

**(23CSE111) OBJECT ORIENTED PROGRAMMING**

**LAB MANUAL**

**CSE-1st YEAR II SEMESTER (2024-2025)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Submitted by** | | **Submitted to** | |
| **NAME:** | **Harsha Vardhini. M** | **NAME:** | **RAJ KUMAR BATCHU** |
| **ROLL NO:** | **AV.SC.U4CSE24225** | **DEPARTMENT:** | **CSE** |
| **SECTION:** | **CSE-C** | **DESIGNATION:** |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 program to print the student details name, course,semester,roll no |  |  |  |
| WEEK 2 |  |  |  |  |
| 1 | Write a Java program to calculate Simple interest. |  |  |  |
| 2 | Write a Java program to calculate factorial of a number. |  |  |  |
| 3 | write a program to convert the temperature from Celsius to Fahrenheit and Fahrenheit to Celsius |  |  |  |
| 4 | Write a Java program to calculate fibonacci of a number. |  |  |  |
| 5 | write a java program to find the area of a rectangle, where all the inputs are taken from the user |  |  |  |
| 6 | Write a Java program to find the area of the triangle using herons formula where inputs are taken from the user. |  |  |  |
| WEEK 3 |  |  |  |  |
| 1 | Write a java program with  1. A class with name Car.  2.Create 4 attributes named car\_color , car\_brand ,fuel\_type ,mileage. 3.Create 3 methods named start() ,stop() ,service(). 4.Create 3 objects named car1 ,car2 ,car3. |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | Write a java program with 1.Create a class named Bankaccount. 2.Create a constructor. 3.Create 2 methods which are withdrawl() and deposit(). |  |  |  |
| WEEK 4 |  |  |  |  |
| 1 | Write a java program with class named book. This class should contain various attributes such as title, author ,year of publication. It should also contain a constructor with parameters which initializers title ,author ,year of publication. Create a method which displays the details of the book .Display the details of 2 books. |  |  |  |
| 2 | Create a java program with class named myclass with a static variable count of int type ,initial value to zero and a constant variable "pi" of type double initialize to 3.14 as attributes of that class ,now define a constructor for "myclass" that increments the count variable each time an object of myclass is created. Finally print the values of count and pi variables . Create 3 objects. |  |  |  |
| WEEK 5 |  |  |  |  |
| 1 | Create a calculator using the operations including addition, subtraction, multiplication, and division using multi-level inheritance and display the desired output. |  |  |  |
| 2 | 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?  a.Truck should include an additional property capacity (in tons)  b.Create a showTruckdetails() method to display the truck’s capacity.  c.Write a constructor for Truck that initializes all properties Implement the truck class and update the main method to create a Truck object and also create an object for car and bike sub classes Finally, display the details. |  |  |  |
| 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 admissions systems that verifies students eligibility for undergraduate(UG) and postgraduate(PG) programs. Each program has different eligibility. Criteria based on the students percentage in their previous qualification.  1. UG admission require min of 60%  2. PG admission require min of 70% |  |  |  |
| 3 | Create a calculator class with overloaded methods to perform addition.  A. Add two integers  B. Add two double  C. Add three integer |  |  |  |
| 4 | Create a shape class with a method CalculateArea() that is overloaded for different shapes (e.g square, rectangle) then, create a subclass circle that overrides the calculatearea() 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 extends animal class and implement the sound() method to make a specific sound for each animal. |  |  |  |

Week -1

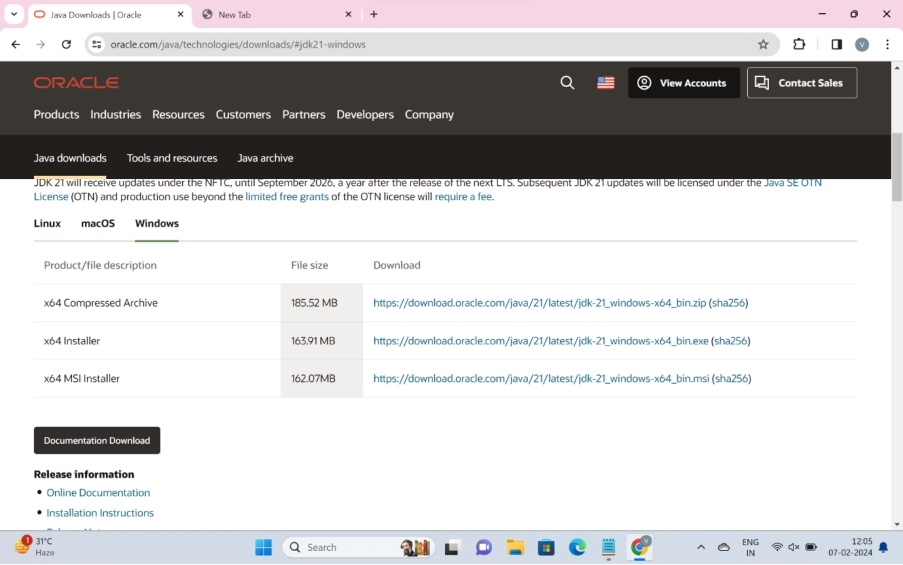
***AIM***: 1. To Download and install Java and Execute the First Java Program.

***PROCEDURE***:

1)First,we need to open google chrome and search for java download oracle.

2)Click on the first web site which appears with name JAVA downloads.

3)After entering into web site,we need to scroll down and choose the operating system which we are using Linux(or)Macos(or)Windows

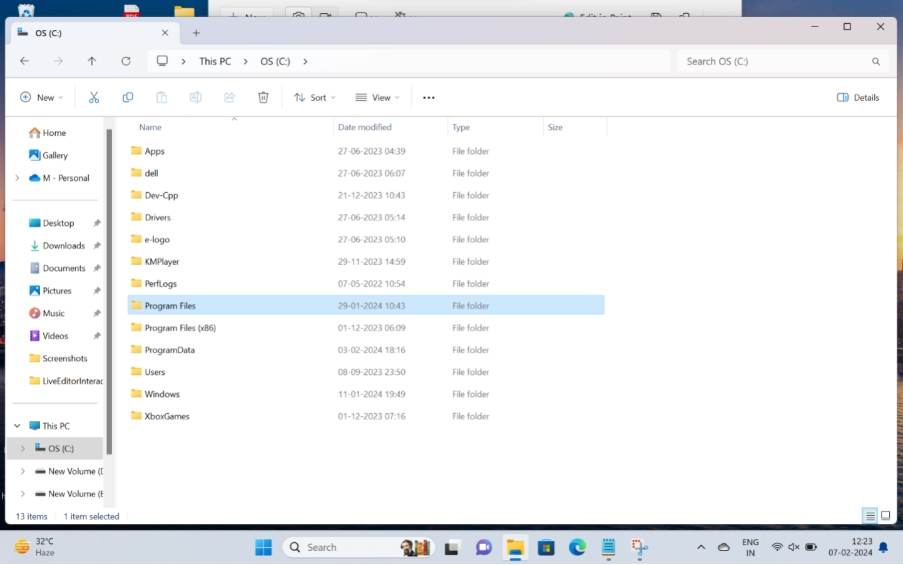


4)According to my lap top, I choose windows and click on **x64** **installer** which results download of java.

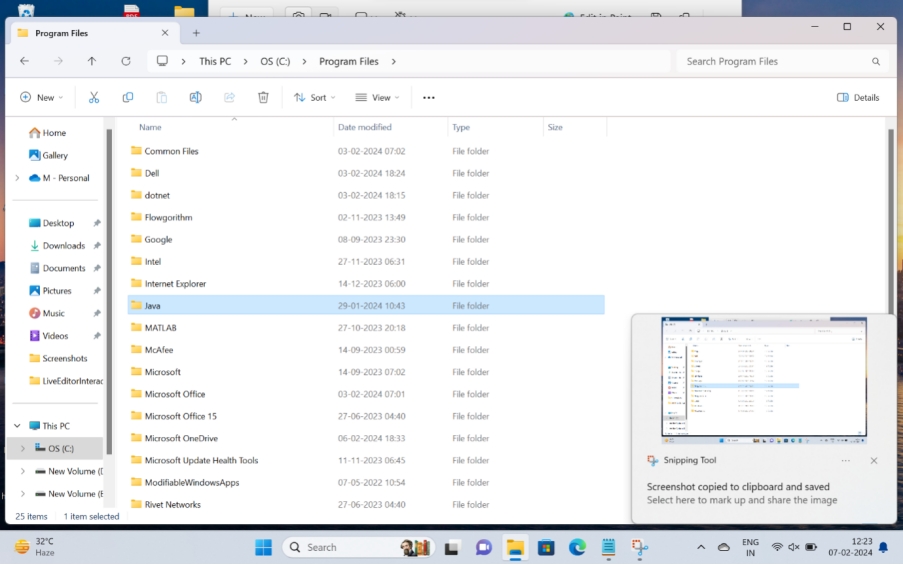
5)Then we need to go to downloads of laptop and allow the downloaded application to install in our laptop.

6)All the programming related files were stored in local disk(c) drive until then we changed it.

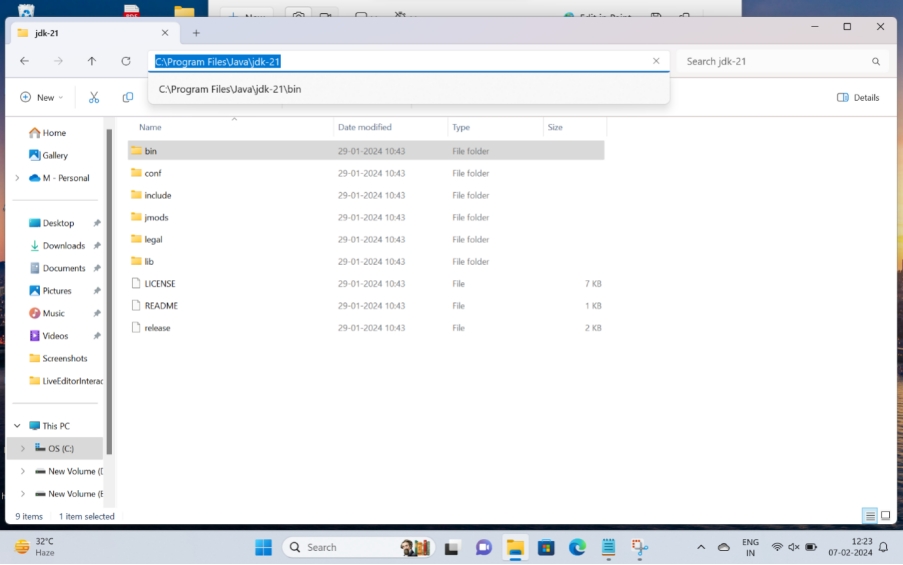
7)By moving into local disk(c),we need to move into program files.In it we find a file named java.



8)After entering into JAVA file. we need to move on to file JDK-21and then into bin.

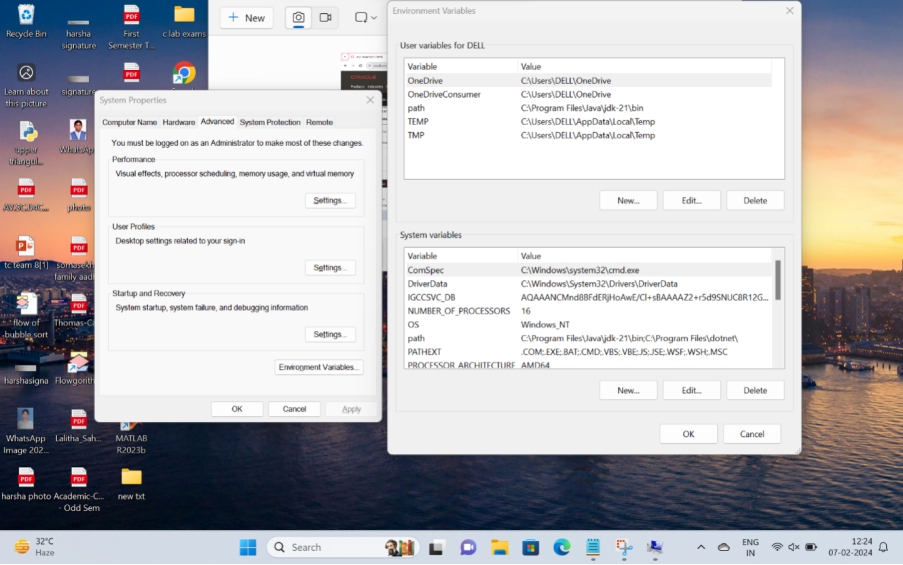


9)Now,By clicking on file address (or) location,we get c:\program files\JAVA\jdk-21\bin which we need to copy.



10)Now move on to windows search and search for environmental variables under advanced tab.

11)we need to click on environmental variables which is located in Advanced file.



12)Now,we can find user variables and system variables.

13)we need to choose system variables and click on new.

14)It shows variable name and variable value.In which we need to fill as variable name:JAVA\_HOME and variable value is to paste the copied content in file address at point(9).

15)By clicking “ok” we completed this process.

16)Now, to write a java program, we need to create a folder named java and a text document in the folder.

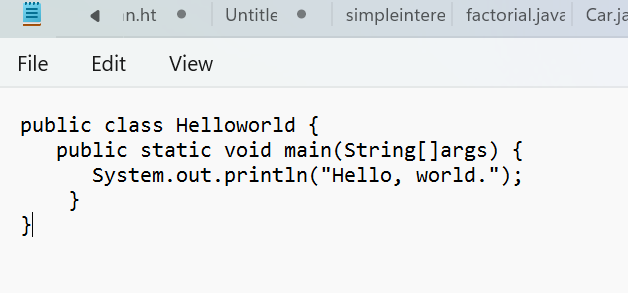
17)write the program in text document and save it by adding extension “.java”.

Type the following command: **java --version** and press Enter.

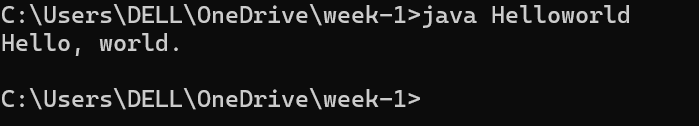
|  |
| --- |
|  |
|  |  |

Program 2:

Aim : write a java code to print the message “Hello world”



Output:



Error table:

|  |  |
| --- | --- |
| Error | rectification |
| Helloworld.java:3: error: ';' expected  System. out. println("Hello, world.")  ^  1 error | Need to keep semicolon after  System. out. println (“Hello, world.”); |

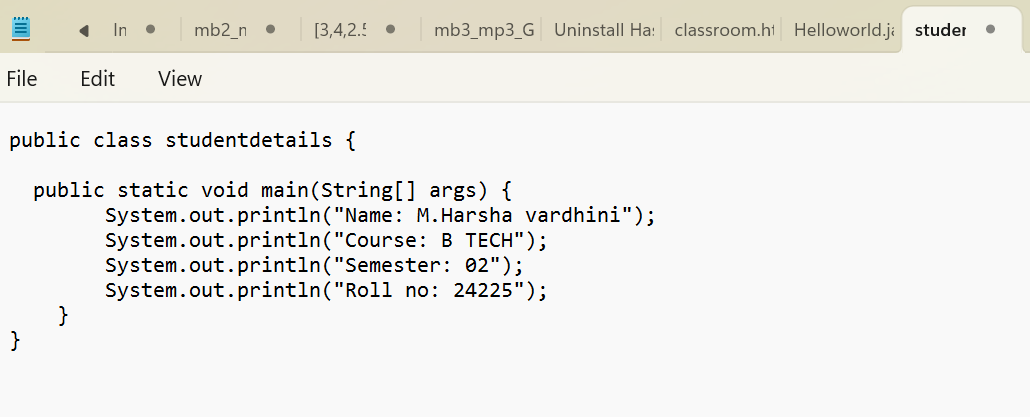
Important things :

* Everything you want to print should be kept inside parentheses ().
* The text to be printed is enclosed within double quotes "".
* Each System.out.println() statement ends with a semicolon ;

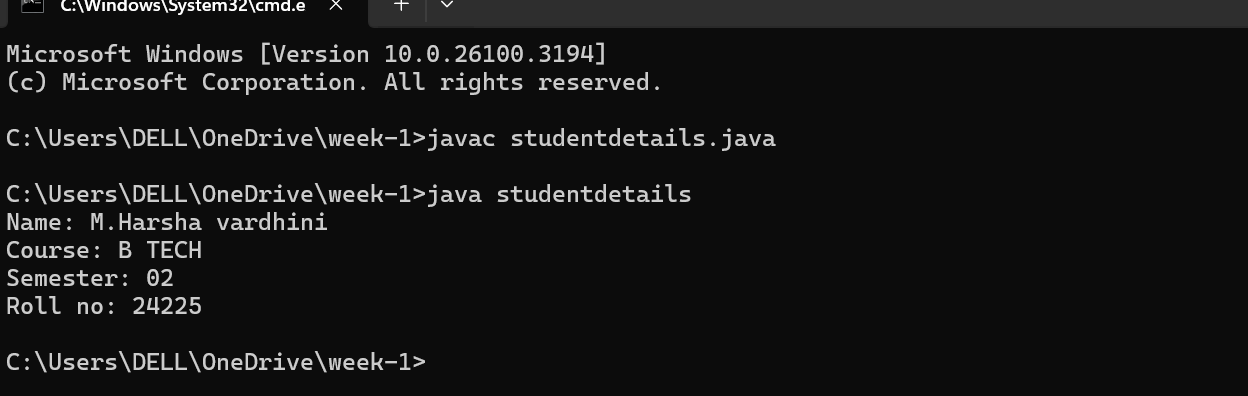
Program : 3

AIM: write a java program to print the student details name, course,semester,roll no

Code



Output:



Error table:

|  |  |
| --- | --- |
| error | rectification |
| Writing small “s” in place of  “S” | Code is rectified by keeping capital “S” |

Key points:

1)class definition:

.The program defines a class named studentdetails

2)main method

The main method is the entry point of the program defined as public static void main(String[] args), which means:

* public: The method is accessible from anywhere.
* static: The method can be called without creating an instance of the class.
* void: The method doesn't return anything.
* String[] args: It accepts an array of String arguments, though they are not used in this program.

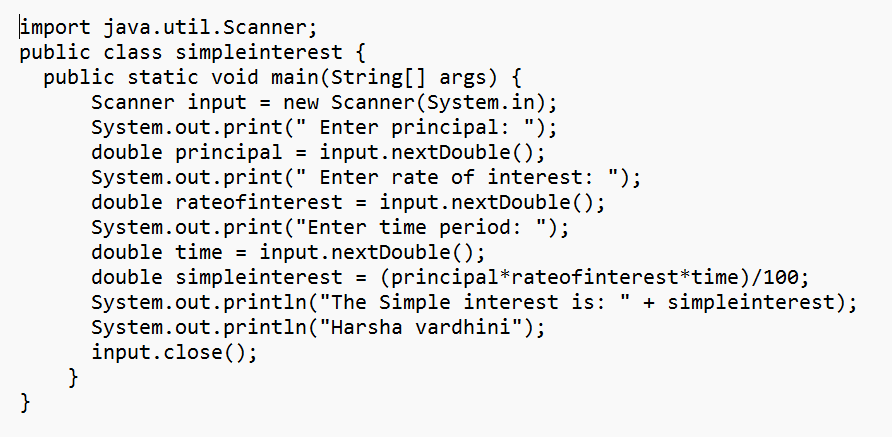
3) the program prints the the following details using System.out.println()

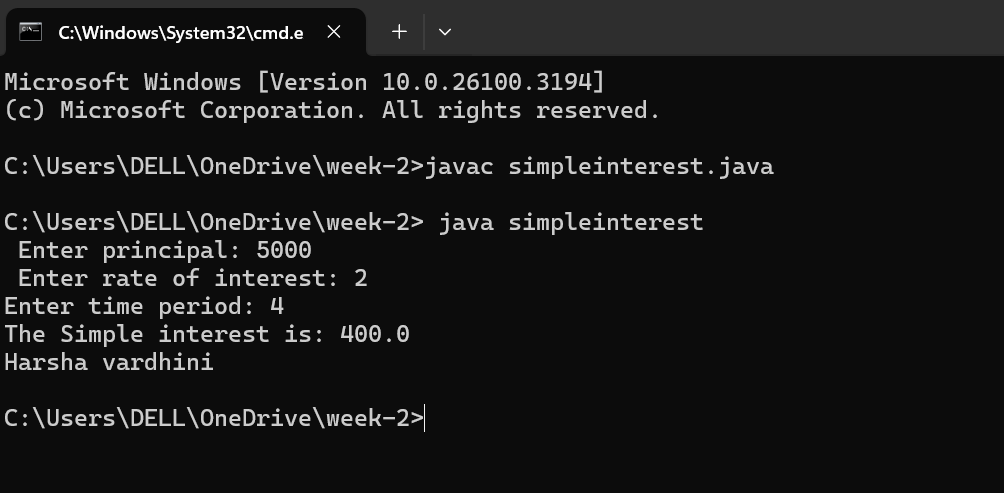
WEEK-2

Program :1

Aim : write a java program to write simple interest where all the inputs are taken from the user

Code :



Output: 

Error table:

|  |  |
| --- | --- |
| Error | rectification |
| 1.Giving space between next  and double | 1. Should not give space between nextDouble |

Key points:

JAVA.util package: The java.util package in java is a versatile buit-in package that contains various utility classes and interfaces.it provides basic functionality for commonly ocuuring use cases.it contains Java’s collections framework,date and time utilities,string-tokenizer,event-model utilities,etc

**import keyword**: It allows you to bring external classes or packages into your Java program. In this case, you're importing the Scanner class.

**java.util.Scanner**: This specifies the full path to the class. The Scanner class resides in the java.util package, which is why you need to import it.

The Scanner class is used to take user input in Java.

. The nextDouble() method is used to read double values (for principal, rate of interest, and time period).

. The input.close() line at the end of the program is used to close the Scanner object and free up resources after the input is completed.

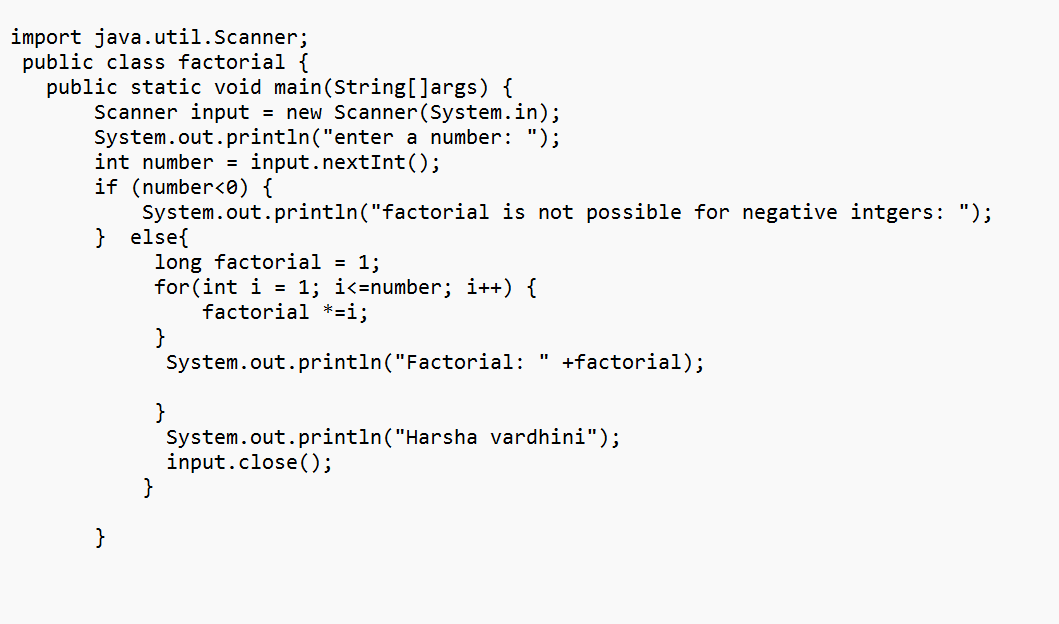
* **import java.util.Scanner;** allows your program to use the Scanner class for input.

Without it, the Scanner class wouldn't be recognized by your program, and you wouldn’t be able to read user input using methods like nextInt(), nextDouble(), etc.

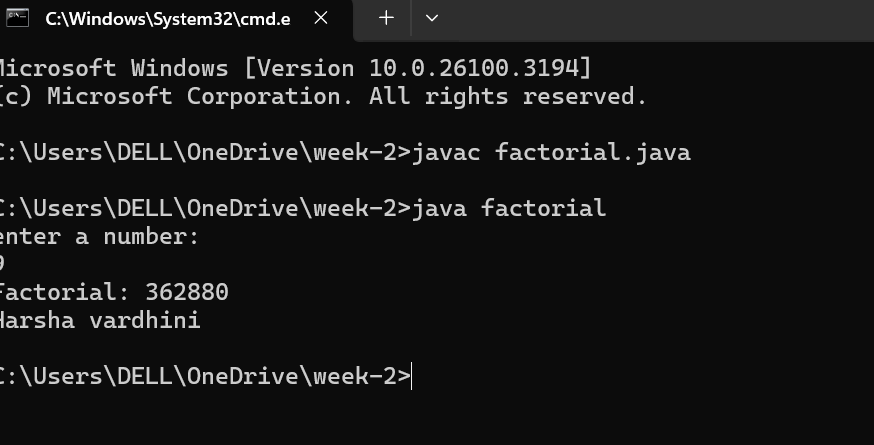
Program 2 :

Aim: write a java program to find the factorial of a number where alll the inputs are taken from a user.

Code:



Output:



Error table:

|  |  |
| --- | --- |
| Error | Rectification |
| factorial.java:1: error: ';' expected  import java. util. Scanner  ^  factorial.java:17: error: unclosed string literal  System.  out. Println("Harsha vardhini);  ^  2 errors | import java. util. Scanner;  semicolon and quotation should be there  System. out. println(“Harsha vardhini”); |

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.

Program - 3

Aim: write a program to convert the temperature from Celsius to Fahrenheit and Fahrenheit to Celsius

Code:

import java.util.Scanner;

class Temperature {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

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

float F = input.nextFloat();

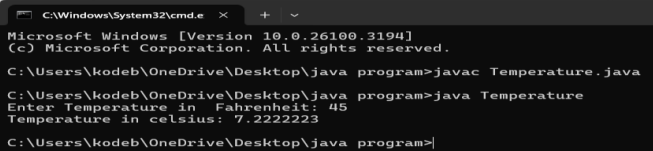
input.close();

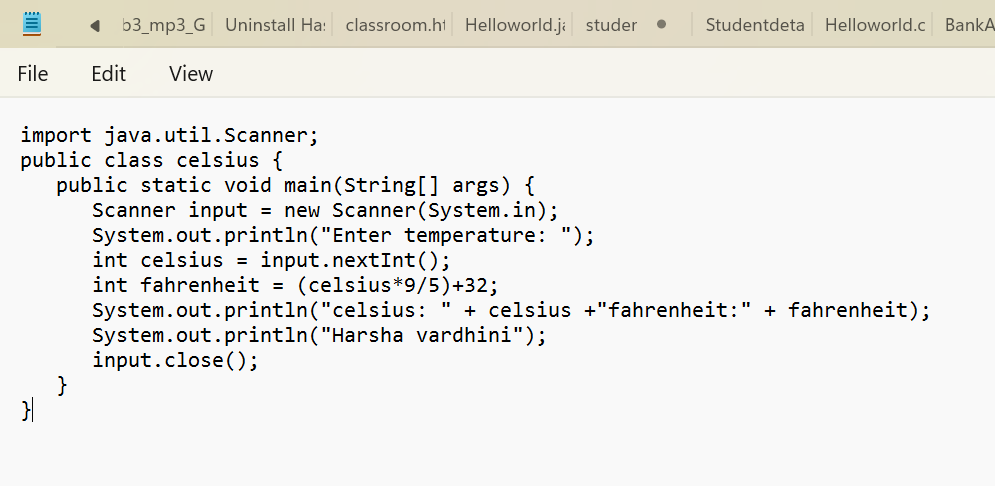
float C = (F - 32)\*5/9;

System.out.println("Temperature in celsius: " + C);

}

}

**OUTPUT: **

Code: 

Output:



Error table

|  |  |
| --- | --- |
| Error | Rectification |
| Error in line :3  System.out.println | S should be caapital |

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

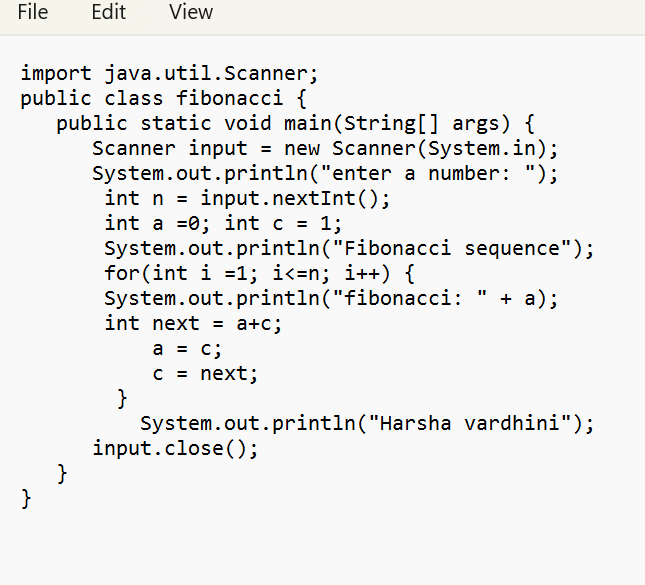
Fahrenheit = (Celsius\*9/5)+32.

1. 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.

Program 4:

Aim: write a java program to find the Fibonacci series of a given number where all the inputs are taken from the user

Code :



Output:



Error table:

|  |  |
| --- | --- |
| Error | rectification |
| fibonacci.java:8: error: ')' or ',' expected  System.out.println("Fibonacci sequence";  ^  1 error | Need to give parentheses |

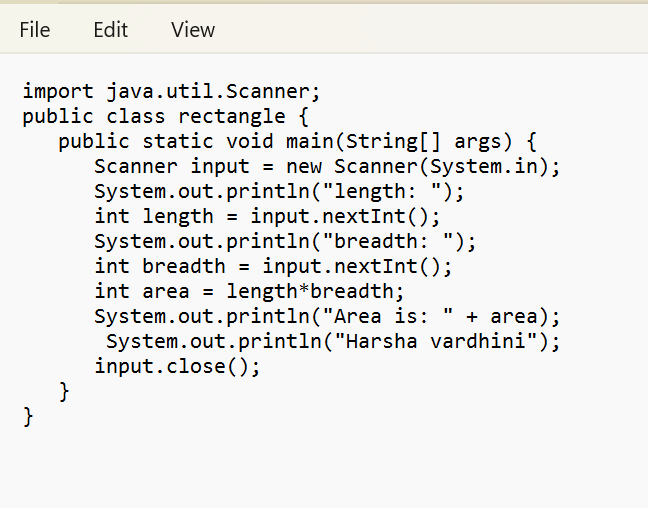
IMPORTANT POINTS:

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.

Program – 5:

AIM : write a java program to find the area of a rectangle, where all the inputs are taken from the user

Code :



Output:



Error table:

|  |  |
| --- | --- |
| Code error | Code rectification |
| While using for iteration, not  Giving the conditions correctly    Declaring the data type as double instead of int | We should give iterative statements correctly  We should give the data type  Type as int for integers |

IMPORTANT POINTS:

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

L = length of a side of the rectangle,

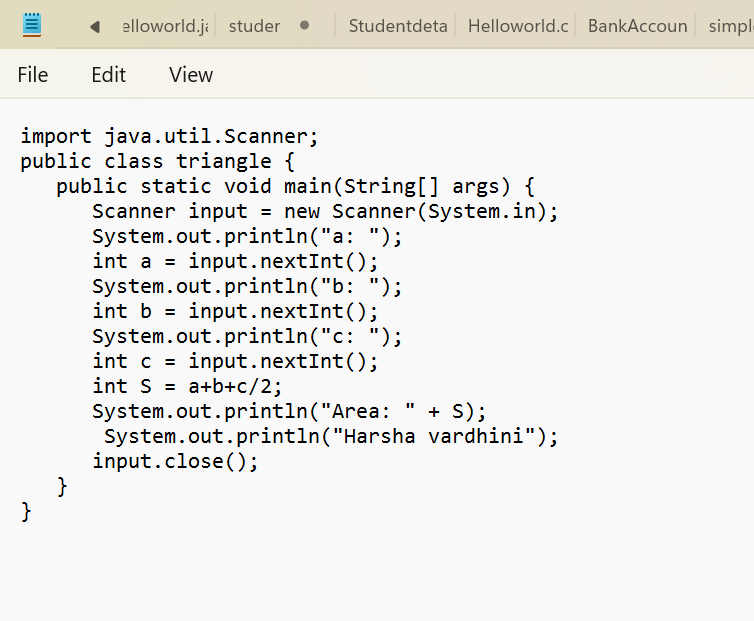
B= breadth of a side of the rectangle.

1. Here, we must be sure that all the expressions/conditions inside for the for loop must be given correctly.

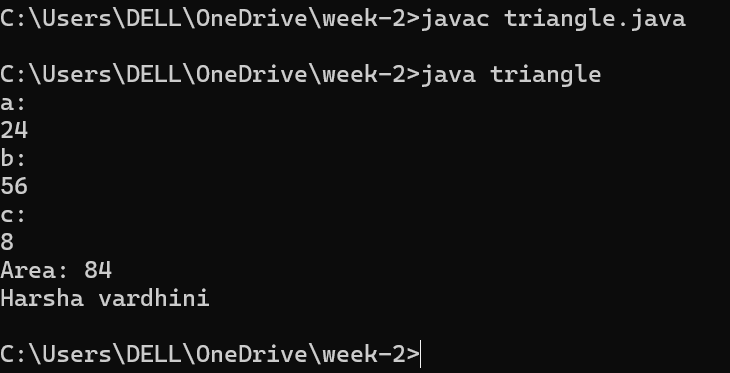
Program – 6

Aim : write a java program to find the area of the triangle using herons formula where inputs are taken from the user.

Code:



Output:



Error table:

|  |  |
| --- | --- |
| Code error | Code rectifiction |
| 1. While prining the variable not giving +sign 2. Not closing the scanner | 1. We should give correct   Indentaion   1. Closing the scanner is must |

IMPORTANT POINTS:

1. Here, we’re finding the area of a triangle using heron’s formula.
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)).

Week – 3

AIM: To create a java program with following instructructions

* Creating class with name car
* Creating 4 attributes named car\_color,brand,fueltype,milage
* Creating 3 methods named start(),stop() and service().
* Creating 3 objects named car1,car2 and car3 for the class car.

DEFINITIONS:

Class: Classes are blueprint of any number of objects

Method: Block of code which executes when they are called in main class

Object : Object is an instance of a class. The subclass elements or attributes can be accessed by using object and performing dot operation

Attribute: Named Objects with assigned values.

Class diagram:

|  |
| --- |
| Class car |
| -car\_ color: String  -car\_brand: String  -car\_fuel : String  -car\_milage:String |

|  |
| --- |
| +start(): void  +stop(): void  +service(): void |

PROGRAM:

Input: public class car {

private String car\_color;

private String car\_brand;

private String car\_fueltype;

private int car\_milage;

public void start() {

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

}

public void stop() {

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

}

public void service() {

System.out.println("car service");

}

public static void main(String args[]) {

car car1 = new car();

car1.car\_color = "blue";

car1.car\_brand ="SUPRA";

car1.car\_fueltype ="disel";

car1.car\_milage = 5;

car1.start();

car1.stop();

car1.service();

System.out.println("car color: " + car1.car\_color + " brand: " + car1.car\_brand + " fueltype: " + car1.car\_fueltype + " milage: " + car1.car\_milage);

car car2 = new car();

car2.car\_color = "yellow";

car2.car\_brand = "BUGATTI";

car2.car\_fueltype = "petrol";

car2.car\_milage = 6;

car2.start();

car2.stop();

car2.service();

System.out.println("car color: " + car2.car\_color + " brand: " + car2.car\_brand + " fueltype: " + car2.car\_fueltype + " milage: " + car2.car\_milage);

car car3 = new car();

car3.car\_color = "green";

car3.car\_brand = "PORSCHE";

car3.car\_fueltype = "disel";

car3.car\_milage = 13;

car3.start();

car3.stop();

car3.service();

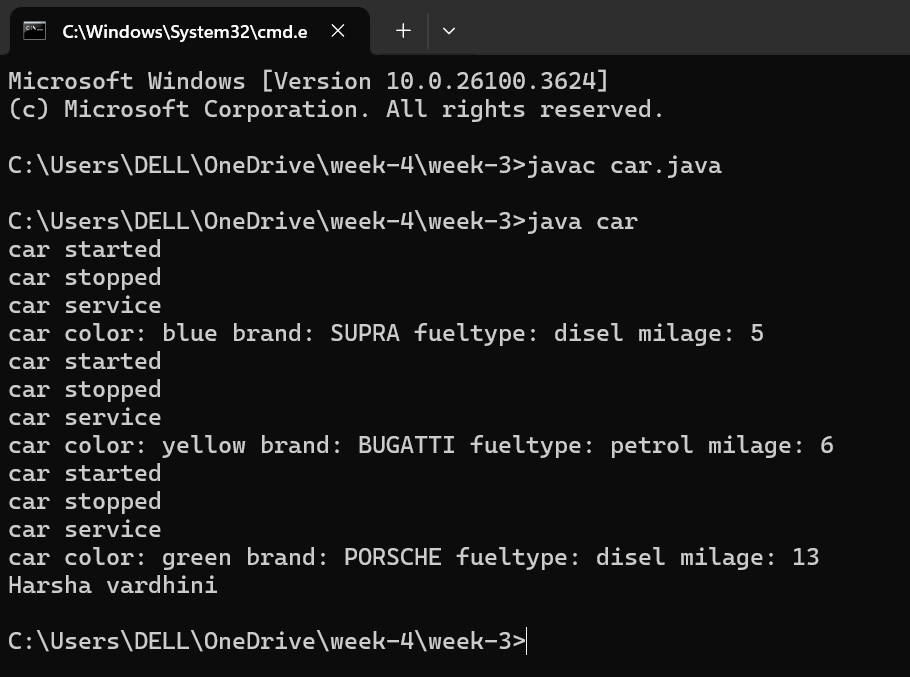
System.out.println("car color: " + car3.car\_color + " brand: " + car3.car\_brand + " fueltype: " + car3.car\_fueltype + " milage: " + car3.car\_milage);

System.out.println("Harsha vardhini");

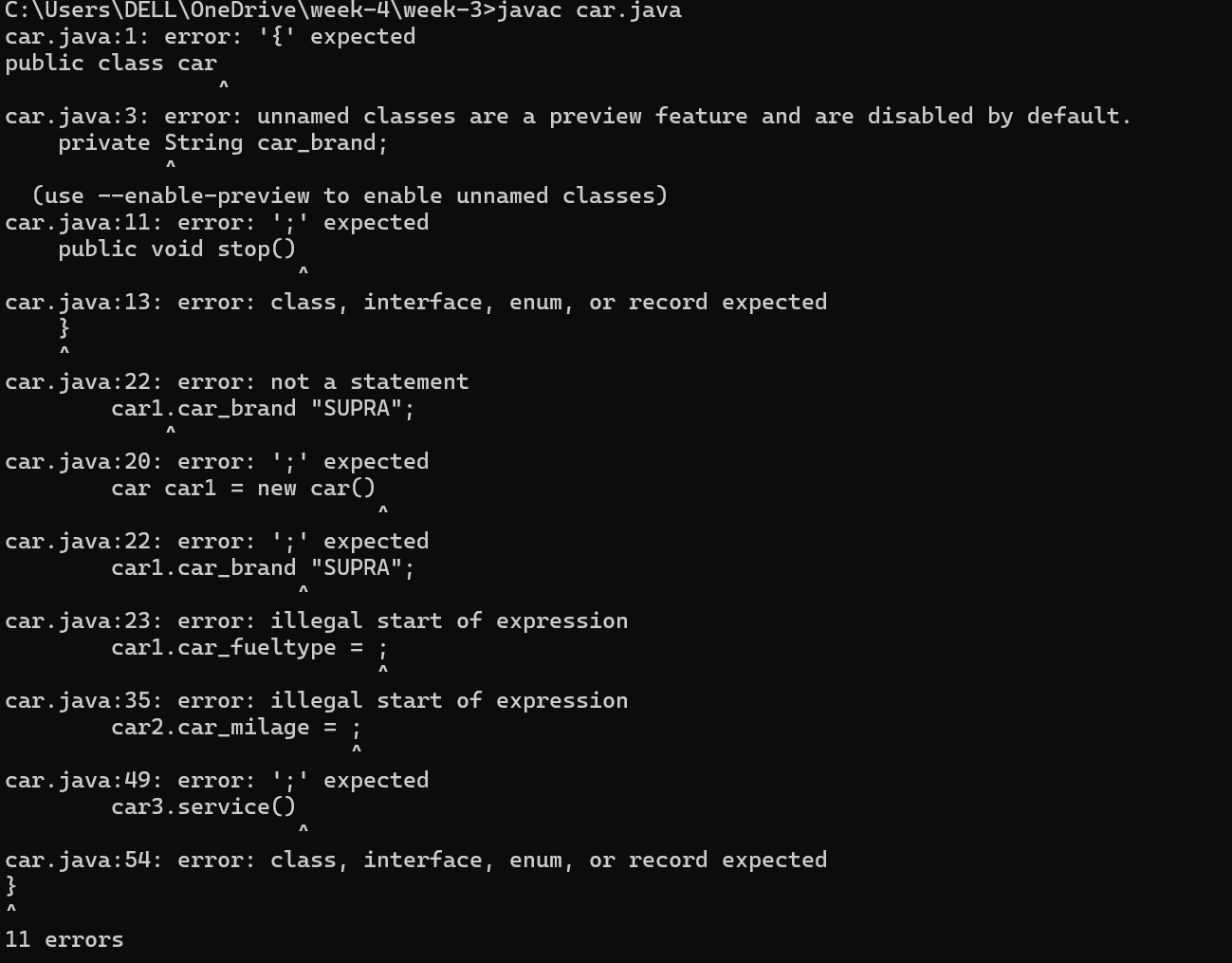
}

}

Output:

****

Negative case:



Error table

|  |  |
| --- | --- |
| Error | rectification |
| 1 . car.java:26: error: ';' expected  car1.stop():  ^  1 error  2. car.java:23: error: cannot find symbol  car1.car\_fueltyper = "disel";  ^  symbol: variable car\_fueltyper  location: variable car1 of type car | Need to keep semicolon instead I kept colon  Car1.stop();  Need to keep which car fuel we are reffering in the line  Car1\_fueltyper |

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

IMPORTANT POINTS :

Writing syntax for class,method and object creation using new keyword.

Syntax for Class: public class classname{ }

Syntax for Method: public returntype methodname(){ }

Syntax for Object: classname objectname = new classname();

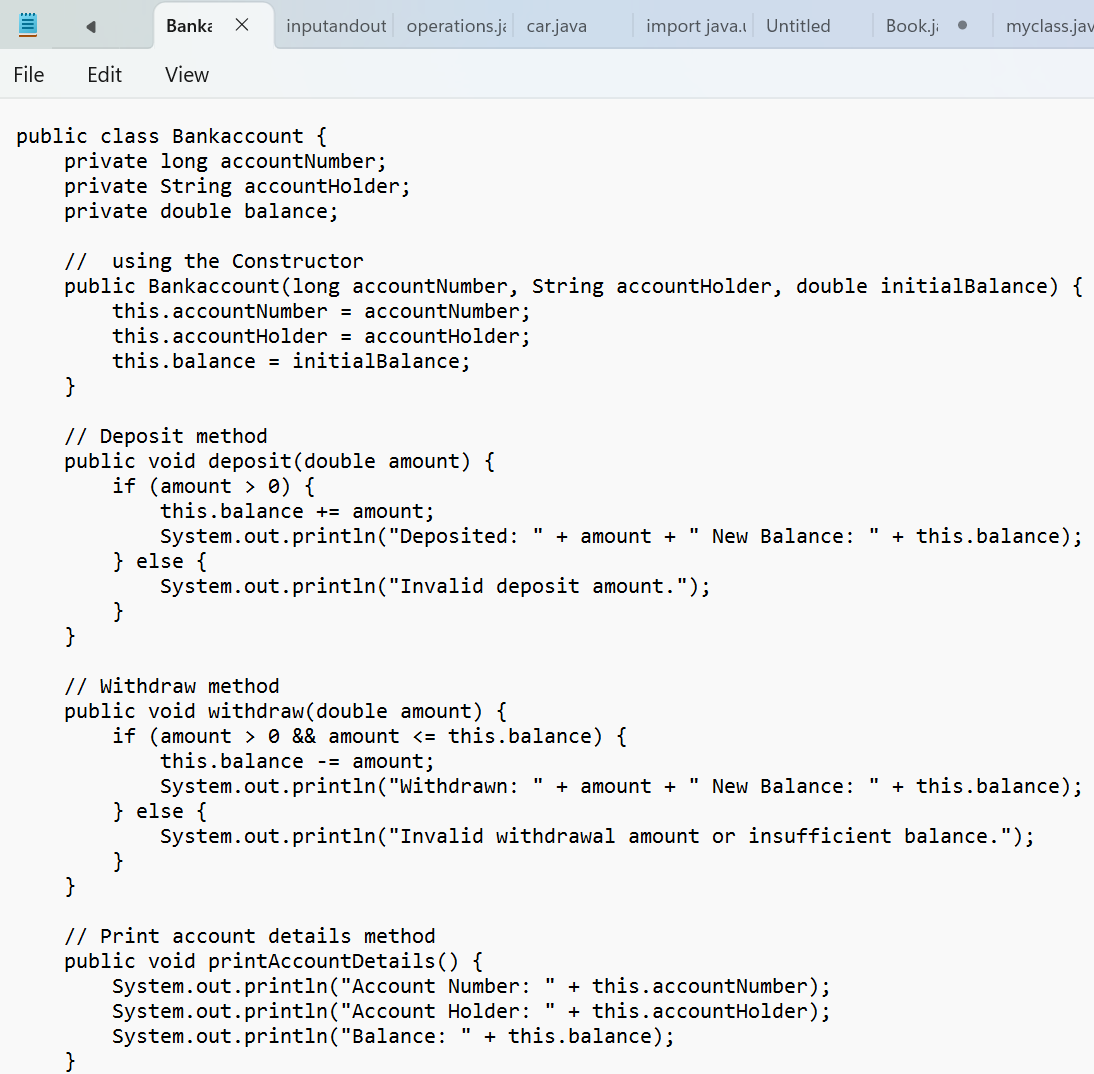
4.b)***AIM***: To create a class Bankaccount with methods deposit() and withdraw().create two subclasses savingsaccount and checkingaccount override the withdraw() method in each subclass to impose different withdrawal limits and fees

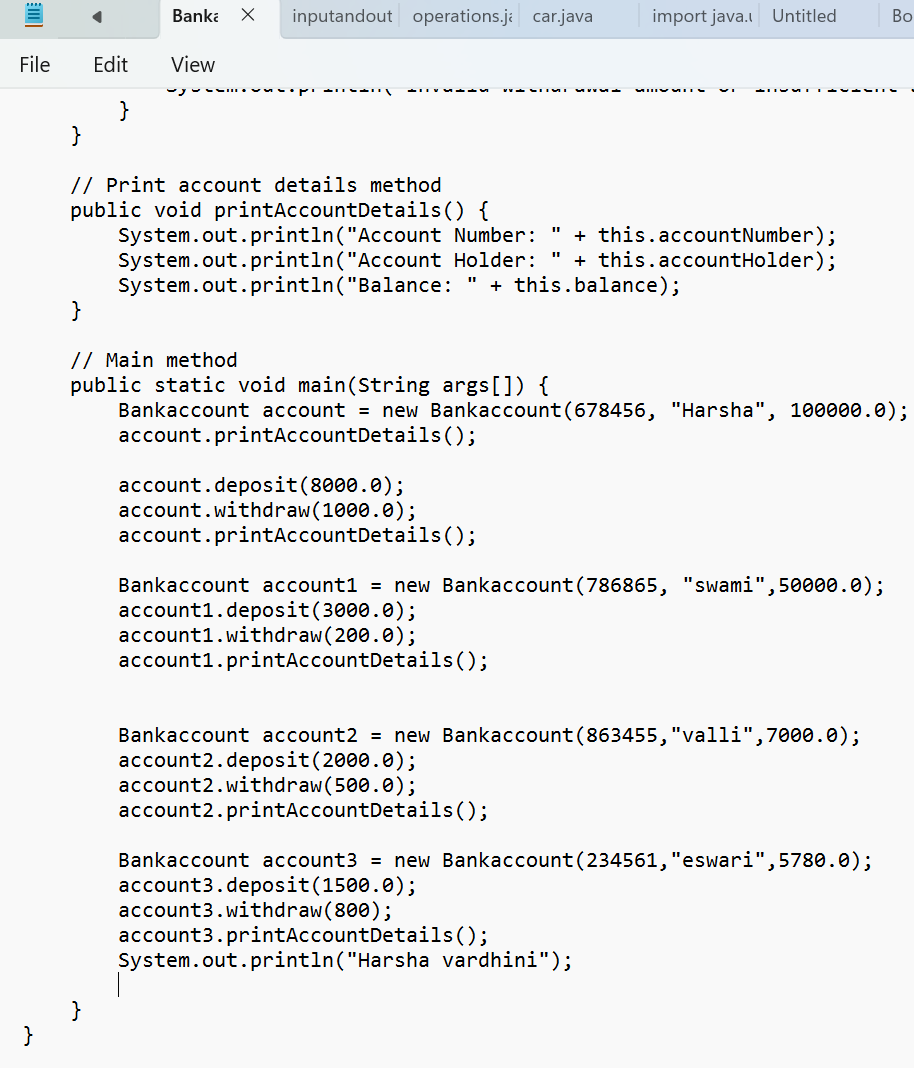
PROCEDURE:

Class diagram :

|  |
| --- |
| Class bank account |
| -ACCNAME : STRING  - ACCNO : int  -Currentbalance : float |
| +withdraw() :void  +deposit() : void |

Code :

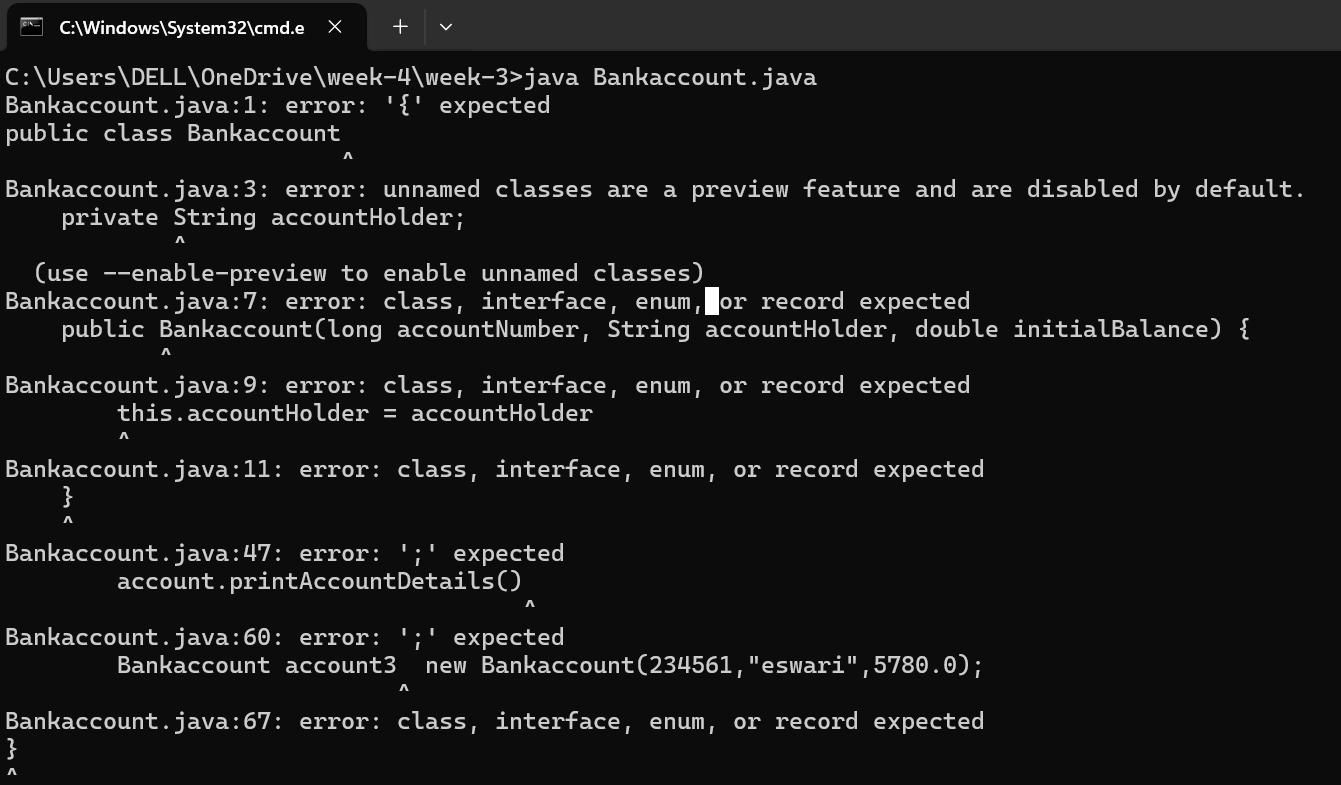




Output:



Negative case:



Error table:

|  |  |
| --- | --- |
| Error | rectification |
| Bankaccount.java:42:  error: illegal character:  Bankaccount account = new BankAccount(678456, ?Harsha", 100000.0);  ^ | Forgot to keep punctuation |

* Important points: **Encapsulation**:
  + Data members accountNumber, accountHolder, and balance are marked private to protect data.
  + Methods like deposit(), withdraw(), and printAccountDetails() provide controlled access to data.
* **Constructor Usage**:
  + The constructor initializes the object with custom values (account number, holder name, and initial balance).
* **Instance Methods**:
  + Actions that operate on individual objects (e.g., deposit, withdraw).

Week – 4

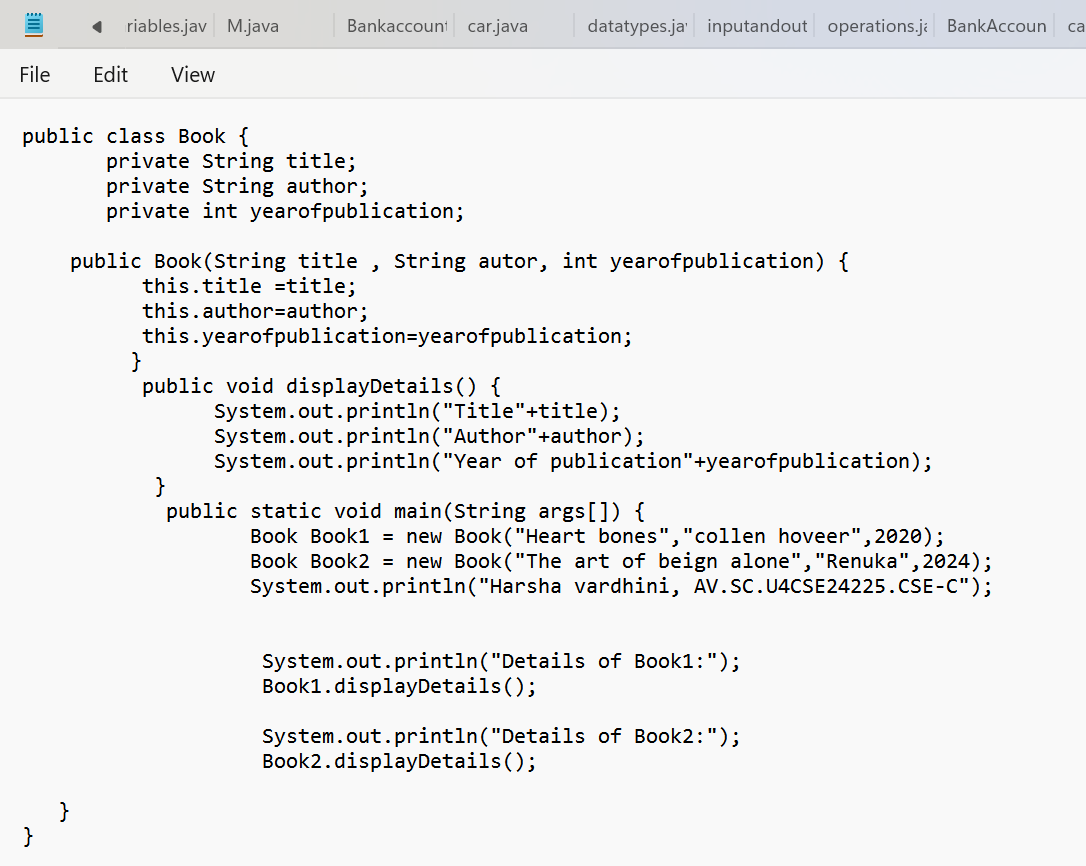
1. AIM: Write a java program with class named “book”. The class should contain various attributes such as “title, author, yearofpublication”. It should also contain a “constructor” with parameters which initializes “title”, ”author”, and “yearofpublication”.Create a method which displays the details of the book i.e. “author, title, yearofpublication”.(Display the details of two books i.e. create 2 objects and display their details).

Class diagram:

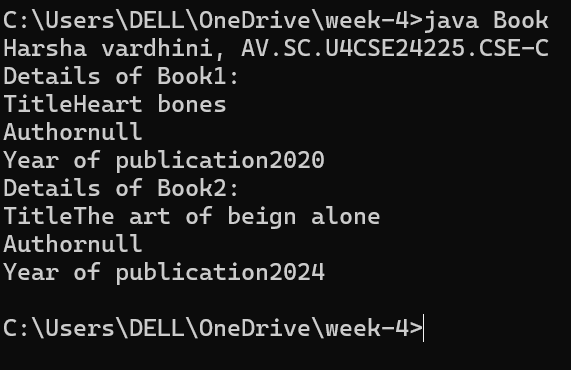
|  |
| --- |
| Book |

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

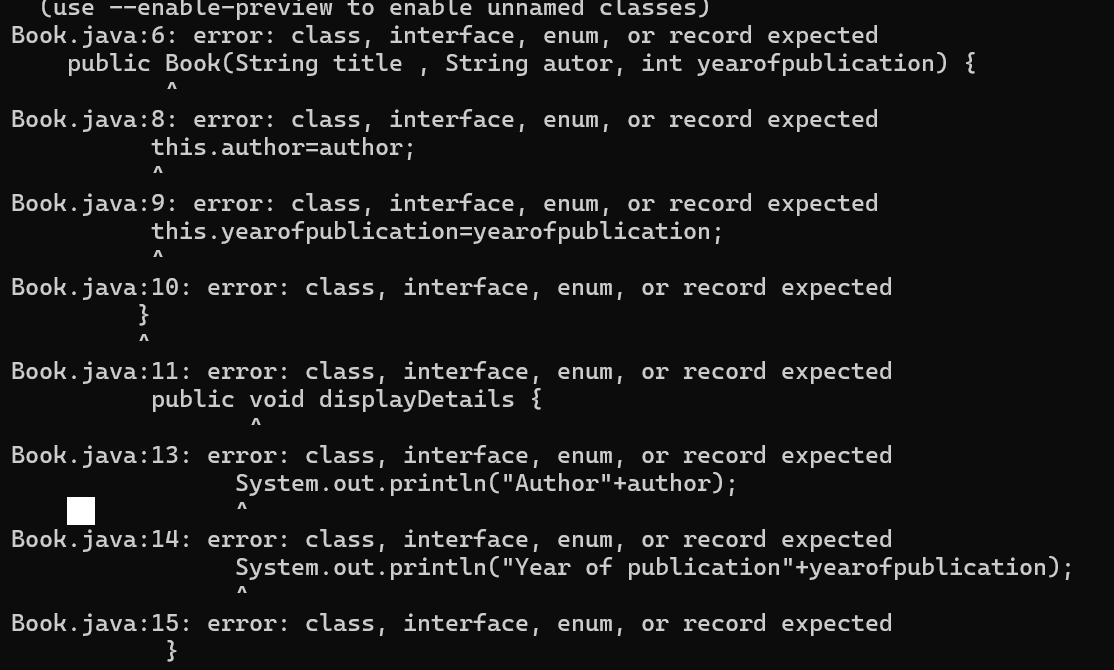
Program:



Output:

·

Negative case :



Error table:

|  |  |
| --- | --- |
| Error | Rectification |
| Book.java:19: error: ';' expected  System.out.println("Harsha vardhini, AV.SC.U4CSE24225.CSE-C")  ^  Book.java:23: error: ';' expected  Book1.displayDetails():  2 errors | After the line need to keep  Semicolon(;)  Book1.displayDetails(); |

**Class Definition:** The code defines a class named Book that represents a book. The class has three attributes: title, author, and year\_of\_publication.

· **Constructor:** The class has a constructor that initializes the attributes of a Book object. The constructor takes three arguments: title, author, and year\_of\_publication.

· **Book Details Method:** The class has a method named book\_details that prints the details of a book object. The method prints the title, author, and year of publication of the book.

· **Main Method:** The code has a main method that creates two instances of the Book class: one for the and one for the heart bones comic. The main method then calls the book\_details method on each book object to print its details.

Program -2

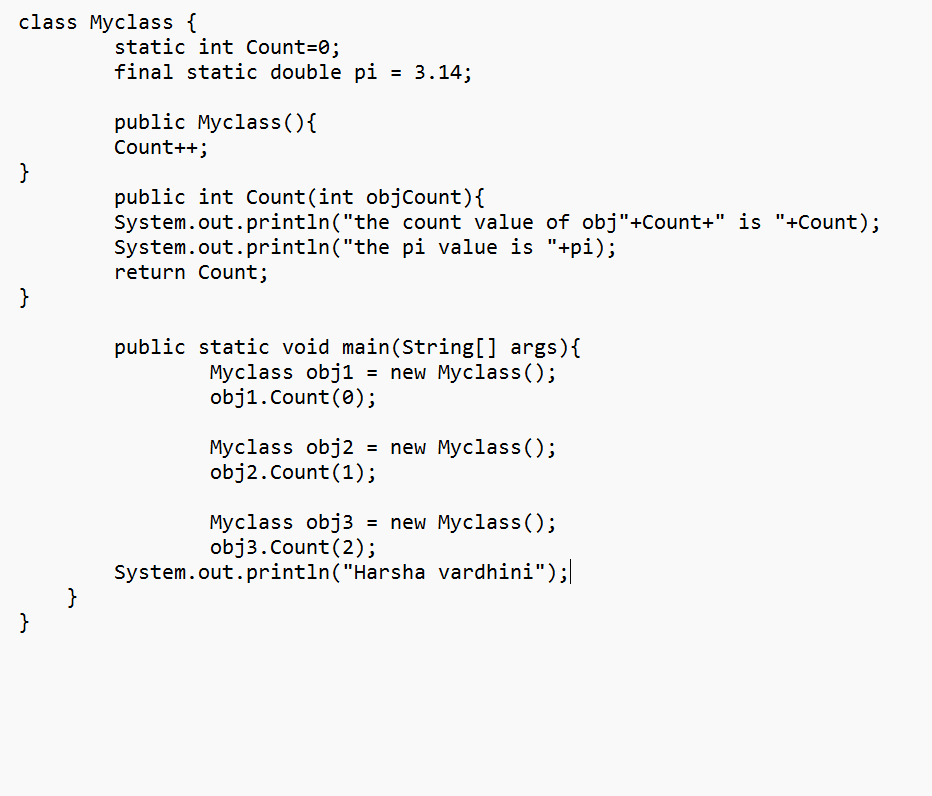
Aim: Write a java program with class named “MyClass”, with a static variable “count” of “int” type, initialized to “0” and a constant variable “PI” of type “double” initialized to 3.14159 as attributes of that class. Now define a constructor for “MyClass” that increments the “count” variable each time an object of “MyClass” is created. Finally, print the final values of “count” and “PI” variables.

Class diagram:

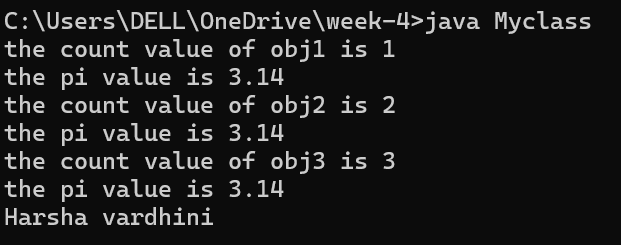
Class diagram:

|  |
| --- |
| My class |
| Count:int  Pi : double |
| +my class()  +main(args:String[]):void |

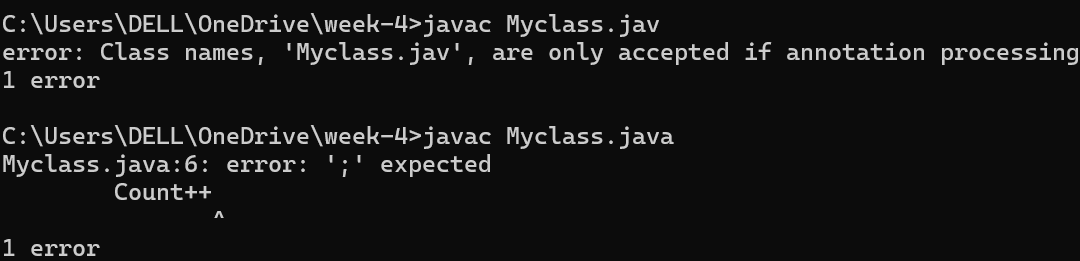
Code:



Output:



Negative case :



Error table:

|  |  |
| --- | --- |
| Code error | Code rectification |
| 1.not keeping semicolon  After calling a function | 1.put semicolon after calling a function |

**Important points:**

1. **Class Variable count:**
   * It's declared as static, meaning there's only one copy shared by all objects of the MyClass class.
   * It's initialized to 0, keeping track of the number of instantiated objects.
   * It's incremented by 1 in the constructor (MyClass()) whenever a new object is created.
2. **Constructor MyClass():**
   * It's a public constructor, accessible from anywhere in the code.
   * It increments the count variable to track object creation.
3. **Main Method:**
   * The main method is the entry point of the program.
   * It creates an object c1 of the MyClass class, triggering the constructor and incrementing count.

Week – 5

Aim : create a calculator using operations including addition, subtraction, multiplication and division using multilevel inheritance display desired output .Each class should have method it should pass a parameter.

Class diagram:

Class diagram :

|  |
| --- |
| Calculator |
| -a : double  -b : double |
| + calculator(a,b) |

|  |
| --- |
| Addition |
| +add(): double |

|  |
| --- |
| subtraction |
| +subtract() : double |

|  |
| --- |
| multiplication |
| +multiply() : double |

|  |
| --- |
| Division |
| +divide() : double |

Code:

class calculator {

    protected double a, b;

    public calculator(double a, double b) {

        this.a = a;

        this.b = b;

    }

}

class Addition extends calculator {

    public Addition(double a, double b) {

        super(a, b);

    }

    public double add() {

        return a + b;

    }

}

class Subtraction extends Addition {

    public Subtraction(double a, double b) {

        super(a, b);

    }

    public double subtract() {

        return a - b;

    }

}

class Multiplication extends Subtraction {

    public Multiplication(double a, double b) {

        super(a, b);

    }

    public double multiply() {

        return a \* b;

    }

}

class Division extends Multiplication {

    public Division(double a, double b) {

        super(a, b);

    }

    public double divide() {

        if (b != 0) {

            return a / b;

        } else {

            System.out.println("Error");

            return Double.NaN;

        }

    }

}

class Final extends Division {

    public Final(double a, double b) {

        super(a, b);

    }

    public void displayResults() {

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

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

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

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

    }

}

import java.util.Scanner;

public class allcalculator {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

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

        double a = input.nextDouble();

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

        double b = input.nextDouble();

        Final calc = new Final( a,  b);

        calc.displayResults();

        System.out.println("Harsha vardhini");

        input.close();

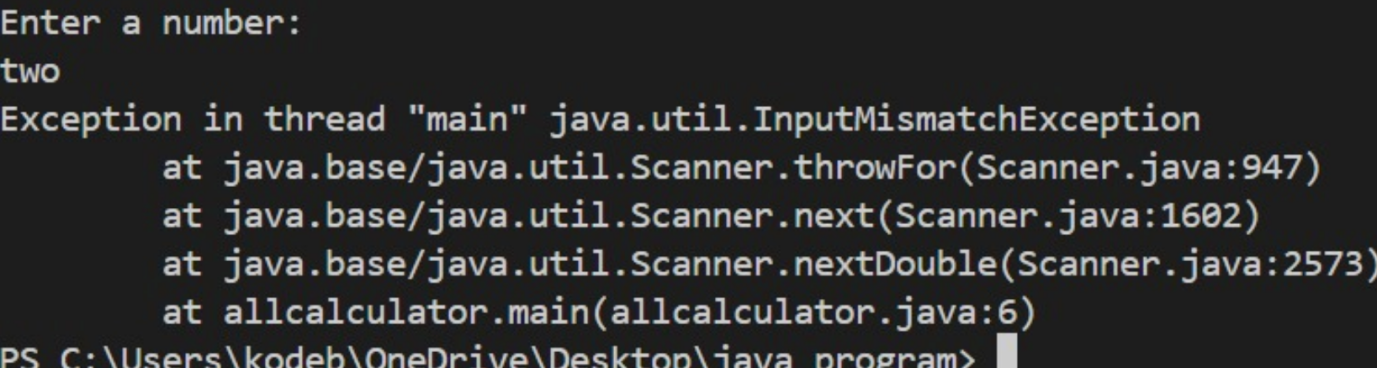
    }

}

Output:



Negative case:



Error table

|  |  |
| --- | --- |
| 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   1. 2. To obtain the super class we need to mention super. |

IMPORTANT POINTS:

1. To get the inputs from the user we use import java.util.Scanner; this is a package.
2. Scanner class is used to get the user input.
3. in java.util.Scanner, the java.util is a package while Scanner is a class of the java.util package.
4. to import a whole package, end the sentence with an asterisk sign(\*).

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.

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.

Code:

public class vehicle {

    public String brand;

    public int speed;

    public vehicle(String brand, int speed) {

        this. brand = brand;

        this .speed = speed;

    }

    public void start() {

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

    }

    public void show Details() {

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

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

    }

}

class Car extends vehicle {

    private int noOfDoors;

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

        super(brand, speed);

        this.noOfDoors = noOfDoors;

    }

    public void showDetails() {

        super.showDetails();

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

    }

}

class Bike extends vehicle {

    private boolean hasGears;

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

        super(brand, speed);

        this.hasGears = hasGears;

    }

    public void showDetails() {

        super.showDetails();

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

    }

}

class Truck extends vehicle {

    private int capacity;

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

        super(brand, speed);

        this.capacity = capacity;

    }

    public void showTruck() {

        super.showDetails();

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

    }

}

public static void main(String[] args) {

        Car car = new Car("Toyota", 150, 4);

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

        Truck truck = new Truck("Volvo", 90, 10);

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

        car.start();

        car.showDetails();

        System.out.println("Bike Details");

        bike.start();

        bike.showDetails();

        System.out.println("Truck Details");

        truck.start();

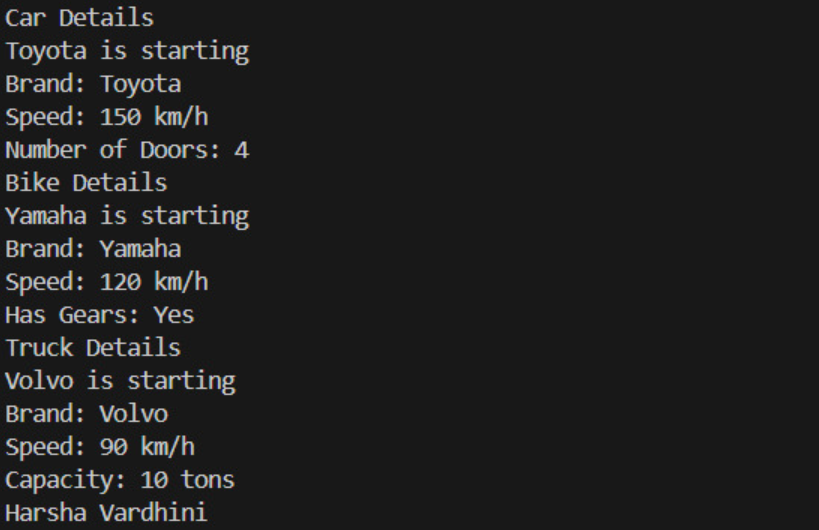
        truck.showTruck();

        System.out.println("Harsha vardhini");

    }

}

Output:



|  |
| --- |
| Vehicle   * Brand : str * Speed : int   + in it (brand ,speed)  +start\_vechicle()  + display\_details() |

|  |
| --- |
| Car |
| * No.   of. doors: int |
| + in it ( brand ,speed  display details()  ,no. of. doors ()  +display details()  + show truck details () |

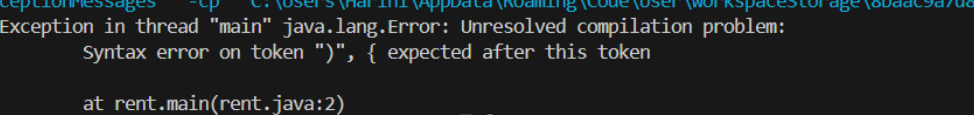
|  |
| --- |
| Bike |
| * Has gears : bool |
| +int (brand, speed, has gears)  + displaydetails() |

|  |
| --- |
| truck |
| * Capacity :float |

Error code :

|  |  |
| --- | --- |
| **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. |

Negative case:



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. a double method can represent more decimal point numbers than float method.
4. 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.

The oops concepts used in the above program are:

Inheritance, encapsulation, polymorphism, abstraction.

To add a new vehicle type truck we need to create a truck class that will:

* Include an additional property capacity (in tons).
* Implement a showtruckdetials() method to display the truck's capacity.
* Implement a constructor for the truck class to initialize all its properties.

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 |
| * Brand : String * Speed: int |
| + Vehicle(brand :string,  Speed: int)  +startVechicle(): void  +displatDetails() : void |

CODE :

public class Vehicle {

public String carmodel;

public String color;

public String fuel\_type;

public Vehicle(String carmodel, String color, String fuel\_type){

this.carmodel = carmodel;

this.color = color;

this.fuel\_type = fuel\_type;

}

public void displayinfo() {

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

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

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

System.out.println("I own a BMW.");

}

}

class Car extends Vehicle {

public Car(String carmodel, String color, String fuel\_type) {

super(carmodel, color, fuel\_type);

}

public void displayinfo() {

super.displayinfo();

System.out.println("I own an Audi.");

}

}

public class Vehicletest {

public static void main(String[]args) {

System.out.println("Harsha vardhini");

Vehicle v = new Vehicle("Audi", "White", "Petrol");

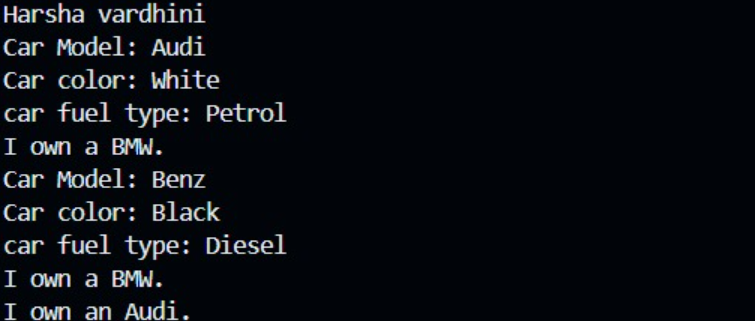
v.displayinfo();

Car c = new Car("Benz", "Black", "Diesel");

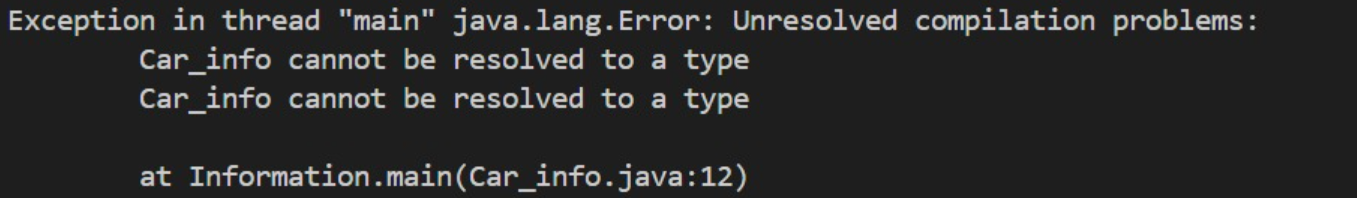
c.displayinfo();

}

Output :



Negative case :



ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Not writing the super method. 2. Inconsistent car model output in displayinfo(). | 1. Write the super method to get the variables of the superclass. 2. Ensure that car correctly passes to the models. |

IMPORTANT POINTS:

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

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

**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.

**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 admissions systems that verifies students eligibility for undergraduate(UG) and postgraduate(PG) programs. Each program has different eligibility. Criteria based on the students percentage in their previous qualification.

1. UG admission require min of 60%
2. PG admission require min of 70%

Code:

import java.util.Scanner;

public class Admission {

public static void main(String[]args) {

Scanner input = new Scanner(System.in);

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

String name = input.nextLine();

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

double percentage = input.nextDouble();

input.nextLine();

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

String program = input.nextLine();

if (program.equals("UG")) {

if (percentage>=60) {

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

} else {

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

}

} else if (program.equals("PG")) {

if (percentage>= 70) {

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

} else {

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

}

} else {

System.out.println("Invalid course entered.");

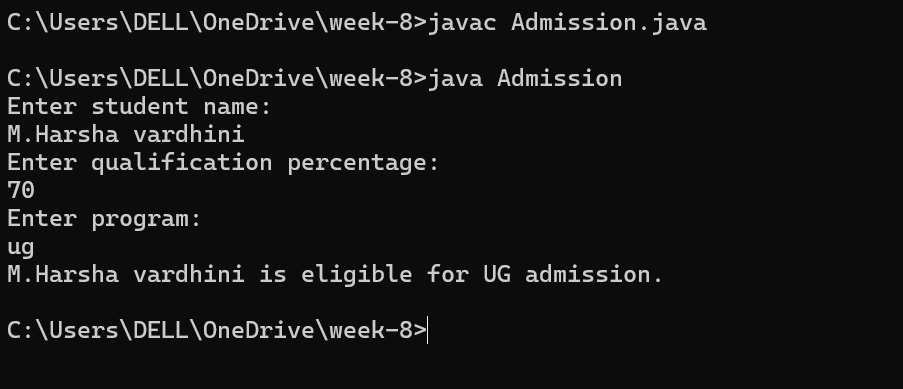
}

input.close();

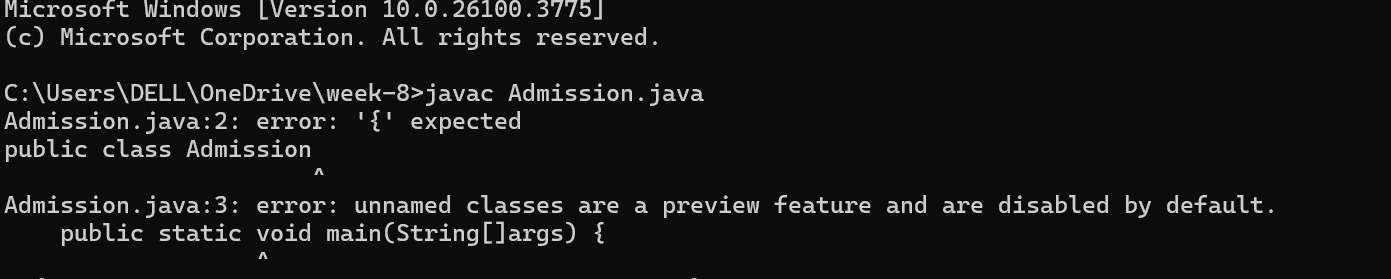
}

}

Output:



Negative case :



Class diagram:

|  |
| --- |
| Admission |
| - scanner: Scanner  - name: String  - percentage: double  - program: String |
| + main(args: String[]): void  + takeInput(): void  + checkEligibility(): void  + closeScanner(): void |

ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Scanner nextLine() issue after nextDouble(): after input.nextDouble(), the newline character remains in the buffer, causing nestLine() 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. To UpperCase() 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 input: properly closes scanner using scanner.close(); to prevent resource leaks.

PROGRAM – 3

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

1. Add two integers
2. Add two double
3. Add three integer

Code:

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public double add(double a, double b) {

return a + b;

}

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

return a + b + c;

}

}

public class caltest {

public static void main(String[]args) {

Calculator c = new Calculator();

System.out.println("Addition of two integers: " + c.add(2,4));

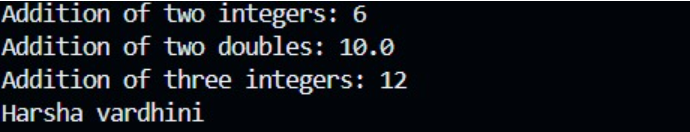
System.out.println("Addition of two doubles: " + c.add(3.5,6.5));

System.out.println("Addition of three integers: " + c.add(3,4,5));

System.out.println("Harsha vardhini");

}

}

Output:

Class diagram:

|  |
| --- |
| Calculator |
| +add(int, int): int  +add(double, double): double  +add(int, int, int): int  +main(String[]): void |

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 types during compilation.

PROGRAM – 4

AIM

Create a shape class with a method CalculateArea() that is overloaded for different shpaes (e.g square, rectangle) then, create a subclass circle that overrides the calculatearea() method for a circle.

CODE:

public class Shape {

public double calculatearea(double s) {

return s\*s;

}

public int calculatearea(int l, int b) {

return l\*b;

}

}

class circle extends Shape {

public double calculatearea(double s) {

return (3.14\*s\*s);

}

}

public class shapetest {

public static void main(String[]args) {

Shape s = new Shape();

System.out.println("Area of square: " + s.calculatearea(3.5));

System.out.println("Area of a rectangle: " + s.calculatearea(6,3));

circle c = new circle();

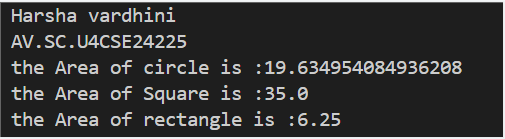
System.out.println("Area of circle: " + c.calculatearea(2.5));

System.out.println("Harsha vardhini");

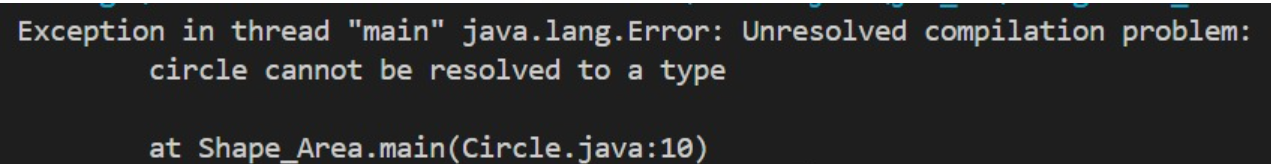
}

}

OUTPUT:



Negative case:



ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Method calls in main are missing an object reference(e.g., calculate Area(a) instead of s.calculatearea(a). 2. Circle class method does not override theparent class method properly. | 1. Use s.calculatearea(4) and c.calculatearea to call the method correctly. 2. Ensure@override is used, when certain error shows sometimes. |

Class diagram:

|  |
| --- |
| Shape |
| +calculateArea(side: double): double  +calculateArea(width: double, length: double): double |

|  |
| --- |
| CIRCLE |
| +calculateArea(radius: double): double |

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

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 subclasses Lion and tiger that extends animal class and implement the sound() method to make a specific sound for each animal.

Class diagram:

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

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

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

Code:

abstract class Ani {

abstract void sound();

}

class Lion extends Ani {

@Override

void sound() {

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

}

}

class Tiger extends Ani {

@Override

void sound() {

System.out.println("tiger grils");

}

}

public class Animalsound {

public static void main(String[] args) {

System.out.println("Name : Harsha vardhini");

System.out.println("Roll number : AV.SC.U4CSE24225");

System.out.println("Section : CSE C");

System.out.println();

Lion l1 = new Lion();

l1.sound();

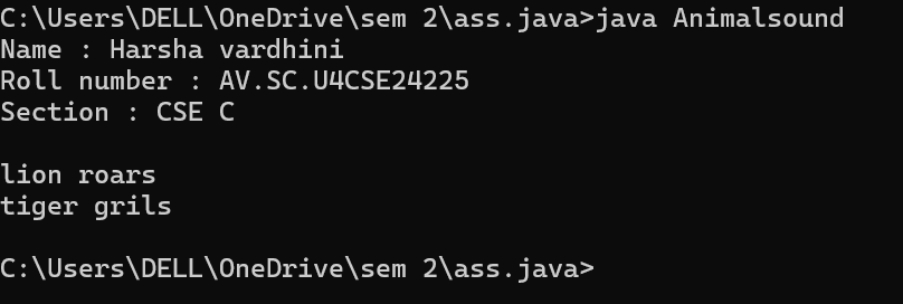
Tiger t1 = new Tiger();

t1.sound();

}

}

Output:



|  |  |
| --- | --- |
| Error Found | Error Rectification |
| In the code created a obj for animal class | I removed obj for animal class |

Important points:

Use of Abstract Class for Generalization

The Animal class is abstract and defines a common method sound() for all animals, which ensures that all subclasses provide their own specific implementation.

Method Implementation in Subclasses (Lion & Tiger)

Subclasses Lion and Tiger override the abstract method sound() to give specific behavior, showing how abstract classes enforce method implementation in child classes.

Polymorphism in Action

By using Animal references to call sound() on Lion and Tiger objects, the program demonstrates runtime polymorphism, allowing the program to decide which method to call at execution time

**Program no : 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 :**

|  |
| --- |
| Shape 3D |
| Calculate Volume: double()  Calculate surface area |

|  |
| --- |
| Sphere |
| Radius : double() |
| Sphere : radius()  Calculate volume: double() |

|  |
| --- |
| Cube |
| Side : double() |
| Cube: side()  Calculate volume: double() |

**CODE:**

abstract class Shape3D {

abstract double calculateVolume();

abstract double calculateSurfaceArea();

}

class Sphere extends Shape3D {

double radius;

Sphere(double radius) {

this.radius = radius;

}

@Override

double calculateVolume() {

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

}

@Override

double calculateSurfaceArea() {

return 4 \* Math.PI \* Math.pow(radius, 2);

}

}

class Cube extends Shape3D {

double side;

Cube(double side) {

this.side = side;

}

@Override

double calculateVolume() {

return Math.pow(side, 3);

}

@Override

double calculateSurfaceArea() {

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

}

}

public class Sphe {

public static void main(String[] args) {

System.out.println("Name : Harsha vardhiniI");

System.out.println("Roll number : AV.SC.U4CSE24225");

System.out.println("Section : CSE C");

System.out.println();

// Updated values

Shape3D sphere = new Sphere(5.5);

Shape3D cube = new Cube(3.2);

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

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

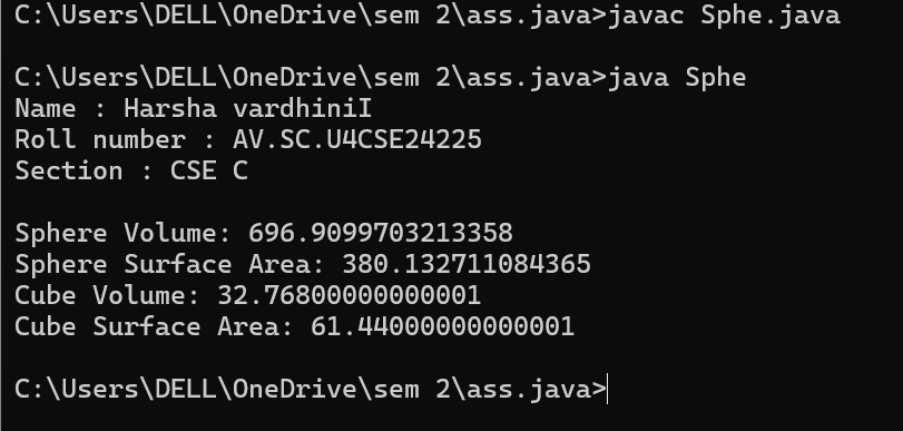
System.out.println("Cube Volume: " + cube.calculateVolume());

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

}

}

Output:



**ERRORS:**

|  |  |
| --- | --- |
| Error Found | Error Rectification |
| The code is Math.Pow | The rectification code is Math.pow |
| In the code I didn’t mention return. | I mentioned return. |

**Important points**:

Each subclass (Sphere and Cube) provides its own **specific implementation** of the methods to calculate the volume and surface area using the formulas appropriate for each 3D shape.

By using Shape3D references to store both Sphere and Cube objects, we demonstrate **runtime polymorphism**, allowing us to call calculateVolume() and calculateSurfaceArea() on the base class type, but the correct method is executed based on the actual object t

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

•Implement two subclasses :

1.Star Pattern - prints a right angled triangle of stars(\*)

2.Number Pattern – prints a right angled triangle of increasing numbers. In the main() method, Create objects of both subclasses and print the patterns for a given no.of rows.

Class diagram:

|  |
| --- |
| PatternPrinter |
| -n: int  - title: string |

|  |
| --- |
| StarPattern |
| + PrintPattern(n: int): void |

|  |
| --- |
|  |

**CODE:**

abstract class PatternPrinter {

    public int n;

    private String title;

abstract void PrintPattern(int n)

public void Patterntitle(String title) {

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

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

    }

}

class StarPattern extends PatternPrinter {

    public void PrintPattern(int n) {

        this.n = n;

        for (int i = 0; i <= n; i++) {

            for (int j = 0; j <= i; j++) {

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

            }

            System.out.println(); // Move to the next line after inner loop

        }

    }

}

class NumberPattern extends PatternPrinter {

    public void PrintPattern(int n) {

        this.n = n;

        for (int i = 0; i <= n; i++) {

            for (int j = 0; j <= i; j++) {

                System.out.print(j + 1 + " ");

            }

            System.out.println(); // Move to the next line after inner loop

        }

    }

}

class Pattern {

    public static void main(String[] args) {

        StarPattern s = new StarPattern();

        s.Patterntitle("StarPattern");

        s.PrintPattern(5);

        NumberPattern num = new NumberPattern();

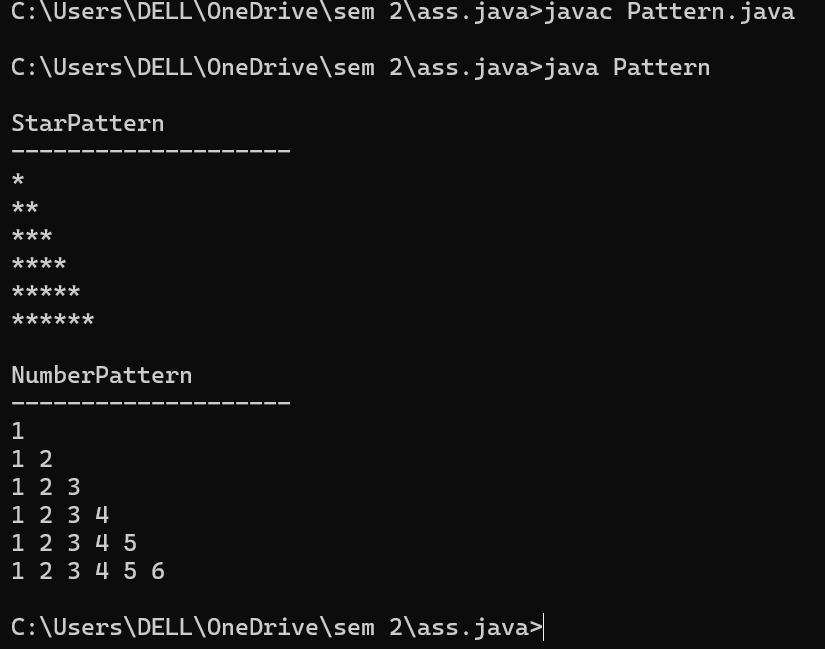
        num.Patterntitle("NumberPattern");

        num.PrintPattern(5);

    }

}

**OUTPUT:**

****

IMPORTANT POINTS:

1. An abstract method printPattern(int n) that must be implemented by subclasses.

2. Both StarPattern and NumberPattern extend PatternPrinter and provide specific implementations for printPattern(int n).

ERROR TABLE:

|  |  |
| --- | --- |
| ERROR FOUND | RECTIFICATION |
| Forgot to keep (int n) | Semicolon (int n); |

Week – 8

**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:**

|  |
| --- |
| shape |
| Getperimeter(): (int,int,int) |

|  |
| --- |
| Triangle |
| getPerimeter():  (in t, in t ,in t) |

|  |
| --- |
| **R**ectangle |
| getPerimete():  (int, int ,int) |

|  |
| --- |
| Circle |
| getPerimeter() :  (in t ,in t, in t) |

**CODE:**

interface Shape {

double getPerimeter(int a, int b, int c);

}

class Circle implements Shape {

public double getPerimeter(int a, int b, int c) {

// let 'a' be the radius of the circle

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

System.out.println("The perimeter of the circle: " + perimeter);

return (double)perimeter;

}

}

class Rectangle implements Shape {

public double getPerimeter(int a, int b, int c) {

// let 'a' be length and 'b' be width

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

System.out.println("The perimeter of the rectangle: " + perimeter);

return perimeter;

}

}

class Triangle implements Shape {

public double getPerimeter(int a, int b, int c) {

// let 'a', 'b', and 'c' be the sides of the triangle

int perimeter = a + b + c;

System.out.println("The perimeter of the triangle: " + perimeter);

return perimeter;

}

}

public class peri {

public static void main(String[] args) {

System.out.println("Name : Harsha vardhini");

System.out.println("Roll number : AV.SC.U4CSE24225");

System.out.println("Section : CSE C\n");

Circle c1 = new Circle();

c1.getPerimeter(4,0,0); // radius = 4

Rectangle r1 = new Rectangle();

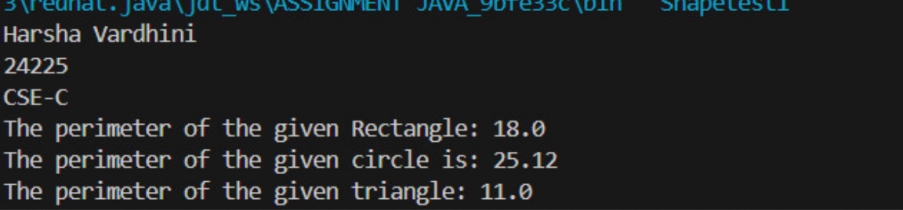
r1.getPerimeter(3,6 0); // length = 3, width = 6

Triangle t1 = new Triangle();

t1.getPerimeter(2,4,5); // sides = 2,4,5

}

Output:



**ERRORS:**

|  |  |
| --- | --- |
| Error Found | Error Rectification |
| In the code I wrote perimeter = a + b + c; | I rectified it as int perimeter = a + b + c; |

**IMPORTANT POINTS:**  
  
1.All classes correctly implement the Shape interface and override the getPerimeter(int a, int b, int c) method returning double.

2.In Circle, only the first parameter a (radius) is used; b and c are ignored but required due to the interface method signature.

**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:**

|  |
| --- |
| playable |
| Play: void() |

|  |
| --- |
| Basket ball |
| Play : void() |

|  |
| --- |
| Volley ball |
| Play: void() |

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

**Code:**

interface Playable {

public void play();

}

class FootBall implements Playable {

public void play() {

System.out.println("Kicking off an intense Football match!");

}

}

class VolleyBall implements Playable {

public void play() {

System.out.println("Serving the ball high in an exciting Volleyball rally!");

}

}

class BasketBall implements Playable {

public void play() {

System.out.println("Dribbling down the court in a thrilling Basketball game!");

}

}

public class Allplayer {

public static void main(String[] args) {

System.out.println("Name : Harsha vardhini");

System.out.println("Roll number : AV.SC.U4CSE24225");

System.out.println("Section : CSE C");

System.out.println();

FootBall f1 = new FootBall();

f1.play();

VolleyBall v1 = new VolleyBall();

v1.play();

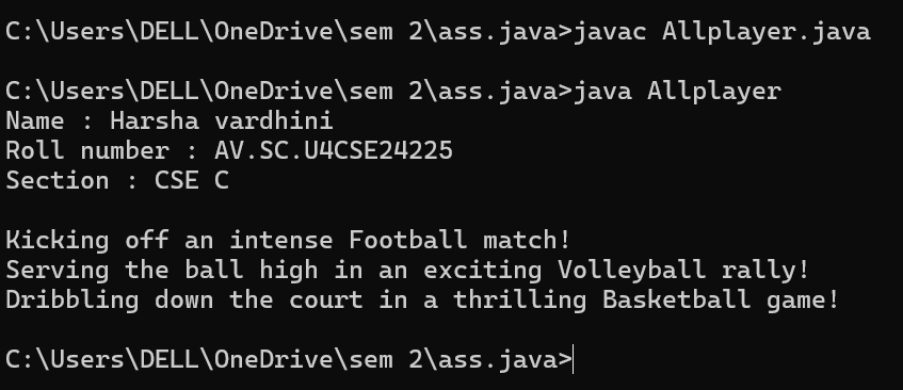
BasketBall b1 = new BasketBall();

b1.play();

}

}

Output:



Error table:

|  |  |
| --- | --- |
| Error Found | Error Rectification |
| The code is System.out.println(“……”); | The rectification code is System.out.println(“….”); |
| In the code I wrote Football f1 = new football(); | I rectified it as Football f1 = new Football(); |

Important points:

**Interface Implementation:**

* The program uses the Playable **interface** to define a common contract (play() method) that all sports classes (Football, Volleyball, Basketball) must follow.
* This ensures consistency and enforces a shared behavior across different sport types.

**Polymorphism:**

* Objects of different classes are referred to using the **interface type** Playable, allowing us to call the play() method on any sport object without knowing its exact class.
* This demonstrates **polymorphism** and makes the code flexible and easy to extend.

**Easy Extensibility:**

* Adding new sports is simple: just create a new class that implements Playable and define the play() method.
* This design follows good **object-oriented principles** and makes the program scalable.