# Experiment No: 1

**Q. Write a Program in Java to print table of given number.**

import java.lang.\*; import java.util.\*;

public class Table

{

public static void main(String[] args)

{

int No = 0, i = 1;

Scanner S = new Scanner(System.in);

System.out.print("\n Enter a Number : "); No = S.nextInt();

while( i <= 10 )

{

System.out.println(" " + No + " \* " + i + " = " + No \* i); i++;

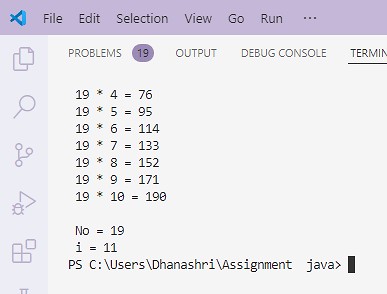
}

System.out.println("\n No = " + No + "\n i = " + i);

}

}

**OUTPUT:**



# Experiment No: 2

**Q. Write a Program in Java to print factorial of given number.**

import java.lang.\*; import java.util.\*;

class Factorial

{

public int No; private int Fact;

private Scanner scn = new Scanner(System.in);

public Factorial()

{

Fact = 1;

System.out.print("\n Enter a Number : "); No = scn.nextInt();

Find\_Factorial();

}

public Factorial(int Num)

{

No = Num; Fact = 1;

Find\_Factorial();

}

private void Find\_Factorial()

{

int Temp = No;

while ( Temp > 0 )

{

Fact \*= Temp; Temp--;

}

}

public void Display\_Factorial()

{

System.out.println("\n Factorial of Given Number " + No + " is = " + Fact + "."); System.out.print("\n Press Enter Key To Move Next Code...");

scn.nextLine();

}

}

public class Calculate\_Factorial

{

public static void main(String[] args)

{

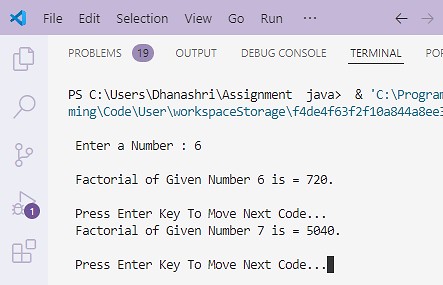
Factorial Obj1 = new Factorial(); Obj1.Display\_Factorial();

Factorial Obj2 = new Factorial(7); Obj2.Display\_Factorial();

}

}

**OUTPUT:**



# Experiment No: 3

**Q. Write a Program in Java to create console based calculator (Casestudy-1).**

import java.lang.\*; import java.util.\*;

public class Calculator

{

public static void main(String[] args)

{

int N1 = 0, N2 = 0, Res = 0, Choice = 0; Scanner S = new Scanner(System.in);

while(true)

{

System.out.print("\n============\*\*\*\*\*============\n");

System.out.print("\n \*\*\*\*\* Calculator \*\*\*\*\* \n"); System.out.print("\n Choices : "); System.out.print("\n\t 1. Addition");

System.out.print("\n\t 2. Subtraction");

System.out.print("\n\t 3. Multiplication");

System.out.print("\n\t 4. Division");

System.out.print("\n\t 5. Remainder");

System.out.print("\n\t 6. Exit");

System.out.print("\n============\*\*\*\*\*============\n"); System.out.print("\n Select Your Choice : ");

Choice = S.nextInt();

if((Choice > 0) && (Choice < 6))

{

System.out.print("\n Enter 1st Number : "); N1 = S.nextInt();

System.out.print("\n Enter 2nd Number : "); N2 = S.nextInt();

}

switch(Choice)

{

case 1:

case 2:

case 3:

case 4:

/// Add

Res = N1 + N2;

System.out.println("\n Addition of " + N1 + " & " + N2 + " is = " + Res + "."); S.next();

break;

/// Sub

Res = N1 - N2;

System.out.println("\n Subtraction of " + N1 + " & " + N2 + " is = " + Res + "."); break;

/// Mult

Res = N1 \* N2;

System.out.println("\n Multiplication of " + N1 + " & " + N2 + " is = " + Res + "."); break;

case 5:

case 6: default:

}

/// Div

Res = N1 / N2;

System.out.println("\n Division of " + N1 + " & " + N2 + " is = " + Res + "."); break;

// Rem

Res = N1 % N2;

System.out.println("\n Remainder of " + N1 + " & " + N2 + " is = " + Res + "."); break;

break;

/// Invalid

System.out.println("\n Invalid Input!!!");

if(Choice == 6)

{

break;

}

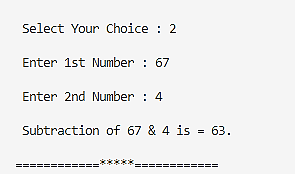
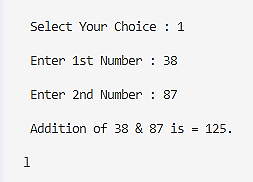
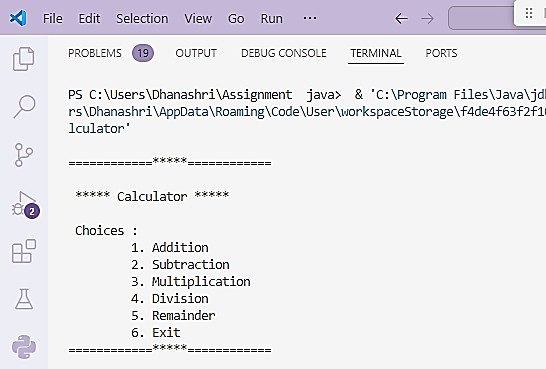
}

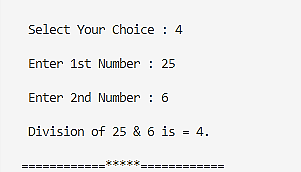
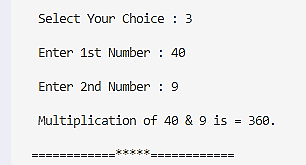
System.out.print("\n Thanks For Using this Calculator Service...\n ");

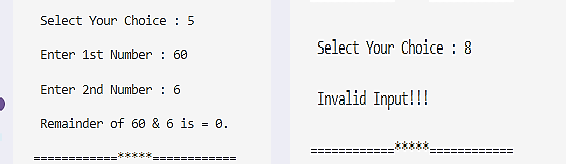
}

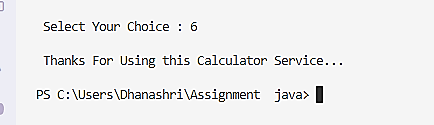
}

**OUTPUT:**









# Experiment No: 4

**Q. Write a Program in Java to demonstrate all type of constructors.**

import java.lang.\*; import java.util.\*;

class Circle

{

private float Rad; // Private Characteristic or Data Member of Class Circle public float Area, Circum; // Public Characteristics or Data Members of Class Circle

// Default Constructor public Circle()

{

Rad = Area = Circum = 0.0f;

System.out.println("\n Inside Default Constructor!!!");

}

// Parameterized Constructor public Circle(float R)

{

Rad = R;

Area = Circum = 0.0f;

System.out.println("\n Inside Parameterized Constructor!!!");

}

// Copy Constructor public Circle(Circle Ref)

{

this.Rad = Ref.Rad; this.Area = Ref.Area; this.Circum = Ref.Circum;

System.out.println("\n Inside Copy Constructor!!!");

}

// Accept Radius Member Function public void Accept\_Radius()

{

Scanner scanner = new Scanner(System.in); System.out.print("\n Enter Radius = "); this.Rad = scanner.nextFloat();

}

// Calculate Area\_Of\_Circle Member Function public void Area\_Of\_Circle()

{

Area = (float) (3.14 \* Rad \* Rad);

System.out.println("\n Area of Circle Calculated by Function as => " + Area);

}

// Calculate Circumference\_Of\_Circle Member Function public void Circumference\_Of\_Circle()

{

Circum = (float) (2 \* 3.14 \* Rad);

System.out.println("\n Circumference of Circle Calculated by Function as => " + this.Circum);

}

}

public class Circle\_Client

{

public static void main(String[] args)

{

Circle Obj1 = new Circle(); Circle Obj2 = new Circle(7.5f);

Obj1.Accept\_Radius(); Obj1.Area\_Of\_Circle(); Obj1.Circumference\_Of\_Circle();

Obj2.Area\_Of\_Circle(); Obj2.Circumference\_Of\_Circle();

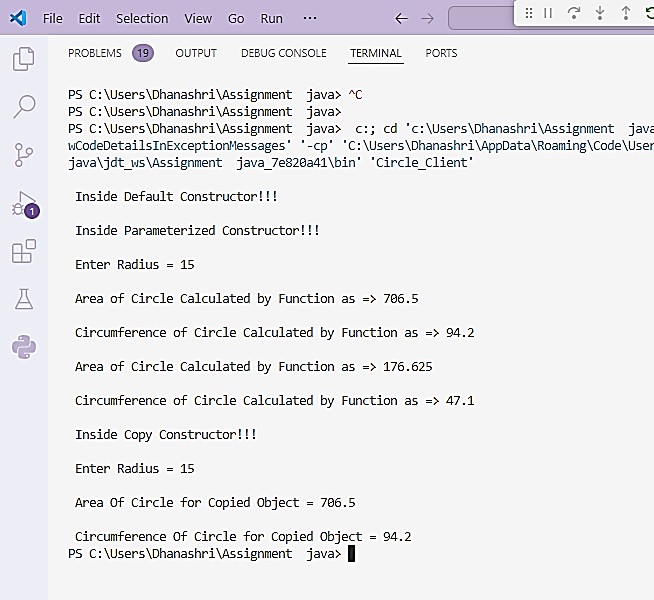
Circle Obj3 = new Circle(Obj1); Obj3.Accept\_Radius();

System.out.println("\n Area Of Circle for Copied Object = " + Obj3.Area); System.out.println("\n Circumference Of Circle for Copied Object = " + Obj3.Circum);

}

}

**OUTPUT:**



# Experiment No: 5

**Q. Write a Program in Java to find out maximum element from an array.**

import java.lang.\*; import java.util.\*;

public class MaxElementInArray

{

public static void main(String[] args)

{

int[] Numbers = {3, 5, 7, 2, 8, -1, 4}; // Sample array int MaxEle = findMax(Numbers);

System.out.println("The maximum element in the array is : " + MaxEle);

}

public static int findMax(int[] Num)

{

int Max = Num[0]; // Assume first element is the max

for (int i = 1; i < Num.length; i++)

{

if (i == 0 || Num[i] > Max)

{

Max = Num[i];

}

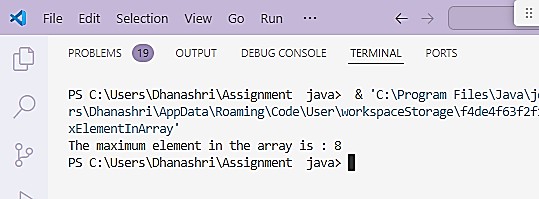
}

return Max;

}

}

**OUTPUT:**



# Experiment No: 6

**Q. Write a Program in Java to Addition of Matrix.**

**public class ArrayMatrixAddition**

**{**

**public static void main(String args[])**

**{**

**int a[][]={{1,3,4},{2,4,3},{3,4,5}};**

**int b[][]={{1,3,4},{2,4,3},{1,2,4}};**

**int c[][]=new int[3][3]; for(int i=0;i<3;i++)**

**{**

**for(int j=0;j<3;j++)**

**{**

**c[i][j]=a[i][j]+b[i][j];**

**System.out.print(c[i][j]+" ");**

**}**

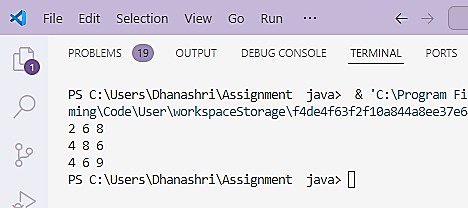
**System.out.println();**

**}**

**}**

**}**

**OUTPUT:**



# Experiment No: 7

**Q. Write a Program in Java to demonstrate arraylist .**

import java.lang.\*; import java.util.\*;

public class ArrayListExample

{

public static void main(String[] args)

{

ArrayList<String> fruits = new ArrayList<>();

fruits.add("apple"); fruits.add("banana"); fruits.add("orange");

System.out.println("Fruits in the ArrayList:"); for (String fruit : fruits)

{

System.out.println(fruit);

}

fruits.remove("banana");

System.out.println("Fruits after removing banana:"); for (String fruit : fruits)

{

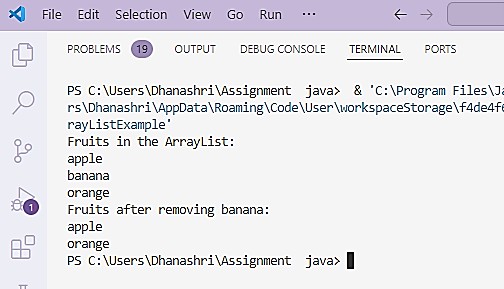
System.out.println(fruit);

}

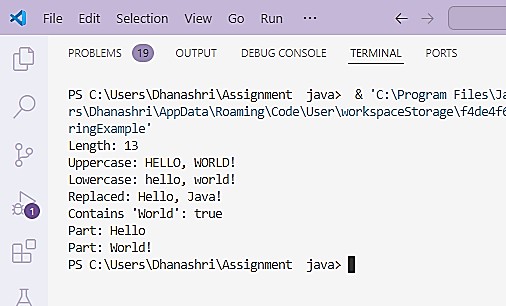
}

}

**OUTPUT:**



# Experiment No: 8



**Q. Write a Program in Java for implementation of string functions .**

import java.lang.\*; import java.util.\*;

public class StringExample

{

public static void main(String[] args)

{

String str = "Hello, World!";

// Print length of string System.out.println("Length: " + str.length());

// Convert to uppercase and lowercase System.out.println("Uppercase: " + str.toUpperCase()); System.out.println("Lowercase: " + str.toLowerCase());

// Replace substring

String newStr = str.replace("World", "Java"); System.out.println("Replaced: " + newStr);

// Check if string contains a substring System.out.println("Contains 'World': " + str.contains("World"));

// Split string

String[] parts = str.split(", ");

for (String part : parts)

{

System.out.println("Part: " + part);

}

}

}

**OUTPUT:**

# Experiment No: 9

**Q. Write a Program in Java to implement Student admission system with use of arraylist.( Casestudy-2)**

import java.lang.\*; import java.util.\*;

class Student

{

private int Roll\_No; private String Name;

private int Phy, Chem, Maths, Tot; private float Per;

private String Course;

public Student(int RNo, String Nm, int P, int C, int M, String Crs)

{

this.Roll\_No = RNo; this.Name = Nm; this.Phy = P; this.Chem = C; this.Maths = M; this.Course = Crs;

this.Calulate();

}

private void Calulate()

{

this.Tot = this.Phy + this.Chem + this.Maths; this.Per = ((float)this.Tot)/ 3;

}

@Override

public String toString()

{

return "\n Roll Number : " + Roll\_No + "\n Student Name : " + Name + ". \n Marks => Physics = " + Phy + ", Chemistry = " + Chem + ", Mathematics = " + Maths + ". \n\n Total Marks = " + Tot + ".\n Percentage = " + Per + ".\n Course : " + Course + ".\n====######====\n";

}

}

public class StudentAdmissionSystem

{

private static int RNo = 101;

private ArrayList<Student> StudentsList; private Scanner scanner;

public StudentAdmissionSystem()

{

StudentsList = new ArrayList<>(); scanner = new Scanner(System.in);

}

public void AddNewStudent()

{

Scanner scn = new Scanner(System.in);

System.out.print("\n Enter Student Details for Roll Number : " + RNo); System.out.print("\n\n Enter Student Name : ");

String SName = scanner.nextLine();

System.out.print("\n Enter Student Marks : "); System.out.print("\n Physics : ");

int P = Integer.parseInt(scanner.nextLine()); System.out.print("\n Chemistry : ");

int C = Integer.parseInt(scanner.nextLine()); System.out.print("\n Mathematics : ");

int M = Integer.parseInt(scanner.nextLine());

System.out.print("\n Enter Course Name : "); String CourseNm = scanner.nextLine();

Student NewStud = new Student(RNo, SName, P, C, M, CourseNm); StudentsList.add(NewStud);

System.out.println("\n Student Details Added Successfully!\n \n");

RNo++;

System.out.print("\n Press Enter Key To Go To Main Menu ..."); scn.nextLine();

}

public void DisplayAllStudents()

{

Scanner scn = new Scanner(System.in);

if (StudentsList.isEmpty())

{

}

else

{

System.out.println("\n No Student Added Yet.");

System.out.println("\n\n List of Students => \n"); for (Student Std : StudentsList)

{

System.out.println(Std);

}

}

System.out.print("\n Press Enter Key To Go To Main Menu ..."); scn.nextLine();

}

public void menu()

{

while (true)

{

System.out.println("\n \*\*\_\*\* Student Admission System \*\*\_\*\*\n");

System.out.println(" Choices => \n"); System.out.println(" 1. Add New Student"); System.out.println(" 2. Display Students List"); System.out.println(" 3. Exit"); System.out.print("\n Enter Choice : ");

int choice = Integer.parseInt(scanner.nextLine()); switch (choice)

{

case 1:

case 2:

AddNewStudent(); break;

DisplayAllStudents(); break;

case 3:

default:

}

}

}

System.out.println("\n Exiting the system.<\*Thanks\*>\n"); return;

System.out.println("\n Invalid option, please try again.\n");

public static void main(String[] args)

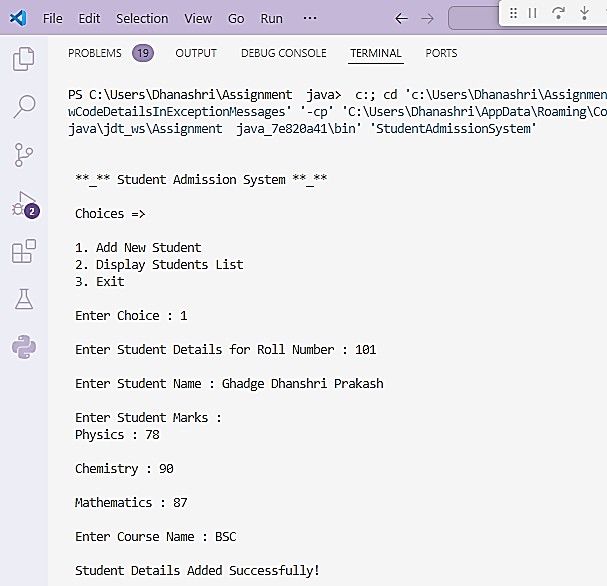
{

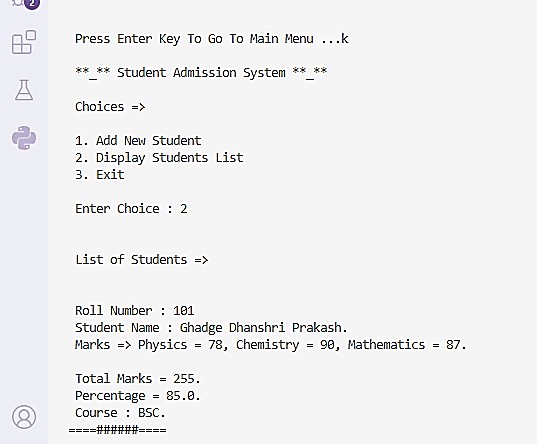
StudentAdmissionSystem system = new StudentAdmissionSystem(); system.menu();

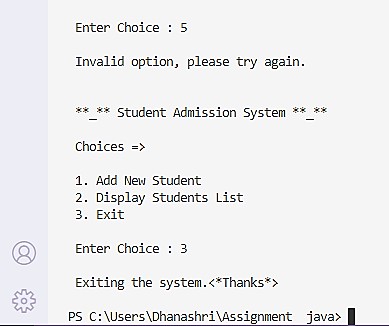
}

}

**OUTPUT:**







# Experiment No : 10

**Q. Write a Program in Java to demonstrate use of exception handeling.**

import java.lang.\*; import java.util.\*;

// Custom Exception for Insufficient Funds

class InsufficientFundsException extends Exception

{

public InsufficientFundsException(String message)

{

super(message);

}

}

// Custom Exception for Negative Amount

class NegativeAmountException extends Exception

{

public NegativeAmountException(String message)

{

super(message);

}

}

// Bank Account class

class BankAccount

{

private double balance;

public BankAccount(double initialBalance)

{

if (initialBalance < 0)

{

throw new IllegalArgumentException("Initial balance cannot be negative.");

}

this.balance = initialBalance;

}

public void deposit(double amount) throws NegativeAmountException

{

if (amount < 0)

{

throw new NegativeAmountException("Deposit amount cannot be negative.");

}

balance += amount;

System.out.println("\n Deposited: " + amount);

}

public void withdraw(double amount) throws InsufficientFundsException, NegativeAmountException

{

if (amount < 0)

{

throw new NegativeAmountException("Withdrawal amount cannot be negative.");

}

if (amount > balance)

{

throw new InsufficientFundsException("Insufficient funds for this withdrawal.");

}

balance -= amount;

System.out.println("\n Withdrew: " + amount);

}

public double getBalance()

{

return balance;

}

}

// Main class

public class BankApp

{

public static void main(String[] args)

{

BankAccount account = new BankAccount(1000);

try

{

}

account.deposit(500); account.withdraw(200);

account.withdraw(1500); // This will cause InsufficientFundsException

catch (InsufficientFundsException | NegativeAmountException e)

{

}

try

{

}

System.out.println("\n Exception: " + e.getMessage());

account.deposit(-100); // This will cause NegativeAmountException

catch (NegativeAmountException e)

{

System.out.println("Exception: " + e.getMessage());

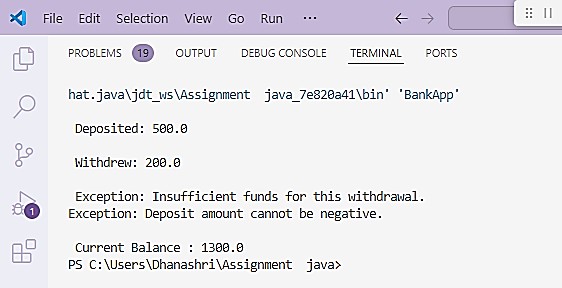
}

System.out.println("\n Current Balance : " + account.getBalance());

}

}

**OUTPUT:**



# Experiment No : 11

## Q.Write a Program in Java to demonstrate Multilevel Inheritance.

**class Shape**

**{**

**public void display()**

**{**

**System.out.println("Inside display");**

**}**

**}**

**class Rectangle extends Shape**

**{**

**public void area()**

**{**

**System.out.println("Inside area");**

**}**

**}**

**class Cube extends Rectangle**

**{**

**public void volume()**

**{**

**System.out.println("Inside volume");**

**}**

**}**

**public class MultilevelInheritance**

**{**

**public static void main(String[] arguments)**

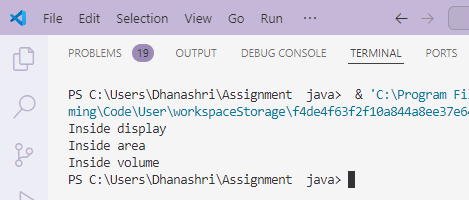
**{**

**Cube cube = new Cube(); cube.display(); cube.area(); cube.volume();**

**}**

**}**

**OUTPUT:**



# Experiment No: 12

**Q. Write a Program in Java to demonstrate Hierarchical Inheritance.**

import java.lang.\*; import java.util.\*;

// Superclass

class Animal

{

void eat()

{

System.out.println("This animal eats food.");

}

}

// Subclass 1

class Dog extends Animal

{

void bark()

{

System.out.println("The dog barks.");

}

}

// Subclass 2

class Cat extends Animal

{

void meow()

{

System.out.println("The cat meows.");

}

}

public class Animals\_Test

{

public static void main(String[] args)

{

Dog dog = new Dog(); Cat cat = new Cat(); dog.eat();

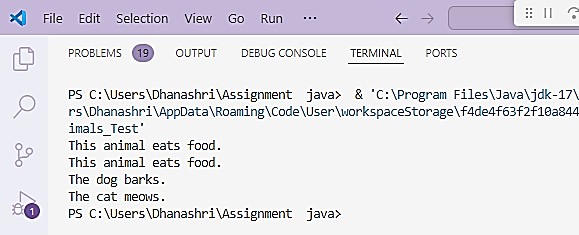
cat.eat();

dog.bark(); .

cat.meow();

}

}

**OUTPUT** :

# Experiment No: 13

**Q. Write a Program in Java to demonstrate use of interface.**

import java.lang.\*; import java.util.\*; import java.io.\*;

interface Vehicle

{

}

// All Abstract Methods. void changeGear(int a); void speedUp(int a); void applyBrakes(int a);

class Bicycle implements Vehicle

{

int speed; int gear;

// to change gear @Override

public void changeGear(int newGear)

{

gear = newGear;

}

// to increase speed @Override

public void speedUp(int increment)

{

speed = speed + increment;

}

// to decrease speed @Override

public void applyBrakes(int decrement)

{

speed = speed - decrement;

}

public void printStates()

{

System.out.println("speed: " + speed + " gear: " + gear);

}

class Bike implements Vehicle

{

int speed; int gear;

// to change gear @Override

public void changeGear(int newGear)

{

gear = newGear;

}

// to increase speed @Override

public void speedUp(int increment)

{

speed = speed + increment;

}

// to decrease speed @Override

public void applyBrakes(int decrement)

{

speed = speed - decrement;

}

public void printStates()

{

System.out.println("speed: " + speed + " gear: " + gear);

}

}

class Interface\_Client

{

public static void main (String[] args)

{

// Creating an Object of Bicycle Bicycle bicycle = new Bicycle(); bicycle.changeGear(2); bicycle.speedUp(3); bicycle.applyBrakes(1);

System.out.println("\n Bicycle present state : "); bicycle.printStates();

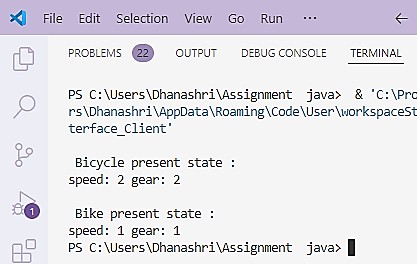
// Creating Object of the bike. Bike bike = new Bike(); bike.changeGear(1); bike.speedUp(4); bike.applyBrakes(3);

}

}

**OUTPUT:**

System.out.println("\n Bike present state : "); bike.printStates();



# Experiment No: 14

**Q. Write a Program in Java to Designing and using Thread class.**

**A. Using the Thread Class**

// Custom Thread class

class MyThread extends Thread

{

@Override public void run()

{

for (int i = 1; i <= 5; i++)

{

System.out.println("Thread: " + i); try

{

Thread.sleep(500); // Sleep for 500 milliseconds

}

catch (InterruptedException e)

{

System.out.println("Thread interrupted: " + e.getMessage());

}

}

}

}

// Main class

public class ThreadExample

{

public static void main(String[] args)

{

MyThread thread = new MyThread(); // Create a new thread thread.start(); // Start the thread

// Main thread printing numbers for (int i = 1; i <= 5; i++)

{

System.out.println("Main: " + i); try

{

Thread.sleep(300); // Sleep for 300 milliseconds

}

catch (InterruptedException e)

{

System.out.println("Main thread interrupted: " + e.getMessage());

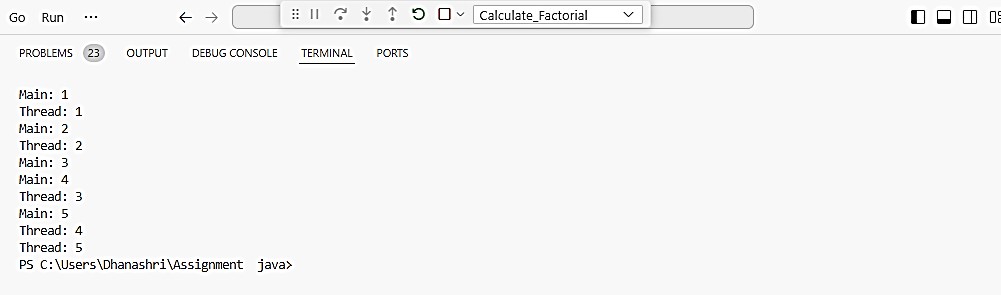
}

}

}

}

**OUTPUT:**



**B Using the Runnable Interface**

// Custom Runnable class

class MyRunnable implements Runnable

{

@Override public void run()

{

for (int i = 1; i <= 5; i++)

{

System.out.println("Runnable: " + i); try

{

Thread.sleep(500); // Sleep for 500 milliseconds

}

catch (InterruptedException e)

{

System.out.println("Runnable interrupted: " + e.getMessage());

}

}

}

}

// Main class

public class RunnableExample

{

public static void main(String[] args)

{

MyRunnable myRunnable = new MyRunnable(); // Create a new Runnable Thread thread = new Thread(myRunnable); // Create a thread using Runnable thread.start(); // Start the thread

// Main thread printing numbers for (int i = 1; i <= 5; i++)

{

System.out.println("Main: " + i); try

{

Thread.sleep(300); // Sleep for 300 milliseconds

}

catch (InterruptedException e)

{

System.out.println("Main thread interrupted: " + e.getMessage());

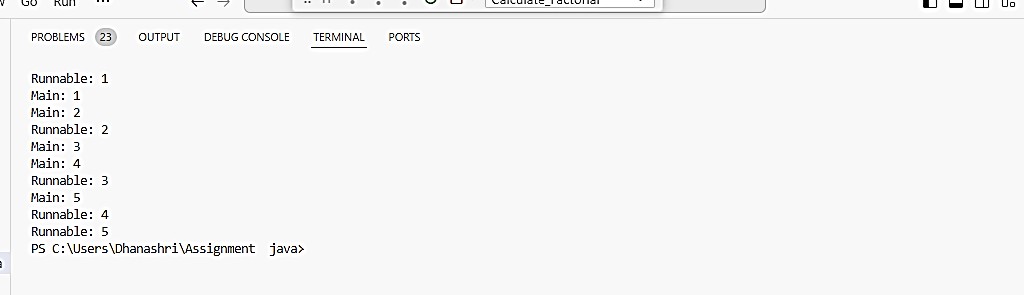
}

}

}

}

**OUTPUT:**



# Experiment No :15

**Q. Write a Program in Java to Using readers and writers to write data into Files.**

1. **Writing Data to a File**

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

public class FileWrite

{

public static void main(String[] args)

{

String filename = "example.txt";

// Data to be written to the file String[] data = {

"Hello, World!",

"Welcome to Java File I/O.", "This is a simple example.", "Goodbye!"

};

try (BufferedWriter writer = new BufferedWriter(new FileWriter(filename)))

{

for (String line : data)

{

writer.write(line);

writer.newLine(); // Write a new line after each entry

}

System.out.println("Data written to the file successfully.");

}

catch (IOException e)

{

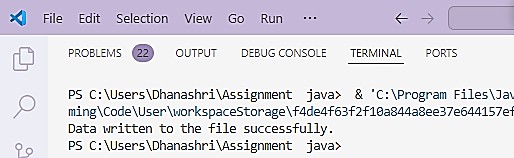
System.out.println("An error occurred while writing to the file: " + e.getMessage());

}

}

}

**OUTPUT:**



## Reading Data from a File

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

public class FileRead

{

public static void main(String[] args)

{

String filename = "example.txt";

try (BufferedReader reader = new BufferedReader(new FileReader(filename)))

{

String line;

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

{

System.out.println(line); // Print each line read from the file

}

}

catch (IOException e)

{

System.out.println("An error occurred while reading the file: " + e.getMessage());

}

}

}

**OUTPUT:**

