**23CSE111**

**OBJECT ORIENTED PROGRAMMING**

**LAB REPORT**



**Department of Computer Science Engineering**

**Amrita School of Computing**

**Amrita Vishwa Vidyapeetham, Amaravati Campus.**

Verified by Name: G. Pooja Priyanka

Roll no: AV.SC.U4CSE24107

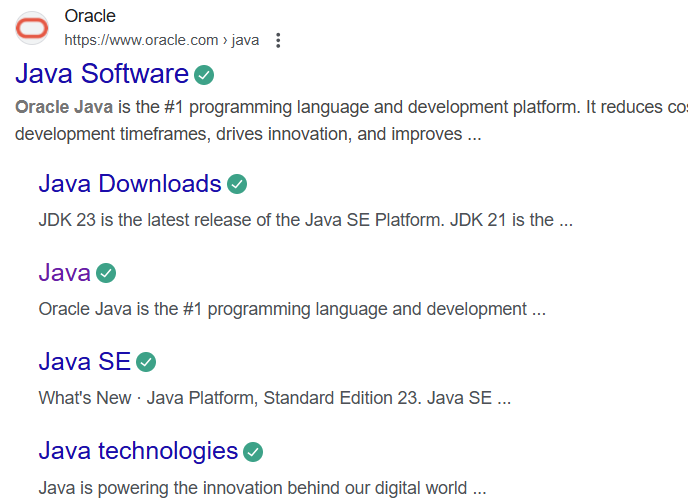
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No |  | Date | Page No | Signature |
| LAB-1 |  |  |  |  |
| a | How to download JDK-21 in your PC |  |  |  |
| b | Set environment variables in your PC |  |  |  |
| c | Checking of JDK-21 version |  |  |  |
| d | Write a simple java program to about an student |  |  |  |

**WEEK-1**

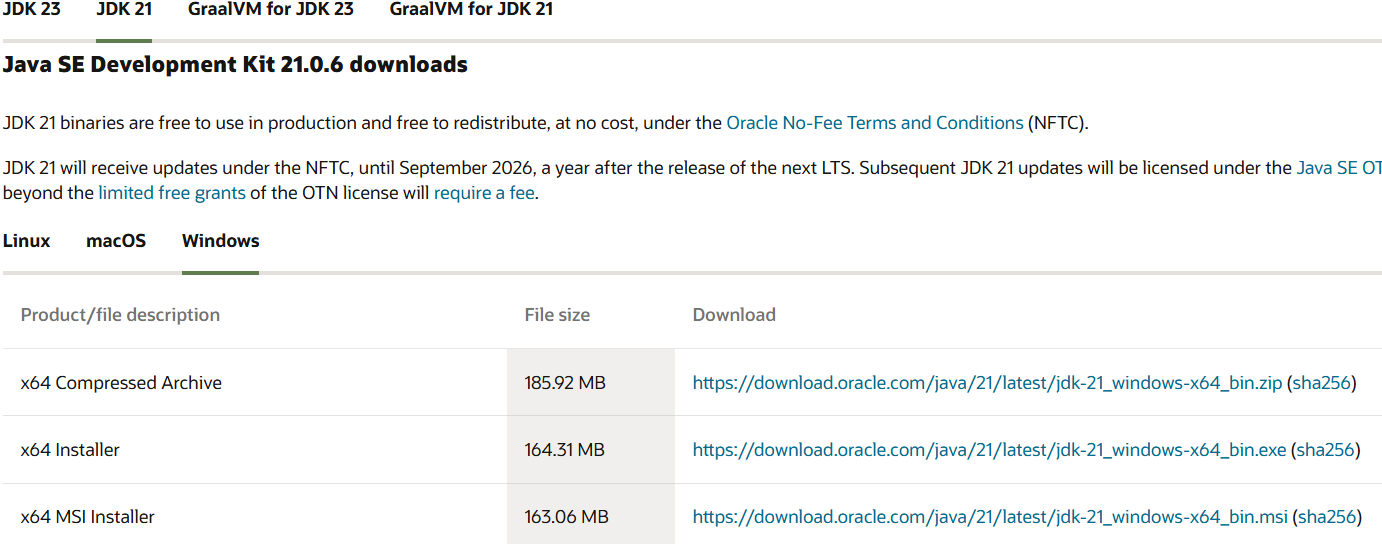
1) Explain the process of Installing JDK (Java Development Kit)

**Download:**

* Open web browser and search “Oracle Java”.



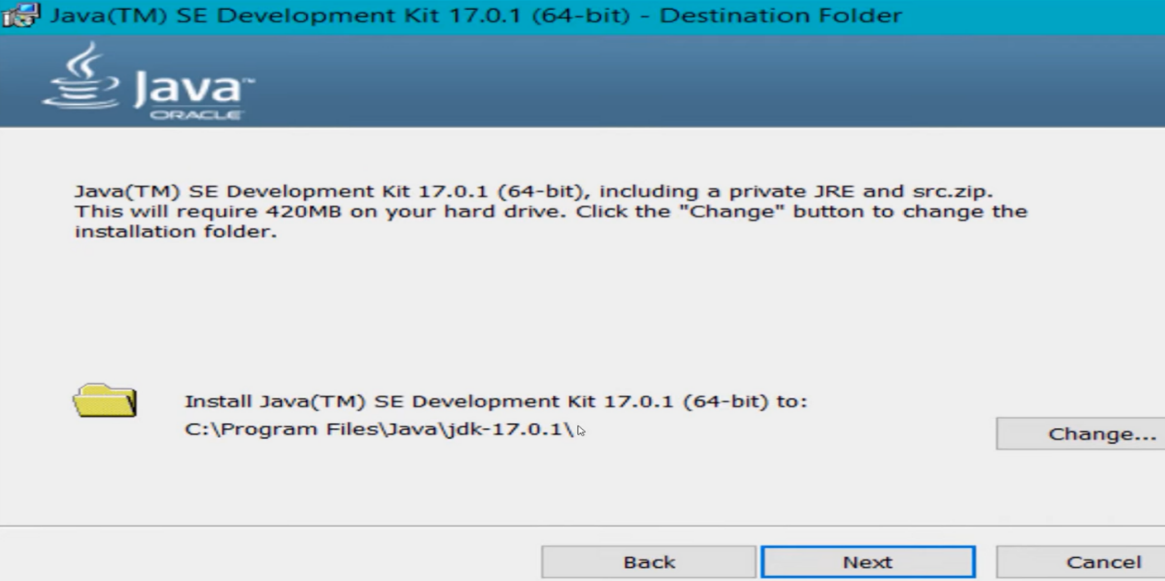
* Open the first website oracle.
* Click on “download Java” in the oracle interface.
* Select the version you want to download (version JDK21).
* Select the Operating System (OS) of your PC and click on the link “x64 installer” to download.



**Installation:**

* + Once downloaded, run the installer.
  + Follow the instructions and keep clicking "Next" until it's done.



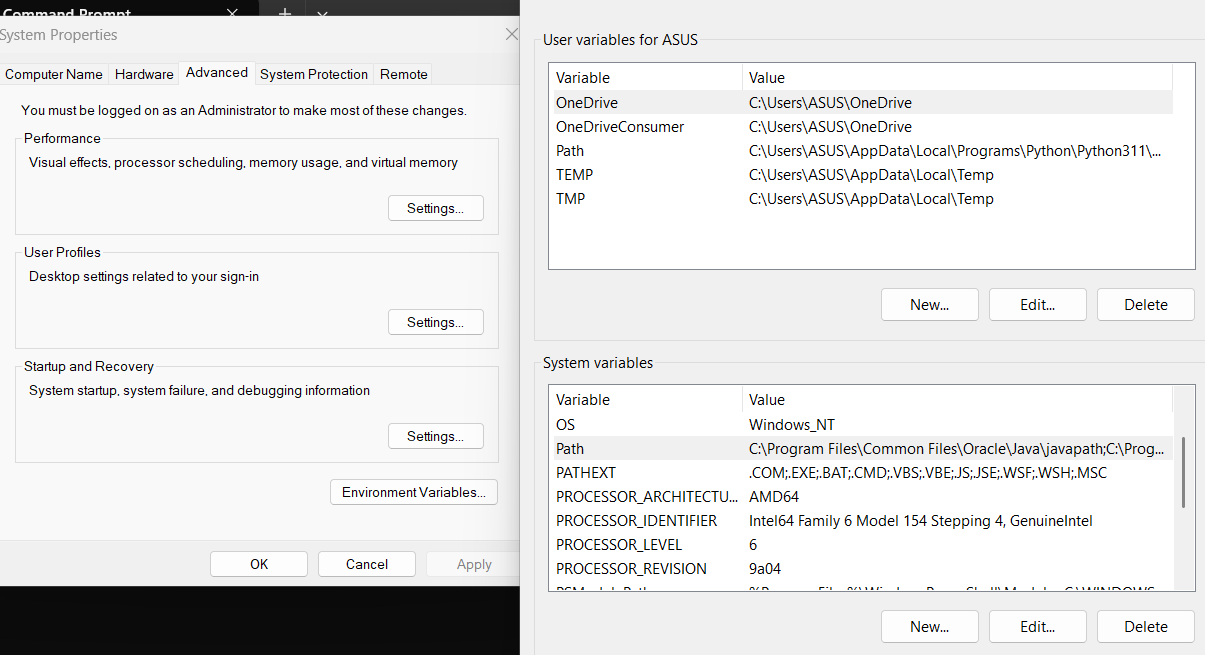




**Set Environment Variables (Windows):**

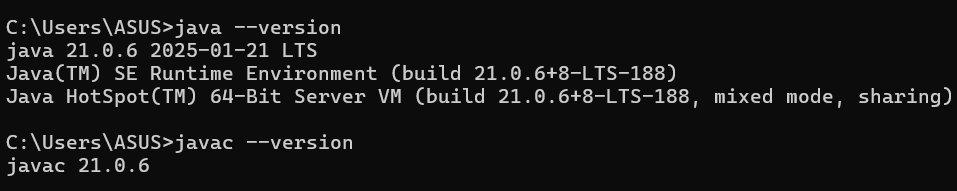
* + Open file explorer, then right click on This PC next select on properties then it will take you to the settings app then click on advanced system settings and then  
    click on **Environment Variables**.
  + Click **New** under **System Variables**:
    - **Set Variable name as:** java
    - **Variable value:** The folder address where JDK is installed ("C:\Users\ASUS\Downloads\jdk-21\_windows-x64\_bin.exe")
* Find Path under System Variables, click Edit, and add the path of the

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



**Checking of JDK Version:**

1. **Open Command Prompt:**
   * Press win+R, type cmd, and press Enter.
2. **Check Version:**
   * Type java --version and press Enter.
   * Type javac --version and press Enter.



**PROGRAMS:**

1)Write a Java program to print the message “Welcome to Java Programming.”

**CODE:**

class Hello{

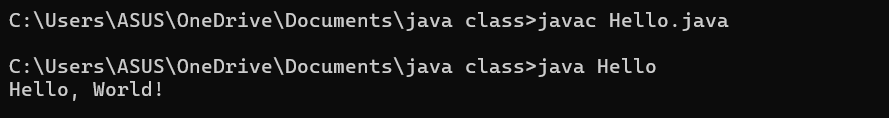
    public static void main(String[] args){

        System.out.println("Hello, World!");

    }

}

**OUTPUT:**



**ERRORS:** Found none

2) Write a Java Program that prints Name, Roll No, Section of a student.

**CODE:**

class student{

public static void main (String[] args){

System.out.println("Name: Pooja Priyanka");

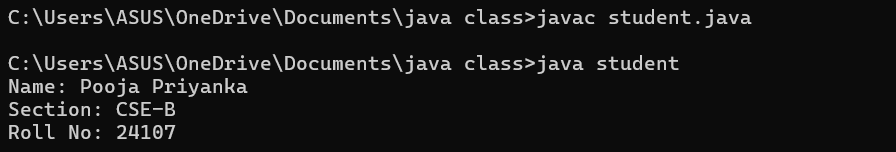
System.out.println("Section: CSE-B");

System.out.println("Roll No: 24107");

}

}

**OUTPUT:**



**ERROR TABLE:**

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

**IMPORTANT POINTS:**

1)System.out.println(“this string will be printed”) – this line of code is used to print any string.

2)if you want to save your java file as student then “class student { “ should be written”}

**WEEK-2**

1)write a java program on largest of 3 numbers using ternary operators.

**CODE:**

import java.util.Scanner;

class largest\_number{

public static void main(String[] args){

Scanner input = new Scanner(System.in);

System.out.println("enter the first number:");

int a=input.nextInt();

System.out.println("enter the second number:");

int b=input.nextInt();

System.out.println("enter the third number:");

int c=input.nextInt();

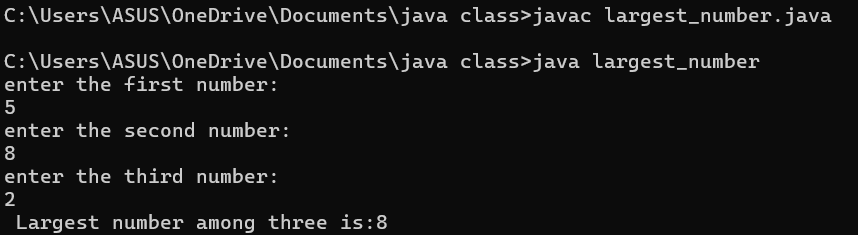
int largest\_number = ((a > b && a > c)? a: (b > a && b>c)?b :c);

System.out.println(" Largest number among three is:" + largest\_number);

}

}

**OUTPUT:**

****

**ERROR:**

|  |  |
| --- | --- |
| **Error** | **Rectification** |
| For root in java we don’t use \*\*. | Use Math.sqrt statement.  Double area=Math.sqrt(x); |

2)write a java program for factorial of a number.

**CODE:**

import java.util.Scanner;

class factorial {

public static void main(String[] args) {

Scanner f=new Scanner(System.in);

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

int number=f.nextInt();

long factorial = 1;

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

factorial \*= i;

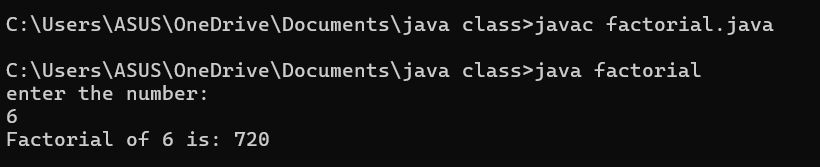
}

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

}

}

**OUTPUT:**

****

**ERROR:**

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

3) write a java program on Fibonacci series.

**CODE:**

import java.util.Scanner;

class fibonacci {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.println("enter the number of terms : ");

int n = input.nextInt();

int f0 = 0, f1 = 1;

System.out.println("Fibonacci Series up to " + n + " numbers:");

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

System.out.print(f0+ " ");

int fn = f0 + f1;

f0 = f1;

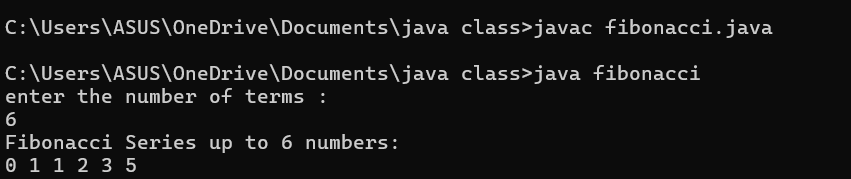
f1 = fn;

}

}

}

**OUTPUT:**

****

**ERROR:**

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

4.a) Write a program in java for converting temperature from celsius to fahrenheit***.***

**CODE:**

import java.util.Scanner;

class TemperatureConversion {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

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

float ctemp = input.nextFloat();

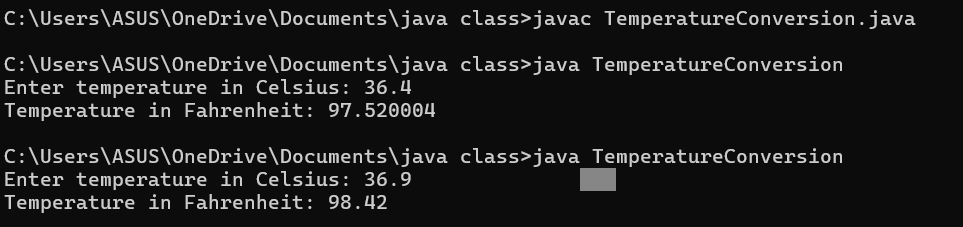
float ftemp = (ctemp \* 9 / 5) + 32;

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

}

}

**OUTPUT:**



**ERROR:**

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

**IMPORTANT POINTS:**

1.The formula to convert a Celsius to Fahrenheit is

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

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

4.b) Write a program in java for converting temperature from Fahrenheit to celsius***.***

**CODE:**

import java.util.Scanner;

class TemperatureConversion1 {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

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

float ftemp = input.nextFloat();

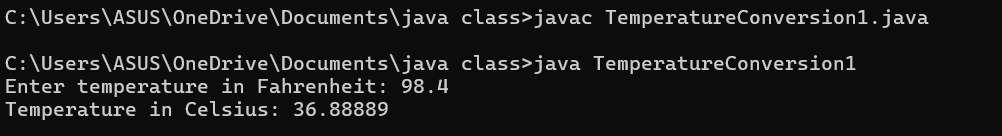
float ctemp=(ftemp-32)\*5/9;

System.out.println("Temperature in Celsius: " + ctemp);

}

}

**OUTPUT:**



**ERROR:**

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

**IMPORTANT POINTS:**

1.The formula to convert a Fahrenheit to Celsius is

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

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

5) write a java program for area of rectangle.

**CODE:**

import java.util.Scanner;

class AreaRectangle{

public static void main(String[] args){

Scanner input = new Scanner(System.in);

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

double length = input.nextDouble();

System.out.println("enter width:");

double width = input.nextDouble();

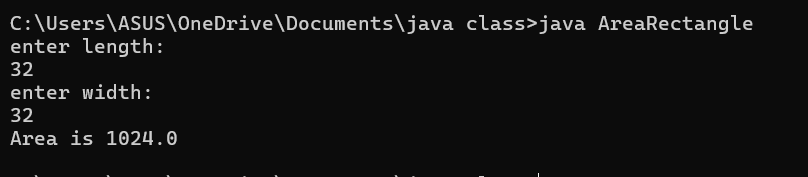
double area = length\*width;

System.out.println("Area is "+area);

}

}

**OUTPUT:**

****

**ERROR:**

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

**IMPORTANT POINTS:**

1. declare the data type as double

2.formula for calculating are of rectangle

Area = length\*width

6)write a java program on calculating simple interest.

**CODE:**

import java.util.Scanner;

class SimpleInterest {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

// Reading principal amount,time period,rate of interest from user

System.out.print("Enter the principal amount: ");

double principal = input.nextDouble();

System.out.print("Enter the rate of interest: ");

double rate = input.nextDouble();

System.out.print("Enter the time period (in years): ");

double time = input.nextDouble();

// Calculating simple interest

double simpleInterest = (principal \* rate \* time) / 100;

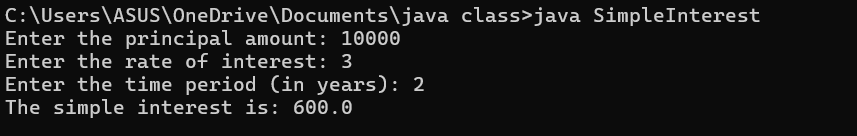
// Displaying the simple interest

System.out.println("The simple interest is: " + simpleInterest);

}

}

**OUTPUT:**

****

**ERROR:**

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

**WEEK-3**

1)write a java program with following instructions:

1. Create a class with name car.
2. Create 4 attributes named(colour,brand,fueltype,mileage).
3. Create 3 methods named(start,stop,service).
4. Create 3 objects named c1,c2,c3.
5. Create a constructor with parameters colours,brand,fueltype,mileage.

**CODE:**

public class car1 {

    // Attributes

    String color;

    String brand;

    String fuelType;

    float mileage;

    // Constructor with parameters

    car1(String color, String brand, String fuelType, float mileage) {

        this.color = color;

        this.brand = brand;

        this.fuelType = fuelType;

        this.mileage = mileage;

    }

    // Declaring methods

    public void car1() {

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

    }

    public void start() {

        System.out.println("The car is started");

    }

    public void stop() {

        System.out.println("The car is stopped");

    }

    public void service() {

        System.out.println("The car is serviced");

    }

public void displayDetails(){

    System.out.println("Color: " + color);

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

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

    System.out.println("Mileage: " + mileage);

}

    // Main method

    public static void main(String[] args) {

        // Using the default constructor

        car1 c1 = new car1();

        c1.color = "Red";

        c1.brand = "Kia";

        c1.fuelType = "Diesel";

        c1.mileage = 24;

    car1 c2 = new car1();

        c2.color = "black";

        c2.brand = "hyundai";

        c2.fuelType = "Diesel";

        c2.mileage = 20;

    car1 c3 = new car1();

        c3.color = "grey";

        c3.brand = "BMW";

        c3.fuelType = "Diesel";

        c3.mileage = 30;

        c1.car1();

        c1.start();

        c1.stop();

        c1.service();

        c1.displayDetails();

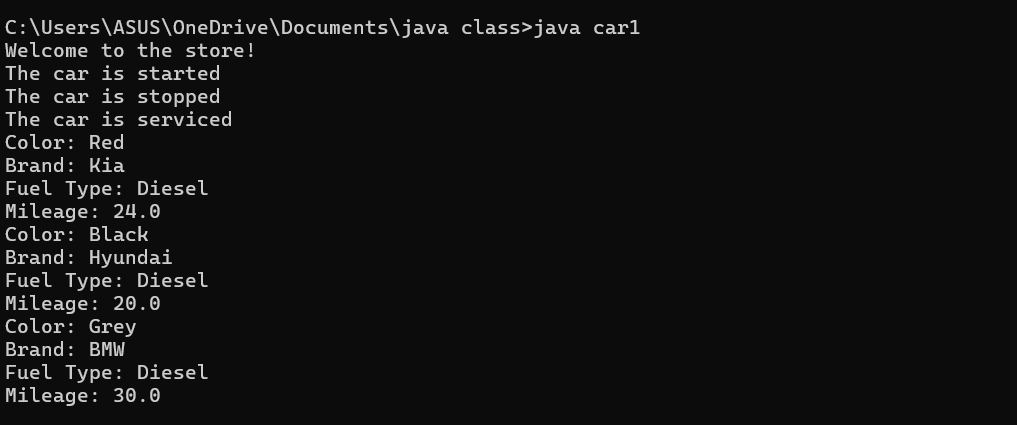
        c2.displayDetails();

        c3.displayDetails();

    }

}

**OUTPUT:**

****

**ERROR:**

|  |  |
| --- | --- |
| **Error** | **Rectification** |
| Error: car\_Color is undefined, should be car\_color | **Replace:** car\_Color with car\_color |
| Missing semicolon after 'car2.start()' will cause an error | **Replace:** add semi colon at the end |

**IMPORTANT POINTS:**

**1. Variable name mismatch**: The variable car\_Color in the code should be car\_color

2.**Incorrect variable name**: car1.car\_color is used when the actual variable is car1.car\_Color, which will cause an error due to case sensitivity.

3. **Missing Semicolon**: Forgetting to add a semicolon at the end of a statement will cause a compilation error.

2) create a class named back account with methods deposit and withdraw where the deposit method should accepts a parameter and when this method is called the deposited amount should be added to the current balance in addition to that when a with draw method is called it has to verify whether the withdraw amount is less-than the current balance if not displaying a message saying that there are insufficient funds. use the constructor to display the details of the costumer (name,account number, IFSC,branch) atleast 2 costumers as objects c1, c2.

**CODE:**

public class bank {

// Attributes

String name;

String accountNumber;

String ifsc;

String branch;

double currentBalance;

// Constructor to initialize customer details

bank(String name, String accountNumber, String ifsc, String branch, double currentBalance) {

this.name = name;

this.accountNumber = accountNumber;

this.ifsc = ifsc;

this.branch = branch;

this.currentBalance = currentBalance;

System.out.println("Account created for " + name);

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

System.out.println("IFSC Code: " + ifsc);

System.out.println("Branch: " + branch);

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

}

// Method to deposit money

public void deposit(double amount) {

if (amount > 0) {

currentBalance += amount;

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

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

} else {

System.out.println("Invalid deposit amount.");

}

}

// Method to withdraw money

public void withdraw(double amount) {

if (amount > 0 && amount <= currentBalance) {

currentBalance -= amount;

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

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

} else {

System.out.println("Invalid withdrawal amount.");

}

}

// Method to display account details

public void displayAccountDetails() {

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

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

System.out.println("IFSC Code: " + ifsc);

System.out.println("Branch: " + branch);

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

}

public static void main(String[] args) {

// Creating c1

bank c1 = new bank("pooja", "123456789", "IFSC1", "Branch1", 5000.0);

c1.deposit(2000.0);

c1.withdraw(3000.0);

c1.withdraw(5000.0);

// Creating c2

bank c2 = new bank("priyanka", "987654321", "IFSC2", "Branch2", 10000.0);

c2.deposit(5000.0);

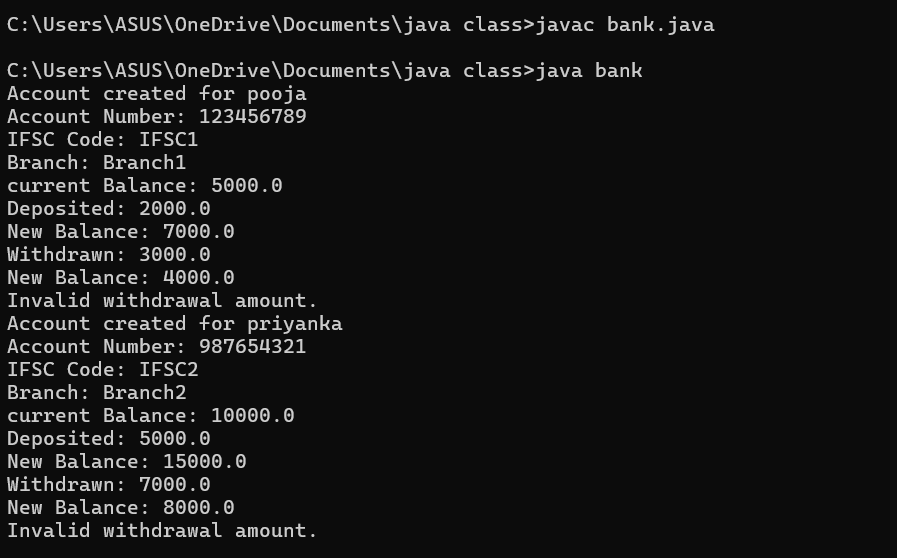
c2.withdraw(7000.0);

c2.withdraw(9000.0);

}

}

**OUTPUT:**



**ERROR:**

|  |  |
| --- | --- |
| **Error** | **Rectification** |
| this.existing = int.nextFloat(); | this.existing = input.nextFloat(); |
| public int deposit() is incorrect return method | REPLACE:public VOID deposit() |

**IMPORTANT POINTS:**

1.The balance should be a float or double to handle decimal values correctly, but it's declared as an int.

2. **Incorrect deposit method signature**: The method DEPOSIT()has an incorrect return type int(), while it should be void since it doesn't need to return any value.

3. **Fixed the return type of deposit**: Changed from int to void, as the method does not need to return anything

**WEEK-4**

1). Write a java program with class named Book

a) a class should contain various attributes such as title, author, year of publication.

b) it should also contain a constructor with parameters which initializes title, author, year of publication.

c) create a method which displays the details of the book title, author, year of publication

Display the details of two books.

**CODE:**

public class book {

String title;

String author;

int year\_of\_publication;

// Constructor to initialize book details

book(String title, String author, int year\_of\_publication) {

this.title = title;

this.author = author;

this.year\_of\_publication = year\_of\_publication;

}

void display() {

System.out.println("Title of the book: " + title);

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

System.out.println("Year of publication: " + year\_of\_publication);

}

public static void main(String[] args) {

book c1 = new book("Harry Potter 1", "J.K. Rowling", 2007);

c1.display();

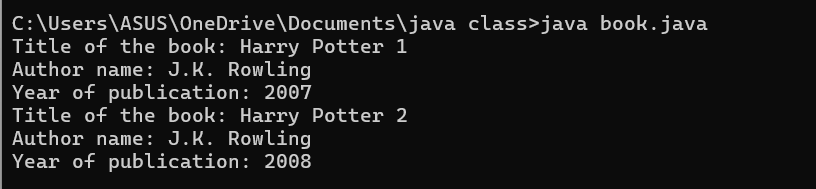
book c2 = new book("Harry Potter 2", "J.K. Rowling", 2008);

c2.display();

}

}

**OUTPUT:**

**ERRORS:**

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

**IMPORTANT POINTS:**

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

**CLASS DIAGRAM:**

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

2)Create a java program with class named ‘MyClass’ with static variable count of int type,initaialized to zero and a constant variable ‘pi’ or type double initialized to 3.14 as attributes of the class. Now define a constructor for “MyClass”that increments the count variable eachtime an object of MyClass is created.Finally print the final values of count and pi variables. Create three objects and a constructor.

**CODE:**

class Myclass {

// Static variable to hold count of objects created

static int count = 0;

// Constant variable for pi, initialized to 3.14

final double pi = 3.14;

// Constructor

public Myclass() {

// Increment count every time an object is created

count++;

}

// Main method to create objects and print values

public static void main(String[] args) {

// Creating three objects of MyClass

Myclass object1 = new Myclass();

Myclass object2 = new Myclass();

Myclass object3 = new Myclass();

// Printing the final values of count and pi

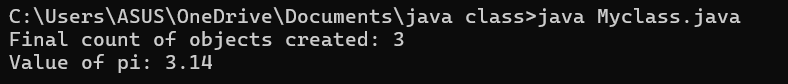
System.out.println("Final count of objects created: " + count);

System.out.println("Value of pi: " + object1.pi); // Pi is constant, and we can access it via any object

}

}

**OUTPUT:**

**ERRORS:**

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

**IMPORTANT POINTS:**

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

**CLASS DIAGRAM:**

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

**WEEK-5**

1Q) Create a calculator using the operations including addition, subtraction, multiplication, and division using multi-level inheritance and display the desired output.

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

**CLASS DIAGRAM:**

|  |
| --- |
| Calc |
| - num1: double |
| - num2: double |
| + Calc(num1, num2) |
| Addition |
| + add(): double |
| Subtraction |
| + subtract(): double |
| Multiplication |
| + multiply(): double |
| Division |
| + divide(): String |
|  |

**CODE:**

public class calc {

    double num1, num2;

    // Constructor to initialize numbers

    public calc(double num1, double num2) {

        this.num1 = num1;

        this.num2 = num2;

    }

    // Addition class

    static class Addition extends calc {

        public Addition(double num1, double num2) {

            super(num1, num2);

        }

        // addition

        public double add() {

            return num1 + num2;

        }

    }

    // Subtraction class

    static class Subtraction extends Addition {

        public Subtraction(double num1, double num2) {

            super(num1, num2);

        }

        // subtraction

        public double subtract() {

            return num1 - num2;

        }

    }

    // Multiplication class

    static class Multiplication extends Subtraction {

        public Multiplication(double num1, double num2) {

            super(num1, num2);

        }

        // multiplication

        public double multiply() {

            return num1 \* num2;

        }

    }

    // Division class

    static class Division extends Multiplication {

        public Division(double num1, double num2) {

            super(num1, num2);

        }

        //  division with condition for zero

        public String divide() {

            if (num2 != 0) {

                return String.valueOf(num1 / num2);

            } else {

                return "Error: Division by zero";

            }

        }

    }

    // main method to execute the calculator operations

    public static void main(String[] args) {

        // Create objects for each operation

        Addition add1 = new Addition(12, 6);

        Subtraction subtract1 = new Subtraction(12, 6);

        Multiplication multiply1 = new Multiplication(12, 6);

        Division divide1 = new Division(12, 6);

        // Displaying the results

        System.out.println("Addition: " + add1.add());

        System.out.println("Subtraction: " + subtract1.subtract());

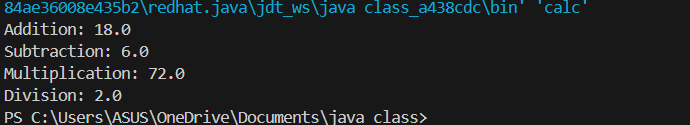
        System.out.println("Multiplication: " + multiply1.multiply());

        System.out.println("Division: " + divide1.divide());

    }

}

**Output:**

****

**ERRORS:**

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

**2Q)** A vehicle rental company wants to develop a system that maintains information about different types of vechicles available for rent the company rents out cars and bikes, and they need a program to store details about each vehicle, such as brand and speed( should be in super class)

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

**Question:**

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

**IMPORTANT POINTS:**

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

**Answer for Q1:**

The oops concepts used in the above program are:

Inheritance, encapsulation, polymorphism, abstraction.

**CLASS DIAGRAM:**

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

|  |  |
| --- | --- |
| Car | |
| -no.of.doors:int | |
| +int (brand, speed,    No.of doors);  +display deatails(); | |
| Bikes |
| -has gears:bool |
|  |

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

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

**CODE:**

class Vehicle {

    private String brand;

    private int speed;

    Vehicle(String brand, int speed) {

        this.brand = brand;

        this.speed = speed;

    }void details() {

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

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

    }

}

class Car extends Vehicle {

    private int doors;

    private int capacity;

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

        super(brand, speed);

        this.doors = doors;

        this.capacity = capacity;

    }

    void carDetails() {

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

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

    }

    @Override

    void details() {

        super.details();

        carDetails();

    }

}

class Bike extends Vehicle {

    private boolean gears;

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

        super(brand, speed);

        this.gears = gears;

    }

    void bikeDetails() {

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

    }

    @Override

    void details() {

        super.details();

        bikeDetails();

    }

}

class Truck extends Vehicle {

    private int tons;

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

        super(brand, speed);

        this.tons = tons;

    }

    void truckDetails() {

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

    @Override

    void details() {

        super.details();

        truckDetails();

    }

}

public class Rent {

    public static void main(String[] args) {

        Car c = new Car("Toyota", 100, 5, 5);

        c.details();

        Bike b = new Bike("KTM", 90, true);

        b.details();

        Truck t = new Truck("TATA", 80, 1);

        t.details();

    }

}

**ERROR:**

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

**WEEK-6**

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

**CLASS DIAGRAM:**

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

**CODE:**

class Vehicle {

    String car\_company;

    String car\_model;

    long car\_price;

    int seating\_capacity;

    boolean petrol;

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

        this.car\_company = car\_company;

        this.car\_model = car\_model;

        this.car\_price = car\_price;

        this.seating\_capacity = seating\_capacity;

        this.petrol = petrol;

    }

    void displayInfo() {

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

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

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

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

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

    } }

class Car extends Vehicle {

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

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

    }

}

public class Main {

    public static void main(String[] args) {

        // Creating a Car object with correct arguments

        Car c1 = new Car("Hyundai", "Creta", 1500000, 5, false);

        c1.displayInfo();

**ERRORS:**

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

**IMPORTANT POINTS:**

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

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

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

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

**PROGRAM-2:**

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

1. UG admission recquire a minimum of 60%.

2. PG admission recquire a minimum of 70%.

**CLASS DIAGRAM:**

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

**CODE:**

class University6 {

    double percentile;

    // Constructor to initialize percentile

    University6(double percentile) {

        this.percentile = percentile;

    }

    // Method to check eligibility for UG

    void ug() {

        if (percentile >= 60) {

            System.out.println("Eligible for UG");

        } else {

            System.out.println("Not eligible for UG");

        }

    }

    // Method to check eligibility for PG

    void pg() {

        if (percentile >= 70) {

            System.out.println("Eligible for PG");

        } else {

            System.out.println("Not eligible for PG");

        }

    }

}

// 'Eli' class inherits 'University6'

class Eli extends University6 {

    // Constructor calls the superclass constructor

    Eli(double percentile) {

        super(percentile);

    }

    // Override the UG eligibility check

    @Override

    void ug() {

        if (percentile >= 60) {

            System.out.println("Eligible for UG");

        } else {

            System.out.println("Not eligible for UG");

        }

    }

    // Override the PG eligibility check

    @Override

    void pg() {

        if (percentile >= 70) {

            System.out.println("Eligible for PG");

        } else {

            System.out.println("Not eligible for PG");

        }

    }

    public static void main(String[] args) {

        // Create an instance of 'Eli' with a percentile of 60.7

        Eli e1 = new Eli(60.7);

        // Check eligibility for UG and PG

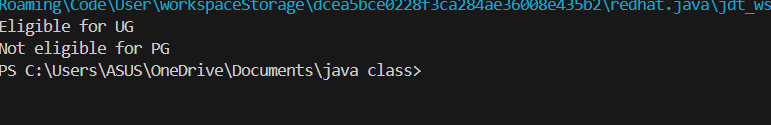
        e1.ug();

        e1.pg();

    }

}

**Output:**



**ERRORS:**

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

**IMPORTANT POINTS:**

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

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

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

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

**PROGRAM-3:**

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

1. Add two integers

2. Add two doubles

3. Add three integers

**CLASS DIAGRAM:**

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

CODE:

public class cal {

    double sum;

    void add(int a, int b) {

        sum = a + b;

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

    }

    void add(double a, double b) {

        sum = a + b;

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

    }

    void add(int a, int b, int c) {

        sum = a + b + c;

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

    }

    public static void main(String[] args) {

        cal c1 = new cal();

        c1.add(2, 7, 3);

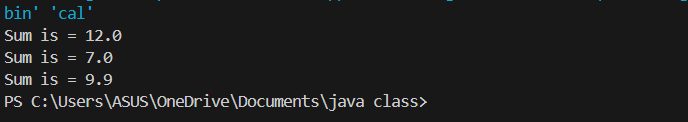
        c1.add(3,4);

        c1.add(3.2, 6.7);

    }

}

**Output:**

****

**ERRORS:**

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

**IMPORTANT POINTS:**

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

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

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

**CLASS DIAGRAM:**

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

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

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

**CODE:**

 public class shape{

    double l;

    double b;

    double area;

    void calarea(double l,double b){

    area=l\*b;

    System.out.println("rect area="+area);

    }

    void calarea(int b){

    area=3.14\*b\*b;

    System.out.println("cir area="+area);

    }

    }

    class cir extends shape{

    void calarea(int b){

    area=3.14\*b\*b;

    System.out.println("cir area="+area);

    }

    public static void main(String[] args){

    shape s1=new shape();

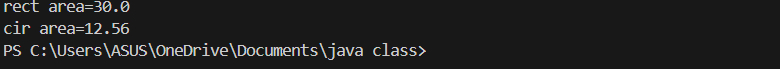
    s1.calarea(5,6);

    s1.calarea(2);

    }

    }

**Output:**



**ERRORS:**

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

**IMPORTANT POINTS:**

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

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

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

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

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

**Week-7**

1Q) creating one superclass Animal and three subclasses, Herbivores, Carnivores, and Omnivores. Subclasses extend the superclass and override its eat() method. Returning the method for the required type of animals.

**Class diagram:**

|  |
| --- |
| <<abstract>> animal |
| +sound():void |
| Lion |
| +sound():void |
| Tiger |
| +sound():void |

**Code:**

public class animal {

    public void eat() {

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

    }

}

class Herbivores extends animal {

    @Override

    public void eat() {

        System.out.println("Herbivore eats only plants.");

    }

}

class Carnivores extends animal {

    @Override

    public void eat() {

        System.out.println("Carnivore eats only meat.");

    }

}

// Subclass Omnivores

class Omnivores extends animal {

    @Override

    public void eat() {

        System.out.println("Omnivore eats both plants and meat.");

    }

}

class test {

    public static void main(String[] args) {

        System.out.println("G.POOJA PRIYANKA,24107,CSE-B");

        animal a = new animal();

        animal herbivore = new Herbivores();

        animal carnivore = new Carnivores();

        animal omnivore = new Omnivores();

        a.eat();

        herbivore.eat();

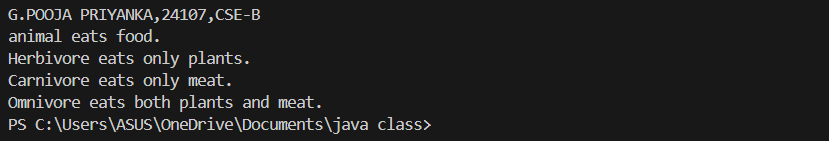
        carnivore.eat();

        omnivore.eat();

    }

}

**Output:**

****

Important points:

1)JVM(java virtual machine) is one which runs our program.

* JVM starts your program by calling the main( ) method from outside your class.
* JVM runs outside our package.
* only public method are accessible from outside a class or package.
* if your main( ) not marked as public ,jam cannot see it and you get an error like above

2Q) 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.

**Class diagram:**

**Code:**

import java.lang.\*;

abstract class shape3D {

    abstract double calculateVolume();

    abstract double calculateSurfaceArea();

}

class Sphere extends shape3D {

    private double radius;

    Sphere(double radius) {

        this.radius = radius;

    }

    @Override

    double calculateVolume() {

        return (4.0 / 3.0) \* Math.PI \* radius \* radius \* radius;

    }

    @Override

    double calculateSurfaceArea() {

        return 4 \* Math.PI \* radius \* radius;

    }

}

class  Cube extends shape3D {

    private double side;

    Cube(double side) {

        this.side = side;

    }

    @Override

    double calculateVolume() {

        return side \* side \* side;

    }

    @Override

    double calculateSurfaceArea() {

        return 6 \* side \* side;

    }

}

class main {

    public static void main(String[] args) {

        System.out.println("G.POOJA PRIYANKA,24107,CSE-B");

        Sphere s = new Sphere(2.3);

        System.out.println("Volume of Sphere: " + s.calculateVolume());

        System.out.println("Surface Area of Sphere: " + s.calculateSurfaceArea());

        Cube c = new Cube(2.3);

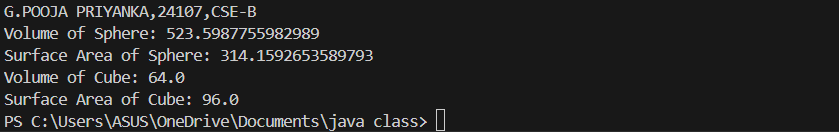
        System.out.println("Volume of Cube: " + c.calculateVolume());

        System.out.println("Surface Area of Cube: " + c.calculateSurfaceArea());

    }

}

**Output:**



3Q)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 angle triangle of Star( \*)

2. Numberpattern-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 Expected output:

Pattern 1:

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

Pattern 2:

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

**Code:**

abstract class patternprinter {

    abstract  void patternprinting(int n);

}

class StarPattern extends patternprinter {

    void patternprinting(int n) {

        System.out.println("Star pattern");

        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 patternprinting(int n) {

        System.out.println("Number pattern");

        for(int i=1; i<=n; i++) {

            for(int j=1; j<=i; j++) {

                System.out.print(j);

            }

            System.out.println();

        }

    }

}

public class patternmain {

    public static void main (String[] args) {

        System.out.println("G.POOJA PRIYANKA,24107,CSE-B");

        StarPattern s = new StarPattern();

        Numberpattern n = new Numberpattern();

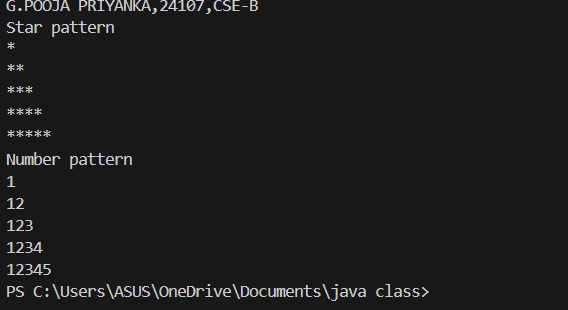
        s.patternprinting(5);

        n.patternprinting(5);

    }

}

**Output:**

****

**WEEK-8**

1Q)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:**

**Code:**

interface Shape {

double getPerimeter();

}

class Rectangle implements Shape {

private double length;

private double width;

public Rectangle(double length, double width) {

this.length = length;

this.width = width;

}

public double getPerimeter() {

return 2 \* (length + width);

}

}

class Circle implements Shape {

private double radius;

public Circle(double radius) {

this.radius = radius;

}

public double getPerimeter() {

return 2 \* Math.PI \* radius;

}

}

class Triangle implements Shape {

private double a, b, c;

public Triangle(double a, double b, double c) {

this.a = a;

this.b = b;

this.c = c;

}

public double getPerimeter() {

return a + b + c;

}

}

public class perimeter {

public static void main(String[] args) {

System.out.println("G.POOJA PRIYANKA,24107,CSE-B");

Shape rectangle = new Rectangle(5, 10);

Shape circle = new Circle(8);

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

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

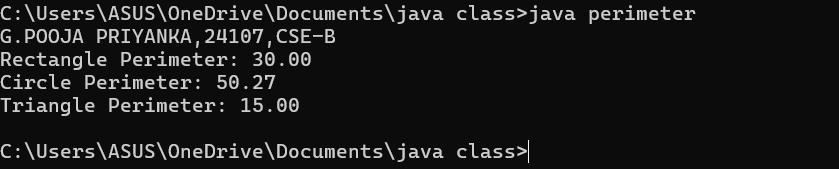
System.out.println("Circle Perimeter: "+circle.getPerimeter());

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

}

}

**Output:**

****

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 implement the Playable interface and override the play() method to play the respective sports.

**Class diagram:**

**Code:**

interface Playable {

    void play();

}

class Football implements Playable {

    @Override

    public void play() {

        System.out.println("Playing Football!");

    }

}

class Volleyball implements Playable {

    @Override

    public void play() {

        System.out.println("Playing Volleyball!");

    }

}

class Basketball implements Playable {

    @Override

    public void play() {

        System.out.println("Playing Basketball!");

    }

}

class Main5{

    public static void main(String[] args) {

        System.out.println("G.POOJA PRIYANKA,24107,CSE-B");

        Playable football = new Football();

        Playable volleyball = new Volleyball();

        Playable basketball = new Basketball();

        football.play();

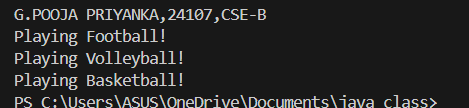
        volleyball.play();

        basketball.play();

    }

}

**Output:**

****

**Errors:**

|  |  |
| --- | --- |
| **Errors** | **Rectification** |
| Football is not abstract and does not override abstract method play() | Implement the play() method from interface |
| Playables.java but public class PlayTest | Class PlayTest is public , should be declared in a file named PlayTest.java |

**Important points:**

1.The Playable interface defines the contract (play() method).

2.Each class (Football, Volleyball, Basketball) implements the interface and provides a specific behavior for play().

3.This demonstrates polymorphism using interfaces in Java.

3Q) write a java program to implement a login system using interface.

**Code:**

interface Loginsystem {

    boolean user(String username, String password);

}

class UserLogin implements Loginsystem {

    private String validUsername;

    private String validPassword;

    public UserLogin(String validUsername, String validPassword) {

        this.validUsername = validUsername;

        this.validPassword = validPassword;

    }

    @Override

    public boolean user(String username, String password) {

        // Check if the entered username and password match the stored values

        return username.equals(validUsername) && password.equals(validPassword);

    }

}

public class login {

    public static void main(String[] args) {

        System.out.println("G. POOJA PRIYANKA,224107,CSE-B");

        UserLogin login = new UserLogin("priya", "pp@123");

        String username = "priya";

        String password = "pp@123";

        if (login.user(username, password)) {

            System.out.println("Login successful!");

        } else {

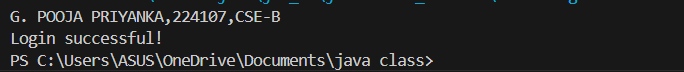
            System.out.println("Invalid  credentials.");

        }

    }

}

**Output:**

****