

**(23CSE111) OBJECT ORIENTED PROGRAMMING**

**LAB MANUAL**

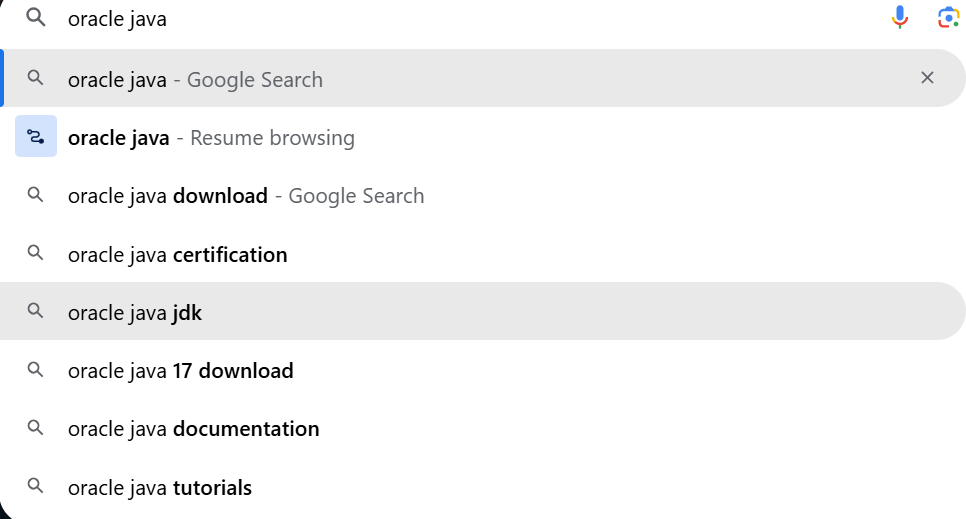
**CSE-1st YEAR II SEMESTER (2025-2026)**

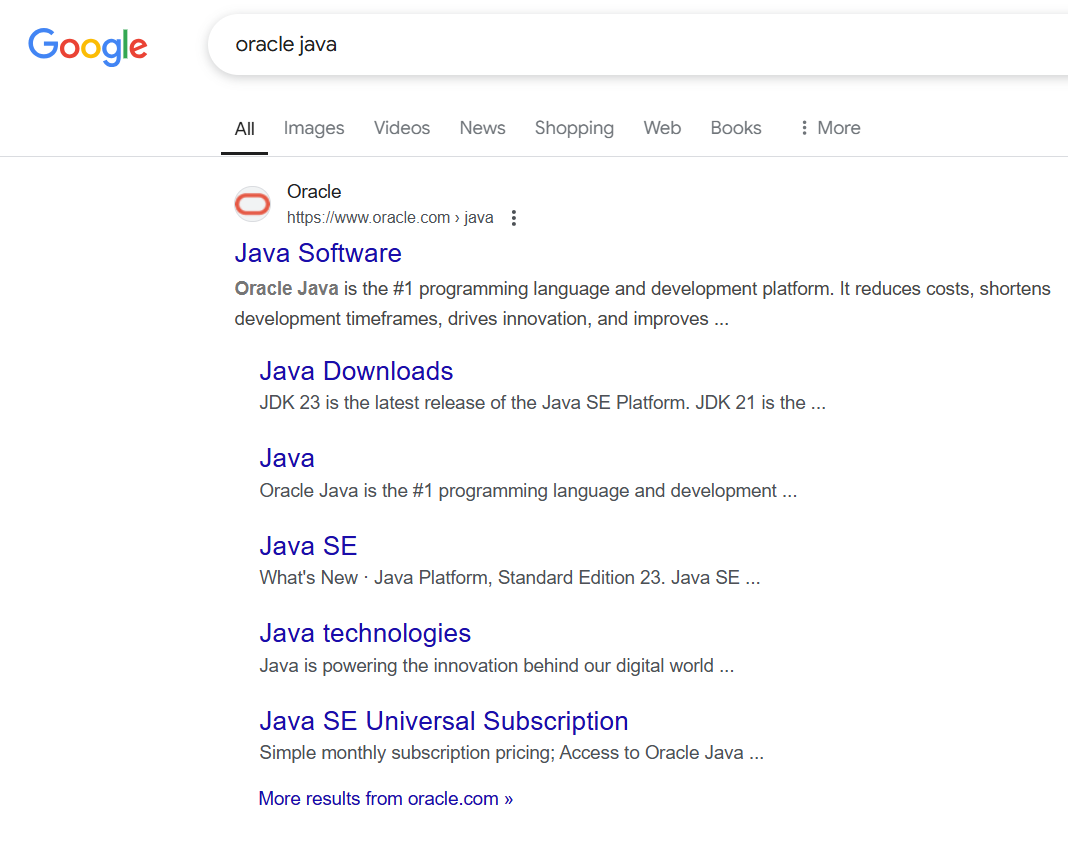
|  |  |  |  |
| --- | --- | --- | --- |
| **Submitted by** | | **Submitted to** | |
| **NAME:** | **K . SAI YASWANTH** | **NAME:** | **RAJ KUMAR BATCHU** |
| **ROLL NO:** | **AV.SC.U4CSE24144** | **DEPARTMENT:** | **CSE** |
| **SECTION:** | **CSE-B** | **DESIGNATION:** | **Asst.Professor** |

|  |  |
| --- | --- |
| **MARKS:** |  |
| **SIGNATURE:** |  |
| **DATE:** |  |

WEEK-1

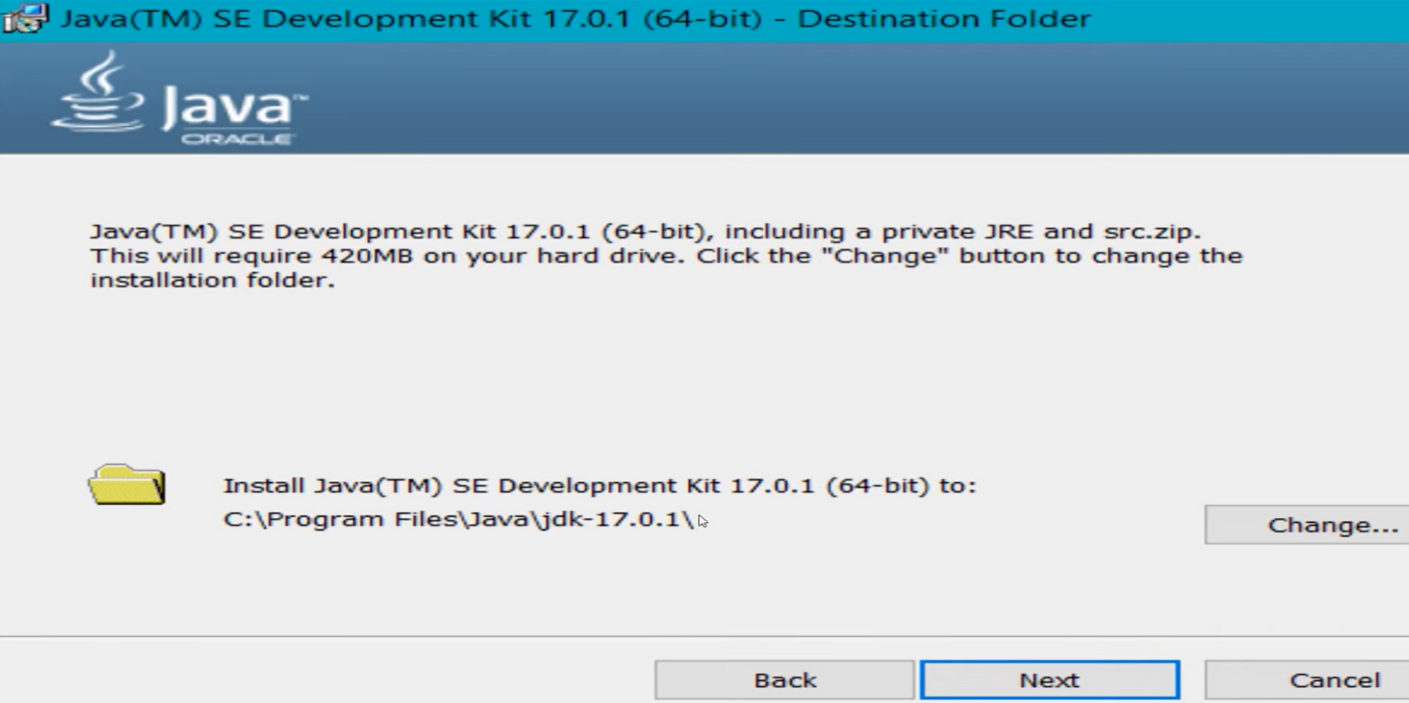
1. Aim : Explain the process of installing JDK (JAVA DEVELOPMENT KIT).

* Open “Oracle” in your web browser.
* Click on “ download Java ” in the oracle interface.
* Select the version you want to download (version JDK21 is best).
* Select the Operating System (OS) of your PC and click on the link “x64 installer” to download .
* **Installation:**
* **Then click on java downloads**

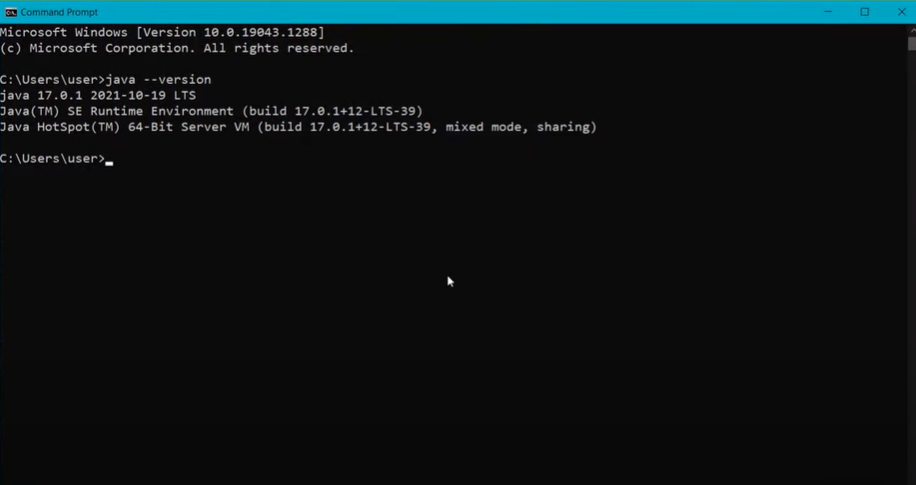
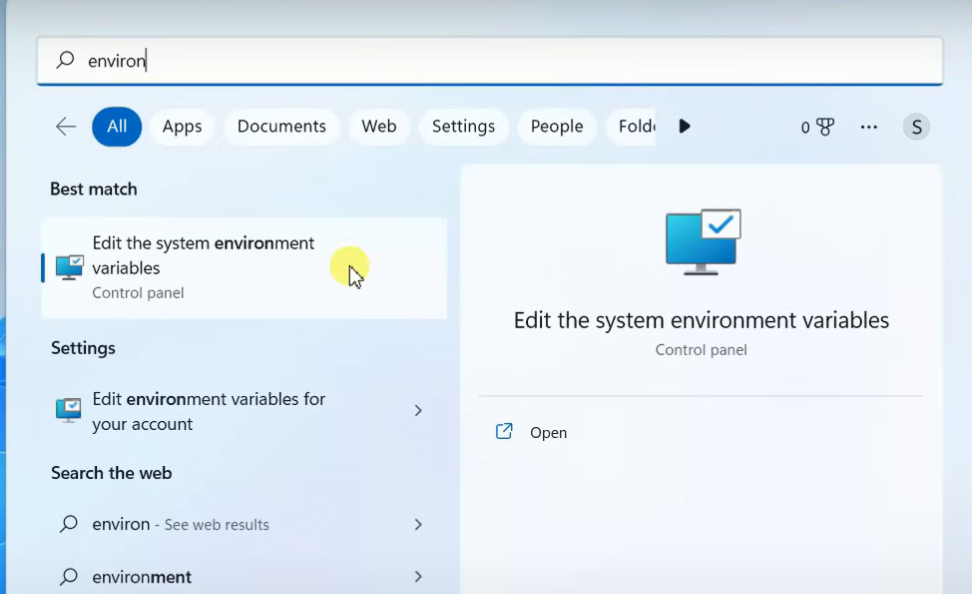
****

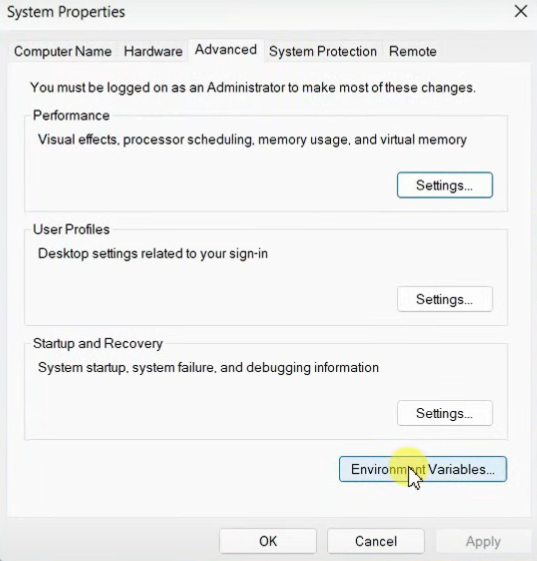
* **Click on next**

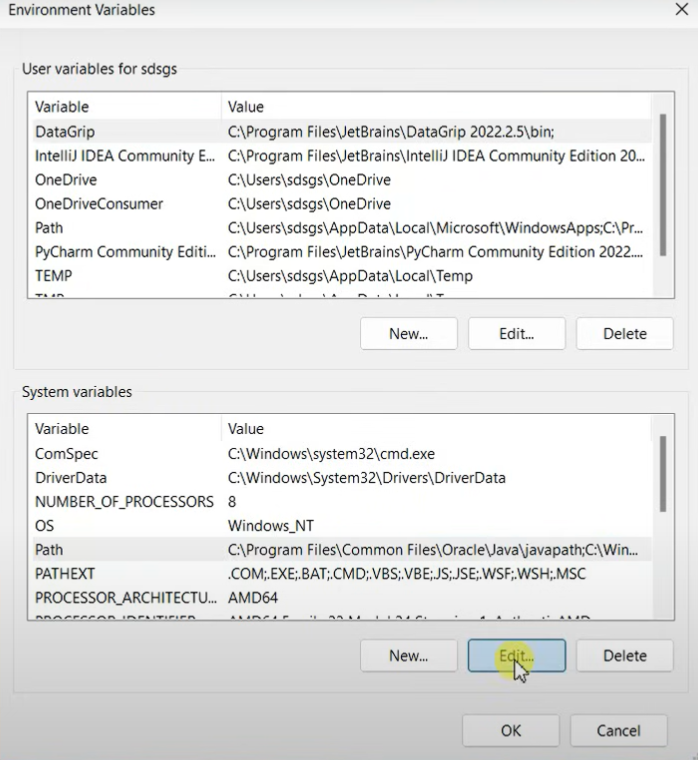
****

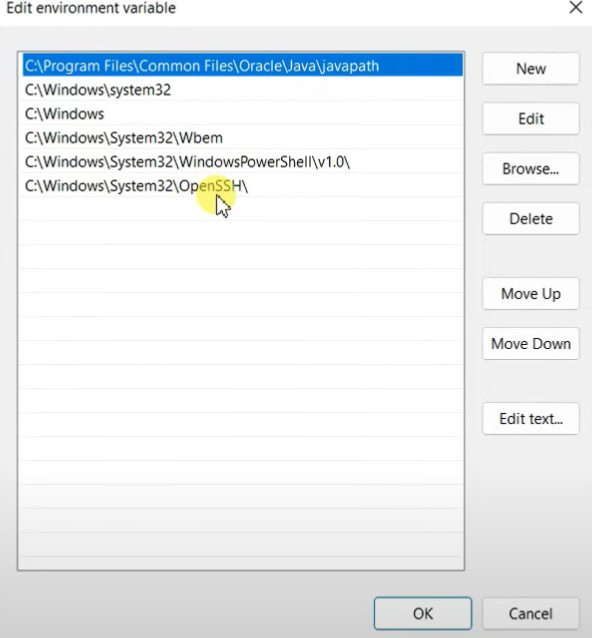
* ****Click on next
* **Click on close**



* To check whether java is downloaded correctly or not.
* Open command window and type “java --version**” .**
* To set the path:
* Go to “start” and se “Edit the search environment variables”.
* ****Click on open and continue to click on Environment variables.

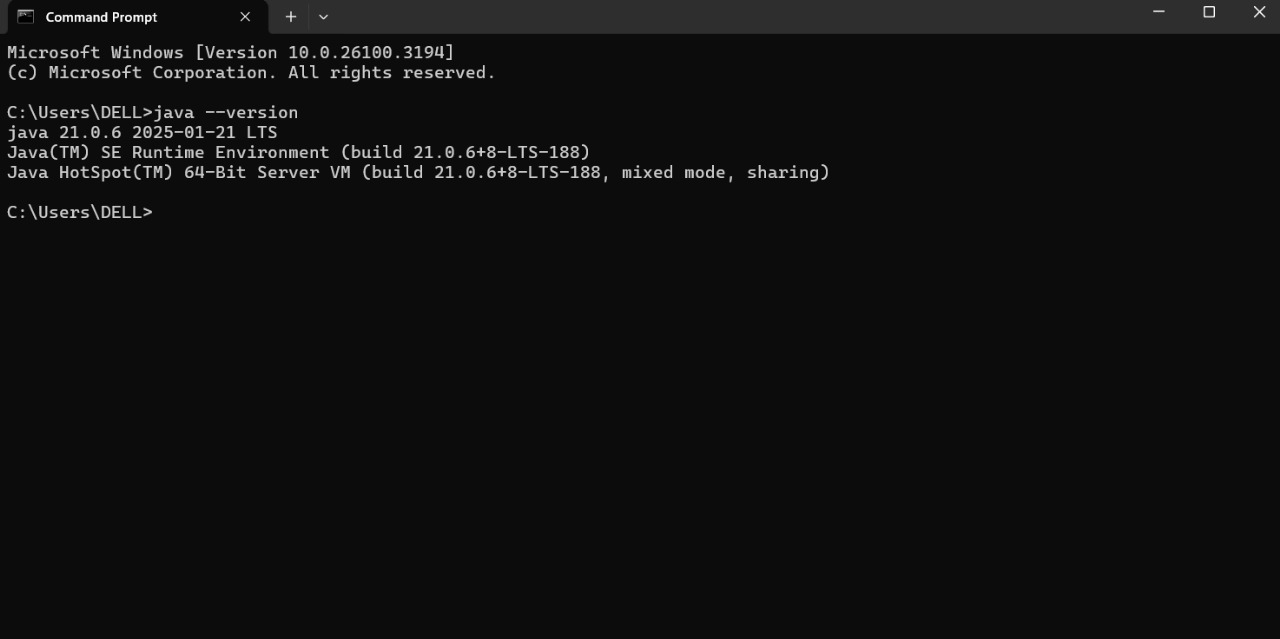
****

* Select the “path” option in “system variables”.
* **Now click on new paste the path of java**

****

* To check whether the path is properly set or not

Open command window and type “javac --version”

If it is showing the exact version of java you are installed then the path is set properly.

* Now create a text document with “.java” extension and open in notepad ,

To write code.

**2) Aim: Write your code in notepad and execute in command prompt.**

**Code:**

class Student\_id

{

public static void main(String[] args) {

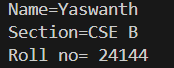
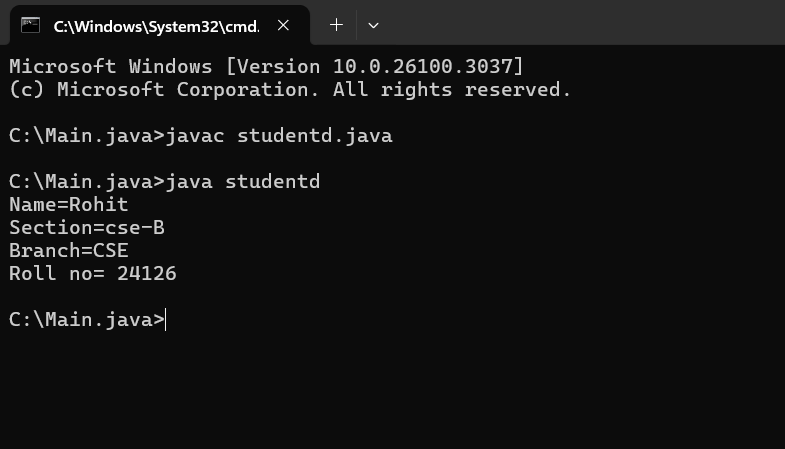
System.out.println("Name= Yaswanth");

System.out.println("Section=CSE B");

System.out.println("Roll no= 24144")

}

}

**** **OUTPUT:**

**ERROR:**

|  |  |  |
| --- | --- | --- |
| **S.no** | **Error** | **Rectification** |
| **1)** | **error: ‘;’ expected**  System.out.println("Roll no= 24144")  **^** | **In the code,semicolon must be added at the end of line.** |

**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 MyFirst then “class MyFirst { “ should be written”}**

WEEK-2

**1)Aim: Write a java program for SI?**

**import java.util.Scanner;**

**class largest\_number{**

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

**Scanner l=new Scanner(System.in);**

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

**int a=l.nextInt();**

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

**int b=l.nextInt();**

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

**int c=l.nextInt();**

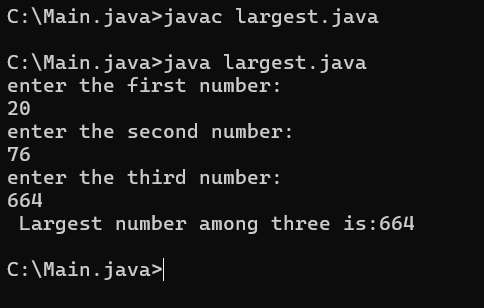
**int largest\_number = ((a > b && a > c)? a: (b > a && b>c)?b :c);**

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

**}**

**}**

**OUTPUT:**

****

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

**2)Aim: Write a program in java for factorial of a number ?**

**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++){**

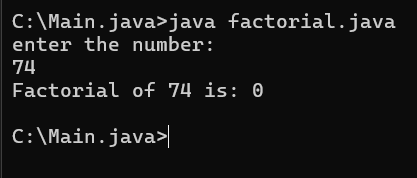
**factorial \*= i;**

**}**

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

**}**

**}**

**OUTPUT:**

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

**3)Aim:Write a program in java for fibonacci series ?**

**import java.util.Scanner;**

**class Fibonacci {**

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

**Scanner f=new Scanner(System.in);**

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

**int n=f.nextInt();**

**int firstTerm = 0, secondTerm = 1;**

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

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

**System.out.print(firstTerm + " ")**

**int nextTerm = firstTerm + secondTerm;**

**firstTerm = secondTerm;**

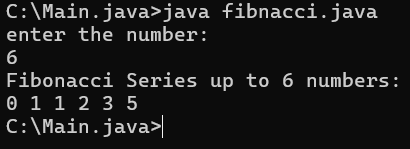
**secondTerm = nextTerm;**

**}**

**}**

**}**

**OUTPUT:**

****

|  |  |  |
| --- | --- | --- |
| code | error | rectification |
|  |  |  |

**4) a) Aim:Write a program in java for converting temperature from celsius to Fahrenheit ?**

**import java.util.Scanner;**

**class Temperature\_conversion{**

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

**Scanner input=new Scanner(System.in);**

**System.out.println("enter the Celcius :");**

**float ctemp=input.nextFloat();**

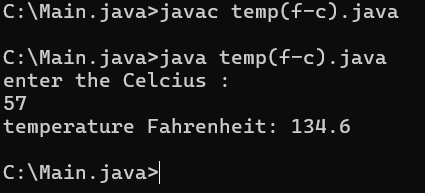
**float ftemp;**

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

**System.out.println("temperature Fahrenheit: " + ftemp);**

**}**

**}**

**OUTPUT:**

|  |  |  |
| --- | --- | --- |
| code | error | rectification |
|  |  |  |

**4)b) Aim:Write a program in java for converting temperature from fahrenheit to Celsius ?**

**import java.util.Scanner;**

**class temperature\_conver{**

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

**Scanner input=new Scanner(System.in);**

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

**float ftemp=input.nextFloat();**

**float ctemp;**

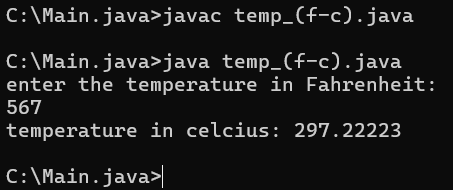
**ctemp=(ftemp-32)\*5/9;**

**System.out.println("temperature in celcius: " +ctemp);**

**}**

**}**

**OUTPUT:**

****

|  |  |  |
| --- | --- | --- |
| code | error | rectification |
|  |  |  |

**5)Aim: Write a program in java for area of rectangle ?**

**import java .util. Scanner;**

**class area\_rectangle{**

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

**Scanner a=new Scanner(System.in);**

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

**float l=a.nextFloat();**

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

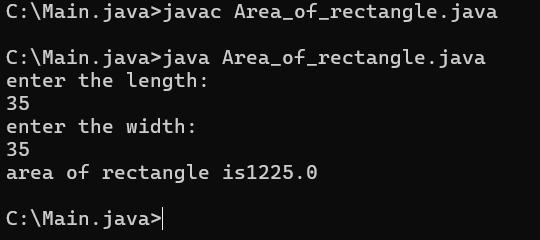
**float w=a.nextFloat();**

**float area=l\*w;**

**System.out.println("area of rectangle is" + area);**

**}**

**}**

**OUTPUT:**

|  |  |  |
| --- | --- | --- |
| code | error | rectification |
|  |  |  |

**6)Aim: Write a java program for SI?**

**import java.util.Scanner;**

**class SI{**

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

**Scanner input=new Scanner(System.in);**

**System.out.print("enter the principal amount");**

**float p=input.nextFloat();**

**System.out.print("rate of the intrest ");**

**float R=input.nextFloat();**

**System.out.print("enter time in years");**

**float T=input.nextFloat();**

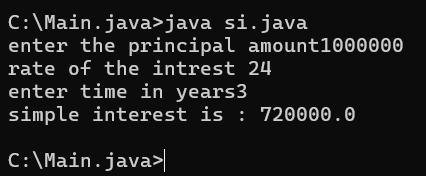
**float SI;**

**SI=(p\*T\*R)/100;**

**System.out.println("simple interest is : "+SI);**

**}**

**}**

**OUTPUT:**

|  |  |  |
| --- | --- | --- |
| code | error | rectification |
|  |  |  |

**7)Aim:Write a program in java for area of triangle using heron’s formula ?**

**import java.util.Scanner;**

**class heronsformula{**

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

**Scanner input=new Scanner(System.in);**

**System.out.println("enter the value for a :");**

**Double a=input.nextDouble();**

**System.out.println("enter the value for b :");**

**Double b=input.nextDouble();**

**System.out.println("enter the value for c :");**

**Double c=input.nextDouble();**

**Double s=(a+b+c)/2;**

**System.out.println("S is the value of semi perimeter"+s);**

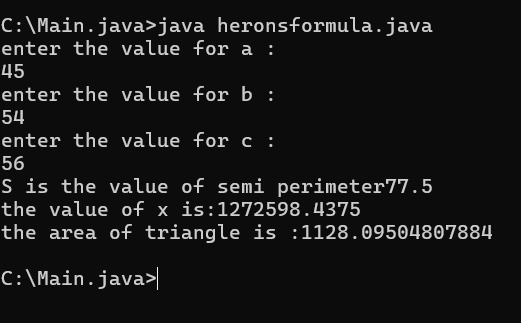
**Double x=s\*(s-a)\*(s-b)\*(s-c);**

**System.out.println("the value of x is:"+x);**

**Double area=Math.sqrt(x);**

**System.out.println("the area of triangle is :"+area);**

**}**

**OUTPUT:**

|  |  |  |
| --- | --- | --- |
| code | error | rectification |
|  |  |  |

**WEEK-3**

1. **Aim:Create the java program with the following instructions  ?**
2. **Create a class with name Car .**
3. **Create 4 attributes named Car\_Color , Car\_brand, fuel\_type, mileage .**
4. **Create 3 method named Start( ) , Stop( ),  Service( ) .**
5. **Create 3 objects Car1 ,  Car2 , Car3  .**
6. **Create a constructor which should print “Welcome to Car Garage” .**

**Code: class Car{**

**public String carColor;**

**private String carBrand;**

**private String fuelType;**

**public int mileage;**

**Car(String carColor , String carBrand , String fuelType , int mileage){**

**this.carColor = carColor;**

**this.carBrand = carBrand;**

**this.fuelType = fuelType;**

**this.mileage = mileage;**

**System.out.println(carColor + " " + carBrand + " " + fuelType + " " + mileage);**

**}**

**public void Start(){**

**System.out.println("The car has just started");**

**}**

**public void Stop(){**

**System.out.println("The car has just stopped");**

**}**

**public void Service(){**

**System.out.println("The car is in good condition");**

**}**

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

**Car Car1 = new Car("Black","Hyundai","Petrol",100);**

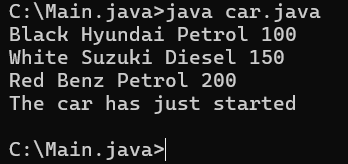
**Car Car2 = new Car("White","Suzuki","Diesel",150);**

**Car Car3 = new Car("Red","Benz","Petrol",200);**

**Car1.Start();**

**}**

**}**

** OUTPUT:**

**Error:**

|  |  |  |
| --- | --- | --- |
| **S.no** | **Expected Error** | **Reason** |
| **1** | **}** | **} is expected at end of the calass** |
| **2** | **Setting the parameters inside  the constructer** | **Without setting the constructor we cannot pass the values** |

**Class Diagram:**

|  |
| --- |
| **Car** |
| **( + )carColor : String**  **( - )carBrand : String**  **( - )fuelType : String**  **( + )mileage : int** |
| **( + )Car( ) : void**  **( + ) Start( ) : void**  **( + ) Stop( ) : void**  **( + ) Service( ) : void** |

**2)Aim:** **Write a java program to create a class BackAccount with two methods deposit( ) and withdraw( ) ?**

1. **In deposit( ) whenever an amount is deposited it has to be updated with current amount**
2. **In withdraw( ) whenever an amount is withdrawn it has to be less than current amount else print “Insufficient funds”.**

**CODE:**

**class BankAccount{**

**private String Name;**

**private int AccNo, CurrBal ;**

**BankAccount(String Name, int AccNo, int CurrBal){**

**this.Name = Name;**

**this.AccNo = AccNo;**

**this.CurrBal = CurrBal;**

**System.out.println("The customers are : " + this.Name + " ");**

**}**

**public int deposit(int dAmt){**

**CurrBal = CurrBal + dAmt ;**

**return CurrBal;**

**}**

**public void withdraw(int wAmount){**

**if(wAmount < CurrBal){**

**CurrBal = CurrBal - wAmount ;**

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

**}**

**else{**

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

**} }**

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

**BankAccount tony = new BankAccount("tony",1500,10000);**

**tony.withdraw(25000);**

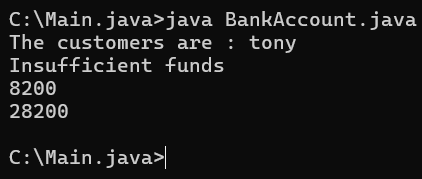
**tony.withdraw(1800);**

**int FinalAmount =tony.deposit(20000);**

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

**} }**

**OUTPUT:**

****

**Error:**

|  |  |  |
| --- | --- | --- |
| **S.no** | **Expected Errors** | **Reason** |
| **1** | **Giving the parameters inside the constuctor** | **We cannot pass the values inside the constructor without setting first** |
| **2** | **}** | **} is sometimes missing at the end of class** |

**Class Diagram:**

|  |
| --- |
| **BankAccount** |
| **( -)Name : String**  **( -)AccNo :String**  **( -)CurrlBal :String** |
| **(+)Bank Account() :  void**  **(+)deposit() :int**  **(+)withdraw() : void** |

**WEEK-4**

**PROGRAM-1:**

**AIM:** 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.

**CODE:**

class myclass{

static int count=0;

    final double pi=3.1415;

    void myclass(){

        count=count+1;

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

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

    }}

class details{

           public static void main(String[] args) {

 System.out.println("NAME: K.Sai Yaswanth ,ROLL NO :AV.SC.U4CSE24144,SEC:CSE-B");

        myclass obj1=new myclass();

        myclass obj2=new myclass();

        myclass obj3=new myclass();

        myclass obj4=new myclass();

        myclass obj5=new myclass();

        obj1.myclass();

        obj2.myclass();

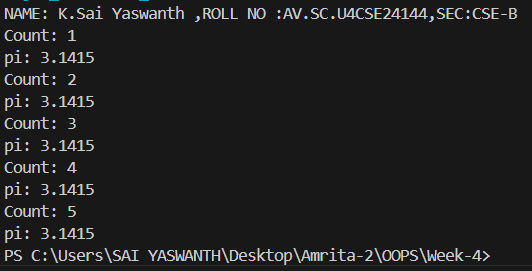
        obj3.myclass();

        obj4.myclass();

        obj5.myclass();

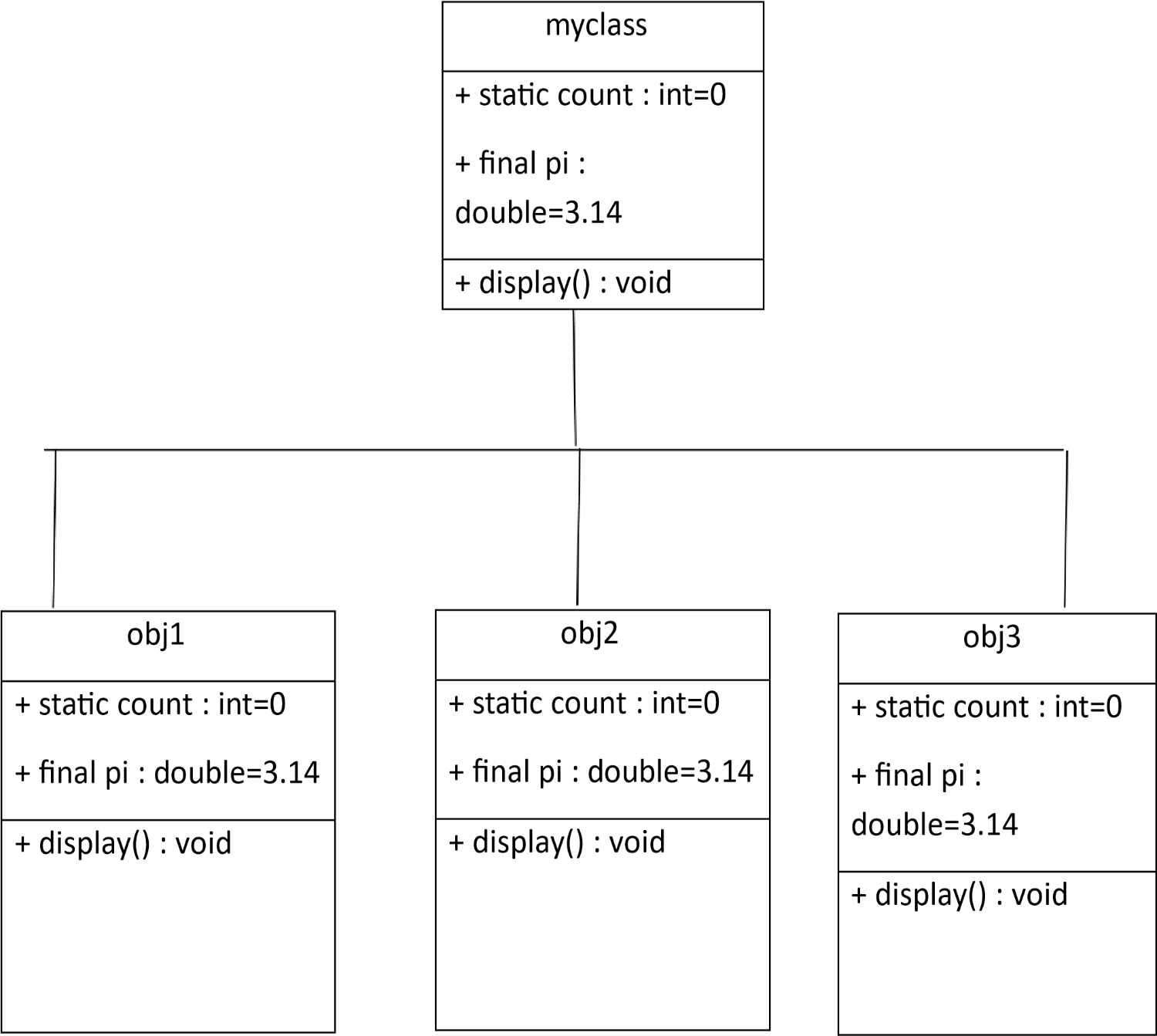
      }}

**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** |



**PROGRAM-2:**

**AIM**: 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 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.

**CODE:**

class Book {

String title;

String author;

int yop;

Book(String title, String author, int yop) {

this.title = title;

this.author = author;

this.yop = yop;

}

void display() {

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

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

System.out.println("Year of Publication: " + yop);

}

}

class Details {

public static void main(String[] args) {

 System.out.println("NAME: K.Sai Yaswanth ,ROLL NO :AV.SC.U4CSE24144,SEC:CSE-B");

Book b1 = new Book("the kill a mocking bird", "Amul", 2005);

Book b2 = new Book("The alchemist", "Madam", 2024);

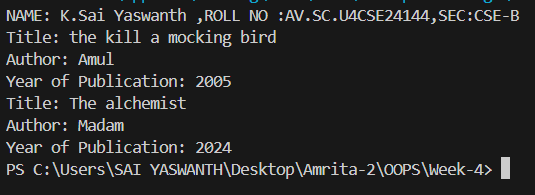
b1.display();

b2.display();

}

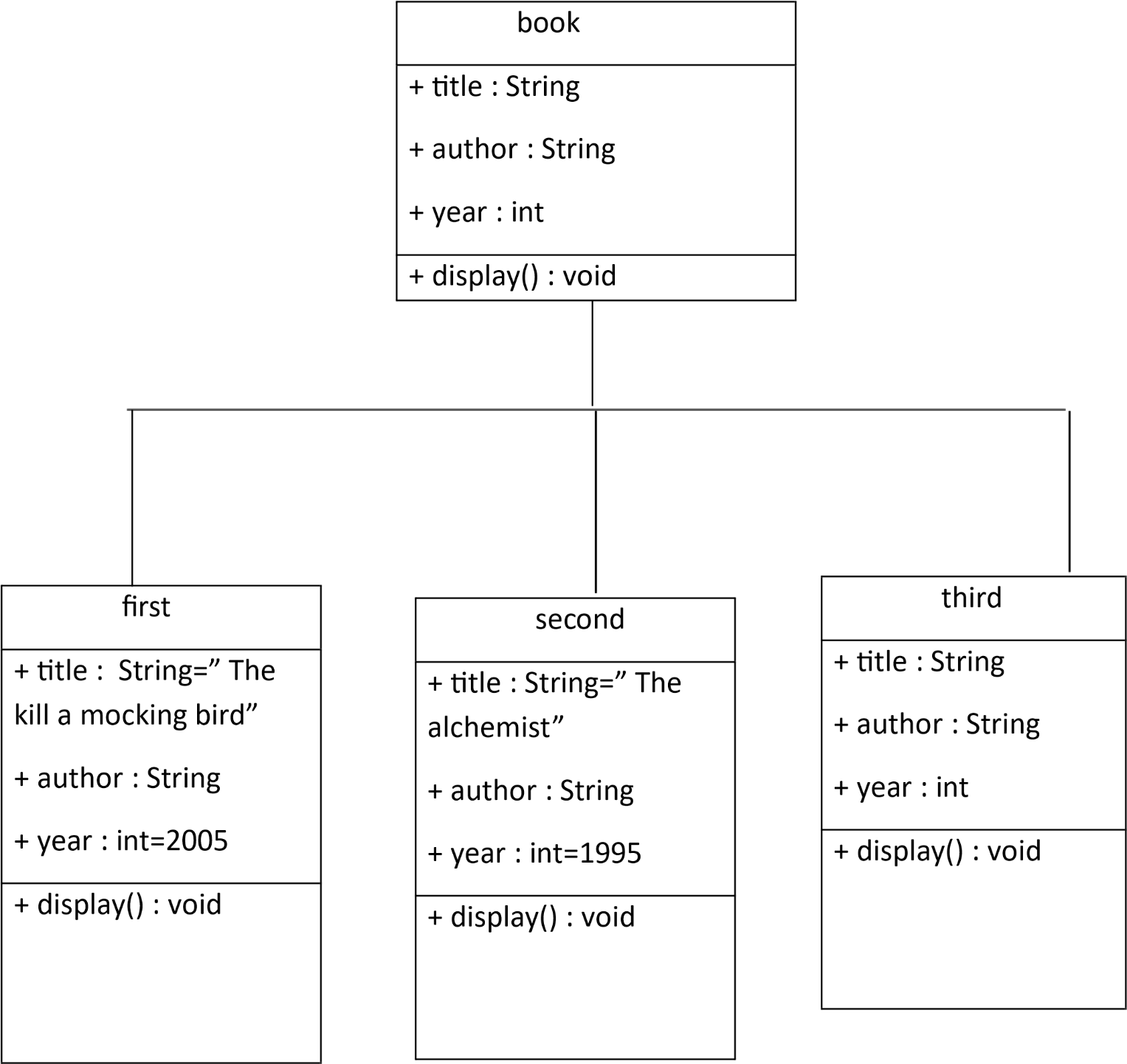
}

**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 |

**CLASS DIAGRAM:**

**WEEK-5**

**PROGRAM-1:**

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

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

**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 Calculator {

    double num1, num2;

    public calc(double num1, double num2) {

        this.num1 = num1;

        this.num2 = num2;

    }

    static class Addition extends calc {

        public Addition(double num1, double num2) {

            super(num1, num2);

        }

        public double add() {

            return num1 + num2;

        }

    }

    static class Subtraction extends Addition {

        public Subtraction(double num1, double num2) {

            super(num1, num2);

        }

        public double subtract() {

            return num1 - num2;

        }

    }

    static class Multiplication extends Subtraction {

        public Multiplication(double num1, double num2) {

            super(num1, num2);

        }

        public double multiply() {

            return num1 \* num2;

        }

    }

    static class Division extends Multiplication {

        public Division(double num1, double num2) {

            super(num1, num2);

        }

        public String divide() {

            if (num2 != 0) {

                return String.valueOf(num1 / num2);

            } else {

                return "Error: Division by zero";

            }

        }

    }

    public static void main(String[] args) {

        Addition add1 = new Addition(12, 6);

        Subtraction subtract1 = new Subtraction(12, 6);

        Multiplication multiply1 = new Multiplication(12, 6);

        Division divide1 = new Division(12, 6);

System.out.println("NAME: K.Sai Yaswanth ,ROLL NO :AV.SC.U4CSE24144,SEC:CSE-B");

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

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

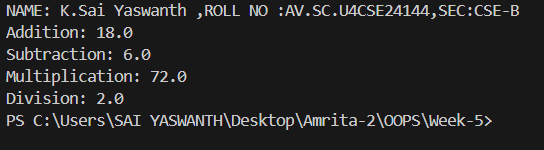
        System.out.println("Multiplication: " + multiply1.multiply());

        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. |

**PROGRAM-2:**

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

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

**Question:**

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

**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 |
| + int (brand, speed)  + start\_vehicle()  + display\_details() |

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

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

**CODE:**

class Vehicle {

    private String brand;

    private int speed;

    Vehicle(String brand, int speed) {

        this.brand = brand;

        this.speed = speed;

    }void details() {

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

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

    }

}

class Car extends Vehicle {

    private int doors;

    private int capacity;

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

        super(brand, speed);

        this.doors = doors;

        this.capacity = capacity;

    }

    void carDetails() {

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

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

    }

    @Override

    void details() {

        super.details();

        carDetails();

    }

}

class Bike extends Vehicle {

    private boolean gears;

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

        super(brand, speed);

        this.gears = gears;

    }

    void bikeDetails() {

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

    }

    @Override

    void details() {

        super.details();

        bikeDetails();

    }

}

class Truck extends Vehicle {

    private int tons;

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

        super(brand, speed);

        this.tons = tons;

    }

    void truckDetails() {

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

    @Override

    void details() {

        super.details();

        truckDetails();

    }

}

public class Rent {

    public static void main(String[] args) {

System.out.println("NAME: K.Sai Yaswanth ,ROLL NO :AV.SC.U4CSE24144,SEC:CSE-B");

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

        c.details();

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

        b.details();

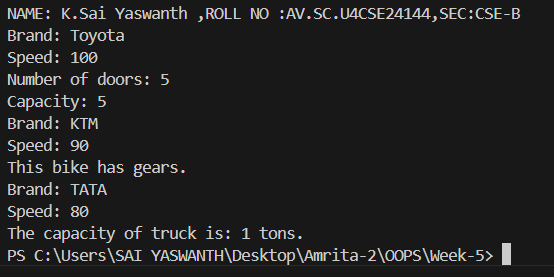
        Truck t = new Truck("TATA", 80, 1);

        t.details();

    }

}

**OUTPUT:**



**ERRORS:**

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

**WEEK-6**

**PROGRAM-1:**

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

**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);

    }

}

class Main {

    public static void main(String[] args) {

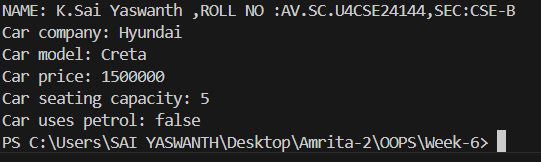
System.out.println("NAME: K.Sai Yaswanth ,ROLL NO :AV.SC.U4CSE24144,SEC:CSE-B");

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

        c1.displayInfo();

}}

**OUTPUT:**

****

**ERRORS:**

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

**IMPORTANT POINTS:**

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

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

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

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

**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;

    University6(double percentile) {

        this.percentile = percentile;

    }

    void ug() {

        if (percentile >= 60) {

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

        } else {

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

        }

    }

    void pg() {

        if (percentile >= 70) {

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

        } else {

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

        }

    }

}

class Eli extends University6 {

        Eli(double percentile) {

        super(percentile);

    }

    @Override

    void ug() {

        if (percentile >= 60) {

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

        } else {

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

        }

    }

    @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) {

        Eli e1 = new Eli(60.7);

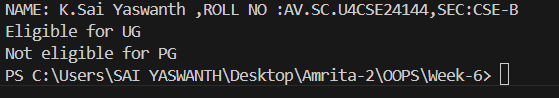
        e1.ug();

        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 Calculator {

    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) {

        Calculator c1 = new Calculator();

        c1.add(2, 7, 3);

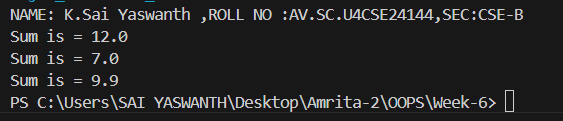
        c1.add(3,4);

        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();

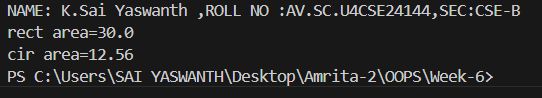
    s1.calarea(5,6);

    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**

**PROGRAM-1:**

**AIM:** Write a java program to create an abstract class animal with an abstract method called sound create a subclass lion and tiger that extend the animal class and implement the sound method to make a specific sound for each animal.

**CODE:**

abstract class Animal {

abstract void sound();

}

class Tiger extends Animal{

public void sound(){

System.out.println("Tiger growls");

}

}

class Lion extends Tiger{

public void sound(){

System.out.println("Lion roars");

}

}

public class Main{

public static void main(String[] args){

System.out.println("Name:Yaswanth"+" "+"Roll No:24144"+" "+"Section:CSEB");

Lion l = new Lion();

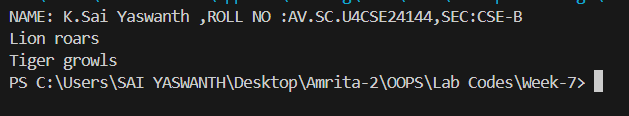
Tiger t = new Tiger();

l.sound();

t.sound();

}

}

**OUTPUT**

**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **Error Name** | **Error Rectification** |
| **1** | Main Class | Better to create main class name same as the file you saved and first letter is capital. |
| **2** | Method | We need to provide return type to the method. |
| **3** | Data type | As per need provide data type |
| **4** | Abstract method | Implementation in subclass only |

**IMPORTANT POINTS:**

1)Here we used abstract class concept it is a restricted class that cannot be instantiated (cannot have objects created directly) and is typically designed to be extended by subclasses.

2)An abstract method is a method declared in an abstract class that does not have an implementation, meaning it doesn't have a body within the abstract class.

**PROGRAM-2:**

**AIM:**Write a java program to create an abstract class Shape3D with abstract method calculate volume ( ) and calculate surfacearea( ).Create subclasses Sphere and Cube that extends the Shape3D class and implement the respective method to calculate the volume and surface area of each shape.

**CODE:**

abstract class Shape3D {

abstract double calculateVolume();

abstract double calculateSurfacearea();

}

class Sphere extends Shape3D{

int radius;

Sphere(int radius){

this.radius=radius;

}

public double calculateVolume(){

return (4.0/3.0)\*Math.PI\*Math.pow(radius,3);

}

public double calculateSurfacearea(){

return 4.0\*Math.PI\*Math.pow(radius,2);

}

}

class Cube extends Shape3D{

int side;

Cube(int side){

this.side=side;

}

public double calculateVolume(){

return Math.pow(side,3);

}

public double calculateSurfacearea(){

return 6\*Math.pow(side,2);

}

}

public class Hello{

public static void main(String[] args) {

System.out.println("Name:Yaswanth"+" "+"Roll No:24144"+" "+"Section:CSEB");

Sphere sp=new Sphere(4);

System.out.println("Volume of sphere is: "+sp.calculateVolume());

System.out.println("Surface area of sphere is: "+sp.calculateSurfacearea());

Cube cb=new Cube(5);

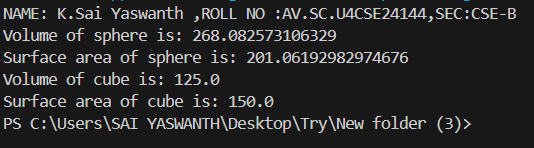
System.out.println("Volume of cube is: "+cb.calculateVolume());

System.out.println("Surface area of cube is: "+cb.calculateSurfacearea());

}

}

**OUTPUT:**

****

**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **Error Name** | **Error Rectification** |
| **1** | Main Class | Better to create main class name same as the file you saved and first letter is capital. |
| **2** | Data type | As per need provide data type |
| **3** | Abstract method | Implementation in subclass only |

**IMPORTANT POINTS:**

1.Here we used abstract class concept it is a restricted class that cannot be instantiated (cannot have objects created directly) and is typically designed to be extended by subclasses.

2.An abstract method is a method declared in an abstract class that does not have an implementation, meaning it doesn't have a body within the abstract class.

**PROGRAM-3:**

**AIM:** write a java program using an abstract class to define a method for pattern printing.

Create an abstract class named pattern printer with an abstract method print pattern (int n) and a concrete method to display the pattern title

Implement two subclass:

* + - * 1. Star pattern: Print a right-angle triangle of stars(\*) .
        2. Number pattern : Print a right-angled triangle of increasing numbers.

In the main ()method, create objects of both subclass and print the pattern of given number of rows.

**CODE:**

abstract class PatternPrinter {

  abstract void printPattern(int n);

  void displayTitle(String title) {

      System.out.println("" + title);

      System.out.println("");

  }

}

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 number = 1;

      for (int i = 1; i <= n; i++) {

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

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

              number++;

          }

          System.out.println();

      }

  }

}

class Main {

  public static void main(String[] args) {

    System.out.println("NAME: K.Sai Yaswanth , ROLL NO:24144,SEC:CSE-B ");

      int rows = 5;

      StarPattern star = new StarPattern();

      star.displayTitle("Star Pattern");

      star.printPattern(rows);

      NumberPattern number = new NumberPattern();

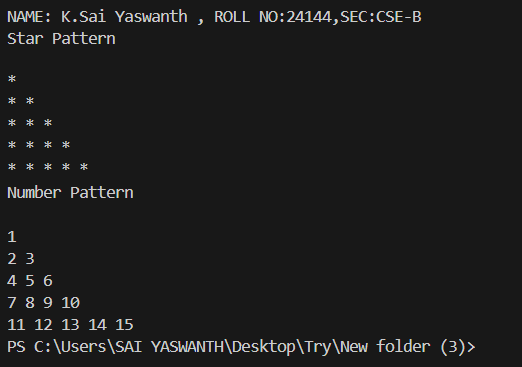
      number.displayTitle("Number Pattern");

      number.printPattern(rows);

  }

}

**OUTPUT:**

****

**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **Error Name** | **Error Rectification** |
| **1** | Main Class | Better to create main class name same as the file you saved and first letter is capital. |
| **2** | Data type | As per need provide data type |
| **3** | Syntax in for | Initializing value and condition should be correct |
| **4** | overridding | Same method names |

**IMPORTANT POINTS:**

1.Here we used nested for loop concept the block of code is executed until the condition is false.

2.Here we used abstract class concept it is a restricted class that cannot be instantiated (cannot have objects created directly) and is typically designed to be extended by subclasses.

**WEEK-8**

**PROGRAM-1:**

**AIM:** Write a java program to create an interface shape with the getperimeter() method.Create 3classes rectangle,circle and triangle that implements shape interface.Implement getperimeter() method for each of 3classes.

**CODE:**

class Shapes {

    private double perimeter;

    public double getperimeter(){

        return perimeter;

    }

    public void setperimeter(double perimeter){

        this.perimeter=perimeter;

    }

}

class rectangle extends Shapes {

    private double length;

    private double breadth;

    rectangle(double length , double breadth){

        this.length =length;

        this.breadth=breadth;

     setperimeter(2 \* (length + breadth)) ;

     }

    }

class circle extends Shapes {

    private double radius;

    circle(double radius){

this.radius=radius;

setperimeter(2 \* 3.14 \* radius) ;

}}

class triangle extends Shapes {

    private double a;

    private double b;

    private double c;

    triangle(double a,double b , double c){

this.a=a;

this.b=b;

this.c=c;

setperimeter(a+b+c);

}}

class main {

public static void main(String[] args) {

System.out.println("NAME: K.Sai Yaswanth ,ROLL NO :AV.SC.U4CSE24144,SEC:CSE-B");

rectangle r = new rectangle(5,3);

System.out.println("perimeter of rectangle is : " + r.getperimeter());

circle c = new circle (5);

System.out.println("perimeter of circle is : " + c.getperimeter());

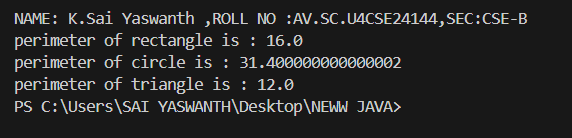
triangle t = new triangle (3,5,4);

System.out.println("perimeter of triangle is : " + t.getperimeter());

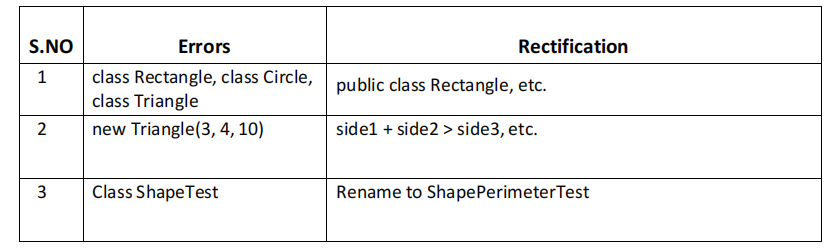
    }

}

**OUTPUT:**

****

**ERRORS:**



**IMPORTANT POINTS:**

1.Interface Shape:

a.The interface Shape contains one method getPerimeter(), which will be implemented by any class that implements this interface.

2.Rectangle Class:

a. The Rectangle class implements the Shape interface.

b.The perimeter of a rectangle is calculated using the formula:   
 Perimeter = 2 \* (length + width)

c.The getPerimeter() method returns the perimeter of the rectangle.

3.Circle Class:

a.The Circle class implements the Shape interface.

b.The perimeter (circumference) of a circle is calculated using the formula:   
Circumference = 2 \* π \* radius

c.The getPerimeter() method returns the circumference of the circle.

4.Triangle Class:

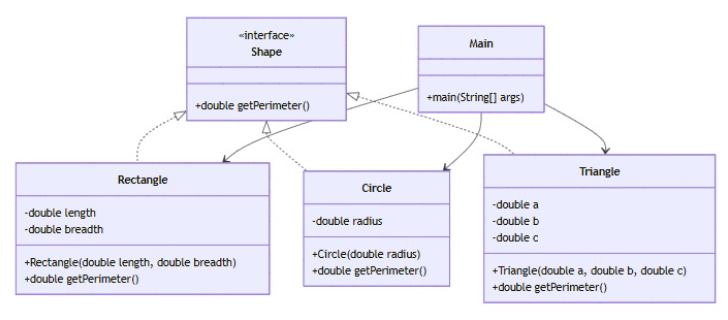
a.The Triangle class implements the Shape interface.

b.The perimeter of a triangle is the sum of its three sides:   
Perimeter = side1 + side2 + side3

c.The getPerimeter() method returns the perimeter of the triangle.

5.Main Class:In the main method, we create objects for Rectangle, Circle, and Triangle and call their respective getPerimeter() methods to display the perimeters.

**CLASS DIAGRAM:**



**PROGRAM-2:**

**AIM:** Write a java program to create an interface label with method play that takes no arguments and return void create 3 classes football,volley ball and basketball that implements interface and override play() method to play respective sports.

**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 main {

    public static void main(String[] args) {

        System.out.println("NAME:HARI SIVA SAI ROHIT ,ROLL NO:24126,SEC:CSE-B");

        Playable football = new Football();

        Playable volleyball = new Volleyball();

        Playable basketball = new Basketball();

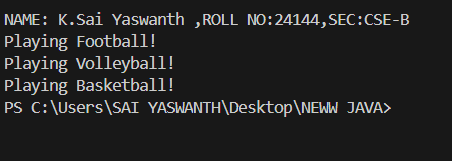
        football.play();

        volleyball.play();

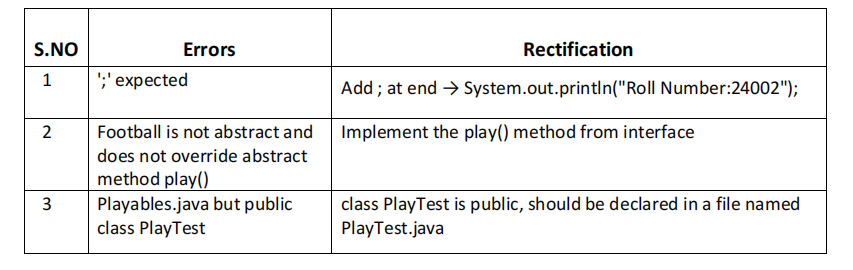
        basketball.play();

    }}

**OUTPUT:**

****

**ERRORS:**



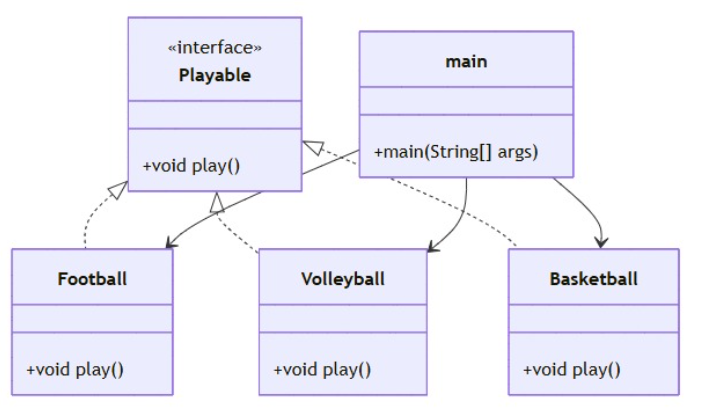
**IMPORTANT POINTS:**

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.

**CLASS DIAGRAM:**



**PROGRAM-3:**

**AIM:** Write a java program to implement a login system using interfaces.

**CODE:**

interface LoginSystem{

boolean login(String id,String password);

}

class StudentPortal implements LoginSystem{

public boolean login(String id, String password){

if(id.equals("Tony123")&&password.equals("Tony@123")){

System.out.println("Login Successful");

return true;

}

else{

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

return false;

}

}

}

class Main{

public static void main(String[] args){

System.out.println("NAME: K.Sai Yaswanth , ROLL NO:24144,SEC:CSE-B ");

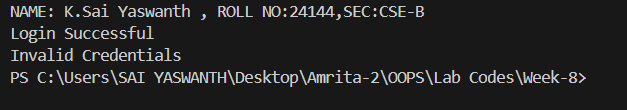
StudentPortal portal = new StudentPortal();

portal.login("Tony123","Tony@123");

portal.login("yash123","yash@286");

}

}

**OUTPUT:**

**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Error** | **Error rectification** |
| **1** | Login\_System() not  overridden in University classes. | Implement Login\_System() in  university class using @override annotation |
| **2** | Error: ”;” expected | Giving the ‘;’ in wanted places |

**IMPORTANT POINTS:**

1.Boolean type is used in many applications when we need to check the input is true or false.

**CLASS DIAGRAM:**

