**23CSE111**

**OBJECT ORIENTED PROGRAMMING**

**LAB REPORT**



**Department of Computer Science Engineering**

**Amrita School of Computing**

**Amrita Vishwa Vidyapeetham,**

**Amaravathi**

**Name: M.N.H.O.K .MANOJ**

**Roll No: 24214**

**Verified By :**

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **s.no** | **Week** | **Experiment** | **Page.no** |
| **1.** | **Week-1** | **1.Installation of jdk21**  **2. Write a simple java program for printing Name, Class, Roll No, of a Student** |  |
| **2.** | **Week-2** | **1.Write a java program to find simple interest where all inputs are taken from user**  **2.Write a java program to calculate factorial of a number , read the input from user**  **3.Write a java program to calculate the Fibonacci Sequence of a input taken from user**  **4.a: Write a java program to convert temperature from Celsius to Fahrenheit**  **B: Write a java program to convert temperature from Fahrenheit to Celsius**  **5. Write a java program to calculate the area of rectangle**  **6. Write a java program to calculate the area of triangle by using heron’s formula** |  |
| **3.** | **Week-3** | **Create a 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 three methods named start(),stop(),service()**  **4.Create three objects named Car1, Car2 and Car3**  **b) Create a class bankAccount with elements deposit() and Withdrawl** |  |
| **4.** | **Week-4** | **a. 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 parameters which initializes title , author , year of publication**  **Create a method which displays details of the book**  **Display details of two books**  **b. Create a java program with class name myclass with static variable count of int type ; initialized to zero and a constant variable “pi’**  **Of type double initialized to 3.14 as attribute of that class , now define a constructor for “ my class “ that increment the count variable each time an object of my class is created count = 0 ,finally print the final values of “count” ans pi variable ,create 3 objects** |  |
| **5.** | **Week-5** | **a)create a calculator using the operation including adding, subtraction,multiplication,and division using multilevel inheritance and display desired output**  **every class one method (int a ,int b)**  **pass values as parameter during ,method call** |  |
| **6.** | **Week-6** | **a: Write a java program to create a vehicle class with a method displayInfo().Override this method in the car subclass to provide specific information a about car**  **b: A college is developing an automated admission system that verifies student**  **eligibility for UG and PG programs. Each program has different eligibility criteria based on the student’s percentage in their previous Qualification:**  **UG admissions require a minimum of 60%**  **PG admissions require a minimum of 70%**  **C: AIM : Create a Shape class with a method calculateArea() that is overloaded forDifferent shapes. Then, create a subclass circle that overrides the calculateArea() method for a circle**  **D: . Create a calculator class with overloaded methods to perform addition**   1. **Add two integers 2. Add two doubles 3. Add three integers** |  |
| **7.** | **Week-7** | **Aim: Write a java program to create an abstract class Animal with an abstract**  **Method called sound. Create subclasses lion and tiger that extends the animal**  **Class and implements the sound() method to make a specific sound for each animal**  **b. :** **Write a java program to create an abstract class shape3D with**  **abstract methods calculateVolume() and calculateSurfaceArea().Create subclasses sphere and cube that extend the shape3D class and implement**  **the respective methods to calculate the volume and surface area of each shape**  **c. Write a java program using an abstract class to define a method for pattern printing. Create an abstract class named PatternPrinter with an**  **Abstract method printpattern(int n) and a concrete method to display the pattern title.**  **Implement two subclasses**  **1.star pattern- prints a right angled triangle of stars(\*)**  **2.Number pattern- prints a right angled triangle of increasing numbers.**  **In the main() method create objects of both subclasses and print the patterns**  **For a given number of rows.** |  |
| **8.** | **Week-8** | 1. **.AIM: Write a Java program to create an interface Shape with the getPerimeter method. Create three classes Rectangle, Circle, and Triangle that implement the Shape interface. Implement the getPerimeter() method for each of the three classes.**   **b.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 implement the Playable interface and override**  **the play() method to play the respective sports.** |  |
|  |  |  |  |

**WEEEK -1 (LAB )**

1. **Process of Installing JDK (Java Development Kit)**

**Installing JDK (Java Development Kit):**

* 1. **Download JDK:**
* **Go to the Oracle JDK download page in google and click on JDK-21 version which is Long term support (LTS) version.**
* **Click the download link as your operating system (Windows, macOS, or Linux).** 
  1. **Install JDK:**
* **Once downloaded, run the installer.**
* **Follow the given instructions and keep clicking "Next" until it is done.** 
  1. **Set Environment Variables (Windows):**
* **Open file explorer, then right click on This PC next select on properties then it will take you to the settings app then click on advanced system settings and then click on Environment Variables.**
* **Click on path and new under System Variables:**

**Variable value: The folder address where JDK is installed (like**

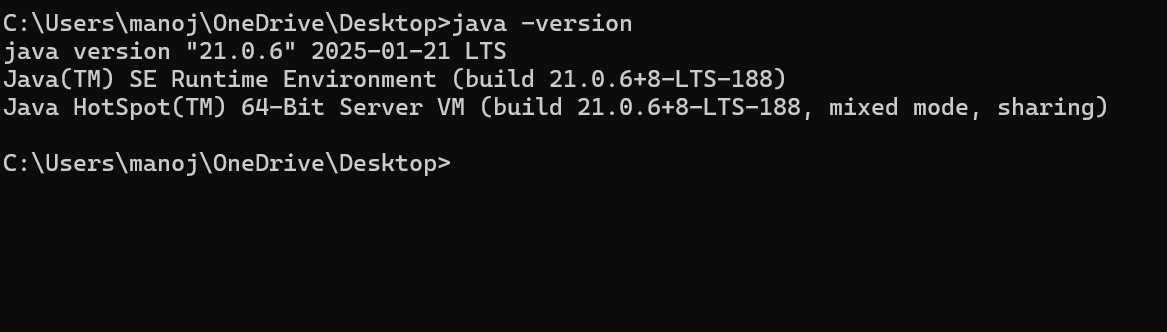
**C:\Program Files\Java\jdk-21\bin)**

* **Find Path under System Variables, click New, and add the path of the jdk-21(C:\Program Files\Java\jdk-21\bin)**

****

**Checking JDK Version: -**

* 1. **Open Command Prompt:**
* **Presswin+R, typecmd, and press Enter.** 
  1. **Check Version:**
* **Type java -version and press Enter.**
* **Type javac --version and press Enter.**

****

**2.Simple Java Program for printing Name, Class, Roll No, of a Student**

**Write your code in Notepad and execute it in cmd prompt**

CODE:

class manoj

{

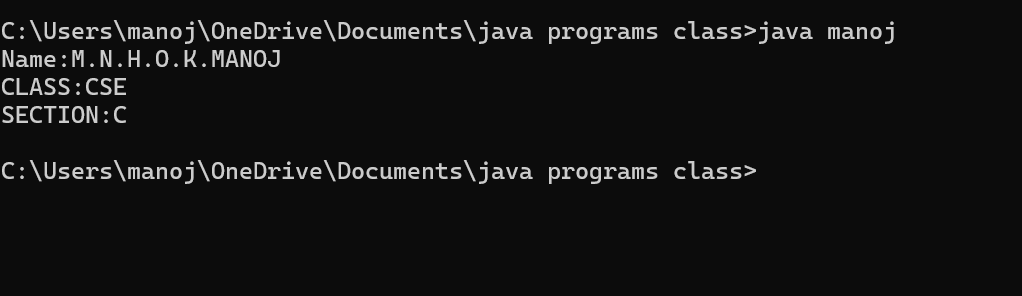
public static void main(String[] args)

{

System.out.println("Name: M.N.H.O.K. MANOJ ");

System.out.println("CLASS: CSE");

System.out.println("SECTION:C"); }}

**Output: -**

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| **1** | **Syntax error** | **Semicolon added** |
| **2** | **Runtime error** | **Copied correct path** |
| **3** | **Name error** | **rectified** |

**Important points :** System.out.println( ); is used to print the statements , “ ; ” is used at end of a statement

**WEEK -2 (LAB)**

**SIMPLE JAVA PROGRAMS**

**1.AIM:**

**Write a java program to write area of rectangle.**

**Code:**

class demo {

public static void main(string[] args) {

Scanner sc = new Scanner(System.in);

System.out.println(“enter a number:”);

int l = sc.nextInt();

System.out.println(“enter a number;”);

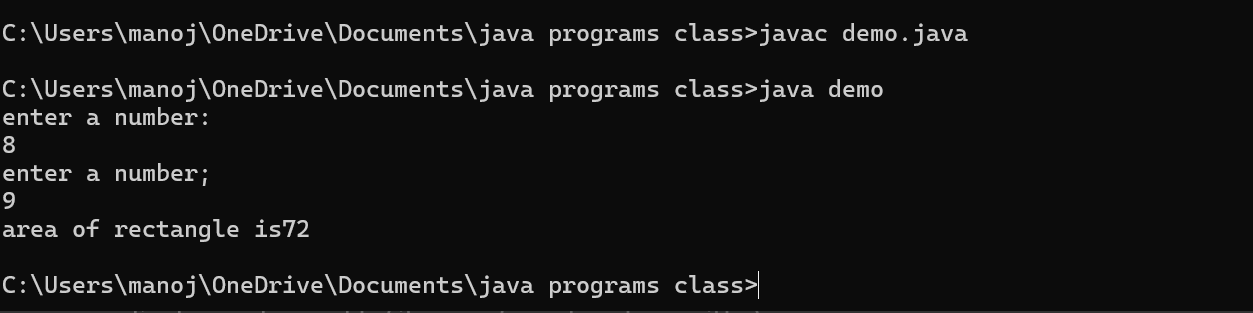
int b = sc.nextInt();

int a = l\*b;

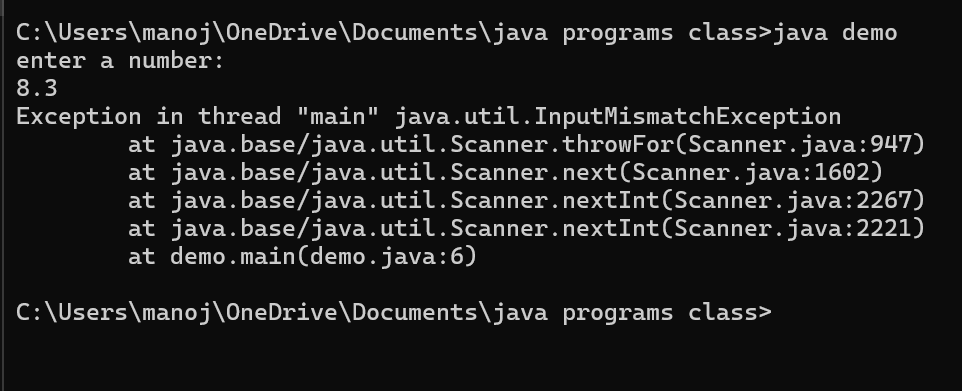
System.out.println(“area of rectangle is “+a); }}

**Output :**

**1.Positive case**

****

**2.Negative case :**



**Errors table :**

|  |  |  |
| --- | --- | --- |
| 1 | Syntax error | Semicolon added |
| 2. | Name error | rectified |

**IMPORTANT POINTS:** AREA OF RECTANGLE IS AREA = LENGTH \* BREATH

**2.AIM :**

**Write a java program to find simple intrest where all input from user**

**Code :**

import java.util.\*;

class test{

public static void main(String[] args)

{

System.out.println(" taking input");

Scanner sc = new Scanner(System.in);

System.out.println("enter INTa number:");

float p = sc.nextFloat();

System.out.println("enter a number:");

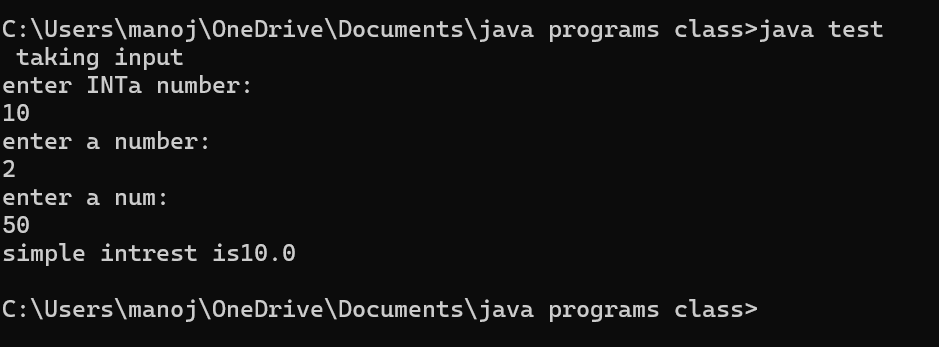
float t = sc.nextFloat();

System.out.println("enter a num:");

float r = sc.nextFloat();

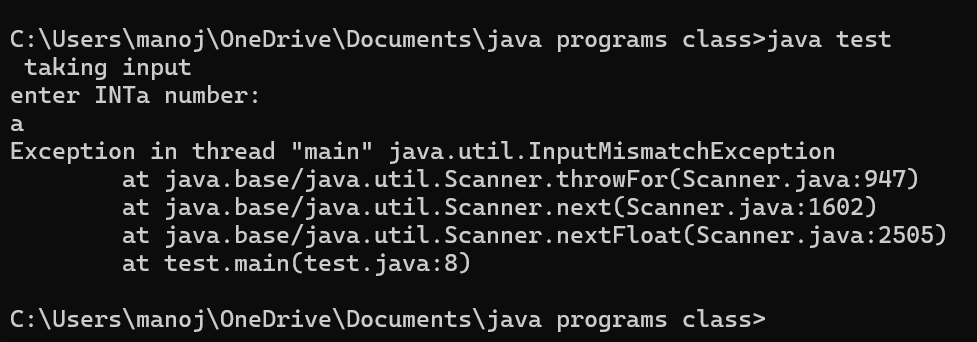
float s = (p\*t\*r)/100;

System.out.println("simple intrest is"+s);

****}}

**Output:**

1.Positive case :

**2.Negative case :**

**Error table :**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **rectification** |
| **1** | **Runtime error** | **Incorrect path** | **Copied correct path** |
| **2** | **Syntax error** | **Using incorrect symbols** | **Using correct symbols** |
| **3** | **Logical error** | **Wrong formula** | **Formula rectified** |

**IMPORTANT POINTS :**

**simple intrest formula : p\*t\*r/100.**

**also to give value from input we used :**

float r = sc.nextFloat();

3.**Fibonacci sequence to caculate Fibonacci sequence read from user .**

**Code:**

Import java.util.\*;

class zeebra {

public static void main(String args[]) {

Scanner sc = new Scanner(System.in);

int num;

int f3;

int f1 = 0;

int f2 = 1;

int i = 2;

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

num = sc.nextInt();

System.out.println(f1);

System.out.println(f2);

while(i<num ){

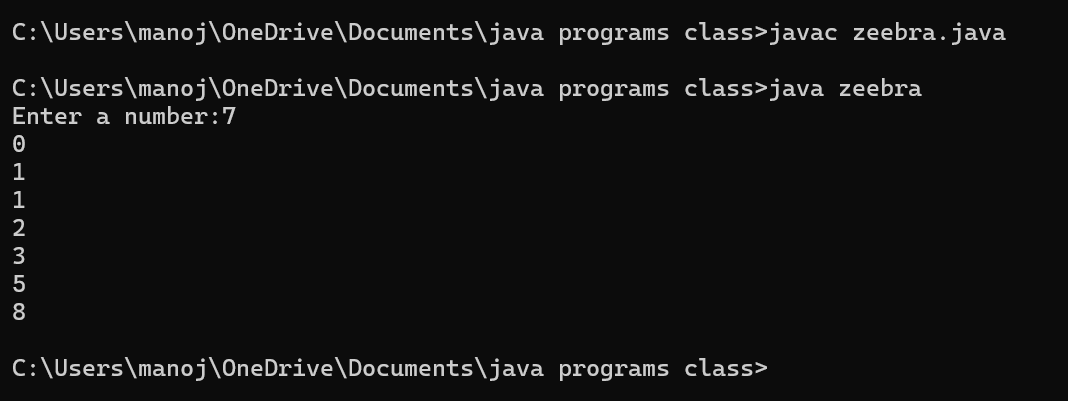
f3 = f1+f2;

f1 = f2;

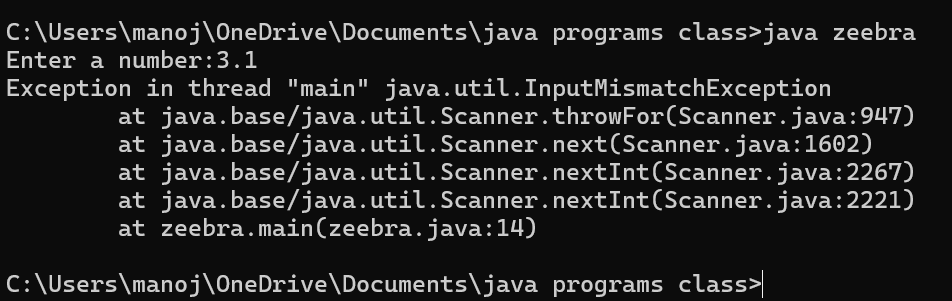
f2 = f3;

System.out.println(f3);

i = i+1; } } }

**1.Positive case :**

**2.Negative case :**

****

**Error table :**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| **1** | **Logical error** | **Incorrect formula** | **Using correct formula** |
| **2** | **Syntax error** | **Incorrect symbols** | **Using correct symbols** |
| **3** | **Runtime error** | **Incorrect path** | **Copied correct path** |

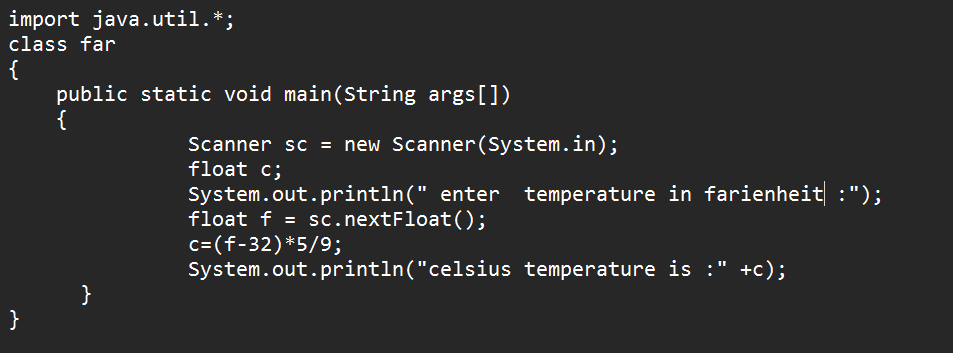
**Important points:**

1. **The code generates and prints the first num terms of the Fibonacci series using a while loop.**

**It starts with initial values f1 = 0 and f2 = 1, then iteratively calculates the next terms by summing the previous two**

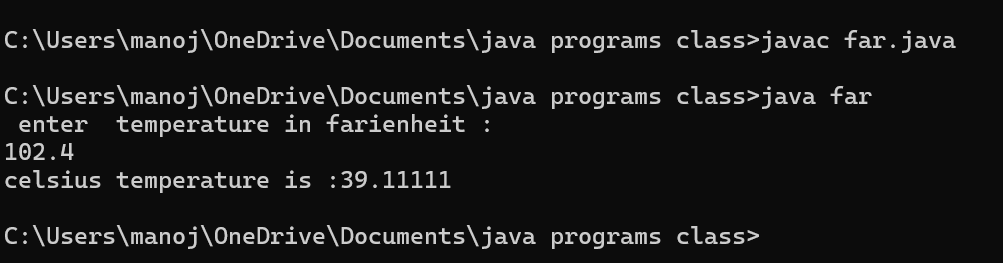
**4.AIM: write a java program to convert temperature from fahrenheat to celcius**

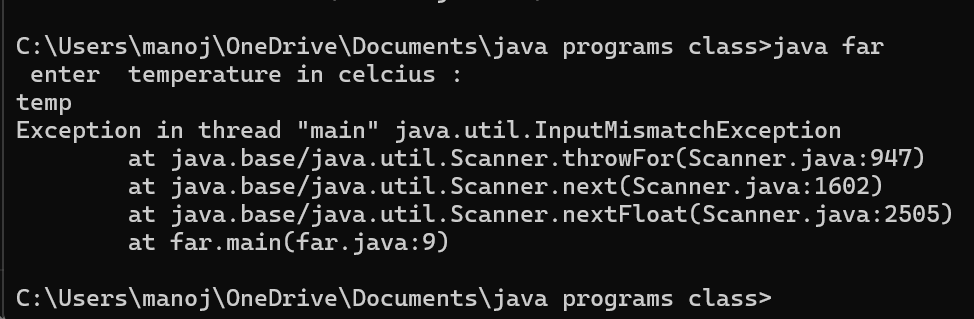
**Code :**

****

**OUTPUT:**

1. Positive case :



**2.negative case:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **rectification** |
| **1.** | **Syntax** | **Incorrect symbol** | **Using correct symbols** |
| **2.** | **Logical error** | **Due to incorrect formula** | **Corrected by giving correct formula** |
| **3.** | **Runtime error** | **Incorrect path** | **Using correct path** |

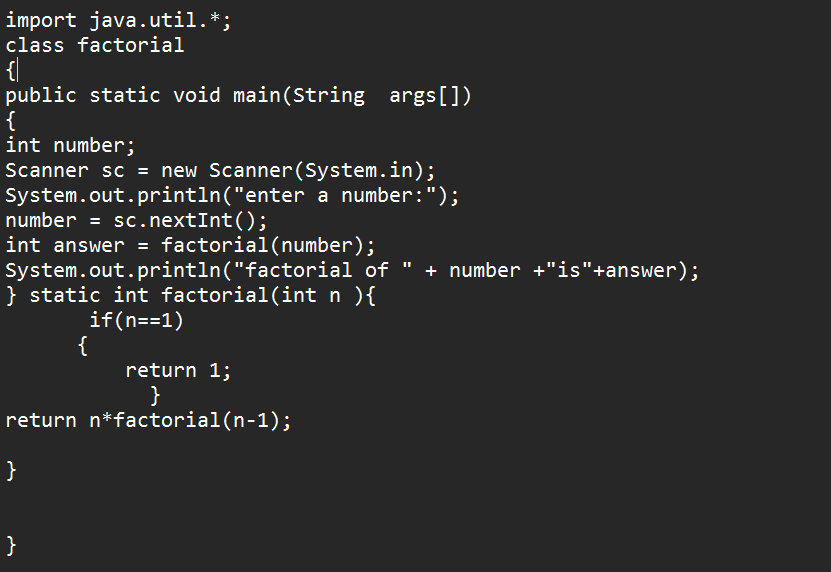
**IMPORTANT POINTS :**

1. The program takes temperature input in Celsius and converts it to Fahrenheit using the formula: F = (C × 9/5) + 32.

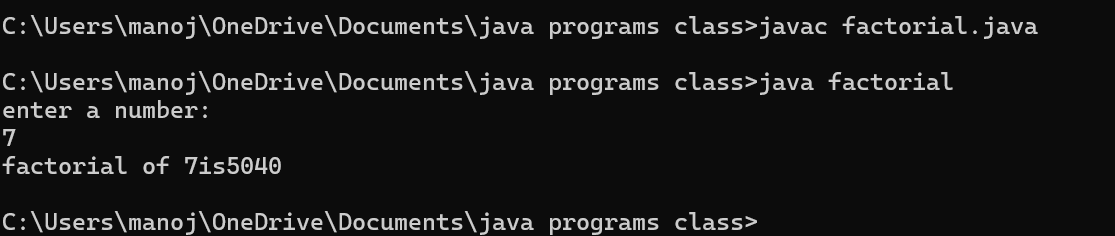
2. It uses Scanner for input and prints the converted temperature as output.

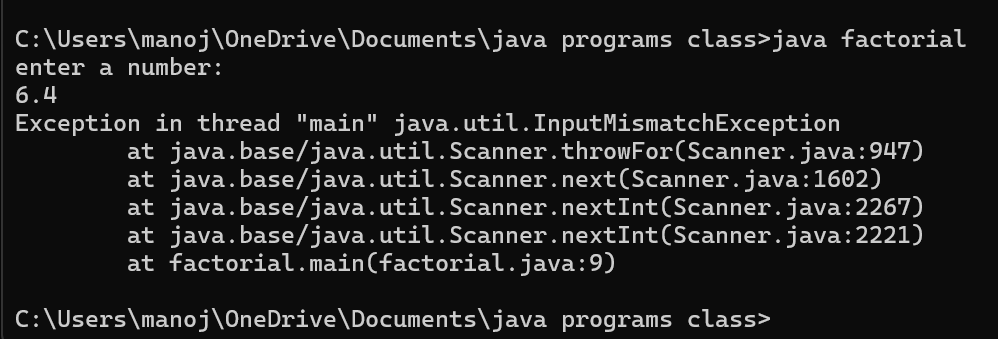
**5.AIM:**

**Write a java program to calculate factorial of a number read from user**

**Code:**

Output:

1.Positive case :

2.Negative case:

**Error table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **rectification** |
|  | **Logical error** | **Incorrect formula** | **Correcting formula** |
|  | **Runtime error** | **Incorrect path** | **Using correct path** |
|  | **syntax** | **Incorrect symbols** | **Using correct symbols** |

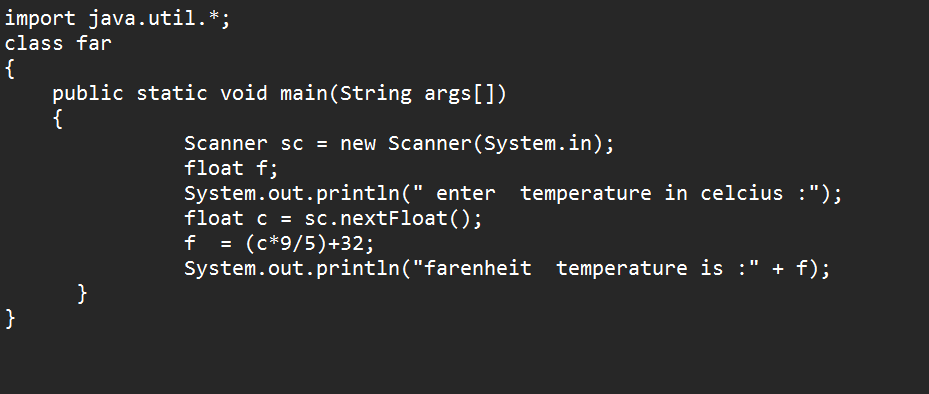
IMPORTANT POINTS :

1.The program calculates the factorial of a given number using a loop (or recursion) by multiplying numbers from 1 to n.

2. It takes user input with Scanner and displays the result after computation.

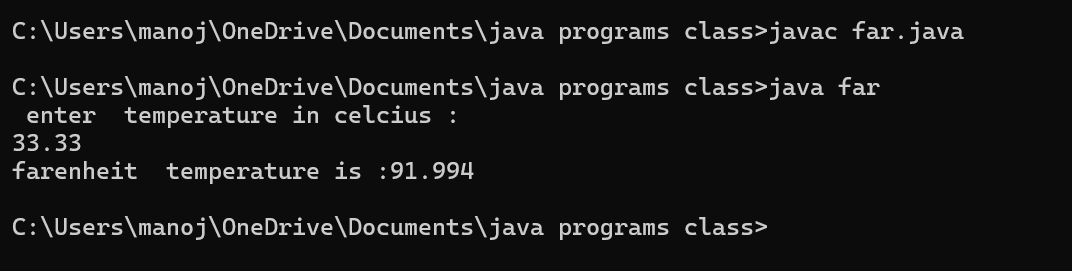
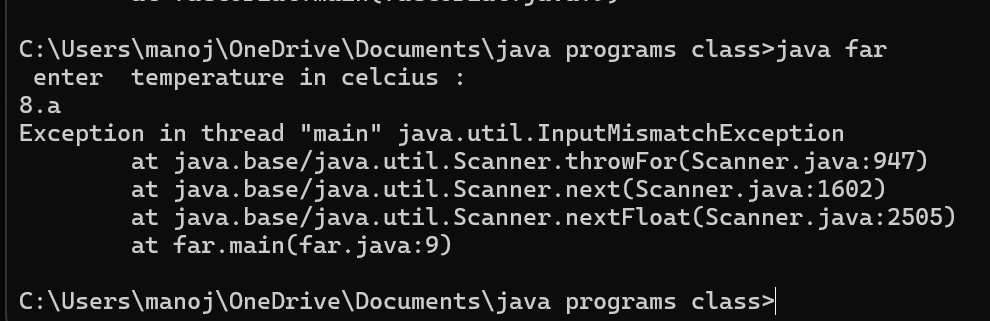
**6. AIM:Write a Java program to convert temperature from celcius to farenheit:**

**Code:**



**Output:**

1. Positive case :

2. Negative case:

Error table:

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **rectification** |
| **1.** | **Logical error** | **Incorrect formula** | **Correcting formula** |
| **2.** | **Runtime errror** | **Incorrect path** | **Using correct path** |
| **3.** | **Syntax error** | **No symbols** | **Using correct symbols** |

**IMPORTANT POINTS :**

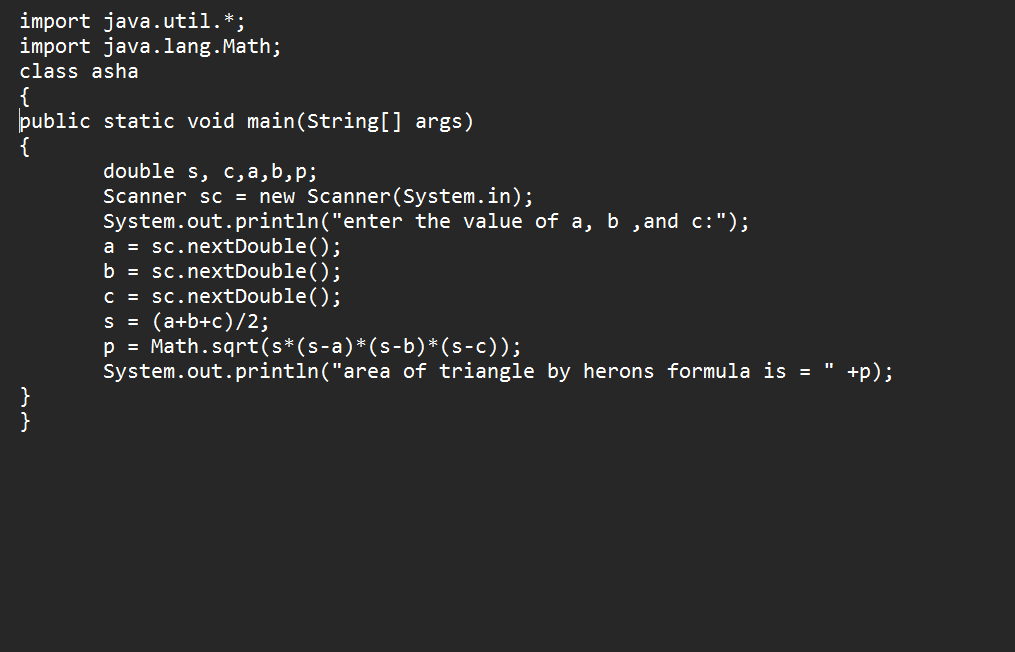
1.The program converts temperature from Fahrenheit to Celsius using the formula: C = (F - 32) × 5/9.

2. It takes Fahrenheit input from the user using Scanner and prints the Celsius result.

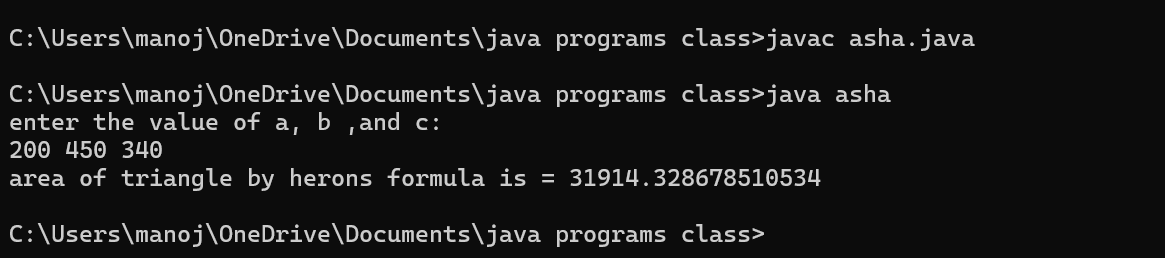
**7. Aim :**

**Write a Java program to find area of triangle using herons’s formula :**

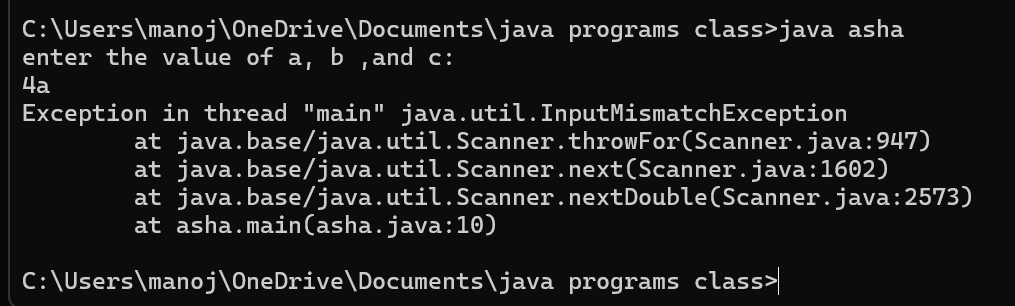
**Code:**

Output:

1.Positive case :



2.Negative case :



**Error table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **rectification** |
| **1.** | **Logical error** | **Incorrect formula** | **Correcting formula** |
| **2.** | **Runtime errror** | **Incorrect path** | **Using correct path** |
| **3.** | **Syntax error** | **Incorrect symbols** | **Using correct symbols** |

**IMPORTANT POINTS:**

1. The program calculates the area of a triangle using **Heron's formula**:  
Area = √(s(s−a)(s−b)(s−c)), where s = (a + b + c) / 2.

2. It takes the lengths of all three sides as input using Scanner and then computes the area using the for

**Week -3(lab)**

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, milage

1. Create three methods named start(),stop(),service()
2. Create three objects named car1 ,car2 ,car3.

**CLASS DIAGRAM :**

|  |
| --- |
| **Class Name = Car** |
| **Attributes :**  **-Color= String**  **-Brand = String**  **- Fuel = String**  **-Milage= Int** |
| **Methods**  **+Start():Void**  **+Stop():Void**  **+Service() :Void** |
|  |

**Code :**

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

**class car {**

**public String color;**

**public String brand;**

**public String fuel;**

**public int mileage;**

**public void start() {**

**System.out.println("Car Started:");**

**System.out.println("Car color is :"+color);**

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

**System.out.println("Car fuel type is:"+fuel);**

**System.out.println("Car mileage is:"+mileage); }**

**public void service(){**

**\System.out.println("Car Service:");**

**System.out.println("Car color is :"+color);**

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

**System.out.println("Car fuel type is:"+fuel);**

**System.out.println("Car mileage is:"+mileage); }**

**public void stop(){**

**System.out.println("Car Stop:");**

**System.out.println("Car color is :"+color);**

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

**System.out.println("Car fuel type is:"+fuel);**

**System.out.println("Car mileage is:"+mileage)}**

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

**car car1 = new car();**

**car1.color = "Black";**

**car1.brand = "MARUTHI";**

**car1.fuel = "Deisel";**

**car1.mileage = 80;**

**car1.start();**

**car car2 = new car();**

**car2.color = "BLUE";**

**car2.brand = "hundai";**

**car2.fuel = "petrol";**

**car2.mileage = 120;**

**car2.stop();**

**car car3 = new car();**

**car3.color = "maroon";**

**car3.brand = "shift";**

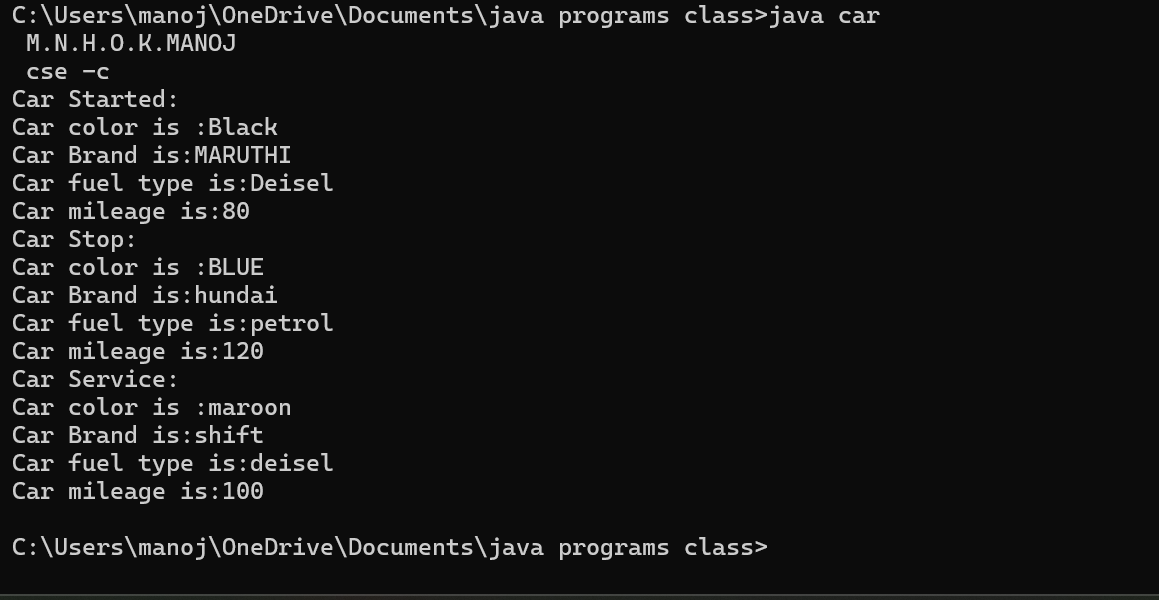
**car3.fuel = "deisel";**

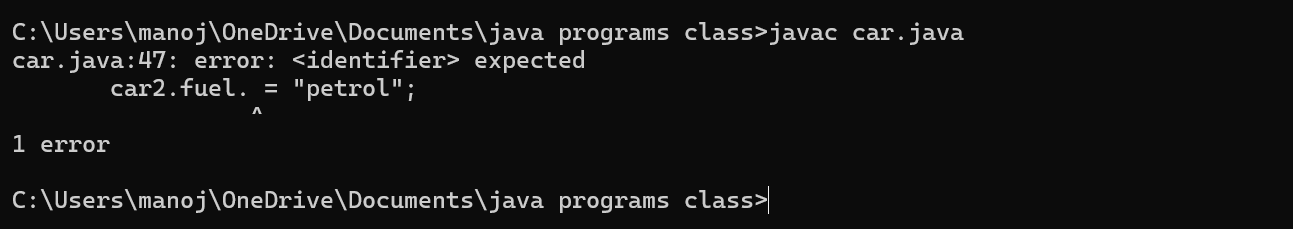
**car3.mileage = 100;**

**car3.service() } }**

**Output :**

**1.Positive case**

****

**2.Negative case** 

**Error table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **rectification** |
| **1.** | **Logical error** | **Incorrect formula** | **Correcting formula** |
| **2.** | **Runtime errror** | **Incorrect path** | **Using correct path** |
| **3.** | **Syntax error** | **Incorrect symbols** | **Using correct symbols** |

**IMPORTANT POINT:**

1. CONSTRUCTOR: A constructor in Java is a special method that is used to initialize objects. It is automatically called when an object of a class is created.

Key Features of a Constructor:

1. Same Name as the Class: The constructor name must be the same as the class name.
2. No Return Type: Constructors do not have a return type (not even void).
3. Called Automatically: It runs automatically when an object is created.

ATTRIBUTES:

In Java, attributes (also called fields or instance variables) are used to store the state or properties of an object.

METHODS:

A method in Java is a block of code that performs a specific task. It is used to increase code reusability, modularity, and maintainability

**2.AIM:**

**TO CREATE A CLASS BANK ACCOUNT WITH METHODS DEPOSIT() AND WITHDRAWL()**

**CLASS DIAGRAM:**

|  |
| --- |
| **CLASS = BankAccount** |
| **ATTRIBUTES:**  **- Acchname: String**  **- Accnumber: int**  **- curramount: double** |
| **METHODS:**  **+ BankAccount(Acchname: String, Accnumber: int, curramount: double)**  **+ Withdrawl:Void**  **+Deposit :Void** |
|  |

**CODE:**

class BankAccount {

public String Acchname;

public int Accnumber;

public double curramount;

BankAccount(String Acchname, int Accnumber, double curramount) {

this.Acchname = Acchname;

this.Accnumber = Accnumber;

this.curramount = curramount;

System.out.println("Account Created:");

System.out.println("Account Holder: " + Acchname);

System.out.println("Account Number: " + Accnumber);

System.out.println("Current Balance: $" + curramount);

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

}

public void withdraw(double withdraw) {

if (withdraw > curramount) {

System.out.println("Insufficient Funds!");

} else {

curramount -= withdraw;

System.out.println("Withdrawn: $" + withdraw);

System.out.println("Updated Balance: $" + curramount);

}

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

}

public void deposit(double deposit) {

curramount += deposit;

System.out.println("Deposited: $" + deposit);

System.out.println("Updated Balance: $" + curramount);

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

}

public static void main(String args[]) {

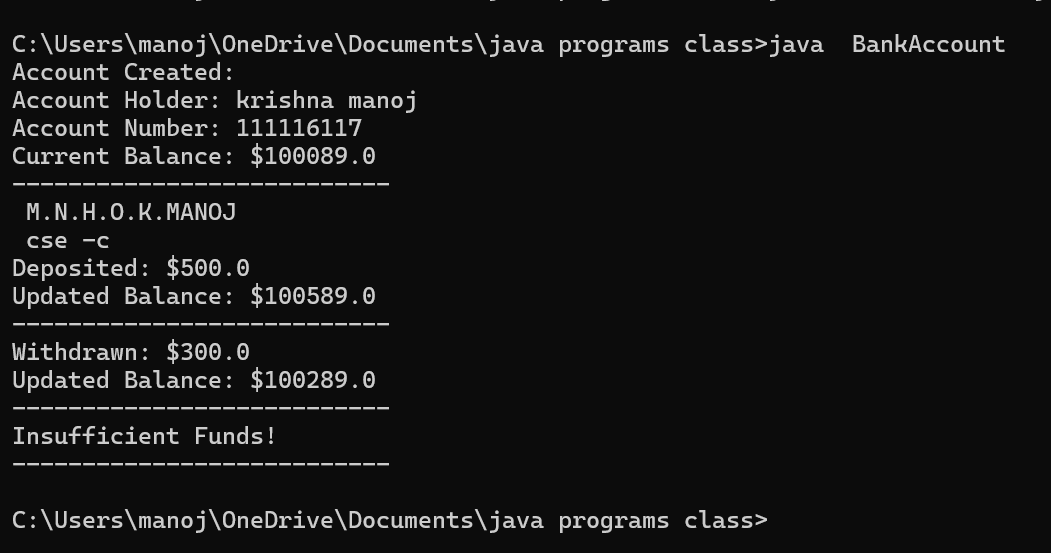
BankAccount account1 = new BankAccount("krishna manoj", 116117, 100089.0);

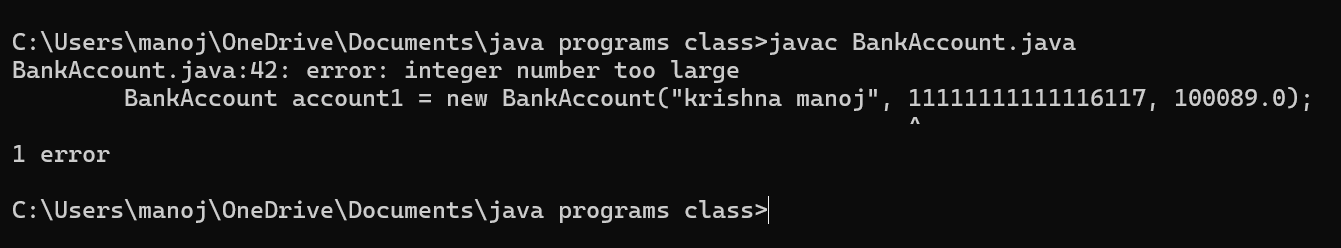
account1.deposit(500);

account1.withdraw(300);

account1.withdraw(1000500); Funds"

**OUTPUT:**

**1.POSITIVE CASE:** ****

**2. NEGATIVE CASE:**

**Error table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **rectification** |
| **1.** | **Logical error** | **Incorrect formula** | **Correcting formula** |
| **2.** | **Runtime errror** | **Incorrect path** | **Using correct path** |
| **3.** | **Syntax error** | **Incorrect symbols** | **Using correct symbols** |

**IMPORTANT POINTS:**

**1. The BankAccount class simulates a simple bank account with methods to deposit and withdraw money.**

**2.It uses a constructor to initialize account details like holder name, account number, and current balance.**

**3. The deposit and withdraw methods update and display the account balance, handling insufficient funds in withd**

**WEEK – 4 (LAB)**

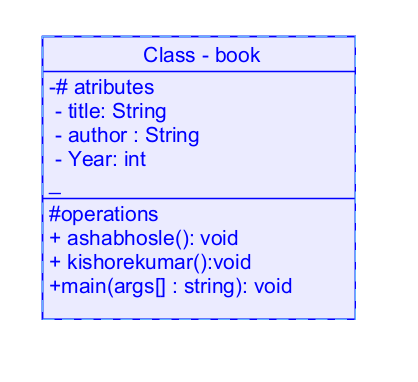
1. **AIM:**

**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 parameters which initializes title , author , year of publication**

**Create a method which displays details of the book**

**Display details of two books**

**CLASS DIAGRAM :**

****

**CODE:**

import java.util.\*;

class book {

public String title;

public String author;

public int year;

public void ashabhosle() {

System.out.println(" book - ashabhosle:");

System.out.println("the title is :"+title);

System.out.println("THE NAME OF AUTHOR : " +author);

System.out.println("YEAR OF PUBLICATION :"+year);

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

public void kishorekumar(){

System.out.println(" book -kishorekumar:");

System.out.println("the title is :"+title);

System.out.println("THE NAME OF AUTHOR : " +author);

System.out.println("YEAR OF PUBLICATION :"+year); System.out.println("-------------------------------------------");

public static void main(String args[]) {

book book1 = new book();

book1.author = " R .K .NARAYAN " ;

book1.title = " MALGUDI DAYS " ;

book1.year = 1953;

book1.ashabhosle() ;

book book2 = new book();

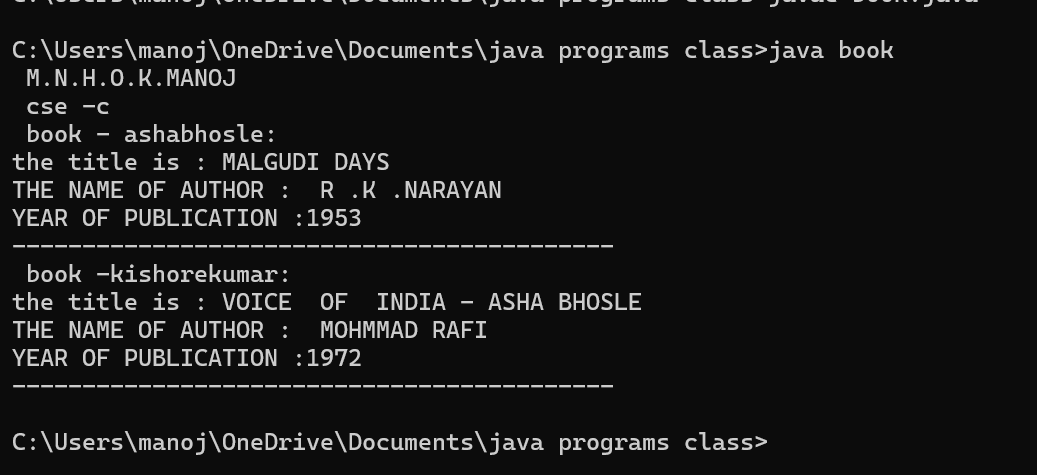
book2.author = " MOHMMAD RAFI ";

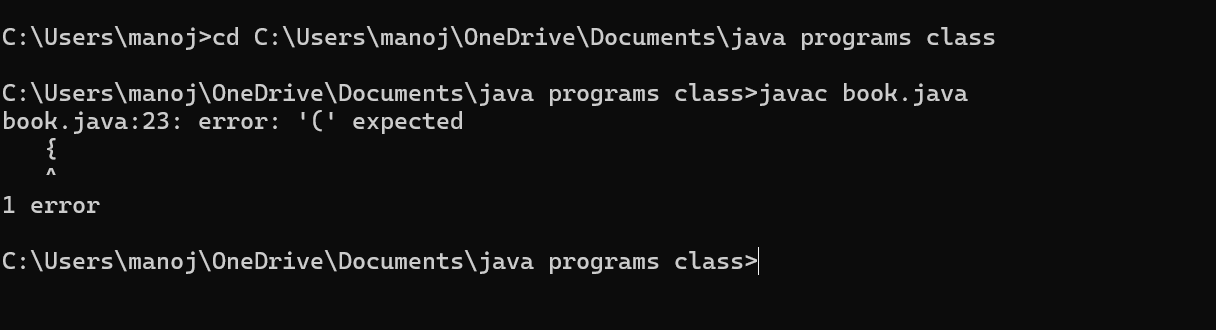
book2.title = " VOICE OF INDIA - ASHA BHOSLE" ;

book2.year = 1972;

book2.kishorekumar(); } }

**OUTPUT** :

**1.Positive cases :**

**2.Negative case :**

**ERROR TABLE:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **rectification** |
| **1.** | **Logical error** | **Incorrect formula** | **Correcting formula** |
| **2.** | **Runtime errror** | **Incorrect path** | **Using correct path** |
| **3.** | **Syntax error** | **Incorrect symbols** | **Using correct symbols** |

**IMPORTANT POINTS:**

**1. The book class stores details about a book: title, author, and year of publication.**

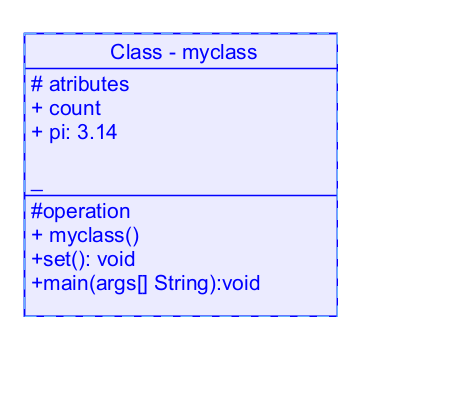
**2. It has two methods, ashabhosle() and kishorekumar(), which display book details in a formatted way.**

**3. In main(), two book objects are created and initialized manually, then their details are printed using the respective methods.**

**2. AIM:**

**Create a java program with class name myclass with static variable count of int type ; initialized to zero and a constant variable “pi’Of type double initialized to 3.14 as attribute of that class , now define a constructor for “ my class “ that increment the count variable each time an object of my class is created count = 0 ,finally print the final values of “count” ans pi variable ,create 3 objects**

**CLASS DIAGRAM :**



**Code:**

class myclass {

static int count=0;

final double pi=3.14;

myclass() {

count = count+1; }

public void set()

{

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

System.out.println("Pi value is:"+pi); }

public static void main(String args[])

{

myclass m = new myclass();

System.out.println(count);

m.set();

myclass n = new myclass();

System.out.println(count);

n.set();

myclass l = new myclass();

System.out.println(count);

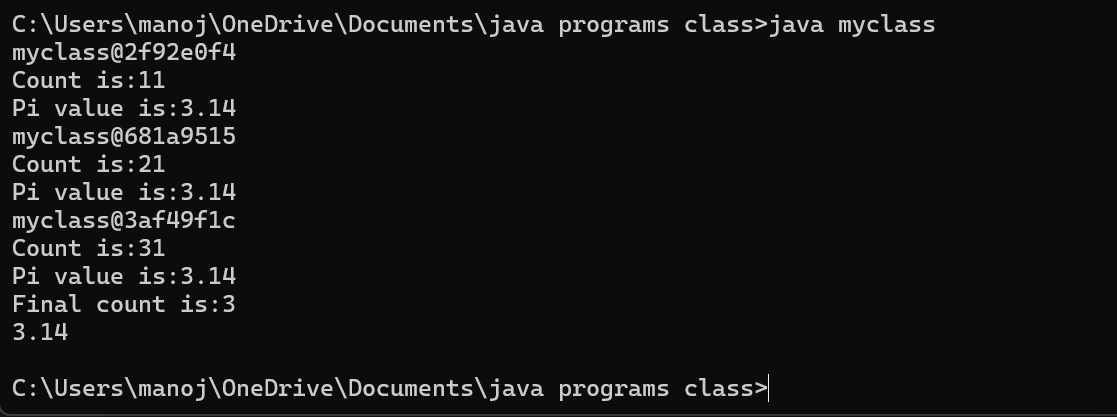
l.set();

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

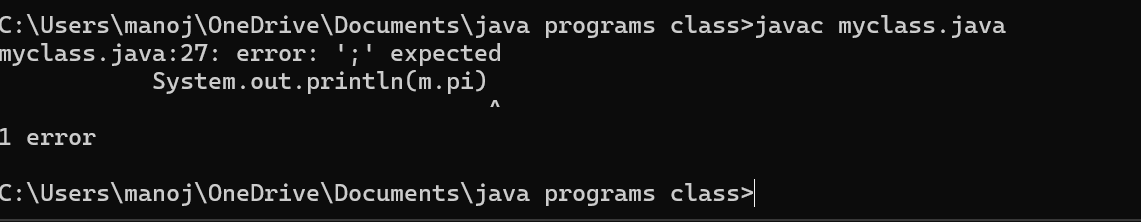
System.out.println(m.pi);}}

**OUTPUT:**

1**.positive case:**



**2.negative case:**

Error table:

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **rectification** |
| **1.** | **Logical error** | **Incorrect formula** | **Correcting formula** |
| **2.** | **Runtime errror** | **Incorrect path** | **Using correct path** |
| **3.** | **Syntax error** | **Incorrect symbols** | **Using correct symbols** |

**IMPORTANT POINTS:**

1. The myclass class uses a **static variable count** to track how many objects are created, incrementing it in the constructor.
2. It also defines a **final constant pi** with a value of 3.14, which cannot be changed.
3. Each time a new object is created, count increases, and the set() method displays the current count and value of pi.

**Week -5**

**Aim:**

1. **create a calculator using the operation including adding, subtraction,multiplication,and division using multilevel inheritance and display desired output**

**every class one method (int a ,int b)**

**pass values as parameter during method call**

**Classs diagram:**

|  |
| --- |
| **Class : addition** |
| **+ add( int a ,int b)** |
|  |

|  |
| --- |
| **Class : subtraction** |
| **+ sub( int a ,int b)** |

|  |
| --- |
| **Class : multiplication** |
| **+ multi( int a ,int b)** |

|  |
| --- |
| **Class : divission** |
| **+ div( int a ,int b)** |

**program:**

**class addition{**

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

**System.out.println("addition is done in calculator below:");**

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

**int addition = a+b;**

**return addition; } }**

**class subtraction extends addition {**

**public int sub(int a, int b){**

**System.out.println("subtraction is done in calculator below:");**

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

**int subtraction = a-b;**

**return subtraction; }}**

**class multiplication extends subtraction{**

**public int mult(int a, int b) {**

**System.out.println("multiplication is done in calculator below:");**

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

**int multiplication = a\*b;**

**return multiplication; }}**

**class division extends multiplication{**

**public int div(int a,int b){**

**System.out.println("divission is done in calculator below:");**

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

**int division = a/b;**

**return division; }}**

**class calculator**

**{**

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

**{**

**division obj = new division();**

**addition img = new addition();**

**subtraction del = new subtraction();**

**multiplication zym = new multiplication();**

**System.out.println("Addition is:"+ img.add(99,1)); System.out.println("----------------------------------");**

**System.out.println ("Subtraction is:"+del.sub(200,100));**

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

**System.out.println("Multiplication is:"+zym.mult(50,2));**

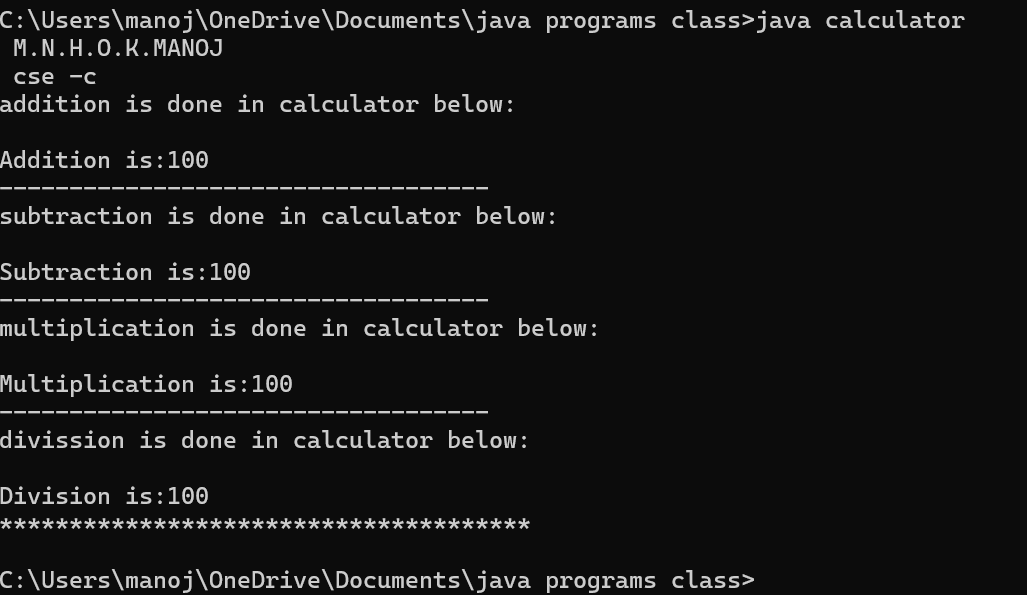
**System.out.println("--------------------------------");**

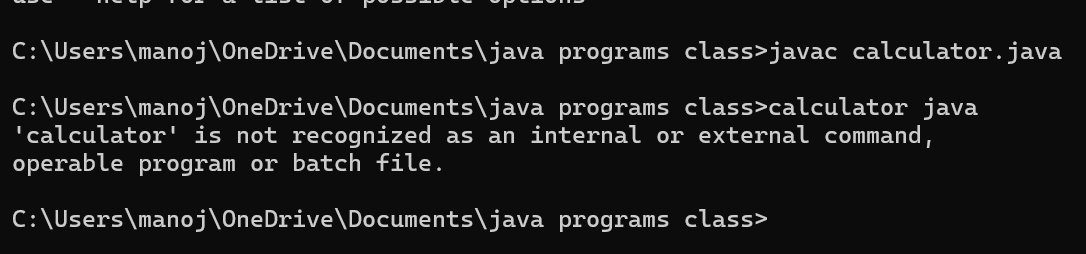
**System.out.println("Division is:"+obj.div(1000,10));**

**System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"); }}**

**OUTPUT:**

1. **positive case:**

****

****

**Error table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **rectification** |
| **1.** | **Logical error** | **Incorrect formula** | **Correcting formula** |
| **2.** | **Runtime errror** | **Incorrect path** | **Using correct path** |
| **3.** | **Syntax error** | **Incorrect symbols** | **Using correct symbols** |

**IMPORTANT POINTS:**

1.This program demonstrates multilevel inheritance, where each class adds a new arithmetic operation: addition, subtraction, multiplication, and division.

2.Methods in each class perform the respective operation and return the result, along with descriptive print statements.

3. In the main() method, objects are created to access and test all operations, showcasing inheritance and method calling.

**B: aim:**

**A vehicle rental company wants to develop a system that maintains Information about different types of vehicles available for rent The Company rents out cars, bikes and truck and they need a program to Store details about each vehicle, such as brand and speed Cars should have an additional property: number of doors Bikes should have a property indicating whether they have gears or not The system should also include a function to display details about each vehicle And indicate when a vehicle is starting**

**CLASS DIAGRAM :**

|  |
| --- |
| Class vehicle |
| + brand:string  + speed :int |
| Vehicle(String brand ,int speed) |

|  |
| --- |
| Class bike |
| + gear : bool |
| bike(String brand ,int speed,Boolean grars) |

|  |
| --- |
| Class car |
| +noofdoors:int |
| car(String brand ,int speed ,noofdoors) |

|  |
| --- |
| Class truck |
| + weight:int |
| truck(String brand ,int speed) |

Program:

class vehicle{

String brand;

int speed;

public vehicle(String brand,int speed){

this.brand=brand;

this.speed=speed; }

public static void main(String[] args) {

car obj1=new car("maruti",133,7);

bike obj2=new bike("unicorn",75,true);

truck obj3=new truck("zimban",120,16); }}

class car extends vehicle{

int noofdoors;

public car(String brand, int speed,int noofdoors) {

super(brand, speed);

this.noofdoors=noofdoors;

System.out.println(brand);

System.out.println(speed);

System.out.println(noofdoors); }

class bike extends vehicle{

boolean gears;

public bike(String brand,int speed,boolean gears){

super(brand, speed);

this.gears=gears;

System.out.println(brand);

System.out.println(speed);

System.out.println(gears); }}

class truck extends vehicle{

int weight;

public truck(String brand,int speed,int weight){

super(brand,speed);

this.weight=weight;

System.out.println(brand);

System.out.println(speed);

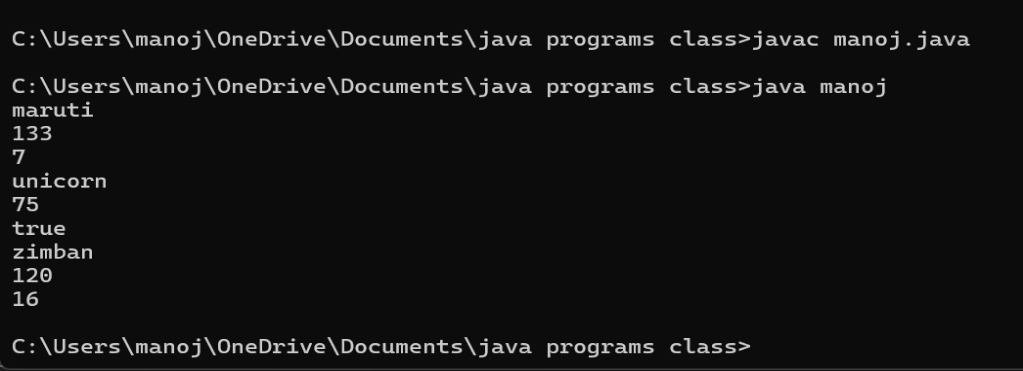
System.out.println(weight); }

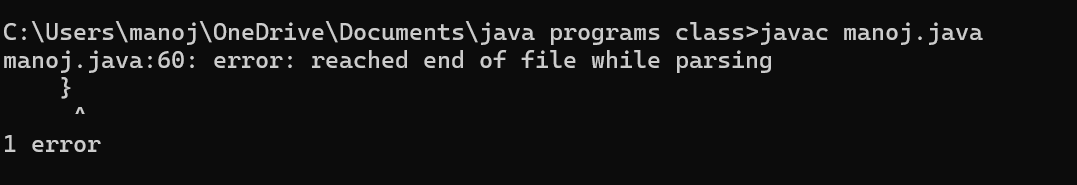
public static void main(String[] args) {

car obj1=new car("maruti",133,7);

bike obj2=new bike("unicorn",75,true);

truck obj3=new truck("zimban",120,16); } }

**Out put: 1.positive case :**

2.negative case:Error table:

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **rectification** |
| **1.** | **Logical error** | **Incorrect formula** | **Correcting formula** |
| **2.** | **Runtime errror** | **Incorrect path** | **Using correct path** |
| **3.** | **Syntax error** | **Incorrect symbols** | **Using correct symbols** |

**Important points:**

1.Inheritance:

Obtaining the qualities from parent class.

2.hirechial inheritance: different child class obtaing qualities from one parent.

Week- 6:

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 a about car**

CLASS DIAGRAM:

|  |
| --- |
| CLASS VEHICLE |
| ATTRIBUTES :  + SPEED  + BRAND  + CAPACITY |
| +DISPLAYINFO():void  + VEHICLE (INT ,SPEED INT CAPACITY ,STRING BRAND) |

|  |
| --- |
| class car |
| car(int speed ,int capacity ,string brand)  displayinfo() : void |

**Code:**

class Vehicle {

String brand;

int speed;

int capacity;

Vehicle(String brand, int speed, int capacity) {

this.brand = brand;

this.speed = speed;

this.capacity = capacity;

public void displayInfo() {

System.out.println("Vehicle Details:");

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

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

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

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

class Car extends Vehicle {

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

super(brand, speed, capacity); }

public void displayInfo() {

System.out.println("Car Details:");

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

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

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

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

public class world {

public static void main(String[] args) {

Vehicle vehicle = new Vehicle("Luna", 78, 5);

vehicle.displayInfo();

System.out.println();

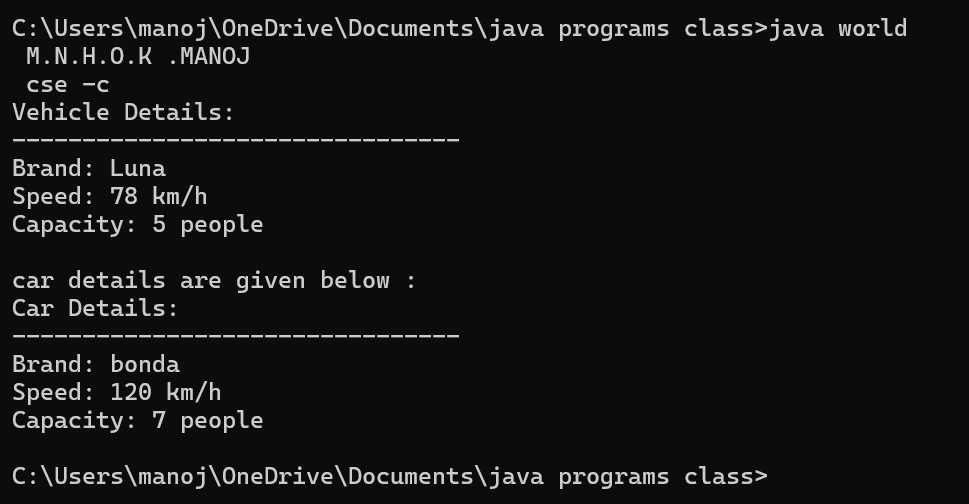
Car car = new Car("bonda", 120, 7);

System.out.println("car details are given below :");

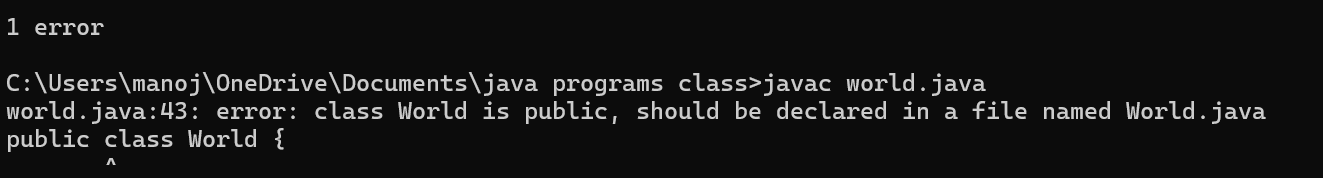
car.displayInfo(); }}

OUT PUT :

1.POSITIVE CASE :



2.NEGATIVE CASE :



**Important points:**

1. **Inheritance** allows the Car class to reuse the Vehicle class functionality without repeating code.

2. **Method Overriding** enables the Car class to provide its own implementation of the displayInfo() method.

3. **Polymorphism** makes the code flexible, allowing different classes (e.g., Car, Bike) to provide customized behavior for displayInfo().

2.AIM:

**A college is developing an automated admission system that verifies student eligibility for UG and PG programs. Each program has different eligibility criteria based on the student’s percentage in their previous Qualification:**

**UG admissions require a minimum of 60%**

**PG admissions require a minimum of 70%**

**Class diagram :**

|  |
| --- |
| University |
| - name: String  - percentile: int |
| + University(String, int)  + office(): void |

|  |
| --- |
| UG |
| + UG(String, int)  + office(): void |

|  |
| --- |
| PG |
| + PG(String, int)  + office(): void |

|  |
| --- |
| admissions |
| + main(String[]): void |

Code:

class University {

String name;

int percentile;

University(String name, int percentile) {

this.name = name;

this.percentile = percentile;}

public void office() {

System.out.println("Admission criteria are mentioned below."); }}

class PG extends University {

PG(String name, int percentile) {

super(name, percentile); }

public void office() {

if (percentile >= 70) {

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

} else {

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

class UG extends University {

UG(String name, int percentile) {

super(name, percentile); }

public void office() {

if (percentile >= 60) {

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

} else {

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

public class admissions {

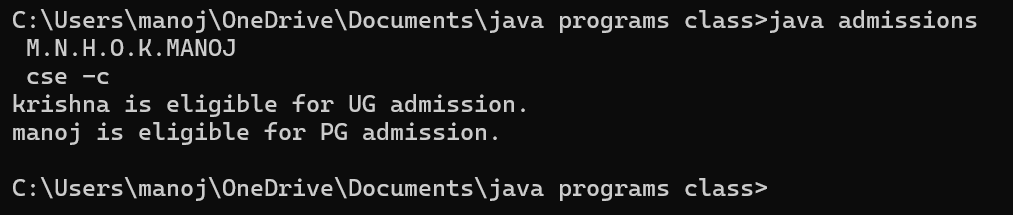
public static void main(String[] args) {

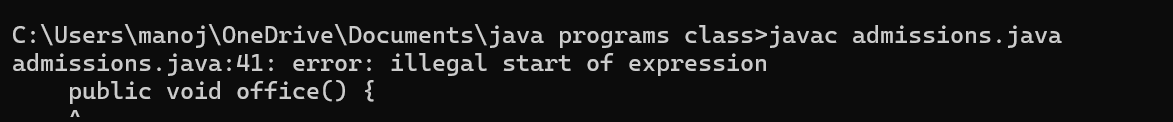
UG ugStudent = new UG("krishna", 95);

PG pgStudent = new PG("manoj", 96);

ugStudent.office();

pgStudent.office(); }

**OUTPUT : 1.positive case :** **2.negative case :**

**Error table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **rectification** |
| **1.** | **Logical error** | **Incorrect formula** | **Correcting formula** |
| **2.** | **Runtime errror** | **Incorrect path** | **Using correct path** |
| **3.** | **Syntax error** | **Incorrect symbols** | **Using correct symbols** |

**IMPORTANT POINTS:**

1.**Inheritance** allows PG and UG classes to reuse the University class’s attributes and constructor.

2. **Method Overriding** customizes the office() method in PG and UG classes to implement specific admission criteria.

3. **Polymorphism** enables the same office() method to behave differently for PG and UG objects based on their typ

3.**AIM** **: Create a Shape class with a method calculateArea() that is overloaded forDifferent shapes. Then, create a subclass circle that overrides the calculateArea() method for a circle**

**Class diagram :**

|  |
| --- |
| **Class shapes** |
| **+ calculateArea(int, int): void**  **+ calculateArea(double, double): void**  **+ calculateArea(int): void** |

|  |
| --- |
| Class circle |
| + calculateArea(double): void |

|  |
| --- |
| Area |
| + main(String[]): void |

**CODE**:

class Shape {

public void calculateArea(int length, int width) {

int area = length \* width;

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

public void calculateArea(double base, double height) {

double area = 0.5 \* base \* height;

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

public void calculateArea(int side) {

int area = side \* side;

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

class Circle extends Shape {

public void calculateArea(double radius) {

double area = 3.14 \* radius \* radius;

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

public class area {

public static void main(String[] args) {

Shape shape = new Shape();

shape.calculateArea(5, 45);

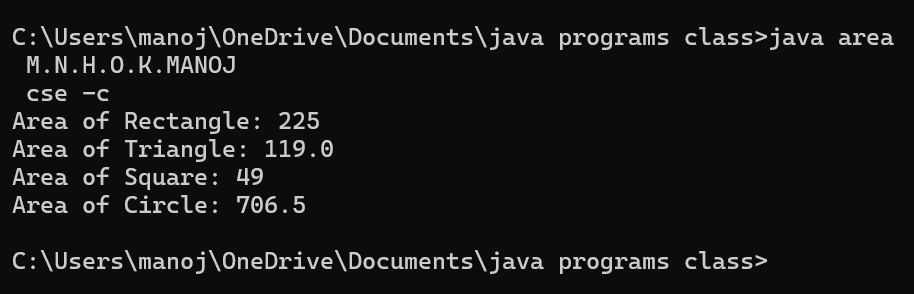
shape.calculateArea(7.0, 34.0);

shape.calculateArea(7);

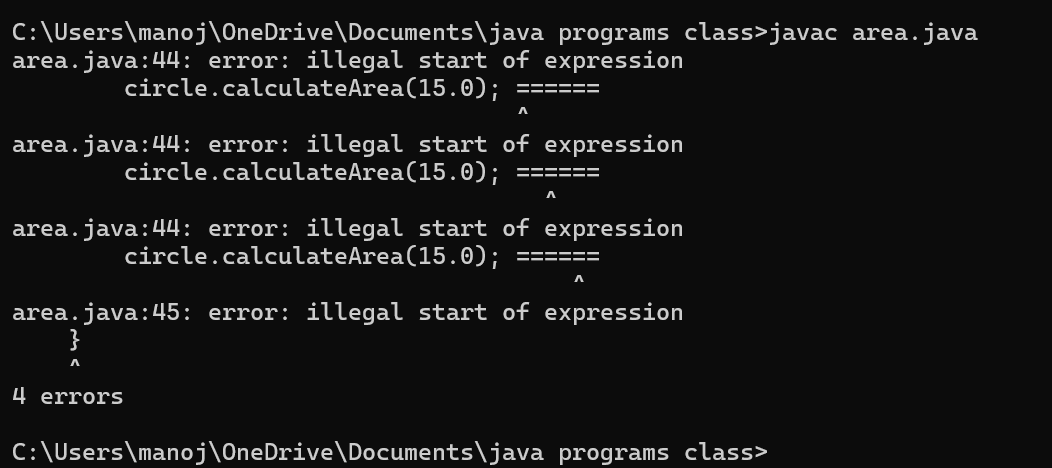
Circle circle = new Circle();

circle.calculateArea(15.0); }}

**Out put:**

1.**positive case :** 

1. **Negative case :**



**Error table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **rectification** |
| **1.** | **Logical error** | **Incorrect formula** | **Correcting formula** |
| **2.** | **Runtime errror** | **Incorrect path** | **Using correct path** |
| **3.** | **Syntax error** | **Incorrect symbols** | **Using correct symbols** |

**IMPORTANT POINTS** :

1. **Method Overloading** allows the Shape class to calculate areas for different shapes (rectangle, triangle, square) using multiple calculateArea() methods with different parameters.
2. **Method Overriding** in the Circle class provides a custom implementation of calculateArea() for calculating the area of a circle.
3. **Polymorphism** enables objects of different types (e.g., Shape, Circle) to call the appropriate version of calculateArea() based on the object type and input parameters.

4**. Create a calculator class with overloaded methods to perform addition**

**1.Add two integers 2.Add two doubles 3.Add three integers**

**CLASS DIAGRAM:**

|  |
| --- |
| **CLASS CALCULATORMAN** |
| **+add(int, int): int**  **+add(double, double): double + +add(int, int, int): int** |

|  |
| --- |
| **HOSTEL** |
| **+main(String[]): void** |

**CODE:**

class calculatorman {

public int add(int a, int b){

System.out.println(" addition of in type: ");

return a+b; }

public double add(double a, double b)

System.out.println(" addition of double type: ");

return a+b; }

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

System.out.println(" addition of int type 3 variables: ");

return a+b+c; }

class HOSTEL {

public static void main(String args[]){

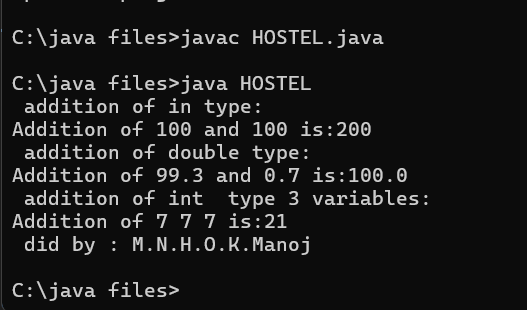
calculatorman kaala = new calculatorman();

System.out.println("Addition of 100 and 100 is:"+kaala.add(100,100));

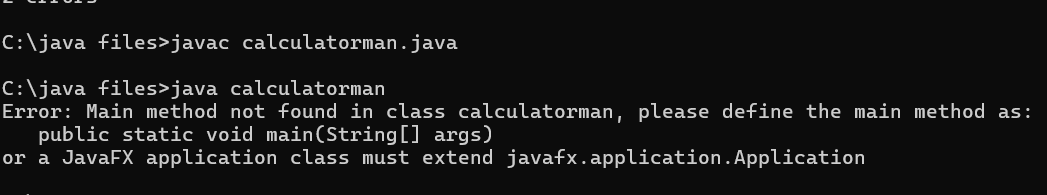
System.out.println("Addition of 99.3 and 0.7 is:"+kaala.add(99.3 , 0.7));

System.out.println("Addition of 7 7 7 is:"+kaala.add(7,7,7));

System.out.println(" did by : M.N.H.O.K.Manoj"); }}

**OUTPUT:**

**1.POSITIVE CASE :**



**2.NEGATIVE CASE:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **rectification** |
| **1.** | **Logical error** | **Incorrect formula** | **Correcting formula** |
| **2.** | **Runtime errror** | **Incorrect path** | **Using correct path** |
| **3.** | **Syntax error** | **Incorrect symbols** | **Using correct symbols** |

**ERROR TABLE:**

**IMPORTANT POINTS :**

1. This program demonstrates **method overloading**, where multiple add() methods have the same name but different parameter types or counts.

2. Each method performs addition and prints the type of addition being done (int, double, or three integers).

3. In the main() method, the correct overloaded method is called based on the arguments passed, showcasing polymorphism.

**Week-7**

**Aim: Write a java program to create an abstract class Animal with an abstract**

**Method called sound. Create subclasses lion and tiger that extends the animal**

**Class and implements the sound() method to make a specific sound for each animal**

**CLASS DIAGRAM:**

|  |
| --- |
| **< ABSTACT CLASS >**  **Animal** |
| **+ sound() : void** |

|  |
| --- |
| **Tiger** |
| **+sound() : void** |

|  |
| --- |
| **Lion** |
| **+sound() : void** |

PROGRAM :

abstract class animal{

abstract void sound(); {

class Lion extends animal{

void sound(){

System.out.println("Lions Roars in the jungle"); }}

class Tiger extends animal

{ void sound() {

System.out.println("Tiger can make sounds"); }}

class Abstraction {

public static void main(String args[]){

System.out.println("M.N.H.O.K Manoj");

System.out.println("24214");

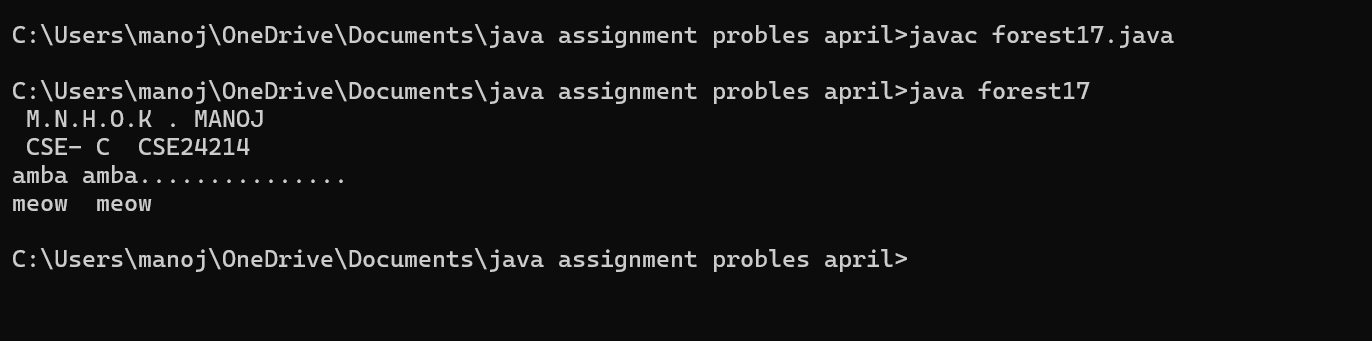
System.out.println("---------");

Lion obj1 = new Lion();

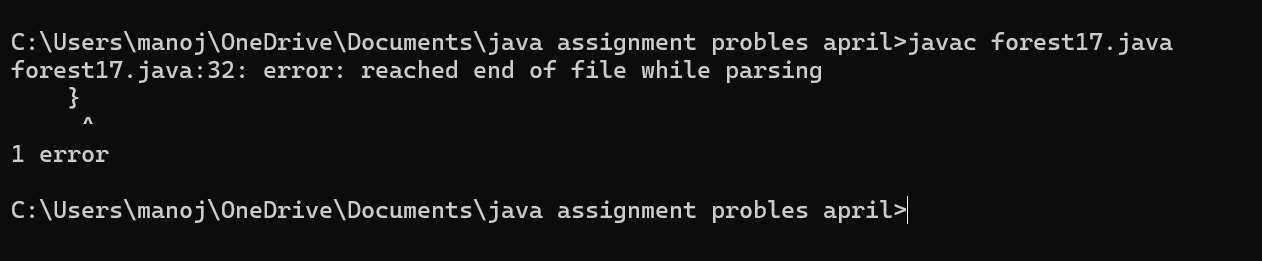
obj1.sound();

Tiger obj2 = new Tiger();

obj2.sound(); }}

**OUTPUT : POSITIVE CASE :**

**NEGATIVE CASE :**



**Error Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Cause** | **Rectification** |
| **1** | **Static reference to non-static method** | **Calling non-static method from static context** | **Create object correctly** |
| **2** | **Instantiation of abstract class(OOP error)** | **Trying to create an object of an abstract class** | **Use subclass or interface implementation** |
| **3** | **Syntax error** | **Missing parenthesis** | **Adding parenthesis** |

**Important points:**  The abstract class animal defines the sound() method that must be implemented by subclasses.

Lion and Tiger extend animal and provide their own versions of the sound() method.

The main() method demonstrates runtime polymorphism by calling sound() on each object

2.**AIM:** Write a java program to create an abstract class shape3D with

abstract methods calculateVolume() and calculateSurfaceArea().Create subclasses sphere and cube that extend the shape3D class and implement

the respective methods to calculate the volume and surface area of each shape

|  |
| --- |
| **<abstract>>**  **Shape3D** |
| **+calculateVolume(): double +calculateSurfaceArea(): double** |

**CLASS DIAGRAM :**

|  |
| --- |
| **CUBE** |
| **- a: int** |
| **+calculateVolume()**  **+calculateSurfaceArea()** |

|  |
| --- |
| **Sphere** |
| **- radius: int** |
| **+calculateVolume() +calculateSurfaceArea()** |

**CODE:**

abstract class Shape3D{

abstract double calculateVolume();

abstract double calculateSurfaceArea(); }

lass Sphere extends Shape3D {

private int radius;

Sphere(int radius)

this.radius = radius; }

public double calculateVolume(){

return 1.33\*2.14\*radius\*radius\*radius; }

public double calculateSurfaceArea() {

return 4\*3.14\*radius\*radius; }}

class Cube extends Shape3D {

private int a;

Cube(int a){

this.a = a; }

public double calculateVolume() {

return a\*a\*a;}

public double calculateSurfaceArea() {

return 6\*a\*a; }}

class Mainn1{

public static void main(String args[]){

System.out.println("M.N.H.O.K Manoj");

System.out.println("CSE24214");

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

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

Sphere obj1 = new Sphere(4);

Cube obj2 = new Cube(3);

System.out.println(obj1.calculateVolume());

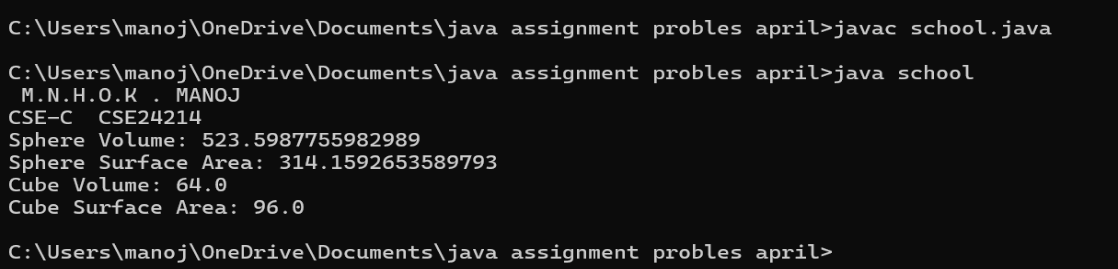
System.out.println(obj1.calculateSurfaceArea());

System.out.println(obj2.calculateVolume());

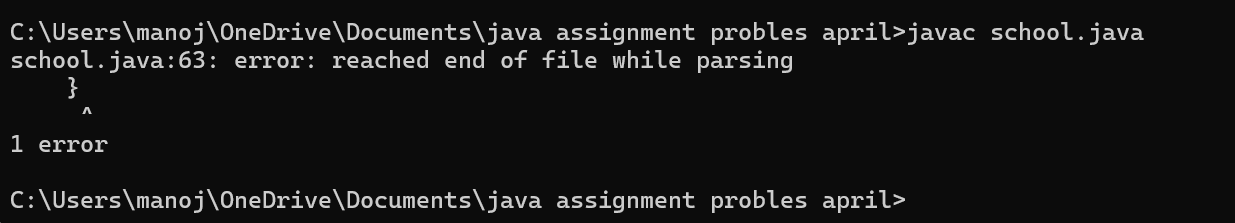
System.out.println(obj2.calculateSurfaceArea()); }}

OUTPUT :

POSITIVE CASE :



Negative case:



**Error Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **Error Type** | **Cause** | **Rectification** |
| **1** | **Return type missing** | **Method doesn’t specify return type** | **Adding the correct return type** |
| **2** | **Instantiation of abstract class** | **Trying to create an object of abstract class without any subclass or interface** | **Use subclass or interface implementation** |
| **3** | **Static reference to Non-static Method** | **Calling non-static method from static context** | **Create object or make method static** |

**IMPORTANT POINT :**

**Shape3D is an abstract class with abstract methods to calculate volume and surface area.**

**Sphere and Cube extend Shape3D and provide specific implementations for volume and surface area.**

**The main method creates objects and prints the calculated values — demonstrating polymorphism and abstraction.**

**3.AIM:**

**Write a java program using an abstract class to define a method for pattern printing. Create an abstract class named PatternPrinter with an**

**Abstract method printpattern(int n) and a concrete method to display the pattern title.**

**Implement two subclasses**

**1.star pattern- prints a right angled triangle of stars(\*)**

**2.Number pattern- prints a right angled triangle of increasing numbers.**

**In the main() method create objects of both subclasses and print the patterns**

**For a given number of rows.**

**CLASS DIAGRAM :**

|  |
| --- |
| **<<abstract>>**  **PatternPrinter** |
| **+printpattern(int): void**  **+display(String): void** |

|  |
| --- |
| **starpattern** |
| **+printpattern(int)** |

|  |
| --- |
| **numberpattern** |
| **+printpattern(int)** |

**PROGRAM :**

abstract class PatternPrinter {

abstract void printpattern(int n);

public void display(String title) {

System.out.println("\n=== "+ title+"==="); }}

class starpattern extends PatternPrinter{

void printpattern(int n) {

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

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

system.out.print("\* ");}

System.out.println(); }}}

class numberpattern extends PatternPrinter{

void printpattern(int n){

int num=1;

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

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

System.out.print(j+" ");

num++; }

System.out.println(); }}}

class Main2 {

public static void main(String args[]) {

System.out.println("M.N.H.O.K Manoj");

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

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

int rows=5;

PatternPrinter star = new starpattern();

star.display("Number pattern");

star.printpattern(rows);

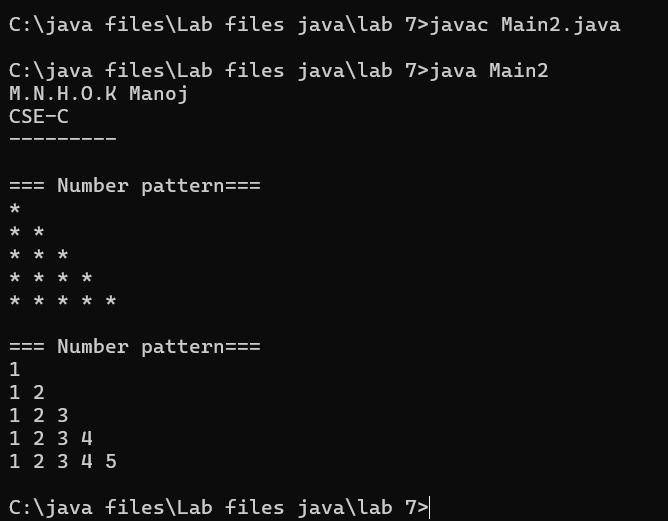
PatternPrinter number = new numberpattern();

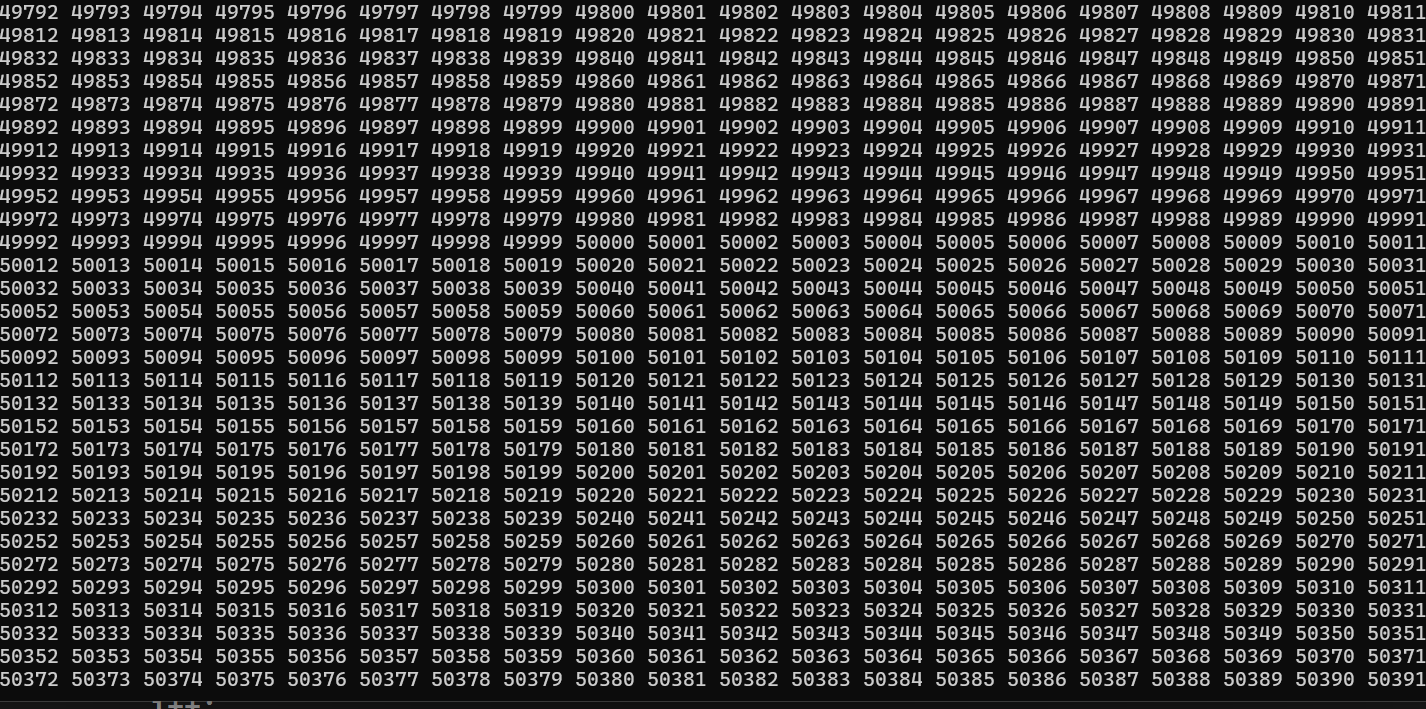
number.display("Number pattern");

number.printpattern(rows);}}

OUTPUT :

POSITIVE CASE:



NEGATIVE CASE :  


**ERROR TABLE**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error Type** | **Cause** | **Rectification** |
| **1** | **Logical error** | **Incorrect use of operator** | **Using correct operator** |
| **2** | **Run time error** | **Incorrect path** | **Selected correct path** |
| **3** | **Syntax error** | **Missing semicolon** | **Semicolon added** |

**IMPORTANT POINTS:**

**Abstraction is used to define a common method printpattern() in the abstract class PatternPrinter, allowing different patterns to be printed through subclasses.**

**The starpattern and numberpattern classes implement their own version of printpattern(), demonstrating polymorphism.**

**The display() method in the abstract class allows for a consistent header before printing any pattern, making the output structured and easy to manage.**

**WEEK- 8**

**1.AIM: Write a Java program to create an interface Shape with the getPerimeter method. Create three classes Rectangle, Circle, and Triangle that implement the Shape interface. Implement the getPerimeter() method for each of the three classes.**

**CLASS DIAGRAM :**

|  |
| --- |
| **<<interface>>**  **Shape** |
| **+getPerimeter()** |

|  |
| --- |
| **Rectangle** |
| **-l : int**  **- b: int** |
| **+getPerimeter()** |

|  |
| --- |
| **Triangle** |
| **- s1, s2, s3** |
| **+getPerimeter()** |

|  |
| --- |
| **Circle** |
| **- r: int** |
| **+getPerimeter()** |

**PROGRAM :**

interface Shape {

void getPerimeter(); }

class Rectangle implements Shape {

int l, b; public Rectangle(int l, int b) {

this.l = l;

this.b = b; }

public void getPerimeter() {

int perimeter = 2 \* (l + b);

System.out.println("Rectangle Perimeter: " + perimeter);}}

class Triangle implements Shape {

int s1, s2, s3;

public Triangle(int s1, int s2, int s3) {

this.s1 = s1;

this.s2 = s2;

this.s3 = s3; }

public void getPerimeter() {

int perimeter = s1 + s2 + s3;

System.out.println("Triangle Perimeter: " + perimeter); }}

class Circle implements Shape {

int r;

public Circle(int r) {

this.r = r; }

public void getPerimeter() {

double perimeter = 2 \* Math.PI \* r;

System.out.println("Circle Perimeter: " + perimeter);}}

class perimeter {

public static void main(String[] args) {

System.out.println("Done by M.N.H.O.K MANOJ FROM CSE C CSE24214");

Shape obj1 = new Rectangle(4, 6);

Shape obj2 = new Triangle(4, 6, 5);

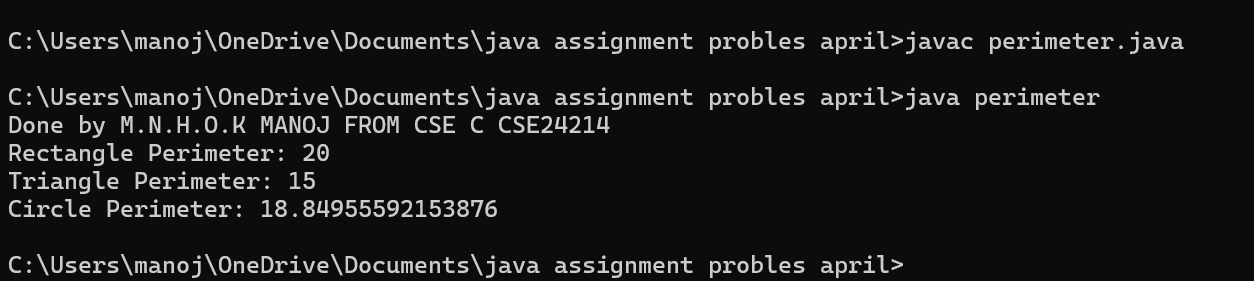
Shape obj3 = new Circle(3);

obj1.getPerimeter();

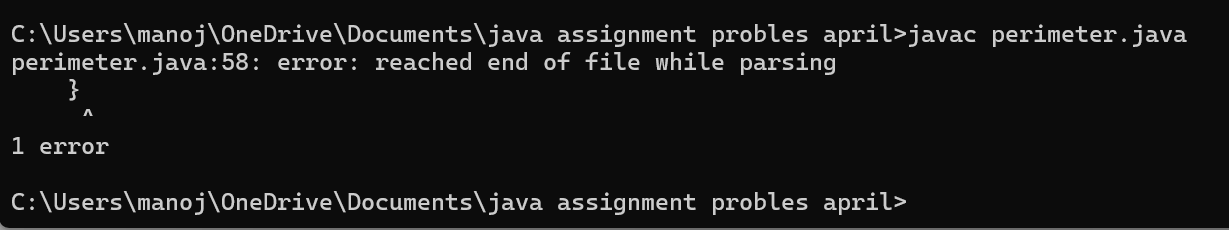
obj2.getPerimeter();

obj3.getPerimeter();} }

**Output: Positive case**



**Negative Case:**



**Error Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error Type** | **Cause** | **Rectification** |
| **1** | **Compilation error** | **Return type missing** | **Add correct return type** |
| **2** | **Access modifier issues(OOP error)** | **Trying to access private members** | **Change the modifier or add getter/setter** |
| **3** | **Syntax Error** | **Mismatched brackets** | **Close properly all brackets** |

**Important points:**

1. Interface Implementation: The Shape interface is implemented by three classes: Rectangle, Triangle, and Circle, each defining the getPerimeter() method for their respective shapes.
2. Polymorphism: The Shape interface allows you to treat different objects (Rectangle, Triangle, Circle) as instances of Shape, demonstrating polymorphism when calling the getPerimeter() method.
3. Flexibility and Extensibility: This design makes it easy to add more shapes (like Square, Ellipse) in the future by simply implementing the Shape interface without modifying existing code.

2.**Aim: 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 implement the Playable interface and override**

**the play() method to play the respective sports.**

**CLASS DIAGRAM :**

|  |
| --- |
| **<<interface>>**  **playable** |
| **+play(): void** |

|  |
| --- |
| **football** |
| **+play()** |

|  |
| --- |
| **basketball** |
| **+play()** |

|  |
| --- |
| **Volleyball** |
| **+play()** |

**PROGRAM :**

interface playable {

abstract void play() ;}

class football implements playable {

public void play() {

System.out.println( " FOOT BALL " ) ;}}

class volleyball implements playable {

public void play() {

System.out.println( " volley ball " ) ; }}

class basketball implements playable {

public void play() {

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

class game {

public static void main( String[] args ) {

System.out.println("M.N.H.O.K.MANOJ CSE-C CSE24214 ");

football obj1 = new football();

volleyball obj2 = new volleyball();

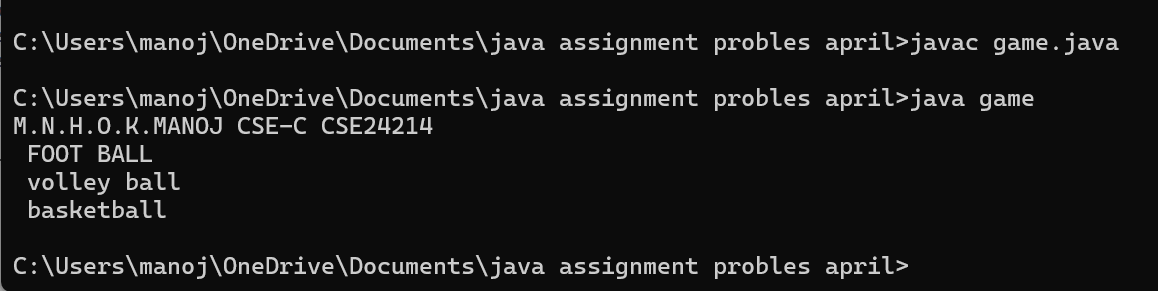
basketball obj3 = new basketball();

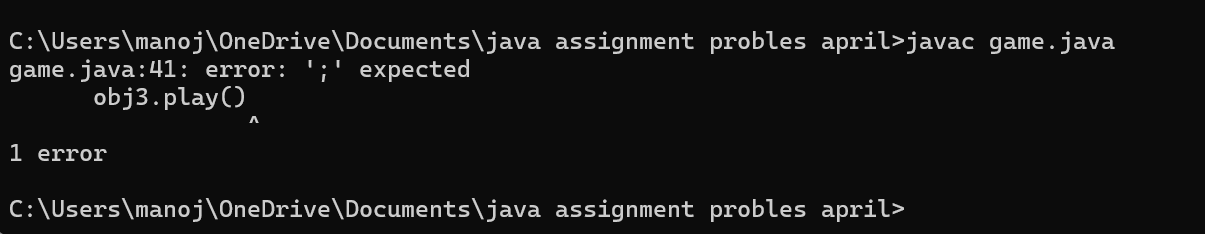
obj1.play();

obj2.play();

obj3.play() ; }}

**Output**

**Positive case:  
**

**Negative Case:**

**Error Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error Type** | **Cause** | **Rectification** |
| **1** | **File not found exception** | **Wrong selection of path** | **Select path correctly** |
| **2** | **Instantiation of abstract class** | **Trying to create an object of abstract class without any subclass or interface** | **Use subclass or interface implementation** |
| **3** | **Static reference to Non-static Method** | **Calling non-static method from static context** | **Create object or make method static** |

**IMPORTANT POINTS :**

**Interface Implementation:  
The playable interface defines the play() method, which is implemented by football, volleyball, and basketball classes, each representing a different sport.**

**Polymorphism in Action:  
The objects of football, volleyball, and basketball are all treated as playable types, allowing the play() method to be called polymorphically.**

**Flexible Design:  
This design allows easy addition of new sports (like cricket or baseball) without modifying the game class, supporting extensibility and maintainability.**