**23CSE111**

**OBJECT ORIENTED PROGRAMMING**

**LAB REPORT**



**Department of Computer Science Engineering**

**Amrita School of Computing**

**Amrita Vishwa Vidyapeetham, Amaravati Campus**

**Name: Y Yashaswi Reddy**

**Verified By Roll No: av.sc.u4cse24351**

**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.

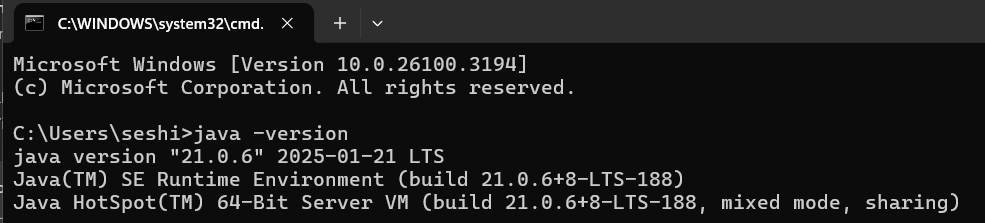
2)Click New and add: C:\Program Files\Java\jdk-21\bin

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.**

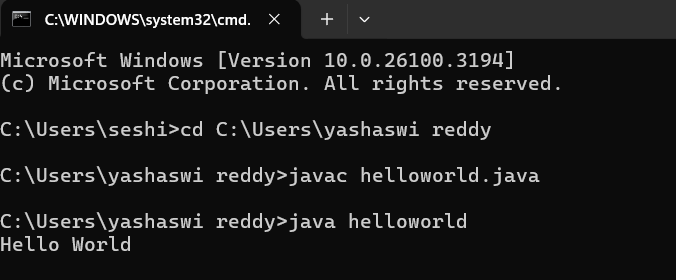
**PROGRAM-2:**

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

**CODE:**



**OUTPUT:**



**ERRORS:** None found

IMPORTANT POINTS:

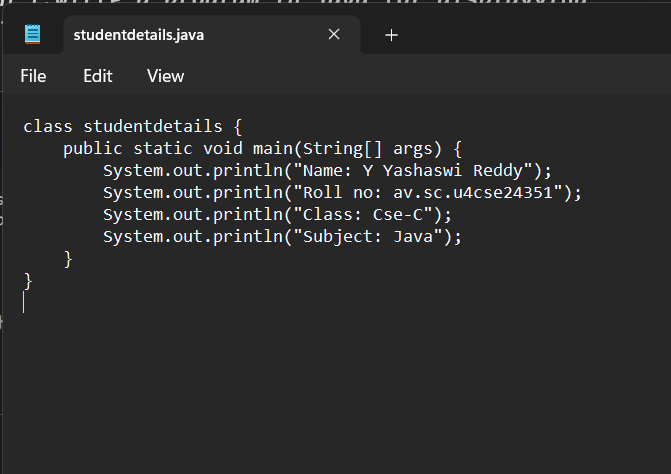
1. Make sure that the file and the class name are the same to avoid confusion.

**PROGRAM-3:**

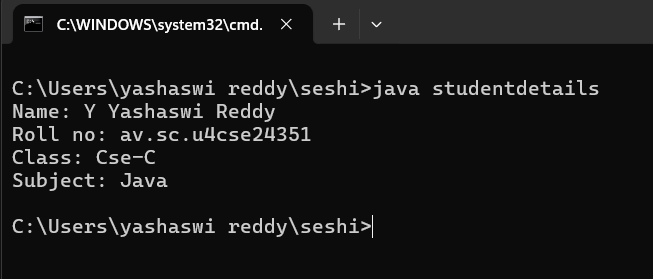
**AIM:** Write a Java Program that prints Name, Roll No, Section of a student.

**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 scetion | 1. code is rectified by keeping capital “S” 2. Giving strings to name and section |

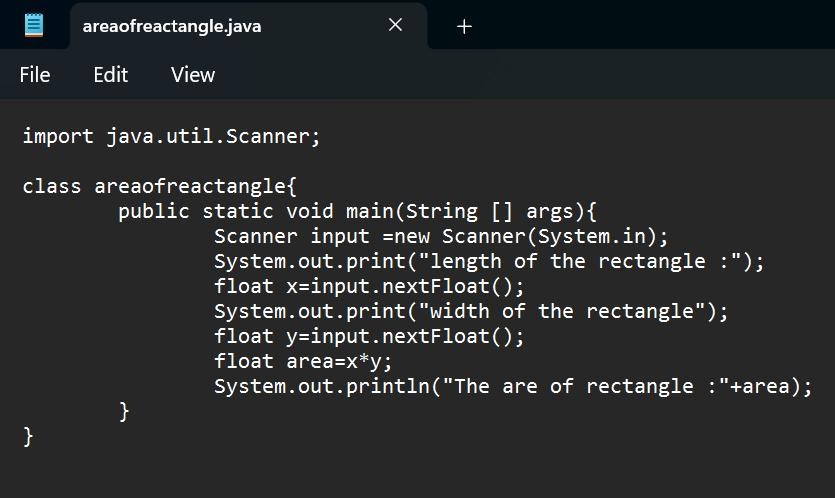
IMPORTANT POINTS:

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

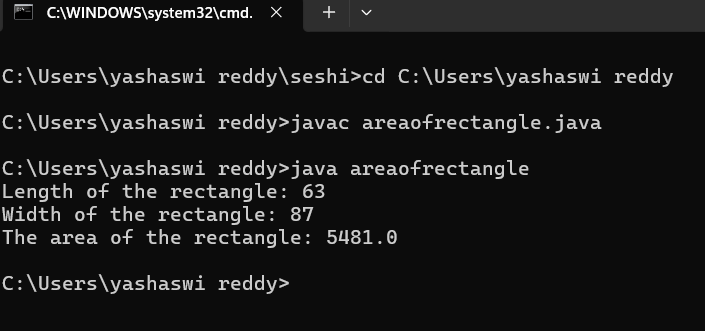
**WEEK-2:**

**1.AIM:** Write a Java Program for calculating area of rectangle

**PROGRAM:**

**

**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

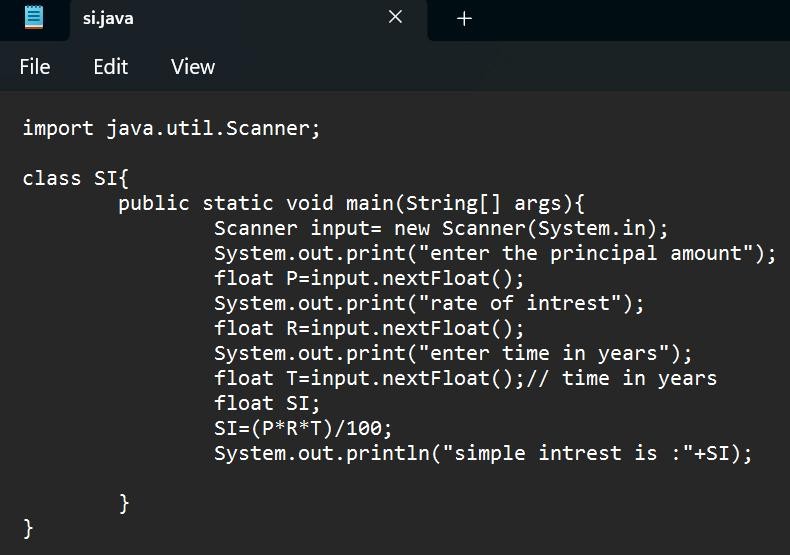
L = length of a side of the rectangle,

B= breadth of a side of the rectangle.

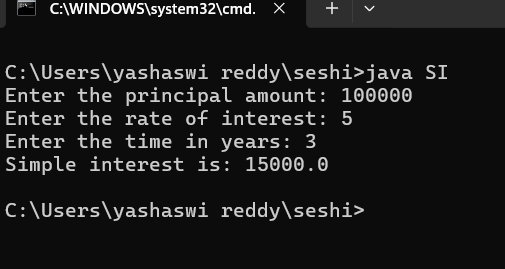
1. Here, we must be sure that all the expressions/conditions inside for the for loop must be given correctly.

**2.AIM:** Write a Java Program for calculating simple intrest

**PROGRAM:**

**

**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:

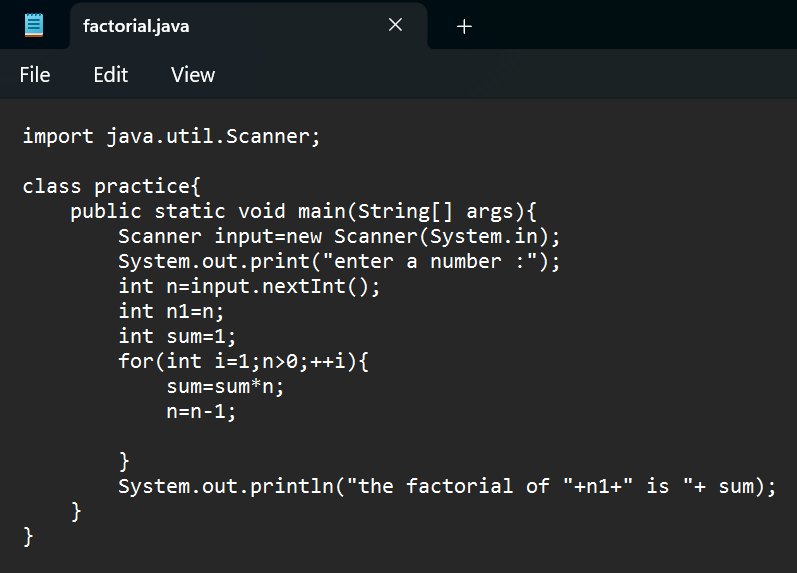
Import: tells the java compiler that we want to use a specific class or package in your code.

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

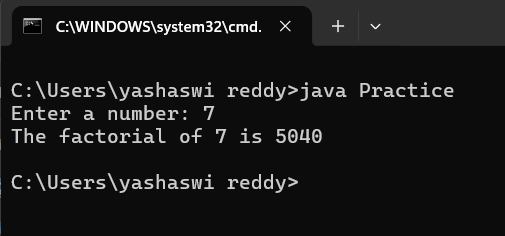
Scanner: this is the class that allows you to read input from the keyboard.

**3.AIM:** Write a Java Program for finding factorial of a number

**PROGRAM:**

**

**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

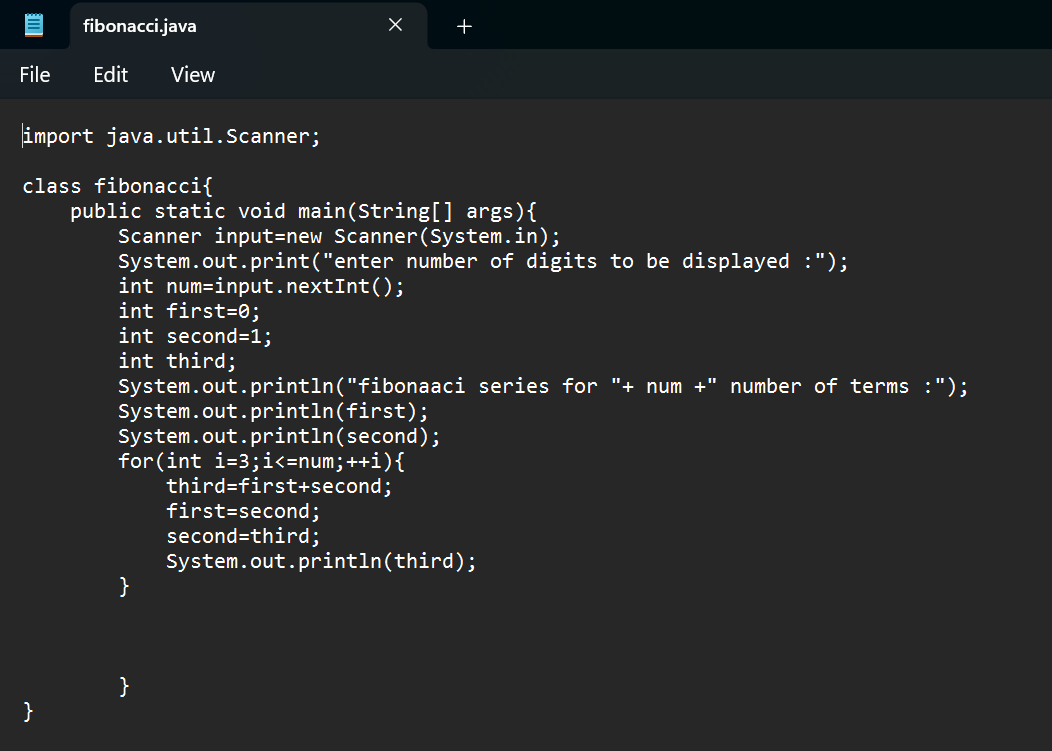
Update expression.

1. Here “factorial\*=I” means factorial = factorial\*I.

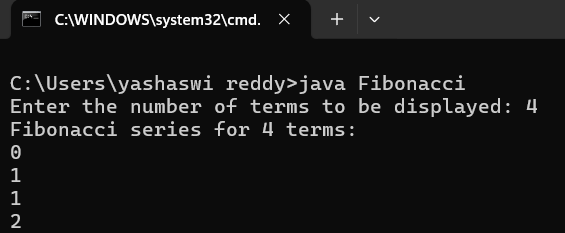
Here we are using the data type “int” just to calculate the integer values and it doesn’t support floating points

**4.AIM:** Write a Java Program for finding Fibonacci series of a number

**PROGRAM:**

**

**OUTPUT:**

**

ERROR TABLE:

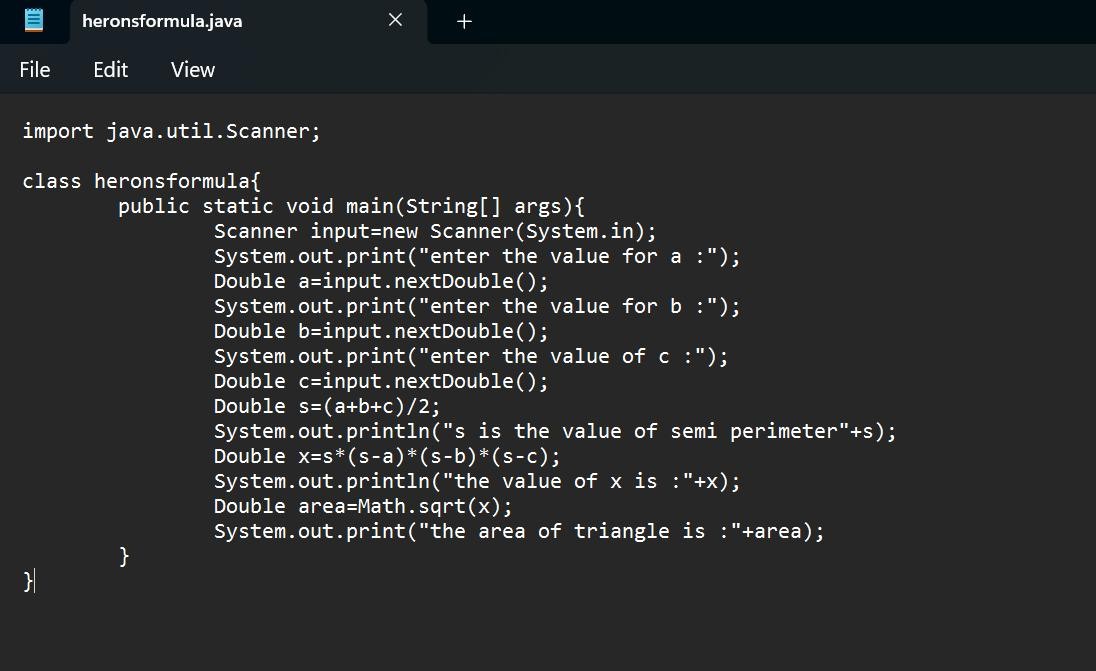
|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1.Giving space between next and Double.  2.Not giving parenthesis after closing the input. | 1.Should not give space between next and Double.  2.We must put parenthesis after closing the input. |

IMPORTANT POINTS:

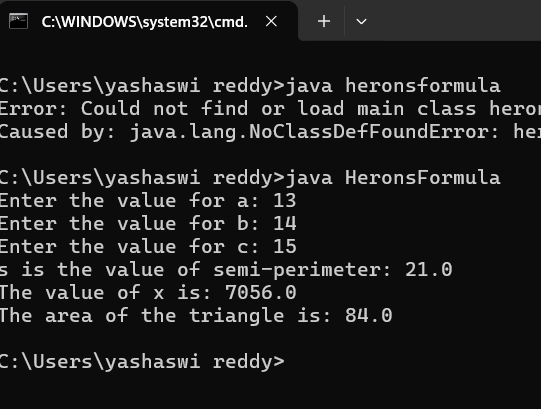
1. In the Fibonacci sequence, the sum value is given to the second variable, and the value of the second variable is given to the first variable.
2. This process is repeated a certain number of times until the conditions are met.

**5.AIM**: Write a Java Program for calculating area of triangle using herons formula

**PROGRAM:**



**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. 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

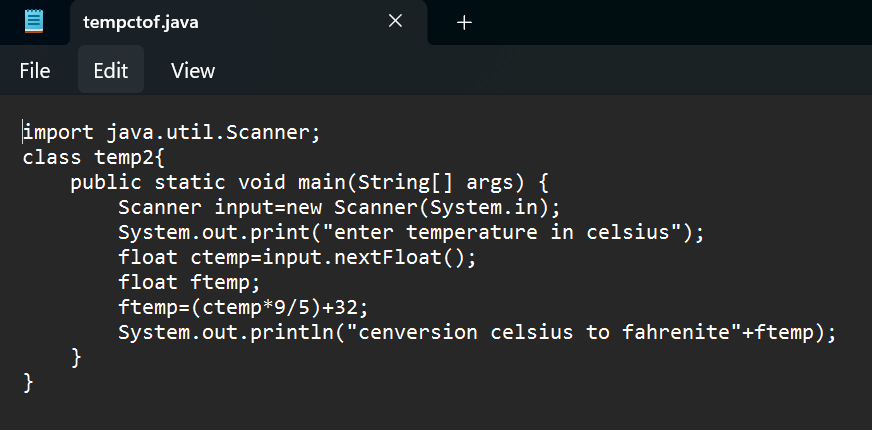
Where S is the semi-perimeter of the triangle.

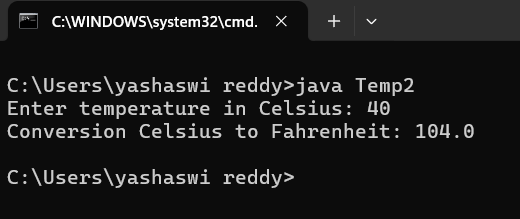
Now the area formula is:

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

**6. I.) AIM:** Write a Java Program for conversion from Celsius to farenheit

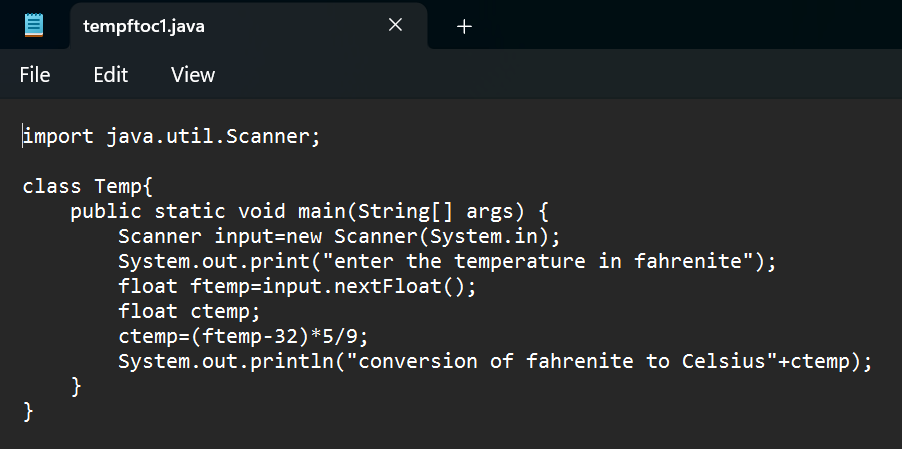
**PROGRAM:**

****OUTPUT:**

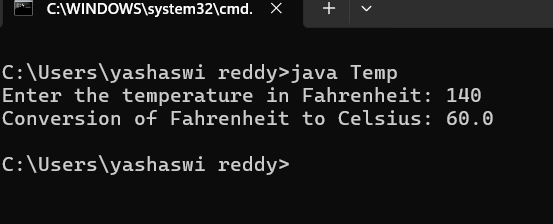
******

**II.) AIM:** Write a Java Program for conversion from farenheit to celsius

**PROGRAM:**

******

**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.

**WEEK -3:**

**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

**PROGRAM:**

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 = "white";

car.car\_brand = "audi";

car.fuel\_type = "petrol";

car.mileage = "20";

car.start();

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

Car car1 = new Car();

car1.car\_color = "white";

car1.car\_brand = "audi";

car1.fuel\_type = "petrol";

car1.mileage = "20";

car1.stop();

System.out.println("car\_color: " + car1.car\_color + " car\_brand: " + car1.car\_brand + " fuel\_type: " + car1.fuel\_type + " mileage: " + car1.mileage);

Car car2 = new Car();

car2.car\_color = "white";

car2.car\_brand = "audi";

car2.fuel\_type = "petrol";

car2.mileage = "20";

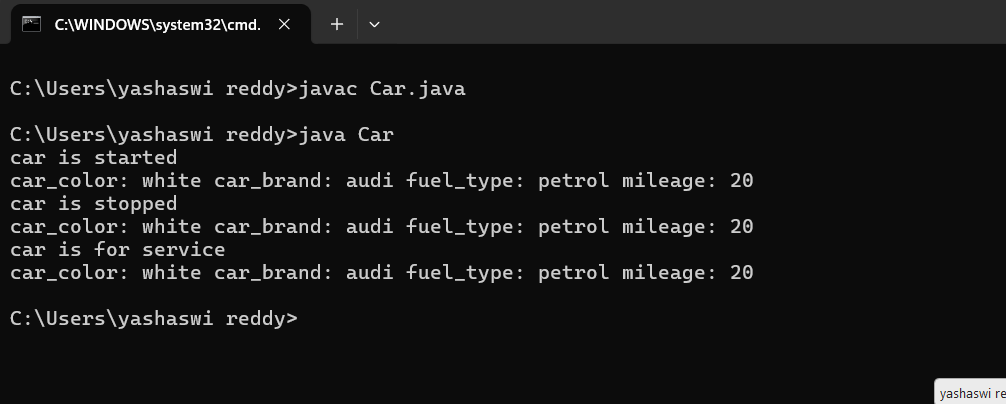
car2.service();

System.out.println("car\_color: " + car2.car\_color + " car\_brand: " + car2.car\_brand + " fuel\_type: " + car2.fuel\_type + " mileage: " + car2.mileage);

}

}

**OUTPUT:**



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:

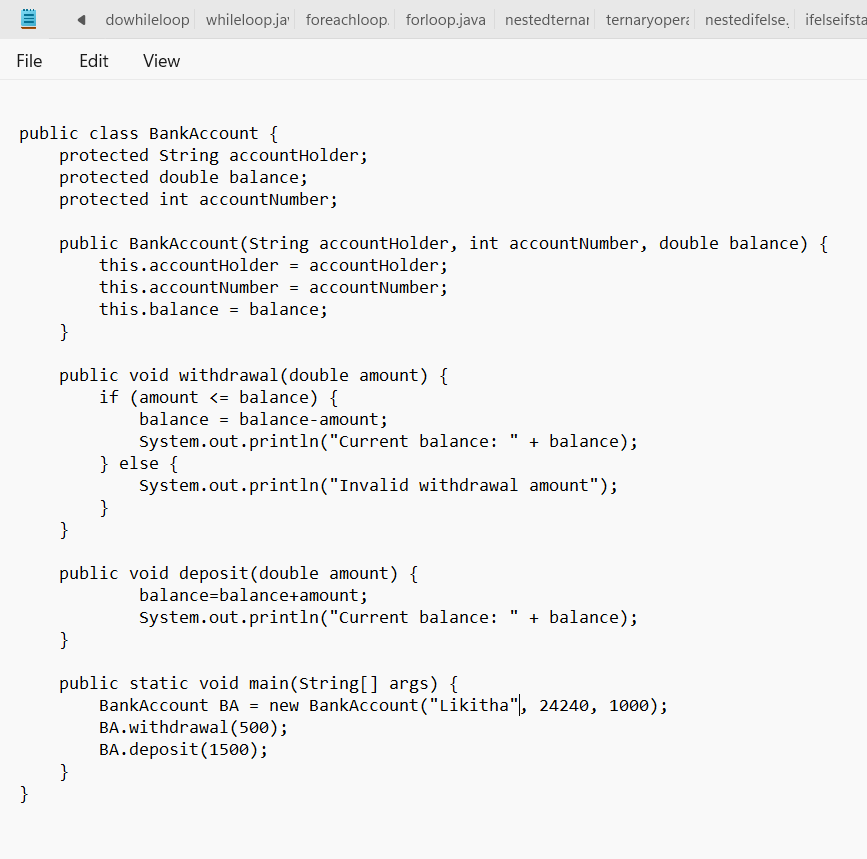
1. Before calling the function we should write the method properly.
2. Here, the “public void start( )” indicates that we are writing a method to call the function.
3. When we call a certain method, the process inside it will be printed as an output of the code.
4. Here the details inside the function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

CLASS DIAGRAM:

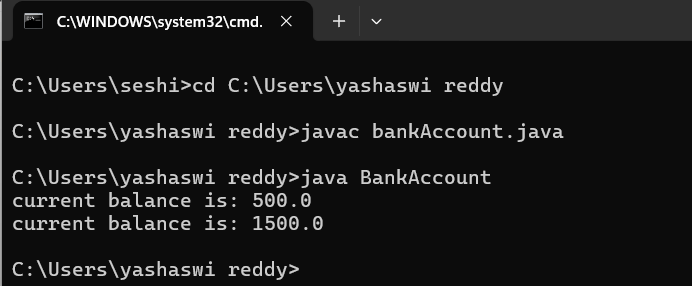
|  |
| --- |
| car  -----------------------  -car\_color:string  -car\_brand:string  -fuel\_type:string  -milage:double  ----------------------  +start():void  +stop():void  +service():void |

**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

**PROGRAM:**

****

**OUTPUT:**



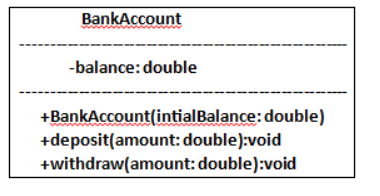
ERROR TABLE:

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

CLASS DIAGRAM:



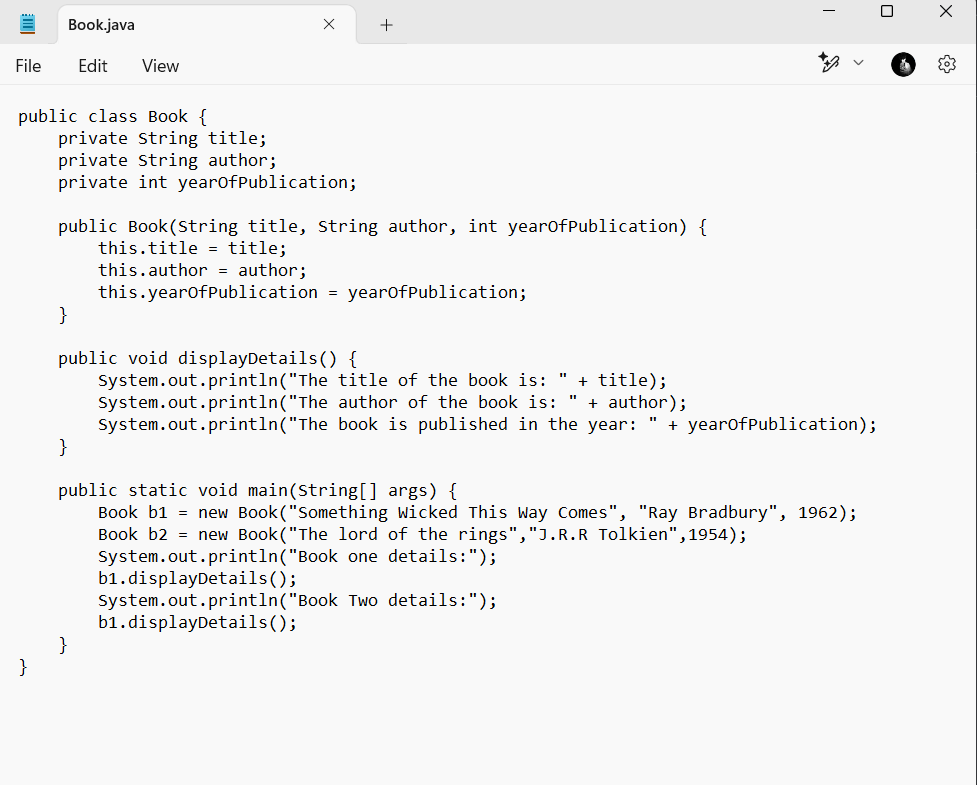
WEEK -4

PROGRAM – 1:

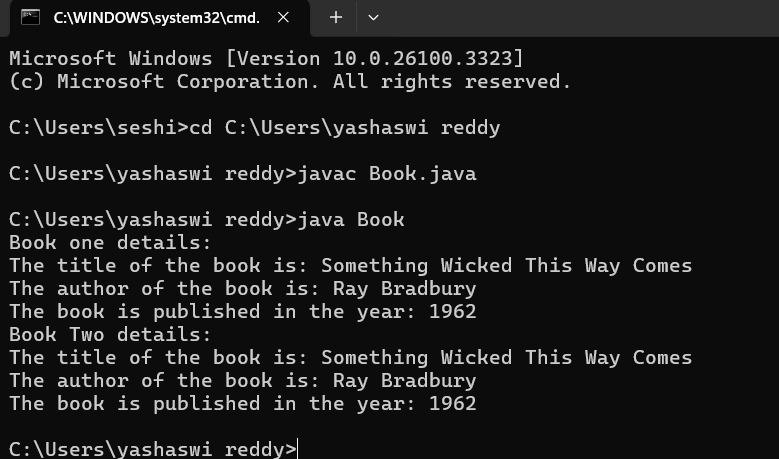
AIM: Write a java program with class named “book”, the class should contain various attributes such as title, author, year of publication it should also contain a constructor with parameters which initializes, title, author, and year of publication.

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

CODE:



OUTPUT:



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.

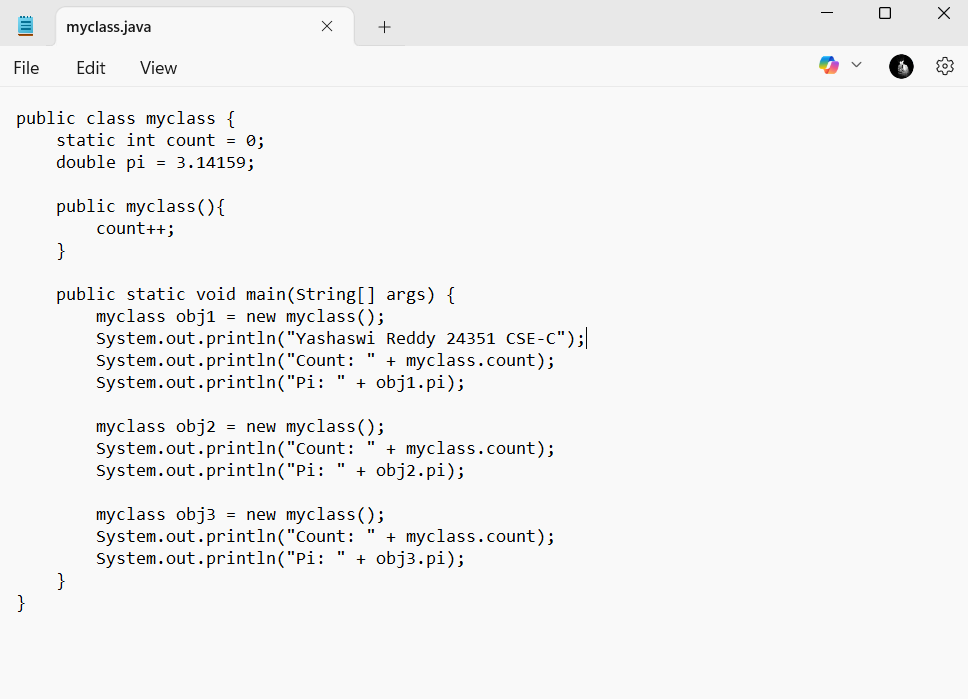
CLASS DIAGRAM:

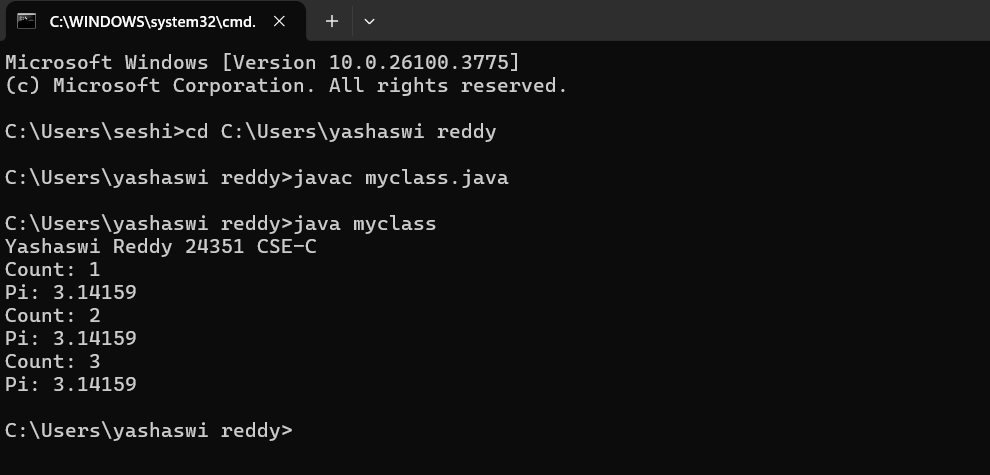
|  |
| --- |
| Book   * Title: String * Author: String * Year of publication: int   + Book(title: String,  Author: String;  Year of publication: int  + displayDetails( ): void |

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.

CODE:



OUTPUT:

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.

CLASS DIAGRAM:

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

Week-5

**1.** **Create a calculator using the operations including**

**addition, subtraction, multiplication and division using**

**multilevel in heritance and display the desired output**

**Code:**

**import java.util.Scanner;**

**class Calculator {**

**public void add(double a, double b) {**

**System.out.println("Sum of two numbers is: " + (a + b));**

**}**

**public void subs(double a, double b) {**

**System.out.println("Difference of two numbers is: " + (a - b));**

**}**

**}**

**class Calculator1 extends Calculator {**

**public void mul(double a, double b) {**

**System.out.println("Multiplication of two numbers is: " + (a \* b));**

**}**

**}**

**class Calculator2 extends Calculator1 {**

**public void div(double a, double b) {**

**if (b == 0) {**

**System.out.println("Division is not allowed when b = 0");**

**} else {**

**System.out.println("Division of two numbers is: " + (a / b));**

**}**

**}**

**}**

**public class Cals {**

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

**Calculator2 C = new Calculator2();**

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

**System.out.println("Yashaswi reddy 24351 CSE-C”)**

**System.out.println("Enter two numbers:");**

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

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

**C.add(a, b);**

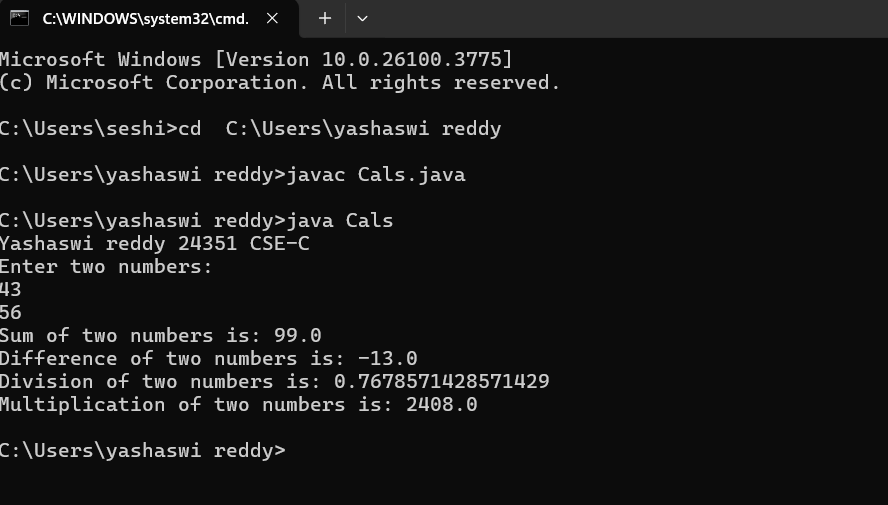
**C.subs(a, b);**

**C.div(a, b);**

**C.mul(a, b);**

**}**

**}**

****

**Important points:**

**In order to do this, we have to use inheritance concept. Here**

**we used the multi-inheritance concept.**

**Class Diagram:**

**+-----------------------------+**

**| Calculator |**

**+-----------------------------+**

**| + add(double, double): void |**

**| + subs(double, double): void|**

**+-----------------------------+**

**▲**

**|**

**+-----------------------------+**

**| Calculator1 |**

**+-----------------------------+**

**| + mul(double, double): void |**

**+-----------------------------+**

**▲**

**|**

**+-----------------------------+**

**| Calculator2 |**

**+-----------------------------+**

**| + div(double, double): void |**

**+-----------------------------+**

**+-----------------------------+**

**| Cals |**

**+-----------------------------+**

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

**+-----------------------------+**

**2.** **A vehicle rental company wants to develop a system**

**that maintains information about different types of vehicles**

**available for rent. The company rents out cars and bikes and**

**they need a program to store details about each vehicle**

**such as brand and speed.**

**i. Cars should have an additional property: number of doors,**

**Seating capacity.**

**ii. Bikes should have a property indicating whether they have**

**gears or not.**

**iii. The system should also include a function to display**

**details about each vehicle and indicate when a vehicle is**

**starting.**

**iv. Each class should have a constructor.**

**Questions:**

**1. Which OOP concept is used in the above program?**

**Explain why it is useful in this scenario.**

**2. If the company decides to add a new type of vehicle**

**‘Truck’, how would you modify the program?**

**a. Truck should include and additional property capacity**

**(in tons).**

**36**

**b. Create a showTruck() method to display the truck’s**

**capacity.**

**c. Write a constructor for truck that initializes all properties.**

**3. Implement the truck class and update the main method to**

**create a Truck object and also create an object for car and**

**bike subclasses. Finally display the details**

**Code:**

**class Vehicle {**

**String brand;**

**int speed;**

**Vehicle(String brand, int speed) {**

**this.brand = brand;**

**this.speed = speed;**

**}**

**void Details() {**

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

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

**}**

**}**

**class Cars extends Vehicle {**

**int doors;**

**int capacity;**

**public Cars(String brand, int speed, int doors, int capacity) {**

**super(brand, speed);**

**this.doors = doors;**

**this.capacity = capacity;**

**}**

**void cardetails() {**

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

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

**}**

**}**

**class Bikes extends Vehicle {**

**boolean gears;**

**Bikes(String brand, int speed, boolean gears) {**

**super(brand, speed);**

**this.gears = gears;**

**}**

**void bikedetails() {**

**if (gears)**

**System.out.println("This bike has gears.");**

**else**

**System.out.println("This bike does not have a gear system.");**

**}**

**}**

**class Trucks extends Vehicle {**

**int tons;**

**Trucks(String brand, int speed, int tons) {**

**super(brand, speed);**

**this.tons = tons;**

**}**

**void truckdetails() {**

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

**}**

**}**

**public class Rent {**

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

**Cars c = new Cars("Toyota", 120, 5, 5);**

**c.cardetails();**

**c.Details();**

**Bikes b = new Bikes("KTM", 80, true);**

**b.bikedetails();**

**b.Details();**

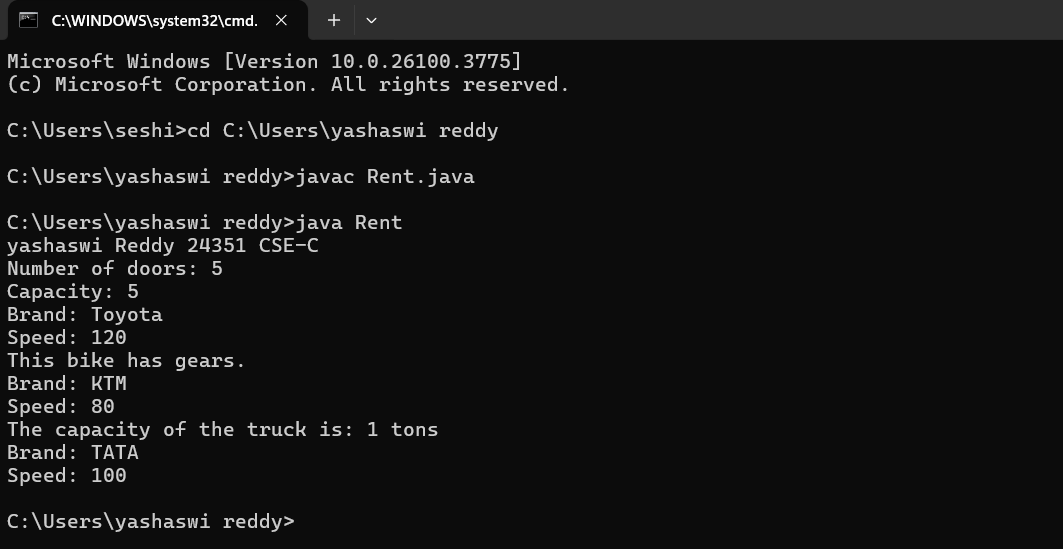
**Trucks t = new Trucks("TATA", 100, 1);**

**t.truckdetails();**

**t.Details();**

**}**

**}**

****

**Class Diagram:**

**+-----------------------------+**

**| Vehicle |**

**+-----------------------------+**

**| - brand: String |**

**| - speed: int |**

**+-----------------------------+**

**| + Details(): void |**

**+-----------------------------+**

**▲ ▲ ▲**

**| | |**

**+------------------+ +------------------+ +------------------+**

**| Cars | | Bikes | | Trucks |**

**+------------------+ +------------------+ +------------------+**

**| - doors: int | | - gears: boolean | | - tons: int |**

**| - capacity: int | +------------------+ +------------------+**

**| + cardetails() | | + bikedetails() | | + truckdetails() |**

**+------------------+ +------------------+ +------------------+**

**+-----------------------------+**

**| Rent |**

**+-----------------------------+**

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

**+-----------------------------+**

**Important points:**

**Multi-inheritance: It is one of the types of the inheritance**

**where subclass 2 inherits subclass1 and superclass.**

**Here Vehicle is the super class or parent class and**

**remaining cars, bikes, trucks are the subclasses or child**

**classes.**

**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, model,**

**fuel type, and colour using the constructor**

**Code:**

**class Vehicle {**

**String brand;**

**int speed;**

**public Vehicle() {**

**this.brand = "Unknown";**

**this.speed = 0;**

**}**

**void displayInfo() {**

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

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

**}**

**}**

**class Car extends Vehicle {**

**String model;**

**String company;**

**int mileage;**

**String fuelType;**

**public Car(String brand, int speed, String model, String company, int mileage, String fuelType) {**

**this.brand = brand;**

**this.speed = speed;**

**this.model = model;**

**this.company = company;**

**this.mileage = mileage;**

**this.fuelType = fuelType;**

**}**

**@Override**

**void displayInfo() {**

**super.displayInfo();**

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

**System.out.println("Company: " + company);**

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

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

**}**

**}**

**public class Help {**

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

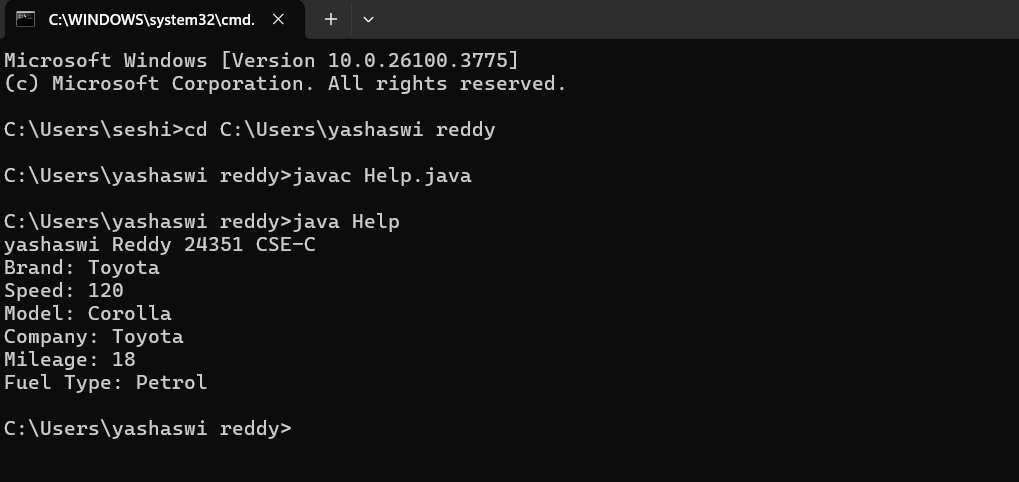
**System.out.println(“Yashaswi Reddy 24351 CSE-C”);**

**Car car = new Car("Toyota", 120, "Corolla", "Toyota", 18, "Petrol");**

**car.displayInfo();**

**}**

**}**

****

**Class Diagram:**

**+------------------+**

**| Vehicle |**

**+------------------+**

**| - brand: String |**

**| - speed: int |**

**+------------------+**

**| + Vehicle() |**

**| + displayInfo() |**

**+------------------+**

**▲**

**|**

**|**

**+------------------------------+**

**| Car |**

**+------------------------------+**

**| - model: String |**

**| - company: String |**

**| - mileage: int |**

**| - fuelType: String |**

**+------------------------------+**

**| + Car(brand, speed, ...) |**

**| + displayInfo() |**

**+------------------------------+**

**Main Method:**

**+----------------------------+**

**| Help |**

**+----------------------------+**

**| + main(String[] args) |**

**+----------------------------+**

**Important points:**

**In order to do this, we have to use inheritance concept. Here**

**we used the multi-inheritance concept.**

**2.** **Create a Java program for the scenario.**

**A college is developing an automated admission system that**

**verifies student eligibility for undergraduate (UG) and**

**postgraduate(PG) programs. Each program has different**

**eligibility criteria based on the student's percentage in their**

**previous qualification.**

**i) UG admissions require a minimum of 60%**

**ii) PG admissions require a minimum of 70%**

**Code:**

**class AdmissionSystem {**

**String studentName;**

**double percentage;**

**public AdmissionSystem(String studentName, double percentage) {**

**this.studentName = studentName;**

**this.percentage = percentage;**

**}**

**void checkEligibility() {**

**System.out.println("Checking eligibility for: " + studentName);**

**}**

**}**

**class UGAdmission extends AdmissionSystem {**

**public UGAdmission(String studentName, double percentage) {**

**super(studentName, percentage);**

**}**

**@Override**

**void checkEligibility() {**

**super.checkEligibility();**

**if (percentage >= 60) {**

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

**} else {**

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

**}**

**}**

**}**

**class PGAdmission extends AdmissionSystem {**

**public PGAdmission(String studentName, double percentage) {**

**super(studentName, percentage);**

**}**

**@Override**

**void checkEligibility() {**

**super.checkEligibility();**

**if (percentage >= 70) {**

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

**} else {**

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

**}**

**}**

**}**

**public class Main {**

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

**System.out.println(“Yashaswi Reddy 24351 CSE-C”);**

**UGAdmission ugStudent = new UGAdmission("John", 65);**

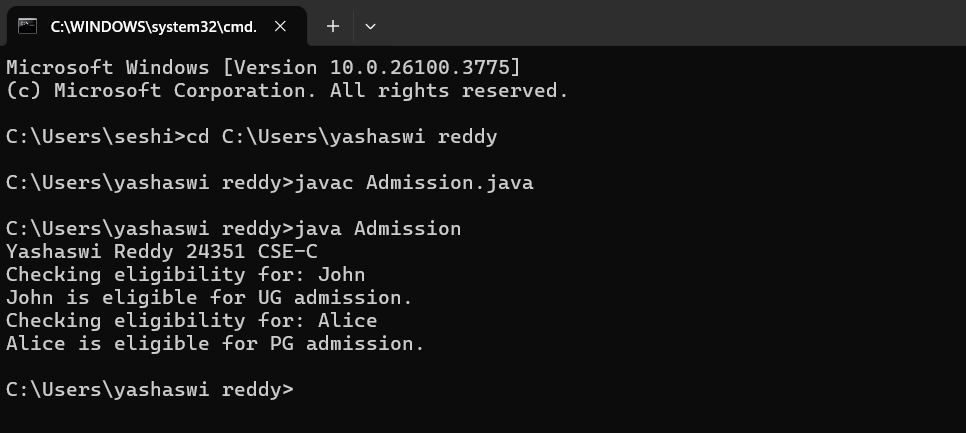
**ugStudent.checkEligibility();**

**PGAdmission pgStudent = new PGAdmission("Alice", 72);**

**pgStudent.checkEligibility();**

**}**

**}**

****

**+-------------------------------+**

**| AdmissionSystem |**

**+-------------------------------+**

**| - studentName: String |**

**| - percentage: double |**

**+-------------------------------+**

**| + AdmissionSystem(name, %) |**

**| + checkEligibility() |**

**+-------------------------------+**

**▲ ▲**

**| |**

**+------------------+ +------------------+**

**| UGAdmission | | PGAdmission |**

**+------------------+ +------------------+**

**| + UGAdmission(...)| | + PGAdmission(...)|**

**| + checkEligibility()| | + checkEligibility()|**

**+------------------+ +------------------+**

**Main Method:**

**+------------------------------+**

**| Main |**

**+------------------------------+**

**| + main(String[] args) |**

**+------------------------------+**

**Important points:**

**Super keyword is used take the method,variable,constructor**

**from the super class.**

**3.** **Write a Java Program to create a Calculator class with**

**overloaded methods to perform addition: Take the integer**

**values a and b from the user.**

**i) Add two integers**

**ii) Add two doubles**

**iii) Add three integer**

**Code:** **class Calculatoroverloading {**

**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 loading {**

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

**Sysstem.out.println(“Yashaswi Reddy 24351 CSE-C”);**

**Calculatoroverloading c = new Calculatoroverloading();**

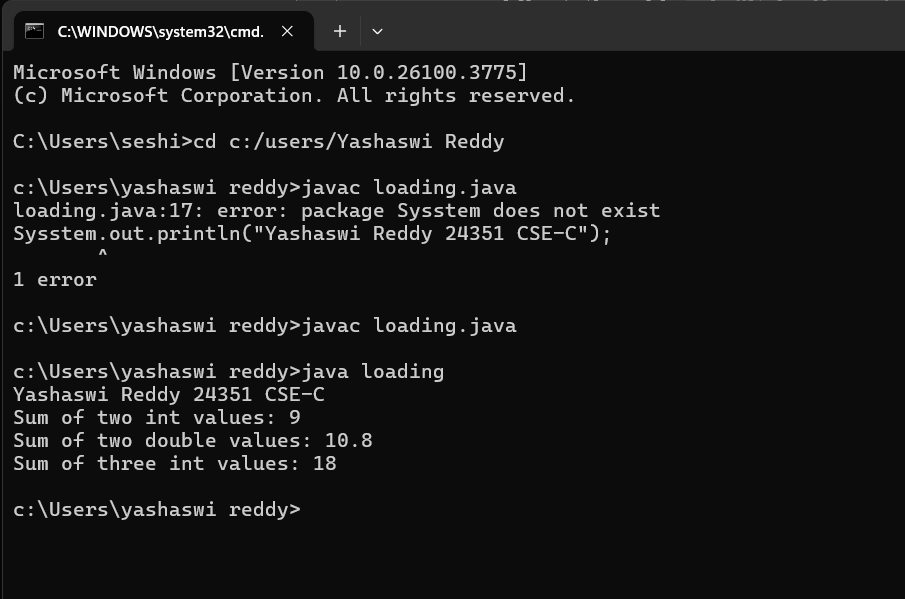
**System.out.println("Sum of two int values: " + c.add(5, 4));**

**System.out.println("Sum of two double values: " + c.add(5.9, 4.9));**

**System.out.println("Sum of three int values: " + c.add(5, 4, 9));**

**}**

**}**

****

**Class Diagram:**

**+------------------------------------+**

**| Calculatoroverloading |**

**+------------------------------------+**

**| + add(int a, int b): int |**

**| + add(double a, double b): double |**

**| + add(int a, int b, int c): int |**

**+------------------------------------+**

**▲**

**|**

**+------------------------------+**

**| loading |**

**+------------------------------+**

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

**+------------------------------+**

**Important points:**

**We should carefully pass the double and integer and**

**different types of input to an constructor when creating an**

**object to access the different constructors based on the**

**parameter.**

**4.** **Write a Java Program to 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.**

**Code:**

**class Shape {**

**void calculatearea(int l, int b) {**

**int area = l \* b;**

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

**}**

**void calculatearea(int x) {**

**int area = x \* x;**

**System.out.println("Area of Square is: " + area);**

**}**

**}**

**class Circle {**

**public void calculatearea(double pi, double r) {**

**double area = pi \* (r \* r);**

**System.out.println("Area of circle is: " + area);**

**}**

**}**

**public class Areas2 {**

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

**Shape s = new Shape();**

**Circle c = new Circle();**

**System.out.println(“Yashaswi Reddy 24351 CSE-C”);**

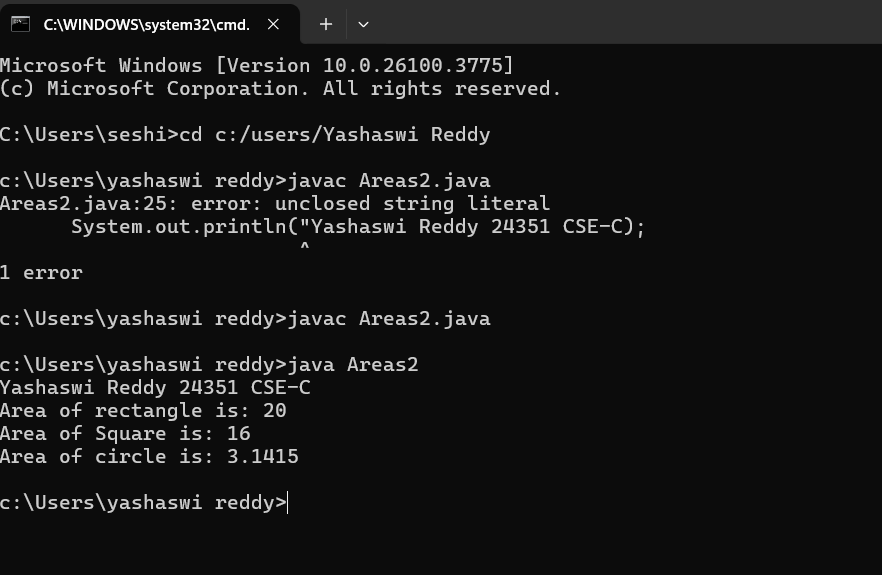
**s.calculatearea(4, 5);**

**s.calculatearea(4);**

**c.calculatearea(3.1415, 1);**

**}**

**}**

****

**Class Diagram:**

**+---------------------------+**

**| Shape |**

**+---------------------------+**

**| + calculatearea(int, int): void | ← Rectangle**

**| + calculatearea(int): void | ← Square**

**+---------------------------+**

**+---------------------------+**

**| Circle |**

**+---------------------------+**

**| + calculatearea(double, double): void | ← Circle**

**+---------------------------+**

**+---------------------------+**

**| Areas2 |**

**+---------------------------+**

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

**+---------------------------+**

**Important points:**

**In this program we use both method overloading and**

**overriding to calculate area of diterent shapes.**