



Devang Patel Institute of Advance Technology and Research

(A Constitute Institute of CHARUSAT)

Certificate

This is to certify that

Mr. / Mrs. Apshul Jangid

of 3CSE1 Class,

ID. No. 230CS038 has satisfactorily completed

his/ her term work in Java programming for

the ending in november 2024 /2025

Date : 07/10/24

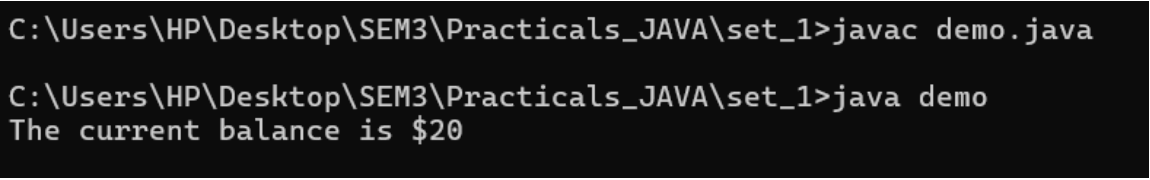
Sign. of Faculty

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

No.	Aim of the Practical
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><u>PROGRAM CODE:</u></p> <pre>class demo { public static void main(String a[]) { int x=20; System.out.println("The current balance is \$" + x); } }</pre> <p><u>OUTPUT:</u></p>  <p><u>CONCLUSION:</u></p> <p>The variable is stored in x and the function println() displays its value.</p>

3.

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

PROGRAM CODE:

```
import java.util.Scanner;

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

        System.out.println("Enter the distance: ");
        float d=s.nextFloat();

        System.out.println("Enter time in hr,min,sec: ");
        float hr=s.nextFloat();
        float min=s.nextFloat();
        float sec=s.nextFloat();

        float t=(hr*60*60)+(min*60)+sec;
        float speed=d/t;
        System.out.println("Speed in m/s is "+speed);

        float sk=speed*(18/5);
        System.out.println("Speed in km/h is "+sk);

        System.out.println("Speed in mi/h is "+(sk/1.609));

    }
}
```

OUTPUT:

```
C:\Users\HP\Desktop\SEM3\Practicals_JAVA\set_1>java speed
Enter the distance:
10000
Enter time in hr,min,sec:
1
10
50
Speed in m/s is 2.3529413
Speed in km/h is 7.0588236
Speed in mi/h is 4.387087374462557
```

CONCLUSION:

We take the user's input for distance (in meters), hours, minutes, and seconds. We calculate the total time in seconds. We compute the speed in meters per second, kilometres per hour, and miles per hour using the given formulas.

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

PROGRAM CODE:

```
import java.util.Scanner;

class budget
{
    public static void main(String args[])
    {
        Scanner s=new Scanner(System.in);
        System.out.println("Enter the number of days you want to enter the
amount for: ");
        int n=s.nextInt();
```

```
float a[]=new float [n];
int i,sum=0;
System.out.println("enter the values: ");
for(i=0;i<n;i++)
{
    a[i]=s.nextFloat();
    sum+=a[i];
}
System.out.println("Sum of the amount is "+sum);
}
}
```

OUTPUT:

```
C:\Users\HP\Desktop\SEM3\Practicals_JAVA\set_1>javac budget.java
C:\Users\HP\Desktop\SEM3\Practicals_JAVA\set_1>java budget
Enter the number of days you want to enter the amount for:
5
enter the values:
2
3
45
67
32
Sum of the amount is 149
C:\Users\HP\Desktop\SEM3\Practicals_JAVA\set_1>
```

CONCLUSION:

This program efficiently calculates the total expenses for a month based on daily inputs using an array. It demonstrates basic array handling and iteration in Java.

Supplementary Experiment:

You are creating a library management system. The library has two separate lists of books for fiction and non-fiction. The system should merge these lists into a single list for inventory purposes. Write a Java program to merge two arrays.

PROGRAM CODE:

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

```
int[] arr1 = { 1, 3, 4, 5, 9};
int[] arr2 = { 2, 4, 6, 8, 7};
int l1 = arr1.length;
int l2 = arr2.length;
int result = l1 + l2;

System.out.println("Array 1: ");
for(int i=0;i<l1;i++)
{
    System.out.print(arr1[i]+" ");
}
System.out.println();

System.out.println("Array 2: ");
for(int i=0;i<l2;i++)
{
    System.out.print(arr2[i]+" ");
}
System.out.println();

int[] mergearray = new int[result];
for(int i=0;i<l1;i++)
{
    mergearray[i]=arr1[i];
    mergearray[i+5]=arr2[i];
}

System.out.println("Merged Array:");
for (int i = 0; i < result; i++)
{
    System.out.print(mergearray[i] + " ");
}
}
```

OUTPUT:

```

C:\Users\HP\Desktop\SEM3\Practicals_JAVA\set_1>javac suppl.java

C:\Users\HP\Desktop\SEM3\Practicals_JAVA\set_1>java suppl
Array 1:
1 3 4 5 9
Array 2:
2 4 6 8 7
Merged Array:
1 3 4 5 9 2 4 6 8 7
C:\Users\HP\Desktop\SEM3\Practicals_JAVA\set_1>

```

CONCLUSION:

Hence, we successfully merged the two arrays.

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

PROGRAM CODE:

```

import java.util.Scanner;

class app
{
    public static void main(String args[])
    {
        Scanner s=new Scanner(System.in);
        System.out.println("Enter the choice of code from the following: ");
        System.out.println("1:Motor");
        System.out.println("2:Fan");
        System.out.println("3:Tube");
        System.out.println("4:Wires");
        System.out.println("5:Others");
        System.out.println("6:Exit");
        int ch=s.nextInt();
        double price=100;
        switch(ch)
        {
            case 1:

```

```
price+=8;
System.out.println("The total amount of motor is "+price);
break;
case 2:
price+=12;
System.out.println("The total amount of fan is "+price);
break;
case 3:
price+=5;
System.out.println("The total amount of tube is "+price);
break;
case 4:
price+=7.5;
System.out.println("The total amount of wires is "+price);
break;
case 5:
price+=3;
System.out.println("The total amount of others is "+price);
break;
default:
break;
    }
}
}
```

OUTPUT:


```

C:\Users\HP\Desktop\SEM3\Practicals_JAVA\set_1>javac app.java

C:\Users\HP\Desktop\SEM3\Practicals_JAVA\set_1>java app
Enter the choice of code from the following:
1:Motor
2:Fan
3:Tube
4:Wires
5:Others
6:Exit
1
The total amount of motor is 108.0

C:\Users\HP\Desktop\SEM3\Practicals_JAVA\set_1>java app
Enter the choice of code from the following:
1:Motor
2:Fan
3:Tube
4:Wires
5:Others
6:Exit
2
The total amount of fan is 112.0

```

CONCLUSION:

This program demonstrates how to use a switch statement in Java to calculate a bill based on product codes and prices, applying specific tax rates depending on the product type.

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.lang.*;
import java.util.Scanner;

class fseries
{
    public static void main(String []args)
    {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter the numer of days you want to work out;

```

```
");  
  
    int n=s.nextInt();  
    int n1=0,n2=1,i,n3;  
    System.out.print(n1+" "+n2);  
    for(i=2;i<n;i++)  
    {  
        n3=n1+n2;  
        System.out.print(" "+n3);  
        n1=n2;  
        n2=n3;  
    }  
}
```

OUTPUT:

```
C:\Users\HP\Desktop\SEM3\Practicals_JAVA\set_1>java fseries  
Enter the number of days you want to work out:  
8  
0 1 1 2 3 5 8 13  
C:\Users\HP\Desktop\SEM3\Practicals_JAVA\set_1>|
```

CONCLUSION:

This program successfully creates and exercise routine based on the Fibonacci series and displays the formatted output to the user.

Supplementary Experiment:

Imagine you are developing a classroom management system. You need to keep track of the grades of students in a class. After collecting the grades, you want to display each student's grade along with a message indicating if they have passed or failed. Let's assume the passing grade is 50.

PROGRAM CODE:

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

```
public class supp2
{
    public static void main(String[] args)
    {
        Scanner scanner = new Scanner(System.in);

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

        int passingGrade = 50;

        for (int i = 1; i <= numStudents; i++)
        {
            System.out.print("Enter grade for student " + i + ": ");
            int studentGrade = scanner.nextInt();

            if (studentGrade >= passingGrade)
            {
                System.out.println("Student " + i + ": Passed");
            }
            else
            {
                System.out.println("Student " + i + ": Failed");
            }
        }
    }
}
```

OUTPUT:

```
C:\Users\HP\Desktop\SEM3\Practicals_JAVA\set_1>javac supp2.java
C:\Users\HP\Desktop\SEM3\Practicals_JAVA\set_1>java supp2
Enter the number of students: 3
Enter grade for student 1: 67
Student 1: Passed
Enter grade for student 2: 45
Student 2: Failed
Enter grade for student 3: 89
Student 3: Passed
C:\Users\HP\Desktop\SEM3\Practicals_JAVA\set_1>
```

CONCLUSION:

This Java program effectively manages and displays students' grades along with their pass/fail status based on a predefined passing grade condition.

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:

```
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

	<p>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 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</p> <ul style="list-style-type: none"> array. array_count9([1, 2, 9]) → 1 array_count9([1, 9, 9]) → 2 array_count9([1, 9, 9, 3, 9]) → 3 <p><u>PROGRAM CODE:</u></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>

OUTPUT:

```
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
```

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:

```
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 Perform following functionalities of the string:

·

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

PROGRAM CODE:

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

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

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

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

    public static String reverseString(String str) {
        StringBuilder sb = new StringBuilder(str);
```



```

        return sb.reverse().toString();
    }

    public static String sortString(String str) {
        char chars[] = str.toCharArray();
        Arrays.sort(chars);
        return new String(chars);
    }

    public static void main(String args[]) {
        String str = "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 short the String and display the output.

11. Perform following Functionalities of the string:

“CHARUSAT UNIVERSITY”

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

PROGRAM CODE:

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

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

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

}}
```

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 - 3

No .	Aim of the Practical
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><u>PROGRAM CODE :</u></p> <pre>import java.util.Scanner; class poundconvert { public static void main(String a[]) { int x=Integer.parseInt(a[0]); int y=x*100; System.out.println("The value is rupees is: "+y); } }</pre> <p><u>OUTPUT:</u></p> <pre>The value is rupees is: 200</pre>

CONCLUSION:

By this code I learnt to use parse integer and to convert pounds into rupees .

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

PROGRAM CODE :

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

```
class Emp
{
    Scanner s=new Scanner(System.in);
    String fs;
    String ls;
    double sal;

    Emp()
    {
    }

    Emp(String f,String l,double s)
    {
        fs=f;
        ls=l;
        sal=s;
    }
}
```

```
}

void setfs()
{
    System.out.println("Enter the first name of employee: ");
    fs=s.nextLine();
}

void setls()
{
    System.out.println("Enter the last name of employee: ");
    ls=s.nextLine();
}

void setsal()
{
    System.out.println("Enter the salary of employee: ");
    sal=s.nextDouble();
    if(sal<0)
    {
        sal=0.0;
    }
}

double newsal()
{
    sal=sal+(0.1*sal);
    System.out.println("The raise in salary is :");
    return sal;
}

String getfs()
{
    return fs;
}

String getls()
{
    return ls;
}
```

```
        double getsal()
        {
            return sal;
        }

    }

class Emptest
{
    public static void main(String a[])
    {
        Emp e1=new Emp();
        Emp e2=new Emp();
        e1.setfs();
        e1.setls();
        e1.setsal();
        e2.setfs();
        e2.setls();
        e2.setsal();
        System.out.println(e1.getfs());
        System.out.println(e1.getls());
        System.out.println(e1.getsal());
        System.out.println(e1.newsal());
        String fname=e2.getfs();
        String lname=e2.getls();
        double salary=e2.getsal();
        System.out.println(fname);
        System.out.println(lname);
        System.out.println(salary);
        System.out.println(e2.newsal());

    }
}
```

OUTPUT:

```
Enter the first name of employee:
James
Enter the last name of employee:
Dhandhukiya
Enter the salary of employee:
5000
Enter the first name of employee:
Evan
Enter the last name of employee:
Gregor
Enter the salary of employee:
10000
James
Dhandhukiya
5000.0
the raise in salary is :
5500.0
Evan
Gregor
10000.0
the raise in salary is :
11000.0
```

CONCLUSION:

By this experiment I learnt that how to use gets and sets function in java and how to call by object.

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.Scanner;

class date
{
    Scanner s=new Scanner(System.in);
    int d,m,y;
    date()
    {}
}
```



```
date(int date,int month,int year)
{
    d=date;
    m=month;
    y=year;
}

void setd()
{
    System.out.println("Enter date: ");
    d=s.nextInt();
}

void setm()
{
    System.out.println("Enter month: ");
    m=s.nextInt();
}

void sety()
{
    System.out.println("Enter year: ");
    y=s.nextInt();
}

int getd()
{
    return d;
}

int getm()
{
    return m;
}

int gety()
{
    return y;
}

void displayDate()
```

```
        {  
            System.out.println(d+"/"+m+"/"+y);  
        }  
    }  
  
class dateTest  
{  
    public static void main(String a[])  
    {  
        date d1=new date();  
        d1.setd();  
        d1.setm();  
        d1.sety();  
        d1.displayDate();  
    }  
}
```

OUTPUT:

```
Enter date:  
09  
Enter month:  
11  
Enter year:  
2005  
9/11/2005
```

CONCLUSION:

By this experiment I learnt how to use constructor ,methods classes and objects in java etc.

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

PROGRAM:

```
import java.util.Scanner;

class area
{
    Scanner s=new Scanner(System.in);
    int l,b;
    area()
    {}
    area(int len,int bred)
    {
        l=len;
        b=bred;
    }
    int returnarea()
    {
        return l*b;
    }
}

class rectTest
{
    public static void main(String a[])
    {
        rect r1=new rect(5,7);
        System.out.println("The area is: "+r1.area());
    }
}
```

OUTPUT:

```
The area is: 35
```

CONCLUSION:

By this experiment I learnt how to use default constructor and paramitarized constructor in java.

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

PROGRAM:

```
import java.util.Scanner;

class complex
{
    Scanner s=new Scanner(System.in);
    int r,i;

    void readdata()
    {
        System.out.println("Enter r: ");
        r=s.nextInt();
        System.out.println("Enter i: ");
        i=s.nextInt();
    }
    void sum(complex c)
    {
        int sumr=r+c.r;
        int sumi=i+c.i;
        System.out.println("The sum is: "+sumr+" "+sumi+"i");
    }
}
```

```
}
void sub(complex c)
{
    int subr=r-c.r;
    int subi=i-c.i;
    System.out.println("The difference is: "+subr+" "+subi+"i");
}
void mul(complex c)
{
    int mulr=r*c.r;
    int muli=i*c.i;
    System.out.println("The product is: "+mulr+" "+muli+"i");
}
}

class complexTest
{
    public static void main(String a[])
    {

        complex c1=new complex();
        complex c2=new complex();
        c1.readdata();
        c2.readdata();
        c1.sum(c2);
        c1.sub(c2);
        c1.mul(c2);
    }
}
```

OUTPUT:

```
Enter r:
5
Enter i:
5
Enter r:
3
Enter i:
3
The sum is: 8+8i
The difference is: 2+2i
The product is: 15+15i
```

CONCLUSION:

This Java code provides a user-friendly way to work with complex numbers. You can enter two complex numbers, and the program calculates their sum, difference, and product, presenting the results in a clear format.

Part - 4

No	Aim of the Practical
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><u>PROGRAM CODE:</u></p> <pre> class parent { public void displayparent() { System.out.println("this is parent class"); } } class child extends parent { public void displaychild() { System.out.println("this is child class"); } } public class Practical17 { public static void main(String[] args) { parent p1=new parent(); child c1= new child(); System.out.println("calling parent class method"); </pre>

```

        p1.displayparent();

        System.out.println("calling inherited
method from child class");

        c1.displayparent();

        System.out.println("calling child class
method");

        c1.displaychild();

    }

}

```

OUTPUT:

```

calling parent class method
this is parent class
calling inherited method from child class
this is parent class
calling child class method
this is child class

```

CONCLUSION:

By performing this experiment we learnt how to use the concept of inheritance in java language. It uses the extends keyword to inherit a base class into a derived class, and the object of the derived class can access all the methods and data of the base class provided that they are inherited publicly or protected.

18. Create a class named 'Member' having the following members:

Data members

- 1 - Name
- 2 - Age
- 3 - Phone number
- 4 - Address
- 5 – Salary

It also has a method named 'printSalary' which prints the salary of the members. Two classes 'Employee' and 'Manager' inherits the 'Member' class. The 'Employee' and 'Manager' classes have data members 'specialization' and 'department' respectively. Now, assign name, age, phone number, address and salary to an employee and a manager by making an object of both of these classes and print the same.

PROGRAM:

```
import java.util.*;
```

```
class Member {
```

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

```
    int age;
```

```
    int phone;
```

```
    String name;
```

```
    String address;
```

```
    int salary;
```

```
    public void printSalary() {
```

```
        System.out.println("The salary of the  
member is" + this.salary);
```

```
    }
```

```
    public void getdata() {
```

```
        System.out.println("enter the age of  
member");
```

```
        this.age=sc.nextInt();
```

```
        System.out.println("enter the Name of
```

```
member");

    this.name=sc.next();

    System.out.println("enter the phone
number of member");

    this.phone=sc.nextInt();

    System.out.println("enter the address of
member");

    this.address=sc.next();

    System.out.println("enter the salary of
member");

    this.salary=sc.nextInt();
}

public void printdata() {

    System.out.println("The name of the
member is " + this.name);

    System.out.println("The age of the
member is " + this.age);

    System.out.println("The phone number of
the member is " + this.phone);

    System.out.println("The address of the
```

```
member is " + this.address);
```

```
        System.out.println("The salary of the  
member is "+this.salary);
```

```
    }
```

```
}
```

```
class employee extends Member {
```

```
    String specialization;
```

```
}
```

```
class manager extends Member {
```

```
    // String specialization;
```

```
    String department;
```

```
}
```

```
public class Practical18 {
```

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

```
        employee e1 = new employee();
```

```
        manager m1 = new manager();
```

```
        e1.getdata();
```

```
        e1.printdata();
```

```
        m1.getdata();
```

```
        m1.printdata();
```

```
    }
```

```
}
```

OUTPUT:

```

enter the age of member
21
enter the Name of member
vansh
enter the phone number of member
234123
enter the address of member
anand
enter the salary of member
20000
The name of the member is vansh
The age of the member is 21
The phone number of the member is 234123
The address of the member is anand
The salary of the member is 20000
enter the age of member
23
enter the Name of member
dhairya
enter the phone number of member
95100
enter the address of member
bombay
enter the salary of member
30000

```

CONCLUSION:

By performing this experiment we were able to create constructors and methods to input data for data members of derived class and base class. And further we can extend by creating a method which can display the data.

- 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:

```

class Rectangle {
    int length, breadth;

    public Rectangle(int l, int b) {
        this.length = l;
        this.breadth = b;
    }

    public void area() {
        int area = this.length * this.breadth;
        System.out.println("the area is " + area);
    }
}

```

```

        public void perimeter() {
            int perimeter = 2 * (this.length + this.breadth);
            System.out.println("the perimeter is " + perimeter);
        }
    }

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

public class Practical19 {
    public static void main(String[] args) {
        Square[] s1 = new Square[2];

        s1[0] = new Square(10);

        s1[1] = new Square(5);

        s1[0].area();
        s1[0].perimeter();

        s1[1].area();
        s1[1].perimeter();

    }
}

```

OUTPUT:

```

the area is 100
the perimeter is 40
the area is 25
the perimeter is 20

```

CONCLUSION:

By this experiment we learned how constructors work in inheritance and how they are called. They are called by using the super keyword and passing the parameter given to the constructor of the derived class to the constructor of the base class.

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

PROGRAM:

```
class Shape {
    public void displayshape(){
        System.out.println("This is Shape");
    }
}

class Rectangle extends Shape {
    public void displayRectangle() {
        System.out.println("this is Rectangular shape");
    }
}

class Circle extends Shape {
    public void displaycircle() {
        System.out.println("this is circular shape");
    }
}

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

public class Practical20 {
    public static void main(String[] args) {
        Square s1 = new Square();
        s1.displayRectangle();
        s1.displayshape();
    }
}
```

OUTPUT:

```
this is Rectangular shape  
this is Shape
```

CONCLUSION:

By performing this program we learnt how to perform hierarchal inheritance in java.

21

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

PROGRAM:

```
class Degree {  
    public void getDegree() {  
        System.out.println("i got a degree");  
    }  
}  
  
class Undergraduate extends Degree {  
    public void getDegree() {  
        System.out.println("i am undergraduate");  
    }  
}  
  
class Postgraduate extends Degree {  
    public void getDegree() {  
        System.out.println("i am postgraduate");  
    }  
}  
  
public class Practical21 {  
    public static void main(String[] args) {  
        Degree d1 = new Degree();  
        Undergraduate u1 = new Undergraduate();  
        Postgraduate p1 = new Postgraduate();  
  
        d1.getDegree();  
  
        u1.getDegree();  
        p1.getDegree();  
    }  
}
```

```
}  
}
```

OUTPUT:

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

CONCLUSION:

In this practical we see another example of inheritance.

22

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

`divisorSum` function just takes an integer as input and return the sum of all its divisors.

For example, divisors of 6 are 1, 2, 3 and 6, so `divisor_sum` should return 12. The value of `n` will be at

most 1000.

PROGRAM:

```
import java.util.*;  
import java.io.*;
```

```
interface AdvancedArithmetic{  
    int divisor_sum(int n);  
  
}
```

class called `MyCalculator` implements `AdvancedArithmetic`{

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



```

}
public class Practical22 {
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        calledMyCalulator s = new calledMyCalulator();
        System.out.println("Enter the number: ");
        int n = sc.nextInt();
        System.out.println("The sum of the divisors of the number is: " + s.divisor_sum(n));

    }
}

```

OUTPUT:

```

Enter the number:
2
The sum of the divisors of the number is: 3

```

CONCLUSION:

By this experiment I learnt how to use

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

PROGRAM:

```

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

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

```

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

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

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

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

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

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

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

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

```

}

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

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

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

// Main.java
public class 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:

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

CONCLUSION:

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

Part – 5 Exception Handling

No.	Aim of the Practical
1.	<p>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.</p> <p><u>PROGRAM CODE: PROGRAM CODE :</u></p> <pre> public class ErrorsNExceptions{ public static void main(String[] args) { int a=5, b=0, c; // System.out.println("Try block will run"); try { c=a/b; System.out.println(c); } catch (Exception e) { System.out.println(e.toString()); //Method 1 // e.getMessage(); //Method 2 // e.printStackTrace(); //Method 3 // System.out.println("Catch block successfully executed"); } } } </pre> <p><u>OUTPUT:</u></p> <div style="background-color: black; color: white; padding: 5px;"> Name: Krishil Agrawal ID: 23DCS001 </div>

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

PROGRAM CODE :

```
public class ErrorNException2 {
    public static void main(String[] args) {
        int a[] = {1,2,3};
        String s = "Charusat";
        String s1 = null;
        int r;
        try {
            try {
                r=a[0]/0;
            } catch (ArithmeticException e) {
                System.out.println("Hello 1");
                e.printStackTrace();
            }
            try {
                System.out.println(a[3]);
            } catch
(ArrayIndexOutOfBoundsException e) {
                System.out.println("Hello 2");
                e.printStackTrace();
            }
            try {
                System.out.println(s1.length());
            } catch (NullPointerException e) {
                System.out.println("Hello 3");
                e.printStackTrace();
            }
            System.out.println(s.charAt(9));
        } catch
(StringIndexOutOfBoundsException e) {
            System.out.println("Hello 4");
            e.printStackTrace();
        }
    }
}
```

```
}  
}
```

OUTPUT:

```
● Exception Handling\" ; if ($?) { javac ErrorNException2.java } ; if ($?) { java ErrorNException2 }  
Hello 1  
java.lang.ArithmeticException: / by zero  
    at ErrorNException2.main(ErrorNException2.java:13)  
Hello 2  
java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3  
    at ErrorNException2.main(ErrorNException2.java:19)  
Hello 3  
java.lang.NullPointerException: Cannot invoke \"String.length()\" because \"<local3>\" is null  
    at ErrorNException2.main(ErrorNException2.java:25)  
Hello 4  
java.lang.StringIndexOutOfBoundsException: Index 9 out of bounds for length 8  
    at java.base/jdk.internal.util.Preconditions$1.apply(Preconditions.java:55)  
    at java.base/jdk.internal.util.Preconditions$1.apply(Preconditions.java:52)  
    at java.base/jdk.internal.util.Preconditions$4.apply(Preconditions.java:213)  
    at java.base/jdk.internal.util.Preconditions$4.apply(Preconditions.java:210)  
    at java.base/jdk.internal.util.Preconditions.outOfBounds(Preconditions.java:98)  
    at java.base/jdk.internal.util.Preconditions.outOfBoundsCheckIndex(Preconditions.java:106)  
    at java.base/jdk.internal.util.Preconditions.checkIndex(Preconditions.java:302)  
    at java.base/java.lang.String.checkIndex(String.java:4881)  
    at java.base/java.lang.StringLatin1.charAt(StringLatin1.java:46)  
    at java.base/java.lang.String.charAt(String.java:1582)  
    at ErrorNException2.main(ErrorNException2.java:30)
```

3. Write a java program to generate user defined exception using “throw” and “throws” keyword. Also Write a java that differentiates checked and unchecked exceptions. (Mention at least two checked and two unchecked exceptions in program).

PROGRAM CODE :

```
class MyException extends Exception {
    MyException(String s){super(s);}
}

public class Throw{
    public static void main(String[] args){
        // System.out.println("user define
message");
        try {
            // throw new MyException("Java user
defined exception");
            // throw new Exception();
            throw new Exception("Base exception
class message");
        } catch (Exception e) {
            System.out.println(e.getMessage());
            System.out.println("Exception Caught
Successfully");
        }
    }
}

public class Throw2 {
    public static void main(String[] args) {
        System.out.println("User Defined Exception");
        throw new ArithmeticException("Charusat/by zero");
        // System.out.println("Rest of the code");
        System.out.println("Name: Krishil Agrawal\nID: 23DCS001");
    }
}
```


OUTPUT:

```
Base exception class message  
Exception Caught Successfully
```

CONCLUSION:

This Java program calculates and displays the speed of an object in meters per second (m/s), kilometers per hour (km/h), and miles per hour (mph) based on user input for distance in meters and time in hours, minutes, and seconds.

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);
        }
    }
}
```

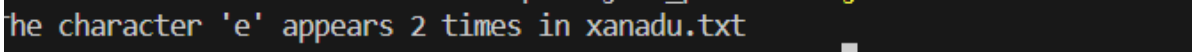
OUTPUT:

```
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 .	<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>Program:</p> <pre>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) {</pre>

	<pre> 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); } } } } </pre> <p>OUTPUT:</p>  <p><u>CONCLUSION:</u></p> <p>Java program efficiently counts the occurrences of a specified character in a file, demonstrating practical text analysis and command-line argument handling capabilities.</p>
29	<p>Write a Java Program to Search for a given word in a File. Also show use of Wrapper Class with an example.</p> <p>Program:</p> <pre> import java.io.BufferedReader; import java.io.FileReader; import java.io.IOException; </pre>

```
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;  
                }  
            }  
        }  
    }  
}
```

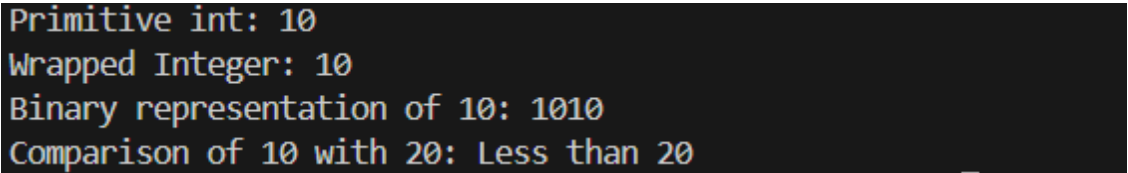
```
    }  
    if (!found) {  
        System.out.println("Word '" + searchWord + "' not found in the  
file.");  
    }  
} catch (IOException e) {  
    System.out.println("Error reading file: " + fileName);  
}  
}  
}
```

OUTPUT:

```
Word 'world' not found in the file.
```

Program:

```
public class P29_2 {  
    public static void main(String[] args) {  
  
        int num = 10;  
  
        Integer wrappedNum = num;  
  
        int unwrappedNum = wrappedNum;
```

	<pre>String binaryString = Integer.toBinaryString(num); int comparison = Integer.compare(wrappedNum, 20); System.out.println("Primitive int: " + num); System.out.println("Wrapped Integer: " + wrappedNum); System.out.println("Binary representation of " + num + ": " + binaryString); System.out.println("Comparison of " + wrappedNum + " with 20: " + (comparison < 0 ? "Less than 20" : "Greater or equal to 20")); } }</pre> <p>OUTPUT:</p>  <p><u>CONCLUSION:</u></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>Write a program to copy data from one file to another file. If the destination file does not exist, it is created automatically.</p> <p>Program:</p>


```
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 <source file> <destination
file>");
            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) {
                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());
        }
    }
}
```

	<pre> } } } </pre> <p>OUTPUT:</p> <div style="background-color: #2e2e2e; color: #eeeeec; padding: 5px; margin: 10px 0;">Usage: java P30 <source file> <destination file></div> <p><u>CONCLUSION:</u></p> <p>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.</p>
31 .	<p>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.</p> <p>Program:</p> <pre> 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 { </pre>

```
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.
```

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 – 7 Multithreading

No.	Aim of the Practical
1.	<p>Write a program to create thread which display “Hello World” message. A. by Thread class B. by using Runnable interface</p> <p><u>PROGRAM CODE</u></p> <pre>// Extending the Thread class class Multithreading1 extends Thread { public void run() { System.out.println("Hello World"); } public static void main(String[] args) { // Create a new thread Multithreading1 thread = new Multithreading1(); // Start the thread thread.start(); } }</pre> <p><u>OUTPUT:</u></p> <div style="background-color: black; color: white; padding: 5px; text-align: center;">Hello World</div>

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

PROGRAM CODE :

```
class SumTask implements Runnable {
    private int start;
    private int end;
    private int[] result;
    private int index;

    public SumTask(int start, int end, int[] result,
int index) {
        this.start = start;
        this.end = end;
        this.result = result;
        this.index = index;
    }

    @Override
    public void run() {
        int sum = 0;
        for (int i = start; i <= end; i++) {
            sum += i;
        }
        result[index] = sum; // Store the partial
sum in the result array
    }
}

public class SumWithThreads {
    public static void main(String[] args) {
        if (args.length < 2) {
            System.out.println("Please provide two
```

```
arguments: N and the number of threads.");
    return;
}

int N = Integer.parseInt(args[0]);
int numThreads =
Integer.parseInt(args[1]);

// Array to hold the partial results
int[] result = new int[numThreads];

// Calculate the range for each thread
int range = N / numThreads;
int remainder = N % numThreads;

Thread[] threads = new
Thread[numThreads];

int start = 1;
for (int i = 0; i < numThreads; i++) {
    int end = start + range - 1;

    if (i == numThreads - 1) {
        end += remainder; // Add the
remainder to the last thread's range
    }

    threads[i] = new Thread(new
SumTask(start, end, result, i));
    threads[i].start();

    start = end + 1;
}

// Wait for all threads to finish
try {
```

```

        for (Thread thread : threads) {
            thread.join();
        }
    } catch (InterruptedException e) {
        System.out.println("Thread interrupted:
" + e.getMessage());
    }

    // Calculate the final sum
    int finalSum = 0;
    for (int sum : result) {
        finalSum += sum;
    }

    // Display the final result
    System.out.println("The sum of the first "
+ N + " numbers is: " + finalSum);

}
}

```

OUTPUT:

```

Hello 1
java.lang.ArithmeticException: / by zero
    at ErrorNException2.main(ErrorNException2.java:13)
Hello 2
java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3
    at ErrorNException2.main(ErrorNException2.java:19)
Hello 3
java.lang.NullPointerException: Cannot invoke "String.length()" because "<local3>" is null
    at ErrorNException2.main(ErrorNException2.java:25)
Hello 4
java.lang.StringIndexOutOfBoundsException: Index 9 out of bounds for length 8
    at java.base/jdk.internal.util.Preconditions$1.apply(Preconditions.java:55)
    at java.base/jdk.internal.util.Preconditions$1.apply(Preconditions.java:52)
    at java.base/jdk.internal.util.Preconditions$4.apply(Preconditions.java:213)
    at java.base/jdk.internal.util.Preconditions$4.apply(Preconditions.java:210)
    at java.base/jdk.internal.util.Preconditions.outOfBounds(Preconditions.java:98)
    at java.base/jdk.internal.util.Preconditions.outOfBoundsCheckIndex(Preconditions.java:106)
    at java.base/jdk.internal.util.Preconditions.checkIndex(Preconditions.java:302)
    at java.base/java.lang.String.checkIndex(String.java:4881)
    at java.base/java.lang.StringLatin1.charAt(StringLatin1.java:46)
    at java.base/java.lang.String.charAt(String.java:1582)
    at ErrorNException2.main(ErrorNException2.java:30)

```


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

PROGRAM CODE :

```
import java.util.Random;

class RandomNumberGenerator extends Thread
{
    public static int number;

    public void run() {
        Random rand = new Random();
        while (true) {
            number = rand.nextInt(100); //
Generates a random number between 0 and 99
            System.out.println("Generated Number:
" + number);
            try {
                Thread.sleep(1000); // Pauses for 1
second
            } catch (InterruptedException e) {

System.out.println("RandomNumberGenerator
interrupted.");
            }
        }
    }
}

class SquareCalculator extends Thread {
    public void run() {
        while (true) {
            if (RandomNumberGenerator.number %
```

```
2 == 0) {  
    int square =  
RandomNumberGenerator.number *  
RandomNumberGenerator.number;  
    System.out.println("Square of " +  
RandomNumberGenerator.number + " is: " +  
square);  
}  
    try {  
        Thread.sleep(1000); // Pauses for 1  
second  
    } catch (InterruptedException e) {  
        System.out.println("SquareCalculator  
interrupted.");  
    }  
}  
}
```

```
class CubeCalculator extends Thread {  
    public void run() {  
        while (true) {  
            if (RandomNumberGenerator.number %  
2 != 0) {  
                int cube =  
RandomNumberGenerator.number *  
RandomNumberGenerator.number *  
RandomNumberGenerator.number;  
                System.out.println("Cube of " +  
RandomNumberGenerator.number + " is: " +  
cube);  
            }  
            try {  
                Thread.sleep(1000); // Pauses for 1  
second  
            } catch (InterruptedException e) {
```

```
        System.out.println("CubeCalculator
interrupted.");
    }
}
}
}

public class MultiThreadedApp {
    public static void main(String[] args) {
        RandomNumberGenerator
randomNumberGenerator = new
RandomNumberGenerator();
        SquareCalculator squareCalculator = new
SquareCalculator();
        CubeCalculator cubeCalculator = new
CubeCalculator();

        randomNumberGenerator.start();
        squareCalculator.start();
        cubeCalculator.start();
    }
}
```

OUTPUT:

```
Generated Number: 78
Square of 0 is: 0
Generated Number: 4
Square of 78 is: 6084
Generated Number: 41
Cube of 41 is: 68921
Cube of 41 is: 68921
Generated Number: 36
Square of 36 is: 1296
Generated Number: 46
Square of 36 is: 1296
Generated Number: 51
Generated Number: 65
Cube of 51 is: 132651
Generated Number: 8
Square of 8 is: 64
Generated Number: 65
Square of 8 is: 64
Cube of 65 is: 274625
Generated Number: 46
Cube of 65 is: 274625
Square of 46 is: 2116
Generated Number: 75
Square of 46 is: 2116
Cube of 75 is: 421875
Generated Number: 94
```

CONCLUSION:

This Java program calculates and displays the speed of an object in meters per second (m/s), kilometers per hour (km/h), and miles per hour (mph) based on user input for distance in meters and time in hours, minutes, and seconds.

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

PROGRAM CODE :

```
class SharedBuffer {  
    int item; // A shared place for the item  
    boolean isProduced = false; // Whether the item is produced or not  
  
    public synchronized void produce() throws InterruptedException {  
        if (isProduced) {  
            return; // If an item is already produced, do nothing  
        }  
        item = (int) (Math.random() * 100); // Produce a random item  
        System.out.println("Produced: " + item);  
        isProduced = true; // Mark the item as produced  
        notify(); // Notify the consumer that the item is ready  
    }  
  
    public synchronized void consume() throws InterruptedException {  
        if (!isProduced) {  
            return; // If no item is produced, do nothing  
        }  
        System.out.println("Consumed: " + item); // Consume the item  
        isProduced = false; // Mark that the item has been consumed  
        notify(); // Notify the producer that the buffer is now empty  
    }  
}  
  
class Producer extends Thread {
```

```
SharedBuffer buffer;

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

@Override
public void run() {
    try {
        for (int i = 0; i < 10; i++) {
            buffer.produce(); // Produce an item
            Thread.sleep(1000); // Simulate some delay
        }
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}

class Consumer extends Thread {
    SharedBuffer buffer;

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

    @Override
    public void run() {
        try {
```

```
        for (int i = 0; i < 10; i++) {
            buffer.consume(); // Consume an item
            Thread.sleep(1000); // Simulate some delay
        }
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}

}

public class ProducerConsumerThreadExample {
    public static void main(String[] args) throws InterruptedException {
        SharedBuffer buffer = new SharedBuffer(); // Shared buffer

        // Create producer and consumer threads by extending Thread
        Producer producerThread = new Producer(buffer);
        Consumer consumerThread = new Consumer(buffer);

        // Start the threads
        producerThread.start();
        consumerThread.start();

        // Wait for both threads to complete
        producerThread.join();
        consumerThread.join();

        System.out.println("Producer and Consumer have finished execution.");
    }
}
```

```
}
```

OUTPUT:

```
Produced: 11
Consumed: 11
Produced: 41
Consumed: 41
Produced: 75
Consumed: 75
Produced: 78
Consumed: 78
Produced: 79
Consumed: 79
Produced: 52
Consumed: 52
Produced: 14
Consumed: 14
Produced: 50
```

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 FirstThread extends Thread {
    public void run() {
        System.out.println("Thread 'FIRST' is running with priority: " + this.getPriority());
    }
}

class SecondThread extends Thread {
    public void run() {
        System.out.println("Thread 'SECOND' is running with priority: " + this.getPriority());
    }
}
```



```
class ThirdThread extends Thread {  
    public void run() {  
        System.out.println("Thread 'THIRD' is running with priority: " + this.getPriority());  
    }  
}  
  
public class ThreadPriorityDemo {  
    public static void main(String[] args) {  
        // Create the thread objects  
        FirstThread first = new FirstThread();  
        SecondThread second = new SecondThread();  
        ThirdThread third = new ThirdThread();  
  
        // Set thread names  
        first.setName("FIRST");  
        second.setName("SECOND");  
        third.setName("THIRD");  
  
        // Set priorities  
        first.setPriority(3); // Priority of FIRST thread set to 3  
        second.setPriority(Thread.NORM_PRIORITY); // Default priority 5  
        third.setPriority(7); // Priority of THIRD thread set to 7  
  
        // Start the threads  
        first.start();  
        second.start();  
        third.start();  
    }  
}
```

OUTPUT:

```
Thread 'FIRST' is running with priority: 3  
Thread 'SECOND' is running with priority: 5  
Thread 'THIRD' is running with priority: 7
```

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

PROGRAM CODE :

```
class thread extends Thread{  
    public void run(){  
        try {  
            for(int i=1;i<=10;i++){  
                System.out.println(i);  
                Thread.sleep(1000);  
            }  
  
            } catch (Exception e) {  
                System.out.println(e);  
            }  
        }  
    }  
}  
  
public class Increment {
```

```
public static void main(String[] args) {  
    Thread t1=new Thread(new thread());  
    t1.start();  
}  
}
```

OUTPUT:

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

PART-VIII Collection Framework and Generic

38	<p>Design a Custom Stack using ArrayList class, which implements following functionalities of stack. My Stack</p> <ul style="list-style-type: none"> -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. <p>Program:</p> <pre>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() {</pre>
-----------	---

```
        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:

```

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:

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.

- 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:

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

```
public class P39 {
```

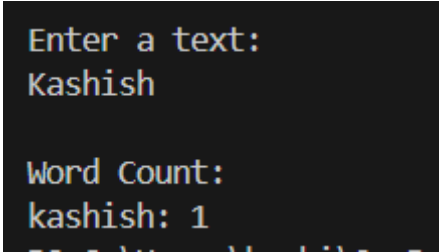
```

    public static <T extends Comparable<T>> void sort(T[] array) {
        int n = array.length;
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                if (array[j].compareTo(array[j + 1]) > 0) {
                    T temp = array[j];
                    array[j] = array[j + 1];
                    array[j + 1] = temp;
                }
            }
        }
    }
}

```

```
    }  
    }  
}  
  
static class Product implements Comparable<Product> {  
    String name;  
    double price;  
    double rating;  
  
    public Product(String name, double price, double rating) {  
        this.name = name;  
        this.price = price;  
        this.rating = rating;  
    }  
  
    @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);  
}
```

	<pre> System.out.println("After sorting: " + Arrays.toString(products)); } } </pre> <p>OUTPUT:</p> <pre> 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}] </pre> <p>CONCLUSION:</p> <p>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.</p>
40 .	<p>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.</p> <p>Program:</p> <pre> 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); } } } } </pre>

	<pre> } 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(); } } </pre> <p>OUTPUT:</p>  <p>CONCLUSION:</p> <p>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."</p>
41	<p>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.</p> <p>Program:</p> <pre> 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) { </pre>

```
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:

```
Enter the path of the Java source file: c:\Users\kashi\OneDrive\desktop\vs\java_prc\set8\P38.jav
Number of Java keywords in the Drivwfile: 28
```

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.