23CSE111

OBJECT ORIENTED PROGRAMMING LANGUAGE

LAB MANUAL



Amrita School of Engineering

Amrita Vishwa Vidyapeetham, Amaravathi

Name : P.Prashanth Kumar

Roll.No : AV.SC.U4CSE24238

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SNo** | **Title** | **Date** | **Page No** | **Signature** |
| **WEEK 1** |  |  |  |  |
| 1 | Download and Install Java Software |  |  |  |
| 2 | Write a Java program to print the message “HELLO WORLD.” |  |  |  |
| 3 | Write a java code to print student details. |  |  |  |
| **WEEK 2** |  |  |  |  |
| 1 | Write a Java program to calculate fibonacci of a number. |  |  |  |
| 2 | Write a Java program to calculate factorial of a number. |  |  |  |
| 3 | Write a Java program to calculate Simple interest. |  |  |  |
| 4 | Write a Java program to convert temperature from Celsius to Fahrenheit and vice versa. |  |  |  |
| 5 | Write a java program to calculate the area of the triangle and rectangle |  |  |  |
| **WEEK 3** |  |  |  |  |
| 1 | Write a JAVA Code to create java program with following instructions  1.Create a class with name car  2. Create four attributes named car\_color ,Car\_brand,fuel\_type,mileage  3. Create three methods named start(), stop(). Service()  4. Create three objects named car1,car2 and car3 |  |  |  |
| 2 | Write a JAVA Code to create a class bankAccount with methods deposit() and withdrawl |  |  |  |
| **WEEK 4** |  |  |  |  |
| 1 | Write a java program with class named book. A class should contain various attributes such as title, author, year of publication. It should also contain a constructor with parameters which initializes title, author and year of publication. Create a method which displays the details of the book. Display the details of two books. |  |  |  |
| 2 | Create a java program with class name “myclass” with static variable count of int type. Initialize it to zero and a constant variable pi of type double. Initialize pi to 3.14 as attributes of class. Now define a constructor for “myclass” that increments the count variable each time an object of “myclass” is created. Finally print the final values of count and pi variables. Create three objects. |  |  |  |
| **WEEK 5** |  |  |  |  |
| 1 | Create a calculator using the operations including addition, subtraction ,multiplication and division using multilevel inheritance and display the desired output. |  |  |  |
| 2 | A vehicle rental company wants to develop a system that maintains information about different type of vehicles 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.   * Cars should have an additional property such as number of doors. * Bikes should have a property indicating whether they have gears or not. * The system should also include a function to display details about each vehicle and indicate where a vehicle is starting. * Every class should have a constructor. |  |  |  |
| **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 a car. |  |  |  |
| 2 | :- A college is developing an automated admission system that verifies student eligibility for undergraduate (UG) and postgraduate (PG) programs. Each program has different eligibility criteria based on the students percentage in their previous qualification.   * UG admission requires a minimum of 60% * PG admission requires a minimum of 70% |  |  |  |
| 3 | Create a calculator class with overloaded methods to perform addition.   * Adding two Integers. * Adding three Integers. * Adding two Doubles. |  |  |  |
| 4 | :- Create a class Shape with a method CalculateArea() that is overloaded for different shapes (eg square, rectangle,….). create a sub class Circle that overrides the CalculateAreea()  Method for a circle. |  |  |  |
| **WEEK 7** |  |  |  |  |
| 1 | Write a Java program to create an abstract class Animal with an abstract method called sound(). Create subclasses Lion and Tiger that extend the Animal class and implement the sound() method to make a specific sound for each animal. |  |  |  |
| 2 | 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. |  |  |  |
| 3 | 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 a) and a concrete method to display the pattern title. * Implement two subclasses  1. Star Pattern – To print a right angled triangle with stars. 2. Number Pattern – To print a right angled triangle of increasing numbers. |  |  |  |
| **WEEK 8** |  |  |  |  |
| 1 | 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. |  |  |  |
| 2 | 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. |  |  |  |

WEEK – 1

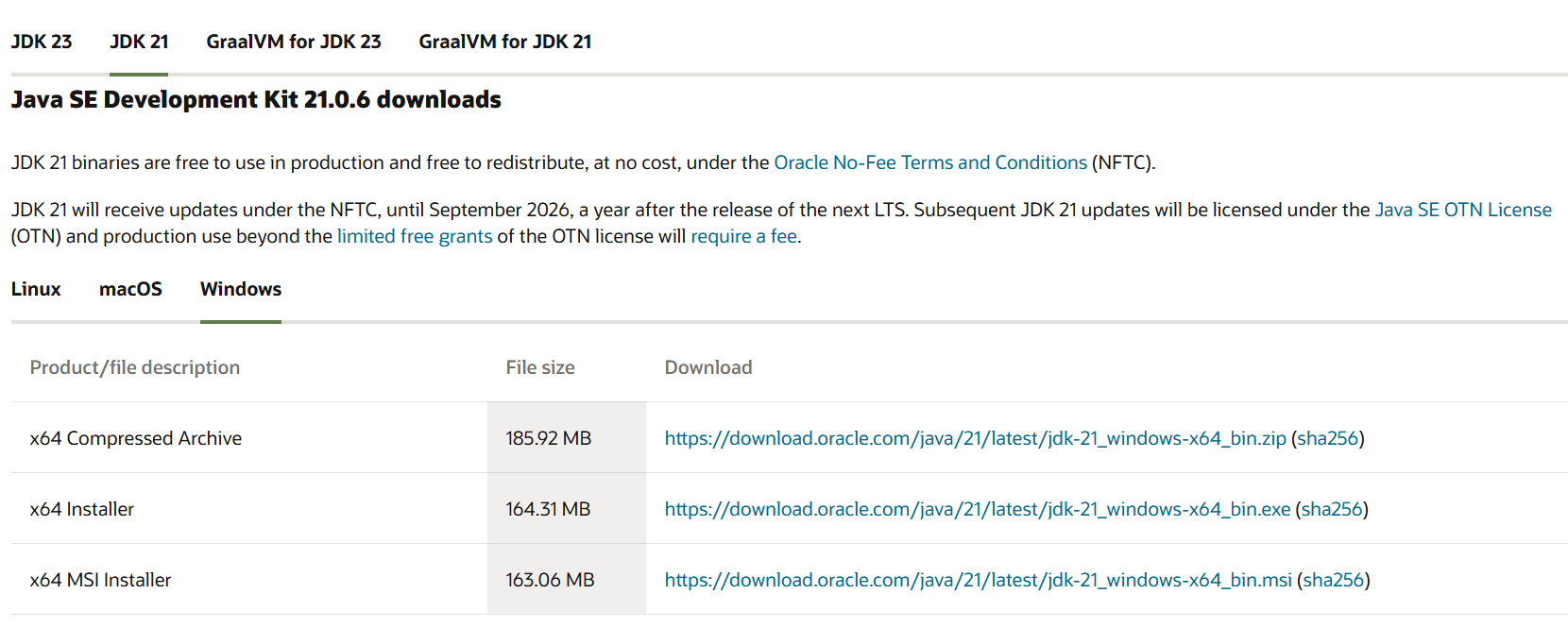
Aim :- To Download and Install Java (JDK21)

Procedure :-

**Step 1: Download JDK**

Go to the official [Oracle website](https://www.oracle.com/java/technologies/downloads/#jdk22-windows) to download the JDK.

Choose **x64 MSI Installer** on the windows tab and click on download link.



**Step 2: Run the Installer**

Now, go to your **downloads** folder and run the installer you just downloaded. You will be prompted to the screen below.

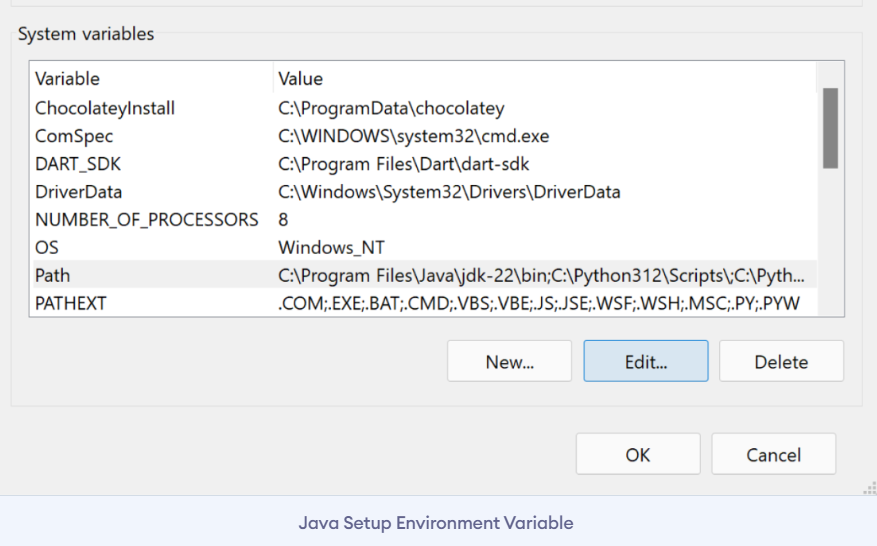


**Step 3: Configure Environment Variables**

After installation, you will need to tell your system where to find Java. This is done by setting environment variables.

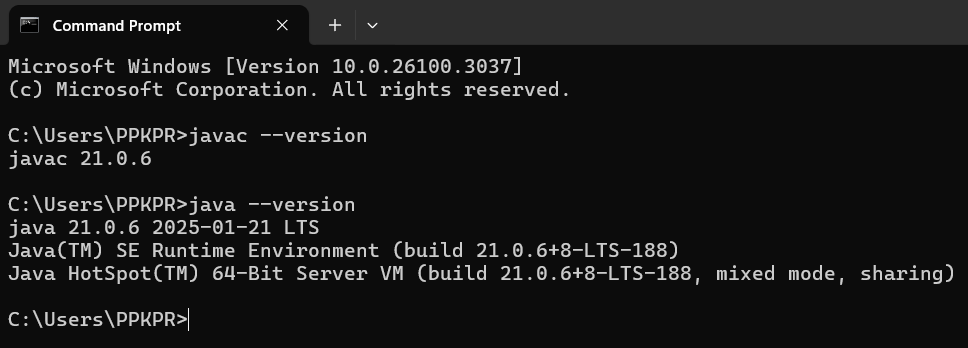
**Update the Path Variable:**

Find the **Path** variable in the System variables section and click on **Edit**. Then, click **New** and paste your JDK bin path.



**Step 4: Verify your Installation**

After the installation, you can verify whether Java is installed by using the following command in the command prompt.



Program – 1

AIM :- Write a Java program to print the message “HELLO WORLD”

Code :-

class first

{

    public static void main (String ar[ ] )

    {

        //to print hello world

        System.out.println("HELLO WORLD....");

    }

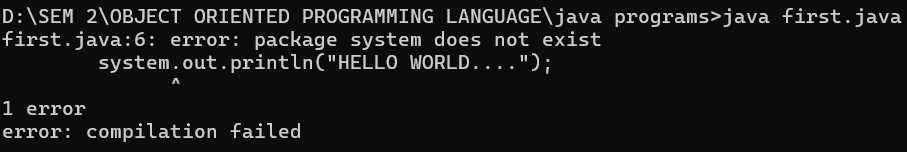
}

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

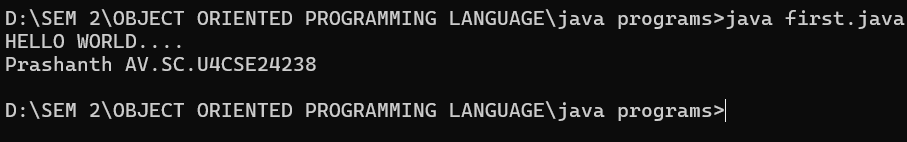
IMPORTANT POINTS:

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

Negative Case :-



OUTPUT :



Program – 2

AIM :- Write a JAVA Code to Print Student Details

Code :-

class student

{

    public static void main(String ar[])

    {

        System.out.println("Name    : RAM");

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

        System.out.println("Roll.No : AV.SC.U4CSE24238");

        System.out.println("Branch  : COMPUTER SCIENCE ENGINEERING");

        System.out.println("College : AMRITA VISHWA VIDHYAPEETHAM");

        System.out.println("AMARAVATHI");

    }

}

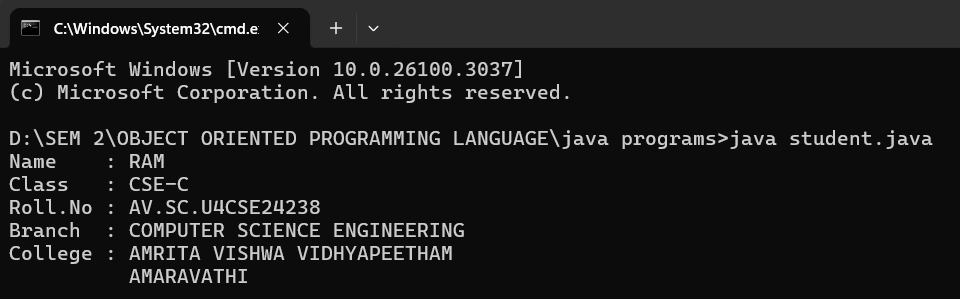
ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. writing small “S”in place of”S”   In system.out.println() | 1. code is rectified by keeping capital “S” |

IMPORTANT POINTS:

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

OUTPUT :



WEEK – 2

Program – 1

AIM :- Write a JAVA Code to print Fibonacci Series.

Code :-

import java.util.Scanner;

class fibonacci

{

 public static void main(String ar[])

 {

    Scanner sc = new Scanner(System.in);

    System.out.println("Enter a value : ");

    int x=sc.nextInt();

    int a=0,b=1,c;

    System.out.println(a);

    System.out.println(b);

    for(int i=0;i<=x;i++)

    {

        System.out.println(a+"");

        c=a+b;

        a=b;

        b=c;

    }

 }

}

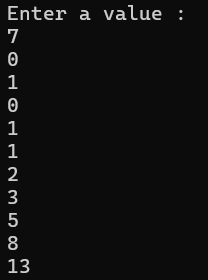
ERROR TABLE:

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

IMPORTANT POINTS:

1. In the Fibonacci sequence, the sum value is given to the second variable, and the value of the second variable is given to the first variable.
2. This process is repeated a certain number of times until the conditions are met.

Output :-



Program – 2

AIM :- Write a JAVA Code to find Factorial of a given number

Code :-

import java.util.Scanner;

class factorial

{

    public static void main(String ar[])

    {

        Scanner sc = new Scanner(System.in);

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

        int n = sc.nextInt();

        int x=1;

        for(int i=1;i<=n;i++)

        {

            x=x\*i;

        }

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

    }

}

ERROR TABLE:

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

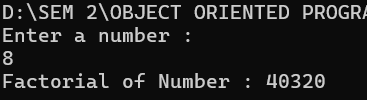
IMPORTANT POINTS:

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

Test expression and Update expression.

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

Output:



Program – 3

AIM :- Write a JAVA Code to calculate Simple Interest.

Code :-

import java.util.Scanner;

class SI

{

    public static void main(String ar[])

    {

        float si;

        Scanner input=new Scanner(System.in);

        System.out.println("Enter Principle amount : ");

        int p=input.nextInt();

        System.out.println("Enter Rate of Interest : ");

        float r=input.nextFloat();

        System.out.println("Enter Time in Years : ");

        int t=input.nextInt();

        si=(p\*t\*r)/100;

        System.out.println("Simple Interest : "+si);

    }

}

ERROR TABLE:

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

IMPORTANT POINTS:

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

P: Principal amount

R: Rate of interest

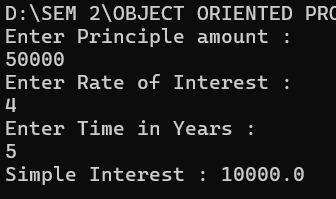
T: Time period

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

Import: tells the java compiler that we want to use a specific class or package in your code.

Java.util : This is the package that contains utility classes for Java programming, including the “Scanner” class.

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



Program – 4

AIM :- Write a JAVA Code to convert Celsius temperature into Fahrenheit temperature and vice versa.

Code :-

import java.util.Scanner;

class temp

{

public static void main (String ar[])

{

Scanner sc = new Scanner(System.in);

System.out.println("Enter 0 to convert Celcius into Fahrenheit Temperature.");

System.out.println("Enter 1 to convert Fahrenheit into Celcius Temperature.");

int n = sc.nextInt();

if (n==0)

{

System.out.println("Enter Temperature : ");

float x = sc.nextFloat();

double t=(x\*1.8)+32;

System.out.println("Converted Temperature : "+t);

}

else if (n==1)

{

System.out.println("Enter Temperature : ");

float x = sc.nextFloat();

double t=(x-32)/1.8;

System.out.println("Converted Temperature : "+t);

}

else

{

System.out.println("Enter Valid Input");

}

}

}

ERROR TABLE:

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

IMPORTANT POINTS:

1. The formula to convert a Fahrenheit to Celsius is

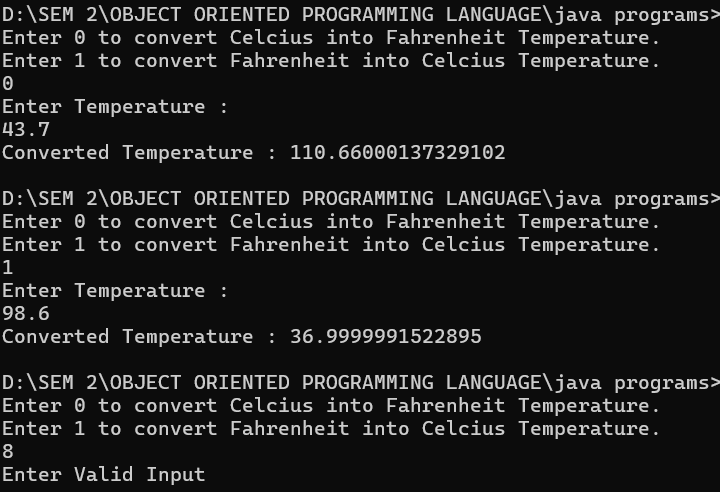
Celsius = (Fahrenheit-32)\*5/9

1. The formula to convert a Celsius to Fahrenheit is

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

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.

OUTPUT :



Program – 5

AIM :- // Write a JAVA Code to calculate Area of Rectangle and Area of Triangle using herons formula.

Code :-

import java.util.Scanner;

class area

{

public static void main(String ar[])

{

Scanner sc = new Scanner(System.in);

System.out.println("Enter Length of Rectangle : ");

float x = sc.nextFloat();

System.out.println("Enter Breadth of Rectangle : ");

float y = sc.nextFloat();

float z=x\*y;

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

System.out.println(“To find Area of Triangle”);

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

double a=sc.nextDouble

System.out.println("b : ");

double b=sc.nextDouble();

System.out.println("c : ");

double c=sc.nextDouble();

double s = a+b+c/2;

double area = Math.sqrt(s\*(s-a)\*(s-b)\*(s-c));

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

input.close();

}

}

ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| Not closing scanner with semicolon | Closing the scanner is must |

IMPORTANT POINTS:

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

L = length of a side of the rectangle,

B= breadth of a side of the rectangle.

1. Here, we must be sure that all the expressions/conditions inside for the for loop must be given correctly.
2. Heron’s formula for finding a triangle is:

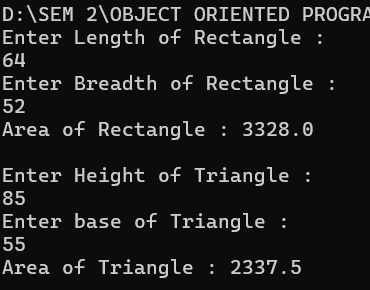
S = (a +b +c)/2

Where S is the semi-perimeter of the triangle.

Now the area formula is:

Area = sqrt(s\*(s-a)\*(s-b)\*(s-c)).

OUTPUT:



WEEK – 3

Program - 1

AIM :- Write a JAVA Code to create java program with following instructions

1.Create a class with name car

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

3. Create three methods named start(), stop(). Service()

4. Create three objects named car1,car2 and car3

**Class diagram:**

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

Code:-

import java.util.\*;

class car

{

    public String Car\_color;

    public String Car\_brand;

    public String fuel\_type;

    public int mileage;

   public void start()

   {

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

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

       System.out.println("Car Brand is:"+Car\_brand);

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

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

   }

    public void service()

   {

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

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

       System.out.println("Car Brand is:"+Car\_brand);

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

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

   }

    public void stop()

   {

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

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

       System.out.println("Car Brand is:"+Car\_brand);

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

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

   }

public static void main(String args[])

{

    car car1 = new car();

    car1.Car\_color = "Pink";

    car1.Car\_brand = "Audi";

    car1.fuel\_type = "Deisel";

    car1.mileage = 100;

   car1.start();

    car car2 = new car();

    car2.Car\_color = "Black";

    car2.Car\_brand = "Tesla";

    car2.fuel\_type = "EV";

    car2.mileage = 200;

    car2.stop();

    car car3 = new car();

    car3.Car\_color = "Blue";

    car3.Car\_brand = "BMW";

    car3.fuel\_type = "Petrol";

    car3.mileage = 300;

    car1car3.service();

 }

}

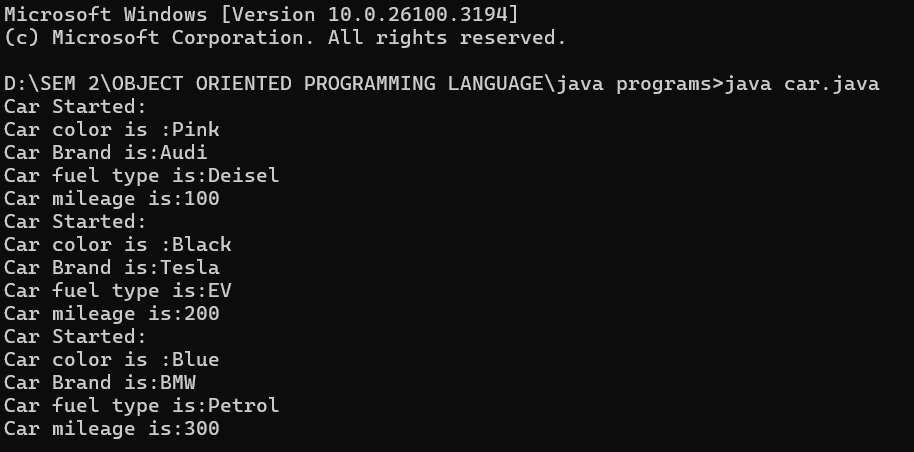
ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| car.java:44: error: ';' expected  car1.start() | By giving semicolon at last to statement. |

IMPORTANT POINTS:

1. Before calling the function we should write the method properly.
2. Here, the “public void start( )” indicates that we are writing a method to call the function.
3. When we call a certain method, the process inside it will be printed as an output of the code.
4. Here the details inside the function are called objects, we can give any objects

OUTPUT



Program – 2

AIM :-Write a JAVA Code to create a class bankAccount with methods deposit() and withdrawl

Class diagram:

|  |
| --- |
| BankAccount  ----------------------------------------------------------  -balance: double  ----------------------------------------------------------  +BankAccount(intialBalance: double)  +deposit(amount: double):void  +withdraw(amount: double):void |

Code :-

public class BankAccount {

    public double balance;

    public BankAccount(double initialBalance) {

        balance = initialBalance;

    }

    public void deposit(double amount) {

        if (amount > 0) {

            balance += amount;

            System.out.println("Deposited " + amount + ". New balance is " + balance);

        } else {

            System.out.println("Deposit amount must be positive.");

        }

    }

    public void withdraw(double amount) {

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

            balance -= amount;

            System.out.println("Withdrew " + amount + ". New balance is " + balance);

        } else {

            System.out.println("Invalid withdrawal amount. Check your balance.");

        }

    }

    public static void main(String[] args) {

        BankAccount account = new BankAccount(100); // Initial balance of 100

        account.deposit(100000);

        account.withdraw(30000);

        account.withdraw(1500); // Invalid withdrawal

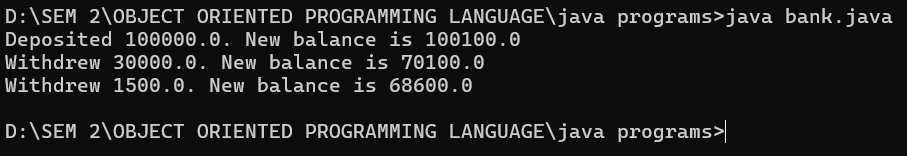
    }

}

ERROR TABLE :

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| Parenthesis is missing | Added parenthesis to the statement. |

OUTPUT :



WEEK – 4

Program - 1

Aim :- Write a java program with class named book. A class should contain various attributes such as title, author, year of publication. It should also contain a constructor with parameters which initializes title, author and year of publication. Create a method which displays the details of the book. Display the details of two books.

Class diagram:

|  |
| --- |
| book |
| - title: String  - author: String  - year: int |
| + Book(title: String, author: String, year: int)  + displayDetails(): void |
| + main(args: String[]): void |

Code :-

public class book

{

    String title;

    String author;

    int year;

    // Constructor

    public book(String title, String author, int year) {

        this.title = title;

        this.author = author;

        this.year = year;

    }

    // Method

    public void displayDetails() {

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

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

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

        System.out.println();

    }

    public static void main(String[] args) {

        //objects

        book book1 = new book("book 1", "author 1", 1999);

        book book2 = new book("book 2", "author 2", 2000);

        //book details

        System.out.println("Book 1 Details:");

        book1.displayDetails();

        System.out.println("Book 2 Details:");

        book2.displayDetails();

    }

}

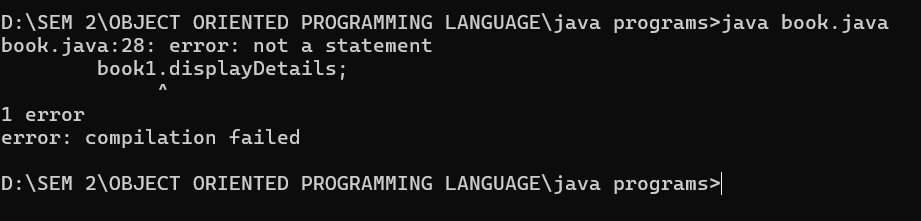
ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| book.java:28: error: cannot find symbol  book1.display(); | By adding,  book1.displayDetails(); |

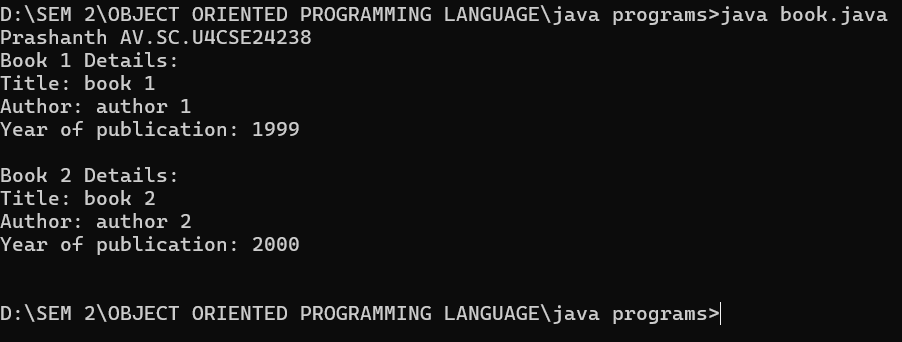
IMPORTANT POINTS:

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.

Negative Case :-



OUTPUT :



Program – 2

Aim :- Create a java program with class name “myclass” with static variable count of int type. Initialize it to zero and a constant variable pi of type double. Initialize pi to 3.14 as attributes of class. Now define a constructor for “myclass” that increments the count variable each time an object of “myclass” is created. Finally print the final values of count and pi variables. Create three objects.

Class diagram:

|  |
| --- |
| myclass |
| - static count: int  - final PI: double |
| + myClass()  + display(): void  + main(args: String[]): void |

Code :-

public class myclass

{

    static int count = 0;

    final double PI = 3.14;

    myclass() {

        count++;

    }

    void display() {

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

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

    }

    public static void main(String[] args) {

        myclass obj1 = new myclass();

        myclass obj2 = new myclass();

        myclass obj3 = new myclass();

        obj3.display();

    }

}

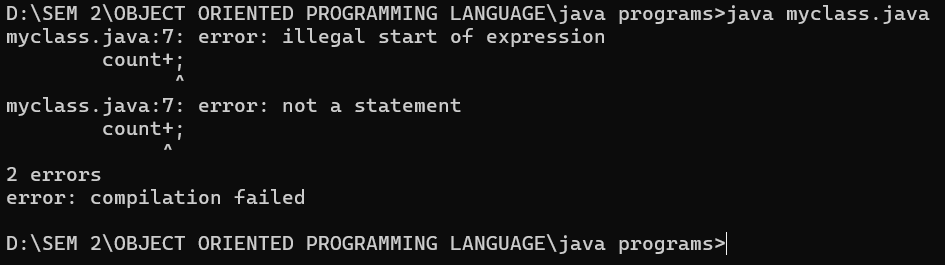
ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| myclass.java:7: error: illegal start of expression  count+; | By adding,  count++; |

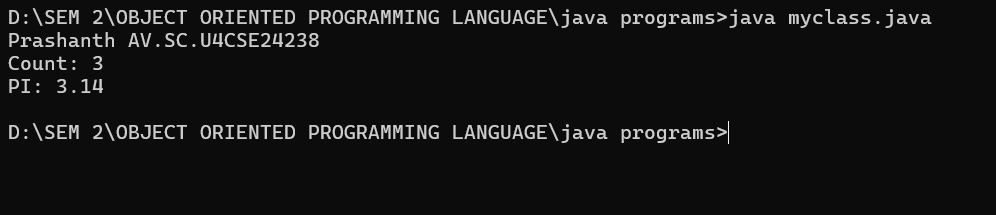
IMPORTANT POINTS:

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.

Negative Case :-



Output :-



WEEK – 5

Program - 1

AIM :- Create a calculator using the operations including addition, subtraction ,

multiplication and division using multilevel inheritance and display the desired output.

Class Diagram :

|  |
| --- |
| Calculator |
| +addition(int, int)  +diff(int, int) |

↓

|  |
| --- |
| Calci2 |
| +multi(int, int) |

↓

|  |
| --- |
| Calci3 |
| +divide(int, int) |

↓

|  |
| --- |
| Multilevel |
| +main(String[]) |

Code :-

import java.util.Scanner;

class Calculator {

    public void addition(int a, int b) {

        System.out.println("Addition: " + (a+b));

    }

    public void diff(int a, int b) {

        System.out.println("Subtraction: " + (a-b));

    }

}

class Calci2 extends Calculator {

    public void multi(int a, int b) {

        System.out.println("Multiplication: " + (a\*b));

    }

}

class Calci3 extends Calci2 {

    public void divid(int a, int b) {

        if (b == 0) {

            System.out.println("b should not be zero");

        } else {

            System.out.println("Division: " + ((double)a / b));

        }

    }

}

class Main {

    public static void main(String[] args) {

        System.out.println("Prashanth AV.SC.U4CSE24238");

        Scanner sc = new Scanner(System.in);

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

        int num1 = sc.nextInt();

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

        int num2 = sc.nextInt();

        Calculator c1 = new Calculator();

        c1.addition(num1, num2);

        c1.diff(num1, num2);

        Calci2 c2 = new Calci2();

        c2.multi(num1, num2);

        Calci3 c3 = new Calci3();

        c3.divid(num1, num2);

    }

}

ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| can't find main(String[]) method in class: Calculator | By adding this method |

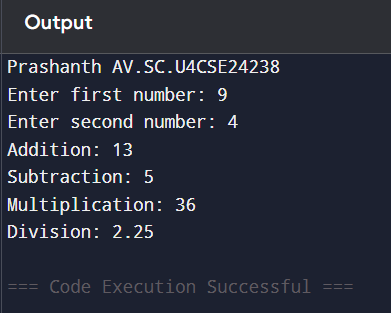
IMPORTANT POINTS :

This program includes **multilevel inheritance** where Calci3 inherits from Calci2, and Calci2 inherits from Calculator.

Negative Case :-



Output :



Program - 2

AIM :- A vehicle rental company wants to develop a system that maintains information about different type of vehicles 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.

* Cars should have an additional property such as number of doors.
* Bikes should have a property indicating whether they have gears or not.
* The system should also include a function to display details about each vehicle and indicate where a vehicle is starting.
* Every class should have a constructor.

Class Diagram :-

+----------------------------+

| Vehicle |

+----------------------------+

| - brand: String |

| - speed: int |

+----------------------------+

| +Vehicle(String, int) |

| +start(): void |

+----------------------------+

▲

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

| | |

+---------------+ +---------------+ +------------------+

| Car | | Bike | | Truck |

+---------------+ +---------------+ +------------------+

| - doors: int | | - hasGears:| | - capacity: int|

| | | boolean | | |

+-----------------+ +---------------+ +------------------+

| +Car(...) | | +Bike(...) | | +Truck(...) |

| +showCar...()| | +showBike...()| | +showTruck...()|

+---------------+ +------------------+ +-------------------+

Code :-

public class Vehicle {

    String brand;

    int speed;

    Vehicle(String brand, int speed) {

        this.brand = brand;

        this.speed = speed;

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

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

    }

    void start() {

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

    }

}

class Car extends Vehicle {

    int doors;

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

        super(brand, speed);

        this.doors = doors;

    }

    public void showCarDetails() {

        System.out.println("Car has " + doors + " doors.");

    }

}

class Bike extends Vehicle {

    boolean hasGears;

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

        super(brand, speed);

        this.hasGears = hasGears;

    }

    public void showBikeDetails() {

        System.out.println("Bike has gears: " + hasGears);

    }

}

class Truck extends Vehicle {

    int capacity;

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

        super(brand, speed);

        this.capacity = capacity;

    }

    public void showTruckDetails() {

        System.out.println("Truck has a capacity of " + capacity + " tons.");

    }

}

class Main {

    public static void main(String[] args) {

        System.out.println("Prashanth AV.SC.U4CSE24238");

        Car c1 = new Car("Mahindra", 58, 4);

        c1.showCarDetails();

        c1.start();

        Bike b1 = new Bike("Yamaha", 80, true);

        b1.showBikeDetails();

        b1.start();

       Truck t1 = new Truck("Tata", 60, 15);

        t1.showTruckDetails();

        t1.start();

    }

}

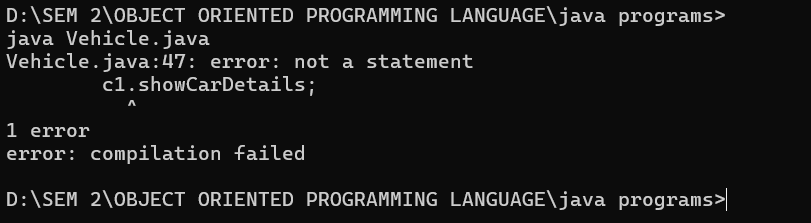
ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| Vehicle.java:47: error: cannot find symbol  c1.showCarDetails; | By adding  c1.showCarDetails(); |

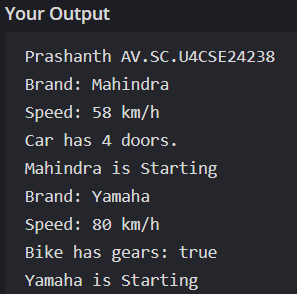
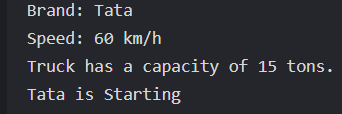
IMPORTANT POINTS :

The program demonstrates **inheritance**, where Car, Bike, and Truck inherit from the Vehicle class. This allows for **code reusability**, as common properties (brand, speed) and methods (start()) are defined once in the Vehicle class and reused in all subclasses.

Negative Case :-



Output :-

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 a car.

Class Diagram :-

+---------------------------+

| Vehicle |

+----------------------------+

| +displayInfo(): void |

+---------------------------+

▲

|

+---------------------------+

| Car |

+---------------------------+

| +displayInfo(): void |

+--------------------------+

+------------------+

| info |

+------------------+

| +main(String[]): void |

+------------------+

Code :-

class Vehicle1 {

    String brand;

    String model;

    String color;

    public Vehicle1(String brand, String model, String color) {

        this.brand = brand;

        this.model = model;

        this.color = color;

    }

    void displayInfo(){

        System.out.println("Vehicle");

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

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

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

    }

}

class Car extends Vehicle1{

    public Car(String brand, String model, String color) {

        super(brand, model, color);

    }

    void displayInfo(){

        System.out.println("Car");

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

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

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

    }

}

class info{

    public static void main(String[] args) {

        System.out.println("Prashanth AV.SC.U4CSE24238");

        Vehicle1 obj2 = new Vehicle1("Lorry", "Ashok", "Yellow");

        obj2.displayInfo();

        Car obj1 = new Car("Benz", "C200", "Blue");

        obj1.displayInfo();

    }

}

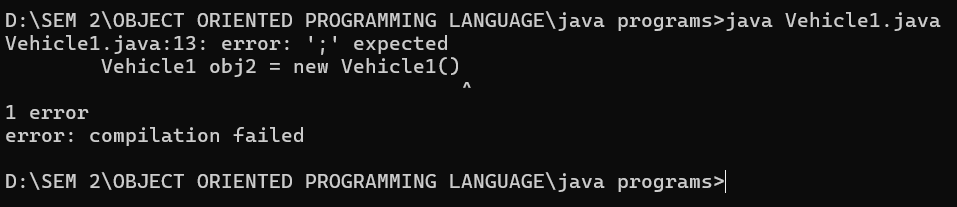
ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| Missing semicolon in line 13 | By adding semicolon to that line |

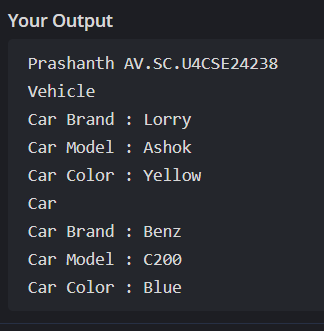
IMPORTANT POINTS :-

1. The code demonstrates inheritance, where car extends vehicle.
2. It shows method overriding using displayInfo() in both classes.
3. The main() method in a separate class runs and displays the output.

Negative Case :-



Output :-



Program - 2

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

* UG admission requires a minimum of 60%
* PG admission requires a minimum of 70%

Class Diagram :-

+---------------------+

| College |

+---------------------+

| + main(String[]): void |

+---------------------+

| - Scanner object used |

| - String a (program) |

| - double b (percentage)|

| - if-else logic for eligibility |

+---------------------+

Code :-

import java.util.Scanner;

public class College {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

System.out.println("Prashanth AV.SC.U4CSE24238");

        System.out.println("Enter the program (UG/PG): ");

        String a = sc.nextLine().toUpperCase();

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

        double b = sc.nextDouble();

        if (a.equals("UG")) {

            if (b >= 60) {

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

            }

            else {

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

            }

        }

        else if (a.equals("PG")) {

            if (b >= 70) {

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

            } else {

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

            }

        }

        else {

            System.out.println("Invalid program selected.");

        }

    }

}

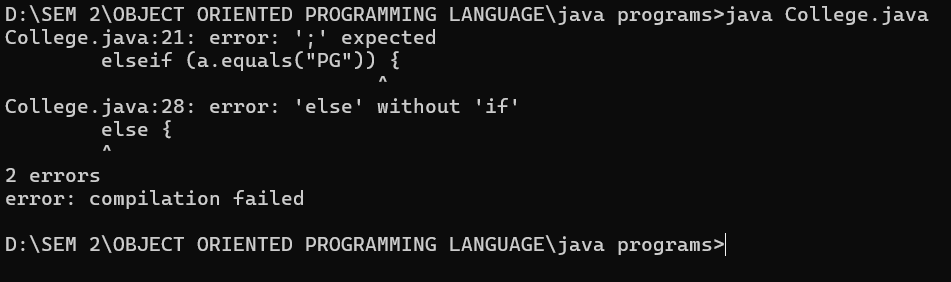
ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| College.java:21: error: ';' expected  elif (a.equals("PG")) {  ^  College.java:28: error: 'else' without 'if'  else {  ^ | Replacing elif with else if. |

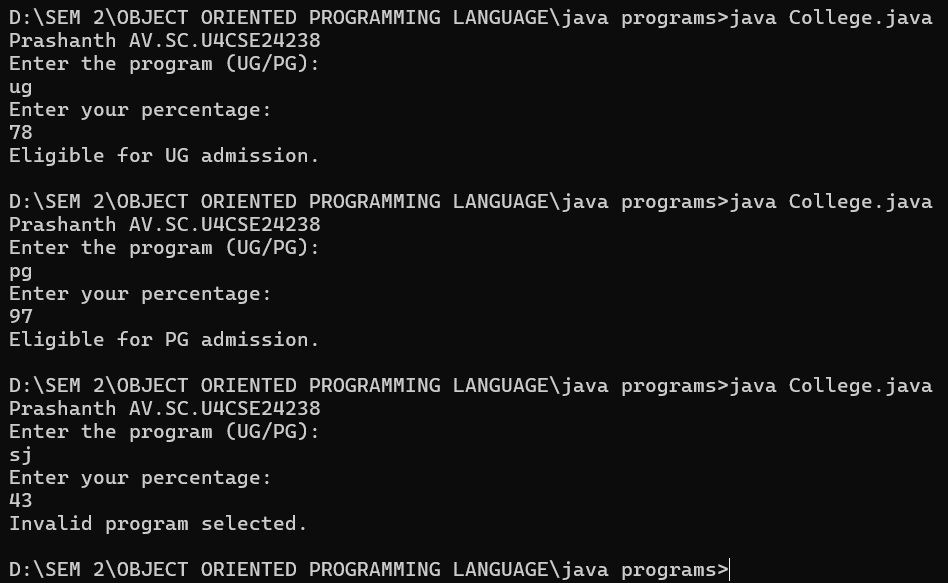
IMPORTANT POINTS :-

1. The program checks eligibility for UG and PG admissions based on percentage.
2. UG requires a minimum of 60%, PG requires 70% or more.
3. User input is taken using the Scanner class and handled with conditional statements.
4. The toUpperCase() method ensures the program input to be in capital letters.

Negative Case :-



Output :-



Program - 3

AIM :- Create a calculator class with overloaded methods to perform addition.

* Adding two Integers.
* Adding three Integers.
* Adding two Doubles.

Class Diagram :-

+-------------------+

| Addition |

+-------------------+

| + addition(int, int) |

| + addition(double, double) |

| + addition(int, int, int) |

+-------------------+

+-------------------+

| overload |

+-------------------+

| + main(String[]): void |

+-------------------+

Code :-

public class Addition {

    void addition(int a, int b) {

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

    }

    void addition(double a, double b){

        System.out.println("Addition of two double numbers :"+(a+b));

    }

    void addition(int a,int b,int c){

        System.out.println("Addition of three integers :"+(a+b+c));

    }

}

class overload{

    public static void main(String[] args) {

 System.out.println("Prashanth AV.SC.U4CSE24238");

    Addition obj = new Addition();

    obj.addition(10, 15);

    obj.addition(20, 15, 15);

    obj.addition(55.5, 44.5);

}

}

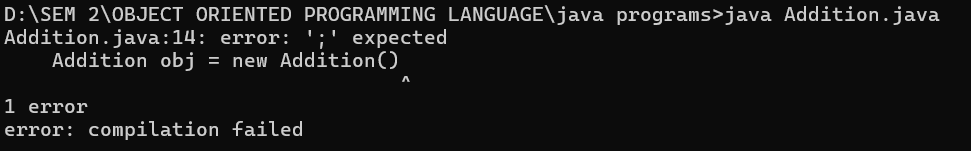
ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| Missing semicolon in line 14 | By adding semicolon to that line |

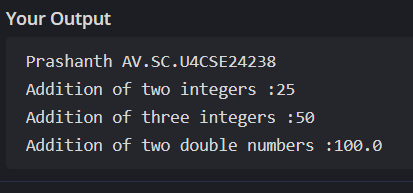
IMPORTANT POINTS :-

1. The program demonstrates method overloading multiple methods with the same name but different parameters.
2. The addition class has three addition() methods handling different data types and number of arguments.
3. Overloading improves code readability and reusability by grouping similar operations under one method name.
4. The main() method in the overload class creates an object and calls all overloaded versions of addition().

Negative Case :-



Output :-



Program - 4

AIM :- Create a class Shape with a method CalculateArea() that is overloaded for different shapes (eg square, rectangle,….). create a sub class Circle that overrides the CalculateAreea()

Method for a circle.

Class Diagram :-

+-----------------------------+

| Shape |

+-----------------------------+

| + calcArea(double side) |

| + calcArea(double length, double breadth) |

+-----------------------------+

▲

|

+-----------------------------+

| Circle |

+-----------------------------+

| + calcArea(double radius) |

+-----------------------------+

+-----------------------------+

| Area |

+-----------------------------+

| + main(String[] args): void |

+-----------------------------+

Code :-

public class Shape {

    void calcArea(double side){

        System.out.println("Area of Square : "+(side\*side));

    }

    void calcArea(double length,double breadth){

        System.out.println("Area of Rectangle : "+(length\*breadth));

    }

}

class Circle extends Shape{

    void calcArea(double radius){

        System.out.println("Area of Circle : "+(3.14\*radius\*radius));

    }

}

class Area{

    public static void main(String[] args) {

System.out.println("Prashanth AV.SC.U4CSE24238");

        Shape obj1 = new Shape();

        obj1.calcArea(12);

        obj1.calcArea(4, 9);

        Circle obj2 = new Circle();

        obj2.calcArea(15);

    }

}

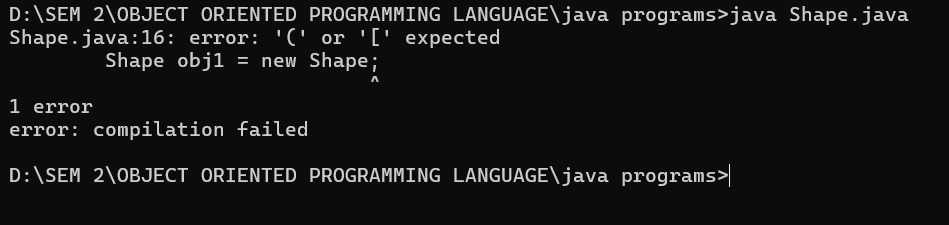
ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| Parenthesis is missing in line 16 | Adding parenthesis to the statement. |

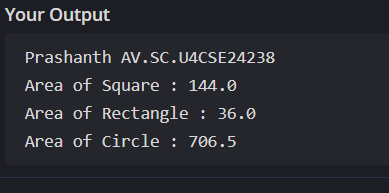
IMPORTANT POINTS :-

1. The Shape class demonstrates method overriding by defining multiple calcArea() methods with different parameters.
2. The circle class inherits from shape and defines its own calcArea() method to calculate the area of a circle.

Negative Case :-



Output :-



WEEK – 7

Program - 1

AIM :- Write a Java program to create an abstract class Animal with an abstract method called sound(). Create subclasses Lion and Tiger that extend the Animal class and implement the sound() method to make a specific sound for each animal.

Class Diagram :-

+------------------+

| <<abstract>> |

| Animal2 |

+------------------+

| +sound(): void | <--- Abstract Method

+------------------+

↑

------------------------------

| |

+---------------+ +---------------+

| Lion | | Tiger |

+---------------+ +---------------+

| +sound(): void| | +sound(): void|

+---------------+ +---------------+

Code :-

abstract class Animal2 {

    abstract void sound ();

}

class Lion extends Animal2 {

    @Override

    void sound() {

        System.out.println("Roar");

    }

}

class Tiger extends Animal2 {

    @Override

    void sound() {

        System.out.println("Purrrrr");

    }

}

class Main {

    public static void main(String[] args) {

        System.out.println("Prashanth P");

        System.out.println("AV.SC.U4CSE24238");

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

        System.out.println();

        Lion obj1 = new Lion();

        Tiger obj2 = new Tiger();

        obj1.sound();

        obj2.sound();

    }

}

ERROR TABLE :-

|  |  |
| --- | --- |
| Error | Rectification |
| Return type for the method is missing | By adding return statement |

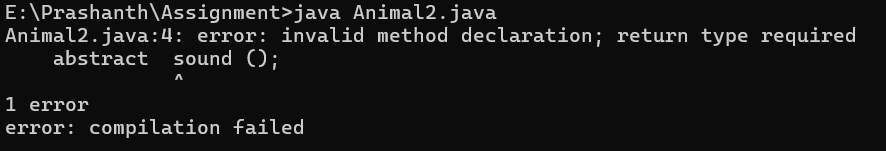
IMPORTANT POINTS :-

1. Uses abstract `Animal2` class with abstract `sound()` method, implemented differently by `Lion` ("Roar") and `Tiger` ("Purrrrr") subclasses.

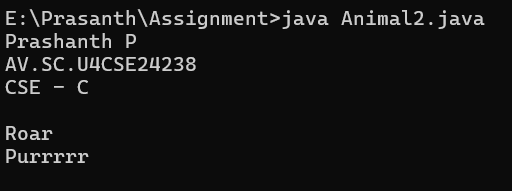
2. Demonstrates polymorphism by calling `sound()` on both animal types, producing distinct outputs.

3. Includes student details header while showing how abstract classes enforce method implementation in child classes.

Negative Case :-



Output :-



Program - 2

AIM :- Write a java program to create an abstract class Shape3D with abstract methods calculateVolume() and calculateSurfaceArea(). Create subclasses Sphere and Cube that extend the Shape3D class and implement the respective methods to calculate the volume and surface area of each shape.

Class Diagram :-

+-----------------------------------+

| <<abstract>> |

| Shape3D |

+----------------------------------------+

| +calculateVolume(): double |

| +calculateSurfaceArea(): double |

+-----------------------------------------+

↑

---------------------------------------

| |

+--------------------------+ +-----------------------+

| Sphere | | Cube |

+--------------------------+ +-----------------------+

| - radius: double | | - side: double |

+--------------------------+ +-----------------------+

| +calculateVolume(): double | | +calculateVolume(): double |

| +calculateSurfaceArea(): double | | +calculateSurfaceArea(): double|

+--------------------------+ +-----------------------+

Code :-

import java.util.Scanner;

abstract class Shape3D {

    abstract double calculateVolume();

    abstract double calculateSurfaceArea();

}

class Sphere extends Shape3D {

    double radius;

    public Sphere(double radius) {

        this.radius = radius;

    }

    @Override

    public double calculateVolume() {

        return (4/3)\*3.14\*radius\*radius\*radius;

    }

    @Override

    public double calculateSurfaceArea() {

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

    }

}

class Cube extends Shape3D {

    double side;

    public Cube(double side) {

        this.side = side;

    }

    @Override

    public double calculateVolume() {

        return side\*side\*side;

    }

    @Override

    public double calculateSurfaceArea() {

        return 6\*side\*side;

    }

}

class Main {

    public static void main(String[] args) {

        System.out.println("Prashanth P");

        System.out.println("AV.SC.U4CSE24238");

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

        System.out.println();

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the radius of the sphere: ");

        double radius = sc.nextDouble();

        Sphere obj1 = new Sphere(radius);

        System.out.println("SPHERE");

        System.out.println("VOLUME OF SPHERE : "+obj1.calculateVolume());

        System.out.println("SURFACE AREA OF SPHERE : "+obj1.calculateSurfaceArea());

        System.out.println();

        System.out.println("Enter the side of the cube: ");

        double side = sc.nextDouble();

        Cube obj2 = new Cube(side);

        System.out.println("CUBE");

        System.out.println("VOLUME OF CUBE : "+obj2.calculateVolume());

        System.out.println("SURFACE AREA OF CUBE : "+obj2.calculateSurfaceArea());

    }

}

ERROR TABLE :-

|  |  |
| --- | --- |
| **Error** | **Rectification** |
| **Semicolon expected in line 5 and 6.** | **By adding semicolon** |

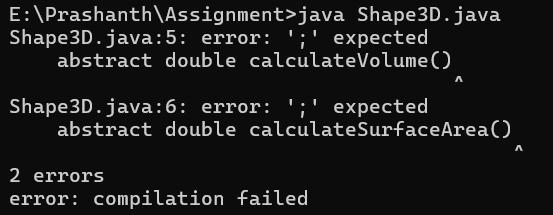
IMPORTANT POINTS :-

1. Defines abstract `Shape3D` class with volume and surface area methods implemented differently by `Sphere` and `Cube` subclasses.

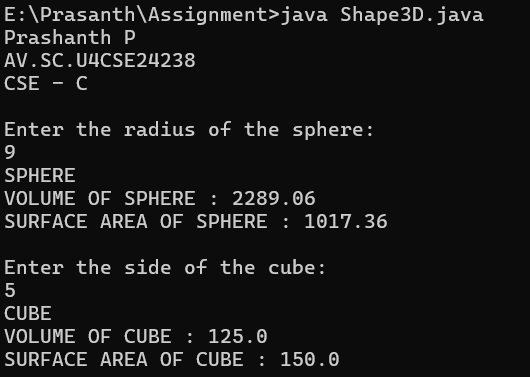
2. `Sphere` uses radius for calculations (volume: 4/3πr³, surface: 4πr²), while `Cube` uses side length (volume: s³, surface: 6s²).

3. Takes user input for dimensions, creates objects, and displays calculated results with student info header.

Negative Case :-



Output :-



Program - 3

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

* Create an abstract class named PatternPrinter with an abstract method printPattern(int a) and a concrete method to display the pattern title.
* Implement two subclasses

Star Pattern – To print a right angled triangle with stars.

Number Pattern – To print a right angled triangle of increasing numbers.

Class Diagram :-

+--------------------------------+

| PatternPrinter | (abstract class)

+--------------------------------+

| +displayTitle(title): void |

| +printPattern(a): void | (abstract)

+--------------------------------+

/\

|

---------------------------------

| |

+---------------------+ +----------------------+

| StarPattern | | NumberPattern |

+---------------------+ +----------------------+

| +printPattern(a): void | | +printPattern(a): void |

+---------------------+ +----------------------+

Code :-

abstract class PatternPrinter {

    void displayTitle(String title) {

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

    }

    abstract void printPattern(int a);

}

class StarPattern extends PatternPrinter {

    void printPattern(int a) {

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

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

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

            }

            System.out.println();

        }

    }

}

class NumberPattern extends PatternPrinter {

    void printPattern(int a) {

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

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

                System.out.print(i);

            }

            System.out.println();

        }

    }

}

public class Main {

    public static void main(String[] args) {

        System.out.println("Prashanth P");

        System.out.println("AV.SC.U4CSE24238");

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

        PatternPrinter pattern1 = new StarPattern();

        pattern1.displayTitle("Star Triangle");

        pattern1.printPattern(5);

        System.out.println();

        PatternPrinter pattern2 = new NumberPattern();

        pattern2.displayTitle("Number Triangle");

        pattern2.printPattern(5);

    }

}

ERROR TABLE :-

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| Placing comma instead of semicolon. | Rewriting the code using semicolon. |

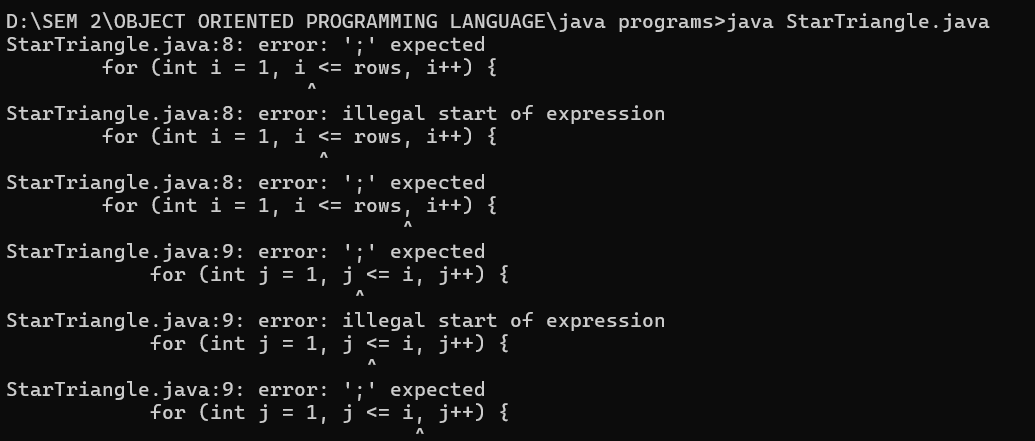
IMPORTANT POINTS :-

1. PatternPrinter is an abstract class with a concrete and an abstract method.

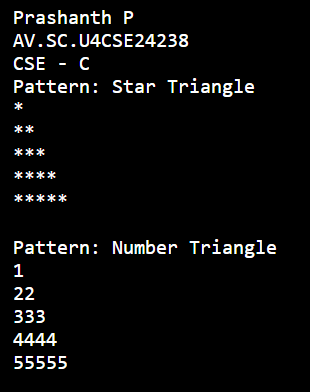
2. Subclasses StarPattern and NumberPattern implement the printPattern method differently.

3. The Main class demonstrates polymorphism by using the abstract class reference.

Negative Case :-



Output :-



WEEK – 8

Program - 1

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

Class Diagram :-

+----------------------+

| <<interface>> |

| Shape2 |

+----------------------+

| +getPerimeter(): double |

+----------------------+

↑

|

-----------------------------------------------------------------

| | |

+----------------+ +----------------+ +--------------------+

| Rectangle | | Circle | | Triangle |

+----------------+ +----------------+ +--------------------+

| - a: int | | - r: int | | - a: int |

| - b: int | +----------------+ | - b: int |

| +getPerimeter(): double || +getPerimeter(): double | | - c: int |

+----------------+ +----------------+ +----------------+

| +getPerimeter(): double |

+--------------------+

Code :-

public interface Shape2 {

    double getPerimeter();

}

class Rectangle implements Shape2 {

    private int a;

    private int b;

    public Rectangle(int a, int b) {

        this.a = a;

        this.b = b;

    }

    @Override

    public double getPerimeter() {

        return 2 \* (a + b);

    }

}

class Circle implements Shape2 {

    private int r;

    public Circle(int r) {

        this.r = r;

    }

    @Override

    public double getPerimeter() {

        return 2 \* 3.14 \* r;

    }

}

class Triangle implements Shape2 {

    private int a;

    private int b;

    private int c;

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

        this.a = a;

        this.b = b;

        this.c = c;

    }

    @Override

    public double getPerimeter() {

        return a+b+c;

    }

}

class Main {

    public static void main(String[] args) {

        System.out.println("Prashanth P");

        System.out.println("AV.SC.U4CSE24238");

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

        System.out.println();

        Rectangle obj1 = new Rectangle(8, 3);

        Circle obj2 = new Circle(5);

        Triangle obj3 = new Triangle(3, 4, 5);

        System.out.println("Rectangle : "+obj1.getPerimeter());

        System.out.println("Circle : "+obj2.getPerimeter());

        System.out.println("Triangle : "+obj3.getPerimeter());

    }

}

ERROR TABLE :-

|  |  |
| --- | --- |
| **Error** | **Rectification** |
| **Brackets are required for calling a method.** | **By placing the brackets.** |

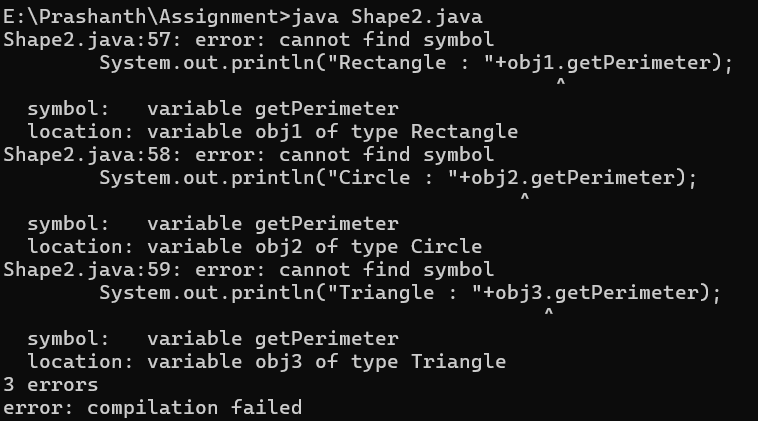
IMPORTANT POINTS :-

1. The code defines a `Shape2` interface with a single `getPerimeter()` method, which is implemented by three different shape classes (`Rectangle`, `Circle`, `Triangle`), each providing their own perimeter calculation logic.

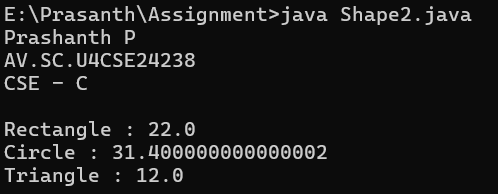
2. Each shape class contains specific attributes needed for its perimeter calculation (sides for rectangle/triangle, radius for circle) and implements the interface method accordingly, demonstrating polymorphic behavior through a common interface.

3. The `Main` class creates instances of each shape, calls their `getPerimeter()` methods, and prints the results, showing how different objects can respond to the same method call in different ways while including student identification details.

Negative Case :-



Output :-



Program - 2

AIM :- Write a Java program to create an interface Playable with a method play() that takes no arguments and returns void. Create three classes Football, Volleyball, and Basketball that implement the Playable interface and override the play() method to play the respective sports.

Class Diagram :-

+---------------------+

| <<interface>> |

| Playable |

+---------------------+

| +play(): void |

+---------------------+

↑

|

------------------------------------------------

| | |

+----------------+ +------------------+ +-------------------+

| Football | | Volleyball | | Basketball |

+----------------+ +------------------+ +-------------------+

| +play(): void | | +play(): void | | +play(): void |

+----------------+ +------------------+ +-------------------+

Code :-

public interface Playable {

    void play();

}

class Football implements Playable {

    public void play() {

        System.out.println("All are Playing Football");

    }

}

class Volleyball implements Playable {

    public void play() {

        System.out.println("All are Playing Volleyball");

    }

}

class Basketball implements Playable {

    public void play() {

        System.out.println("All are Playing Basketball");

    }

}

class Main {

    public static void main(String[] args) {

        System.out.println("Prashanth P");

        System.out.println("AV.SC.U4CSE24238");

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

        System.out.println();

        Football obj1 = new Football();

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

        obj1.play();

        System.out.println();

        Volleyball obj2 = new Volleyball();

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

        obj2.play();

        System.out.println();

        Basketball obj3 = new Basketball();

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

        obj3.play();

    }

}

ERROR TABLE :-

|  |  |
| --- | --- |
| Error | Rectification |
| Private method cant be accessed. | Method should be default or abstract. |

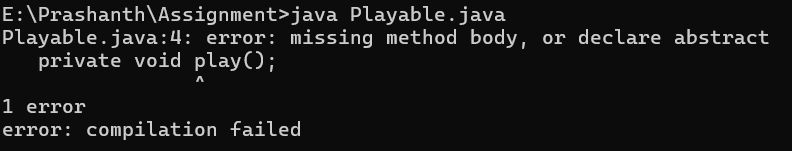
IMPORTANT POINTS :-

1. The code defines a `Playable` interface with a single `play()` method that multiple sports classes implement differently.

2. Three sports classes (`Football`, `Volleyball`, `Basketball`) implement the `Playable` interface, each providing their own version of the `play()` method with sport-specific messages.

3. The `Main` class demonstrates polymorphism by creating instances of each sport class and calling their `play()` methods, showing how the same interface can have different implementations.

Negative Case :-



Output :-

