**23CSE111**

OOPS

**LAB MANUAL**

A logo with pink letters

Description automatically generated

**Department of CSE**

**Amrita School of Engineering**

**Amrita Vishwa Vidyapeetham, Amaravati Campus**

**Verified By Name: E. Hima Teja Goud**

**Roll No: 24048**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.NO | Programs | Date | Pg:No | Signature |
| WEEK-1 | 1. Download and Install Java Software. 2. Write a java program to print message “Welcome to java programming”. 3. Write a java program that prints name,roll number,section of a student. |  |  |  |
| WEEK-2 | a) Write a java program to calculate the area of the rectangle.  b) Write a java program to convert temperature from Celsius to  Fahrenheit and viceversa.  c) Write a java program to calculate the simple intrest.  d) Write a java program to find largest of three numbers using  ternary operator.  e) Write a java program to find factorial of number |  |  |  |
| WEEK-3 | 1. Write a java program with the following instructions.  a) Create a class with name car.  b) Create four attributes named car\_colour,car\_brand,fuel  type,top\_speed.  c) Create three method named  “Start\_Racing”,”End\_Race”.{ }  d) Create three objects named Car1,Car2,Car3.  e) Create a constructor which should print “Welcome to car  garage”.  2. Write a class by writing java program named Bank Account  with two methods “deposits and withdraw”.  a) In deposit method whenever an amount is deposited it  has to be updated with current amount (logic C.A+D.A).  b) With draw amount whenever an amount is being  withdraw it has to be less than the current amount less  than the amount else print “Insufficient funds”. |  |  |  |
| WEEK-4 |  |  |  |  |
| WEEk-5 |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Week-1

* Program : 1
* Aim : Download and Install Java Software.
* Step 1 : Visit chrome and search “ java download”.And select Oracle website.

A screenshot of a computer

Description automatically generated

* Step 2 : Now open Oracle website scroll down and now select “JDK 21” for

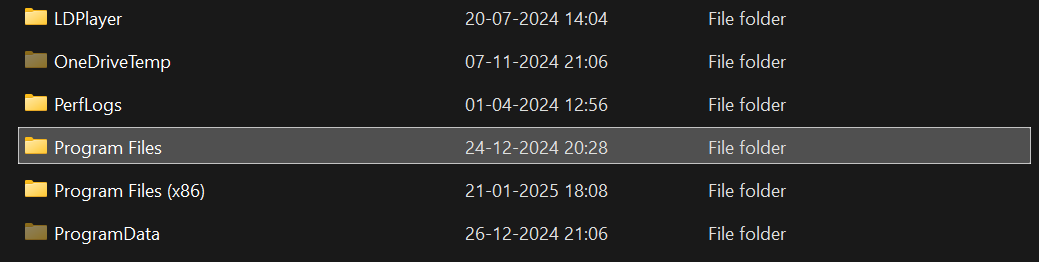
Windows and select “X64 installer” and download it.

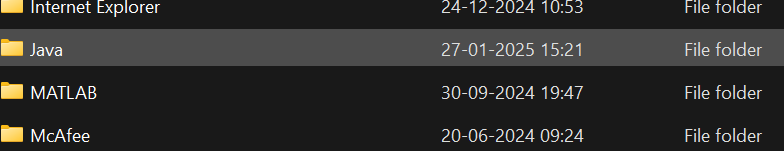
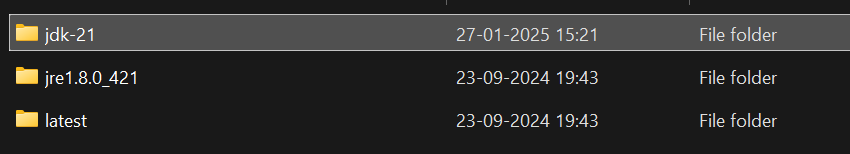
A screenshot of a computer

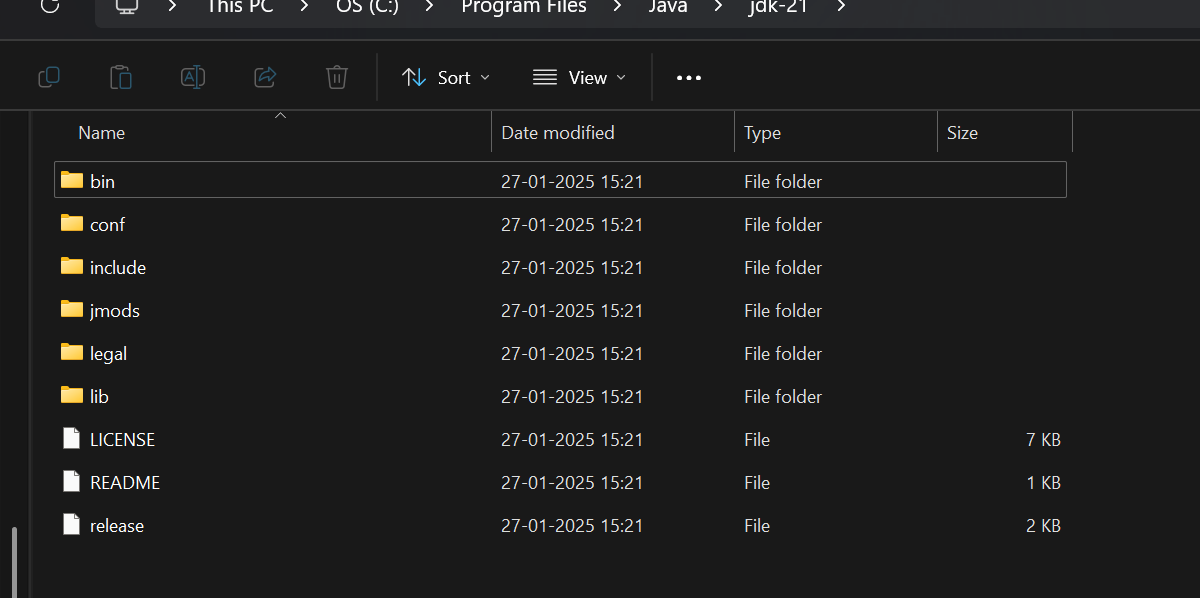
Description automatically generated

* Step 3 : After downloading open “this pc” in our laptop and open “program

files”,open “java”,open “JDK 21”



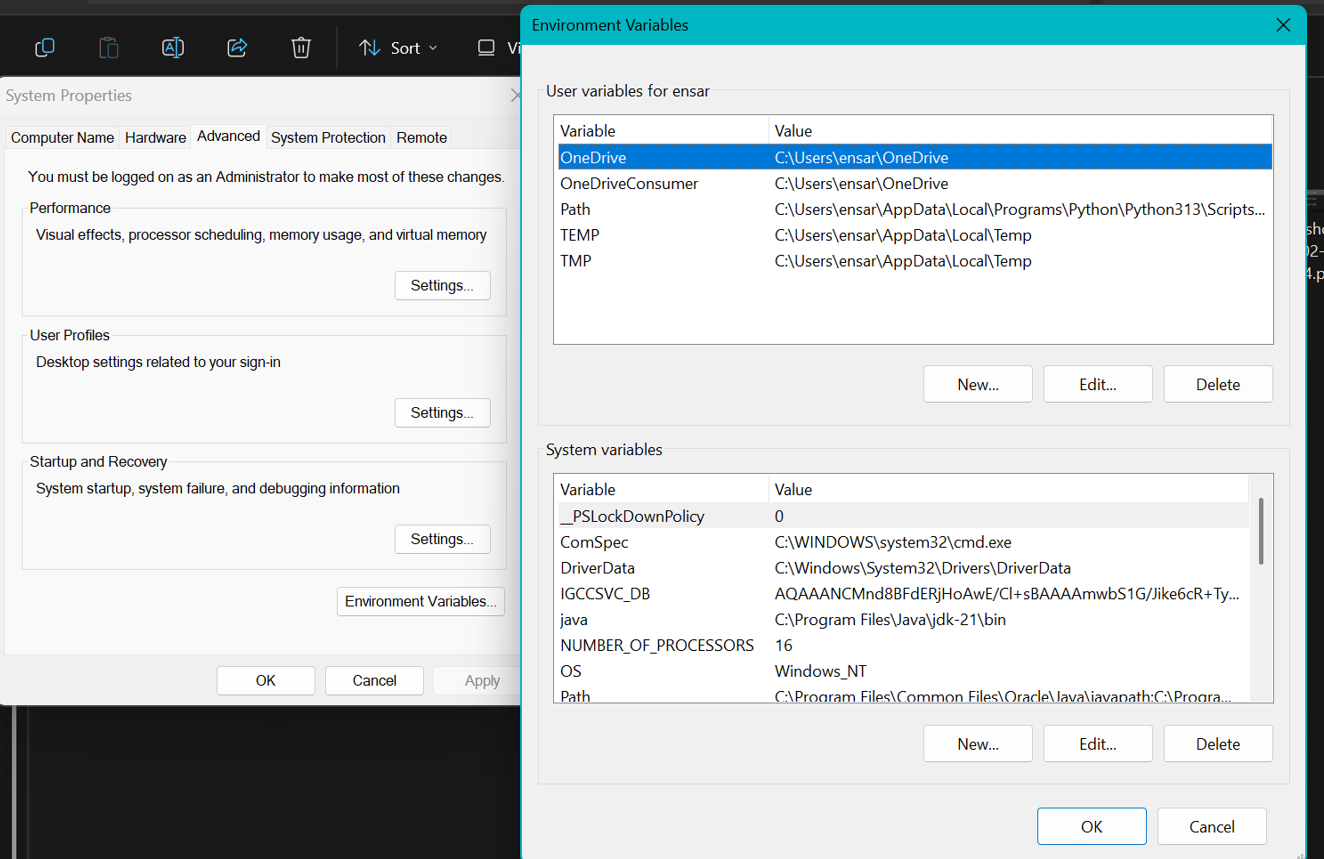
 



* Step 4 : In the task bar search and open “environment variables of system”,after opening environment variables, go to the system variables and see for java if there leave it. Or click path and add “JAVA” in ‘variable name’ and copy link in ‘variable value’

A screenshot of a computer

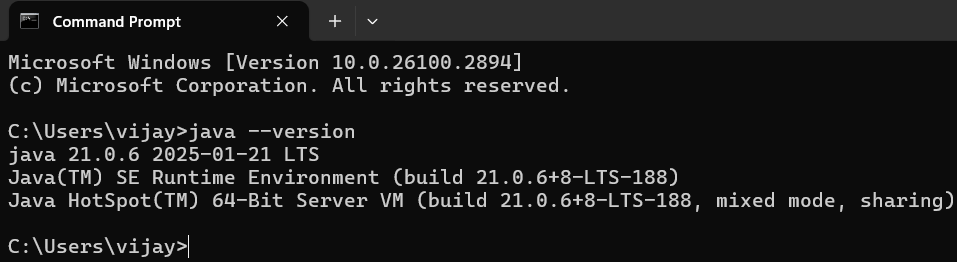
Description automatically generated



* Step 5 : Verifying Installation of Java. Again open task bar and search “cmd”,

open it ant type “java –version” and press enter. It will show the

version of installation of java.



Successfully Java is installed and it will show the version otherwise it will show error and command is not recognized.

* Program : 2

Q) Write a java program to print the message “welcome to java program”.

class Main{

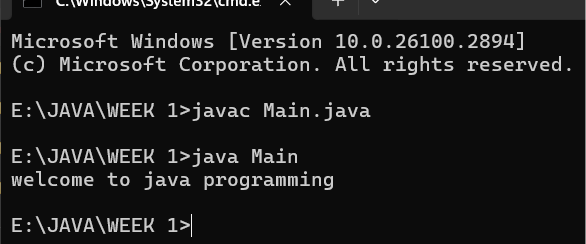
public static void main(String[] args){

System.out.println("welcome to java programming");

}

}

OUTPUT



* Program : 3

Q) Write a java program that prints name,roll number,section of a student.

public class my\_profile{

public static void main(String[] arg){

System.out.println("name:E.Hima Teja Goud");

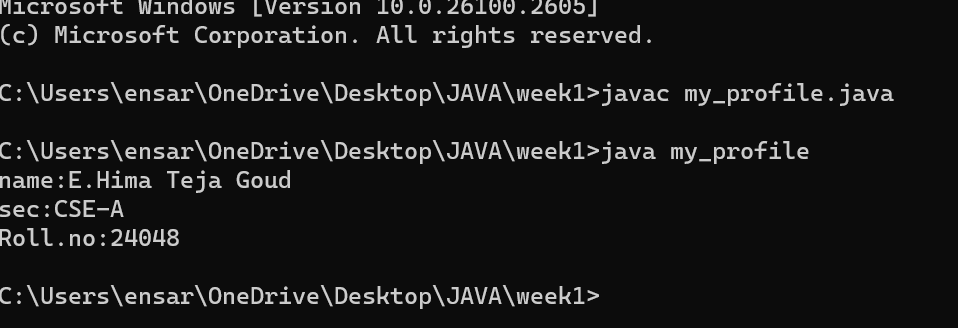
System.out.println("sec:CSE-A");

System.out.println("Roll.no:24048");

}

}

OUTPUT



Week-2

* Program : 1

import java.util.Scanner;

class rectangle{

public static void main(String[]args){

Scanner input=new Scanner(System.in);

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

int len=input.nextInt();

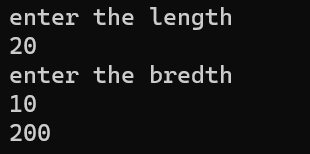
System.out.println("enter the bredth");

int bred=input.nextInt();

int area=len\*bred;

System.out.println(area);

}}

OUTPUT: 

|  |  |  |
| --- | --- | --- |
| s.no | EXPECTED ERROR | REASON |
| 1. | ; | **; is expected at end** |
| 2. | AREA | Declaration of int type variable |

* Program : 2

import java.util.Scanner;

class tem{

public static void main(String[]args){

Scanner input =new Scanner(System.in);

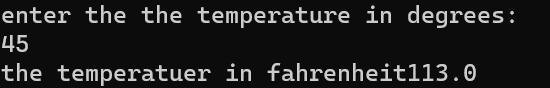
System.out.println("enter the the temperature in degrees:");

double deg=input.nextDouble();

System.out.println("the temperatuer in fahrenheit"+((deg\*9/5)+32));

}

}

OUTPUT: 

ERRORS:

|  |  |  |
| --- | --- | --- |
| S.No | EXPECTED ERROR | REASON |
| 1. | ; | ;expected at end |
| 2. | Input().close | The input is expected to closed |

* Program : 3

import java.util.Scanner;

class simpleintrest{

public static void main(String[]args){

Scanner input=new Scanner(System.in);

System.out.println("enter the p value");

int p=input.nextInt();

System.out.println("enter the t value");

int t=input.nextInt();

System.out.println("enter the r value");

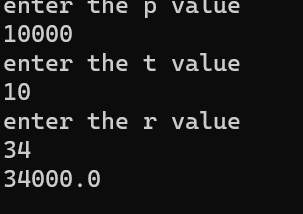
int r=input.nextInt();

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

System.out.println(si);

}

}

OUTPUT: 

|  |  |  |
| --- | --- | --- |
| S.No | EXPECTED ERROR | REASON |
| 1. | ; | ; is expected at end |
| 2. | Int t | **Without declaring t the compiler cannot execute the program.** |

* Program : 4

import java.util.Scanner;

class largest{

public static void main(String[]args){

Scanner input=new Scanner(System.in);

System.out.println("enter value of A");

int a=input.nextInt();

System.out.println("enter value of B");

int b=input.nextInt();

System.out.println("enter value of C");

int c=input.nextInt();

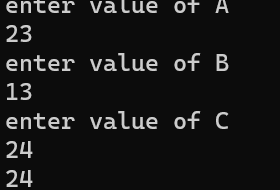
int large=(a>b)?((a>c)?a:c):((b>c)?b:c);

System.out.println(large);

}

}

OUTPUT:



ERRORS:

|  |  |  |
| --- | --- | --- |
| S.No | EXPECTED ERROR | REASON |
| 1. | ? | Checks the condition |
| 2. | : | Comparing between two variables |

* Program : 5

import java.util.Scanner;

class factorial{

public static void main(String[]args){

Scanner input=new Scanner(System.in);

System.out.println("enter the number to find its factorial");

int n=input.nextInt();

int sum=1;

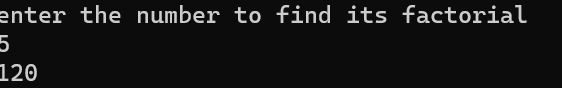
for(int i=1;i<=n;i++){

sum=sum\*i;}

System.out.println(sum);

}

}

OUTPUT: 

ERRORS:

|  |  |  |
| --- | --- | --- |
| S.No | EXPECTEED ERRORS | REASON |
| 1. | } | To close for loop |
| 2. | ; | ; expected |

WEEK-3

* Program : 1

Q) **Write a java program with the following instructions**.

1. Create a class with name car.
2. Create four attributes named car\_colour,car\_brand,fuel\_type,top\_speed.
3. Create three method named “Start\_Racing”,”End\_Race”.{ }
4. Create three objects named Car1,Car2,Car3.
5. Create a constructor which should print “Welcome to Garage”.

Class Diagram:

|  |
| --- |
| **Car** |
| * carColor: String |
| * carBrand: String |
| * fuelType: String |
| * topSpeed: int |
| + Car(String,String,String,int) |
| + startRacing() |
| + endRace() |

// Car.java

public class Car {

// Attributes

private String carColour;

private String carBrand;

private String fuelType;

private int topSpeed;

// Constructor

public Car(String carColour, String carBrand, String fuelType, int topSpeed) {

this.carColour = carColour;

this.carBrand = carBrand;

this.fuelType = fuelType;

this.topSpeed = topSpeed;

System.out.println("Welcome to car garage");

}

// Method to start racing

public void startRacing() {

System.out.println(carBrand + " (" + carColour + ") is starting the race with a top speed of " + topSpeed + " km/h and runs on " + fuelType + "!");

}

// Method to end race

public void endRace() {

System.out.println(carBrand + " (" + carColour + ") has finished the race!");

}

// Main method to create objects and demonstrate functionality

public static void main(String[] args) {

// Creating three objects

Car car1 = new Car("Red", "Ferrari", "Petrol", 200);

Car car2 = new Car("Blue", "Tesla", "Electric", 250);

Car car3 = new Car("Black", "BMW", "Diesel", 220);

// Starting and ending races

car1.startRacing();

car1.endRace();

car2.startRacing();

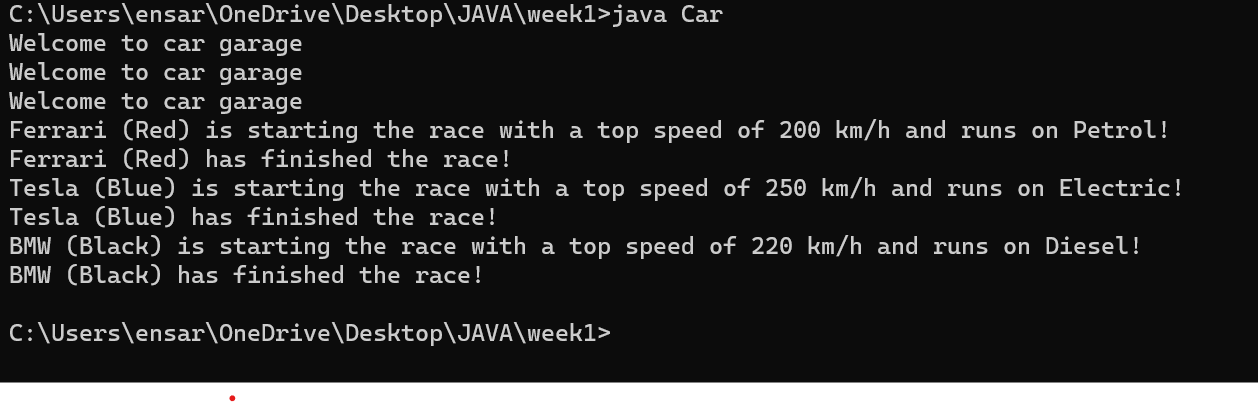
car2.endRace();

car3.startRacing();

car3.endRace();

}

}



|  |  |  |
| --- | --- | --- |
| s.no | ERRORS | RECTIFICATION |
| 1. | } | Close the loop |
| 2. | System.out.println | If we place the print statement inside the for loop it will print the each i value everytime but to print only the final value we must place it outside the for loop. |

* Program : 2

Q ) Write a class by writing java program named Bank Account

with two methods “deposits and withdraw”.

a) In deposit method whenever an amount is deposited it

has to be updated with current amount (logic C.A+D.A).

b) With draw amount whenever an amount is being

withdraw it has to be less than the current amount less

than the amount else print “Insufficient funds”.

Class Diagram:

|  |
| --- |
| Bank Account |
| * currentAmount: double |
| + BankAccount(initialAmount:double) |
| + deposit(amount: double):void |
| + withdraw(amount: double):void |
| + getCurrentAmount():double |

import java.util.Scanner;

class BankAccount {

String name;

int accountNumber;

int currentBalance;

// Constructor to initialize the bank account

BankAccount(String name, int accountNumber, int currentBalance) {

this.name = name;

this.accountNumber = accountNumber;

this.currentBalance = currentBalance;

System.out.println("Customer Details: " + name + ", Account Number: " + accountNumber + ", Current Balance: " + currentBalance);

}

// Method to withdraw an amount

public void withdraw(int withdrawAmount) {

if (withdrawAmount <= currentBalance) {

currentBalance -= withdrawAmount;

System.out.println("Withdrawn: " + withdrawAmount);

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

} else {

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

}

}

// Method to deposit an amount

public int deposit(int depositAmount) {

currentBalance += depositAmount;

System.out.println("Deposited: " + depositAmount);

return currentBalance;

}

// Main method to run the program

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input for account details

System.out.print("Enter your name: ");

String name = scanner.nextLine();

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

int accountNumber = scanner.nextInt();

System.out.print("Enter your initial balance: ");

int initialBalance = scanner.nextInt();

// Create a new bank account

BankAccount account = new BankAccount(name, accountNumber, initialBalance);

// Input for withdrawal and deposit

System.out.print("Enter amount to withdraw: ");

int withdrawAmount = scanner.nextInt();

account.withdraw(withdrawAmount);

System.out.print("Enter amount to deposit: ");

int depositAmount = scanner.nextInt();

account.deposit(depositAmount);

// Final balance

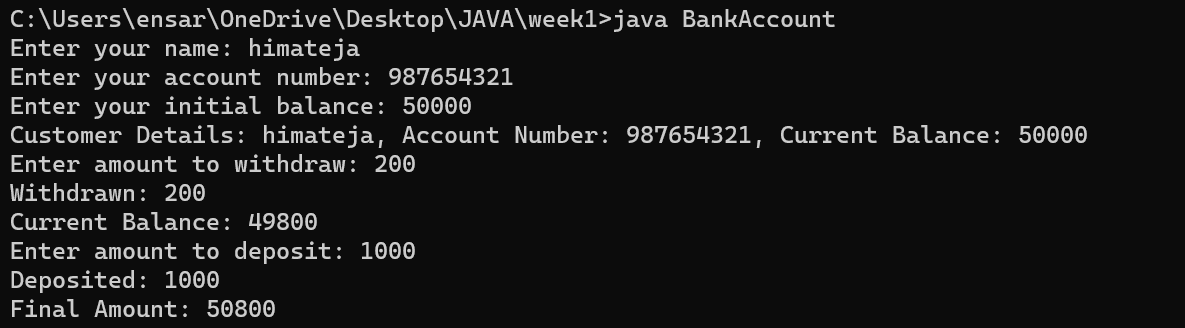
System.out.println("Final Amount: " + account.currentBalance);

// Close the scanner

scanner.close();

}

}



|  |  |  |
| --- | --- | --- |
| **S.NO** | **Errors** | **Rectification** |
| 1 | ; | ; is expected at end |
| 2 | Int t | Without declaring the compiler cannot execute the  program. |

**WEEK – 4**

1. **Write a java program with class named “Book”. The class should contain various attributes such as**

**“Title of the book , author , year of publication “. It should also contain a constructor with parameters**

**which initializes “ Title of the book, author, year of publication”. Create a method which displays the**

**details of the book. i.e. “ Title of the book, author and year of publication”. Display the details of two**

**books by creating two objects.**

**Class Diagram:**

|  |
| --- |
| **Book** |
| **- title: String**  **- author: String**  **- yearOfPublication: int** |
| **+ Book(title: String, author: String,**  **yearOfPublication: int)**  **+ displayDetails(): void** |

**Code :**

**class Book {**

**// beginning of the class Book**

**public String title; // Changed Title to title for consistency**

**private String author;**

**public int yearOfPublication;**

**// beginning of constructor**

**Book(String title, String author, int yearOfPublication) {**

**this.title = title; // Changed Title to title for consistency**

**this.author = author;**

**this.yearOfPublication = yearOfPublication;**

**}**

**// constructor ends here**

**// method display starts here**

**public void display() {**

**System.out.println("Title of the book is: " + title +**

**", The name of the author is: " + author +**

**", The year of publication is: " + yearOfPublication);**

**}**

**// method display ends here**

**// creating objects**

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

**Book book1 = new Book("solo leveling", "himateja", 2020);**

**Book book2 = new Book("naruto", "tej", 2010);**

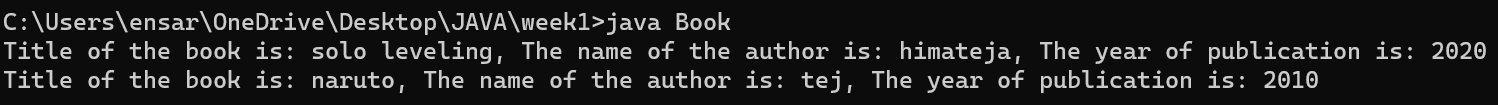
**book1.display();**

**book2.display();**

**}**

**}**

**// class ends here**

Output: 

**Errors :**

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

b). **To create a java program with class named Myclass with a static variable “Count” of “int type”,**

**Initialized to 0 and a constant variable “pi” of type double , initialized to 3.1415 as attributes of that class**

**Now, define a constructor for “Myclass” that increments the “Count” variable each that an object of**

**Myclass is created. Finally , print the final values of “Count” and “pi” variables .**

**Class Diagram:**

|  |
| --- |
| **MyClass** |
| **- Count: int**  **+ pi: double** |
| **+ MyClass()**  **+ getCount(): int** |

**Code:**

class Myclass{

// class starts here

static int Count = 0;

final double pi = 3.1415;

// the constructor starts here

Myclass(){

Count++;

}

// the constructor ends here

public static void main(String[] args){

Myclass c1 = new Myclass();

Myclass c2 = new Myclass();

System.out.println("Count : " + c1.Count);

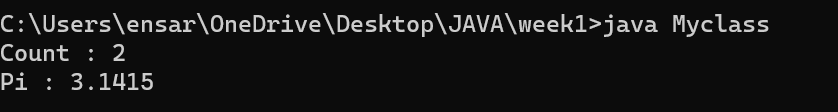
System.out.println("Pi : " + c1.pi);

}

}

// class ends here

Output:



**Errors :**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Expected Error** | **Reason** |
| **1** | **.variable** | **We must mention variable name to call the variable** |
| **2** | **static** | **Static variables contain only one value** |

WEEK-5

* Program : 1

Q)  **Create a calculator using the operations including addition using subtraction**

**multiplication and division using multilateral inheritance and display thr desired output.**

Class Diagram:

|  |
| --- |
| **Basic Operations** |
| + add (a,b)  +subtract (a,b) |

|  |
| --- |
| **Multiplication** |
| +Multiply (a,b) |

|  |
| --- |
| **Division** |
| + Divide (a,b) |

|  |  |
| --- | --- |
| **Subtraction** | |
| + subtraction(a,b) | |
|  | |  | |

|  |
| --- |
| **Calculator** |
| +calculate (op,a,b) |

CODE:

class bcalc {

int a, b;

int sum, diff;

bcalc(int a, int b) {

this.a = a;

this.b = b;

}

public void add()

{ diff = a - b;

sum = a + b;

System.out.println("Difference: " + diff);

System.out.println("Sum: " + sum);

}

}

class acalc extends bcalc {

int mul; acalc(int a, int b) {

super(a, b);

}

public void mult() {

mul = a \* b;

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

}

}

class aacalc extends acalc {

float div;

aacalc(int a, int b) {

super(a, b);

}

public void divi()

{

if (b != 0) { // Check to avoid division by zero

div = (float) a / b;

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

}

else {

System.out.println("Division by zero error!");

}

}

}

class ocalc {

public static void main(String[] args) {

aacalc c = new aacalc(10, 2);

c.divi();

c.mult();

c.add();

}

}

Output:



|  |  |  |
| --- | --- | --- |
| **S.NO** | **Errors** | **Rectification** |
| 1 | .variable | We must mention variable name to call the variable. |
| 2 | static | Static variables contain only one value. |

* Program : 2

Q)  **A Vechile rental company wants to develop a system ,that maintains information about**

**different types of vehicles available for rent.The company rents out cars and bikes and**

**they a need a program to store details about each vehicle such as brand and speed .**

* **Cars should have an additional properties .**
* **“Number of doors “ seating capacity.**
* **Bikes should have a property indicating whether they have gears are not ?**
* **The system should also include a fuction to display details about each vehicle and indicate when a vechicle is starting .**
* **If the company describes to add a new type of vechile ‘truck’ how would you modify above program.**
* **Truck should include an addition property capacity ‘in tons’.**
* **Create a show truck details method to display the trucks capacity.**
* **Write a constructor for truck that initializes all properties.**
* **Implement the truck class and update the main method to create a truck object and also create an object and also create an object car and bike subclass find display it details.**

Class Diagram:

|  |
| --- |
| Vechile |
| * Brand: String * speed: int |
| + Vechile(String,b int)  + Start()  + DisplayDetails() |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | Car | | * numberofdoors: int * seatingCapacity: int | | + car(String,int,int,int)  + displaydetails() | | |  | | --- | | Bike | | -hasGears: boolean | | + Bike(String,int,Boolean)  +displayetails() | |

|  |
| --- |
| Truck |
| -capacity: double |
| + truck(String,int,double)  +showtruckdetails()  +displaydetails() |

// Base class for Vehicle

class Vehicle {

protected String brand;

protected int speed;

public Vehicle(String brand, int speed) {

this.brand = brand;

this.speed = speed;

}

public void start() {

System.out.println(brand + " is starting.");

}

public void displayDetails() {

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

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

}

}

// Car class that extends Vehicle

class Car extends Vehicle {

private int numberOfDoors;

private int seatingCapacity;

public Car(String brand, int speed, int numberOfDoors, int seatingCapacity) {

super(brand, speed);

this.numberOfDoors = numberOfDoors;

this.seatingCapacity = seatingCapacity;

}

@Override

public void displayDetails() {

super.displayDetails();

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

System.out.println("Seating Capacity: " + seatingCapacity);

}

}

// Bike class that extends Vehicle

class Bike extends Vehicle {

private boolean hasGears;

public Bike(String brand, int speed, boolean hasGears) {

super(brand, speed);

this.hasGears = hasGears;

}

@Override

public void displayDetails() {

super.displayDetails();

System.out.println("Has Gears: " + (hasGears ? "Yes" : "No"));

}

}

// Truck class that extends Vehicle

class Truck extends Vehicle {

private double capacity; // in tons

public Truck(String brand, int speed, double capacity) {

super(brand, speed);

this.capacity = capacity;

}

public void showTruckDetails() {

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

}

@Override

public void displayDetails() {

super.displayDetails();

showTruckDetails();

}

}

// Main class to test the implementation

public class Main {

public static void main(String[] args) {

// Create a Car object

Car car = new Car("Toyota", 180, 4, 5);

car.start();

car.displayDetails();

System.out.println();

// Create a Bike object

Bike bike = new Bike("Yamaha", 120, true);

bike.start();

bike.displayDetails();

System.out.println();

// Create a Truck object

Truck truck = new Truck("Volvo", 100, 10.5);

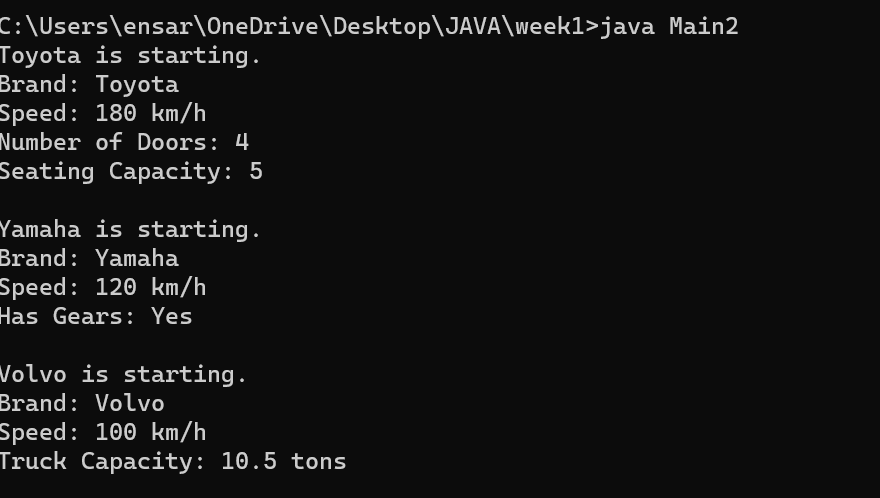
truck.start();

truck.displayDetails();

}

}

Output:



|  |  |  |
| --- | --- | --- |
| **S.NO** | **Errors** | **Rectification** |
| 1 | .variable | We must mention variable name to call the variable. |
| 2 | static | Static variables contain only one value. |

WEEK-6

1)Write a java program to create a vechiles class with a method displayinfo() override this method in the car subclass to provide specific information about a car

* Company
* Model
* Price
* Seating capacity
* Petrol or not

Pogram:

public class vehicles {

void displayinfo() {

System.out.println("This is a vehicle");

}

}

class Car extends vehicles {

String Name, Model, Capacity;

int Price;

boolean Petrol;

Car(String Name, String Model, String Capacity, int Price, boolean Petrol) {

this.Name = Name;

this.Model = Model;

this.Capacity = Capacity;

this.Price = Price;

this.Petrol = Petrol;

}

@Override

public void displayinfo() {

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

System.out.println("The model is: " + Model);

System.out.println("The price of the car is: " + Price);

System.out.println("The seating capacity of the car is: " + Capacity);

System.out.println("Is it petrol? " + Petrol);

}

}

class Main {

public static void main(String[] args) {

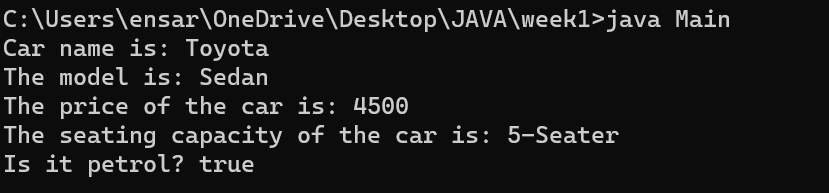
Car c1 = new Car("Toyota", "Sedan", "5-Seater", 4500, true);

c1.displayinfo();

}

}

**Output:**

****

**CLASS DIAGRAM:**

|  |
| --- |
| **Vehicle** |
| **+display info():void** |

|  |
| --- |
| **Car** |
| **+displayinfo():void** |

**ERROR:**

|  |  |  |
| --- | --- | --- |
| s.no | Expected error | reason |
| 1. | Settingtheparametersinside the constructor | Wecannotpassthe valuesinsideconstructor without setting them first |
| 2. | } | Ending the class and main method is required |

2Q) A college is developing automated admission system that verifies students eligibility for UG and PG programs .Each program has different eligibility criteria based on the students percentage in their previous qualification.

* UG admission require minimum 60%
* PG admission require minimum 70%

**Program:**

import java.util.Scanner;

class College {

String name;

int qualification;

int percentage;

// Constructor

College(String name, int qualification, int percentage) {

this.name = name;

this.qualification = qualification;

this.percentage = percentage;

}

// Default Eligibility method

public void Eligibility() {

System.out.println("Name: " + name + ", Qualification: " + qualification + ", Percentage: " + percentage);

System.out.println("The candidate is a fluke");

}

}

class UG extends College {

UG(String name, int qualification, int percentage) {

super(name, qualification, percentage);

}

@Override

public void Eligibility() {

System.out.println("Name: " + name + ", Qualification: " + qualification + ", Percentage: " + percentage);

System.out.println("The candidate is eligible for UG");

}

}

class PG extends College {

PG(String name, int qualification, int percentage) {

super(name, qualification, percentage);

}

@Override

public void Eligibility() {

System.out.println("Name: " + name + ", Qualification: " + qualification + ", Percentage: " + percentage);

System.out.println("The candidate is eligible for PG");

}

}

public class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

// Taking inputs

System.out.println("Enter your name:");

String name = input.nextLine();

System.out.println("Enter your qualification (e.g., 12 for high school, 10 for 10th, etc.):");

int qualification = input.nextInt();

System.out.println("Enter your percentage:");

int percentage = input.nextInt();

// Close scanner

input.close();

// Logic to check eligibility

College candidate;

if (percentage >= 70) {

candidate = new PG(name, qualification, percentage);

} else if (percentage >= 60) {

candidate = new UG(name, qualification, percentage);

} else {

candidate = new College(name, qualification, percentage);

}

candidate.Eligibility();

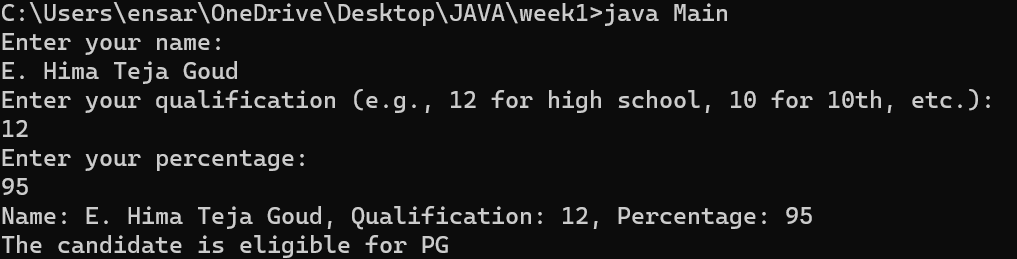
}

}

**Input:**

* **Name:E. Hima Teja Goud**
* **Qualification: 12**
* **Percentage: 95**

**Output:**

****

**CLASS DIAGRAM:**

|  |
| --- |
| **adm** |
| **+elg():void** |

|  |
| --- |
| **pg** |
| **+elg():void** |

|  |
| --- |
| **ug** |
| **+elg():void** |

**ERROR:**

|  |  |  |
| --- | --- | --- |
|  | 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)Create a calculator class with overloading methods to perform addition**

* **Add two doubles**
* **Add two integer**
* **Add three integer**

**Program:**

public class Calculator{

// Method to add two integers

public int add(int a, int b) {

return a + b;

}

// Method to add two tuples (represented as arrays)

public int add(int[] tuple1, int[] tuple2) {

int sum = 0;

for (int i = 0; i < tuple1.length; i++) {

sum += tuple1[i] + tuple2[i];

}

return sum;

}

// Method to add three integers

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

return a + b + c;

}

public static void main(String[] args) {

Calculator calc = new Calculator();

// Adding two integers

int result1 = calc.add(10, 20);

System.out.println("Addition of two integers: " + result1);

// Adding two tuples (arrays)

int[] tuple1 = {1, 2};

int[] tuple2 = {3, 4};

int result2 = calc.add(tuple1, tuple2);

System.out.println("Addition of two tuples: " + result2);

// Adding three integers

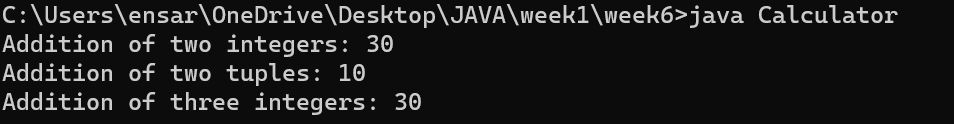
int result3 = calc.add(5, 10, 15);

System.out.println("Addition of three integers: " + result3);

}

}

OUTPUT:



CLASS DIAGRAM:

|  |
| --- |
| cal |
| +add(int a, int b):int  +add(doublea,doubleb):double  +add(int a,int b,int c):int |

**ERROR:**

|  |  |  |
| --- | --- | --- |
| 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 |

4.Create a shape class with method calculateArea() that is overloaded for different shapes (eg: square, rectangle).Then create a subclass Circle that overrides calculateArea() method for Circle.

**CODE:**

class shape{

public float calarea(float side){ return side\*side;

}

public float calarea(float l,float b){ return l\*b;

}

}

class circle extends shape{

public double calarea(double r){ return 3.14\*r\*r;

}

}

class s{

public static void main(String[] args){ circle c=new circle(); System.out.println(c.calarea(10));

}

}

OUTPUT:



**CLASS DIAGRAM:**

|  |
| --- |
| Shape |
| +calarea(float side):float  +calarea(float l,float b):float |

|  |
| --- |
| Circle |
| +calarea(double r):double |

ERRORS:

|  |  |  |
| --- | --- | --- |
| 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-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();

}

}

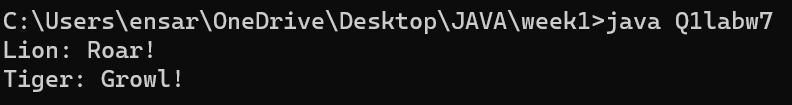
|  |
| --- |
| Abstract Animal |
| Sound():void |

CLASS DIAGRAM:

OUTPUT:

|  |
| --- |
| Lion |
| Sound():void |

|  |
| --- |
| Tiger |
| Sound():void |



ERRORS:

|  |  |  |
| --- | --- | --- |
| S.NO | errors | 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);

}

}

class Tej {

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();

}

}

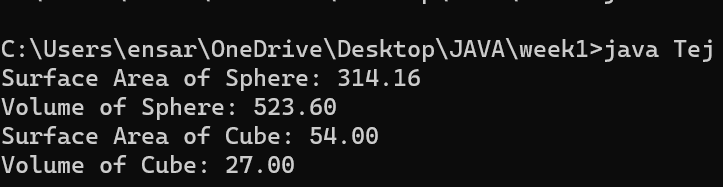
|  |
| --- |
| Abstact  Shape3D |
| a:double |
| Display pattern type(type:string):void |

CLASS DIAGRAM:

|  |
| --- |
| Cube |
| Vol():Void |
| Surface():void |

|  |
| --- |
| Sphere |
| Vol():void |
| Surface():void |

OUTPUT:



ERRORS:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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();

}

}

}

class Hima {

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);

}

}

|  |
| --- |
| Abstact  Pattern Print |
| Printing():void |
| Display pattern type(string):void |

CLASS DIAGRAM:

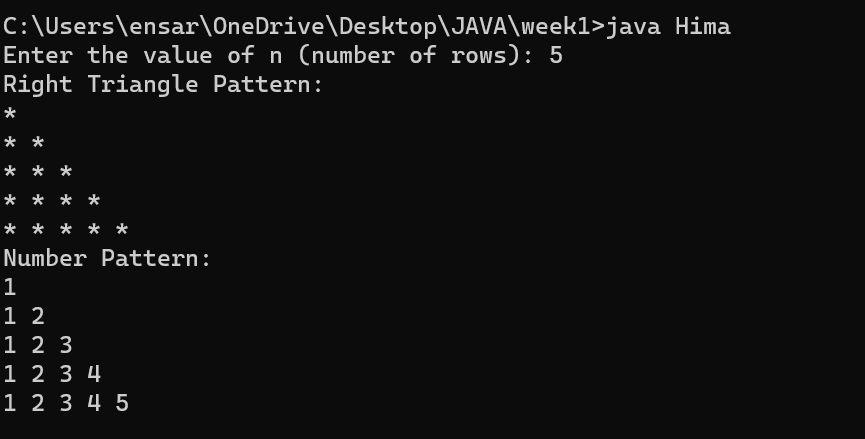
|  |
| --- |
| StarPattern |
| Pattern():void |
| Printing():void |

|  |  |
| --- | --- |
| Num |  |
| Pattern():void |  |
| Printing():void |  |

ERRORS:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 |  |  |  |

OUTPUT:



Week-8

**Q)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**

**CODE:**

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

**public interface Shape {**

**abstract int getperimeter();**

**}**

**class rectangle implements Shape{**

**int len,bred;**

**rectangle(int len, int bred){**

**this.len=len;**

**this.bred=bred;}**

**public int getperimeter(){**

**return 2\*(len+bred);**

**}**

**}**

**class triangle implements Shape{**

**int side1,side2,side3;**

**triangle(int side1, int side2, int side3){**

**this.side1=side1;**

**this.side2=side2;**

**this.side3=side3;}**

**public int getperimeter(){**

**return side1+side2+side3;**

**}**

**}**

**class Circle implements Shape{**

**double radius;**

**Circle(int radius){**

**this.radius=radius;}**

**public int getperimeter(){**

**return (int) (2 \* Math.PI \* radius);**

**}}**

**class interfaces{**

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

**Shape Rec= new rectangle(10,5);**

**Shape Tri=new triangle(10,10,10);**

**Shape circle= new Circle(10);**

**System.out.println("rect:"+Rec.getperimeter());**

**System.out.println("rect:"+Tri.getperimeter());**

**System.out.println("rect:"+ circle.getperimeter());**

**}**

**}**

**OUTPUT:**

****

**Class Diagrams:**

|  |
| --- |
| **<interfaces>**  **Shape** |
| **+getperimeter():double** |

|  |
| --- |
| **Rectangle** |
| **-Width: int**  **-height: int** |
| **+getperimeter():Double** |

|  |
| --- |
| Triangle |
| -side1:int  -side2:int  -side3:int |
| **+getperimeter():double** |

|  |
| --- |
| **Circle** |
| **-radius:int** |
| **+getperimeter():double** |

**Error Table:**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **Errors** | **Rectification** |
| 1 | this.side3=side; | this.side3 = side3; |
| 2 | Shape circle = Circle(4); | Shapecircle =newCircle(4); |

**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**

CODE:

import java.util.\*;

interface Playable{

void Play();

}

class Volleyball implements Playable{

public void Play(){

System.out.println("playing vollyball");

}

}

class Football implements Playable{

public void Play(){

System.out.println("football");

}

}

class Basketball implements Playable {

public void Play() {

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

}

}

class SportsGame {

public static void main(String[] tej){

Playable Volly= new Volleyball();

Playable foot= new Football();

Playable base= new Basketball();

Volly.Play();

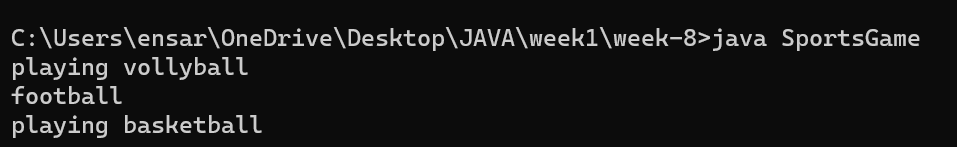
foot.Play();

base.Play();

}

}

OUTPUT:



|  |
| --- |
| <Interface>  Playable |
| +play():void |

CLASS DIAGRAM:

|  |
| --- |
| Basketball |
| +play():void |

|  |
| --- |
| Vollyball |
| +play():void |

|  |
| --- |
| Football |
| +play():void |

**Error Table:**

|  |  |  |
| --- | --- | --- |
| s.no | Expected Error | Reason |
| **1** | Settingteparameters inside the constructor | Wecannotpassthevalues 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 Tej {

public static void main(String[] args) {

UniversityPortal portal = new UniversityPortal();

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

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

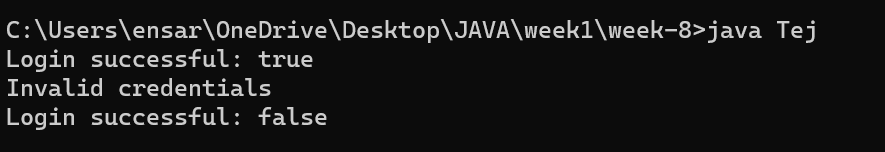
boolean loginSuccess2 = portal.login("student123", "wrong pass");

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

}

}

OUTPUT:



**Error Table:**

|  |  |  |
| --- | --- | --- |
| S.no | Expected Error | Reason |
| **1** | Settingtheparameters inside the constructor | Wecannotpassthe valuesinsideconstructor without setting them first |
| **2** | } | Ending the class and main method is required |