Day1:5 Day2:444 Day3:50 Day4:100 Day5:56 total Expenses:655 </pre> <p><b><u>CONCLUSION:</u></b></p> <p>This program demonstrates the use of arrays and loops to collect and calculate the sum of daily expenses, providing a simple yet effective way to track monthly expenditures.</p>   |
| 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> <p><b><u>PROGRAM CODE:</u></b></p> <pre>class Bill{     public static void main(String[] args) {          int[] codes = {1, 2, 3, 4, 5};         double[] prices = {1000, 500, 200, 150, 800};          double totalBill = 0;         for (int i = 0; i &lt; codes.length; i++) {             totalBill += calculateBill(codes[i], prices[i]);         }          System.out.println("Total Bill: " + totalBill);     } } </pre> |



|    |  |
|----|--|
|    | <pre> public static double calculateBill(int code, double price) {     double bill = 0;     switch (code) {         case 1:             bill = price + (price * 0.08); // 8% tax for motor             break;         case 2:             bill = price + (price * 0.12); // 12% tax for fan             break;         case 3:             bill = price + (price * 0.05); // 5% tax for tube light             break;         case 4:             bill = price + (price * 0.075); // 7.5% tax for wires             break;         default:             bill = price + (price * 0.03); // 3% tax for all other items             break;     }     return bill; } } </pre> <p><b><u>OUTPUT:</u></b></p>  <p><b><u>CONCLUSION:</u></b></p> <p>This program demonstrates the use of a switch statement to apply different tax rates based on product codes, calculating the total bill by iterating through the arrays of codes and prices. It showcases a practical application of conditional statements in Java programming.</p> |
| 6. | <p>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</p>   |

calculate and display the first n terms of the Fibonacci series, representing the exercise duration for each day.

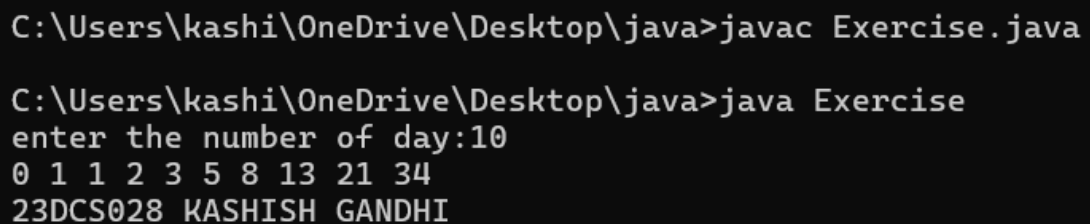
**PROGRAM CODE:**

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

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

}}
```

**OUTPUT:**



```
C:\Users\kashi\OneDrive\Desktop\java>javac Exercise.java

C:\Users\kashi\OneDrive\Desktop\java>java Exercise
enter the number of day:10
0 1 1 2 3 5 8 13 21 34
23DCS028 KASHISH GANDHI
```

**CONCLUSION:**

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

**PART-II Strings**



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

front\_times('Chocolate', 2) → 'ChoCho'  
front\_times('Chocolate', 3) → 'ChoChoCho'  
front\_times('Abc', 3) → 'AbcAbcAbc'

### **PROGRAM CODE:**

```
import java.util.*;

class FrontTimes{
    public static void main(String[] args) {
        System.out.println(frontTimes("Chocolate", 2));
        System.out.println(frontTimes("Chocolate", 3));
        System.out.println(frontTimes("Abc", 3));
    }

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

### **OUTPUT:**

```
C:\Users\kashi\OneDrive\Desktop\java>javac FrontTimes.java
C:\Users\kashi\OneDrive\Desktop\java>java FrontTimes
ChoCho
ChoChoCho
AbcAbcAbc
23DCS028 KASHISH GANDHI
```

### **CONCLUSION:**

To solve the problem of generating n copies of the front part of a string in Java, use the substring function to extract the first three characters (or the entire string if it's shorter). Then, repeat this segment n times. This method ensures correct handling even if the string is shorter than three

|   |  |
|---|--|
|   | <p>characters. The solution is efficient, using simple string slicing and repetition.</p>  |
| 8 | <p>Given an array of ints, return the number of 9's in the array.</p> <ul style="list-style-type: none"> <li>array_count9([1, 2, 9]) → 1</li> <li>array_count9([1, 9, 9]) → 2</li> <li>array_count9([1, 9, 9, 3, 9]) → 3</li> </ul> <p><b><u>PROGRAM CODE:</u></b></p> <pre>import java.util.*; class Count9{ public static int array_count9(int[] nums) {     int count = 0;     for (int num : nums) {         if (num == 9) {             count++;         }     }     return count; }  public static void main(String[] args) {     int[] nums1 = {1, 2, 9};     int[] nums2 = {1, 9, 9};     int[] nums3 = {1, 9, 9, 3, 9};      System.out.println("number of 9 in {1, 2, 9} array:"+ array_count9(nums1)); // Output: 1     System.out.println("number of 9 in {1, 9, 9} array:" + array_count9(nums2)); // Output: 2     System.out.println("number of 9 in {1, 9, 9, 3, 9} array:" +array_count9(nums3)); // Output: 3     ; } }</pre> <p><b><u>OUTPUT:</u></b></p> <pre>number of 9 in {1, 2, 9} array:1 number of 9 in {1, 9, 9} array:2 number of 9 in {1, 9, 9, 3, 9} array:3</pre> |

**CONCLUSION:**

In this practical we learn how to use function argument, use the count variable And print number of int in the array is same.

- 9 Given a string, return a string where for every char in the original, there are two chars.

double\_char('The') → 'TThhee'

double\_char('AAbb') → 'AAAAbbbb'

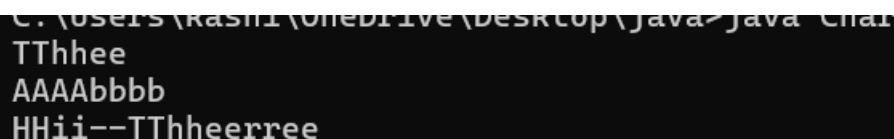
double\_char('Hi-There') → 'HHii--TThheerree'

**PROGRAM CODE:**

```
import java.util.*;
class Char{

public static String double_char(String str) {
    String result = "";
    for (int i = 0; i < str.length(); i++) {
        result += str.substring(i, i + 1) + str.substring(i, i + 1);
    }
    return result;
}

public static void main(String[] args) {
    System.out.println(double_char("The"));
    System.out.println(double_char("AAbb"));
    System.out.println(double_char("Hi-There"));
}
}
```

**OUTPUT:**


```
C:\Users\RASHI\OneDrive\Desktop\java>java Char
TThhee
AAAAbbbb
HHii--TThheerree
```

**CONCLUSION:**

To double every character in a string in Java, iterate through each character, appending it twice to a new string. This method ensures each character is duplicated, creating a new string with repeated characters. It

|    |   |
|----|---|
|    | provides an efficient and straightforward solution for string manipulation tasks.   |
| 10 | <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, Sort the string</li></ul> <p><b><u>PROGRAM CODE:</u></b></p> <pre>import java.util.*; class Fun{  public static int findLength(String str) {     return str.length(); }  public static String toLowercase(String str) {     return str.toLowerCase(); }  public static String toUppercase(String str) {     return str.toUpperCase(); }  public static String reverseString(String str) {     StringBuilder sb = new StringBuilder(str);     return sb.reverse().toString(); }  public static String sortString(String str) {     char chars[] = str.toCharArray();     Arrays.sort(chars);     return new String(chars); }</pre> |

```

}
public static void main(String args[]){
    String str = "Kashish";

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

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

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

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

    String sorted = sortString(str);
    System.out.println("Sorted: " + sorted);
;
}
}

```

### **OUTPUT:**

```

Length: 7
Lowercase: kashish
Uppercase: KASHISH
Reversed: hsihsaK
Sorted: Kakhiss

```

### **CONCLUSION:**

In this practical we learn how to use different type of function use like, how to find length using length(), toLowercase(), toUpperCase(), reversed(), and sort the String and display the output.

- |     |   |
|-----|---|
| 11. | <p>Perform following Functionalities of the string:<br/>“CHARUSAT UNIVERSITY”</p> <ul style="list-style-type: none"> <li>● Find length</li> <li>● Replace ‘H’ by ‘FIRST LATTER OF YOUR NAME’</li> <li>● Convert all character in lowercase</li> </ul> |
|-----|---|

**PROGRAM CODE:**

```
import java.util.*;
class Replace{
public static void main(String args[]){
    String str="CHARUSAT UNIVERCITY";
    int length=str.length();
    System.out.println("String length="+length);
    Scanner sc=new Scanner(System.in);
    System.out.println("enter your name first letter:");
    char n;
    n=sc.next().charAt(0);

    String replacedStr = str.replace('H', n);

    System.out.println("Replaced string: " + replacedStr);
    String lowerCaseStr = replacedStr.toLowerCase();
    System.out.println("Lowercase String:"+lowerCaseStr);
    ;

}}
```

**OUTPUT:**

```
String length=19
enter your name first letter:
K
Replaced string: CKARUSAT UNIVERCITY
Lowercase String:ckarusat univercity
```

**CONCLUSION:**

This code finds the length of the string, replaces 'H' with 'your name first latter', and converts all characters to lowercase, printing the results.



**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.

**PROGRAM CODE:**

```
import java.util.*;

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

**OUTPUT:**

```
C:\Users\Rashi\OneDrive\Desktop\java>java Pra
Enter amount in Pounds: 28
28.0 Pounds = 2800.0 Rupees
```

**CONCLUSION:**

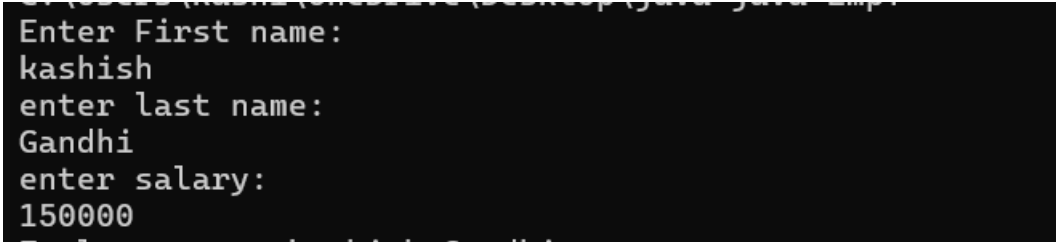
|    |  |
|----|--|
|    | The Java program converts Pounds to Rupees at a fixed rate of 1 Pound = 100 Rupees, handling both command-line arguments and interactive user inputs effectively, ensuring user-friendly currency conversion.  |
| 13 | <p>Create a class called Employee that includes three pieces 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><u>PROGRAM CODE:</u></b></p> <pre>import java.util.*; class Employee{     Scanner sc=new Scanner(System.in);      String fs=" ";     String ls=" ";     double sal;     Employee(String f,String l,double s){         fs=f;         ls=l;         sal=s;     }     void setfs(){         fs=sc.nextLine();     }      void setls(){         ls=sc.nextLine();     }      void setsal(){         sal=sc.nextDouble();         if(sal&lt;0){             sal=0.0;         }         else{             sal=sal+(sal*0.10);         }     }      String getfs(){         return fs;     } }</pre> |

```

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

System.out.println("Employee name:"+E1.getfs()+" "+E1.getls());
System.out.println("Salary:"+E1.getsal());
System.out.println("\n23DCS025 KASHISH GANDHI");
}
}

```

**OUTPUT:**


```

Enter First name:
kashish
enter last name:
Gandhi
enter salary:
150000

```

**CONCLUSION:**

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

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

**PROGRAM CODE:**

```

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

```

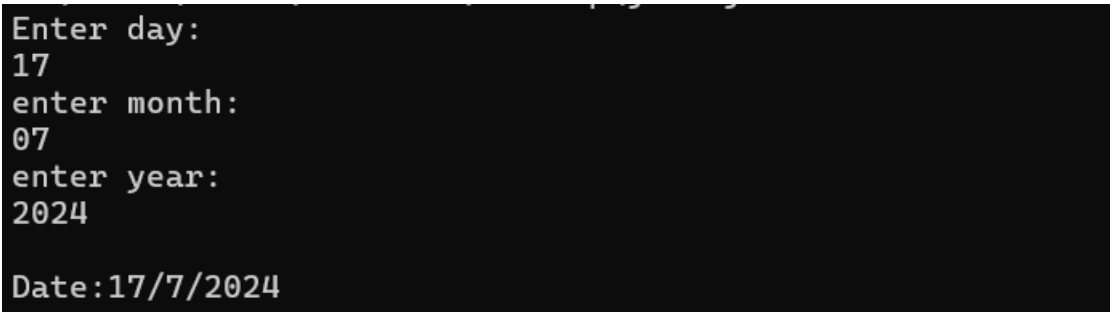
```
int d;
int m;
int y;
Date(int day, int month ,int year){
d=day;
m=month;
y=year;
}
void setd(){
d=sc.nextInt();
if(d>31){
System.out.println("enter right day");
}
}


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

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

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

System.out.println("\nDate:"+d+"/"+m+"/"+y);
}
}
class DateTest{
```

|    |  |
|----|--|
|    | <pre> public static void main(String args[]){     Date D=new Date(0,0,0);     System.out.println("Enter day:");     D.setd();     System.out.println("enter month:");     D.setm();     System.out.println("enter year:");     D.sety();     D.displayDate(); } } </pre> <p><b><u>OUTPUT:</u></b></p>  <p><b><u>CONCLUSION:</u></b></p> <p>The Date class successfully manages date information with proper setters, getters, and a display method. The DateTest application effectively demonstrates the class's ability to handle and display date information.</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><u>PROGRAM CODE:</u></b></p> <pre> import java.util.*; class Area{     int l, b;     Scanner sc=new Scanner(System.in);      Area(){     }     Area(int length, int breadth){         b=breadth;         l=length;     } } </pre>  |

|    |  |
|----|--|
|    | <pre> void setl(){ l=sc.nextInt(); } void setb(){ b=sc.nextInt(); }  void returnArea(){ System.out.println(l*b); } }  class AreaT{  public static void main(String args[]){ Area a=new Area(0,0); System.out.print("Enter the length:"); a.setl(); System.out.print("Enter the Breadth:"); a.setb(); System.out.print("Area of rectangle:"); a.returnArea(); } </pre> <p><b>OUTPUT:</b></p>  <p><b>CONCLUSION:</b></p> <p>The program effectively calculates and prints the area of a rectangle using the Area class, which initializes length and breadth via constructor parameters and provides a method to return the calculated area.</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.Scanner;  class Complex {     private double real;     private double imag;      public Complex(double real, double imag) {         this.real = real;         this.imag = imag;     } </pre>   |



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

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

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

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

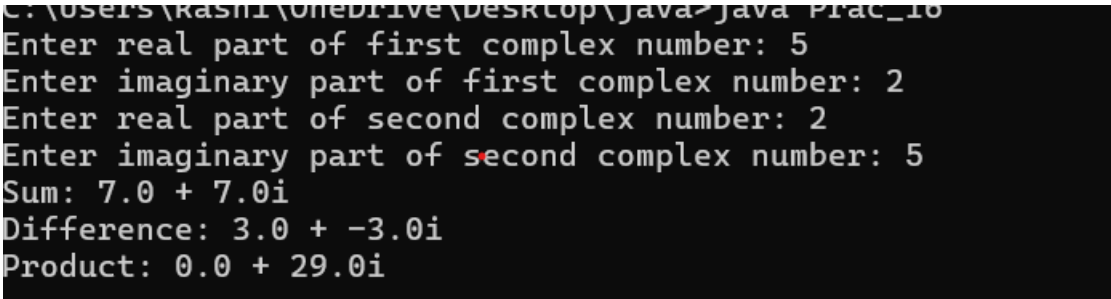
public class Prac_16 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

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

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

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

        Complex sum = num1.add(num2);
```

|  |   |
|--|---|
|  | <pre> Complex difference = num1.subtract(num2); Complex product = num1.multiply(num2);  System.out.println("Sum: " + sum); System.out.println("Difference: " + difference); System.out.println("Product: " + product);     } } </pre> <p><b><u>OUTPUT:</u></b></p>  <p><b><u>CONCLUSION:</u></b></p> <p>The Complex class correctly performs addition, subtraction, and multiplication of two complex numbers, with methods for each operation. User input for real and imaginary parts demonstrates the class's functionality.</p> |
|--|---|

### 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> <p><b><u>PROGRAM CODE:</u></b></p> <pre> import java.util.*; class P17{     P17(){         System.out.println("This is parent class");     }     public static void main(String args[]){          child c=new child();         c.dis();     } } </pre> |
|-----|--|

|     |  |
|-----|--|
|     | <pre>     } } class child extends P17{     void dis(){         System.out.println("This is child class.");     } } </pre> <p><b>OUTPUT:</b></p> <pre> This is parent class This is child class. </pre> <p><b>CONCLUSION:</b></p> <p>The program demonstrates inheritance with a 'child' class extending the 'P17' class. It creates an object of the 'child' class and calls its specific method while also showing the parent class message via the constructor, illustrating basic class inheritance.</p>  |
| 18. | <p>Create a class named 'Member' having the following members: Data members</p> <ol style="list-style-type: none"> <li>1 - Name</li> <li>2 - Age</li> <li>3 - Phone number</li> <li>4 - Address</li> <li>5 – Salary</li> </ol> <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> <p><b><u>PROGRAM CODE:</u></b></p> <pre> import java.util.*; public class P18 {     static String name;     static int age;     static long number;     static String address;     static int salary;     static String dep;     static String special;     static int i; </pre> |

```
static Scanner s= new Scanner(System.in);
public static void getdata(){
    System.out.print("Enter the name:");
    s.nextLine();
    name=s.nextLine();
    System.out.print("Enter the age:");
    age=s.nextInt();
    System.out.print("Enter the phone number:");
    number=s.nextLong();
    System.out.print("Enter the home Address:");
    s.nextLine();
    address=s.nextLine();
    System.out.print("Enter the salary:");
    salary=s.nextInt();
}
static void printSalary(){
    System.out.println("salary:"+salary);
}
public static void setdata(){
    System.out.println("name:"+name);
    System.out.println("age:"+age);
    System.out.println("phone number:"+number);
    System.out.println("home Address:"+address);
    printSalary();
    if(i==1){
        System.out.println("specialization:"+special);
    }
    else{
        System.out.println("department:"+dep);
    }
}
}

public static void main(String args[]){

    System.out.println("Enter the 1 for Employee.");
    System.out.println("Enter the 2 for Manager.");
    System.out.print("enter your choice:");
    i=s.nextInt();
    switch(i){
        case 1:
            employee e=new employee();
            e.specialization();
            System.out.println("----- Employee.-----");
```

```
e.setdata();
break;
case 2:
Manager m=new Manager();
m.department();
System.out.println("----- Manager.-----");
m.setdata();
}
s.close();
}

}
class employee extends P18{
void specialization(){
getdata();
System.out.print("Enter the specialization:");
s.nextLine();
special=s.nextLine();

}
}
class Manager extends P18{
void department(){
getdata();
System.out.print("Enter the department:");
s.nextLine();
dep=s.nextLine();
}
}
}
```

**OUTPUT:**

```

Enter the 1 for Employee.
Enter the 2 for Manager.
enter your choice:1
Enter the name:kashish
Enter the age:19
Enter the phone number:9876543210
Enter the home Address:g 201
Enter the salary:520000
Enter the specialization:cse
----- Employee.-----
name:kashish
age:19
phone number:9876543210
home Address:g 201
salary:520000
specialization:cse
PS C:\Users\kashi\OneDrive\Desktop\vs\java_prc\set4>

```

### **CONCLUSION:**

The program showcases inheritance by having 'Employee' and 'Manager' classes extend the 'P18' class. It collects and prints details using polymorphism, with specific attributes for each subclass, demonstrating data handling and method overriding in Java.

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

### **PROGRAM CODE:**

```

class Rectangle {
    double length, breadth;

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

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

```



```
}

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

// Class Square inheriting Rectangle
class Square extends Rectangle {
    // Constructor
    Square(double side) {
        super(side, side);
    }
}

public class P19 {
    public static void main(String[] args) {
        // Array of objects
        Rectangle[] rectangles = new Rectangle[2];

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

        // Printing area and perimeter
        for (Rectangle rectangle : rectangles) {
            rectangle.printArea();
            rectangle.printPerimeter();
            System.out.println();
        }
    }
}
```

**OUTPUT:**

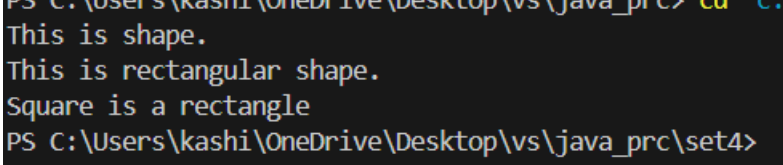
```
Area of Rectangle: 50.0
Perimeter of Rectangle: 30.0

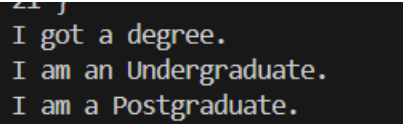
Area of Rectangle: 25.0
Perimeter of Rectangle: 20.0

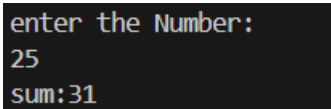
PS C:\Users\kashi\OneDrive\Desktop\vs\java_prc\set4> █
```

**CONCLUSION:**

|     |   |
|-----|---|
|     | <p>The program demonstrates inheritance, where the 'Square' class inherits from the 'Rectangle' class. It also uses constructors to initialize values and an array of objects to print the area and perimeter, showcasing polymorphism and object-oriented principles.</p>  |
| 20. | <p>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.</p> <p><b><u>PROGRAM CODE:</u></b></p> <pre> class Shape{     public static void printShape(){         System.out.println("This is shape.");     } class Rectangle extends Shape{     public static void printRectangle(){         System.out.println("This is rectangular shape.");     } }  class Circle extends Shape{     public static void printCircle(){         System.out.println("This is circular shape.");     } }  class Square extends Rectangle{     public static void printSquare(){         System.out.println("Square is a rectangle");     } }  public class P20 {     public static void main(String[] args) {         Square s=new Square();         s.printShape();         s.printRectangle();         s.printSquare();     } </pre> |

|     |  |
|-----|--|
|     | <pre> } </pre> <p><b>OUTPUT:</b></p>  <p><b>CONCLUSION:</b></p> <p>The program demonstrates inheritance in Java, where the 'Square' class inherits methods from both 'Shape' and 'Rectangle'. The 'Square' object can call methods from its superclass, showcasing method inheritance and polymorphism in object-oriented programming.</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 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 a degree.");     } } class Undergraduate extends Degree{     void getDegree(){         System.out.println("I am an Undergraduate.");     } } class Postgraduate extends Degree{     void getDegree(){         System.out.println("I am a Postgraduate.");     } } public class P21 {     public static void main(String[] args) {         Degree D=new Degree();         Undergraduate UG=new Undergraduate();         Postgraduate PG=new Postgraduate();          D.getDegree();         UG.getDegree();         PG.getDegree();     } } </pre> |

|     |   |
|-----|---|
|     | <pre>     } } </pre> <p><b>OUTPUT:</b></p>  <p><b>CONCLUSION:</b></p> <p>This program demonstrates method overriding in Java, where the subclasses 'Undergraduate' and 'Postgraduate' provide their own implementations of the 'getDegree()' method. Each object calls its respective method, showcasing polymorphism and inheritance in object-oriented programming.</p>  |
| 22. | <p>Write a java that implements an interface AdvancedArithmetic which contains amethod signature int divisor_sum(int n). You need to write a class calledMyCalculator which implements the interface. divisorSum function just takes an integer as input and return the sum of all its divisors.</p> <p>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.*; interface AdvancedArithmetic {     int divisor_sum(int n); } class calledMyCalculator implements AdvancedArithmetic{     public int divisor_sum(int n){         int sum=0;         int sqrt =(int) Math.sqrt(n);         for(int i=1;i&lt;=sqrt;i++){             if(n%i==0){                 sum+=i;                 if(i!=n/i){                     sum +=n/i;                 }             }         }         return sum;     } } </pre> |

|     |   |
|-----|---|
|     | <pre>     } }  class P22 {     static Scanner sc=new Scanner(System.in);     public static void main(String args[]){         calledMyCalculator cmc=new calledMyCalculator();         System.out.println("enter the Number:");         int x=sc.nextInt();          System.out.println("sum:"+cmc.divisor_sum(x));      } } </pre> <p><b>OUTPUT:</b></p>  <p><b>CONCLUSION:</b></p> <p>The program demonstrates implementing the 'AdvancedArithmetic' interface with the 'calledMyCalculator' class. It efficiently calculates and returns the sum of divisors for a given integer using a loop up to its square root, optimizing performance.</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> // Shape.java interface Shape {     String getColor();     double getArea();      default void printDetails() {         System.out.println("Color: " + getColor());         System.out.println("Area: " + getArea());     } } </pre> |

```
}

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

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

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

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

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

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

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

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



```
}  
}  
  
// Sign.java  
class Sign {  
    private Shape backgroundShape;  
    private String text;  
  
    public Sign(Shape backgroundShape, String text) {  
        this.backgroundShape = backgroundShape;  
        this.text = text;  
    }  
  
    public void display() {  
        System.out.println("Sign Text: " + text);  
        System.out.println("Background Shape Details:");  
        backgroundShape.printDetails();  
    }  
}  
  
// Main.java  
public class P23 {  
    public static void main(String[] args) {  
        Shape circle = new Circle(5, "Red");  
        Shape rectangle = new Rectangle(4, 6, "Blue");  
  
        Sign sign1 = new Sign(circle, "Welcome to the Campus!");  
        Sign sign2 = new Sign(rectangle, "Library Entrance");  
  
        System.out.println("Sign 1:");  
        sign1.display();  
  
        System.out.println("\nSign 2:");  
        sign2.display();  
    }  
}  
  
OUTPUT:
```

```

23 }
Sign 1:
Sign Text: Welcome to the Campus!
Background Shape Details:
Color: Red
Area: 78.53981633974483

```

```

Sign 2:
Sign Text: Library Entrance
Background Shape Details:
Color: Blue
Area: 24.0

```

### **CONCLUSION:**

The program highlights the use of an interface default method, allowing shared functionality like `printDetails()` across shape classes (`Circle` and `Rectangle`). This reduces code duplication, promotes code reuse, and demonstrates backward compatibility while simplifying object design in Java.

## **PART-V Exception Handling**

- 24 . Write a java program which takes two integers x & y as input, you have to compute x/y. If x and y are not integers or if y is zero, exception will occur and you have to report it.**

### **PROGRAM CODE:**

```

import java.util.*;
public class P24 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

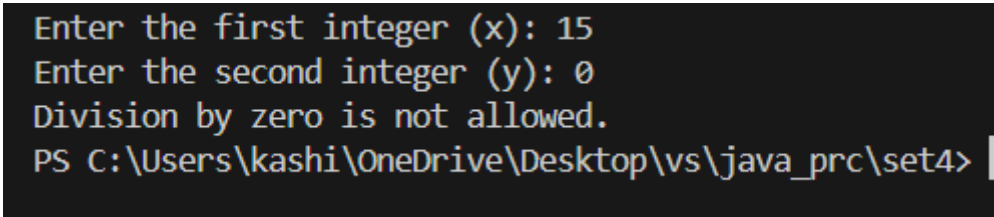
        try {

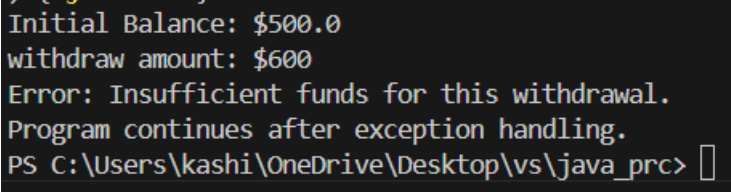
            System.out.print("Enter the first integer (x): ");
            int x = scanner.nextInt();
            System.out.print("Enter the second integer (y): ");
            int y = scanner.nextInt();

            int result = x / y;
            System.out.println("The result of x / y is: " + result);

        }
        catch (ArithmeticException e) {

```

|    |   |
|----|---|
|    | <pre>         System.out.println("Division by zero is not allowed.");      } finally {          scanner.close();     } } </pre> <p><b>OUTPUT:</b></p>  <pre> Enter the first integer (x): 15 Enter the second integer (y): 0 Division by zero is not allowed. PS C:\Users\kashi\OneDrive\Desktop\vs\java_prc\set4&gt; </pre> <p><b>CONCLUSION:</b></p> <p>The provided Java program effectively handles potential exceptions during user input and division. It uses try-catch blocks to catch ArithmeticException for division by zero. Additionally, a finally block ensures the scanner is closed. This program demonstrates robust input validation and error handling.</p> |
| 25 | <p><b>Write a Java program that throws an exception and catch it using a try-catch block.</b></p> <p><b>PROGRAM CODE:</b></p> <pre> class BankAccount {     private double balance;      public BankAccount(double initialBalance) {         this.balance = initialBalance;     }      public void withdraw(double amount) {         if (amount &gt; balance) {             throw new RuntimeException("Insufficient funds for this withdrawal.");         }         balance -= amount;         System.out.println("Successfully withdrew: \$" + amount);     }      public double getBalance() { </pre>  |

|    |   |
|----|---|
|    | <pre>         return balance;     } }  public class P25 {     public static void main(String[] args) {         try {              BankAccount account = new BankAccount(500);              System.out.println("Initial Balance: \$" + account.getBalance());             System.out.println("withdraw amount: \$600");              account.withdraw(600);             System.out.println("Current Balance: \$" + account.getBalance());          } catch (RuntimeException e) {              System.out.println("Error: " + e.getMessage());         }          System.out.println("Program continues after exception handling.");     } } </pre> <p><b>OUTPUT:</b></p>  <pre> Initial Balance: \$500.0 withdraw amount: \$600 Error: Insufficient funds for this withdrawal. Program continues after exception handling. PS C:\Users\kashi\OneDrive\Desktop\vs\java_prc&gt; </pre> <p><b><u>CONCLUSION:</u></b></p> <p>The provided Java code demonstrates the use of a try-catch block to handle a RuntimeException thrown in the BankAccount class. The code effectively catches the exception and prints an error message, ensuring the program continues execution even after the exception occurs.</p> |
| 26 | <p><b>Write a java program to generate user defined exception using “throw” and “throws” keyword.</b></p> <p><b>Also Write a java that differentiates checked and unchecked exceptions. (Mention at least two checked and two unchecked exceptions in program).</b></p> <p><b><u>PROGRAM CODE:</u></b></p> <pre> class InvalidAgeException extends Exception {     public InvalidAgeException(String message) { </pre>  |

```
        super(message);
    }
}

class DivisionByZeroException extends RuntimeException {
    public DivisionByZeroException(String message) {
        super(message);
    }
}

public class P26 {

    public static void checkAge(int age) throws InvalidAgeException {
        if (age < 18) {
            throw new InvalidAgeException("Age is less than 18. Access
denied.");
        } else {
            System.out.println("Access granted. Age is: " + age);
        }
    }

    public static void divide(int a, int b) {
        if (b == 0) {
            throw new DivisionByZeroException("Division by zero is not
allowed.");
        } else {
            System.out.println("Result of division is: " + (a / b));
        }
    }

    public static void main(String[] args) {
        try {
            checkAge(16);
        } catch (InvalidAgeException e) {
            System.out.println("Caught the exception: " + e.getMessage());
        }

        try {
            checkAge(20);
        } catch (InvalidAgeException e) {
            System.out.println("Caught the exception: " + e.getMessage());
        }
    }
}
```

```

    try {
        divide(10, 0);
    } catch (DivisionByZeroException e) {
        System.out.println("Caught the exception: " + e.getMessage());
    }

    try {
        divide(10, 2);
    } catch (DivisionByZeroException e) {
        System.out.println("Caught the exception: " + e.getMessage());
    }

    try {
        int[] arr = new int[2];
        int x = arr[5];
    } catch (ArrayIndexOutOfBoundsException e) {
        System.out.println("Caught ArrayIndexOutOfBoundsException: "
+ e.getMessage());
    }
}

```

**OUTPUT:**

```

) { java P26 }
Caught the exception: Age is less than 18. Access denied.
Access granted. Age is: 20
Caught the exception: Division by zero is not allowed.
Result of division is: 5
Caught ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 2

```

**CONCLUSION:**

The provided Java code demonstrates the creation and handling of user-defined exceptions using throw and throws keywords. It also differentiates between checked and unchecked exceptions by using InvalidAgeException (checked) and DivisionByZeroException (unchecked). The code effectively handles the exceptions and provides appropriate error messages.

## 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.

**Program:**

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

public class P27{
    public static void main(String[] args) {

        if (args.length == 0) {
            System.out.println("No files specified.");
            return;
        }

        for (String fileName : args) {
            int lineCount = 0;

            try (BufferedReader reader = new BufferedReader(new
FileReader(fileName))) {
                while (reader.readLine() != null) {
                    lineCount++;
                }
                System.out.println(fileName + ": " + lineCount + " lines");
            } catch (IOException e) {

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

**OUTPUT:**

```
file1.txt: 3 lines  
file2.txt: 2 lines  
file3.txt: 0 lines
```

**CONCLUSION:**

This Java program effectively counts lines in multiple text files specified on the command line, handling file-not-found errors and providing a clear output for each file's line count.

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

**Program:**

```
import java.io.FileReader;  
import java.io.IOException;  
  
public class P28 {  
    public static void main(String[] args) {  
  
        if (args.length != 2) {  
            System.out.println("Usage: java P28 <character> <filename>");  
            return;  
        }  
  
        char targetChar = args[0].charAt(0);  
        String fileName = args[1];  
  
        int charCount = 0;  
  
        try (FileReader reader = new FileReader(fileName)) {  
            int currentChar;  
            while ((currentChar = reader.read()) != -1) {  
                if (currentChar == targetChar) {  
                    charCount++;  
                }  
            }  
            System.out.println("The character '" + targetChar + "' appears " +  
charCount + " times in " + fileName);  
        }  
    }  
}
```



```

    } catch (IOException e) {
        System.out.println("Error reading file: " + fileName);
    }
}
}

```

### OUTPUT:

```

PS C:\Users\kashi\OneDrive\Desktop\vs\java_prc\set6> cd "c:\Users\kashi\OneDrive\Desktop\vs\java_prc\set6"
Usage: java P28 <character> <filename>
PS C:\Users\kashi\OneDrive\Desktop\vs\java_prc\set6> java P28 e xanadu.txt
The character 'e' appears 2 times in xanadu.txt
PS C:\Users\kashi\OneDrive\Desktop\vs\java_prc\set6>

```

### CONCLUSION:

Java program efficiently counts the occurrences of a specified character in a file, demonstrating practical text analysis and command-line argument handling capabilities.

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

#### Program:

```

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

public class P29_1 {
    public static void main(String[] args) {
        if (args.length != 2) {
            System.out.println("Usage: java P29_1 <word> <filename>");
            return;
        }

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

        int lineNumber = 0;
        boolean found = false;

        try (BufferedReader reader = new BufferedReader(new
FileReader(fileName))) {

```

```
String line;
while ((line = reader.readLine()) != null) {
    lineNumber++;
    if (line.contains(searchWord)) {
        System.out.println("Word " + searchWord + " found at line "
+ lineNumber + ": " + line);
        found = true;
    }
}
if (!found) {
    System.out.println("Word " + searchWord + " not found in the
file.");
}
} catch (IOException e) {
    System.out.println("Error reading file: " + fileName);
}
}
```

**OUTPUT:**

```
Usage: java P29_1 <word> <filename>
PS C:\Users\kashi\OneDrive\Desktop\vs\java_prc\set6> java P29_1 world xanadu
Word 'world' not found in the file.
PS C:\Users\kashi\OneDrive\Desktop\vs\java_prc\set6> java P29_1 kashish file
Word 'kashish' found at line 1: kashish
PS C:\Users\kashi\OneDrive\Desktop\vs\java_prc\set6> █
```

**Program:**

```
public class P29_2 {
    public static void main(String[] args) {

        int num = 10;

        Integer wrappedNum = num;

        int unwrappedNum = wrappedNum;

        String binaryString = Integer.toBinaryString(num);
        int comparison = Integer.compare(wrappedNum, 20);

        System.out.println("Primitive int: " + num);
```

|      |  |
|------|--|
|      | <pre> System.out.println("Wrapped Integer: " + wrappedNum); System.out.println("Binary representation of " + num + ": " + binaryString); System.out.println("Comparison of " + wrappedNum + " with 20: " + (comparison &lt; 0 ? "Less than 20" : "Greater or equal to 20")); } } </pre> <p><b>OUTPUT:</b></p> <pre> Primitive int: 10 Wrapped Integer: 10 Binary representation of 10: 1010 Comparison of 10 with 20: Less than 20 PS C:\Users\kashi\OneDrive\Desktop\vs\java_prc\set6&gt; </pre> <p><b>CONCLUSION:</b></p> <p>This Java program demonstrates file word search and showcases wrapper class usage, highlighting autoboxing and unboxing features, enabling seamless conversions between primitive types and object references.</p>  |
| 30 . | <p><b>Write a program to copy data from one file to another file. If the destination file does not exist, it is created automatically.</b></p> <p><b>Program:</b></p> <pre> import java.io.FileReader; import java.io.FileWriter; import java.io.IOException;  public class P30 {     public static void main(String[] args) {          if (args.length != 2) {             System.out.println("Usage: java P30 &lt;source file&gt; &lt;destination file&gt;");             return;         }          String sourceFile = args[0];         String destinationFile = args[1];          try (FileReader reader = new FileReader(sourceFile); FileWriter writer = new FileWriter(destinationFile)) {             int character;             while ((character = reader.read()) != -1) { </pre> |

```

        writer.write(character);
    }
    System.out.println("File copied successfully from " + sourceFile +
" to " + destinationFile);
} catch (IOException e) {
    System.out.println("Error while copying file: " + e.getMessage());
}
}
}

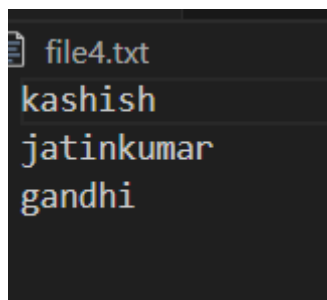
```

**OUTPUT:**

```

Usage: java P30 <source file> <destination file>
PS C:\Users\kashi\OneDrive\Desktop\vs\java_prc\set6> java P30 file1.txt file4.txt
File copied successfully from file1.txt to file4.txt
PS C:\Users\kashi\OneDrive\Desktop\vs\java_prc\set6>

```



```

file4.txt
kashish
jatinkumar
gandhi

```



```

file1.txt
kashish
jatinkumar
gandhi

```

**CONCLUSION:**

This Java program efficiently copies data from a source file to a destination file, automatically creating the latter if it doesn't exist, demonstrating practical file manipulation capabilities.

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

**Program:**

```

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;

```

```

public class P31 {

```

```

public static void main(String[] args) {

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

        System.out.println("Enter text (type 'exit' to stop):");
        String inputLine;

        while (!(inputLine =
consoleReader.readLine()).equalsIgnoreCase("exit")) {
            fileWriter.write(inputLine);
            fileWriter.newLine();
        }
        System.out.println("Text written to output.txt successfully.");
    } catch (IOException e) {
        System.out.println("Error: " + e.getMessage());
    }

    try (FileInputStream byteReader = new
FileInputStream("output.txt");
        FileOutputStream byteWriter = new
FileOutputStream("output_bytes.dat")) {

        byte[] buffer = new byte[1024];
        int bytesRead;
        while ((bytesRead = byteReader.read(buffer)) != -1) {
            byteWriter.write(buffer, 0, bytesRead);
        }
        System.out.println("Byte data written to output_bytes.dat
successfully.");
    } catch (IOException e) {
        System.out.println("Error: " + e.getMessage());
    }
}

```

**OUTPUT:**

```
Enter text (type 'exit' to stop):
kashish gandhi
java
c++
exit
Text written to output.txt successfully.
Byte data written to output_bytes.dat successfully.
PS C:\Users\kashi\OneDrive\Desktop\vs\java_prc\set6>
```

```
set6 > output.txt
1 kashish gandhi
2 java
3 c++
4
```

**CONCLUSION:**

This Java program demonstrates byte and character streams, and utilizes `BufferedReader/BufferedWriter` to efficiently read console input and write to files, showcasing fundamental I/O operations and stream manipulation.

**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.**

**PROGRAM CODE:**

```
public class P32 {

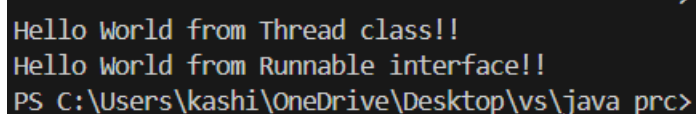
    static class P32B extends Thread {
        @Override
        public void run() {
            System.out.println("Hello World from Thread class!!");
        }
    }

    static class P32A implements Runnable {
        @Override
        public void run() {
            System.out.println("Hello World from Runnable interface!!");
        }
    }

    public static void main(String[] args) {

        P32B thread1 = new P32B();
        thread1.start();

        P32A t1 = new P32A();
        Thread thread2 = new Thread(t1);
        thread2.start();
    }
}
```

**OUTPUT:**

```
Hello world from Thread class!!
Hello world from Runnable interface!!
PS C:\Users\kashi\OneDrive\Desktop\vs\java prc>
```

**CONCLUSION:**

|     |   |
|-----|---|
|     | <p>The program demonstrates creating threads using both the 'Thread' class and 'Runnable' interface, showing flexibility in Java multithreading. Both approaches successfully execute tasks concurrently and handle interruptions.</p>  |
| 33. | <p><b>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.</b></p> <p><b><u>PROGRAM CODE:</u></b></p> <pre> class P33 {      static final int MAX = 5;     static final int MAX_THREAD = 4;      static int[] a = { 1, 5, 7, 10,11 };     static int[] sum = new int[MAX_THREAD];     static int part = 0;      static class SumArray implements Runnable {          @Override         public void run() {              int thread_part = part++;              for (int i = thread_part * (MAX / 4); i &lt; (thread_part + 1) * (MAX / 4); i++) {                 sum[thread_part] += a[i];                 System.out.println(sum[i]);             }         }     }      public static void main(String[] args) throws InterruptedException {          Thread[] threads = new Thread[MAX_THREAD];          // Creating 4 threads         for (int i = 0; i &lt; MAX_THREAD; i++) {             threads[i] = new Thread(new SumArray());             threads[i].start();         }     } } </pre> |



```

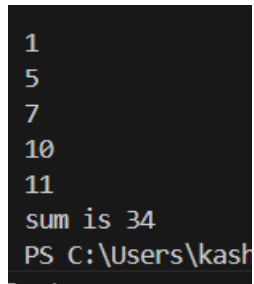
    }

    // Joining 4 threads i.e. waiting for all 4 threads to complete
    for (int i = 0; i < MAX_THREAD; i++) {
        threads[i].join();
    }

    // Adding sum of all 4 parts
    int total_sum = 0;
    for (int i = 0; i < MAX_THREAD; i++) {
        total_sum += sum[i];
    }

    System.out.println("sum is " + total_sum);
}
}

```

**OUTPUT:**


```

1
5
7
10
11
sum is 34
PS C:\Users\kash

```

**CONCLUSION:**

The program divides the summation of an array among multiple threads, efficiently distributing the workload. After each thread computes its portion, the total sum is calculated and displayed.

- 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 NumberGenerator extends Thread{
    private static final boolean True = false;
    public static int num;
    public void run(){
        Random random=new Random();
        while (true) {

```

```
num = random.nextInt(100); // Generate random number between
0 and 99
System.out.println("Generated number: " + num);

try {
    Thread.sleep(1000); // Wait for 1 second
} catch (InterruptedException e) {
    System.out.println("Thread interrupted");
}

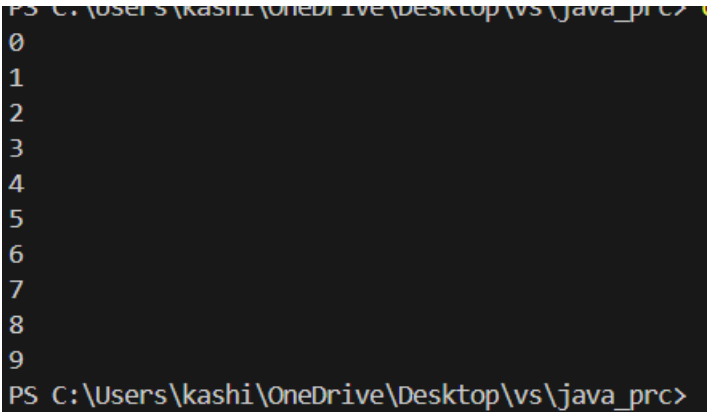
}
}
}
```

**OUTPUT:**

```
java P34 }
Generated number: 44
Square of 0 is: 0
Square of 44 is: 1936
Generated number: 93
Cube of 93is :804357
Generated number: 19
Generated number: 48
Square of 48 is: 2304
Generated number: 49
Square of 48 is: 2304
Generated number: 52
Square of 52 is: 2704
Generated number: 60
Square of 60 is: 3600
Generated number: 31
Generated number: 64
Generated number: 11
Cube of 11is :1331
Cube of 11is :1331
Generated number: 53
Generated number: 27
Cube of 53is :148877
Cube of 27is :19683
Generated number: 39
Generated number: 0
Square of 0 is: 0
Square of 0 is: 0
Generated number: 8
Square of 8 is: 64
```

**CONCLUSION:**

The program continuously generates random integers using a thread. Depending on whether the number is even or odd, other threads will

|     |   |
|-----|---|
|     | either compute the square (for even numbers) or the cube (for odd numbers), effectively demonstrating multithreading for parallel tasks.  |
| 35. | <p><b>Write a program to increment the value of one variable by one and display it after one second using thread using sleep() method.</b></p> <p><b><u>PROGRAM CODE:</u></b></p> <pre>import java.util.*; public class P35 implements Runnable {     public static void main(String args[]){         P35 t1=new P35();         Thread t=new Thread(t1);         t.start();     }      @Override     public void run() {         for(int i=0;i&lt;10;i++){             System.out.println(i);             try {                 Thread.sleep(1000); // Sleep for 2 seconds             } catch (InterruptedException e) {                 System.out.println("Thread was interrupted.");             }         }     } }</pre> <p><b><u>OUTPUT:</u></b></p>  <p><b><u>CONCLUSION:</u></b></p> <p>The program demonstrates how to increment and display a variable's value using a thread. It pauses for one second between increments by utilizing the Thread.sleep() method, simulating a timed interval in multithreading</p> |
| 36. | <p><b>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.</b></p>   |

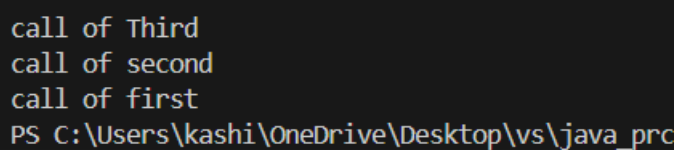
**PROGRAM CODE:**

```
import java.util.*;

public class P36 extends Thread {
    P36(String a){
        super(a);
    }
    public void run(){

        try {
            Thread.sleep(1000);
            System.out.println("call of "+ this.getName());
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}

public static void main(String[] args) {
    P36 p1= new P36("first");
    P36 p2= new P36("second");
    P36 p3= new P36("Third");
    p1.setPriority(3);
    p2.setPriority(5);
    p3.setPriority(7);
    p1.start();
    p2.start();
    p3.start();
}
}
```

**OUTPUT:**

```
call of Third
call of second
call of first
PS C:\Users\kashi\OneDrive\Desktop\vs\java_pro
```

**CONCLUSION:**

This program creates three threads with different priority levels: FIRST (priority 3), SECOND (default priority 5), and THIRD (priority 7). The threads are executed in parallel, and their priorities help the scheduler determine the order in which they run, although exact execution order may vary based on the underlying system.

## PART-VIII Collection Framework and Generic

**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:**

```
import java.util.ArrayList;
```

```
public class P38 {
    private ArrayList<Object> stackList;
```

```
    public P38() {
        stackList = new ArrayList<>();
    }
```

```
    public boolean isEmpty() {
        return stackList.isEmpty();
    }
```

```
    public int getSize() {
        return stackList.size();
    }
```

```
    public Object peek() {
        if (isEmpty()) {
            throw new IllegalStateException("Stack is empty");
        }
        return stackList.get(getSize() - 1);
    }
```

```
    public Object pop() {
        if (isEmpty()) {
            throw new IllegalStateException("Stack is empty");
        }
        return stackList.remove(getSize() - 1);
    }
```

```
}

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

public static void main(String[] args) {
    P38 stack = new P38();

    stack.push("Hello");
    stack.push(123);
    stack.push(45.67);

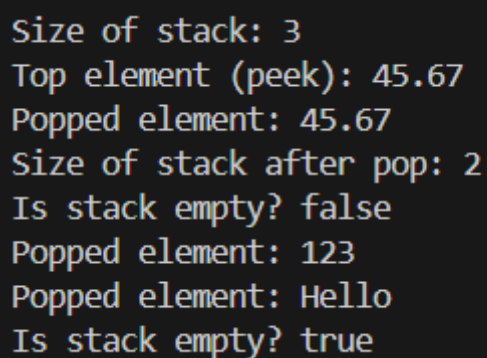
    System.out.println("Size of stack: " + stack.getSize());
    System.out.println("Top element (peek): " + stack.peek());

    System.out.println("Popped element: " + stack.pop());
    System.out.println("Size of stack after pop: " + stack.getSize());

    System.out.println("Is stack empty? " + stack.isEmpty());

    System.out.println("Popped element: " + stack.pop());
    System.out.println("Popped element: " + stack.pop());

    System.out.println("Is stack empty? " + stack.isEmpty());
}
}
```

**OUTPUT:**A screenshot of a terminal window with a black background and white text. The output shows the execution of a Java program that uses a stack. The lines of output are: 'Size of stack: 3', 'Top element (peek): 45.67', 'Popped element: 45.67', 'Size of stack after pop: 2', 'Is stack empty? false', 'Popped element: 123', 'Popped element: Hello', and 'Is stack empty? true'.

```
Size of stack: 3
Top element (peek): 45.67
Popped element: 45.67
Size of stack after pop: 2
Is stack empty? false
Popped element: 123
Popped element: Hello
Is stack empty? true
```

**CONCLUSION:**

|      |   |
|------|---|
|      | <p>This Java program implements a custom stack using ArrayList, providing essential stack operations like push, pop, peek, isEmpty, and getSize, demonstrating efficient and organized data structure management.</p>   |
| 39 . | <p><b>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.</b></p> <p><b>Program:</b></p> <pre>import java.util.Arrays;  public class P39 {      public static &lt;T extends Comparable&lt;T&gt;&gt; void sort(T[] array) {         int n = array.length;         for (int i = 0; i &lt; n - 1; i++) {             for (int j = 0; j &lt; n - i - 1; j++) {                 if (array[j].compareTo(array[j + 1]) &gt; 0) {                     T temp = array[j];                     array[j] = array[j + 1];                     array[j + 1] = temp;                 }             }         }     }      static class Product implements Comparable&lt;Product&gt; {         String name;         double price;         double rating;          public Product(String name, double price, double rating) {             this.name = name;             this.price = price;             this.rating = rating;         }     } }</pre> |

```

@Override
public int compareTo(Product other) {
    return Double.compare(this.price, other.price);
}

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

public static void main(String[] args) {
    Product[] products = {
        new Product("Laptop", 999.99, 4.5),
        new Product("Smartphone", 699.99, 4.7),
        new Product("Tablet", 299.99, 4.2),
        new Product("Monitor", 199.99, 4.6)
    };

    System.out.println("Before sorting: " + Arrays.toString(products));
    System.out.println();

    sort(products);

    System.out.println("After sorting: " + Arrays.toString(products));
}
}

```

### OUTPUT:

```

Before sorting: [Product{name='Laptop', price=999.99, rating=4.5}, Product{name='Smartphone', price=699.99, rating=4.7}, Product{name='Tablet', price=299.99, rating=4.2}, Product{name='Monitor', price=199.99, rating=4.6}]

After sorting: [Product{name='Monitor', price=199.99, rating=4.6}, Product{name='Tablet', price=299.99, rating=4.2}, Product{name='Smartphone', price=699.99, rating=4.7}, Product{name='Laptop', price=999.99, rating=4.5}]
PS C:\Users\kashi\OneDrive\Desktop\vs\java prc\set8> |

```

### CONCLUSION:

This Java method leverages generics and the Comparable interface to sort arrays of diverse object types, ensuring flexibility, reusability, and efficient sorting capabilities for e-commerce applications.

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



**Program:**

```
import java.util.Map;
import java.util.TreeMap;
import java.util.Set;
import java.util.Scanner;

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 inputText = scanner.nextLine();

        String[] words = inputText.toLowerCase().split("[\\W]+");

        for (String word : words) {
            if (wordMap.containsKey(word)) {
                wordMap.put(word, wordMap.get(word) + 1);
            } else {
                wordMap.put(word, 1);
            }
        }

        Set<Map.Entry<String, Integer>> entrySet = wordMap.entrySet();
        System.out.println("\nWord Count:");
        for (Map.Entry<String, Integer> entry : entrySet) {
            System.out.println(entry.getKey() + ": " + entry.getValue());
        }

        scanner.close();
    }
}
```

**OUTPUT:**

```
Enter a text:
Kashish
```

```
Word Count:
kashish: 1
```

### **CONCLUSION:**

This Java program efficiently counts word occurrences in text using HashMap and TreeMap, demonstrating effective word counting and sorting capabilities, and showcasing Java's robust Map and Set classes."

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

#### **Program:**

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.HashSet;
import java.util.Scanner;

public class P41 {

    public static void main(String[] args) {

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

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

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

```
System.out.print("Enter the path of the Java source file: ");
String filePath = scanner.nextLine();

int keywordCount = 0;

try {
    Scanner fileScanner = new Scanner(new File(filePath));

    while (fileScanner.hasNext()) {
        String word = fileScanner.next();

        if (javaKeywords.contains(word)) {
            keywordCount++;
        }
    }

    fileScanner.close();

} catch (FileNotFoundException e) {
    System.out.println("File not found: " + filePath);
}

System.out.println("Number of Java keywords in the Drivwfile: " +
keywordCount);

scanner.close();
}
```

**OUTPUT:**

```
S C:\Users\kashi\OneDrive\Desktop\vs\java_prc\set8> cd "c:\Users\kashi\OneDrive\Desktop\vs\java_prc\set8"
Enter the path of the Java source file: c:\Users\kashi\OneDrive\desktop\vs\java_prc\set8\P
Number of Java keywords in the Drivwfile: 28
S C:\Users\kashi\OneDrive\Desktop\vs\java_prc\set8> █
```

**CONCLUSION:**

This Java program efficiently counts keywords in a source file using a HashSet, demonstrating effective keyword identification and counting capabilities in Java programming language.

