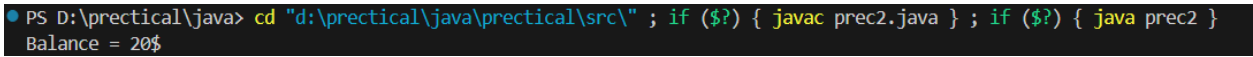


CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY**DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY & RESEARCH**

Department of Computer Science & Engineering

Subject Name: Java Programming**Semester:** 3rd**Subject Code:** CSE201**Academic year:** 2024-25**Part - 1**

No.	Data Types, Variables, String, Control Statements, Operators, Arrays
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>public class prec2 { public static void main(String[] args) { int a = 20; System.out.println("Balance = "+a+"\$"); } }</pre> <p><u>OUTPUT:</u></p>  <pre>PS D:\prectical\java> cd "d:\prectical\java\prectical\src\" ; if (\$?) { javac prec2.java } ; if (\$?) { java prec2 } Balance = 20\$</pre> <p><u>CONCLUSION:</u> Learnt about the basics of java and basic syntax.</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.*;

public class prec3 {

    public static void main(String[] args) {
        float time[] = new float[3];
        Scanner s = new Scanner(System.in);
        int choice;
        float dis, tmin, tsec, thour;
        float calculate;

        System.out.print("Enter Distance : ");
        dis = s.nextFloat();

        System.out.print("Enter Time In Hours : ");
        time[0] = s.nextFloat();
        System.out.print("Enter Time In Min : ");
        time[1] = s.nextFloat();
        System.out.print("Enter Time In Sec : ");
        time[2] = s.nextFloat();

        tmin = (time[0] * 60);
        tsec = ((tmin + time[1]) * 60) + time[2];
        thour = time[0] + (time[1] / 60) + (time[2] / 3600);

        System.out.println("Choose Unit To Display Velocity");
        System.out.println("1.m/s");
        System.out.println("2.km/h");
        System.out.println("3.miles/h");
        choice = s.nextInt();
```

```

switch (choice) {
    case 1:

        calculate = dis / tsec;
        System.out.println("Velocity = " +
calculate + " m/s");
        break;

    case 2:
        dis = dis / 1000;
        calculate = dis / thour;
        System.out.println("Velocity = " +
calculate + " km/h");
        break;

    case 3:
        dis = dis / 1609;
        calculate = dis / thour;
        System.out.println("Velocity = " +
calculate + " miles/h");
        break;
    }
}
}

```

OUTPUT:

```

Enter Distance : 1000
Enter Time In Hours : 1
Enter Time In Min : 0
Enter Time In Sec : 0
Choose Unit To Display Velocity
1.m/s
2.km/h
3.miles/h
2
Velocity = 1.0 km/h

```

```
Enter Distance : 1000
Enter Time In Hours : 1
Enter Time In Min : 0
Enter Time In Sec : 0
Choose Unit To Display Velocity
1.m/s
2.km/h
3.miles/h
1
Velocity = 0.2777778 m/s
```

```
Enter Distance : 1235
Enter Time In Hours : 1
Enter Time In Min : 2
Enter Time In Sec : 5
Choose Unit To Display Velocity
1.m/s
2.km/h
3.miles/h
3
Velocity = 0.74180055 miles/h
```

CONCLUSION: In this practical we learnt about scanner class by using it we can take input from the user.

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;

public class prec4 {
    public static void main(String[] args) {

        int i;
        float sum=0;
        float arr[]=new float[5];

        Scanner s = new Scanner(System.in);

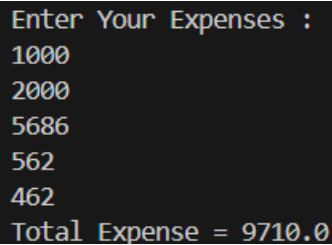
        System.out.println("Enter Your Expenses :
");
        for(i=0;i<5;i++)
```

```

    {
        arr[i]=s.nextFloat();
        sum+=arr[i];
    }
    System.out.println("Total Expense = " +
sum);
    }
}

```

OUTPUT:



```

Enter Your Expenses :
1000
2000
5686
562
462
Total Expense = 9710.0

```

CONCLUSION: Learnt about array, how array works in java, ways to initialize array.

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;

public class prec5 {

    public static void main(String[] args) {
        float sum = 0;
        float price[] = {100, 100, 100, 100, 100};
        int choice, qun;
        Scanner s = new Scanner(System.in);

        System.out.println("1.Motor");
        System.out.println("2.Fan");
        System.out.println("3.Tube");
    }
}

```

```

        System.out.println("4.Wires");
        System.out.println("5.Other");
        System.out.println("Enter Item You Want
to Add : ");
        choice = s.nextInt();

        System.out.println("Total Quntity : ");
        qun = s.nextInt();

        switch (choice) {
            case 1:

                sum += (price[0]+(price[0] * 0.08)) *
qun;

                break;
            case 2:

                sum += (price[1]+(price[1] * 0.12)) *
qun;

                break;
            case 3:

                sum += (price[2]+(price[2] * 0.05))*
qun;

                break;

            case 4:

                sum += (price[3]+(price[3] * 0.075))
* qun;

                break;

            case 5:

                sum += (price[4]+(price[4] * 0.03)) *
qun;

```

```

        break;
    }

    System.out.println("Total = " + sum);

}
}

```

OUTPUT:

```

1.Motor
2.Fan
3.Tube
4.Wires
5.Other
Enter Item You Want to Add :
1
Total Quntity :
5
Total = 540.0

```

CONCLUSION: Learnt about the switch case.

6. Create a Java program that prompts the user to enter the number of days (n) for which they want to generate their exercise routine. The program should then calculate and display the first n terms of the Fibonacci series, representing the exercise duration for each day.

PROGRAM CODE:

```

import java.util.*;

public class prec6{
    public static void main(String[] args) {

        int i,n;
        long sum=0,ref1=0,ref2=1,temp;
        System.out.println("Enter Total No Of Fibbonaci Value");
        Scanner s = new Scanner(System.in);
        n=s.nextInt();

        System.out.println(ref1 + "\n" + ref2);
        for(i=1;i<=n;i++)

```

```

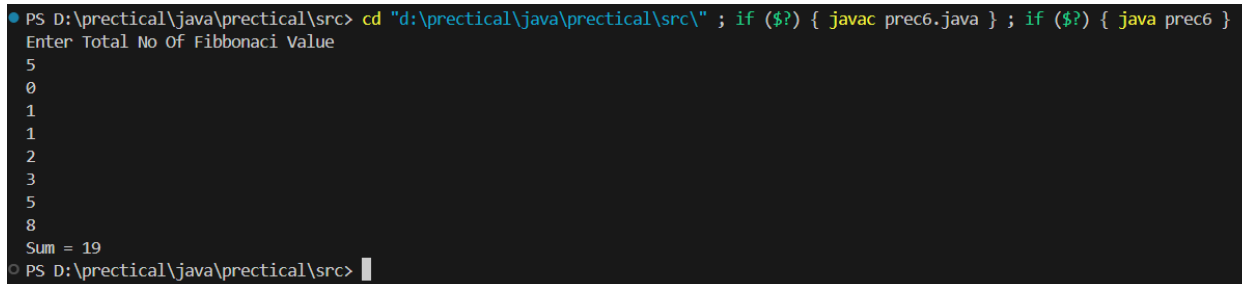
    {
        temp=ref1+ref2;
        ref1=ref2;
        ref2=temp;

        System.out.println(temp);
        sum+=temp;
    }

    System.out.println("Sum = " + sum );
}
}

```

OUTPUT:



```

PS D:\prectical\java\prectical\src> cd "d:\prectical\java\prectical\src\" ; if ($?) { javac prec6.java } ; if ($?) { java prec6 }
Enter Total No Of Fibonacci Value
5
0
1
1
2
3
5
8
Sum = 19
PS D:\prectical\java\prectical\src>

```

CONCLUSION: In this practical we learnt about the implementation of Fibonacci Series.

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

Department of Computer Science & Engineering

Subject Name: Java Programming**Semester:** 3rd**Subject Code:** CSE201**Academic year:** 2024-25

Part - 2

No.	Strings
7	<p>Given a string and a non-negative int n, we'll say that the front of the string is the first 3 chars, or whatever is there if the string is less than length 3. Return n copies of the front; front_times('Chocolate', 2) → 'ChoCho' front_times('Chocolate', 3) → 'ChoChoCho' front_times('Abc', 3) → 'AbcAbcAbc'</p> <p><u>PROGRAM CODE:</u></p> <pre>import java.util.Scanner; public class prec7 { public static void main(String[] args) { String n; Scanner s = new Scanner(System.in); System.out.println("Enter String = "); n = s.nextLine(); String n1 = n.substring(0, 3); front_times(n1, 2); } static void front_times(String n, int a) {</pre>

```

    int i;
    for (i = 0; i < a; i++) {
        System.out.print(n);
    }
}
}

```

OUTPUT:

```

PS D:\prectical\java\prectical\src> cd "d:\prectical\java\prectical\src\" ; if ($?) { javac prec7.java } ; if ($?) { java prec7 }
Enter String =
Chocolate
ChoCho

```

CONCLUSION: Learnt about strings and their functions.

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

PROGRAM CODE:

```

public class prec8 {

    public static void main(String[] args) {

        int[] arr={1,2,4,7,8,9,9,9};
        count9(arr);

    }

    static void count9(int[]arr)
    {
        int count=0;

        for(int i=0;i<arr.length;i++)
        {

```

```

        if(arr[i]==9)
        {
            count++;
        }
    }

```

```

        System.out.println("Total Count = "+
count );

```

```

    }

}

```

OUTPUT:

```

PS D:\prectical\java\prectical\src> cd "d:\prectical\java\prectical\src\" ; if ($?) { javac prec8.java } ; if ($?) { java prec8 }
Total Count = 3

```

CONCLUSION: In this practical we created function which it counts total numbers of 9 in the given array.

- 9 Given a string, return a string where for every char in the original, there are two chars. double_char('The') → 'TThh ee' double_char('AAAbb') → 'AAAAbbbb' double_char('Hi-There') → 'HHii--TThheerree'

PROGRAM CODE:

```

import java.util.Scanner;

public class prec9 {

    public static void main(String[] args) {

        Scanner s1 = new Scanner(System.in);
        System.out.print("Enter your string: ");
        String str = s1.nextLine();
        for (int i = 0; i < str.length(); i++) {
            char c = str.charAt(i);
            System.out.print(c + "" + c);
        }

    }

}

```

OUTPUT:

```
PS D:\prectical\java\prectical\src> cd "d:\prectical\java\prectical\src\" ; if ($?) { javac prec9.java } ; if ($?) { java prec9 }
Enter your string: The
TThhee
PS D:\prectical\java\prectical\src> |
```

CONCLUSION: In this practical we used charAt function to access each character of string and double them.

10 Perform following functionalities of the string:

- Find Length of the String
- Lowercase of the String
- Uppercase of the String
- Reverse String
- Sort The String

PROGRAM CODE:

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

```
class prec10 {
    public static void main(String[] args) {
        String a;
        int i, j;
        char ch1;
        Scanner s = new Scanner(System.in);

        System.out.println("Enter String : ");
        a = s.nextLine();

        char[] c = a.toCharArray();

        System.out.println("String Length = " + a.length());
        System.out.println("Lower String = " + a.toLowerCase());
        System.out.println("Upper String = " + a.toUpperCase());
        System.out.println("Reversed String = ");
        for (i = a.length() - 1; i >= 0; i--) {
            System.out.print(a.charAt(i));
        }
        System.out.println("\nSorted String = ");
    }
}
```

```

    for (i = 0; i < a.length()-1; i++) {
        for (j = 0; j < a.length() - i-1; j++) {
            if (c[j] > c[j + 1]) {
                ch1 = c[j];
                c[j] = c[j + 1];
                c[j + 1] = ch1;
            }
        }
    }
    for(i=0;i<a.length();i++)
    {
        System.out.print(c[i]);

    }
}

```

OUTPUT:

```

Enter String :
Rudra Bhavsar
String Length = 13
Lower String = rudra bhavsar
Upper String = RUDRA BHAVSAR
Reversed String =
rasvahB arduR
Sorted String =
BRaaadhrrsuv
PS D:\prectical\java\prectical\src>

```

CONCLUSION: In this practical we used multiple string manipulation funtions.

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:

```

public class prec11 {
    public static void main(String[] args) {
        String a = "CHARUSAT UNIVERSITY";

        System.out.println("Length Of String = " + a.length());

        char [] c = a.toCharArray();

        for(int i=0;i<a.length();i++)
        {

```

```

        if(c[i]=='H')
        {
            c[i]='R';
        }

    }

    for(int i=0;i<a.length();i++)
    {
        System.out.print(c[i]);

    }

    System.out.println("\nLowered String = ");
    for (int i = 0; i < a.length(); i++) {
        if(c[i]>='A' && c[i]<'Z')
        {
            c[i]+=32;

        }
        System.out.print(c[i]);

    }
}
}}

```

OUTPUT:

```

Length Of String = 19
CRARUSAT UNIVERSITY
Lowered String =
crarusat university
PS D:\prectical\java\prectical\src>

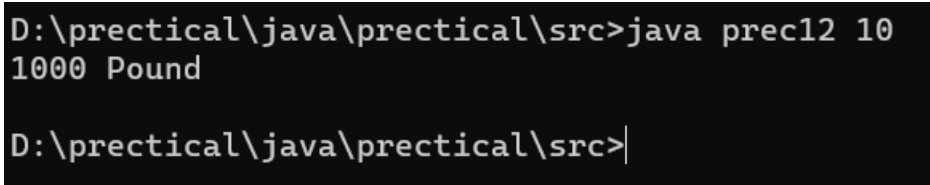
```

CONCLUSION: In this practical we learnt charArray function.

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

Department of Computer Science & Engineering

Subject Name: Java Programming**Semester:** 3rd**Subject Code:** CSE201**Academic year:** 2024-25**Part - 3**

No.	Object Oriented Programming: Classes, Methods, Constructors
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>public class prec12 { public static void main(String []args) { int a = Integer.parseInt(args[0]); int c = a*100; System.out.println(c + " Pound"); } }</pre> <p><u>OUTPUT:</u></p>  <p><u>CONCLUSION:</u> Learnt about the working of command line argument.</p>

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

PROGRAM CODE:

```
import java.util.Scanner;

class employee {
    String fn, ln;
    double salary;
    Scanner s = new Scanner(System.in);

    employee() {
    }

    employee(String f, String l, double sl) {
        fn = f;
        ln = l;
        salary = sl;
    }

    void setfirstname()
    {
        System.out.println("Enter Employee's First Name : ");
        fn=s.nextLine();
    }

    void setlastname()
    {
        System.out.println("Enter Employee's Last Name : ");
        ln=s.nextLine();
    }

    void setsalary()
    {
        System.out.println("Enter Employee's Salary : ");
        salary=s.nextDouble();
        if(salary<0)
        {
```



```

        salary=0;
    }
}

String getfirstname()
{
    return fn;
}
String getlastname()
{
    return ln;
}
double getsalary()
{
    return salary;
}
}

public class prec13 {
    public static void main(String[] args) {

        employee e1=new employee();
        employee e2=new employee();

        e1.setfirstname();
        e1.setlastname();
        e1.setsalary();

        System.out.println("First Name : " + e1.getfirstname());
        System.out.println("Last Name : " + e1.getlastname());
        System.out.println("Salary : " + e1.getsalary());
        System.out.println("New Salary : " + (e1.getsalary()+(e1.getsalary()*0.1)));

        System.out.println("-----");

        e2.setfirstname();
        e2.setlastname();
        e2.setsalary();

        System.out.println("First Name : " + e2.getfirstname());
        System.out.println("Last Name : " + e2.getlastname());
        System.out.println("Salary : " + e2.getsalary());
    }
}

```

```
System.out.println("New Salary : " + (e2.getsalary()+(e2.getsalary()*0.1)));
```

```
}
```

```
}
```

OUTPUT:

```
PS D:\prectical\java\prectical\src> cd "d:\prectical\java\prectical\src\" ; if ($?) { javac prec13.java } ; if ($?) { java prec13 }
Enter Employee's First Name :
Rudra
Enter Employee's Last Name :
Bhavsar
Enter Employee's Salary :
50000
First Name : Rudra
Last Name : Bhavsar
Salary : 50000.0
New Salary : 55000.0
-----
Enter Employee's First Name :
Deep
Enter Employee's Last Name :
Talati
Enter Employee's Salary :
50000
First Name : Deep
Last Name : Talati
Salary : 50000.0
New Salary : 55000.0
```

CONCLUSION: In this practical we learnt about constructors and methods.

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 {
```

```
    int date, month, year;
```

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

```
    date() {
    }
```

```
    date(int dt, int m, int ye) {
        date = dt;
        month = m;
        year = ye;
    }
```

```
    void setdate() {
        System.out.println("Enter Date : ");
```

```

        date = s.nextInt();
    }

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

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

    void putdate() {
        System.out.println(date + "/" + month + "/" + year);
    }
}

public class prec14 {
    public static void main(String[] args) {

        date d1 = new date();

        d1.setdate();
        d1.setmonth();
        d1.setyear();
        d1.putdate();

    }
}

```

OUTPUT:

```

PS D:\prectical\java\prectical\src> cd "d:\prectical\java\prectical\src\" ; if ($?) { javac prec14.java } ; if ($?) { java prec14 }
Enter Date :
07
Enter Month :
09
Enter Year :
2005
7/9/2005

```

CONCLUSION: In this practical we created class by which we can see date in DD/MM/YY format.

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

PROGRAM CODE:

```
class area {  
  
    int length, breadth;  
  
    area() {  
    }  
  
    area(int len, int brth) {  
        length=len;  
        breadth=brth;  
    }  
  
    int getarea()  
    {  
        return length*breadth;  
    }  
}  
  
public class prec15 {  
    public static void main(String[] args) {  
  
        area a1=new area(50, 20);  
        System.out.println("Area : "+ a1.getarea());  
    }  
}
```

OUTPUT:

```
PS D:\prectical\java\prectical\src> cd "d:\prectical\java\prectical\src\" ; if ($?) { javac prec15.java } ; if ($?) { java prec15 }  
Area : 1000  
PS D:\prectical\java\prectical\src>
```

CONCLUSION: In this practical we created a class by which we can calculate area of rectangle.

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

PROGRAM CODE:

```
import java.util.Scanner;

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

    void getvalue()
    {
        System.out.println("Enter Real Part ");
        r=s.nextInt();
        System.out.println("Enter Imaginary Part ");
        i=s.nextInt();
    }

    void add(complex cx)
    {
        int sumr=r+cx.r;
        int sumi=i+cx.i;

        System.out.println("Sum :"+ sumr + "+" + sumi + "i");
    }

    void sub(complex cx)
    {
        int sumr=r-cx.r;
        int sumi=i-cx.i;

        System.out.println("Subtraction :"+ sumr + "+" + sumi + "i");
    }

    void mul(complex cx)
    {
        int sumr=(r*cx.r)-(i*cx.i);
        int sumi=(r*cx.r)+(i*cx.i);

        System.out.println("multiplication :"+ sumr + "+" + sumi + "i");
    }
}
```

```

}

public class prec16 {
    public static void main(String[] args) {

        complex c1=new complex();
        complex c2=new complex();

        c1.getvalue();
        c2.getvalue();
        c1.add(c2);
        c1.sub(c2);
        c1.mul(c2);

    }
}

```

OUTPUT:

```

PS D:\prectical\java\prectical\src> cd "d:\prectical\java\Java\Set-3\" ; if ($?) { javac prec16.java } ; if ($?) { java prec16 }
Enter Real Part
2
Enter Imaginary Part
3
Enter Real Part
1
Enter Imaginary Part
2
Sum :3+5i
Subtraction :1+1i
multiplication :-4+8i

```

CONCLUSION: In this practical we created a class by which we can do addition, subtraction and multiplication of complex numbers.

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

Department of Computer Science & Engineering

Subject Name: Java Programming**Semester:** 3rd**Subject Code:** CSE201**Academic year:** 2024-25**Part - 4**

No.	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><u>PROGRAM CODE:</u></p> <pre>class parent{ void display1() { System.out.println("This is Parent Class"); } } class child extends parent{ void display2() { System.out.println("This is Child Class"); } } public class prec17{</pre>

```

public static void main(String[]args)
{
    parent p=new parent();
    child c=new child();

    c.display1();
    c.display2();

}
}

```

OUTPUT:

```

PS D:\prectical\java\Java> cd "d:\prectical\java\Java\Set-4\" ; if ($?) { javac prec17.java } ; if ($?) { java prec17 }
This is Parent Class
This is Child Class

```

CONCLUSION: Learnt about the working of inheritance.

18. Create a class named 'Member' having the following members: Data members 1 - Name 2 - Age 3 - Phone number 4 - Address 5 – Salary It also has a method named 'printSalary' which prints the salary of the members. Two classes 'Employee' and 'Manager' inherits the 'Member' class. The 'Employee' and 'Manager' classes have data members 'specialization' and 'department' respectively. Now, assign name, age, phone number, address and salary to an employee and a manager by making an object of both of these classes and print the same.

PROGRAM CODE:

```

import java.util.Scanner;

class Member {
    String name, address;
    int age, salary;
    long phone;
    Scanner scan = new Scanner(System.in);

    void printSalary() {
        System.out.println("Salary = " + salary);
    }

    void getdata() {
        System.out.println("Enter Name : ");
        name = scan.nextLine();
        System.out.println("Enter Address : ");
        address = scan.nextLine();
        System.out.println("Enter Age : ");
    }
}

```



```

        age = scan.nextInt();
        System.out.println("Enter Salary : ");
        salary = scan.nextInt();
        System.out.println("Enter Phone : ");
        phone = scan.nextLong();
    }

    void putdata() {

        System.out.println("-----");
        System.out.println("Name : " + name);
        System.out.println("Address : " + address);
        System.out.println("Age : " + age);
        System.out.println("Salary : " + salary);
        System.out.println("Phone : " + phone);
    }
}

class Employee extends Member {
    String specialization;

    void getdata() {
        super.getdata();
        scan.nextLine();
        System.out.println("Enter Specialization : ");
        specialization = scan.nextLine();
    }

    void putdata() {
        super.putdata();
        System.out.println("Specialization : " + specialization);
        System.out.println("-----");
    }
}

class Manager extends Member {
    String department;

    void getdata() {
        super.getdata();
        scan.nextLine();
        System.out.println("Enter Department : ");
    }
}

```

```

        department = scan.nextLine();
    }

    void putdata() {
        super.putdata();
        System.out.println("Department : " + department);
        System.out.println("-----");
    }
}

public class prec18 {
    public static void main(String[] args) {

        Employee e1 = new Employee();
        Manager m1 = new Manager();

        e1.getdata();
        e1.putdata();

        m1.getdata();
        m1.putdata();

    }
}

```

OUTPUT:

```
Rudra
Enter Address :
Shivnagar
Enter Age :
18
Enter Salary :
86546
Enter Phone :
8160588463
Enter Specialization :
Java
```

```
-----
Name : Rudra
Address : Shivnagar
Age : 18
Salary : 86546
Phone : 8160588463
Specialization : Java
-----
```

```
Enter Name :
Deep
Enter Address :
Primehill
Enter Age :
18
Enter Salary :
862445
Enter Phone :
8165234578
Enter Department :
CA
```

```
-----
Name : Deep
Address : Primehill
Age : 18
Salary : 862445
Phone : 8165234578
Department : CA
-----
```

CONCLUSION: In this practical we learnt about SuperKeyword.

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:

```
import java.util.Scanner;

class Rectangle {
    int length, breadth;

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

    void area() {
        System.out.println("Area Of Rectangle : " + length * breadth);
    }

    void parameter() {
        System.out.println("Parameter Of Rectangle : " + 2 * (length + breadth));
    }
}

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

    void area() {
        System.out.println("Area Of Square : " + length * length);
    }

    void parameter() {
        System.out.println("Parameter Of Square : " + 2 * length);
    }
}
```

```

public class prec19 {
    public static void main(String[] args) {

        Rectangle r[] = new Rectangle[5];
        Square s[] = new Square[5];
        Scanner scan = new Scanner(System.in);

        int i, l, b;
        for (i = 0; i < 5; i++) {
            System.out.println("Enter Length For " + (i + 1) + " Rectangle : ");
            l = scan.nextInt();
            System.out.println("Enter Breadth For " + (i + 1) + " Rectangle : ");
            b = scan.nextInt();

            r[i] = new Rectangle(l, b);

            r[i].area();
            r[i].parameter();

            System.out.println("Enter Length For " + (i + 1) + " Square : ");
            l = scan.nextInt();
            s[i] = new Square(l);
            s[i].area();
            s[i].parameter();

        }
    }
}

```

OUTPUT:

```
Enter Length For 1 Reactangle :  
5  
Enter Breadth For 1 Reactangle :  
10  
Area Of Reactangle : 50  
Parameter Of Reactangle : 30  
Enter Length For 1 Square :  
5  
Area Of Square : 25  
Parameter Of Square : 10  
Enter Length For 2 Reactangle :  
5  
Enter Breadth For 2 Reactangle :  
1  
Area Of Reactangle : 5  
Parameter Of Reactangle : 12  
Enter Length For 2 Square :  
2  
Area Of Square : 4  
Parameter Of Square : 4  
Enter Length For 3 Reactangle :  
3  
Enter Breadth For 3 Reactangle :  
1  
Area Of Reactangle : 3  
Parameter Of Reactangle : 8  
Enter Length For 3 Square :  
6  
Area Of Square : 36  
Parameter Of Square : 12  
Enter Length For 4 Reactangle :  
4  
Enter Breadth For 4 Reactangle :  
5  
Area Of Reactangle : 20  
Parameter Of Reactangle : 18  
Enter Length For 4 Square :  
8  
Area Of Square : 64
```

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

PROGRAM CODE:

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

class Rectangle extends Shape {
    void printrect() {
        System.out.println("This is Rectangular shape");
    }
}

class Circle extends Shape {
    void printcir() {
        System.out.println("This is Circle shape");
    }
}

class Square extends Rectangle {
    void printsqu() {
        System.out.println("Square Is a Rectangle");
    }
}

public class prec20 {
    public static void main(String[] args) {

        Square s1 = new Square();
        s1.printbase();
        s1.printrect();

    }
}
```

OUTPUT:

This is This is shape
This is Rectangular shape

CONCLUSION: The example provided illustrates core object-oriented programming principles, including inheritance, polymorphism, method overriding, and encapsulation. By structuring classes in a hierarchical manner that reflects logical relationships between shapes, the design promotes code reusability, flexibility, and maintainability. The use of polymorphism and method resolution order ensures that the correct behavior is invoked for each specific object type, showcasing the power and versatility of object-oriented design.

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

PROGRAM CODE:

```
class Degree {
    void getDegree() {
        System.out.println("I got a Degree");
    }
}

class Undergraduate {
    void print() {
        System.out.println("I am Undergraduate ");
    }
}

class Postgraduate {
    void print() {
        System.out.println("I am a Postgraduate");
    }
}

public class prec21 {
    public static void main(String[] args) {

        Degree d1 = new Degree();
        Undergraduate u1 = new Undergraduate();
    }
}
```



```
Postgraduate p1 = new Postgraduate();
```

```
d1.getDegree();
```

```
u1.print();
```

```
p1.print();
```

```
}
```

```
}
```

OUTPUT:

```
I got a Degree
```

```
I am Undergraduate
```

```
I am a Postgraduate
```

CONCLUSION: This demonstrates **polymorphism**, where the method in the subclass overrides the method from the parent class when called on an instance of the subclass.

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.

PROGRAM CODE:

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

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

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

```
public class prec22 {  
    public static void main(String[] args) {  
        int result,n;
```

```
MyCalculator obj = new MyCalculator();
Scanner scan=new Scanner(System.in);
System.out.println("Enter Number : ");
n=scan.nextInt();
scan.close();
result=obj.divisor_sum(n);
System.out.println("Divisor Sum : " + result);

}
}
```

OUTPUT:

```
● PS D:\prectical\java
Enter Number :
6
Divisor Sum : 12
```

CONCLUSION: implementation of interface.

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

Department of Computer Science & Engineering

Subject Name: Java Programming**Semester:** 3rd**Subject Code:** CSE201**Academic year:** 2024-25**Part - 5**

No.	Exception Handling
24.	<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:</u></p> <pre> import java.util.Scanner; class calculate { void divide(int a,int b) { int c; try { c=a/b; System.out.println("Division = " + c); } catch (Exception e) { System.out.println("Dividing By Zero Is Invalid"); } } } public class prec24{ public static void main(String[] args) { calculate c = new calculate(); </pre>

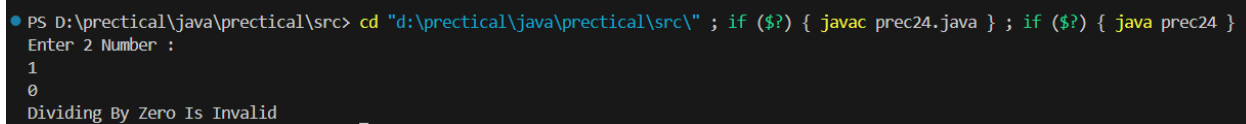
```

int a,b;
Scanner s=new Scanner(System.in);
System.out.println("Enter 2 Number : ");
a=s.nextInt();
b=s.nextInt();

c.divide(a, b);
}
}

```

OUTPUT:



```

PS D:\prectical\java\prectical\src> cd "d:\prectical\java\prectical\src\" ; if ($?) { javac prec24.java } ; if ($?) { java prec24 }
Enter 2 Number :
1
0
Dividing By Zero Is Invalid

```

CONCLUSION: Learnt about the Exception and basics of it.

24

PRACTICE PROGRAM:

1.

```

public class error {
    public static void main(String[] args) {
        // int a = 3, b = 0, c;
        // System.out.println("before");
        // try {
        //     c = a / b;
        // } catch (Exception e) {
        //     System.out.println("Exception resolved ");
        //     System.out.println(e.toString());
        //     System.out.println(e.getMessage());
        //     e.printStackTrace();
        // }

        // System.out.println("After");

        int a[] = { 0, 1, 2 };
        try {
            System.out.println(a[3]);
        } catch (Exception e) {
            System.out.println("handeld" + e);
        }
    }
}

```

```

String s = "Charusat";
try {
    System.out.println(s.charAt(9));
} catch (Exception e) {

    System.out.println("Handeled" + e);
}

String s1 = null;
try
{
    System.out.println(s1.length());
}
catch(Exception e)
{
    System.out.println("handeled" + e);
}
}

```

2.

```

public class exception {
    public static void main(String[] args) {
        // int a[] = { 0, 1, 2 };
        // try {
        // System.out.println(a[3]);
        // }
        // catch(ArrayIndexOutOfBoundsException ab)
        // {
        // System.out.println("hello");
        // }catch (Exception e) {
        // System.out.println("handeld" + e);
        // }

        int a = 3, b = 0, c, arr[] = { 0, 1, 2 };
        String s = "Charusat";
        String s1 = null;
        // try {
        // c = a / b;
        // System.out.println(arr[3]);
    }
}

```

```

// System.out.println(s.charAt(9));
// System.out.println(s1.length());

// } catch (ArrayIndexOutOfBoundsException e) {
// System.out.println("Array");
// } catch (StringIndexOutOfBoundsException e) {
// System.out.println("String");
// } catch (ArithmeticException e) {
// System.out.println("Arithmetic");
// } catch (NullPointerException e) {
// System.out.println();
// }

try {
    try {
        c = a / b;
    } catch (ArithmeticException e) {
        System.out.println("Arithmetic");
    }
    try {
        System.out.println(arr[3]);
    } catch (ArrayIndexOutOfBoundsException e) {
        System.out.println("Array");
    }
    try {
        System.out.println(s.charAt(9));
    } catch (StringIndexOutOfBoundsException e) {
        System.out.println("String");
    }
    try {
        System.out.println(s1.length());
    } catch (NullPointerException e) {
        System.out.println("null");
    }

} catch (Exception e) {

    System.out.println("Anything ");
}
}

```


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 CODE:

```
import java.io.*;

public class P27 {

    public static void main(String[] args) throws Exception {

        if (args.length == 0) {

            System.out.println("No file Found!");

        } else {

            for (int i = 0; i < args.length; i++) {

                try {

                    BufferedReader f = new BufferedReader(new FileReader(args[i]));

                    String j;

                    int count = 0;

                    while ((j = f.readLine()) != null) {

                        count++;

                    }

                    System.out.println("File name is : " + args[i] + " and Number of lines are : " + count);

                } catch (Exception e) {

                    System.out.println(e);

                }

            }

        }

    }

}
```



```
} } }
```

```
System.out.println("ID :23DCS012_Rudra Bhavsar");
```

```
} }
```

OUTPUT:

```
File name is : file1.txt and Number of lines are : 5  
File name is : file2.txt and Number of lines are : 4  
File name is : file3.txt and Number of lines are : 9
```

CONCLUSION:

This Java program reads several files named by the command line arguments and counts the number of lines in each. If no files are provided as command-line arguments, it will print out the appropriate message. Exception handling ensures graceful error management during file reading, thus a stable program.

28

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

PROGRAM CODE:

```
import java.io.BufferedReader;  
  
import java.io.FileReader;  
  
import java.io.IOException;  
  
public class 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 count = 0;

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

    int ch;

    while ((ch = reader.read()) != -1) {

        if (ch == targetChar) {

            count++;

        }

    }

    System.out.println("The character '" + targetChar + "' appears " + count + " times in " +
        fileName);

} catch (IOException e) {

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

}

System.out.println("ID :23DCS012_Rudra Bhavsar");

}}
```

OUTPUT:



```
PS C:\Users\patel\OneDrive\Desktop\EXAM\Sem 3\JAVA\Practical\JAVA_VSCODE> javac P28.java
PS C:\Users\patel\OneDrive\Desktop\EXAM\Sem 3\JAVA\Practical\JAVA_VSCODE> java P28 a file1.txt
The character 'a' appears 16 times in file1.txt
```

CONCLUSION:

The Java program successfully counts the occurrences of a specified character in a given file, providing the result in a clear format. It handles file read errors gracefully, ensuring robust

performance even if issues arise during file access.

29

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

PROGRAM CODE:

```
import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

public class P29 {

    public static void main(String[] args) {

        if (args.length < 2) {

            System.out.println("Usage: java P29 <word> <filename>");

            return;

        }

        String searchWord = args[0];

        String fileName = args[1];

        Integer count = 0;

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

            String line;

            while ((line = reader.readLine()) != null) {

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

                for (String word : words) {
```

```
if (word.equalsIgnoreCase(searchWord)) {  
  
    count++;  
  
} } }  
  
System.out.println("The word '" + searchWord + "' appears " + count + " times in " + fileName);  
  
} catch (IOException e) {  
  
    System.out.println("Error reading " + fileName + ": " + e.getMessage());  
  
}  
  
System.out.println("ID :23DCS012_Rudra Bhavsar");  
  
} }
```

OUTPUT:

```
PS C:\Users\patel\OneDrive\Desktop\EXAM\Sem 3\JAVA\Practical\JAVA_VSCODE> javac P29.java  
PS C:\Users\patel\OneDrive\Desktop\EXAM\Sem 3\JAVA\Practical\JAVA_VSCODE> java P29 and file3.txt  
The word 'and' appears 4 times in file3.txt
```

CONCLUSION:

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

30

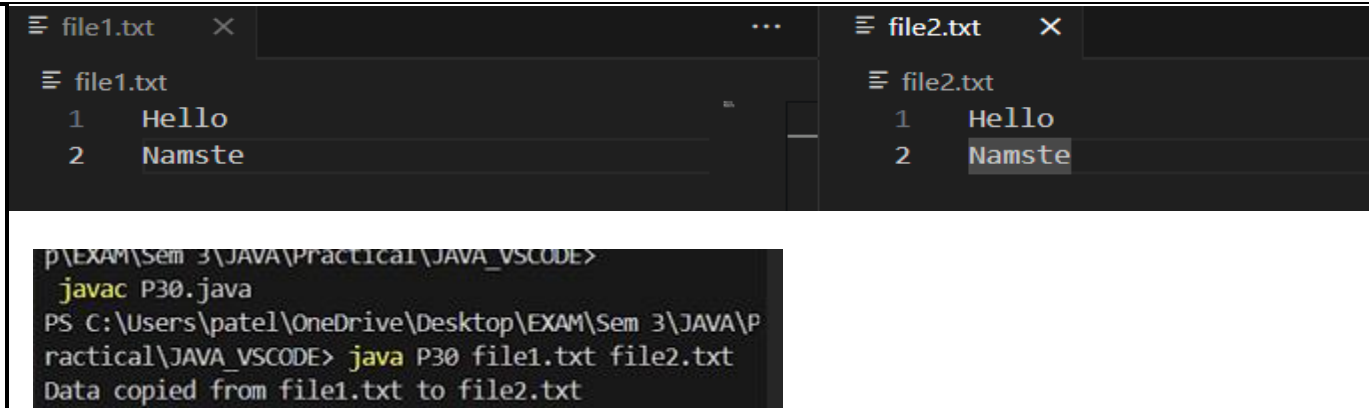
Write a program to copy data from one file to another file. If the destination file does not exist, it is created automatically.

PROGRAM CODE:

```
import java.io.FileReader;  
  
import java.io.FileWriter;  
  
import java.io.IOException;  
  
public class 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 fr = new FileReader(sourceFile);  
        FileWriter fw = new FileWriter(destinationFile)) {  
        int ch;  
        while ((ch = fr.read()) != -1) {  
            fw.write(ch);  
        }  
        System.out.println("Data copied from " + sourceFile + " to " + destinationFile);  
    } catch (IOException e) {  
        System.out.println("Error: " + e.getMessage());  
    }  
    System.out.println("ID :23DCS012_Rudra Bhavsar");  
} }
```

OUTPUT:



```

p\EXAM\Sem 3\JAVA\Practical\JAVA_VSCODE>
javac P30.java
PS C:\Users\patel\OneDrive\Desktop\EXAM\Sem 3\JAVA\Practical\JAVA_VSCODE> java P30 file1.txt file2.txt
Data copied from file1.txt to file2.txt

```

CONCLUSION:

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

31

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

PROGRAM CODE:

```

import java.io.*;

public class P31 {

    public static void main(String[] args) {

        BufferedReader consoleReader = new BufferedReader(new InputStreamReader (System.in));

        String fileName = "output.txt";

        try (BufferedWriter fileWriter = new BufferedWriter(new FileWriter(fileName))) {

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

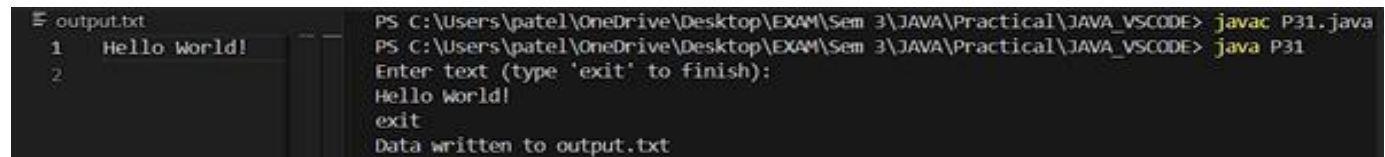
            String input;

            while (!(input = consoleReader.readLine()).equalsIgnoreCase("exit")) {

```

```
fileWriter.write(input);  
  
fileWriter.newLine();  
  
}  
  
System.out.println("Data written to " + fileName);  
  
} catch (IOException e) {  
  
System.out.println("Error: " + e.getMessage());  
  
}  
  
System.out.println("ID :23DCS012_Rudra Bhavsar");  
  
} }
```

OUTPUT:



```
PS C:\Users\patel\OneDrive\Desktop\EXAM\Sem 3\JAVA\Practical\JAVA_VSCODE> javac P31.java  
PS C:\Users\patel\OneDrive\Desktop\EXAM\Sem 3\JAVA\Practical\JAVA_VSCODE> java P31  
Enter text (type 'exit' to finish):  
Hello World!  
exit  
Data written to output.txt
```

CONCLUSION:

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

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

Department of Computer Science & Engineering

Subject Name: Java Programming**Semester:** 3rd**Subject Code:** CSE201**Academic year:** 2024-25**Part - 7**

No.	Multithreading
32.	<p>Write a program to create thread which display “Hello World” message. A. by extending Thread class B. by using Runnable interface.</p> <p><u>PROGRAM CODE:</u></p> <pre>class MyThread extends Thread { public void run() { System.out.println("Hello World"); } } public class prec32_a { public static void main(String[] args) { MyThread t1 = new MyThread(); t1.start(); } } class MyRunnable implements Runnable { public void run() { System.out.println("Hello World"); } }</pre>


```

}

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

```

OUTPUT:

```

● PS D:\prectical\java\Java> java -cp .\oaming\Code\User\workspaceSt
Hello World
○ PS D:\prectical\java\Java>

```

```

● Hello World
○ PS D:\prectical\java\Java>

```

CONCLUSION: Runnable Interface and Thread Class.

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

PROGRAM CODE:

```

import java.util.Scanner;

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

    MyThread(int start, int end) {
        this.start = start;
        this.end = end;
    }

    static void addSum(int partialSum) {
        sum += partialSum;
    }
}

```

```

public void run() {
    int partialSum = 0;
    for (int i = start; i <= end; i++) {
        partialSum += i;
    }
    addSum(partialSum);
}
}

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

        System.out.print("Enter the number 'N': ");
        N = s.nextInt();
        System.out.print("Enter the number of threads to be used (should be less than or equal
to N): ");
        numThreads = s.nextInt();

        MyThread[] threads = new MyThread[numThreads];

        int range = N / numThreads;
        int start = 1, end;

        for (int i = 0; i < numThreads; i++) {
            end = (i == numThreads - 1) ? N : start + range - 1;
            threads[i] = new MyThread(start, end);
            threads[i].start();
            start = end + 1;
        }

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

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

```

```
}
```

OUTPUT:

```
PS D:\prectical\java\Java> cd "d:\prectical\java\Java\Set-7\" ; if ($?) { javac prec33.java } ; if ($?) { java prec33 }
Enter the number 'N': 10
Enter the number of threads to be used (should be less than or equal to N): 2
The sum of numbers from 1 to 10 is: 55
```

CONCLUSION: Multithreading code to do addition of number using multiple thread.

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

PROGRAM CODE:

```
class even extends Thread {
    int n;

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

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

class odd extends Thread {
    int n;

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

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

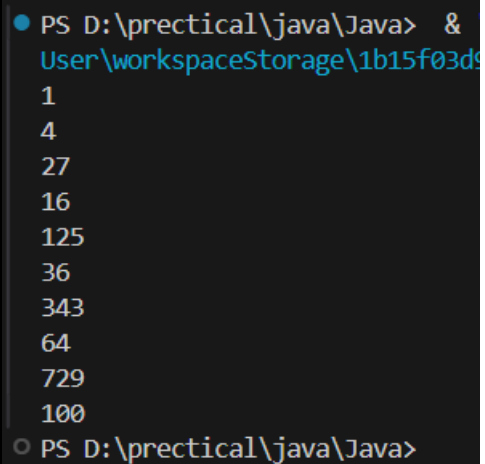
```

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

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

```

OUTPUT:



```

● PS D:\prectical\java\Java> & User\workspaceStorage\1b15f03d9
1
4
27
16
125
36
343
64
729
100
○ PS D:\prectical\java\Java>

```

CONCLUSION: Program print square if number is even and cube if number is odd.

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

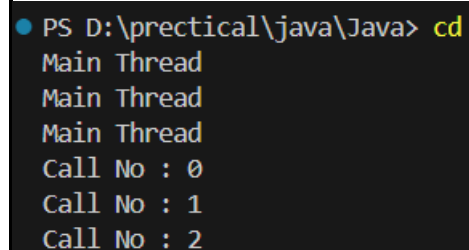
PROGRAM CODE:

```
class MyThread extends Thread {
    public void run() {
        for (int i = 0; i < 3; i++) {
            System.out.println("Call No : " + i);
            try {
                sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

public class test3 {
    public static void main(String[] args) {
        MyThread t1 = new MyThread();
        t1.start();

        for (int i = 0; i < 3; i++) {
            System.out.println("Main Thread");
        }
    }
}
```

OUTPUT:



```
PS D:\prectical\java\Java> cd
Main Thread
Main Thread
Main Thread
Call No : 0
Call No : 1
Call No : 2
```

CONCLUSION: sleep() method of class thread allow user to interrupt the execution for given interval.

36

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

PROGRAM CODE:

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

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

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

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

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

        f.setPriority(3);
        s.setPriority(5);
        t.setPriority(7);

        f.start();
        s.start();
        t.start();
    }
}
```

OUTPUT:

```
PS D:\prectical\java\Java>
oaming\Code\User\workspaces
Second
First
Third
```

CONCLUSION: priority function allow user to prioritize thread.

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

PROGRAM CODE:

```
class Produce extends Thread {
    int n;
    boolean produced = false;

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

    public synchronized void run() {
        for (int i = 1; i <= n; i++) {
            while (produced) {
                try {
                    wait();
                } catch (InterruptedException e) {
                    Thread.currentThread().interrupt();
                }
            }
            System.out.println("Produced: " + i);
            produced = true;
            notify();
        }
    }
}

class Consume extends Thread {
    int n;
```

```

Produce producer;

Consume(int n, Produce producer) {
    this.n = n;
    this.producer = producer;
}

public synchronized void run() {
    for (int i = 1; i <= n; i++) {
        synchronized (producer) {
            while (!producer.produced) {
                try {
                    producer.wait();
                } catch (InterruptedException e) {
                    Thread.currentThread().interrupt();
                }
            }
            System.out.println("Consumed: " + i);
            producer.produced = false;
            producer.notify();
        }
    }
}

}

public class prec37 {
    public static void main(String[] args) {
        int n = 10;

        Produce p = new Produce(n);
        Consume c = new Consume(n, p);

        p.start();
        c.start();
    }
}

```


OUTPUT:

```
● PS D:\prectical\java\Java> &
oaming\Code\User\workspaceStor
Produced: 1
Consumed: 1
Produced: 2
Consumed: 2
Produced: 3
Consumed: 3
Produced: 4
Consumed: 4
Produced: 5
Consumed: 5
Produced: 6
Consumed: 6
Produced: 7
Consumed: 7
Produced: 8
Consumed: 8
Produced: 9
Consumed: 9
Produced: 10
Consumed: 10
```

CONCLUSION: Synchronized function allows to execute thread in a synchronized way.

38

Design a Custom Stack using ArrayList class, which implements following functionalities of stack.

My Stack -list ArrayList<Object>: A list to store elements.

isEmpty(): boolean: Returns true if this stack is empty.

getSize(): int: Returns number of elements in this stack.

peek(): Object: Returns top element in this stack without removing it.

pop(): Object: Returns and Removes the top elements in this stack.

push(o: object): Adds new element to the top of this stack.

PROGRAM CODE:

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

```
class MyStack {
```

```
private ArrayList<Object> list = new ArrayList<>();
```

```
public boolean isEmpty() {
```

```
return list.isEmpty();
```

```
}
```

```
public int getSize() {
```

```
return list.size();
```

```
}
```

```
public Object peek() {
```

```
if (isEmpty()) {
```

```
return "Stack is empty";
```

```
}
```

```
return list.get(list.size() - 1);
```

```
}
```

```
public Object pop() {
```

```
if (isEmpty()) {  
    return "Stack is empty";  
}  
  
return list.remove(list.size() - 1);  
}  
  
public void push(Object o) {  
    list.add(o);  
} }  
  
public class P38 {  
    public static void main(String[] args) {  
        MyStack stack = new MyStack();  
        stack.push(10);  
        stack.push(20);  
        stack.push(30);  
  
        System.out.println("Top element is: " + stack.peek());  
        System.out.println("Popped element: " + stack.pop());  
        System.out.println("Popped element: " + stack.pop());  
        System.out.println("Is stack empty ? " + stack.isEmpty());  
        System.out.println("Current stack size: " + stack.getSize());  
        System.out.println("Top element now: " + stack.peek());  
    } }  
}
```

OUTPUT:

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

CONCLUSION:

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

39

Imagine you are developing an e-commerce application. The platform needs to sort lists of products based on different criteria, such as price, rating, or name. Each product object implements the Comparable interface to define the natural ordering. To ensure flexibility and reusability, you need a generic method that can sort any array of Comparable objects. Create a generic method in Java that sorts an array of Comparable objects. This method should be versatile enough to sort arrays of different types of objects (such as products, customers, or orders) as long as they implement the Comparable interface.

PROGRAM CODE:

```
import java.util.Arrays;

public class P39 {

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

        Arrays.sort(array);

    }

    public static void main(String[] args) {

        Integer[] numbers = {5, 3, 9, 1, 7};

        System.out.println("Before sorting (Integers): " + Arrays.toString(numbers));
```

```
sortArray(numbers);

System.out.println("After sorting (Integers): " + Arrays.toString(numbers));

String[] names = {"John", "Alice", "Bob", "David"};

System.out.println("\nBefore sorting (Strings): " + Arrays.toString(names));

sortArray(names);

System.out.println("After sorting (Strings): " + Arrays.toString(names));

Product[] products = {
    new Product("Laptop", 1000),
    new Product("Phone", 800),
    new Product("Tablet", 600),
    new Product("Smartwatch", 200)
};

System.out.println("\nBefore sorting (Products by price): ");

for (Product p : products) {
    System.out.println(p);
}

sortArray(products);

System.out.println("\nAfter sorting (Products by price): ");

for (Product p : products) {
    System.out.println(p);
} } }

class Product implements Comparable<Product> {
```

```
private String name;

private int price;

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

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

@Override
public String toString() {
    return name + ": $" + price;
} }
```

OUTPUT:

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

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

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

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

CONCLUSION:

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

40

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

PROGRAM CODE:

```
import java.util.*;

public class P40 {

    public static void main(String[] args) {

        Map<String, Integer> wordMap = new TreeMap<>();

        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter a text:");

        String text = scanner.nextLine();
```

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

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

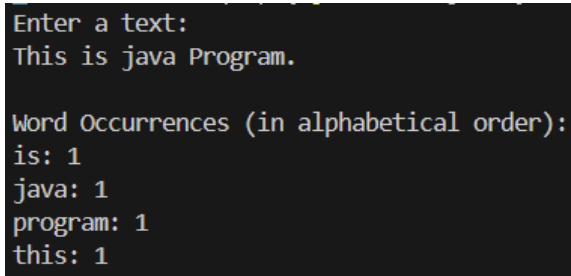
System.out.println("\nWord Occurrences (in alphabetical order):");

Set<Map.Entry<String, Integer>> entrySet = wordMap.entrySet();

for (Map.Entry<String, Integer> entry : entrySet) {

    System.out.println(entry.getKey() + ": " + entry.getValue());

} } }
```

OUTPUT:

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

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

CONCLUSION:

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

41

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

PROGRAM CODE:

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



```
import java.util.*;

public class P41 {

private static final HashSet<String> keywords = new HashSet<>();

static {

String[] keywordArray = {

"abstract", "assert", "boolean", "break", "byte", "case", "catch", "char", "class",

"const", "continue", "default", "do", "double", "else", "enum", "extends", "final",

"finally", "float", "for", "goto", "if", "implements", "import", "instanceof", "int",

"interface", "long", "native", "new", "package", "private", "protected", "public",

"return", "short", "static", "strictfp", "super", "switch", "synchronized", "this",

"throw", "throws", "transient", "try", "void", "volatile", "while"

};

for (String keyword : keywordArray) {

keywords.add(keyword);

} }

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the path of the Java source file: ");

String filePath = scanner.nextLine();

try {

File file = new File(filePath);

Scanner fileScanner = new Scanner(file);
```

```
int keywordCount = 0;

while (fileScanner.hasNext()) {
    String word = fileScanner.next();
    if (keywords.contains(word)) {
        keywordCount++;
    }
}

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

fileScanner.close();

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

OUTPUT:

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

CONCLUSION:

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

