**23CSE111**

**OBJECT ORIENTED PROGRAMMING**

**LAB REPORT**



**Department of Computer Science Engineering**

**Amrita School of Computing**

**Amrita Vishwa Vidyapeetham, Amaravati Campus**

**Name: KASI HARINI**

**Verified By Roll No: AV.SC.U4CSE24203**

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SNo | Title | Date | Page No | Signature |
| WEEK 1 |  |  |  |  |
| 1 | Download and Install Java Software |  | 7-10 |  |
| 2 | Write a Java program to print the message “Welcome to Java Programming.” |  | 11-12 |  |
| WEEK 2 |  |  |  |  |
| 1 | Write a Java program to calculate area of rectangle. |  | 13-14 |  |
| 2 | Write a java program to calculate the area of the triangle. |  | 14-15 |  |
| 3 | Write a Java program to convert temperature from Fahrenheit to Celsius |  | 15-17 |  |
| 4 | Write a Java program to convert temperature from Celsius to Fahrenheit. |  | 18-19 |  |
| 5 | Write a Java program to calculate factorial of a number. |  | 19-20 |  |
| 6 | Write a Java program to calculate fibonacci of a number. |  | 21-22 |  |
| 7 | Write a Java program to calculate Simple interest. |  | 22 |  |
| 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. |  | 23-25 |  |
| 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(). |  | 26-27 |  |
| 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. |  | 28-29 |  |
| 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. |  | 30-31 |  |
| 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 |  |  |  |  |
|  | 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. |  |  |  |
|  | 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. |  |  |  |
|  | write a java program using an bsstract class to define a method for pattern printing.  . create an abstract class named PatternPrinter with an abstract method printPattern(int n) and concrete method to display the pattern title.  . implement two subclasses:  1. StarPattern - prints a right-angled triangle of stars(\*).  2. NumberPattern - prints a right-angled triangle of increasing numbers.  . int the main() method, create objects of both subclasses and print the patterns for a given number of rows. |  |  |  |
| WEEK-8 |  |  |  |  |
|  | 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. |  |  |  |
|  | 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 01**

PROGRAM-1:

AIM: Download and Install Java Software

PROCEDURE:

Step 1: Download JDK 21

1. Open your web browser and go to the Oracle JDK Downloads page
2. Scroll down to the Java SE Development Kit 21 section.
3. Choose the Windows x64 Installer version.
4. Click on Download, then Wait for the download to complete.



Step 2: Install JDK 21

1. Locate the downloaded jdk-21\_windows-x64\_bin.exe file.
2. Double-click to launch the installer.
3. Click Next on the setup wizard.
4. Choose the installation path (default is C:\Program Files\Java\jdk-21).
5. Click Next, then click Install.
6. Wait for the installation to complete.
7. Click Close once the installation is finished.



Step 3: Setting up the path

1) Go to “Windows C” Drive on Desktop

2) Choose Program Files, select Java, then JDK 21, then select Bin.

3) Select and copy the path at the address bar.



Step 4: Open System Properties

1. Press Windows + R, type sysdm.cpl , and click Ok-
2. The System Properties window will open.
3. Navigate to the Advanced tab.
4. Click on Environment Variables at the bottom.



Step 5: Set JAVA\_HOME

1) Under System Variables, click New.

2) Set the Variable name as JAVA\_HOME.

3) Set Variable value as C:\Program Files\Java\jdk-21 (or your installation path).

4) Click OK.



Step 6: Update PATH Variable

1) In System Variables, find Path and click Edit.

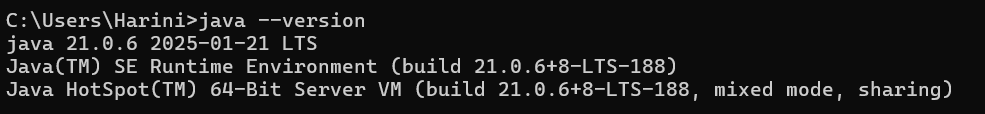
2) Click New and add: C:\Program Files\Java\jdk-21\bin

3) Click OK to save.



Step 7:Verify Installation

1. Open Command Prompt.
2. Type the following command: java --version and press Enter.



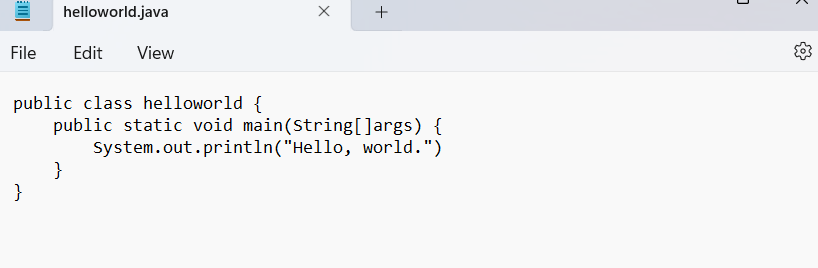
1. To check the java compiler type: **javac –version.**



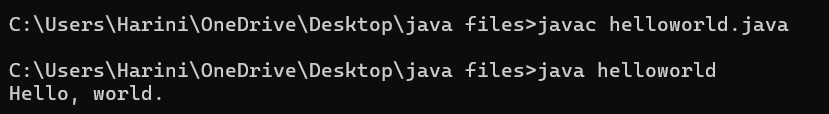
**PROGRAM-2:**

**AIM:** Write a Java program to print the message “Welcome to Java Programming.”

**CODE:**



**OUTPUT:**



**ERRORS:** None found

**IMPORTANT POINTS:**

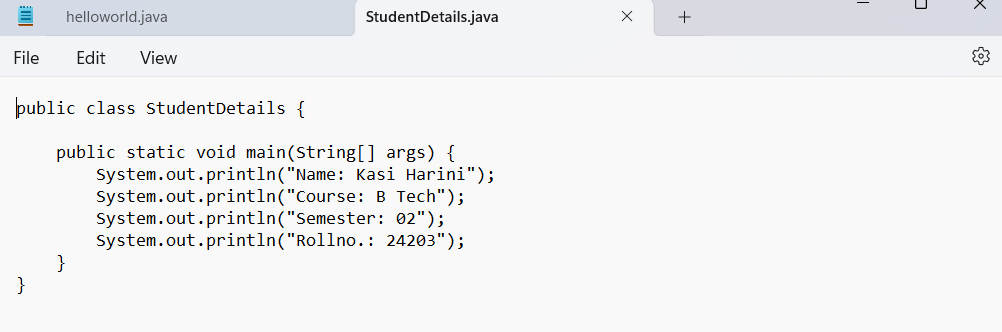
1. Make sure that the file and the class name are the same to avoid confusion.

**PROGRAM-3:**

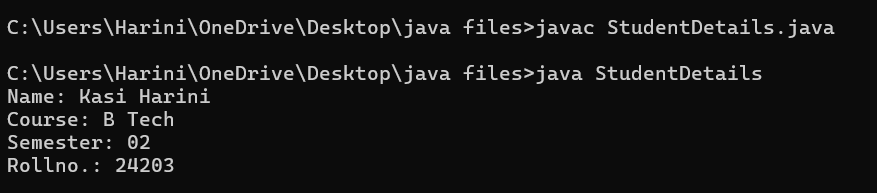
**AIM:** Write a Java Program that prints Name, Roll No, Section of a student.

**CODE:**

**STUDENT DETAILS:**



**OUTPUT:**



ERROR TABLE:

|  |  |
| --- | --- |
| **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:

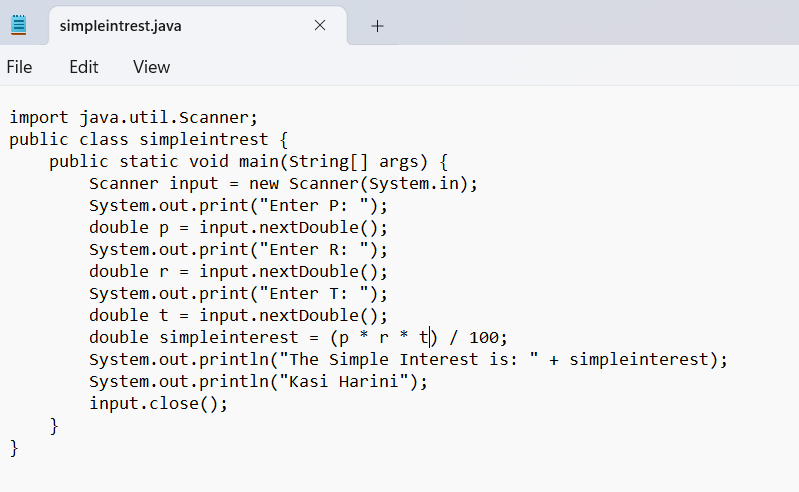
1. When printing the statements, everything should be inside double quotes.

**WEEK – 2:**

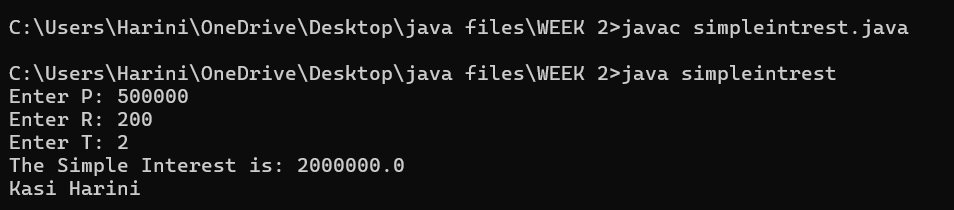
**PROGRAM-1:**

**AIM:** Write a java program to find the simple interest where all the inputs are taken from the user.

**CODE:**



**OUTPUT:**



**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.

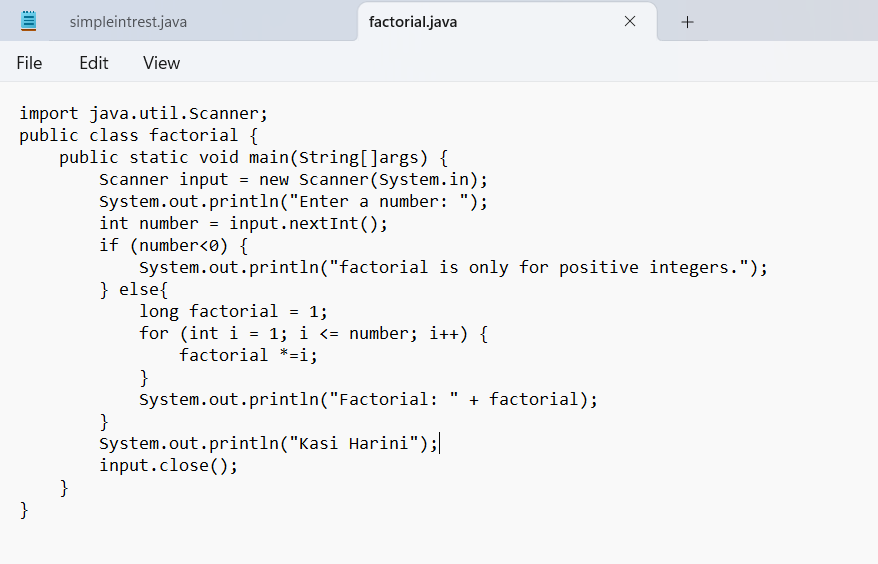
Java.util : This is the package that contains utility classes for Java programming, including the “Scanner” class.

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

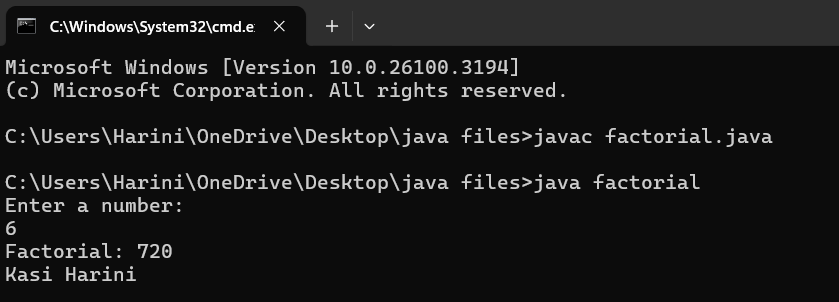
**PROGRAM-2:**

**AIM:** Write a java program to find the factorial of a number where all the inputs are taken from a user.

**CODE:**



**OUTPUT:**



**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

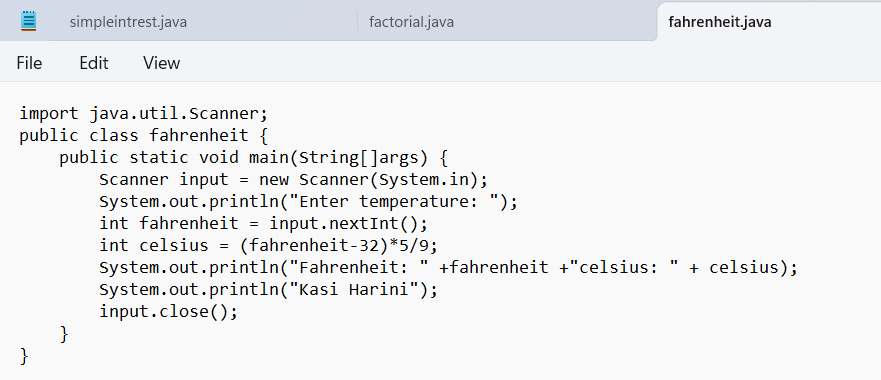
Update expression.

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.

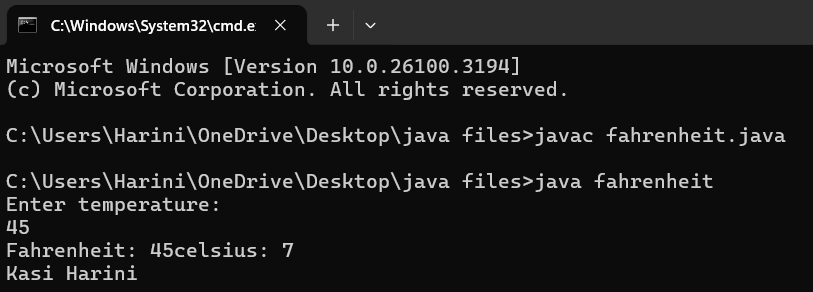
**PROGRAM-3:**

**AIM:** Write a java program to convert the temperature from Celsius to Fahrenheit and Celsius to Fahrenheit.

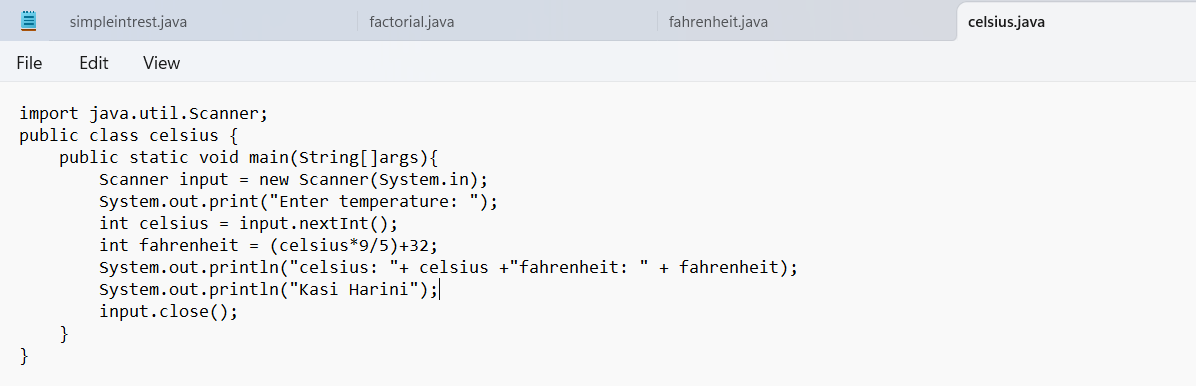
**CODE:**



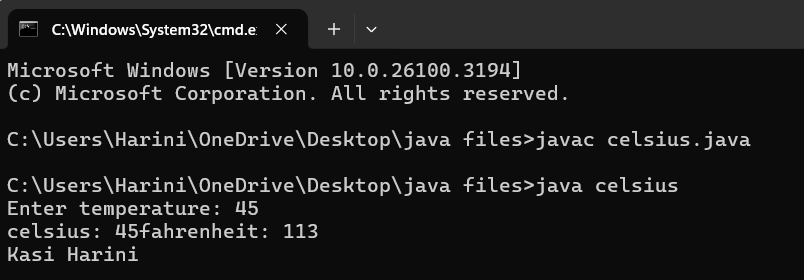
**OUTPUT:**



**CODE:**



**OUTPUT:**



**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

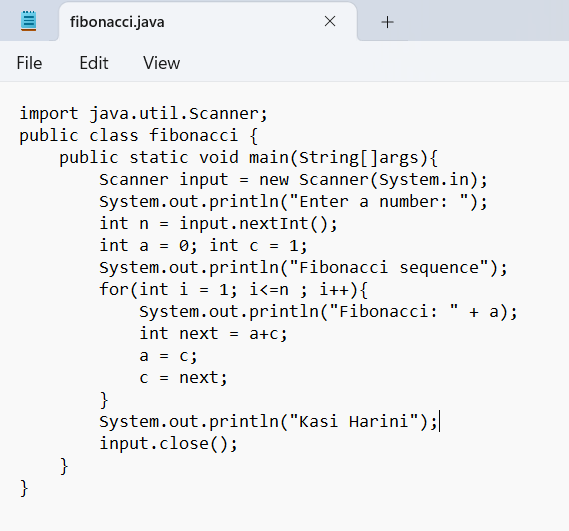
Fahrenheit = (Celsius\*9/5)+32.

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.

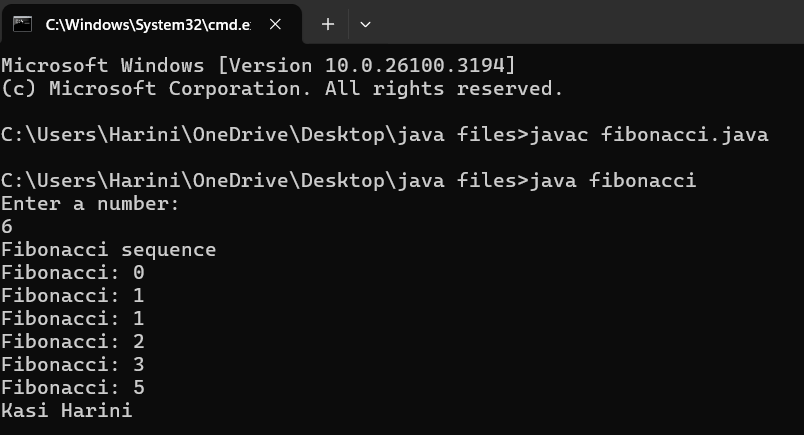
**PROGRAM – 4:**

**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:**

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

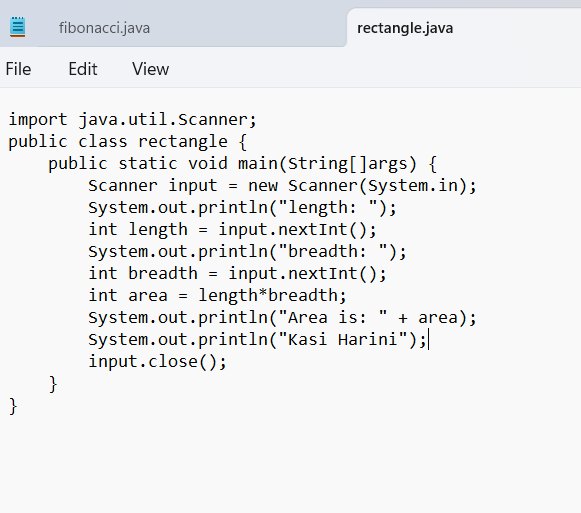
**IMPORTANT POINTS:**

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.

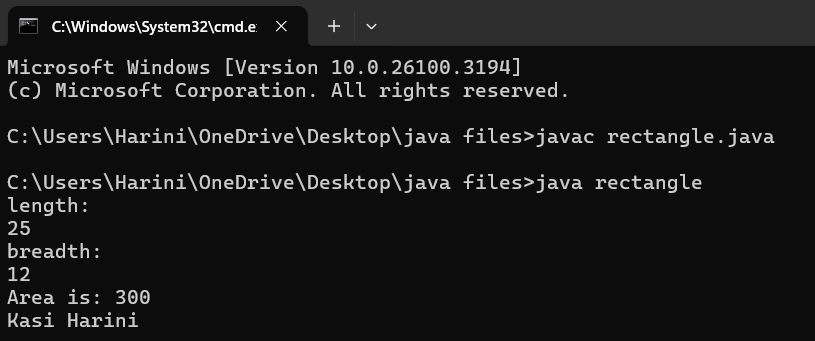
**PROGRAM-5:**

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

L = length of a side of the rectangle,

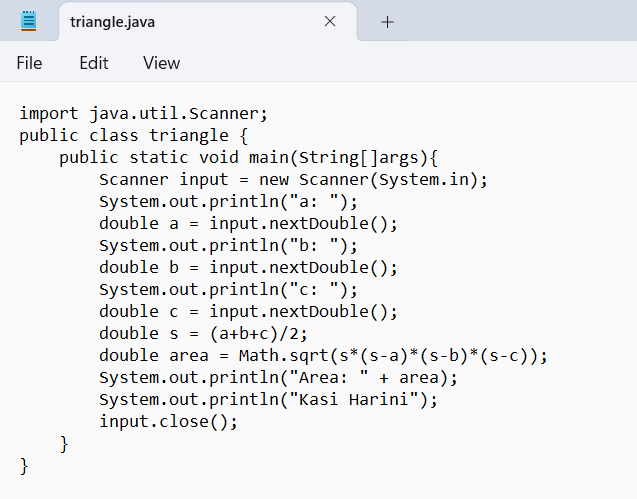
B= breadth of a side of the rectangle.

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

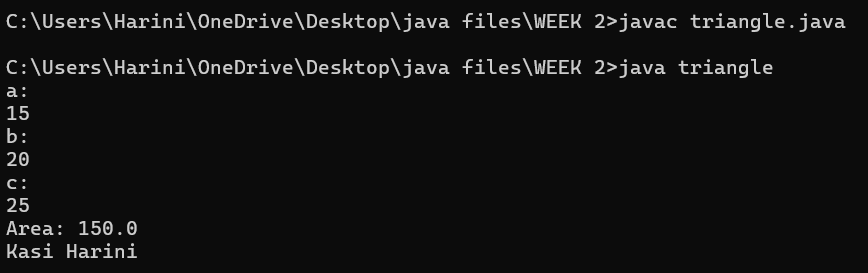
**PROGRAM – 6**

**AIM:** Write a java program to find the Area of a Triangle using heron’s formula, where all the inputs are taken from the user.

**CODE:**



**OUTPUT:**



**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. While printing the variable not giving + sign. 2. Giving int data type to the sqrt method. | 1. We should give correct indentation. 2. Give the double data type to the sqrt method. |

**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:**

**PROGRAM- 1:**

AIM: 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 these methods named start(),stop(),service()

4. Create the objects named car, car1,car2.

**CLASS DIAGRAM:**

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

**CODE:**

public class Car {

private String car\_color;

private String car\_brand;

private String fuel\_type;

private String mileage;

public void start() {

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

}

public void stop() {

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

}

public void service() {

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

}

public static void main(String args[]) {

Car car = new Car();

car.car\_color = "blue";

car.car\_brand = "wolkswagen";

car.fuel\_type = "petrol";

car.mileage = "40";

car.start();

System.out.println("car\_color: " + car.car\_color + " car\_brand: " + car.car\_brand + " fuel\_type: " + car.fuel\_type + " mileage: " + car.mileage);

Car car\_one = new Car();

car\_one.car\_color = "grey";

car\_one.car\_brand = "maruti suzuki";

car\_one.fuel\_type = "petrol";

car\_one.mileage = "75";

car\_one.stop();

System.out.println("car\_color: " + car\_one.car\_color + " car\_brand: " + car\_one.car\_brand + " fuel\_type: " + car\_one.fuel\_type + " mileage: " + car\_one.mileage);

Car car\_two = new Car();

car\_two.car\_color = "white";

car\_two.car\_brand = "Mercedes benz";

car\_two.fuel\_type = "diesel";

car\_two.mileage = "45";

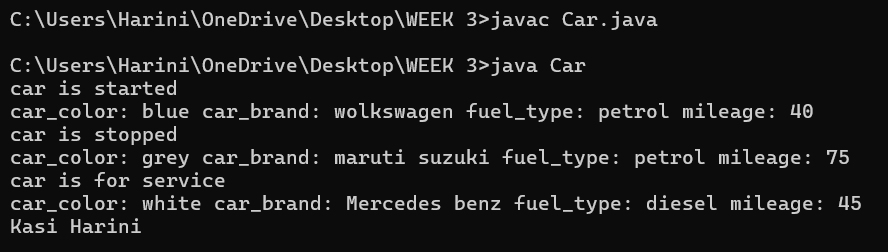
car\_two.service();

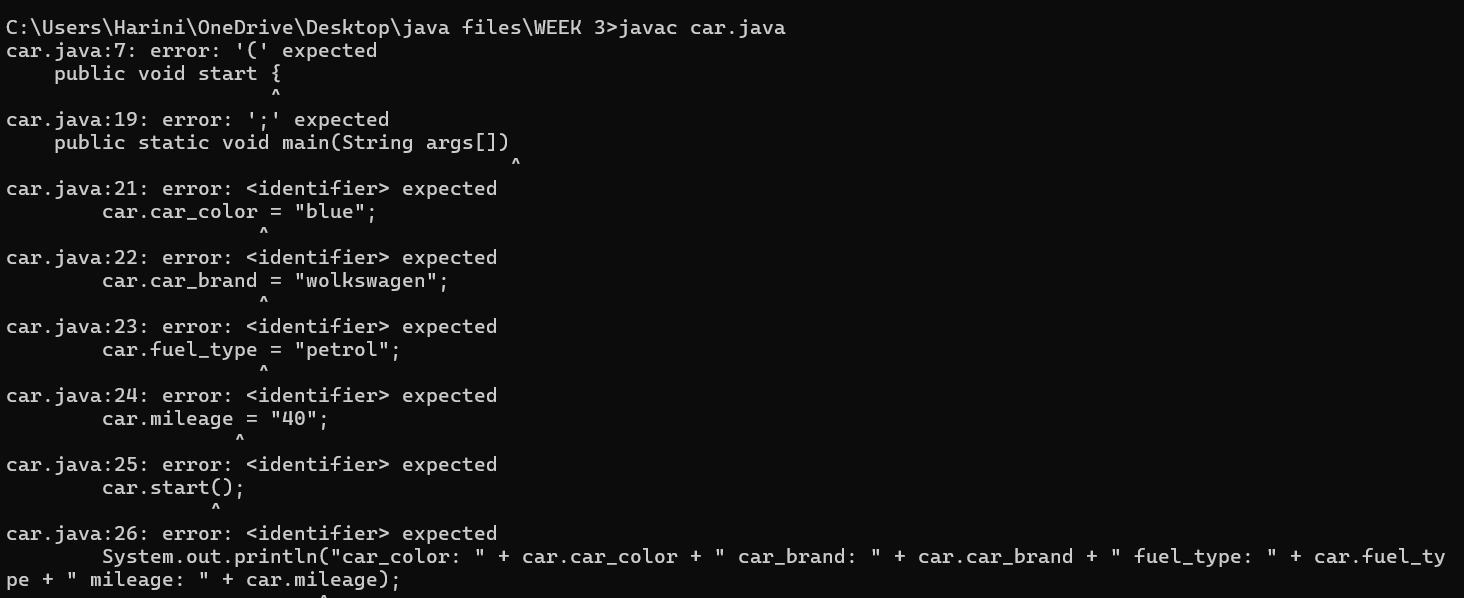
System.out.println("car\_color: " + car\_two.car\_color + " car\_brand: " + car\_two.car\_brand + " fuel\_type: " + car\_two.fuel\_type + " mileage: " + car\_two.mileage);

System.out.println("Kasi Harini");

}}

**OUTPUT:**



**NEGATIVE CASE;** 

**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Not putting the semi-colon; after calling the function. 2. After Start, Stop, Service not giving the parenthesis ( ). | 1. Put the semi-colon after the writing the code. 2. After every method, put the parenthesis ( ). |

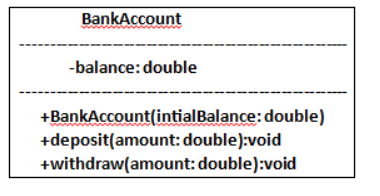
**IMPORTANT POINTS:**

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

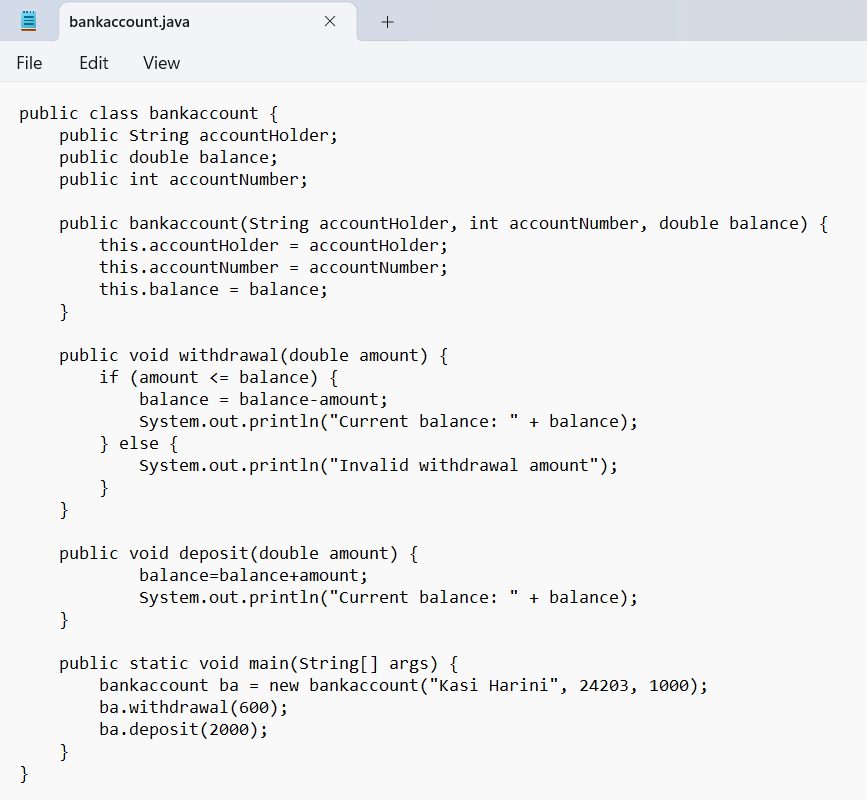
**PROGRAM- 2:**

**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.

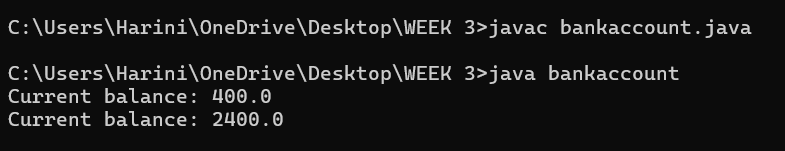
**CLASS DIAGRAM:**



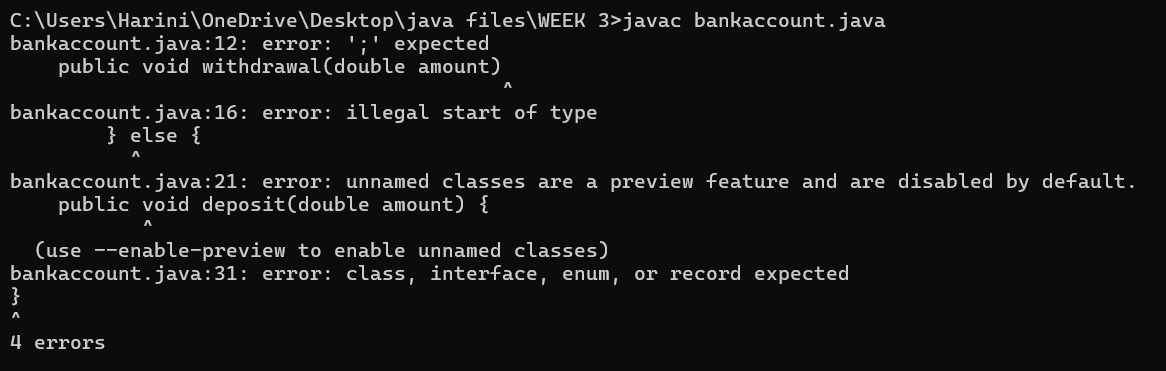
**CODE:**



**OUTPUT:**



**NEGATIVE CASE:**



**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Not putting the semi-colon; after calling the function. 2. After Withdrawal,deposit not giving the parenthesis ( ). | 1. Put the semi-colon after the writing the code. 2. After every method, put the parenthesis ( ). |

**IMPORTANT POINTS:**

1. The condition inside the if statement must be correct.
2. It explains that if the withdrawal money is less than the money in the bank account, then we can withdraw the amount.

**WEEK -4**

**PROGRAM – 1:**

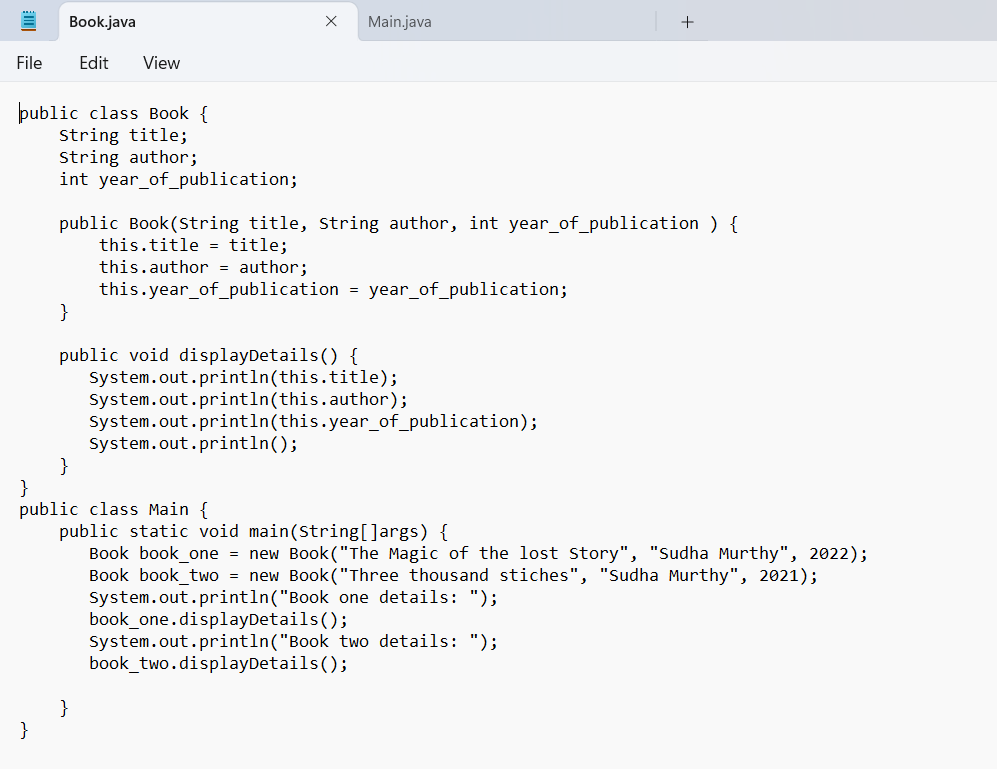
**AIM:** Write a java program with class named “book”, the class should contain various attributes such as title, author, year of publication it should also contain a constructor with parameters which initializes, title, author, and year of publication.

Create a method which displays the details of the book and display the details of two books.

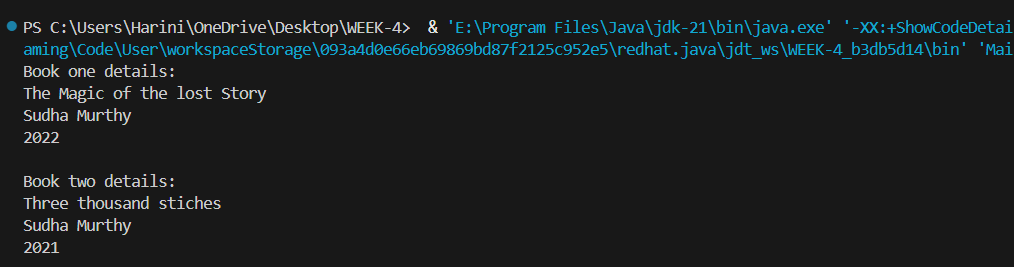
**CLASS DIAGRAM:**

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

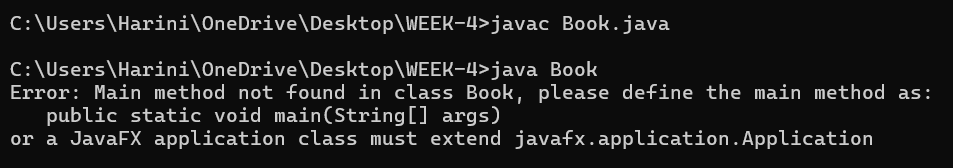
**CODE:**



**OUTPUT:**



**NEGATIVE CASE:**



**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Not defining the function in a file. 2. Two public class files should not be saved in the same file. | 1. To call the method we must define a function in a file. 2. Two public class files should be saved in different files. |

**IMPORTANT POINTS:**

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

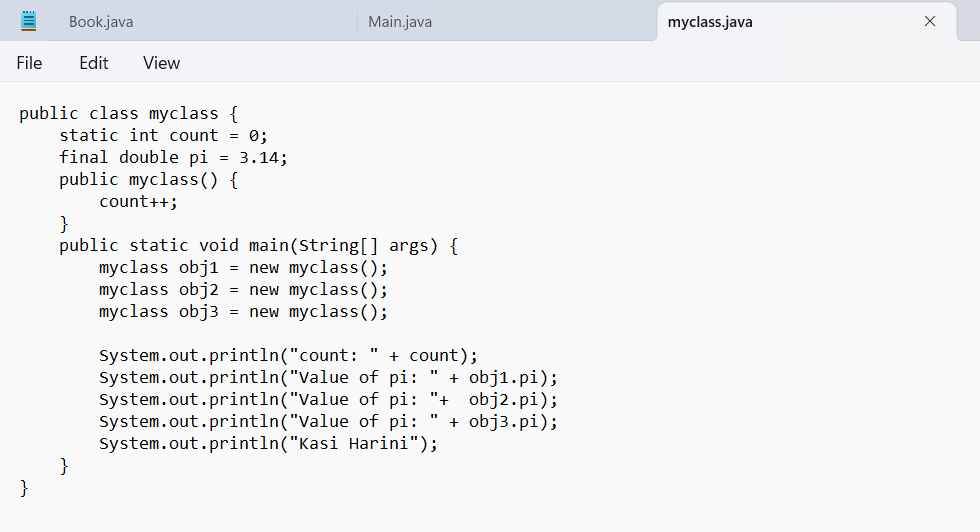
**PROGRAM – 2:**

**AIM:** Create a java Program with class named myclass with static variable count of int type, initialized to zero and a constant variable “pi” of type double initialized to 3.14 as attributes of the class, ow define a constructor for “myclass” that increments the count variable each time an object of my class is created (count++), finally print the final values of count and pi variables create three objects.

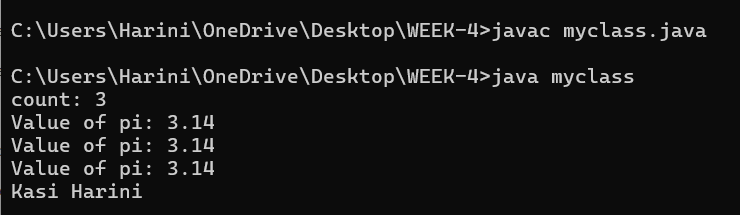
CLASS DIAGRAM:

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

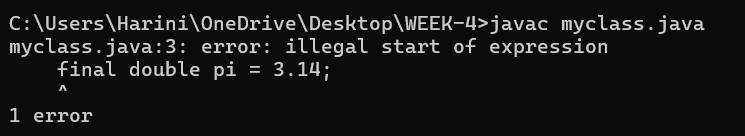
**CODE:**



**OUTPUT:**



**NEGATIVE CASE:**



**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Not Putting the semi-colon after calling a function, 2. Not giving the indentation properly. | 1. Put the semi-colon after calling a function. 2. All the indentation must be correct to run the code correct. |

**IMPORTANT POINTS:**

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

**WEEK-5**

**PROGRAM-1:**

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

Hint: collect required variables using super class,

Create each class for a parameter and each class must contain a method.

**CLASS DIAGRAM:**

|  |
| --- |
| Calulator   * 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();

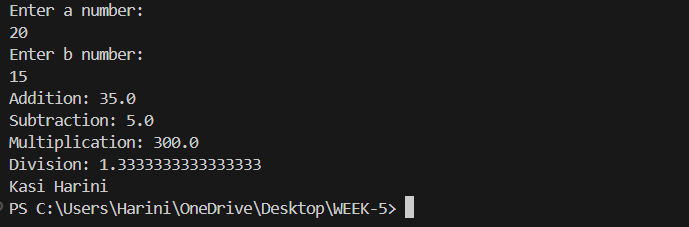
System.out.println("Kasi Harini");

input.close();

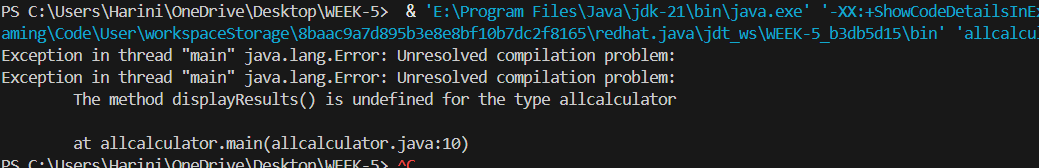
}

}

**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. 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.
4. Every class should have a constructor

**Question:**

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

**Answer:**

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.

**CLASS DIAGRAM:**

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

Car Bike Truck

-no.of.doors: int -has gears: bool -capacity: float

+init (brand, speed +init (brand, speed - displaydetails()

, no.of.doors) , has gears)

**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 showDetails() {

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() {

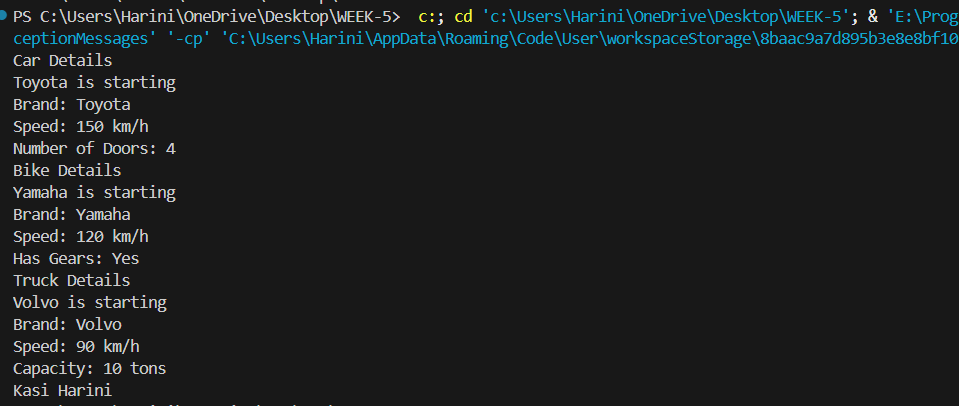
super.showDetails();

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

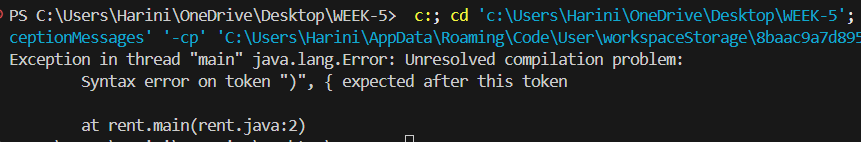
}

}

**OUTPUT:**



**NEGATIVE CASE:**



**ERROR TABLE:**

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

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.

**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)

+ startVehicle(): void

+ displayDetails(): 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("Kasi Harini");

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

v.displayinfo();

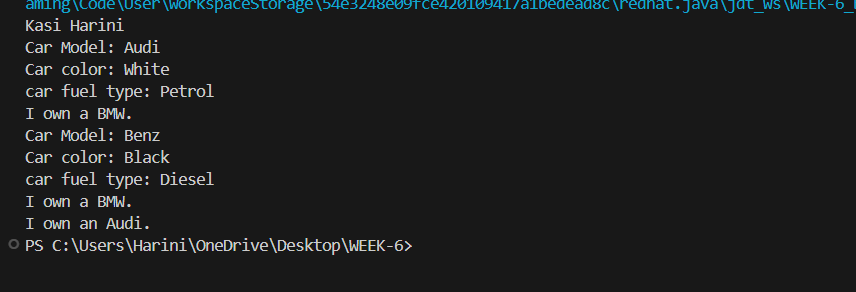
Car c = new Car("Benz", "Black", "Diesel");

c.displayinfo();

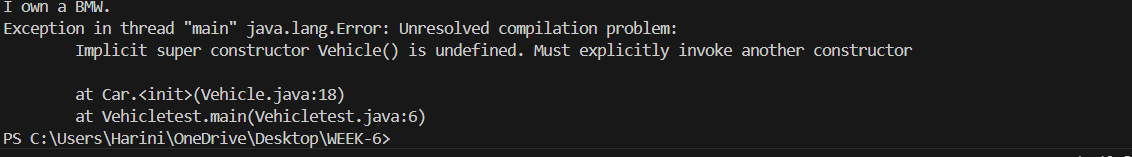
}

}

**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%

CLASS DIAGRAM:

AdmissionSystem

- scanner: Scanner

- name: String

- percentage: double

- program: String

+ main(args: String[]): void

+ takeInput(): void

+ checkEligibility(): void

+ closeScanner(): void

**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.");

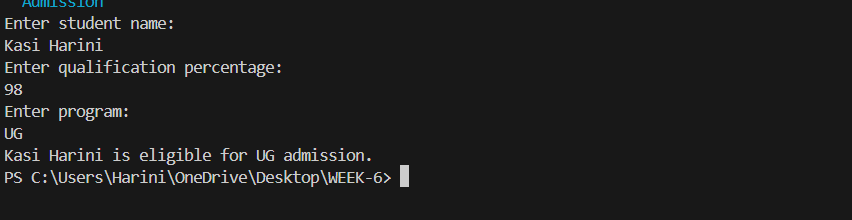
}

input.close();

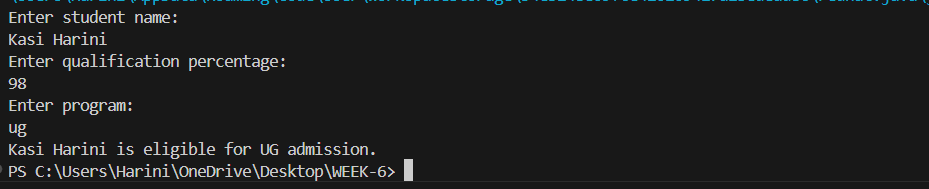
}

}

**OUTPUT:**



**NEGATIVE CASE:**



**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

**CLASS DIAGRAM:**

Calculator

+add(int, int): int

+add(double, double): double

+add(int, int, int): int

+main(String[]): void

**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));

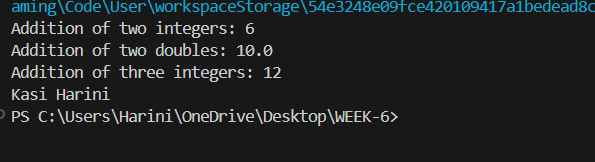
System.out.println("Addition of two doubles: " + c.add(3.5,6.5));

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

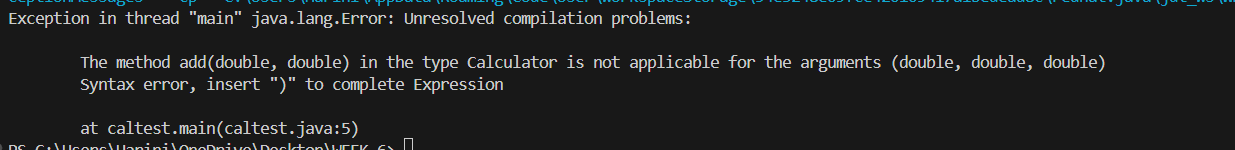
System.out.println("Kasi Harini");

}}

**OUTPUT:**



**NEGATIVE CASE:**



**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.

**CLASS DIAGRAM:**

SHAPE

+calculateArea(side: double): double

+calculateArea(width: double, length: double): double

CIRCLE

+calculateArea(radius: double): double

TOOLS

+main(args: String[]): void

**CODE:**

public class Shape {

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();

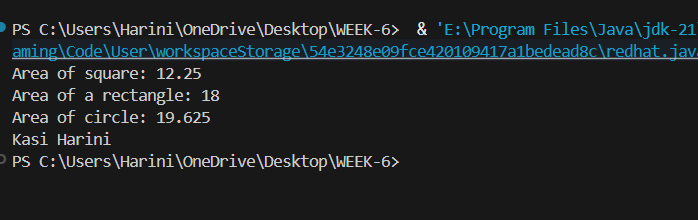
System.out.println("Area of circle: " + c.calculatearea(2.5));

System.out.println("Kasi Harini");

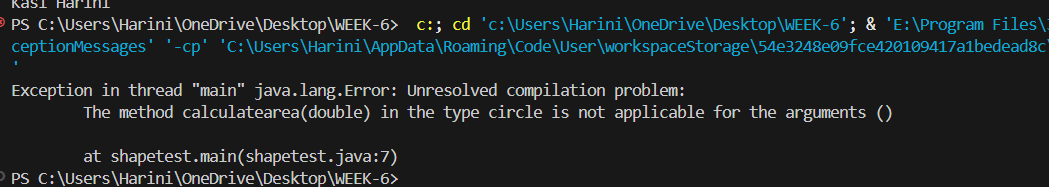
}

}

**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. |

**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 extend the 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 Animal2 {

abstract void sound();

}

class Lion extends Animal2 {

void sound() {

System.out.println("Lion roars in the jungle.");

}

}

class Tiger extends Animal2 {

void sound() {

System.out.println("Tigers roar and attack secretly.");

}

}

public class Animal2test {

public static void main(String[]args) {

System.out.println("Kasi Harini");

System.out.println("AV.SC.U4CSE24203");

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

Lion l = new Lion();

l.sound();

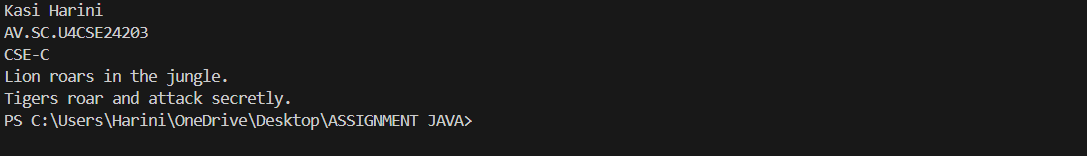
Tiger t = new Tiger();

t.sound();

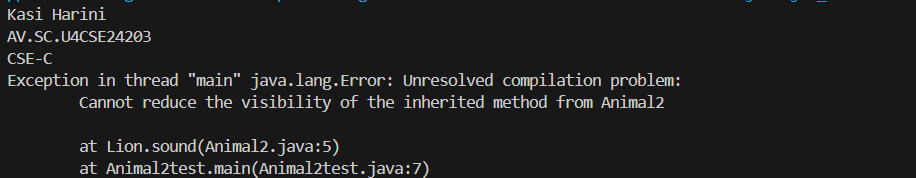
}

}

**OUTPUT:**



**NEGATIVE CASE:**



**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Error while printing the variables. 2. Incorrect declaration of integer. | 1. Give the plus sign while printing. 2. Give input.nextInt(), where I should be capital. |

**IMPORTANT POINTS:**

1. We override the methods in the superclass.
2. Here we are using the heirarchial inheritance.

**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:**

|  |
| --- |
| Shape3D  +calculatevolume(): double  +calculatesurfacearea(): double |

|  |
| --- |
| **Sphere**  -radius: double  +calculateVolume()  +calculateSurfaceArea() |

|  |
| --- |
| **Cube**  -side: double  +calculateVolume()  +calculateSurfaceArea() |

**CODE:**

abstract class Shape3D {

abstract double calculatevolume();

abstract double calculatesurfacearea();

}

class sphere extends Shape3D {

public double r;

public sphere(double r) {

this.r = r;

}

double calculatevolume() {

return (1.34\*3.14\*r\*r\*r);

}

double calculatesurfacearea() {

return (4\*3.14\*r\*r);

}

}

class cube extends Shape3D {

public double s;

public cube(double s) {

this.s = s;

}

double calculatevolume() {

return (s\*s\*s);

}

double calculatesurfacearea() {

return (6\*s\*s);

}

}

public class Shapetest {

public static void main(String[]args) {

System.out.println("Kasi Harini");

System.out.println("AV.SC.U4CSE24203");

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

sphere s = new sphere(5.0);

System.out.println("Volume of the given sphere: " + s.calculatevolume());

System.out.println("Surface Area of the given sphere: " + s.calculatesurfacearea());

cube c = new cube(6.0);

System.out.println("Volume of the given cube: " + c.calculatevolume());

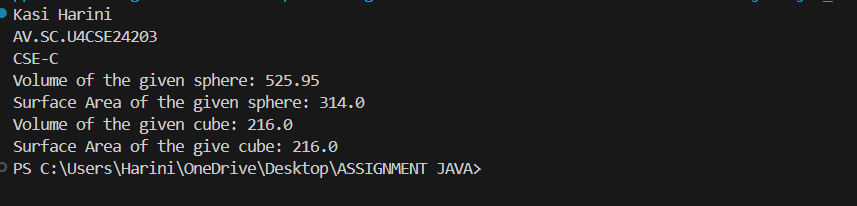
System.out.println("Surface Area of the give cube: " + c.calculatesurfacearea());

}

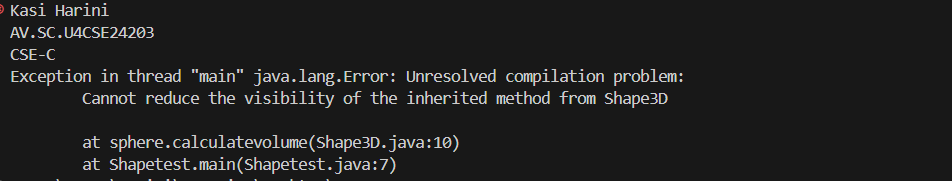
}

**OUTPUT:**

**POSITIVE CASE:**



**NEGATIVE CASE:**



**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Wrong datatype entered. 2. Object not defined. | 1. Enter the correct datatype i.e double instead of int. 2. Enter the correct object and if not create new one. |

**IMPORTANT POINTS:**

1. Here we used the abstract to declare an abstract class.
2. Abstract classes and methods help us to declare the methods without declaring the return type in them.
3. To get the values, we declared a constructor for each subclass and initialized values for them.

**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 n) and concrete method to display the pattern title.

. implement two subclasses:

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

2. NumberPattern - prints a right-angled triangle of increasing numbers.

. int the main() method, create objects of both subclasses and print the patterns for a given number of rows.

Example Output for n = 5: Number Pattern

Star Pattern 1

\* 1 2

\* \* 1 2 3

\* \* \* 1 2 3 4

\* \* \* \* 1 2 3 4 5

\* \* \* \* \*

**CLASS DIAGRAM:**

|  |
| --- |
| Pattern Printer  +printPattern(n): void  +displayTitle(title): void |

|  |  |  |
| --- | --- | --- |
| Star Pattern  +printPattern(n): void |  | NumberPattern  +printPattern(n):void |

**CODE:**

abstract class patternprinter {

abstract void printpattern(int n);

public void displayTitle(String title) {

System.out.println(title);

}

}

class starpattern extends patternprinter {

public void printpattern(int n) {

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

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

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

System.out.println();

}}}

class NumberPattern extends patternprinter {

public void printpattern(int n) {

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

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

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

System.out.println();

} }}

public class ptest {

public static void main(String[] args) {

System.out.println("Kasi Harini");

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

System.out.println("AV.SC.U4CSE24203");

int n = 5;

patternprinter star = new starpattern();

star.displayTitle("Star Pattern");

star.printpattern(n);

System.out.println();

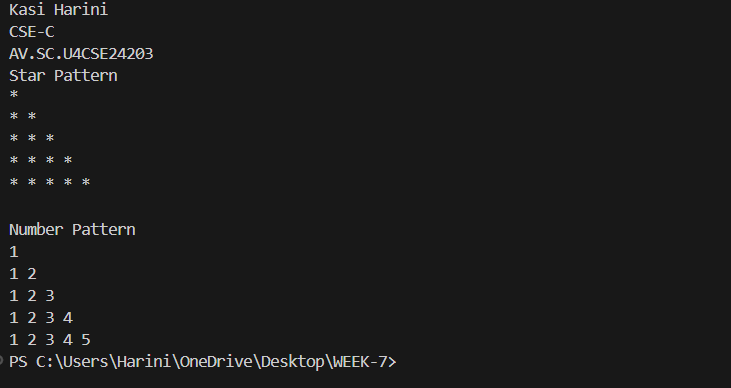
patternprinter number = new NumberPattern();

number.displayTitle("Number Pattern");

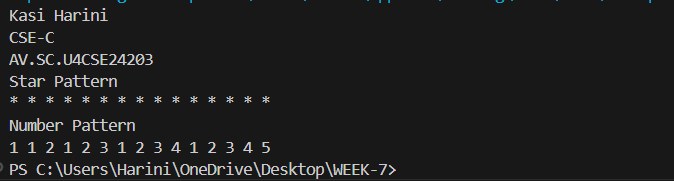
number.printpattern(n);

}}

**OUTPUT:**



**NEGATIVE CASE:**



**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Not mentioning public to override the abstract methods. | 1. Mention the public to override the method. |

**IMPORTANT POINTS:**

1. Use abstract classes to enforce a common structure for pattern printing.
2. PatternPrinter is the abstract class defining the common template.
3. Subclasses (StarPattern, NumberPattern) provide specific implementations.
4. displayTitle() is a concrete method shared by all subclasses.

**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:**

|  |
| --- |
| Shape  +getperimeter(): double |

|  |
| --- |
| Rectangle  +length  +width  +getperimeter() |

|  |
| --- |
| Triangle  +side1  +side2  +side3 |

|  |
| --- |
| Circle  +radius |

CODE:

interface Shape {

double getperimeter();}

class rectangle implements Shape {

public int l;

public int b;

public rectangle(int l, int b) {

this.l = l;

this.b = b;

}

public double getperimeter() {

return 2\*(l+b);

}

}

class circle implements Shape {

public int r;

public circle(int r) {

this.r = r;

}

public double getperimeter() {

return 2\*3.14\*r;

}

}

class triangle implements Shape {

public int a;

public int b;

public int c;

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

this.a = a;

this.b = b;

this.c = c;

}

public double getperimeter() {

return a+b+c;

}

}

public class Shapetest {

public static void main(String[]args) {

System.out.println("Kasi Harini");

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

System.out.println("AV.SC.U4CSE24203");

Shape r = new rectangle(2,3);

Shape c = new circle(4);

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

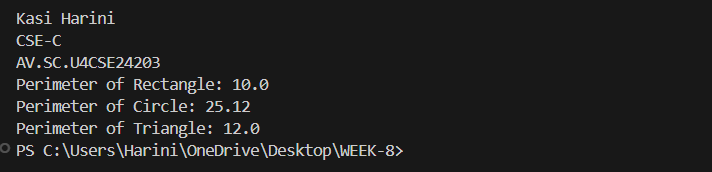
System.out.println("Area of Rectangle: " + r.getperimeter());

System.out.println("Area of Circle: " + c.getperimeter());

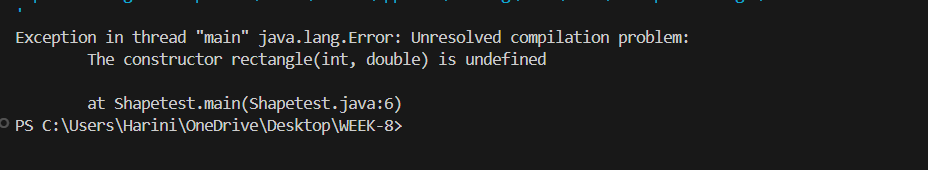
System.out.println("Area of Triangle: " + t.getperimeter());

}}

OUTPUT;



NEGATIVE CASE:



**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. the constructor rectangle(int, double) is undefined. | 1. Give int, int values or declare a constructor with double and int values. |

IMPORTANT POINTS:

1. For interface classes, first declare the main class and then give the subclass name.
2. For methods in interface classes and subclasses, we need to declare public.

**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 |

|  |
| --- |
| Football  +play() |

|  |
| --- |
| Volleyball  +play() |

|  |
| --- |
| BasketBall  +play() |

**CODE:**

interface Playable {

abstract void play();

}

class Football implements Playable {

public void play() {

System.out.println("some people play football in the Ground.");

}

}

class Volleyball implements Playable {

public void play() {

System.out.println("some people play volleyball in the Stadium.");

}

}

class Basketball implements Playable {

public void play() {

System.out.println("some people play basket ball in the Rain.");

}

}

public class playtest {

public static void main(String[]args) {

System.out.println("Kasi Harini");

System.out.println("AV.SC.U4CSE24203");

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

Playable f = new Football();

Playable v = new Volleyball();

Playable b = new Basketball();

f.play();

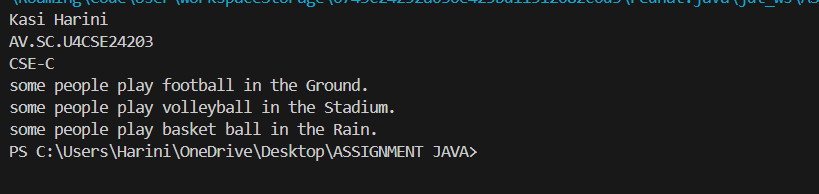
v.play();

b.play();

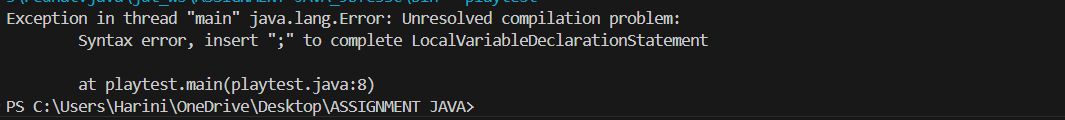
}

}

**OUTPUT:**



**NEGATIVE CASE:**

****

**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Declaring an abstract class instead of interface class. 2. Not declaring public in each class. | 1. Declare an interface class instead of abstract class. 2. Declare public infront of each class. |

**IMPORTANT POINTS:**

1. The playable interface abstracts the play() method, ensuring different classes implement it differently
2. The play() method behaves differently based on the object type football, volleyball, basketball.
3. Each class encapsulates its own implementation of how the sport is played, hiding the details from the user.