**23CSE111**

**OBJECT-ORIENTED PROGRAMMING**

**LAB REPORT**

****

**Department of Computer Science Engineering**

**Amrita School of Computing**

**Amrita Vishwa Vidyapeetham, Amaravati Campus**

**VERIFIED BY: NAME: B. Bharath Naik**

**ROLL NO: AV.SC.U4CSE24030**

**WEEK 01**

**PROGRAM-1:**

**AIM:** Download and Install Java Software

**PROCEDURE:**

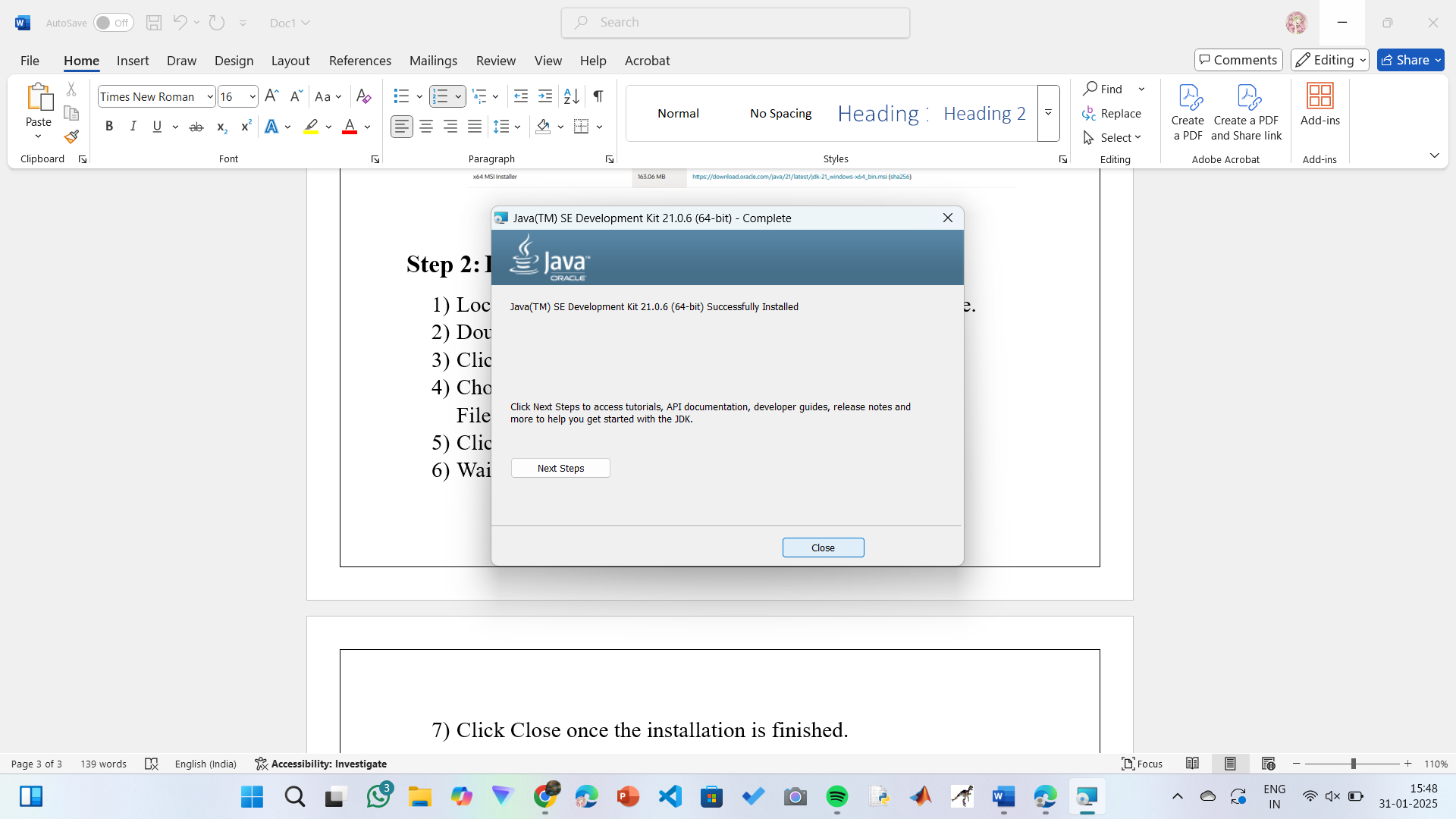
**Step 1: Download JDK 21**

1. Open your web browser and go to the Oracle JDK Downloads page
2. Scroll down to the Java SE Development Kit 21 section.
3. Choose the Windows x64 Installer version.
4. Click on Download, then Wait for the download to complete**.**



**Step 2:** **Install JDK 21**

1. Locate the downloaded jdk-21\_windows-x64\_bin.exe file.
2. Double-click to launch the installer.
3. Click Next on the setup wizard.
4. Choose the installation path (default is C:\Program Files\Java\jdk-21).
5. Click Next, then click Install.
6. Wait for the installation to complete.
7. Click Close once the installation is finished.

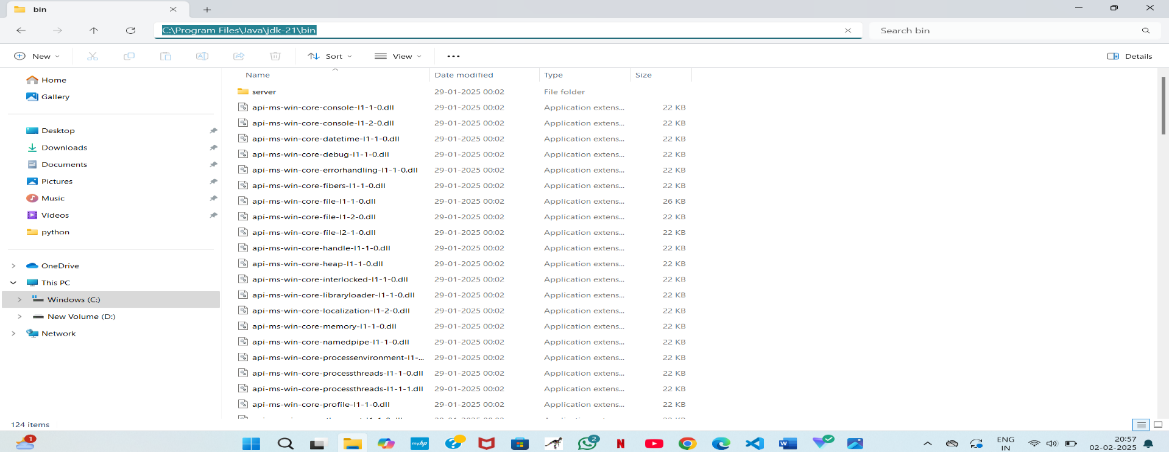


**Step 3: Setting up the path**

1) Go to “Windows C” Drive on Desktop

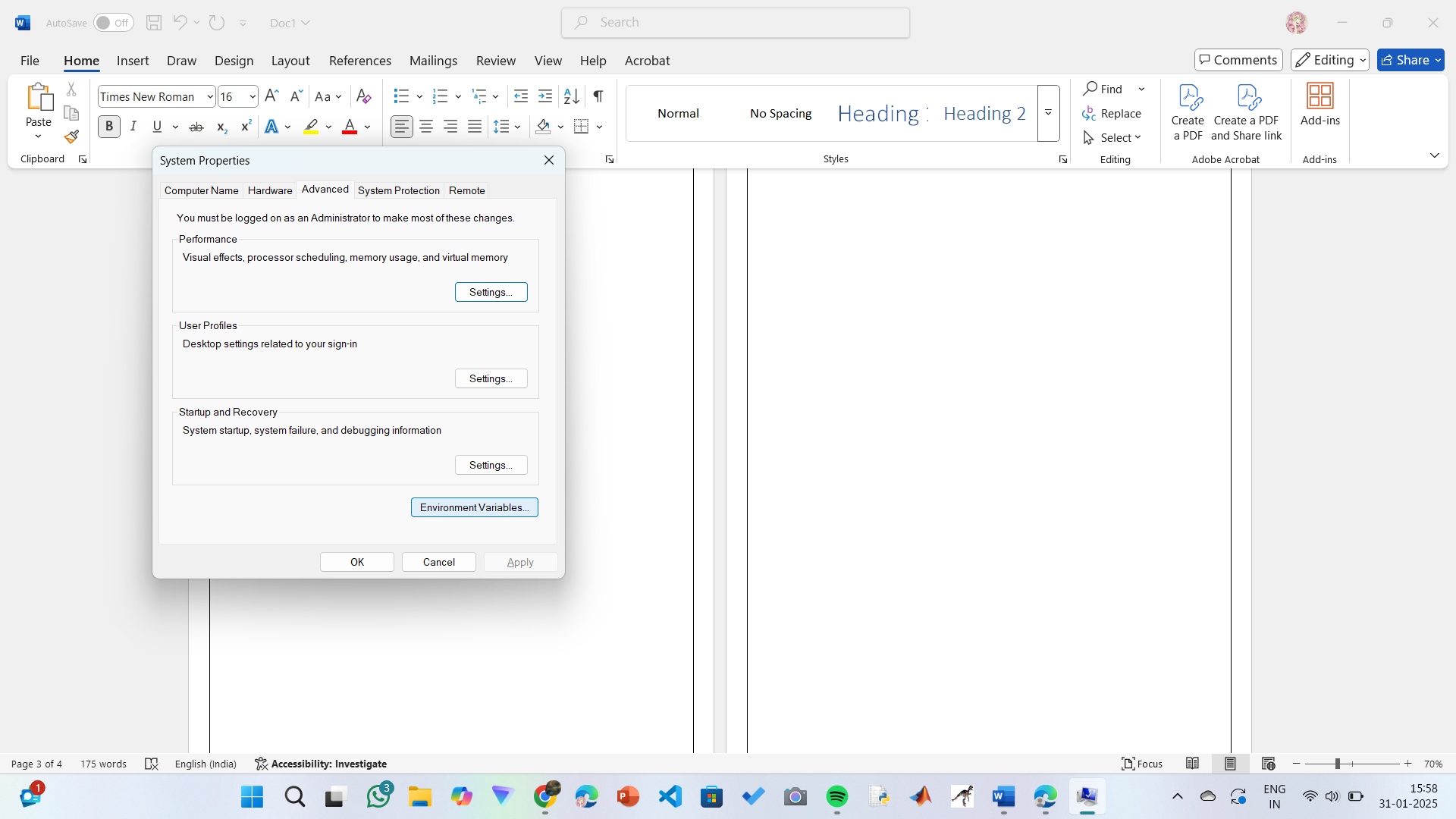
2) Choose Program Files, select Java, then JDK 21, then select Bin.

3) Select and copy the path at the address bar.



**Step 4: Open System Properties**

1. Press Windows + R, type sysdm.cpl, and click Ok-
2. The System Properties window will open.
3. Navigate to the Advanced tab.
4. Click on Environment Variables at the bottom.



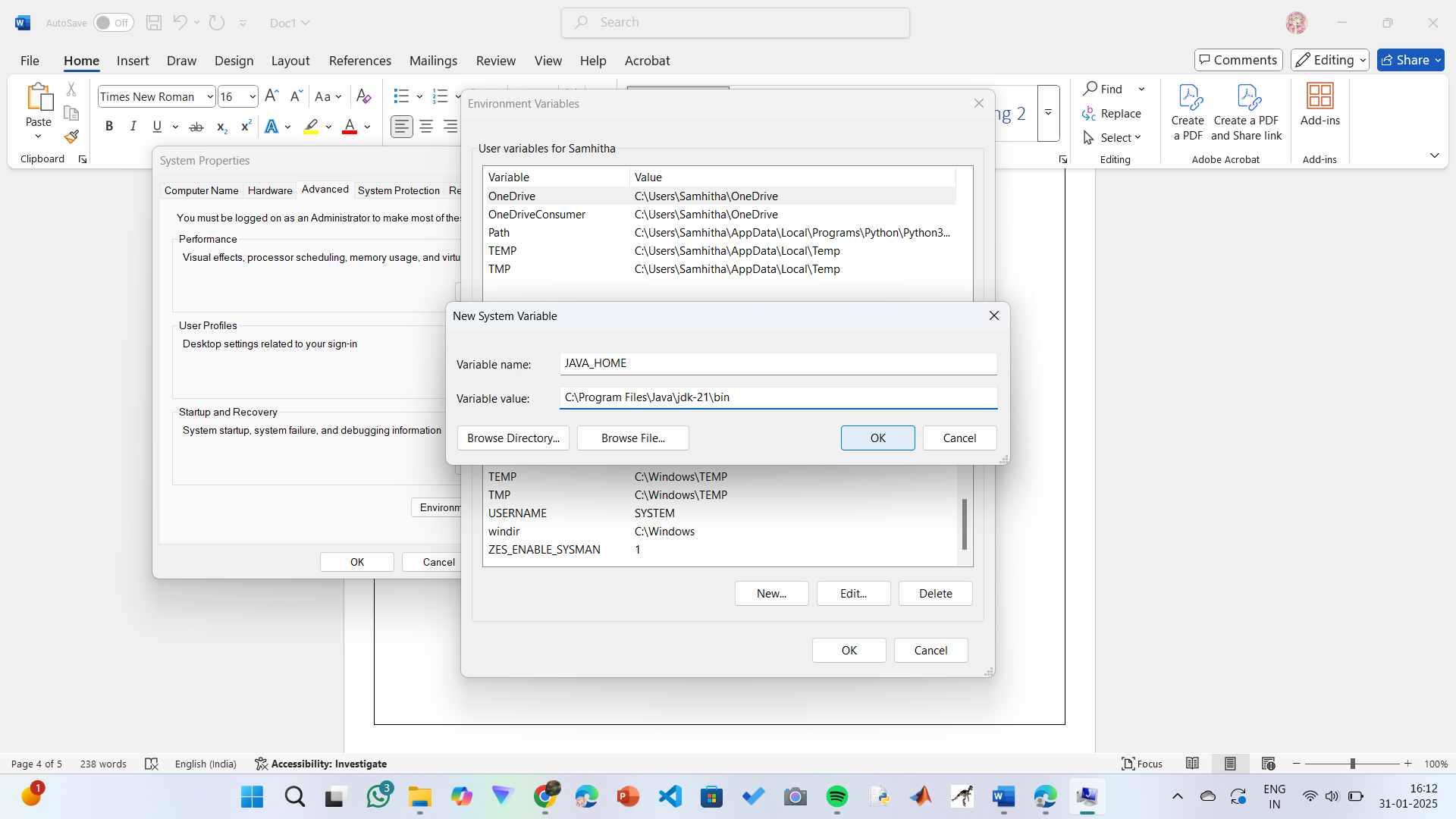
**Step 5: Set JAVA\_HOME**

1)Under System Variables, click New.

2)Set the Variable name as JAVA\_HOME.

3)Set Variable value as C:\Program Files\Java\jdk-21 (or your installation path).

4)Click OK.

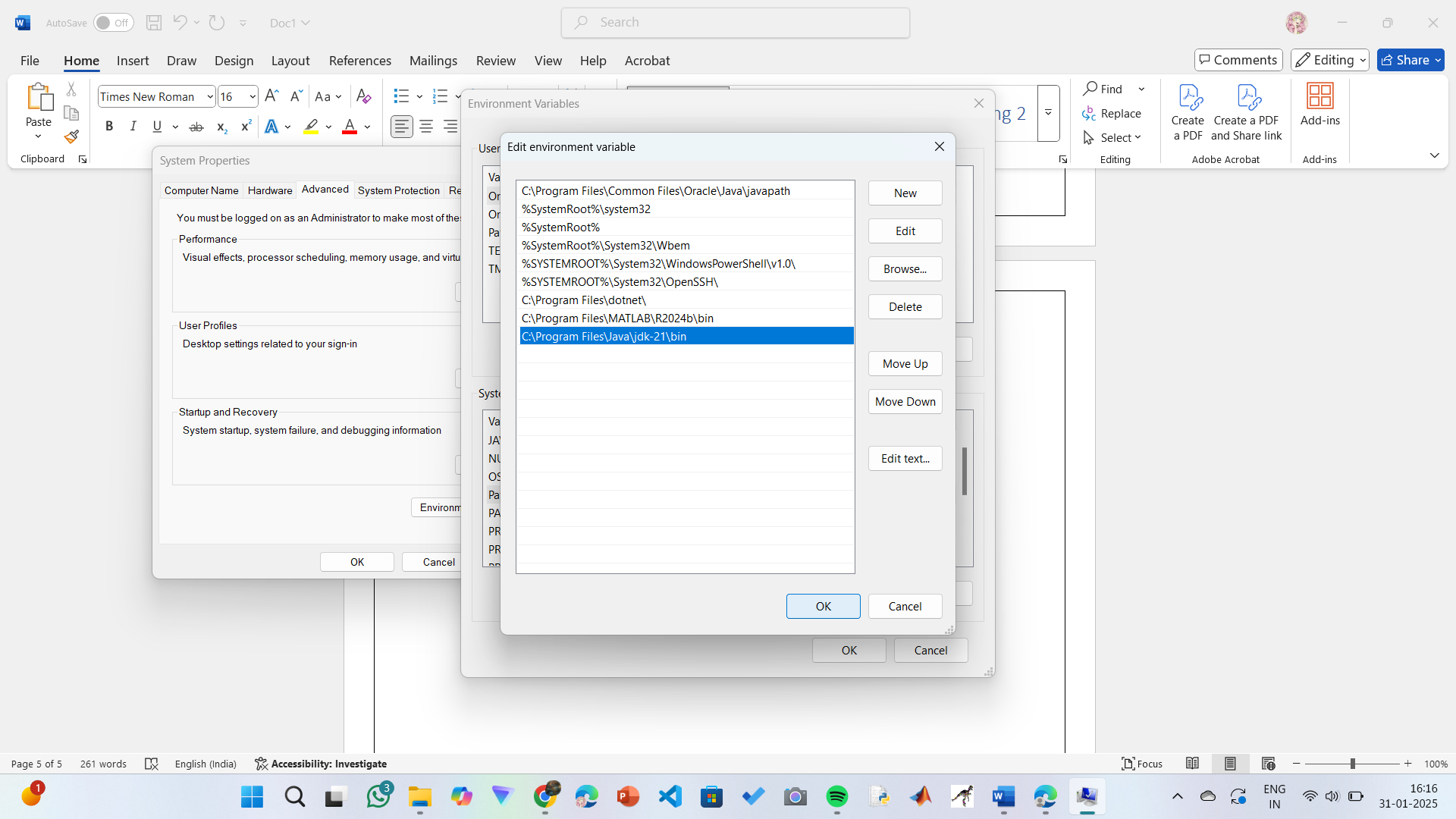


**Step 6: Update PATH Variable**

1)In System Variables, find Path and click Edit.

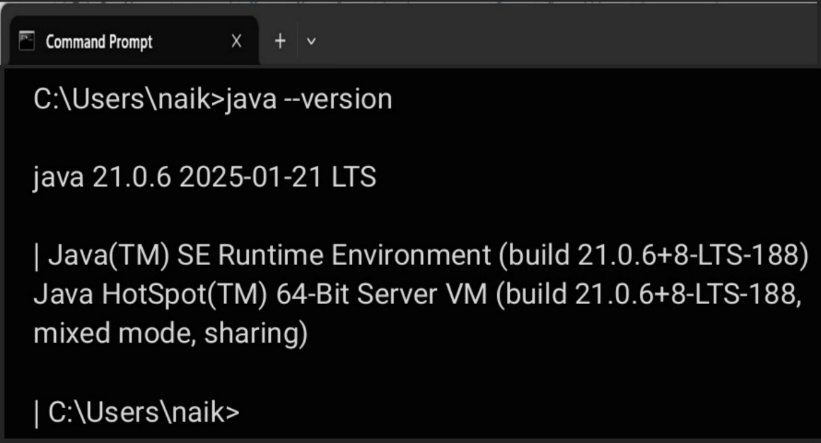
2)Click New and add: C:\Program Files\Java\jdk-21\bin

3)Click OK to save.

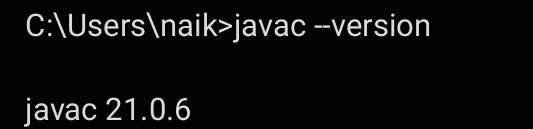


**Step 7:Verify Installation**

1. Open Command Prompt.
2. Type the following command: **java --version** and press Enter.



1. To check the java compiler type: **javac –version.**



**PROGRAM-2:**

**AIM:** Write a Java program to print the message “Welcome to Java Programming.”

**CODE:**

class example1{

public static void main (String [] args) {

System. out. Println ("Welcome to Java Programming");

}

}

**OUTPUT:**

****

**ERRORS:** None found

**IMPORTANT POINTS:**

1. Make sure that the file and the class name are the same to avoid confusion.

**PROGRAM-3:**

**AIM:** Write a Java Program that prints Name, Roll No, Section of a student.

**CODE:**

class example2{

public static void main (String [] args) {

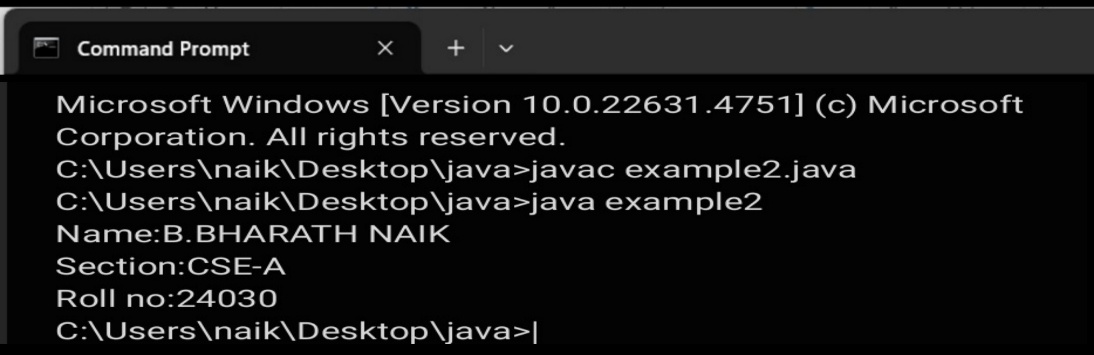
System.out.println("Name: B. BHARRATH NAIK");

System.out.println("Section: CSE-A");

System.out.println("Roll no:24030");

}}

**OUTPUT:**

****

**IMPORTANT POINTS:**

1. When printing the statements, everything should be inside double quotes.

**ERRORS:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. writing small “S” in place of ”S”  In system.out.println()  2.not giving strings to the name and section | 1. code is rectified by keeping capital “S”  2. Giving strings to name and section |

**WEEK 02**

**PROGRAM-1:**

**AIM:** Write a Java program to calculate area of rectangle.

**CODE:**

import java.util.Scanner;

class Area {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Enter length: ");

float l = input.nextFloat();

System.out.print("Enter width: ");

float b = input.nextFloat();

input.close();

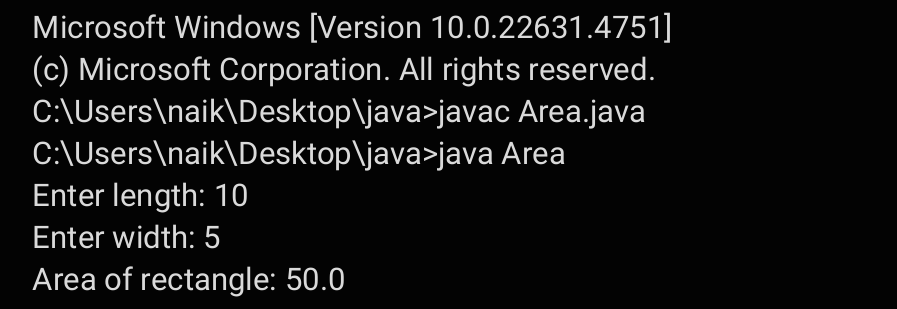
float area = l \* b;

System.out.println("Area of rectangle: " + area);

}

}

**OUTPUT:**



**ERRORS:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. While using for iteration, not giving the conditions correctly. 2. Declaring the data type as double instead of int. | 1. We should give iterative statements correctly. 2. We should give the data type as int for integers. |

**IMPORTANT POINTS:**

1. Area of a rectangle is area = l\*b, where

L = length of a side of the rectangle,

B= breadth of a side of the rectangle.

1. Here, we must be sure that all the expressions/conditions inside for the for loop must be given correctly.

**PROGRAM-2:**

**AIM:** Write a Java program to convert temperature from Fahrenheit to Celsius and vice versa.

**CODE:**

import java.util.Scanner;

class Temperature {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Enter Temperature in Fahrenheit: ");

float F = input.nextFloat();

input.close();

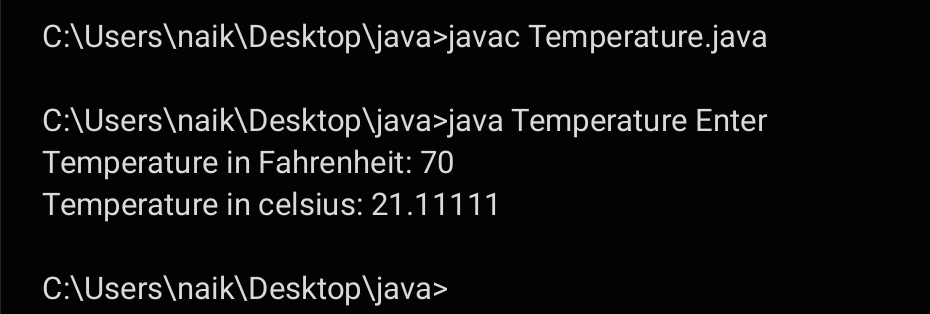
float C = (F - 32)\*5/9;

System.out.println("Temperature in celsius: " + C);

}

}

**OUTPUT:**



**CODE:**

import java.util.Scanner;

class temperature2 {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Enter Temperature in Celsius: ");

float C = input.nextFloat();

input.close();

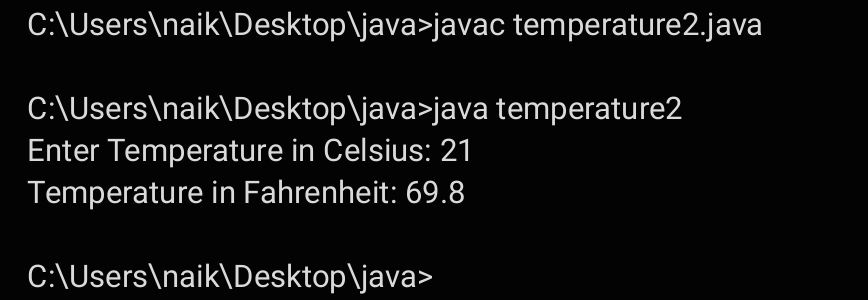
float F = (C \* 9/5) + 32;

System.out.println("Temperature in Fahrenheit: " + F);

}

}

**OUTPUT:**

****

**IMPORTANT POINTS**:

1. The formula to convert a Fahrenheit to Celsius is

Celsius = (Fahrenheit-32)\*5/9

1. The formula to convert a Celsius to Fahrenheit is

Fahrenheit = (Celsius\*9/5)+32.

1. The line “Scanner input = new Scanner(System.in),” tends to create a new Scanner object named “input” that reads input from the standard input stream (System.in), like keyboard.

**ERRORS:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. While printing the variable not giving + sign. 2. Not closing the scanner. | 1. We should give correct indentation. 2. Closing the scanner is must. |

**PROGRAM-3:**

**AIM:** Write a Java program to calculate simple intrest.

**CODE:**

import java.util.Scanner;

class simple\_interest {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Enter principle: ");

float P = input.nextFloat();

System.out.print("Enter time: ");

float T = input.nextFloat();

System.out.print("Enter rate: ");

float R = input.nextFloat();

input.close();

float SI = (P\*T\*R)/100;

System.out.println("simple\_interest : " + SI );

}

}

**OUTPUT:**

****

**ERRORS:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Giving space between next and Double. 2. Not giving parenthesis after closing the input. | 1. Should not give space between next and Double. 2. We must put parenthesis after closing the input. |

**IMPORTANT POINTS:**

1. Simple interest formula is: (p\*t\*r)/100, where:

P: Principal amount

R: Rate of interest

T: Time period

1. The data type double indicates the floating points in the integers.
2. The line “import java.util.Scanner” indicates:

Import: tells the java compiler that we want to use

specific or packagein your code**.**

Java.util : This is the package that contains utility classes

for Java programming, including the “Scanner” class.

Scanner: this is the class that allows you to read input

from the Keyboard.

**PROGRAM-4:**

**AIM:** Write a Java program to calculate largest of 3 numbers using ternary operators.

**CODE:**

import java.util.Scanner;

class largest\_num {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Enter number1: ");

float n1 = input.nextFloat();

System.out.print("Enter number2: ");

float n2 = input.nextFloat();

System.out.print("Enter number3: ");

float n3 = input.nextFloat();

input.close();

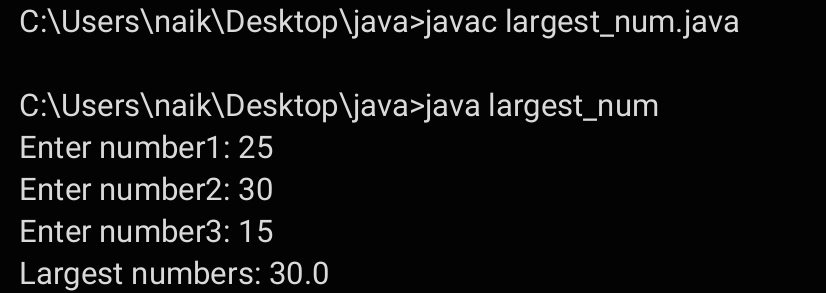
float largest = (n1 >= n2) ? ((n1 >= n3) ? n1 : n3) : ((n2 >= n3) ? n2 : n3);

System.out.println("Largest numbers: " +largest);

}

}

**OUTPUT:**

****

**ERRORS:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Not giving + before the result variable to be printed.  2. Not giving question mark after the condition in ternary operators. | 1. We pust put + before the variable name that must be printed.  2. We must put question mark to check the conditions. |

**PROGRAM-5:**

**AIM:** Write a Java program to calculate factorial of a number.

**CODE:**

import java.util.Scanner;

class factorial {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Enter number: ");

int n = input.nextInt();

input.close();

long factorial = 1;

for (int i = 1; i <= n; i++) {

factorial \*= i;

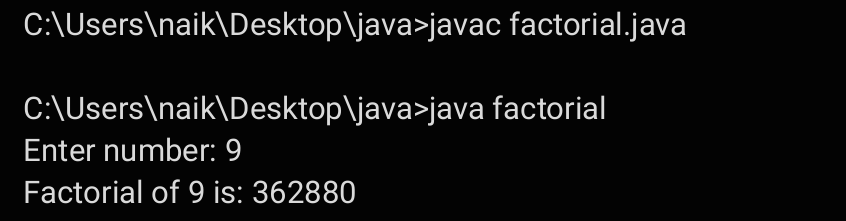
}

System.out.println("Factorial of " + n + " is: " + factorial);

}

}

**OUTPUT:**



**ERRORS:**

|  |  |
| --- | --- |
| 1. While using for iteration, not giving the conditions correctly.   Declaring the data type as double instead of int. | 1. We should give iterative statements correctly. 2. We should give the data type as int for integers. |

**IMPORTANT POINTS:**

1. While the for loop the data inside the parenthesis indicates the Initial expression.

Test expression and

Update expression.

1. Here “factorial\*=I” means factorial = factorial\*I.
2. Here we are using the data type “int” just to calculate the integer values and it doesn’t support floating points.

**WEEK 03**

**PROGRAM-1:**

**AIM:** To create java program with following instructions :

1. Create a class with name Car

2. Create four attributes named car\_color, car\_brand, fuel\_type, mileage

3. Create these methods named start(),stop(),service()

4. Create the objects named car, car1,car2.

**CODE:**

class Car {

private String carColor;

private String carBrand;

private String fuelType;

private int mileage;

public Car(String carColor, String carBrand, String fuelType, int mileage) { // Constructor

this.carColor = carColor;

this.carBrand = carBrand;

this.fuelType = fuelType;

this.mileage = mileage;

}

public void displayCarDetails() { // Display Car Details

System.out.println("Car Details: " + this.carBrand +

" | Color: " + this.carColor +

" | Fuel Type: " + this.fuelType +

" | Mileage: " + this.mileage + " km/l");

System.out.println(); }

public void start() { // Method to start the car

System.out.println(this.carBrand + " is starting..."); }

public void service() { // Method to service the car

System.out.println("Your car is at our service.");

System.out.println("Car Color: " + this.carColor);

System.out.println("Fuel Type: " + this.fuelType);

System.out.println("Mileage: " + this.mileage + " km/l") }

public void stop() { // Method to stop the car

System.out.println(this.carBrand + " is stopping...\n") }

public String getCarColor() { return carColor; } // Getter Methods

public String getCarBrand() { return carBrand; }

public String getFuelType() { return fuelType; }

public int getMileage() { return mileage; }

public static void main(String[] args) { // Creating Car objects

Car car1 = new Car("Red", "Maruti", "Diesel", 20);

System.out.println("Welcome to the Car Garage!");

car1.displayCarDetails();

car1.start();

car1.service();

car1.stop();

Car car2 = new Car("Black", "Mercedes", "Diesel", 40);

car2.displayCarDetails();

car2.start();

car2.service();

car2.stop();

Car car3 = new Car("White", "Mahindra", "Diesel", 30);

car3.displayCarDetails();

car3.start();

car3.service();

car3.stop();

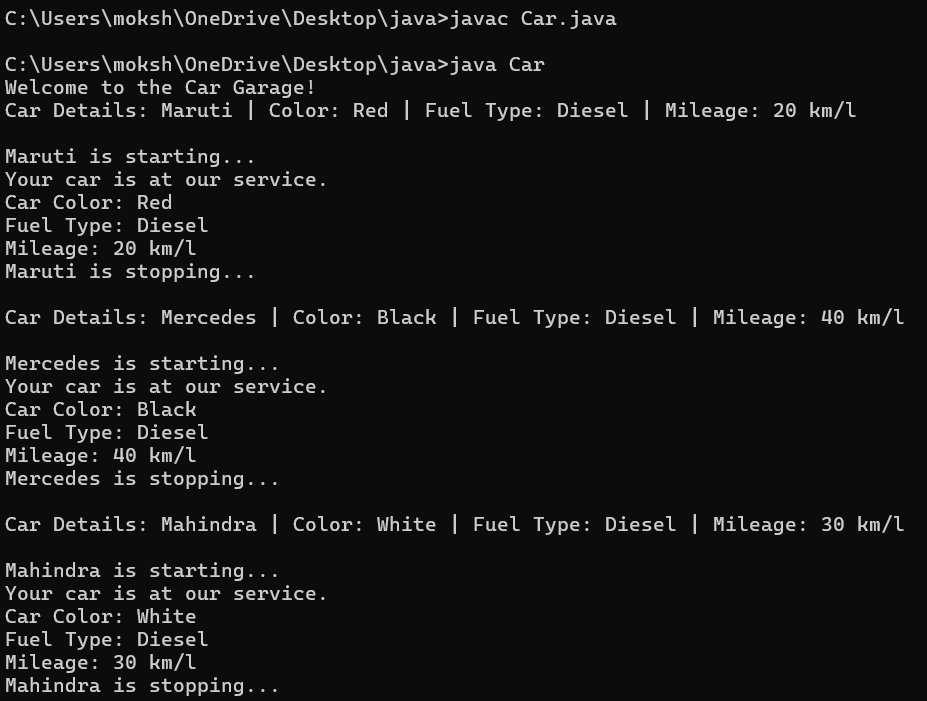
}

}

**IMPORTANT POINTS:**

1. While defining two classes for a code, we must be sure that we save both the classes in separate files.
2. While defining a method we should also define a function to call that method.

**OUTPUT:**

**ERRORS:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Not putting the semi-colon; after calling the function. 2. After Start, Stop, Service not giving the parenthesis ( ). | 1. Put the semi-colon after the writing the code. 2. After every method, put the parenthesis ( ). |

**CLASS DIAGRAM**

|  |
| --- |
| **car** |
| car\_color:string  car\_brand:string  fuel\_type:string  -milage:double |
| +start():void  +stop():void  +service():void |

**PROGRAM-2:**

**AIM:** To create a class BankAccount with methods deposit() and withdraw() . create two subclasses savingsaccount and checkingaccount override the withdraw () method in each subclass to impose different withdrawal limits and fees.

**CODE:**

class BankAccount {

private String name;

private int accNo;

private int currBal;

public BankAccount(String name, int accNo, int currBal) {

// CONSTRUCTOR

this.name = name;

this.accNo = accNo;

this.currBal = currBal;

displayDetails();

}

private void displayDetails() {

// DISPLAY CUSTOMER DETAILS

System.out.println("Customer Details: " + name + ", Account No: " + accNo + ", Balance: " + currBal);

System.out.println();

}

public void withdraw(int withdrawAmount) {

// WITHDRAW METHOD

if (withdrawAmount <= currBal) {

currBal -= withdrawAmount;

System.out.println("Withdrawal successful. New balance: " + currBal);

} else {

System.out.println("Insufficient funds! Cannot withdraw " + withdrawAmount);

}

}

public int deposit(int depositAmount) { // DEPOSIT METHOD

currBal += depositAmount;

return currBal;

}

public static void main(String[] args) { // OBJECT

BankAccount cust1 = new BankAccount("Ram", 5587, 20000);

cust1.withdraw(5000);

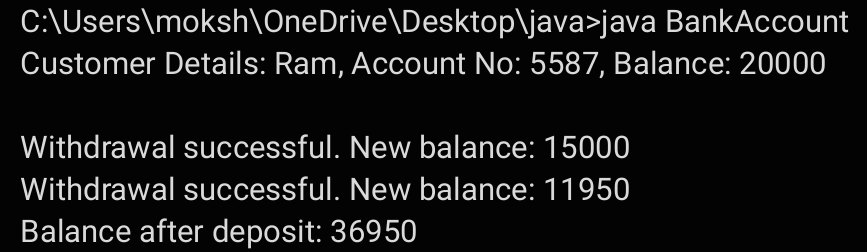
cust1.withdraw(3050);

System.out.println("Balance after deposit:”cust1.deposit(25000));

}

}

**OUTPUT:**



**ERRORS:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Not putting the semi-colon; after calling the function. 2. After Withdrawal,deposit not giving the parenthesis ( ). | 1. Put the semi-colon after the writing the code. 2. After every method, put the parenthesis ( ). |

**IMPORTANT POINTS:**

1. The condition inside the if statement must be correct.
2. It explains that if the withdrawal money is less than the money in the bank account, then we can withdraw the amount.

**CLASS DIAGRAM**:

|  |
| --- |
| **Bank Account** |
| * Balance : double |
| + Bank Account(initialBalance:double)  +deposit(amount:double):void  +withdraw(amount:double):void |

**WEEK 04**

**PROGRAM-1:**

**AIM:** Write a java program with class named “book”, the class should contain various attributes such as title, author, year of publication it should also contain a constructor with parameters which initializes, title, author, and year of publication.

Create a method which displays the details of the book and display the details of two books.

**CODE:**

class Book { // Declaring attributes

String Title\_of\_the\_book;

String Author;

int Year\_of\_publication; // Constructor to initialize values

Book(String Title\_of\_the\_book, String Author, int Year\_of\_publication) {

this.Title\_of\_the\_book = Title\_of\_the\_book;

this.Author = Author;

this.Year\_of\_publication = Year\_of\_publication;

System.out.println("Your book: " + this.Title\_of\_the\_book);

} // Creating a method

public void getbook() {

System.out.println("The Title of the Book: " + Title\_of\_the\_book);

System.out.println("Author: " + Author);

System.out.println("Published Year: " + Year\_of\_publication);

}

public static void main(String[] args)

// Creating objects for class Book

Book book1 = new Book("Harry Potter", "J.K.Rowling", 1997);

book1.getbook();

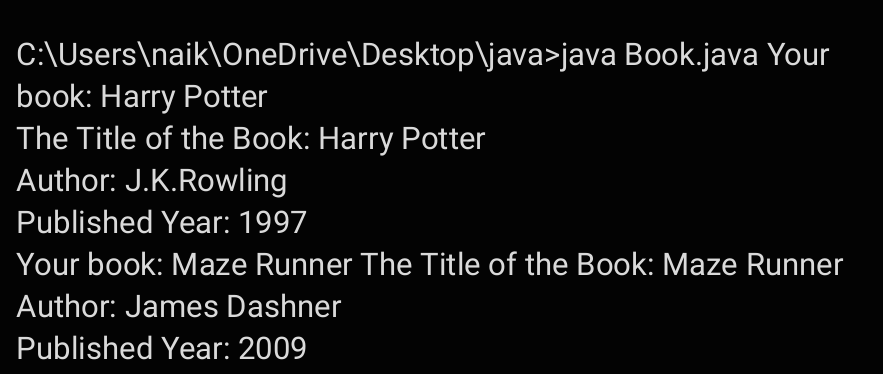
Book book2 = new Book("Maze Runner", "James Dashner", 2009);

book2.getbook();

}

}

**OUTPUT:**



**ERRORS:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Not defining the function in a file. 2. Two public class files should not be saved in the same file. | 1. To call the method we must define a function in a file. 2. Two public class files should be saved in different files. |

**IMPORTANT POINTS:**

1. While defining two classes for a code, we must be sure that we save both the classes in separate files.

2. While defining a method we should also define a function to call that method.

**CLASS DIAGRAM:**

|  |
| --- |
| **Book** |
| Title: String  Author: String  Year of publication: int |
| + Book(title: String,  Author: String;  Year of publication: int  + displayDetails( ): void |

**PROGRAM-2:**

**AIM:** Create a java Program with class named myclass with static variable count of int type, initialized to zero and a constant variable “pi” of type double initialized to 3.14 as attributes of the class, ow define a constructor for “myclass” that increments the count variable each time an object of my class is created (count++), finally print the final values of count and pi variables create three objects.

**CODE:**

class Myclass {

static int count = 0;

final double pi = 3.14;

Myclass() {

count = count+1;

}

public void display() {

System.out.println("Count is: " + count);

System.out.println("Double is: " + pi);

System.out.println();

}

public static void main(String[] args) {

Myclass Asec = new Myclass();

Asec.display();

Myclass Bsec = new Myclass();

Bsec.display();

Myclass Cse = new Myclass();

Cse.display();

System.out.println("The final count is: " + count);

System.out.println("Double is: " + Asec.pi);

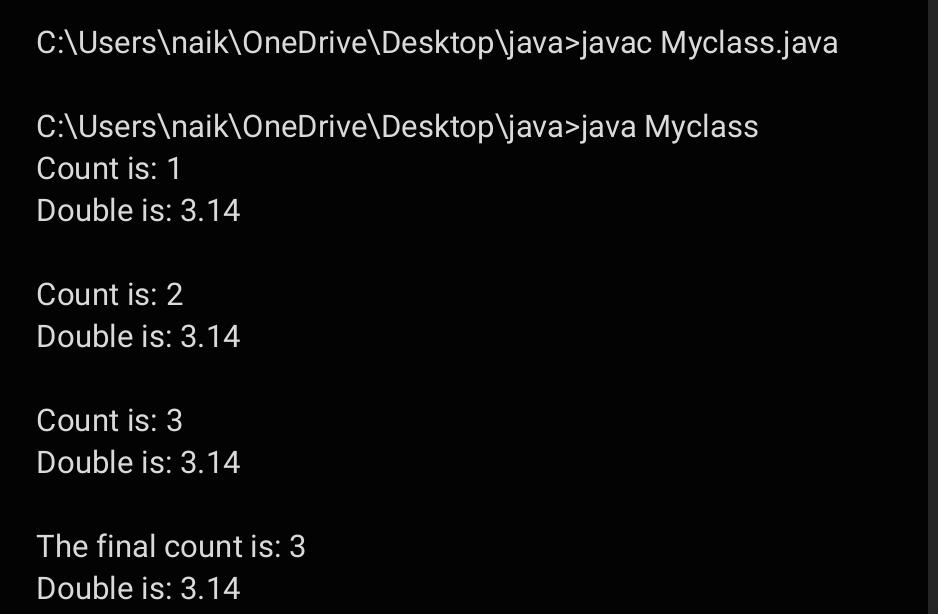
}

}

**ERRORS:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Not Putting the semi-colon after calling a function. 2. Not giving the indentation properly. | 1. Put the semi-colon after calling a function. 2. All the indentation must be correct to run the code correct. |

**OUTPUT:**



**CLASS DIAGRAM:**

|  |
| --- |
| **MyClass** |
| * Count: int * Pi: double |
| + myclass()  +main(args:String[]):void |

**IMPORTANT POINTS:**

1. We must declare the initial value of the variable before declaring the final one.
2. Here the main objective is to increase the count according to the number of objects we make, i.e the count increases when the no.of objects are increasing.

**WEEK 05**

**PROGRAM-1:**

**AIM:** Create a calculator using the operations including addition, subtraction, multiplication, and division using multi-level inheritance and display the desired output.

Hint: collect required variables using super class, create each class for a parameter and each class must contain a method.

**CODE:**

class calculator {

protected double a, b;

public calculator(double a, double b) {

this.a = a;

this.b = b;

}

}

class Addition extends calculator {

public Addition(double a, double b) {

super(a, b);

}

public double add() {

return a + b;

}

}

class Subtraction extends Addition {

public Subtraction(double a, double b) {

super(a, b);

}

public double subtract() {

return a - b;

}

}

class Multiplication extends Subtraction {

public Multiplication(double a, double b) {

super(a, b);

}

public double multiply() {

return a \* b;

}

}

class Division extends Multiplication {

public Division(double a, double b) {

super(a, b);

}

public double divide() {

if (b != 0) {

return a / b;

} else {

System.out.println("Error");

return Double.NaN;

}

}

}

class Final extends Division {

public Final(double a, double b) {

super(a, b);

}

public void displayResults() {

System.out.println("Addition: " + add());

System.out.println("Subtraction: " + subtract());

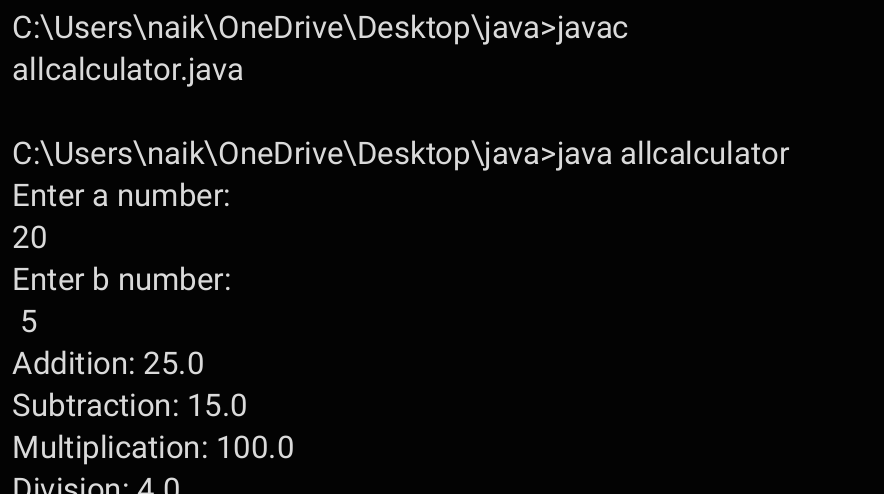
System.out.println("Multiplication: " + multiply());

System.out.println("Division: " + divide());

}

}

**OUTPUT:**

****

**IMPORTANT POINTS:**

1. To get the inputs from the user we use import java.util.Scanner; this is a package.
2. Scanner class is used to get the user input.
3. In java.util.Scanner, the java.util is a package while Scanner is a class of the java.util package.
4. To import a whole package, end the sentence with an asterisk sign(\*).

**ERRORS:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. not providing the return method correctly. 2. Not mentioning super to obtain the super class constructor. | 1. After declaring methods, we must provide the return method correctly. 2. To obtain the super class we need to mention super. |

**CLASS DIAGRAM:**

|  |
| --- |
| **Calculator** |
| * a : double * b : double |
| + Calculator (a,b) |

|  |
| --- |
| **Addition** |
| + add() : double |

|  |
| --- |
| **Subtraction** |
| + subtract() : double |

|  |
| --- |
| **Multiplication** |
| + multiply() : double |

|  |
| --- |
| **Divison** |
| +divide() : double |

**PROGRAM-2:**

**AIM:** A vehicle rental company wants to develop a system that maintains information about different types of vechicles available for rent the company rents out cars and bikes, and they need a program to store details about each vehicle, such as brand and speed( should be in super class)

1. cars should have an additional property: no.of doors
2. Bikes should have a property indicating whether they have gears or not.
3. The system should also include a function to display details about each vehicle and indicate when a vehicle is starting.
4. Every class should have a constructor

**Question:**

1. Which oops concept is used in the above program
2. If the company decides to add a new type of vehicle, Truck, how would you modify the program?
3. Truck should include an additional property capacity (in tons)
4. Create a showTruckdetails() method to display the truck’s capacity.
5. Write a constructor for Truck that initializes all properties
6. Implement the truck class and update the main method to create a Truck object and also create an object for car and bike sub classes Finally, display the details.

**CODE:**

class Vehicle {

private String brand;

private int speed;

Vehicle(String brand, int speed) {

this.brand = brand;

this.speed = speed;

}

void details() {

System.out.println("Brand: " + brand);

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

} }

class Car extends Vehicle {

private int doors;

private int capacity;

public Car(String brand, int speed, int doors, int capacity) {

super(brand, speed);

this.doors = doors;

this.capacity = capacity;

}

void carDetails() {

System.out.println("Number of doors: " + doors);

System.out.println("Capacity: " + capacity);

}

@Override

void details() {

super.details();

carDetails();

} }

class Bike extends Vehicle {

private boolean gears;

Bike(String brand, int speed, boolean gears) {

super(brand, speed);

this.gears = gears;

}

void bikeDetails() {

System.out.println(gears ? "This bike has gears." : "This bike does not have gear system.");

}

@Override

void details() {

super.details();

bikeDetails();

} }

class Truck extends Vehicle {

private int tons;

Truck(String brand, int speed, int tons) {

super(brand, speed);

this.tons = tons;

}

void truckDetails() {

System.out.println("The capacity of truck is: " + tons + " tons.");

}

@Override

void details() {

super.details();

truckDetails();

} }

class Rent {

public static void main(String[] args) {

Car c = new Car("Toyota", 120, 5, 5);

c.details();

Bike b = new Bike("KTM", 80, true);

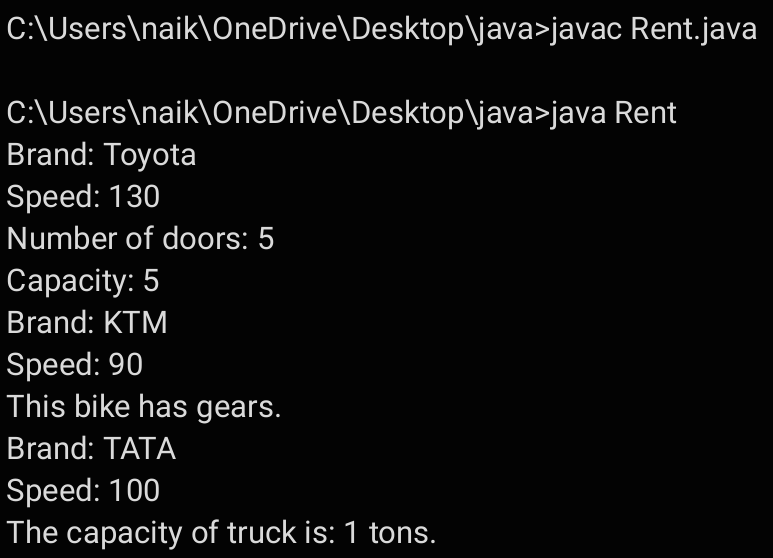
b.details();

Truck t = new Truck("TATA", 100, 1);

t.details();

} }

**OUTPUT:**

****

**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Declaring two superclasses inside the same file. 2. Not declaring the variable using ‘this’ keyword inside the constructor. | 1. Make two separate files to save the two super classes. 2. Declare the variable using this keyword to run the program. |

**CLASS DIAGRAM:**

|  |
| --- |
| **Vehicle** |
| -Brand : string  -Speed: int |
| **+** init (brand, speed)  **+** start\_vehicle()  **+** display\_details() |

|  |
| --- |
| **Car** |
| -no.of.doors:int |
| +int (brand, speed,  No.of doors);  +display deatails(); |

|  |
| --- |
| **Bikes** |
| -has gears:bool |
| +int (brand, speed,  has gears);  +display deatails(); |

|  |
| --- |
| **Truck** |
| -Capacity:float |
| -Show truck detais();  +display deatails(); |

**IMPORTANT POINTS:**

1. a constructor helps in initializing an object that doesn't exist.
2. a method performs functions on pre-constructed or already developed objects.
3. a double method can represent more decimal point numbers than float method.
4. the void keyword in java is used to specify that a method does not return any value. it is a return type that indicates the method performs a function and doesn't produce a result.

Answer:

The oops concepts used in the above program are:

Inheritance, encapsulation, polymorphism, abstraction.

To add a new vehicle type truck we need to create a truck class that will:

* Include an additional property capacity (in tons).
* Implement a showtruckdetials() method to display the truck's capacity.
* Implement a constructor for the truck class to initialize all its properties.

**WEEK-06**

**PROGRAM-1:**

**AIM:** Write a java program to create a vehicle class with a method displayinfo(). Override this method in the car subclass to provide specific information about car (car company, seating capacity, petrol or not).

**CODE:**

class Vehicle {

String car\_company;

String car\_model;

long car\_price;

int seating\_capacity;

boolean petrol;

Vehicle(String car\_company, String car\_model, long car\_price, int seating\_capacity, boolean petrol) {

this.car\_company = car\_company;

this.car\_model = car\_model;

this.car\_price = car\_price;

this.seating\_capacity = seating\_capacity;

this.petrol = petrol;

}

void displayInfo() {

System.out.println("Car company: " + car\_company);

System.out.println("Car model: " + car\_model);

System.out.println("Car price: " + car\_price);

System.out.println("Car seating capacity: " + seating\_capacity);

System.out.println("Car uses petrol: " + petrol);

}

}

class Car extends Vehicle {

Car(String car\_company, String car\_model, long car\_price, int seating\_capacity, boolean petrol) {

super(car\_company, car\_model, car\_price, seating\_capacity, petrol)

}

}

public class Main {

public static void main(String[] args) {

// Creating a Car object with correct arguments

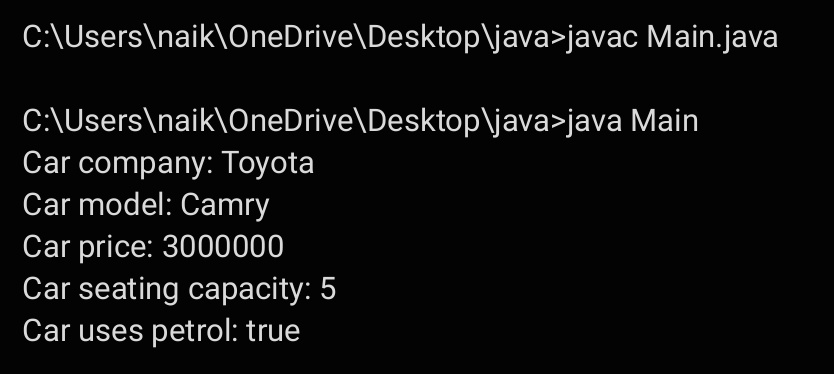
Car c1 = new Car("Toyota", "Camry", 3000000, 5, true);

c1.displayInfo(); // Display car details

}

}

**OUTPUT:**

****

**ERRORS:**

|  |  |
| --- | --- |
| **Code error** | **Code rectification** |
| 1. Incorrect class name for main method(Truck).  2. Inconsistent car model output in displayinfo(). | 1.Rename Truck to Main or place main inside car or vehicle.  2. Ensure Car correctly passes Toyota” to super(car\_model,color,fueltype) |

**IMPORTANT POINTS:**

**1.Inheritance:** The Car class extends the Vehicle class, demonstrating inheritance in Java.

**2.Constructor Chaining:**The Car class calls the parent constructor using super(car\_model, color, fuel\_type); to initialize inherited attributes.

**3.Method Overriding:**The Car class overrides the displayInfo() method from Vehicle and calls super.displayInfo() to reuse the parent method before adding its own output.

**4.Incorrect** main **Class Name:**The main method is inside Truck, which is unrelated to Vehicle and Car. The class should be renamed for clarity.

**CLASS DIAGRAM:**

|  |
| --- |
| **Vehicle** |
| * Brand: String * Speed: int |
| + vehicle(brand: string  Speed: int)  +start vehicle(): void  +displaydetails():void |

**PROGRAM-2:**

**AIM:** A college is developing an automated admission system that verifies students eligibility(UG) and postgraduation(PG) programs. Each program has different eligibility criteria based on the students percentage in their previous qualification.

1. UG admission recquire a minimum of 60%.

2. PG admission recquire a minimum of 70%.

**CODE:**

class Student {

String name;

double percentage

Student(String name, double percentage) {

this.name = name;

this.percentage = percentage;

}

void studentsinfo() {

System.out.println("Student Name: " + name);

System.out.println("Percentage: " + percentage);

}

}

class UG extends Student {

UG(String name, double percentage) {

super(name, percentage);

}

void checkEligibility() {

if (percentage >= 60) {

System.out.println(name + " is eligible for admission in UG.");

}

else {

System.out.println(name + " is not eligible for admission in UG.");

}

}

}

class PG extends Student {

PG(String name, double percentage) {

super(name, percentage);

}

void checkEligibility() {

if (percentage >= 70) {

System.out.println(name + " is eligible for admission in PG.");

}

else {

System.out.println(name + " is not eligible for admission in PG.");

}

}

}

public class AutomatedAdmission {

public static void main(String[] args) {

UG ug = new UG("BHARATH", 80);

ug.studentsinfo();

ug.checkEligibility();

PG pg = new PG("LUCKY", 85);

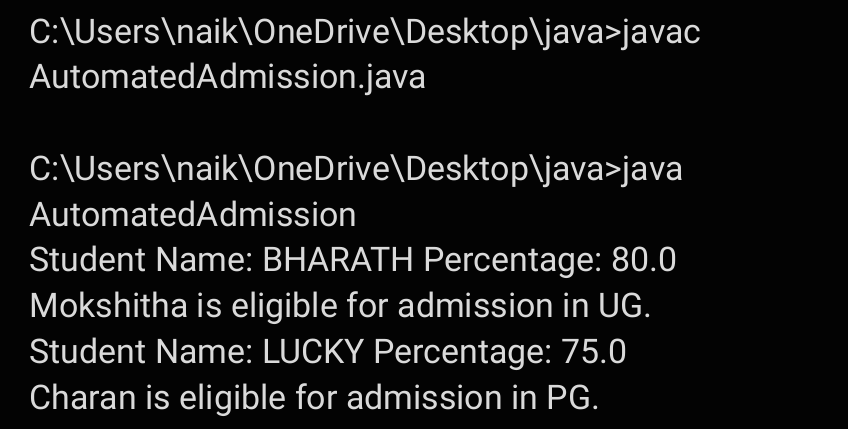
pg.studentsinfo();

pg.checkEligibility();

}

}

**OUTPUT:**

****

**ERRORS:**

|  |  |
| --- | --- |
| **Code error** | **Code rectification** |
| **1.Scanner nextLine() issue after nextDouble():** After scanner.nextDouble(), the newline character remains in the buffer, causing nextLine() to be skipped.  **2.Program type input case sensitivity issue**: If the user enters ug or pg in lowercase, it may cause incorrect comparisons. | **1**.Add scanner.nextLine(); after nextDouble(); to consume the leftover newline.  **2.**Use program.toUpperCase() to ensure case-insensitive comparison. |

**IMPORTANT POINTS:**

**1.User Input Handling:** Uses Scanner to take user input for name, percentage, and program type.

**2.Decision Making with Conditions:** Uses if-else statements to check eligibility criteria.

**3.String Handling:** Converts program input to uppercase (toUpperCase()) to handle case variations.

**4.Closing Scanner:** Properly closes scanner using scanner.close(); to prevent resource leaks.

|  |
| --- |
| **AutomatedAdmission** |
| * Scanner: scanner * Name: String * Percentage : double * Program: stirng |
| + main(args:String[]): void  +takeInput(): void  +checkEligibility(): void  +closeScanner(); void |

**PROGRAM-3:**

**AIM:** Create a calculator class with overloaded methods to perform addition of:

1. Add two integers

2. Add two doubles

3. Add three integers

**CODE:**

class Calculator\_6 {

public int add(int a, int b) {

return a + b;

}

public double add(double a, double b) {

return a + b;

}

public int add(int a, int b, int c) {

return a + b + c;

}

}

class Main\_6 {

public static void main(String[] args) {

Calculator\_6 calculator = new Calculator\_6();

System.out.println("Addition of two integers: " + calculator.add(5, 10));

System.out.println("Addition of two doubles: " + calculator.add(5.5, 2.2));

System.out.println("Addition of three integers: " + calculator.add(1, 2, 3));

}

}

**OUTPUT:**

****

**ERRORS:**

|  |  |
| --- | --- |
| **Code error** | **Code rectification** |
| 1.Method parameters missing spaces. E.g.,”inta, intb”should be “int a, int b”  2.Inconsistent indentation in method bodies | 1**.** Add proper spacing between parameters: (int a, int b)  2.Fix indentation:  Consistent 4 space o indentation. |

**IMPORTANT POINTS:**

**1.Method Overloading:** The add method is overloaded with different parameter types and counts, demonstrating compile-time polymorphism.

**2.Automatic Method Selection:** Java selects the appropriate add method based on the argument types during compilation.

**CLASS DIAGRAM:**

|  |
| --- |
| **Calculator** |
| + add(int, int): int  +add(double, double): double  +add(int,int,int): int |

**PROGRAM-4:  
AIM:** Create a shape class with a method to calculate area i.e., overloaded for different shapes eg: Squares, Recatangle. Then create a subclass circle that overrides the calculateArea() method for a circle.

**CODE:**

class Shape {

public double calculateArea(double side) {

return side \* side;

}

public double calculateArea(double length, double width) {

return length \* width;

}

}

class Circle extends Shape {

private double radius

public Circle(double radius) {

this.radius = radius;

}

public double calculateArea() {

return Math.PI \* radius \* radius;

}

}

class Tools {

public static void main(String[] args) {

Shape shape = new Shape();

Circle circle = new Circle(5);

System.out.println("Area of square: " + shape.calculateArea(4));

System.out.println("Area of rectangle: " + shape.calculateArea(4,

6));

System.out.println("Area of circle: " + circle.calculateArea());

}

}

**OUTPUT:**



**ERRORS:**

|  |  |
| --- | --- |
| **Code error** | **Code rectification** |
| 1. Method calls in main are missing an object reference (e.g., calculateArea(4) instead of s.calculateArea(4)).  2. Circle class method does not override theparent class method properly. | 1.Use s.calculateArea(4) and c.calculateArea(2) to call the method correctly.  2. Ensure @Override is used, and the method signature should match correctly. |

**CLASS DIAGRAM:**

|  |
| --- |
| **SHAPE** |
| + CalculateArea(side:double): double  +CalculateArea(width: double, length: double): double |

|  |
| --- |
| **CIRCLE** |
| + CalculateArea(radius: double): double |

|  |
| --- |
| **Tools** |
| +main(args:String[]): Void |

**IMPORTANT POINTS:**

**1.Inheritance**: Circle class extends Shape, inheriting its methods.

**2.Method Overloading**: Shape has multiple calculateArea methods with different parameters.

**3.Method Overriding**: Circle overrides calculateArea from Shape to implement its own formula.

**4.Polymorphism**: The overridden method in Circle demonstrates runtime polymorphism.

**5.Proper Object Reference**: Methods should be called using an object (s.calculateArea(4), c.calculateArea(2)).

**WEEK-7**

**1Q)Write a java program to create an abstract class Animal with an abstract method sound().Create Subclass Tiger and Lion extends the Animal class and implement the sound() method to make a specific sound for each animal**

**Program:**

**abstract class Animal {**

**public abstract void sound();**

**}**

**class Lion extends Animal {**

**@Override**

**public void sound() {**

**System.out.println("Lion: Roar!");**

**}**

**}**

**class Tiger extends Animal {**

**@Override**

**public void sound() {**

**System.out.println("Tiger: Growl!");**

**}**

**}**

**public class Q1labw7 {**

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

**Animal lion = new Lion();**

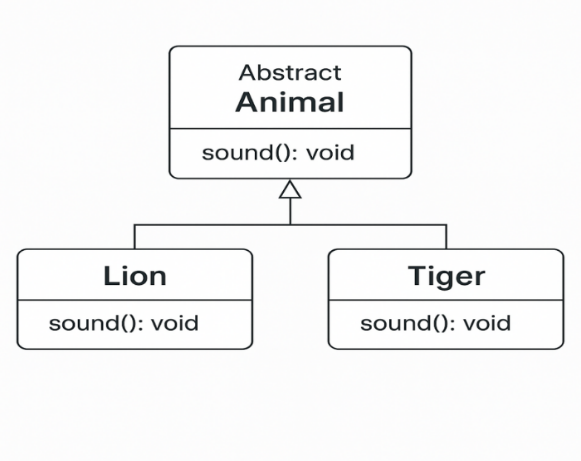
**Animal tiger = new Tiger();**

**lion.sound();**

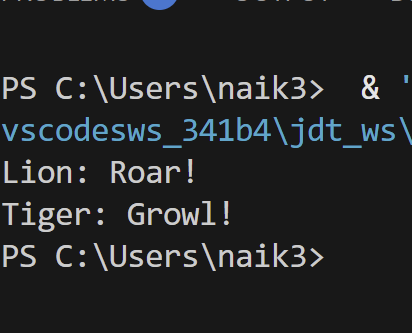
**tiger.sound();**

**}**

**}**

****

OutPut:



**Error Table:**

|  |  |  |
| --- | --- | --- |
| S.no | Expected Error | Reason |
| **1** | Setting the parameters inside the constructor | We cannot pass the values inside constructor without setting them first |
| **2** | } | Ending the class and main method is required |

**2Q)Write a java program to create an abstract class Shape3D with an abstract methods Calculate\_volume() and Calculate\_Surface\_area.Create Subclass Sphere and Cube extends the Shape3D class and implement the respective methods to calculate the volume and surface\_area of each shape.**

**Program:**

abstract class Shape3D {

public abstract void calculate\_volume();

public abstract void calculate\_surf\_a();

}

class Sphere extends Shape3D {

private double radius;

public Sphere(double radius) {

this.radius = radius;

}

@Override

public void calculate\_surf\_a() {

double surfaceArea = 4 \* Math.PI \* Math.pow(radius, 2);

System.out.printf("Surface Area of Sphere: %.2f%n", surfaceArea);

}

@Override

public void calculate\_volume() {

double volume = (4.0 / 3) \* Math.PI \* Math.pow(radius, 3);

System.out.printf("Volume of Sphere: %.2f%n", volume);

}

}

class Cube extends Shape3D {

private double side;

public Cube(double side) {

this.side = side;

}

@Override

public void calculate\_surf\_a() {

double surfaceArea = 6 \* Math.pow(side, 2);

System.out.printf("Surface Area of Cube: %.2f%n", surfaceArea);

}

@Override

public void calculate\_volume() {

double volume = Math.pow(side, 3);

System.out.printf("Volume of Cube: %.2f%n", volume);

}

}

public class Question2 {

public static void main(String[] args) {

Shape3D sphere = new Sphere(5);

Shape3D cube = new Cube(3);

sphere.calculate\_surf\_a();

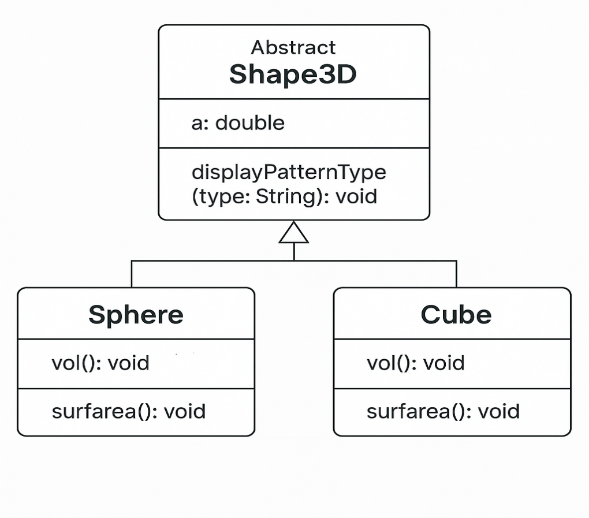
sphere.calculate\_volume();

cube.calculate\_surf\_a();

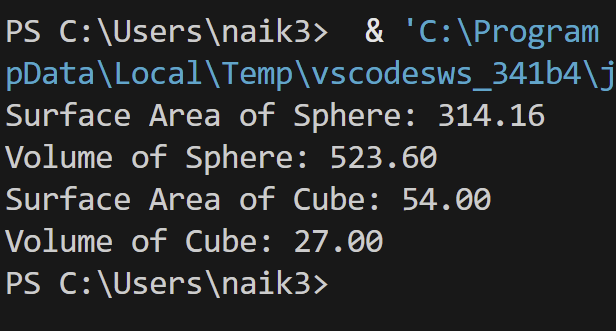
cube.calculate\_volume();

}

}



**Output:**

****

**Error Table:**

|  |  |  |
| --- | --- | --- |
| S.no | Expected Error | Reason |
| **1** | Setting the parameters inside the constructor | We cannot pass the values inside constructor without setting them first |
| **2** | } | Ending the class and main method is required |

**3Q)Write a java program using an abstract class to define a method for pattern printing**

**-->create an abstract class named patternprinting with an abstract method print pattern (int n) and a concrete method to display the pattern title**

**-->impletment two sub classes**

1. **star pattern**

**Prints a right angled triangle of stars**

1. **Number pattern**

**Prints a right angled triangle of increasing numbers**

**-->in the main()method create objects of both sub classes and print the patterns for a given number of rows**

**Program:**

import java.util.Scanner;

abstract class Pattern {

public abstract void printPattern(int n);

}

class RightTrianglePattern extends Pattern {

@Override

public void printPattern(int n) {

System.out.println("Right Triangle Pattern:");

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= i; j++) {

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

}

System.out.println();

}

}

}

class NumberPattern extends Pattern {

@Override

public void printPattern(int n) {

System.out.println("Number Pattern:");

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= i; j++) {

System.out.print(j);

}

System.out.println();

}

}

}

public class Question3 {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.println("Enter the n value to select number of rows:");

int n = input.nextInt();

Pattern rightTriangle = new RightTrianglePattern();

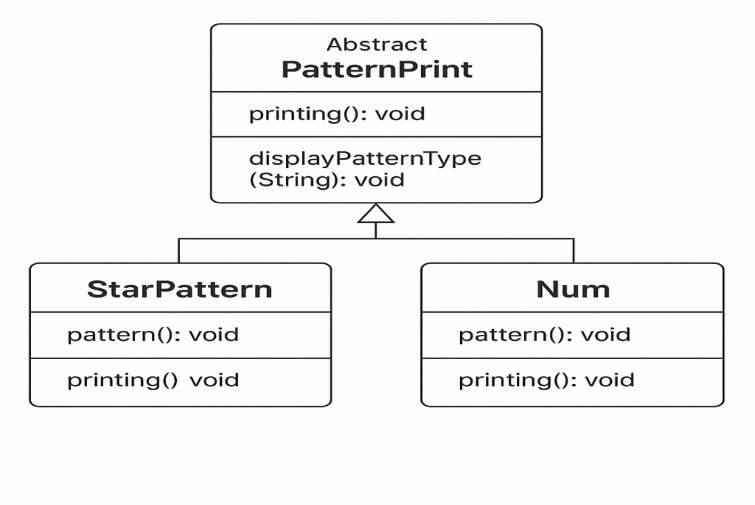
Pattern numberPattern = new NumberPattern();

rightTriangle.printPattern(n);

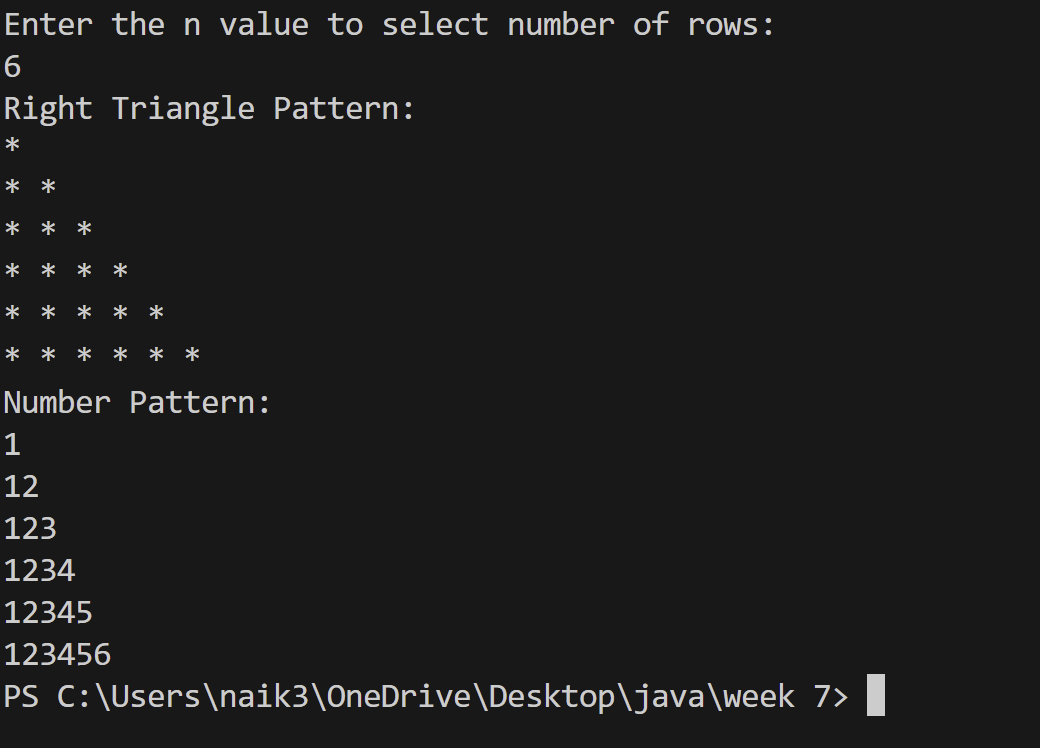
numberPattern.printPattern(n);

}

}

****

**Output:**

****

**Error Table:**

|  |  |  |
| --- | --- | --- |
| S.no | Expected Error | Reason |
| **1** | Setting the parameters inside the constructor | We cannot pass the values inside constructor without setting them first |
| **2** | } | Ending the class and main method is required |

Week-8

**1Q)write a java program creating an interface Shape with the get perimeter method create 3 classes rectangle,triangleand circle that implements the shape interface ,implement the getperimeter method for each of the three classes**

**Program:**

interface Shape {

double getPerimeter();

}

class Rectangle implements Shape {

int width, height;

Rectangle(int width, int height) {

this.width = width;

this.height = height;

}

public double getPerimeter() {

return 2 \* (width + height);

}

}

class Circle implements Shape {

int radius;

Circle(int radius) {

this.radius = radius;

}

public double getPerimeter() {

return 2 \* Math.PI \* radius;

}

}

class Triangle implements Shape {

int side1;

int side2;

int side3;

Triangle(int side1, int side2, int side3) {

this.side1 = side1;

this.side2 = side2;

this.side3 = side3;

}

public double getPerimeter() {

return side1 + side2 + side3;

}

}

public class Q1labweek8{

public static void main(String[] args) { // Corrected brackets

Shape rectangle = new Rectangle(5, 10);

Shape circle = new Circle(7);

Shape triangle = new Triangle(1, 2, 3);

System.out.println("Rectangle perimeter: " + rectangle.getPerimeter());

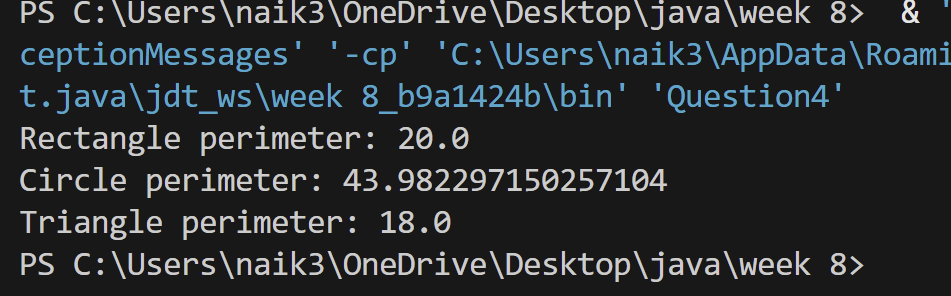
System.out.println("Circle perimeter: " + circle.getPerimeter());

System.out.println("Triangle perimeter: " + triangle.getPerimeter());

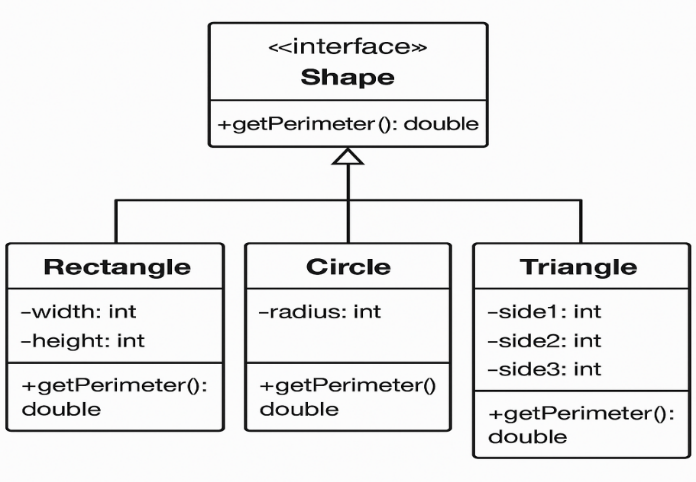
}

}

**Output:**

****

**Class Diagram:**

****

**Error Table:**

|  |  |  |
| --- | --- | --- |
| S.no | Expected Error | Reason |
| **1** | Setting the parameters inside the constructor | We cannot pass the values inside constructor without setting them first |
| **2** | } | Ending the class and main method is required |

**2Q)write a java program to create an interface playable with a method play()**

**That takes no arguments and returns void create three classes football,volleyball and basketball that implements the playable and override the play method to play the respective sports**

**Program:**

interface playable {

void play();

}

class volleyball implements playable {

public void play() {

System.out.println("volleyball is a sport");

}

}

class basketball implements playable {

public void play() {

System.out.println("basketball is also a sport");

}

}

class football implements playable {

public void play() {

System.out.println("football is a sport");

}

}

public class Question5 {

public static void main(String[] args) {

playable vb = new volleyball();

playable bb = new basketball();

playable fb = new football();

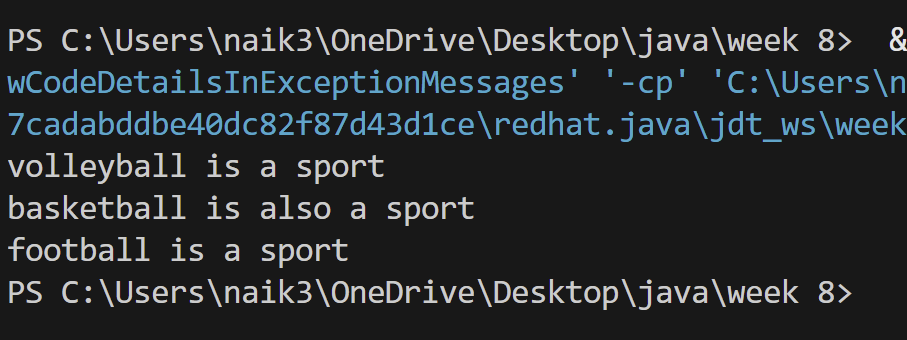
vb.play();

bb.play();

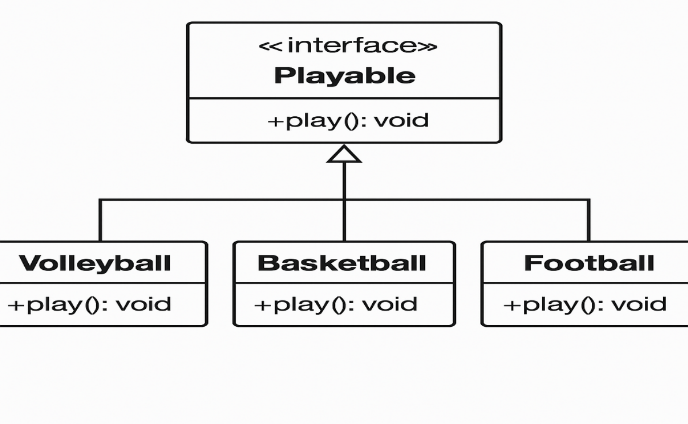
fb.play();

}

}**Output:**

****

**Class Diagram:**

****

**Error Table:**

|  |  |  |
| --- | --- | --- |
| S.no | Expected Error | Reason |
| **1** | Setting the parameters inside the constructor | We cannot pass the values inside constructor without setting them first |
| **2** | } | Ending the class and main method is required |

**3Q)write a java program to implement a login system using interfaces**

**Program:**

interface LoginSystem {

boolean login(String id, String password);

}

class UniversityPortal implements LoginSystem {

@Override

public boolean login(String id, String password) {

if (id.equals("student123") && password.equals("pass123")) {

return true;

} else {

System.out.println("Invalid credentials");

return false;

}

}

}

public class Question6 {

public static void main(String[] args) {

UniversityPortal portal = new UniversityPortal();

boolean loginSuccess1 = portal.login("student123", "pass123");

System.out.println("Login successful: " + loginSuccess1);

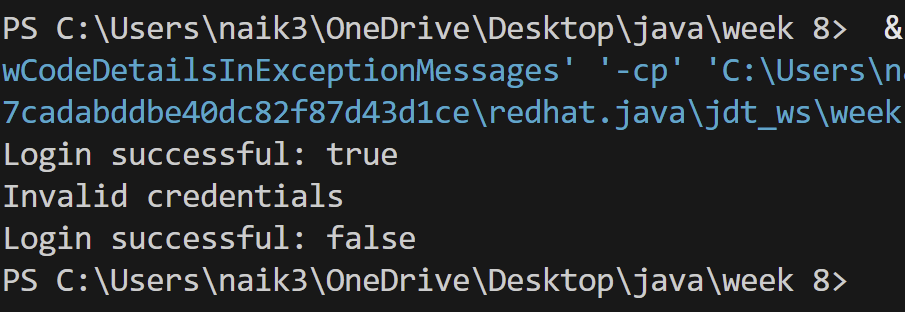
boolean loginSuccess2 = portal.login("student123", "wrong pass");

System.out.println("Login successful: " + loginSuccess2);

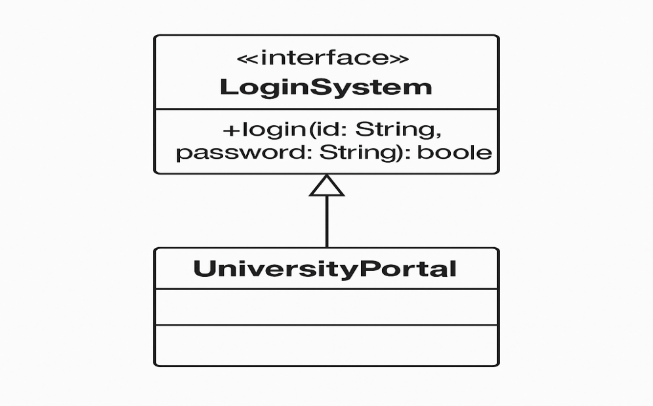
}

}

**Output:**

****

**Class Diagram:**

****

**Error Table:**

|  |  |  |
| --- | --- | --- |
| S.no | Expected Error | Reason |
| **1** | Setting the parameters inside the constructor | We cannot pass the values inside constructor without setting them first |
| **2** | } | Ending the class and main method is required |

Week-9

**1Q)write a java program to create a method that take integer as parameter and throws an example if the number is even**

**Program:**

public class Question7 {

public static void checkNumber(int number) throws Exception {

if (number % 2 == 0) {

throw new Exception("Even number is not allowed: " + number);

} else {

System.out.println("Valid output number: " + number);

}

}

public static void main(String[] args) {

try {

checkNumber(9);

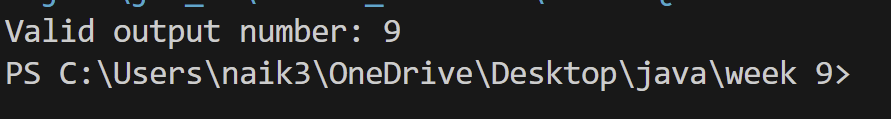
} catch (Exception e) {

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

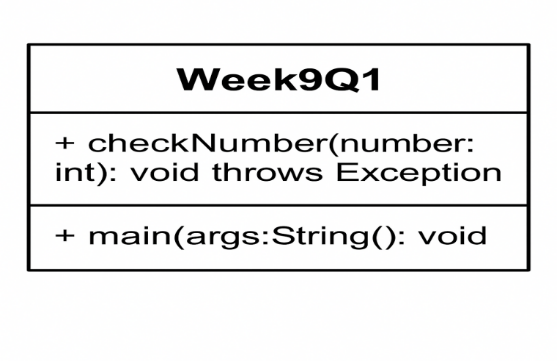
}

}

}

**Output:** ****

**Class Diagram:**

****

**Error Table:**

|  |  |  |
| --- | --- | --- |
| S.no | Expected Error | Reason |
| **1** | Setting the parameters inside the constructor | We cannot pass the values inside constructor without setting them first |
| **2** | } | Ending the class and main method is required |

**2Q)write ajava program to create a method that reads a file and throws an exception if the file is not found**

**Program:**

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

public class Question8 {

public static void main(String[] args) {

BufferedReader br = null;

try {

br = new BufferedReader(new FileReader("E:/Amrita/example.txt"));

String line;

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

System.out.println(line);

}

} catch (IOException e) {

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

} finally {

if (br != null) {

try {

br.close();

} catch (IOException e) {

System.out.println("Error closing the BufferedReader: " + e.getMessage());

}

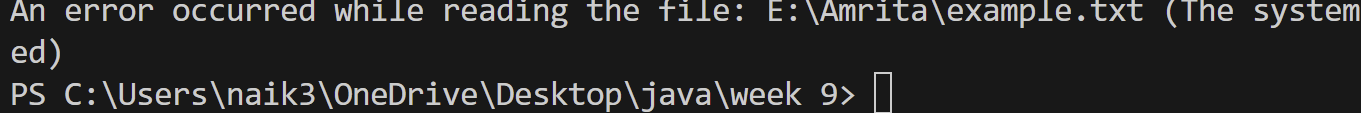
}

}

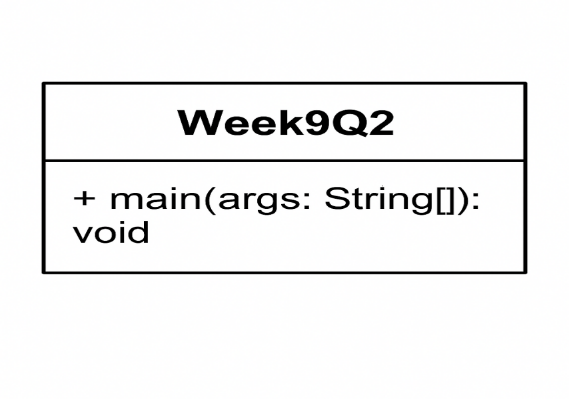
}

}

**Output:**

****

**Class Diagram:**

****

**Error Table:**

|  |  |  |
| --- | --- | --- |
| S.no | Expected Error | Reason |
| **1** | Setting the parameters inside the constructor | We cannot pass the values inside constructor without setting them first |
| **2** | } | Ending the class and main method is required |

**3Q)write ajava program to handle arithematic exception using try catch and finally**

**Program:**

import java.util.Scanner;

public class Week9Q3{

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

try {

System.out.println("Enter first number (numerator):");

int a = input.nextInt();

System.out.println("Enter second number (denominator):");

int b = input.nextInt();

int result = a / b;

System.out.println("Result of division: " + result);

} catch (ArithmeticException e) {

System.out.println("Error: Cannot divide by zero.");

} catch (Exception e) {

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

} finally {

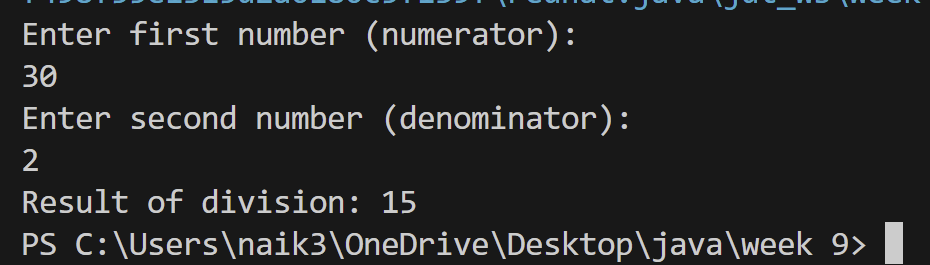
input.close();

}

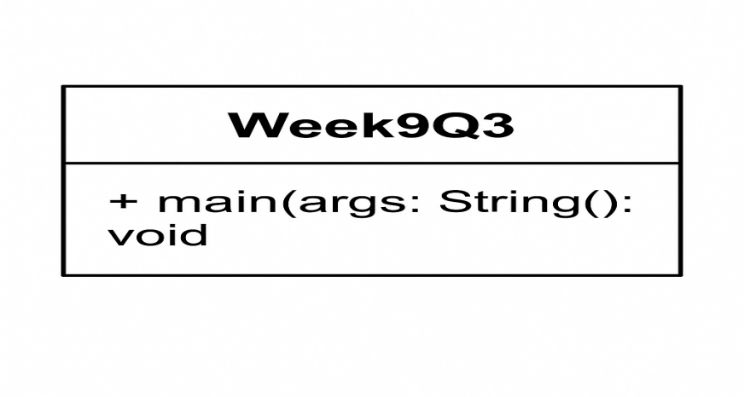
}

}

**Output:**

****

**Class Diagram:**

****

**Error Table:**

|  |  |  |
| --- | --- | --- |
| S.no | Expected Error | Reason |
| **1** | Setting the parameters inside the constructor | We cannot pass the values inside constructor without setting them first |
| **2** | } | Ending the class and main method is required |

**4Q)write a java program to simulate a universityy system using inner classes**

* **Create an outer class namedd University with a variable UniversityName**
* **Inside it defgine two non-static in classes**

1. **Department-With variable like deptName and deptCode and a method to display department details.**
2. **Student-Variable like stdName and stdCode and a method to display Student details.**
3. **Create an object for each class and call their methods to display their details and with the university name.**

**Program:**

public class Week9Q4 {

String universityName = "Amrita University";

class Department {

String deptName = "computer science";

int deptcode = 101;

void displayDepartmentInfo() {

System.out.println("department" + deptName);

System.out.println("department" + deptcode);

}

}

class student {

String stdname = "Ganesh Reddy";

int stdcode = 1877;

void displayStudentInfo() {

System.out.println("department" + stdname);

System.out.println("department" + stdcode);

}

}

public static void main(String[] args) {

Week9Q4 uni = new Week9Q4();

System.err.println("University" + uni.universityName);

System.err.println("Department\_\_Info");

Week9Q4.Department dept = uni.new Department();

dept.displayDepartmentInfo();

System.out.println("\*\*\*\*\* Student Info \*\*\*\*\*");

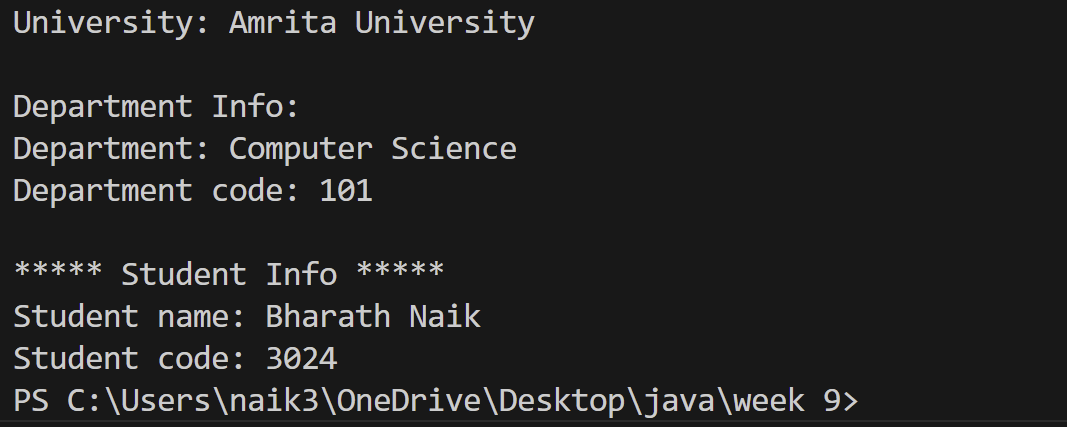
Week9Q4.student stdent = uni.new student();

stdent.displayStudentInfo();

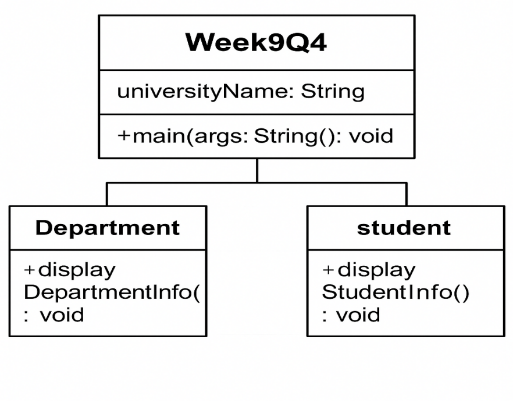
}

}

**Output:**

****

**Class Diagram:**

****

**Error Table:**

|  |  |  |
| --- | --- | --- |
| S.no | Expected Error | Reason |
| **1** | Setting the parameters inside the constructor | We cannot pass the values inside constructor without setting them first |
| **2** | } | Ending the class and main method is required |

Week-10

**1Q)Write a java program to generate a password for a student using his/her intials and age. The password displayed should be the**

**string consists of first character of first name, middle name last name with age.**

**Program:**

import java.lang.String;

import java.util.Scanner;

public class Question {

    public static void main(String[] args) {

      Scanner input=new Scanner(System.in);

      System.out.println("enter the first name");

      String FN=input.next();

      System.out.println("enter the last name");

      String LN=input.next();

      System.out.println("enter the age");

      int AGE=input.nextInt();

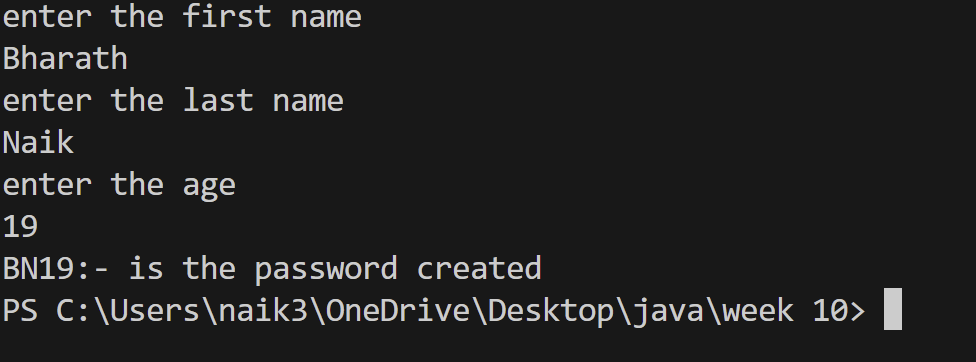
      String initials=FN.substring(0,1)+LN.substring(0,1)+AGE;

      String PIN=initials.toLowerCase();

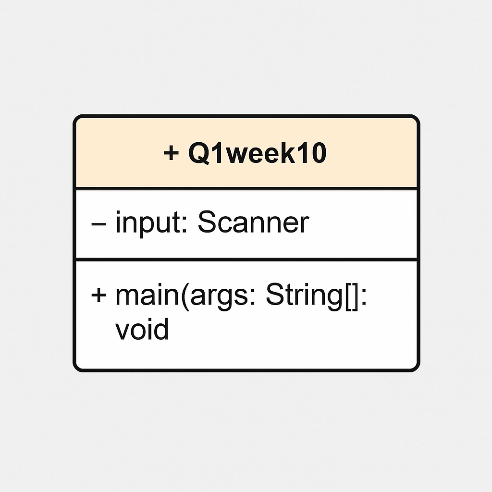
     System.out.println(initials+":- is the password created");

}

}

**Output:** ****

**Class Diagram:**

**:**

**Class Diagram:**

**2Q)Design and implement a java program that will do the following questions to this string "Welcome! You are practicing Strings concept”.**

**i) Convert all the alphabets to capital letters and print out the result**

**ii) Convert all alphabets to lower-case letters and print out the result**

**iii) print out the length of the string**

**iv) Print out the index of the concept.**

**Program:**

public class Question1 {

public static void main(String[] args) {

String Given = "welcome! You are practicing strings concept";

System.out.println("Original String: " + Given);

System.out.println("Converting into uppercase letters: " + Given.toUpperCase());

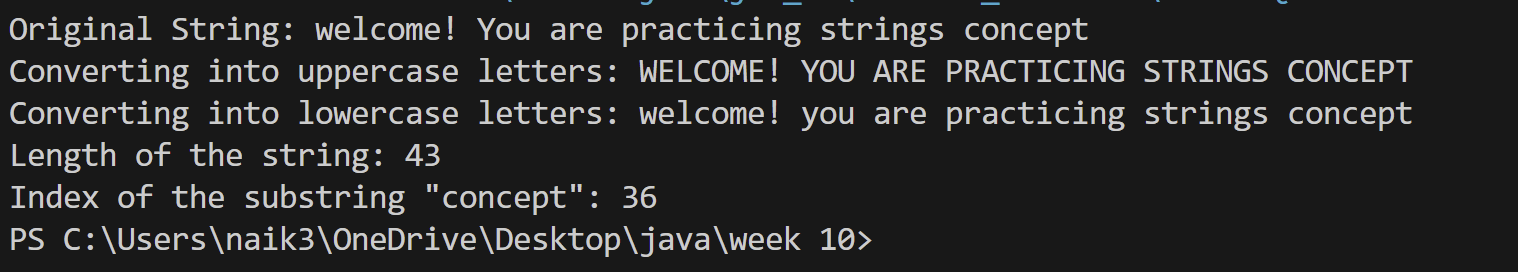
System.out.println("Converting into lowercase letters: " + Given.toLowerCase());

System.out.println("Length of the string: " + Given.length());

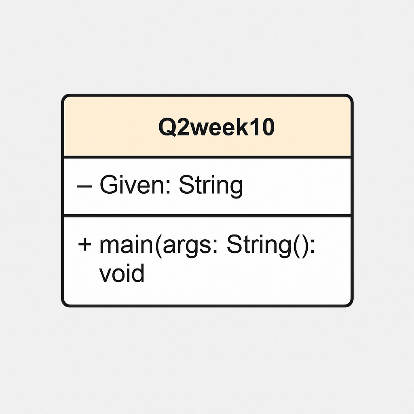
System.out.println("Index of the substring \"concept\": " + Given.indexOf("concept"));

}

}**Output:**

****

**Class Diagram:**



**Class Diagram:**

**3Q) Implement a java program using the below array methods**

**i) Sorting the elements (numbers and strings ) of an array**

**ii) Convert the array elements into string**

**iii) Fill the part of an array**

**iv) Copy the elements of one array into another**

**Program:**

import java.util.Arrays;

public class Question2 {

public static void main(String[] args) {

// 1. Sorting the elements (numbers)

int[] numbers = {5, 3, 8, 1, 2};

System.out.println("Original numbers array: " + Arrays.toString(numbers));

Arrays.sort(numbers);

System.out.println("Sorted numbers array: " + Arrays.toString(numbers));

// 1. Sorting the elements (strings)

String[] strings = {"Banana", "Apple", "Orange", "Mango"};

System.out.println("Original strings array: " + Arrays.toString(strings));

Arrays.sort(strings);

System.out.println("Sorted strings array: " + Arrays.toString(strings));

// 2. Convert the array elements (numbers) into strings

String[] stringArray = Arrays.stream(numbers)

.mapToObj(String::valueOf)

.toArray(String[]::new);

System.out.println("Converted numbers array to strings: " + Arrays.toString(stringArray));

// 3. Fill part of an array

int[] filledArray = new int[10];

Arrays.fill(filledArray, 0, 5, 7); // Fill first 5 elements with 7

System.out.println("Array after filling part of it: " + Arrays.toString(filledArray));

// 4. Copy the elements of one array into another

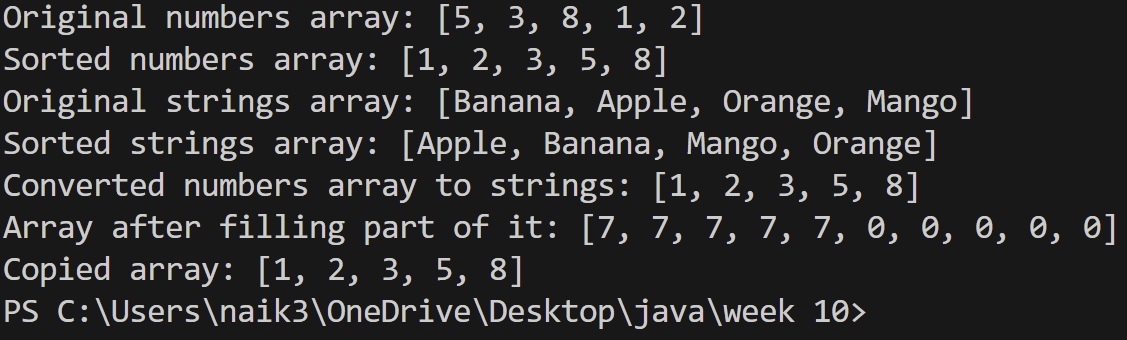
int[] copiedArray = new int[numbers.length];

System.arraycopy(numbers, 0, copiedArray, 0, numbers.length);

System.out.println("Copied array: " + Arrays.toString(copiedArray));

}

}**Output:**

****

**Class Diagram:**



**4) implement a java program using the below Array list**

**i) Insert an element at particular index in the array list**

**ii) Modify an element in the array list**

**iii)access an element from the array list**

**iv) remove an element from the array list**

**Program:**

import java.util.ArrayList;

public class Q4week10 {

public static void main(String[] args) {

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

fruits.add("Apple");

fruits.add("Banana");

fruits.add("Orange");

System.out.println("Original ArrayList: " + fruits);

fruits.add(1, "Mango");

System.out.println("After inserting 'Mango' at index 1: " + fruits);

fruits.set(2, "Grapes");

System.out.println("After modifying element at index 2: " + fruits);

String fruitAtIndex3 = fruits.get(3);

System.out.println("Element at index 3: " + fruitAtIndex3);

fruits.remove("Banana");

System.out.println("After removing 'Banana': " + fruits);

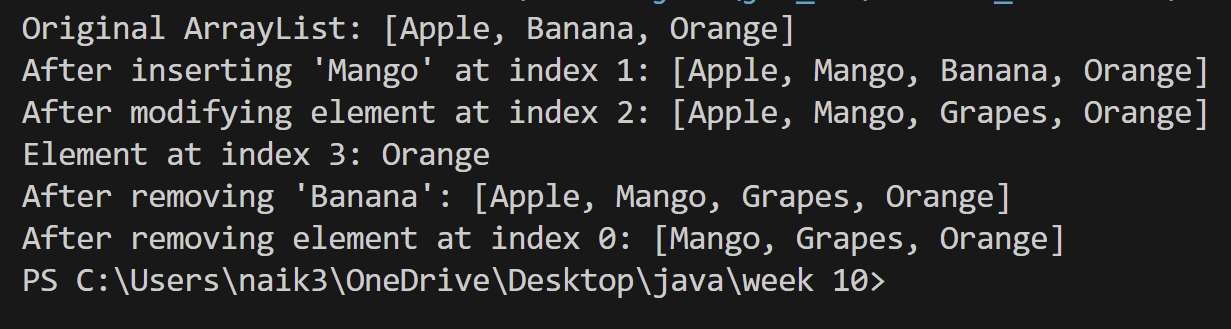
fruits.remove(0);

System.out.println("After removing element at index 0: " + fruits);

}

}

**Output:**

****

**Class Diagram:**

