23CSE111

# OBJECT-ORIENTED PROGRAMMING

**LAB REPORT**

**Department of Computer Science Engineering**
**Amrita School of Computing**
**Amrita Vishwa Vidyapeetham, Amaravati Campus**

**VERIFIED BY:**

**NAME : A. SAMHITHA**
**ROLLNO:AV.SC.U4CSE24011**

| | | | | |
|---|---|---|---|---|
| | a) Create a class with name “Car”<br>b) Create 4 attributes, named: car_color, car_brand, fuel_type, mileage<br>c) Create 3 methods, named: start(), service(), stop()<br>d) Create 3 objects, named: car1, car2, car3<br>e) Create a constructor, which should print, “Welcome to car garage”. | | | |
| 2. | To write a java program to create a class named BankAccount, with 2 methods deposit() and withdraw().<br>a) deposit(): Whenever an amount is deposited, it has to be update the current amount.<br>b) withdraw(): Whenever an amount is withdrawn, it has to be less than the current amount , else print (“Insufficient funds”) | | 15 | |
| | | | | |

## WEEK 01

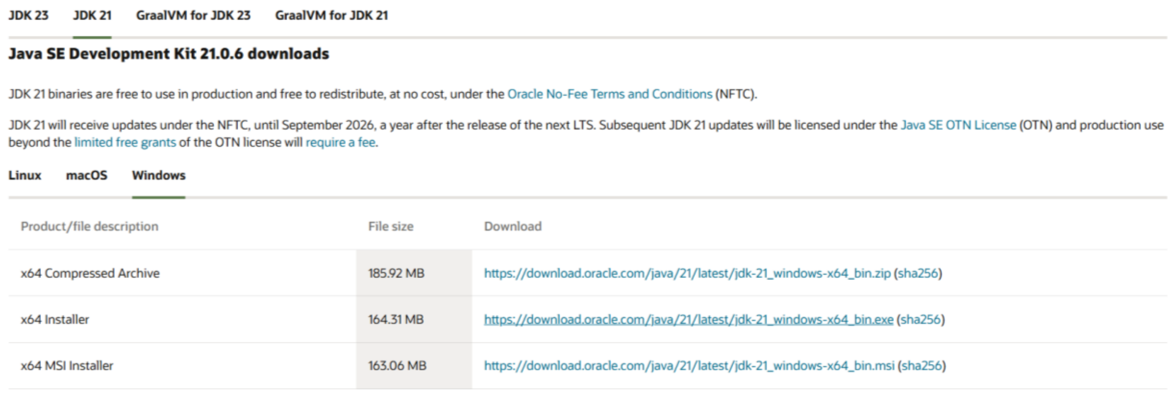
## PROGRAM-1:

**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.



**Java SE Development Kit 21.0.6 downloads**

JDK 21 binaries are free to use in production and free to redistribute, at no cost, under the Oracle No-Fee Terms and Conditions (NFTC).

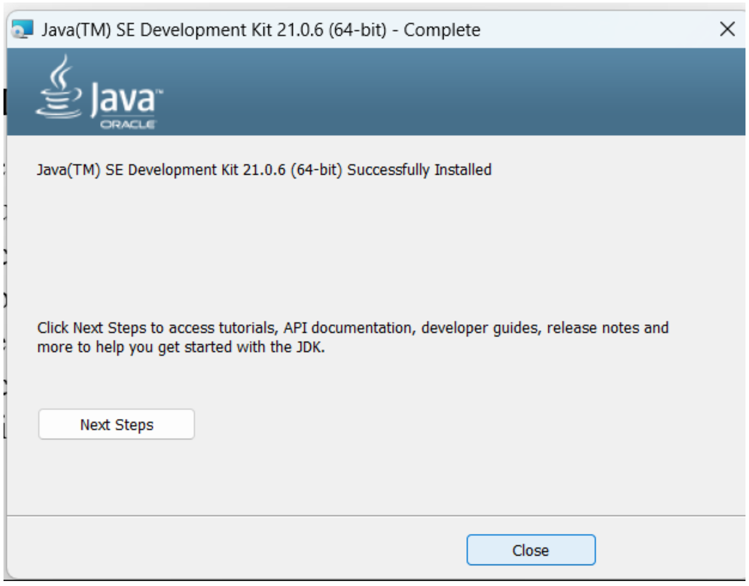
JDK 21 will receive updates under the NFTC, until September 2026, a year after the release of the next LTS. Subsequent JDK 21 updates will be licensed under the Java SE OTN License (OTN) and production use beyond the limited free grants of the OTN license will require a fee.

| Product/file description | File size | Download |
|---|---|---|
| x64 Compressed Archive | 185.92 MB | https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.zip (sha256) |
| x64 Installer | 164.31 MB | https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.exe (sha256) |
| x64 MSI Installer | 163.06 MB | https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.msi (sha256) |

**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.


Java(TM) SE Development Kit 21.0.6 (64-bit) - Complete

Java(TM) SE Development Kit 21.0.6 (64-bit) Successfully Installed

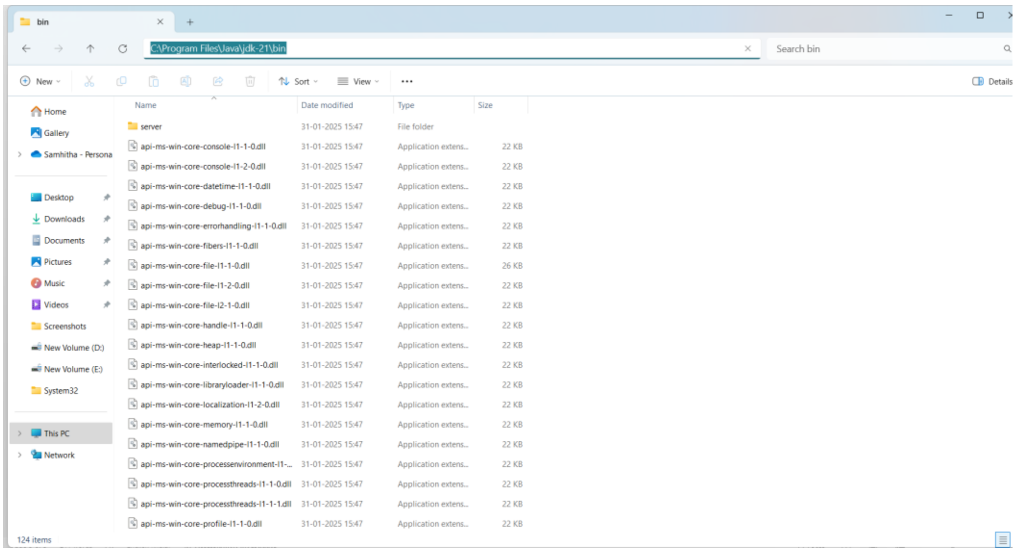
Click Next Steps to access tutorials, API documentation, developer guides, release notes and more to help you get started with the JDK.

## 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.


C:\Program Files\Java\jdk-21\bin

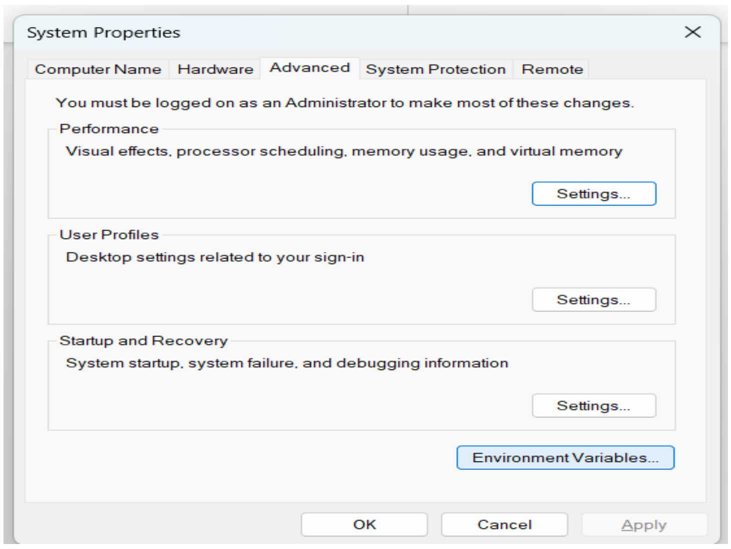
| Name | Date modified | Type | Size |
|---|---|---|---|
| server | 31-01-2025 15:47 | File folder | |
| api-ms-win-core-console-l1-1-0.dll | 31-01-2025 15:47 | Application extens... | 22 KB |
| api-ms-win-core-console-l1-2-0.dll | 31-01-2025 15:47 | Application extens... | 22 KB |
| api-ms-win-core-datetime-l1-1-0.dll | 31-01-2025 15:47 | Application extens... | 22 KB |
| api-ms-win-core-debug-l1-1-0.dll | 31-01-2025 15:47 | Application extens... | 22 KB |
| api-ms-win-core-errorhandling-l1-1-0.dll | 31-01-2025 15:47 | Application extens... | 22 KB |
| api-ms-win-core-fibers-l1-1-0.dll | 31-01-2025 15:47 | Application extens... | 22 KB |
| api-ms-win-core-file-l1-1-0.dll | 31-01-2025 15:47 | Application extens... | 26 KB |
| api-ms-win-core-file-l1-2-0.dll | 31-01-2025 15:47 | Application extens... | 22 KB |
| api-ms-win-core-file-l2-1-0.dll | 31-01-2025 15:47 | Application extens... | 22 KB |
| api-ms-win-core-handle-l1-1-0.dll | 31-01-2025 15:47 | Application extens... | 22 KB |
| api-ms-win-core-heap-l1-1-0.dll | 31-01-2025 15:47 | Application extens... | 22 KB |
| api-ms-win-core-interlocked-l1-1-0.dll | 31-01-2025 15:47 | Application extens... | 22 KB |
| api-ms-win-core-libraryloader-l1-1-0.dll | 31-01-2025 15:47 | Application extens... | 22 KB |
| api-ms-win-core-localization-l1-2-0.dll | 31-01-2025 15:47 | Application extens... | 22 KB |
| api-ms-win-core-memory-l1-1-0.dll | 31-01-2025 15:47 | Application extens... | 22 KB |
| api-ms-win-core-namedpipe-l1-1-0.dll | 31-01-2025 15:47 | Application extens... | 22 KB |
| api-ms-win-core-processenvironment-l1-... | 31-01-2025 15:47 | Application extens... | 22 KB |
| api-ms-win-core-processthreads-l1-1-0.dll | 31-01-2025 15:47 | Application extens... | 22 KB |
| api-ms-win-core-processthreads-l1-1-1.dll | 31-01-2025 15:47 | Application extens... | 22 KB |
| api-ms-win-core-profile-l1-1-0.dll | 31-01-2025 15:47 | Application extens... | 22 KB |

124 items

## 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.



You must be logged on as an Administrator to make most of these changes.

Performance

Visual effects, processor scheduling, memory usage, and virtual memory

User Profiles

Desktop settings related to your sign-in

Startup and Recovery

System startup, system failure, and debugging information

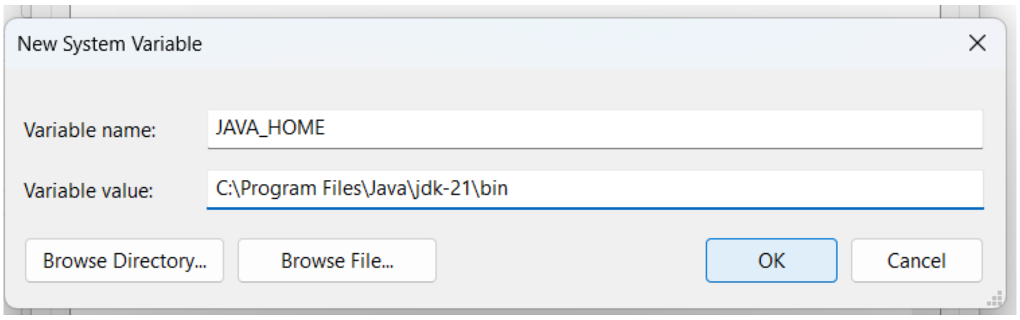
## Step 5: Set JAVA_HOME

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.



Variable name: JAVA_HOME

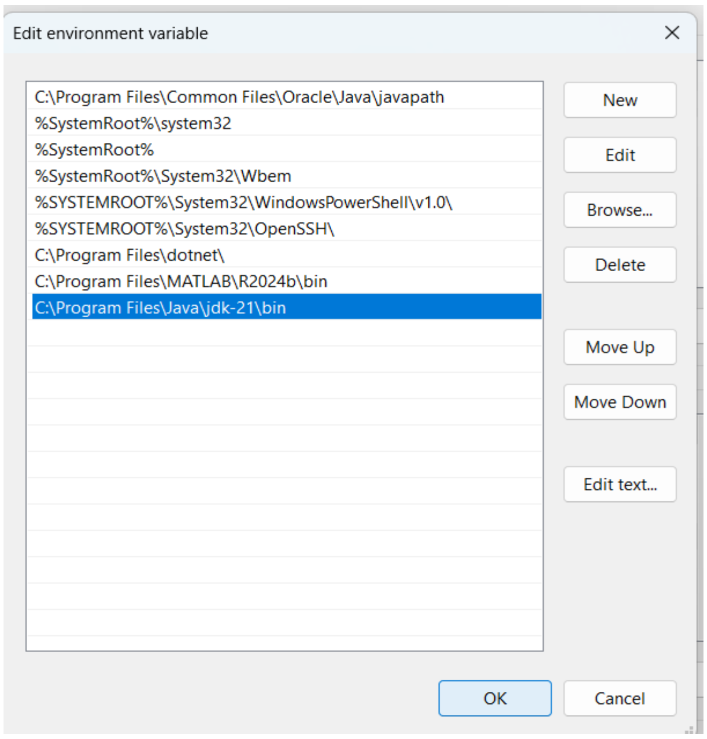
Variable value: C:\Program Files\Java\jdk-21\bin

## 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.

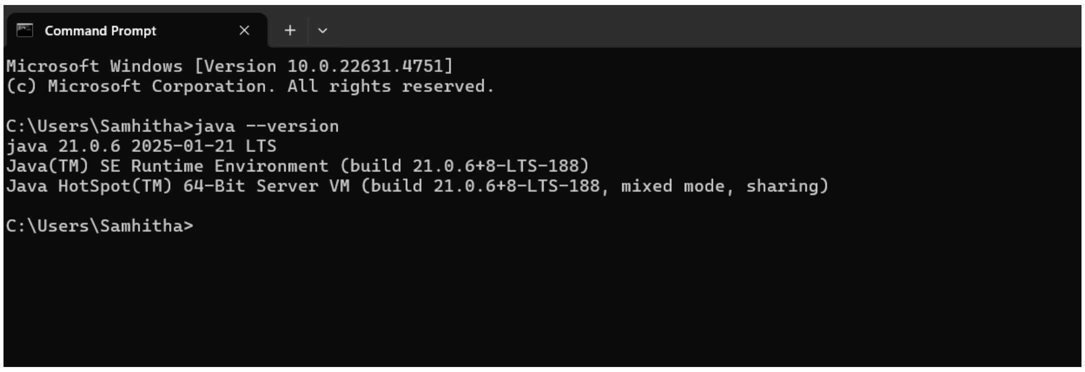


```
C:\Program Files\Common Files\Oracle\Java\javapath
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\dotnet\
C:\Program Files\MATLAB\R2024b\bin
C:\Program Files\Java\jdk-21\bin
```

## Step 7: Verify Installation

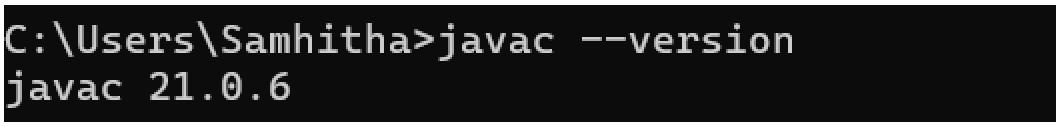
1) Open Command Prompt.

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



```
Microsoft Windows [Version 10.0.22631.4751]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Samhitha>java --version
java 21.0.6 2025-01-21 LTS
Java(TM) SE Runtime Environment (build 21.0.6+8-LTS-188)
Java HotSpot(TM) 64-Bit Server VM (build 21.0.6+8-LTS-188, mixed mode, sharing)

C:\Users\Samhitha>
```

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

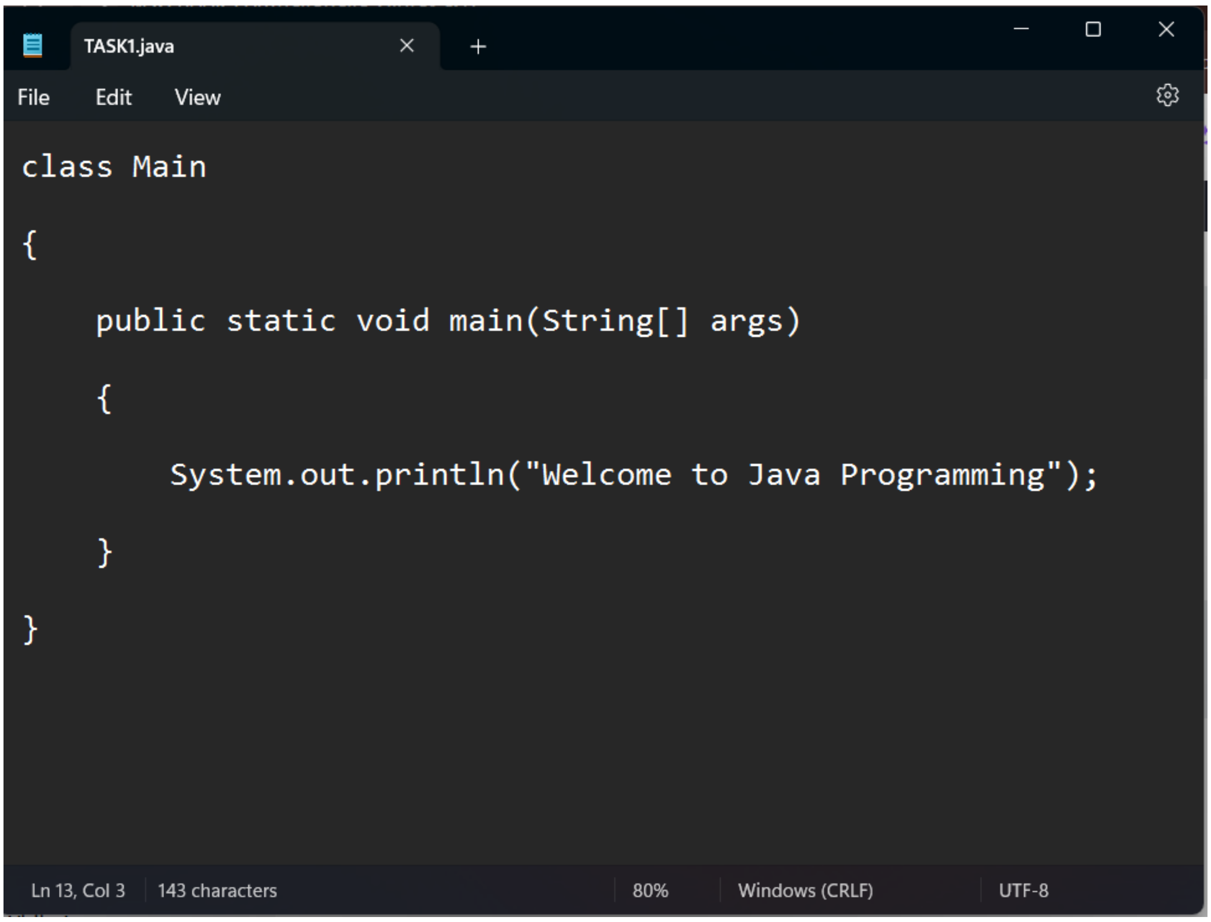


```
C:\Users\Samhitha>javac --version
javac 21.0.6
```

**PROGRAM-2:**

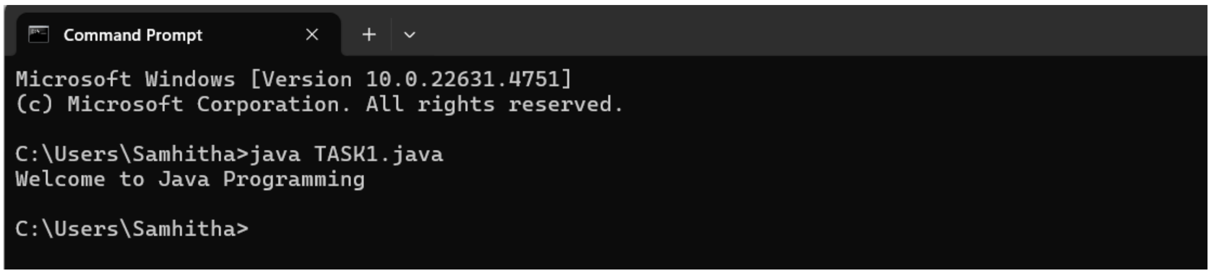
**AIM:** Write a Java program to print the message “Welcome to Java Programming.”

**CODE:**



```java
class Main
{
    public static void main(String[] args)
    {
        System.out.println("Welcome to Java Programming");
    }
}
```

**OUTPUT:**



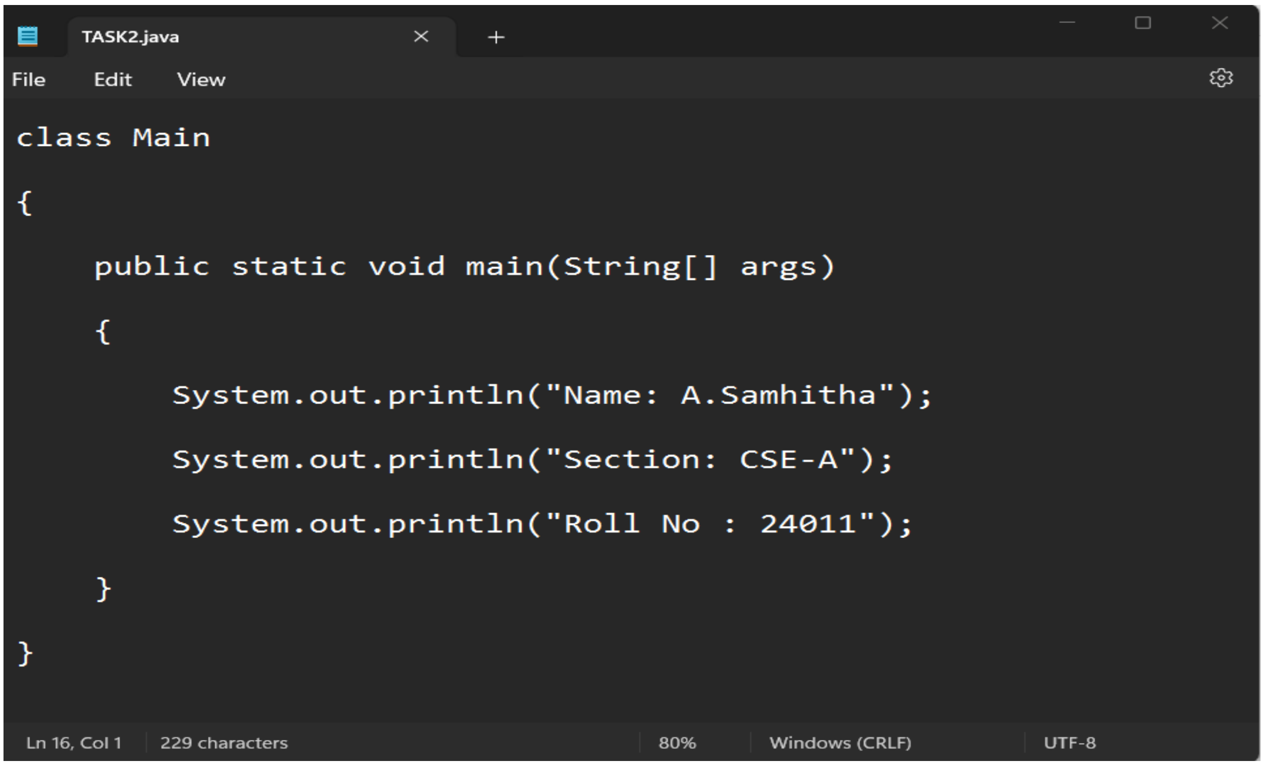
```
Microsoft Windows [Version 10.0.22631.4751]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Samhitha>java TASK1.java
Welcome to Java Programming

C:\Users\Samhitha>
```

**ERRORS:** None found

## PROGRAM-3:

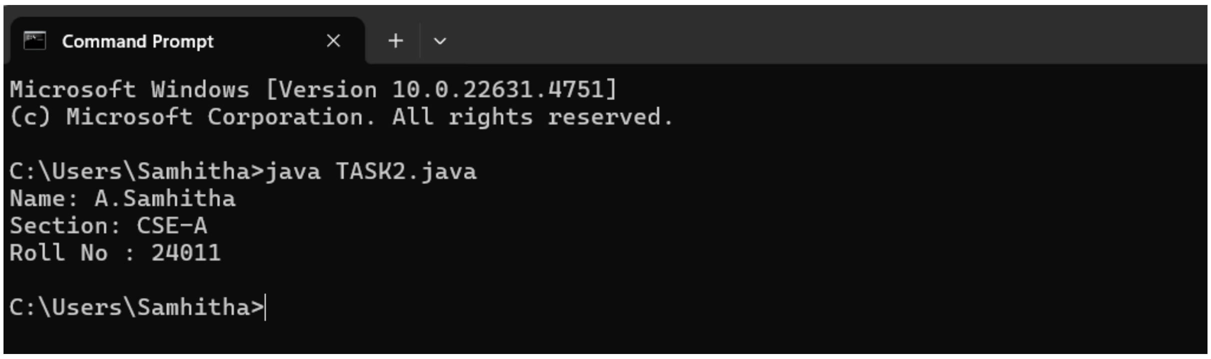
**AIM:** Write a Java Program that prints Name, Roll No, Section of a student.

**CODE:**



```java
class Main
{
    public static void main(String[] args)
    {
        System.out.println("Name: A.Samhitha");
        System.out.println("Section: CSE-A");
        System.out.println("Roll No : 24011");
    }
}
```

**OUTPUT:**



```
Microsoft Windows [Version 10.0.22631.4751]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Samhitha>java TASK2.java
Name: A.Samhitha
Section: CSE-A
Roll No : 24011

C:\Users\Samhitha>
```

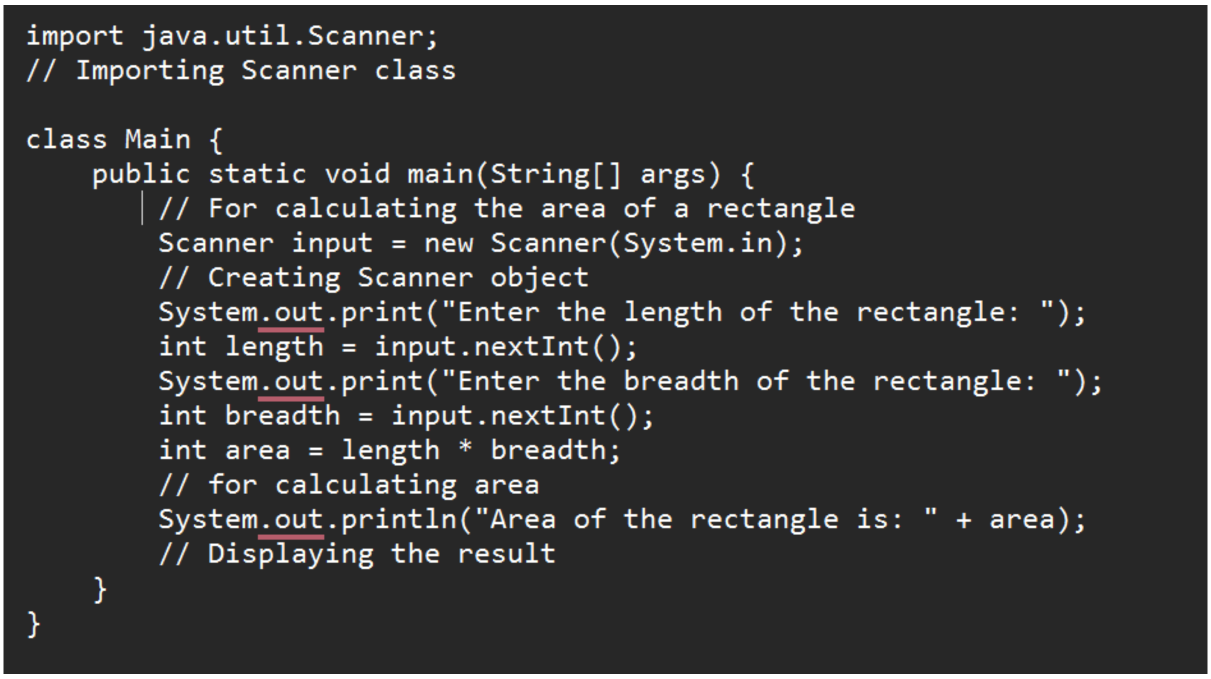
**ERRORS:** None Found

**WEEK-02**

**PROGRAM-01:**

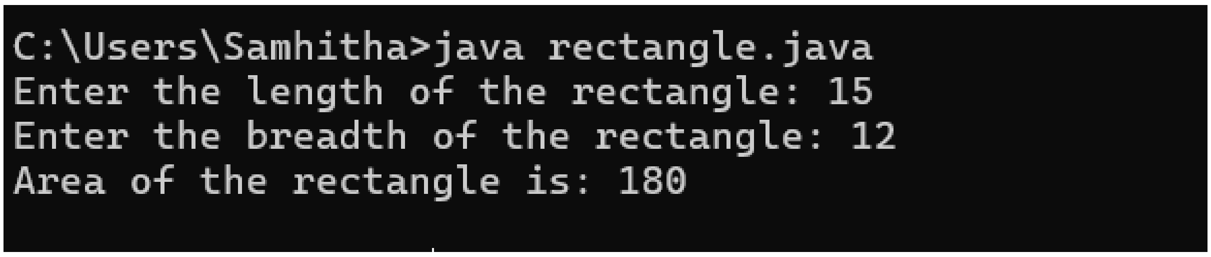
**AIM:** Write a java program to calculate the area of a rectangle.

**CODE:**



```
import java.util.Scanner;
// Importing Scanner class

class Main {
    public static void main(String[] args) {
        // For calculating the area of a rectangle
        Scanner input = new Scanner(System.in);
        // Creating Scanner object
        System.out.print("Enter the length of the rectangle: ");
        int length = input.nextInt();
        System.out.print("Enter the breadth of the rectangle: ");
        int breadth = input.nextInt();
        int area = length * breadth;
        // for calculating area
        System.out.println("Area of the rectangle is: " + area);
        // Displaying the result
    }
}
```

**OUTPUT:**



```
C:\Users\Samhitha>java rectangle.java
Enter the length of the rectangle: 15
Enter the breadth of the rectangle: 12
Area of the rectangle is: 180
```

**ERRORS:**

| S.No: | Errors | Error rectification |
|---|---|---|
| 1) | cannot find symbol: class Scanner | Import java.util.Scanner at the beginning of the |

|  |  |  |
|---|---|---|
|  |  | code: import java.util.Scanner; |
| **2)** | Scanner input = new Scanner(System.in); | Ensure the correct spelling and case of Scanner. No change is needed after import |

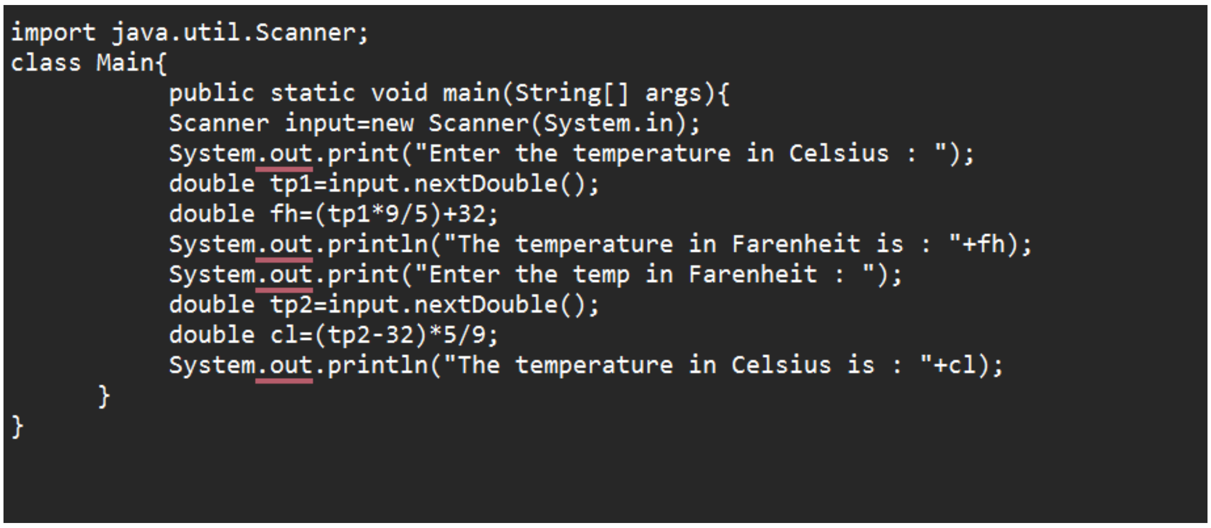
**Concepts to be known:**

1. import java.util.Scanner; - To accept input from user, Scanner class under util package has to be imported.
2. Scanner input=new Scanner(System.in); - Used to create a Scanner object
3. int ln=input.nextInt(); - Used to read the integer data type stored under the object created
4. System.out.println(" "); - It is used to print string inside the quotes. After printing, the cursor moves to the beginning of the next line.

**PROGRAM-02:**

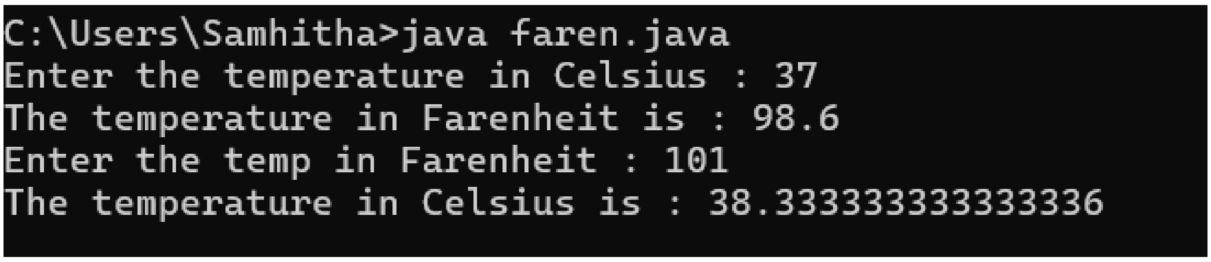
**AIM:** Write a java program to convert temperature from Celsius to Fahrenheit and vica-versa.

**CODE:**



```java
import java.util.Scanner;
class Main{
          public static void main(String[] args){
          Scanner input=new Scanner(System.in);
          System.out.print("Enter the temperature in Celsius : ");
          double tp1=input.nextDouble();
          double fh=(tp1*9/5)+32;
          System.out.println("The temperature in Farenheit is : "+fh);
          System.out.print("Enter the temp in Farenheit : ");
          double tp2=input.nextDouble();
          double cl=(tp2-32)*5/9;
          System.out.println("The temperature in Celsius is : "+cl);
     }
}
```

**OUTPUT:**



```
C:\Users\Samhitha>java faren.java
Enter the temperature in Celsius : 37
The temperature in Farenheit is : 98.6
Enter the temp in Farenheit : 101
The temperature in Celsius is : 38.333333333333336
```

**ERRORS:**

| S.No | Errors | Error Rectification |
|---|---|---|
| 1) | variable tp is already defined in method main(String[]) | Rename the first and second tp variable to avoid duplication. Variable names must be unique within the same scope. |

**Concepts to be known:**

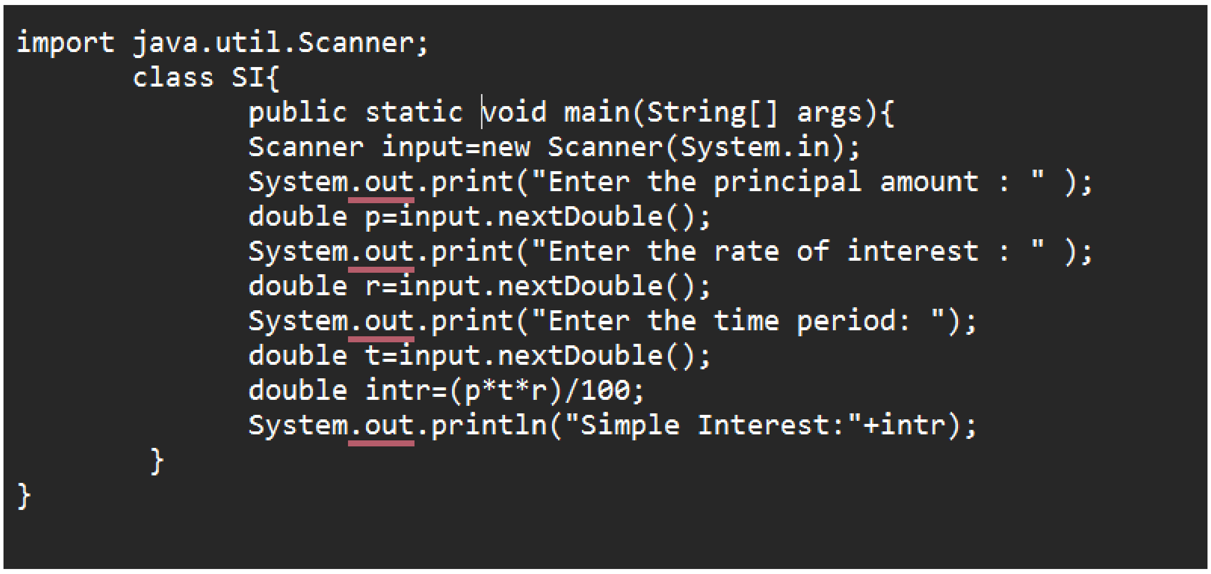
1. import java.util.Scanner; - To accept input from user, Scanner class under util package has to be imported.

2. Scanner input=new Scanner(System.in); - Used to create a Scanner object
3. double fh=input.nextDouble(); - Used to read double data type stored under the object created
4. System.out.println(“ “); - It is used to print string inside the quotes. After printing, the cursor moves to the beginning of the next line.

**PROGRAM-03:**

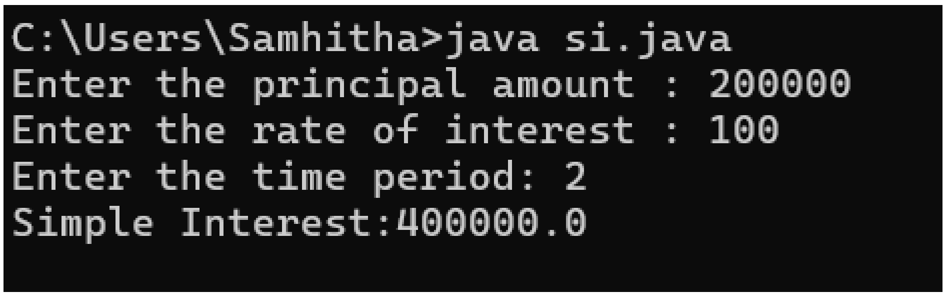
**AIM:** Write a java program to calculate the simple interest.

**CODE:**



```
import java.util.Scanner;
       class SI{
              public static void main(String[] args){
              Scanner input=new Scanner(System.in);
              System.out.print("Enter the principal amount : " );
              double p=input.nextDouble();
              System.out.print("Enter the rate of interest : " );
              double r=input.nextDouble();
              System.out.print("Enter the time period: ");
              double t=input.nextDouble();
              double intr=(p*t*r)/100;
              System.out.println("Simple Interest:"+intr);
        }
}
```

**OUTPUT:**



```
C:\Users\Samhitha>java si.java
Enter the principal amount : 200000
Enter the rate of interest : 100
Enter the time period: 2
Simple Interest:400000.0
```

**ERRORS:**

| Sno. | Error | Error rectification |
|---|---|---|
| 1. | error: ';' expected<br>double intr=(p*r*t)/100 | Add a semicolon at the end of the statement<br>double intr=(p*r*t)/100; |
| 2. | error: cannot find symbol<br>double intr=(p*r*t)/100;<br>symbol: variable p<br>location: class interest | Create a reader object<br>double p=input.nextDouble(); |

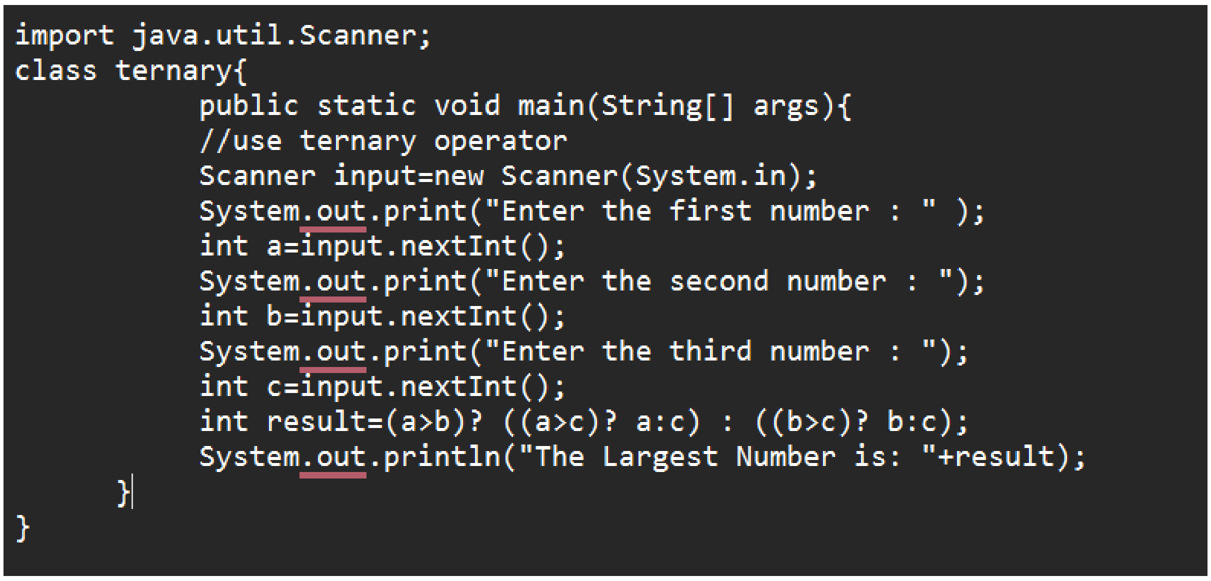
**Concepts to be known:**

1. import java.util.Scanner; - To accept input from user, Scanner class under util package has to be imported.
2. Scanner input=new Scanner(System.in); - Used to create a Scanner object
3. double p=input.nextDouble(); - Used to read double data type stored under the object created
4. System.out.println(“ “); - It is used to print string inside the quotes. After printing, the cursor moves to the beginning of the next line.

**PROGRAM-04:**

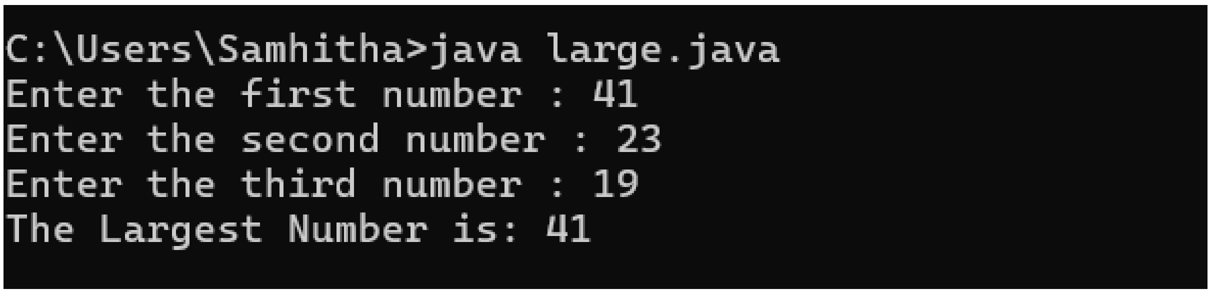
**AIM:** Write a java program to find the largest of three numbers, using ternary operator.

**CODE:**



```
import java.util.Scanner;
class ternary{
           public static void main(String[] args){
           //use ternary operator
           Scanner input=new Scanner(System.in);
           System.out.print("Enter the first number : " );
           int a=input.nextInt();
           System.out.print("Enter the second number : ");
           int b=input.nextInt();
           System.out.print("Enter the third number : ");
           int c=input.nextInt();
           int result=(a>b)? ((a>c)? a:c) : ((b>c)? b:c);
           System.out.println("The Largest Number is: "+result);
     }
}
```

**OUTPUT:**



```
C:\Users\Samhitha>java large.java
Enter the first number : 41
Enter the second number : 23
Enter the third number : 19
The Largest Number is: 41
```

**ERRORS:**

| **Sno.** | **Error** | **Error rectification** |
|---|---|---|
| 1) | error: ';' expected<br><br>int result=(a>b) ((a>c)? a:c) : ((b>c)? b:c);<br><br>error: not a statement<br><br>int result=(a>b) ((a>c)? a:c) : ((b>c)? b:c); | Add a '?'<br><br>int result=(a>b)? ((a>c)? a:c) : ((b>c)? b:c); |
| 2) | error: ';' expected | Add a ';' |

| | | |
|---|---|---|
| | int result=(a>b)? ((a>c)? a:c) : ((b>c)? b:c) | int result=(a>b)? ((a>c)? a:c) : ((b>c)? b:c); |

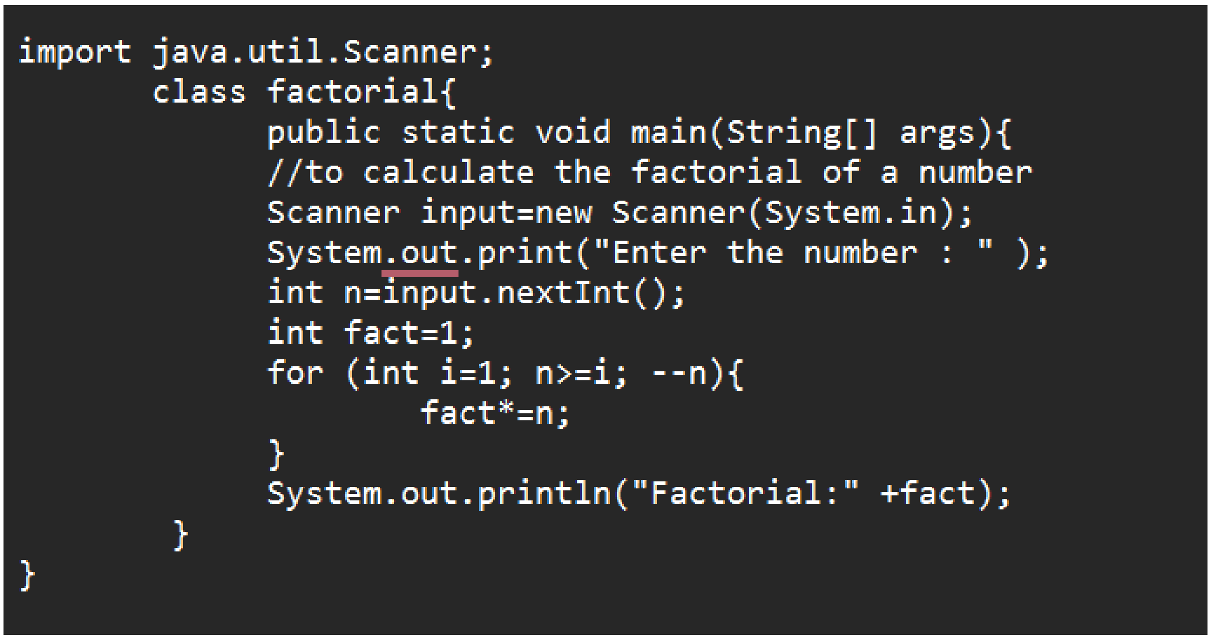
**Concepts to be known:**

1. import java.util.Scanner; - To accept input from user, Scanner class under util package has to be imported.
2. Scanner input=new Scanner(System.in); - Used to create a Scanner object
3. int a=input.nextInt (); - Used to read integer data type stored under the object created
4. int result=(a>b)? ((a>c)? a:c) : ((b>c)? b:c); - Nested Ternary operator is used here.Syntax for ternary operator is- condition? expression 1: expression 2; , whose answer is stored in a variable and then used.

**PROGRAM-05:**

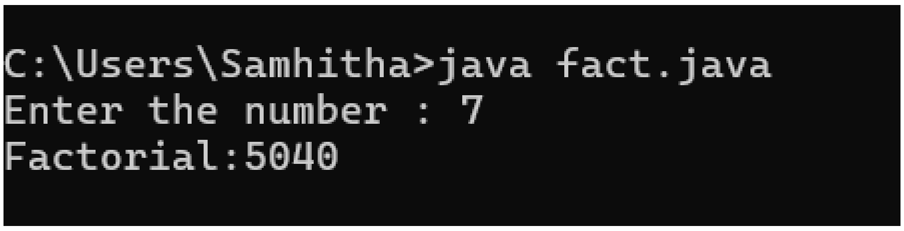
**AIM:** Write a java program to find the factorial of a number.

**CODE:**



```
import java.util.Scanner;
       class factorial{
              public static void main(String[] args){
              //to calculate the factorial of a number
              Scanner input=new Scanner(System.in);
              System.out.print("Enter the number : " );
              int n=input.nextInt();
              int fact=1;
              for (int i=1; n>=i; --n){
                     fact*=n;
              }
              System.out.println("Factorial:" +fact);
        }
}
```

**OUTPUT:**



```
C:\Users\Samhitha>java fact.java
Enter the number : 7
Factorial:5040
```

**ERROR:**

| **Sno.** | **Error** | **Error rectification** |
|---|---|---|
| 1. | error: ';' expected<br>fact*=n | Add a “;” fact*=n; |

**Concepts to be known:**

1. for (int i=1; n>=i;--n){ } - For loop syntax: for(initial expression; test expression; update expression){} The loop is executed, until the test expression evaluates to be false.

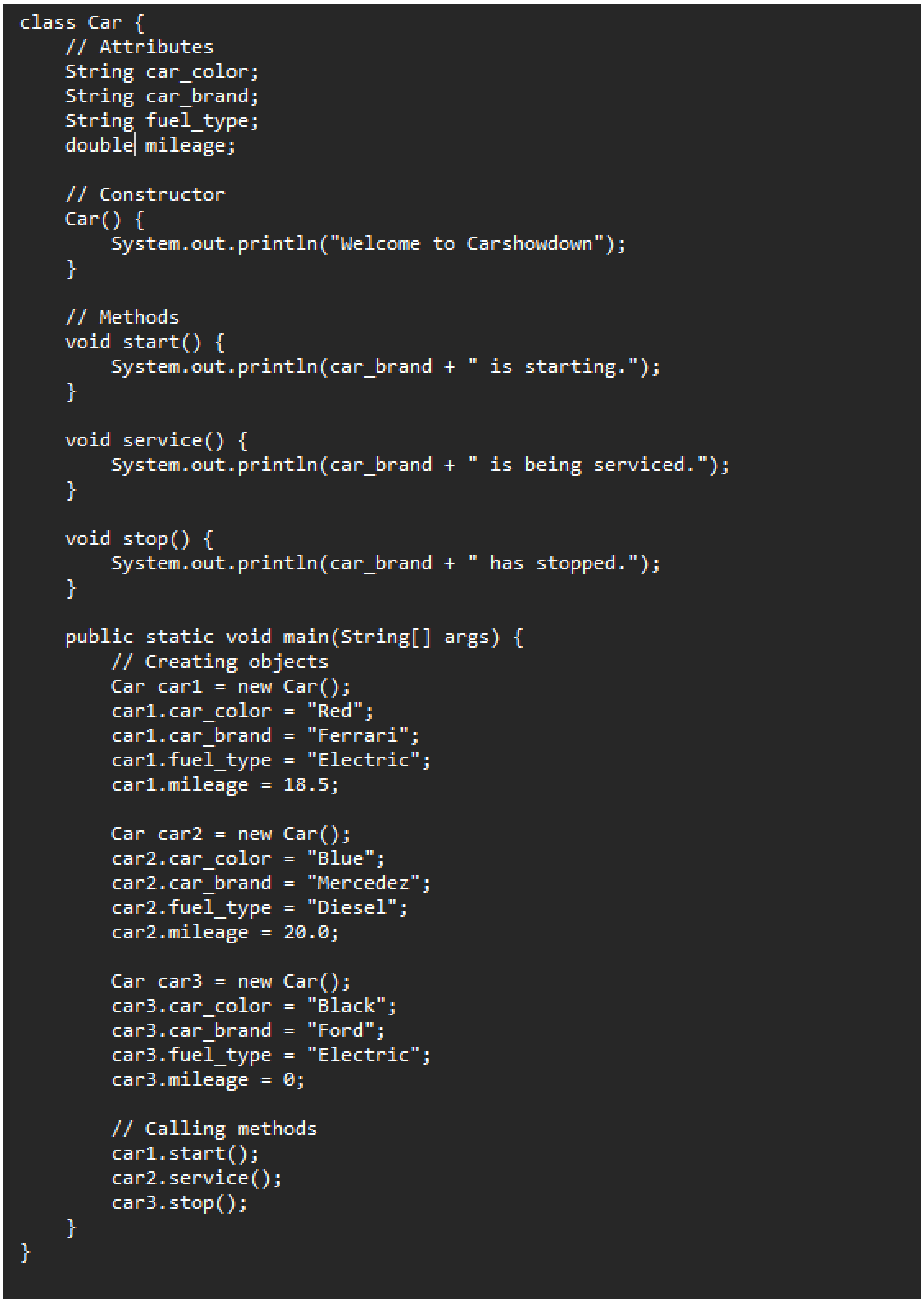
## WEEK-03

**PROGRAM-01:**

**AIM:** To create a java program with the following instructions:

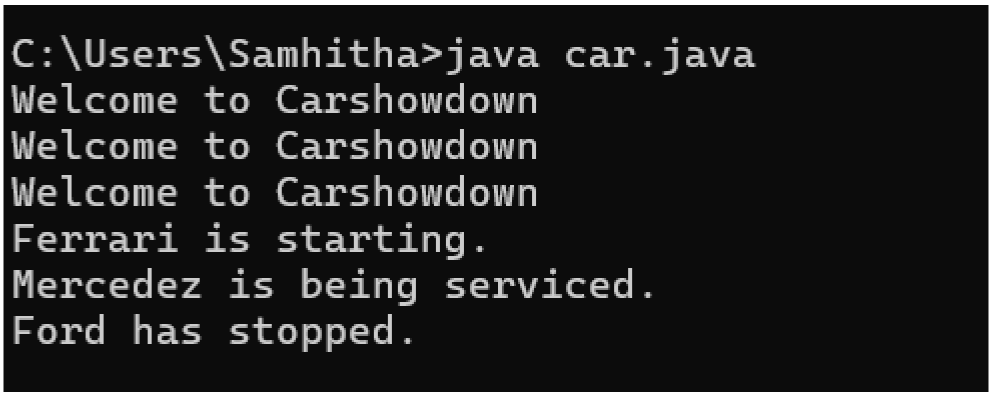
a) Create a class with name “Car”
b) Create 4 attributes, named: car_color, car_brand, fuel_type, mileage
c) Create 3 methods, named: start(), service(), stop()
d) Create 3 objects, named: car1, car2, car3
e) Create a constructor, which should print, “Welcome to car garage” .

**CODE:**



```
class Car {
    // Attributes
    String car_color;
    String car_brand;
    String fuel_type;
    double mileage;

    // Constructor
    Car() {
        System.out.println("Welcome to Carshowdown");
    }

    // Methods
    void start() {
        System.out.println(car_brand + " is starting.");
    }

    void service() {
        System.out.println(car_brand + " is being serviced.");
    }

    void stop() {
        System.out.println(car_brand + " has stopped.");
    }

    public static void main(String[] args) {
        // Creating objects
        Car car1 = new Car();
        car1.car_color = "Red";
        car1.car_brand = "Ferrari";
        car1.fuel_type = "Electric";
        car1.mileage = 18.5;

        Car car2 = new Car();
        car2.car_color = "Blue";
        car2.car_brand = "Mercedez";
        car2.fuel_type = "Diesel";
        car2.mileage = 20.0;

        Car car3 = new Car();
        car3.car_color = "Black";
        car3.car_brand = "Ford";
        car3.fuel_type = "Electric";
        car3.mileage = 0;

        // Calling methods
        car1.start();
        car2.service();
        car3.stop();
    }
}
```

## OUTPUT:



```
C:\Users\Samhitha>java car.java
Welcome to Carshowdown
Welcome to Carshowdown
Welcome to Carshowdown
Ferrari is starting.
Mercedez is being serviced.
Ford has stopped.
```

**ERRORS:**

| S.No | Error | Error rectification |
|---|---|---|
| **1)** | error: ')' or ',' expected at line System.out.println(car_brand + " is being serviced."); | There is an issue with the string concatenation. Ensure proper quotes and spacing. |
| **2)** | error: not a statement at line System.out.println(car_brand + " is being serviced."); | Verify that all syntax is correct. It is likely due to a misplaced or missing character. |
| **3)** | error: ';' expected at line System.out.println(car_brand + " is being serviced."); | Check for missing or extra quotes, plus signs, or misplaced semicolons. |

**Class Diagram:**

| Car |
|---|
| + car_color: String<br>+ car_brand: String<br>+ fuel_type: String<br>+ mileage: int |
| + Car(): void<br>+ start(): void |

|  |
|---|
| + service(): void<br>+ stop(): void |

**Concepts to be known:**

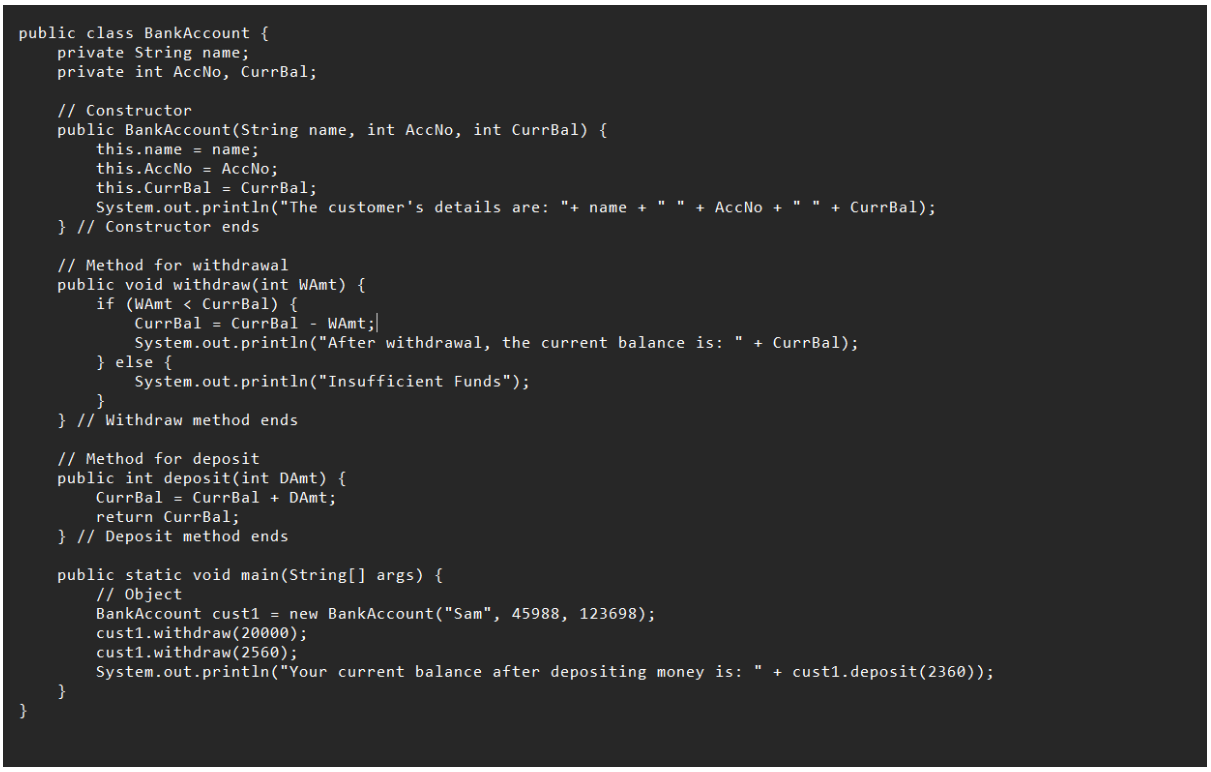
1. String car_color; declares an instance variable car_color of type String, which will store the color of the car.
2. Car() is a constructor that gets executed when an object of the Car class is created, printing "Welcome to Carshowdown".
3. this.car_color = car_color; is used inside a constructor to assign the passed parameter to the instance variable of the same name.
4. Methods like start(), service(), and stop() define specific actions for the car, such as printing messages related to the car's state.
5. Creating objects like Car car1 = new Car(); allows assigning values to attributes and calling methods like car1.start(); to execute their functionality.

**PROGRAM-02:**

**AIM:** To write a java program to create a class named BankAccount, with 2 methods deposit() and withdraw().

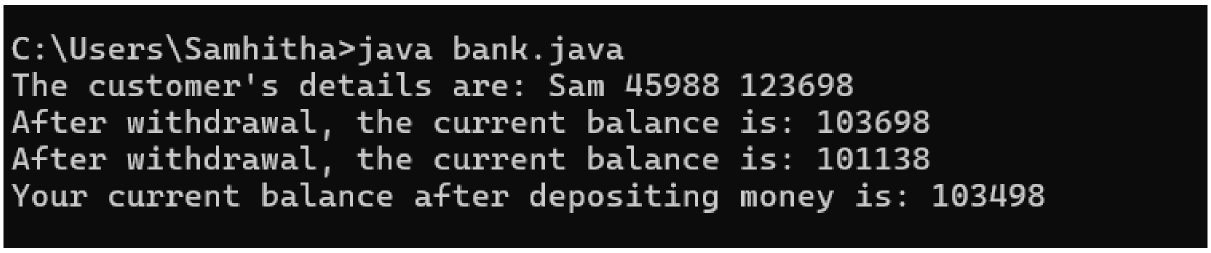
a) deposit(): Whenever an amount is deposited, it has to be update the current amount.
b) withdraw(): Whenever an amount is withdrawn, it has to be less than the current amount , else print ("Insufficient funds")

**CODE:**



```java
public class BankAccount {
    private String name;
    private int AccNo, CurrBal;

    // Constructor
    public BankAccount(String name, int AccNo, int CurrBal) {
        this.name = name;
        this.AccNo = AccNo;
        this.CurrBal = CurrBal;
        System.out.println("The customer's details are: "+ name + " " + AccNo + " " + CurrBal);
    } // Constructor ends

    // Method for withdrawal
    public void withdraw(int WAmt) {
        if (WAmt < CurrBal) {
            CurrBal = CurrBal - WAmt;
            System.out.println("After withdrawal, the current balance is: " + CurrBal);
        } else {
            System.out.println("Insufficient Funds");
        }
    } // Withdraw method ends

    // Method for deposit
    public int deposit(int DAmt) {
        CurrBal = CurrBal + DAmt;
        return CurrBal;
    } // Deposit method ends

    public static void main(String[] args) {
        // Object
        BankAccount cust1 = new BankAccount("Sam", 45988, 123698);
        cust1.withdraw(20000);
        cust1.withdraw(2560);
        System.out.println("Your current balance after depositing money is: " + cust1.deposit(2360));
    }
}
```

## OUTPUT:



```
C:\Users\Samhitha>java bank.java
The customer's details are: Sam 45988 123698
After withdrawal, the current balance is: 103698
After withdrawal, the current balance is: 101138
Your current balance after depositing money is: 103498
```

## ERRORS:

| Sno. | Error | Error rectification |
|---|---|---|
| 1. | error: ';' expected cust1.withdraw(3050) | Add a ";" <br><br> cust1.withdraw(3050); |
| 2. | error: cannot find symbol thisCurrBal=CurrBal; | Add a "." <br> this.CurrBal=CurrBal; |

**CLASS DIAGRAM:**

| BankAccount |
|---|
| - name: String<br>- Accno: int<br>- CurrBal: int |
| BankAccount: void<br>+ withdraw(int WAmt): void<br>+ deposit(int DAmt): int |

**Concepts to be known:**

1) Classes and Objects – The program defines a BankAccount class and creates an object (cust1) to access methods and store account details.
2) Constructors – The constructor BankAccount(String name, int AccNo, int CurrBal) initializes the object with values when a new bank account is created.
3) Instance Variables – The program uses name, AccNo, and CurrBal as private instance variables to store customer details and account balance.
4) Access Modifiers – The private keyword ensures that instance variables cannot be accessed directly from outside the class, maintaining encapsulation.
5) Methods (Functions) – The withdraw(int WAmt) method deducts money from the balance, and deposit(int DAmt) adds money and returns the updated balance.
6) Conditional Statements – The if-else condition in withdraw checks if the withdrawal amount is less than the current balance before proceeding.

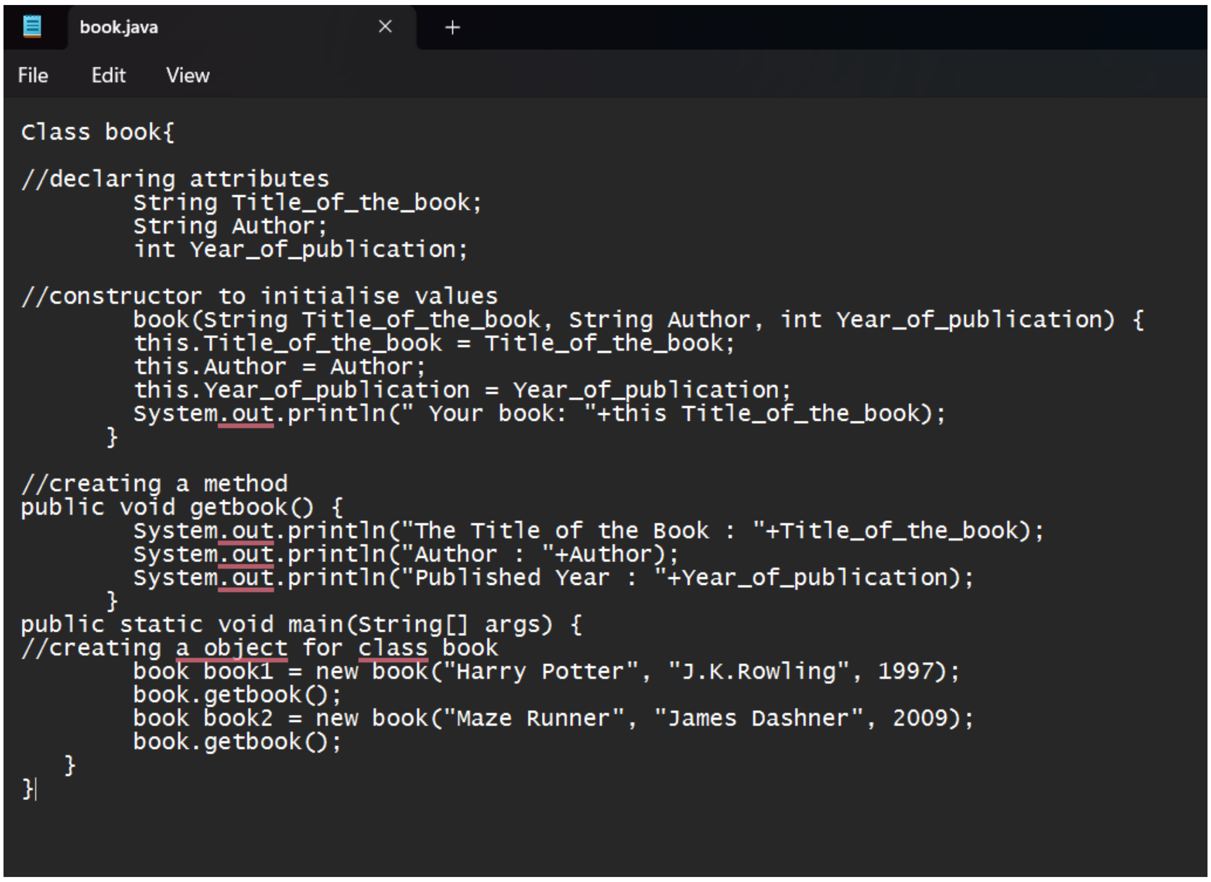
7) Return Statements – The deposit method returns the updated balance after adding the deposited amount.
8) Printing Output (System.out.println) – The program prints account details, withdrawal status, and the new balance after deposits.
9) main Method – The program starts execution from the main method, where an object is created, and methods are called.
10) Basic Error Debugging – Understanding common Java errors like misspelled method names (depost → deposit), incorrect keywords (retirn → return), and missing braces helps in fixing compilation issues.

## WEEK-04

**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.

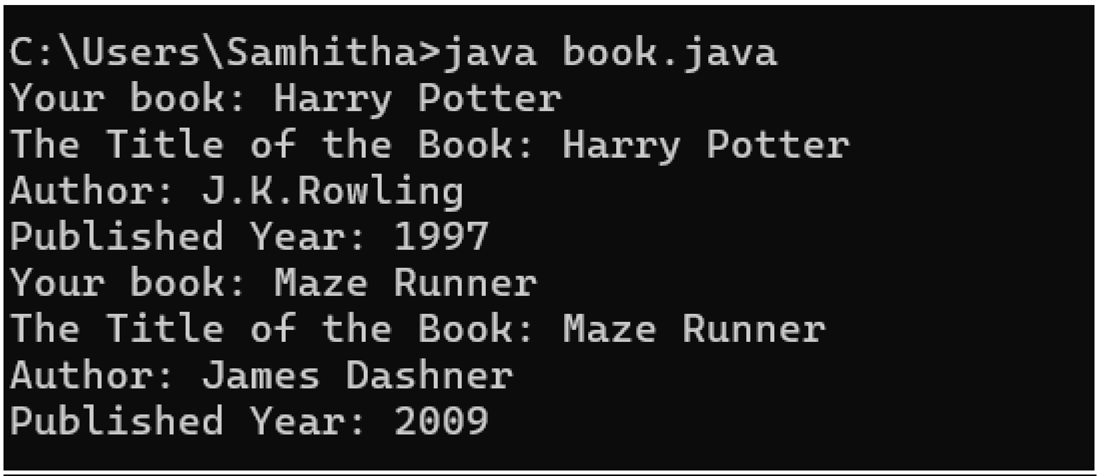
Create a method which displays the details of the book and display the details of two books.

**CODE:**



```
Class book{

//declaring attributes
        String Title_of_the_book;
        String Author;
        int Year_of_publication;

//constructor to initialise values
        book(String Title_of_the_book, String Author, int Year_of_publication) {
        this.Title_of_the_book = Title_of_the_book;
        this.Author = Author;
        this.Year_of_publication = Year_of_publication;
        System.out.println(" Your book: "+this Title_of_the_book);
      }

//creating a method
public void getbook() {
        System.out.println("The Title of the Book : "+Title_of_the_book);
        System.out.println("Author : "+Author);
        System.out.println("Published Year : "+Year_of_publication);
      }
public static void main(String[] args) {
//creating a object for class book
        book book1 = new book("Harry Potter", "J.K.Rowling", 1997);
        book.getbook();
        book book2 = new book("Maze Runner", "James Dashner", 2009);
        book.getbook();
   }
}
```

## OUTPUT:



```
C:\Users\Samhitha>java book.java
Your book: Harry Potter
The Title of the Book: Harry Potter
Author: J.K.Rowling
Published Year: 1997
Your book: Maze Runner
The Title of the Book: Maze Runner
Author: James Dashner
Published Year: 2009
```

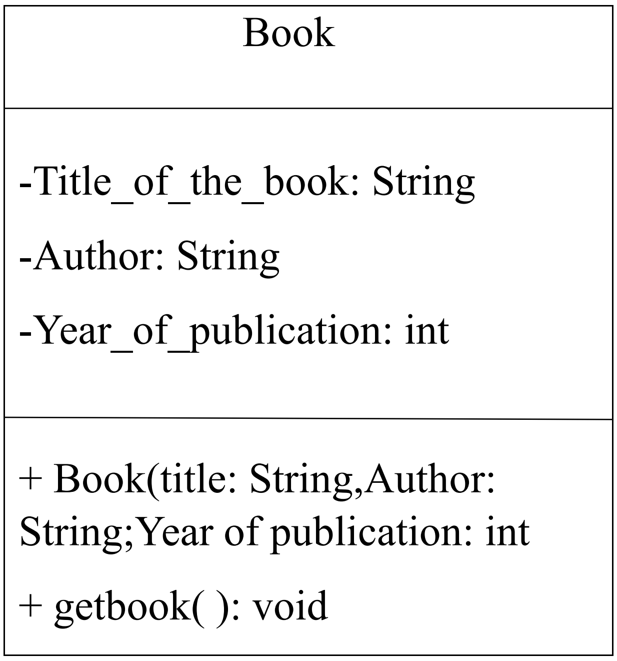
## ERROR:

| **Sno.** | **Error Message** | **Error rectification** |
|---|---|---|
| 1. | System.out.println(" Your book: "+this Title_of_the_book); - Syntax error | Change this Title_of_the_book to this.Title_of_the_book. |
| 2. | book.getbook(); - Error in calling method | Replace book.getbook(); with book1.getbook(); and book2.getbook();. |
| 3. | book book1 = new book("Harry Potter", "J.K.Rowling", 1997); - Class name issue | Change book to Book everywhere in the file |

## IMPORTANT POINTS:

1) The class Book follows Java naming conventions and represents a real-world book entity.
2) Stores book details (Title_of_the_book, Author, Year_of_publication).
3) Initializes object properties using the this keyword to avoid variable name conflicts.
4) Displays book details, demonstrating encapsulation and controlled data access.
5) Two book objects (book1, book2) are instantiated and used.
6) Class names should start with an uppercase letter, and the constructor must match the class name exactly.

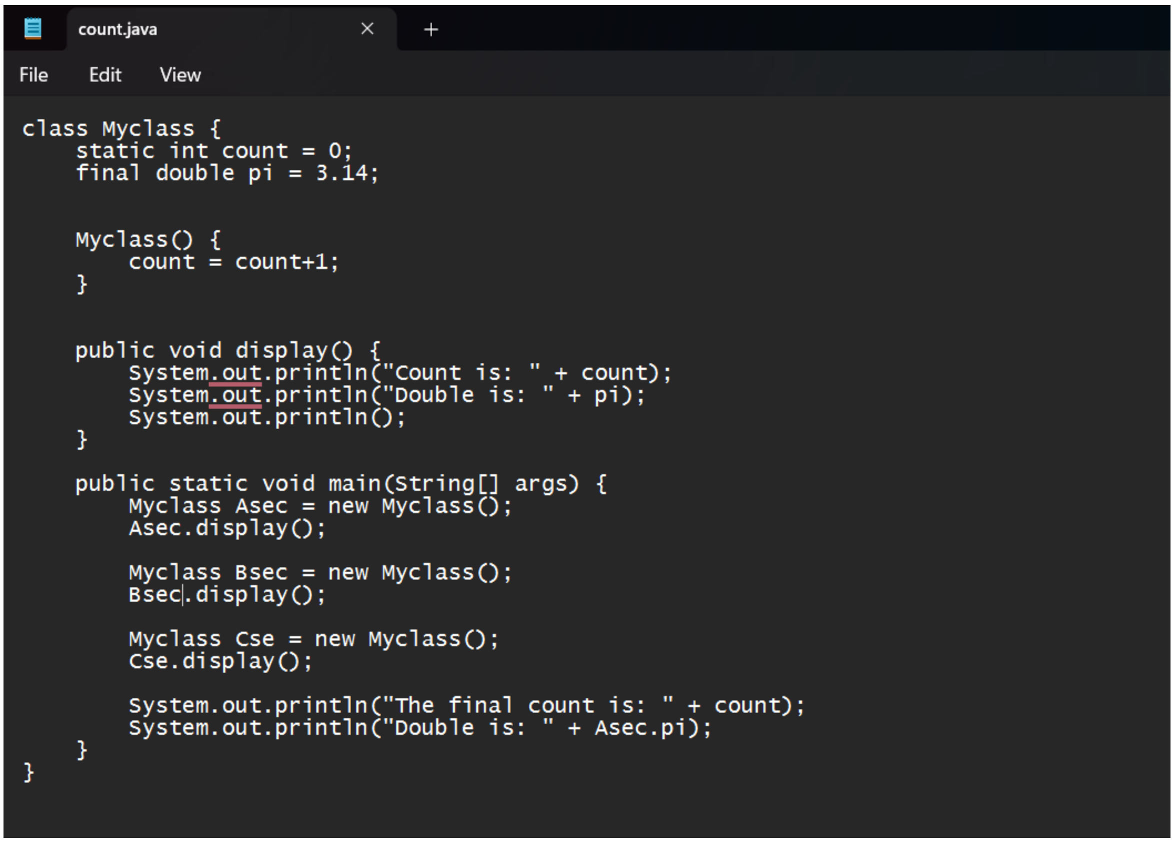
## CLASS DIAGRAM:


Book
-Title_of_the_book: String
-Author: String
-Year_of_publication: int
+ Book(title: String,Author: String;Year of publication: int
+ getbook( ): void

**PROGRAM-2:**

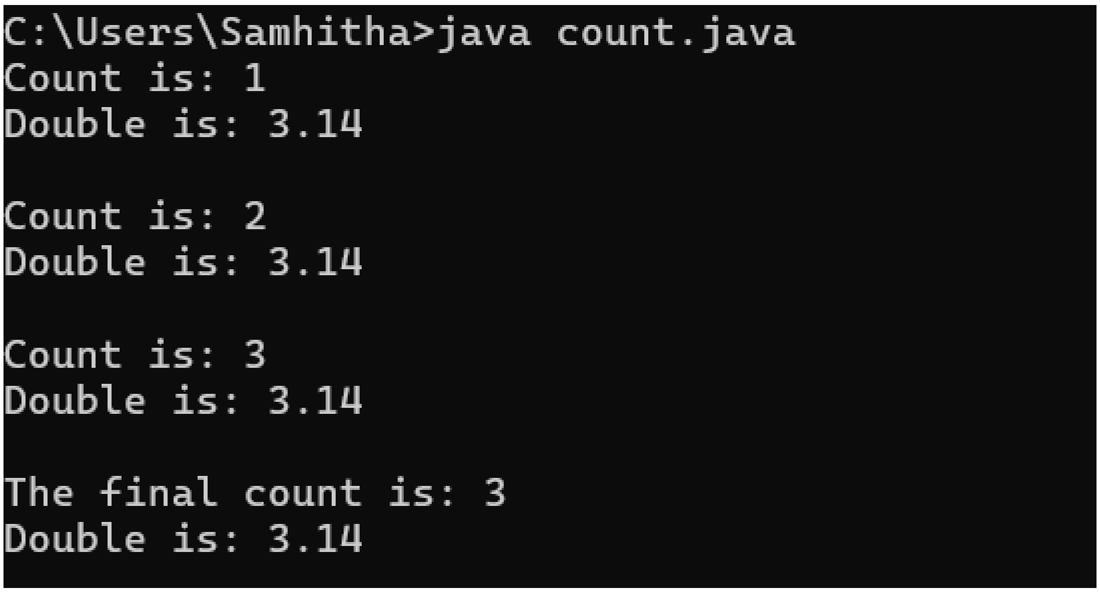
**AIM:**Create a java Program with class named myclass with static variable count of int type, initialized to zero and a constant variable "pi" of type double initialized to 3.14 as attributes of the class, ow define a constructor for "myclass" that increments the count variable each time an object of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

**CODE:**



```
class Myclass {
    static int count = 0;
    final double pi = 3.14;


    Myclass() {
        count = count+1;
    }


    public void display() {
        System.out.println("Count is: " + count);
        System.out.println("Double is: " + pi);
        System.out.println();
    }

    public static void main(String[] args) {
        Myclass Asec = new Myclass();
        Asec.display();

        Myclass Bsec = new Myclass();
        Bsec.display();

        Myclass Cse = new Myclass();
        Cse.display();

        System.out.println("The final count is: " + count);
        System.out.println("Double is: " + Asec.pi);
    }
}
```

**OUTPUT:**



```
C:\Users\Samhitha>java count.java
Count is: 1
Double is: 3.14

Count is: 2
Double is: 3.14

Count is: 3
Double is: 3.14

The final count is: 3
Double is: 3.14
```

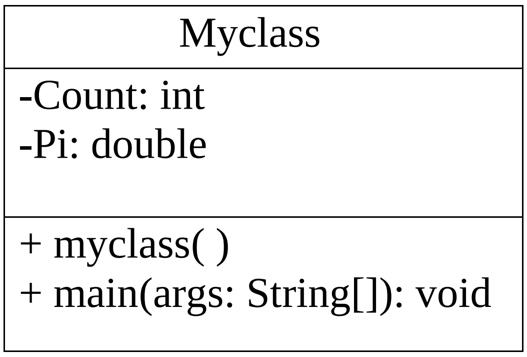
**ERROR:**

| Sno. | Error Message | Error rectification |
|---|---|---|
| 1. | count.java:2: error: <identifier> expected at Static int count = 0; | Change Static to static. |
| 2. | count.java:3: error: <identifier> expected at final doublepi = 3.1415; | Change final doublepi = 3.1415; to final double pi = 3.1415;. |
| 3. | count.java:16: error: <identifier> expected at public Static void main(String[] args) { | Change Static to static. |

**IMPORTANT POINTS:**

1) Asec.display() and Bsec.display() access the instance methods and variables through their respective object references.
2) System.out.println("Double is :"+Bsec.pi); accesses that pi variable of the Bsec object.
3) new keyword followed by the class constructor. This allocates memory for the object and initializes its attributes.
4) new is necessary for creating objects and invoking constructors.
5) Object References are needed to access instance variables and methods.
6) final double pi means that once pi is initialized with the value 3.14, it cannot be changed.

**CLASS DIAGRAM:**

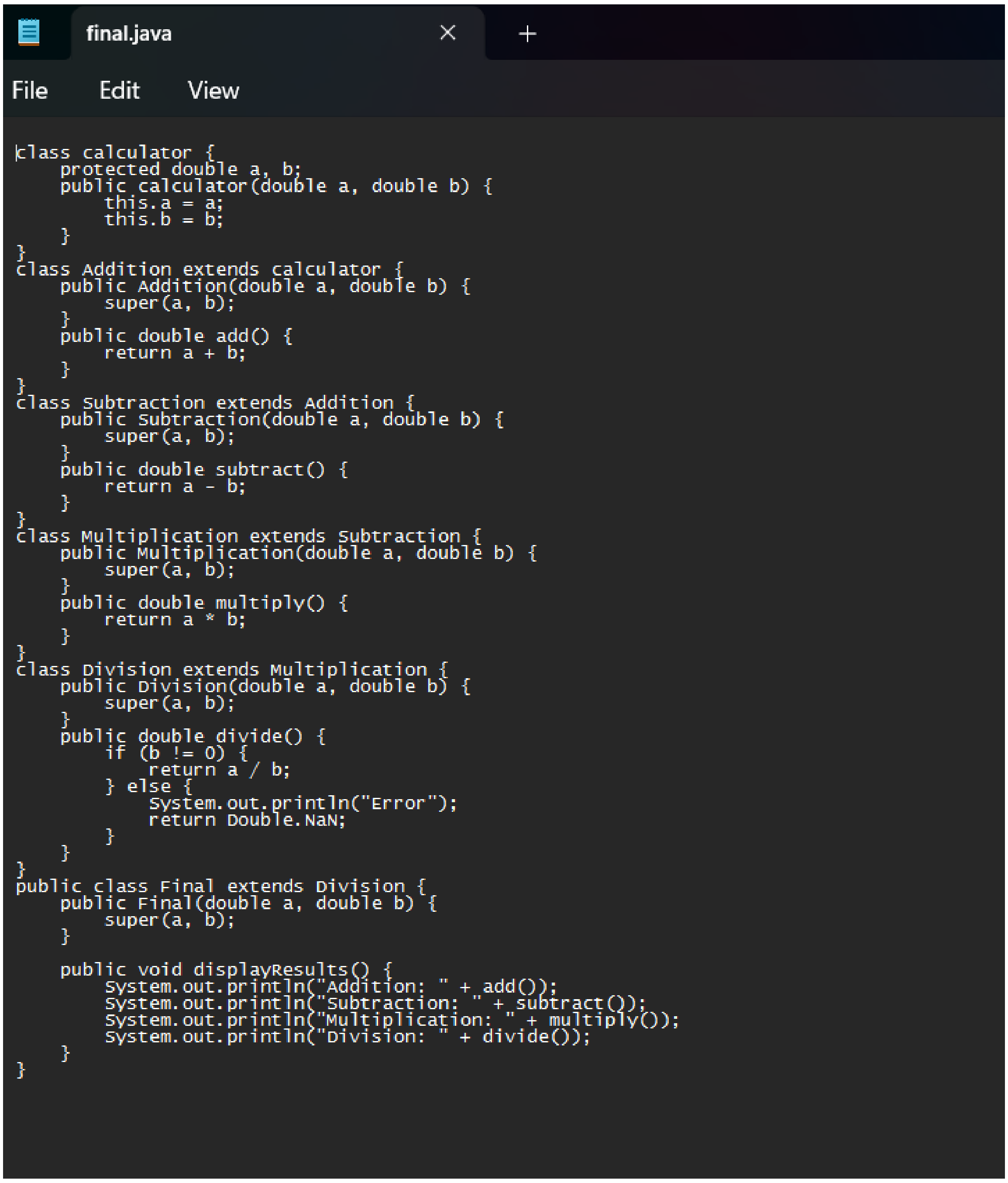


## WEEK-05

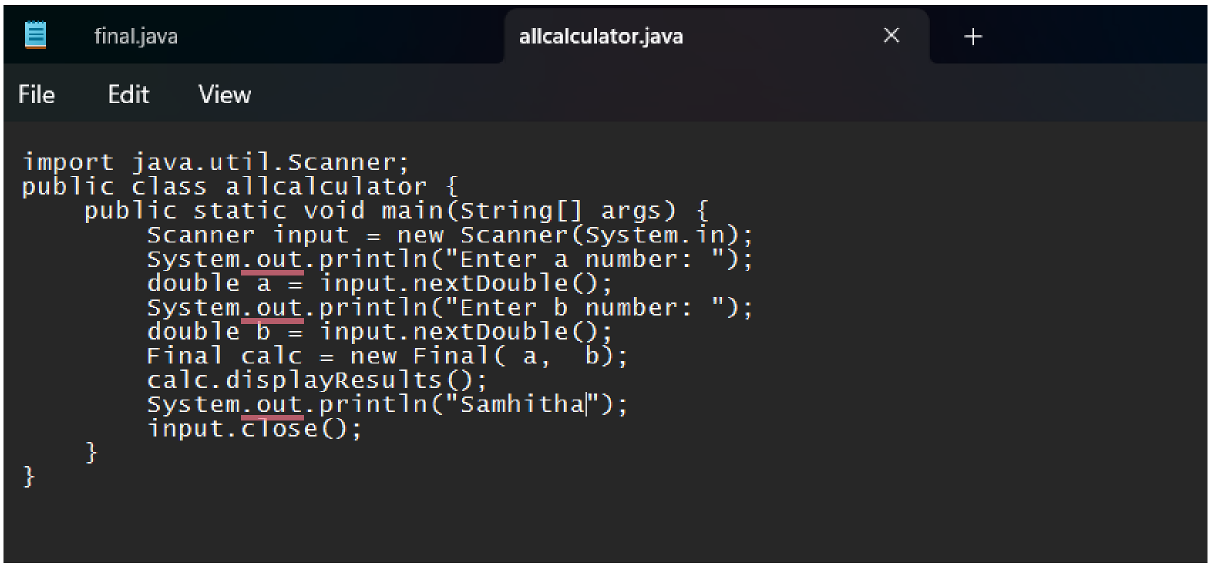
**AIM:** Create a calculator using the operations including addition, subtraction, multiplication, and division using multi-level inheritance and display the desired output.

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

**CODE:**

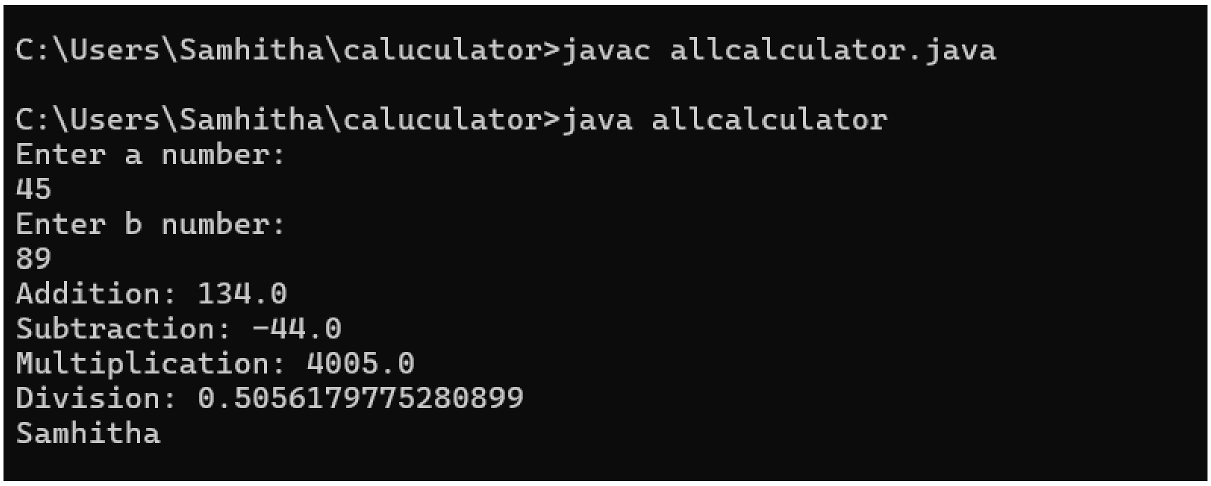


```
class calculator {
    protected double a, b;
    public calculator(double a, double b) {
        this.a = a;
        this.b = b;
    }
}
class Addition extends calculator {
    public Addition(double a, double b) {
        super(a, b);
    }
    public double add() {
        return a + b;
    }
}
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;
        }
    }
}
public 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("Samhitha");
        input.close();
    }
}
```

## OUTPUT:



```
C:\Users\Samhitha\caluculator>javac allcalculator.java

C:\Users\Samhitha\caluculator>java allcalculator
Enter a number:
45
Enter b number:
89
Addition: 134.0
Subtraction: -44.0
Multiplication: 4005.0
Division: 0.5056179775280899
Samhitha
```

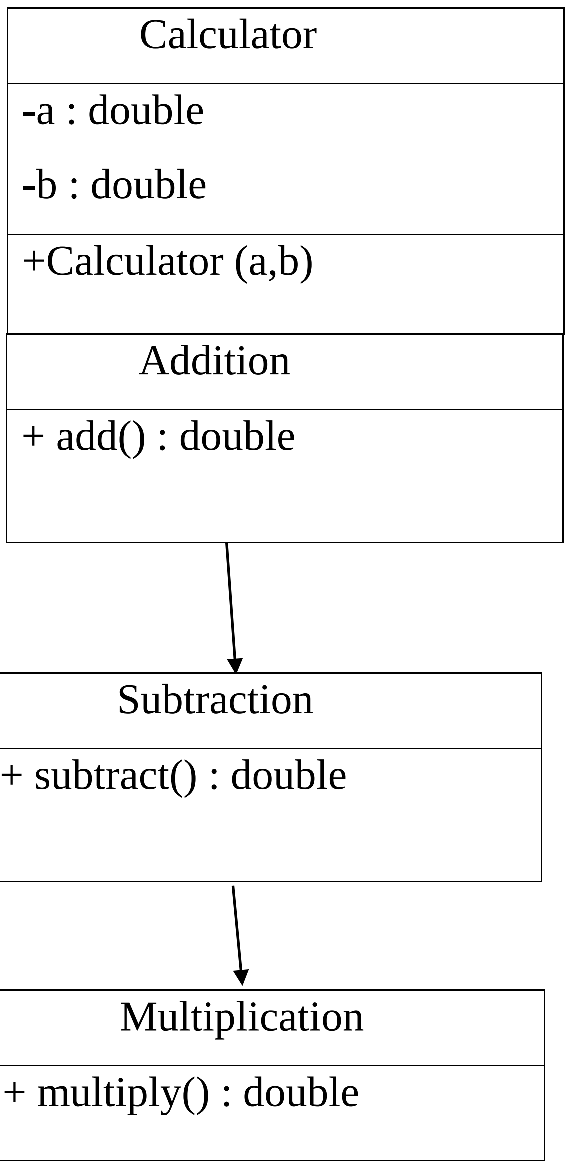
## CONCEPTS TO BE KNOWN:

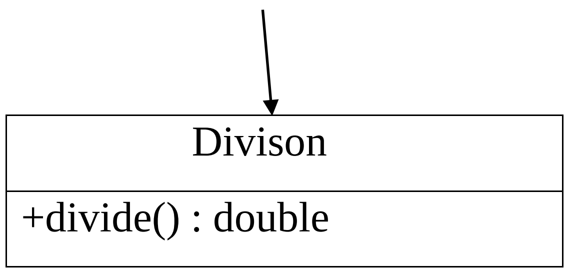
1. To get the inputs from the user we use import java.util.Scanner; this is a package.
2. Scanner class is used to get the user input.
3. In java.util.Scanner, the java.util is a package while Scanner is a class of the java.util package.
4. To import a whole package, end the sentence with an asterisk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

## ERRORS:

| **Error Message** | **Error rectification** |
|---|---|
| 1. not providing the return method correctly.<br>2. Not mentioning super to obtain the super class constructor. | 1. After declaring methods, we must provide the return method correctly.<br>2. To obtain the super class we need to mention super. |

**CLASS DIAGRAM:**





**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?
   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
3. 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.

**CONCEPTS TO BE KNOWN:**

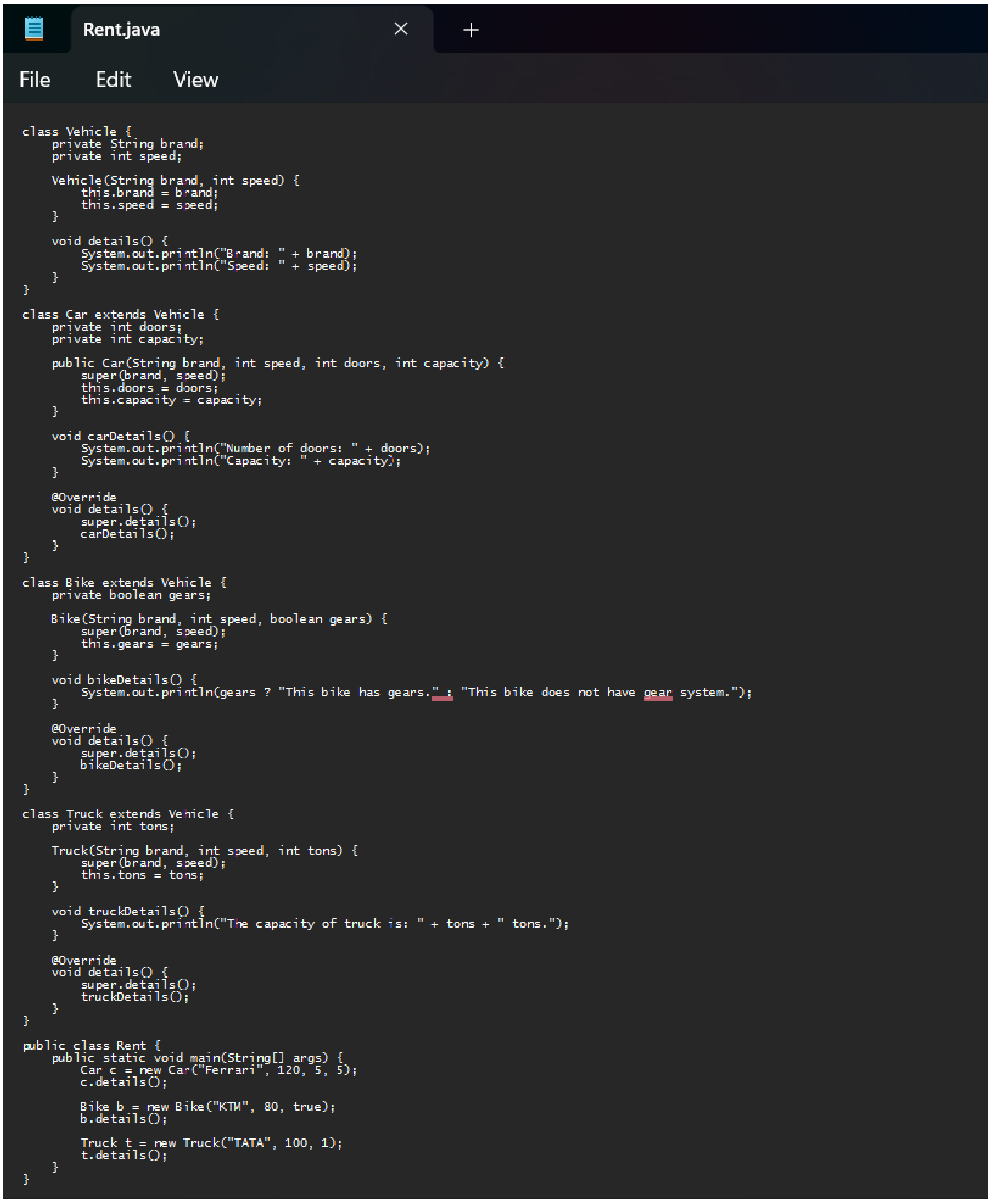
1. a constructor helps in initializing an object that doesn't exist.
2. a method performs functions on pre-constructed or already developed objects.
3. a double method can represent more decimal point numbers than float method.
4. the void keyword in java is used to specify that a method does not return any value. it is a return type that indicates the method performs a function and doesn't produce a result.

**<u>Answer for Q1:</u>**

The oops concepts used in the above program are:

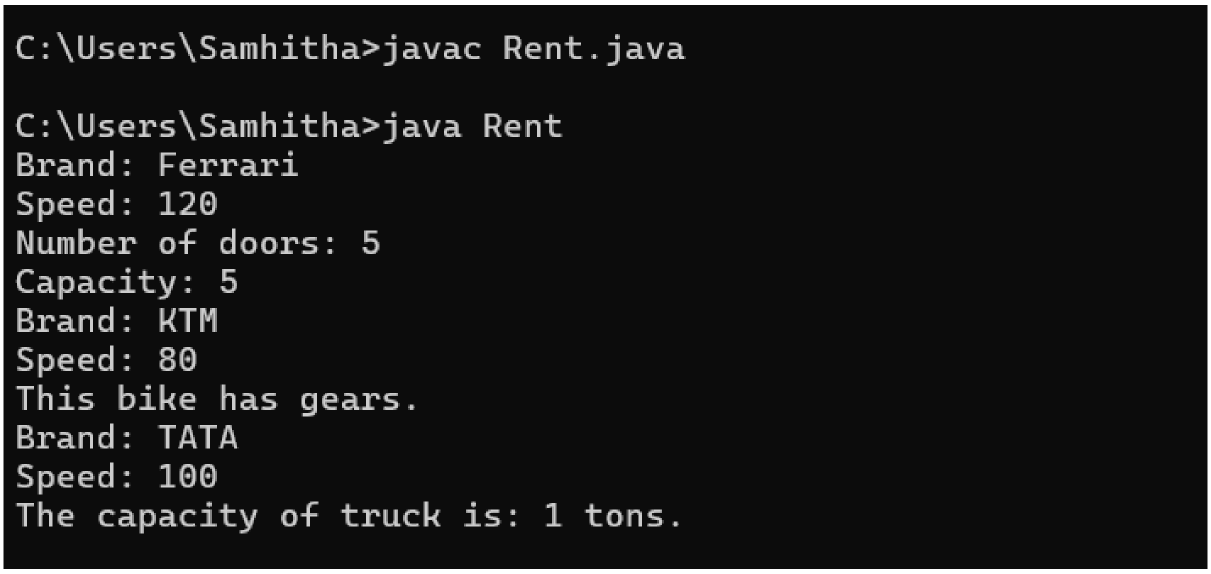
Inheritance, encapsulation, polymorphism, abstraction.

**<u>CODE:</u>**



```java
class Vehicle {
    private String brand;
    private int speed;

    Vehicle(String brand, int speed) {
        this.brand = brand;
        this.speed = speed;
    }

    void details() {
        System.out.println("Brand: " + brand);
        System.out.println("Speed: " + speed);
    }
}

class Car extends Vehicle {
    private int doors;
    private int capacity;

    public Car(String brand, int speed, int doors, int capacity) {
        super(brand, speed);
        this.doors = doors;
        this.capacity = capacity;
    }

    void carDetails() {
        System.out.println("Number of doors: " + doors);
        System.out.println("Capacity: " + capacity);
    }

    @Override
    void details() {
        super.details();
        carDetails();
    }
}

class Bike extends Vehicle {
    private boolean gears;

    Bike(String brand, int speed, boolean gears) {
        super(brand, speed);
        this.gears = gears;
    }

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

    @Override
    void details() {
        super.details();
        bikeDetails();
    }
}

class Truck extends Vehicle {
    private int tons;

    Truck(String brand, int speed, int tons) {
        super(brand, speed);
        this.tons = tons;
    }

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

    @Override
    void details() {
        super.details();
        truckDetails();
    }
}

public class Rent {
    public static void main(String[] args) {
        Car c = new Car("Ferrari", 120, 5, 5);
        c.details();

        Bike b = new Bike("KTM", 80, true);
        b.details();

        Truck t = new Truck("TATA", 100, 1);
        t.details();
    }
}
```

**OUTPUT:**

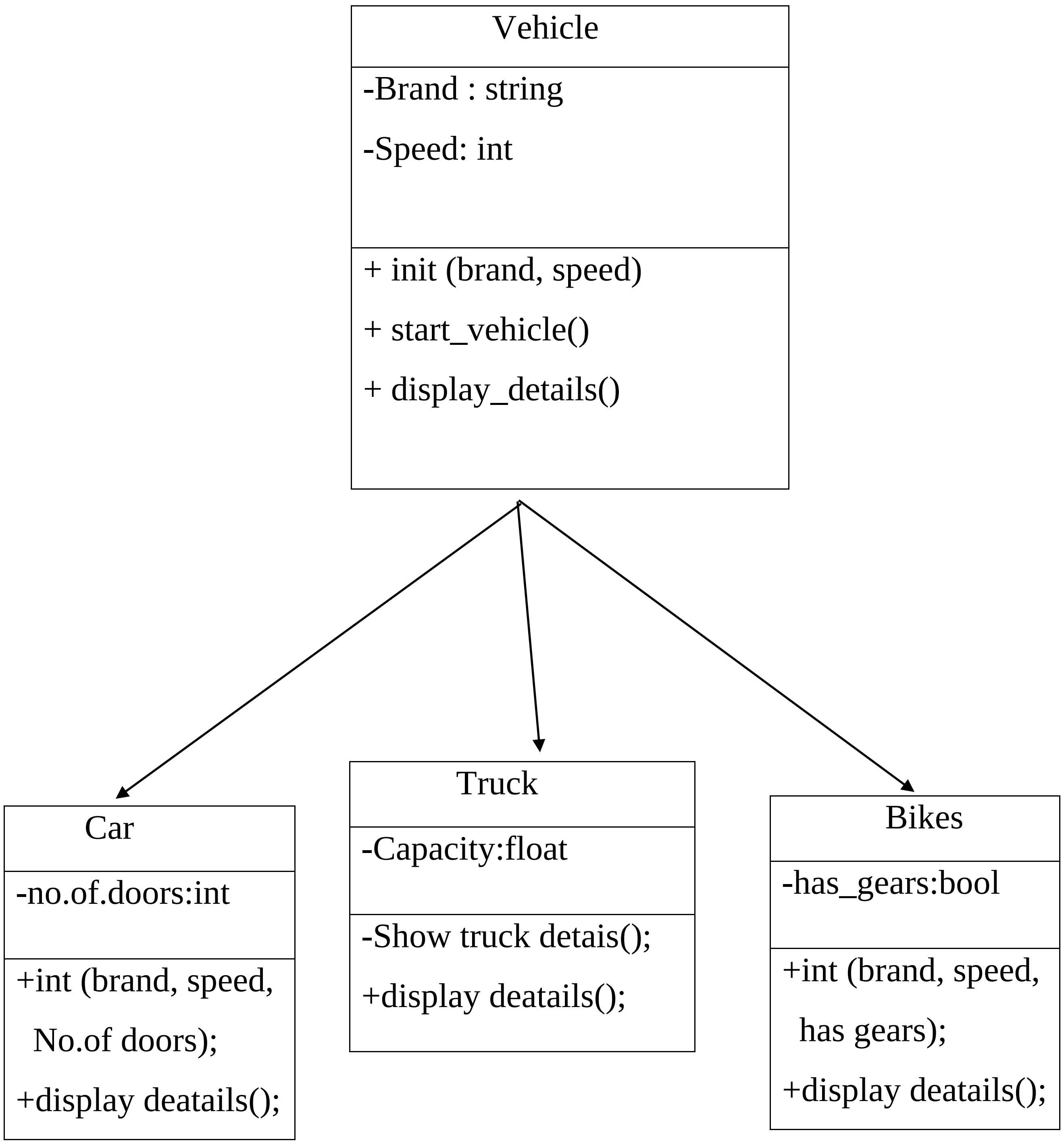


```
C:\Users\Samhitha>javac Rent.java

C:\Users\Samhitha>java Rent
Brand: Ferrari
Speed: 120
Number of doors: 5
Capacity: 5
Brand: KTM
Speed: 80
This bike has gears.
Brand: TATA
Speed: 100
The capacity of truck is: 1 tons.
```

**ERROR TABLE:**

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

**CLASS DIAGRAM:**

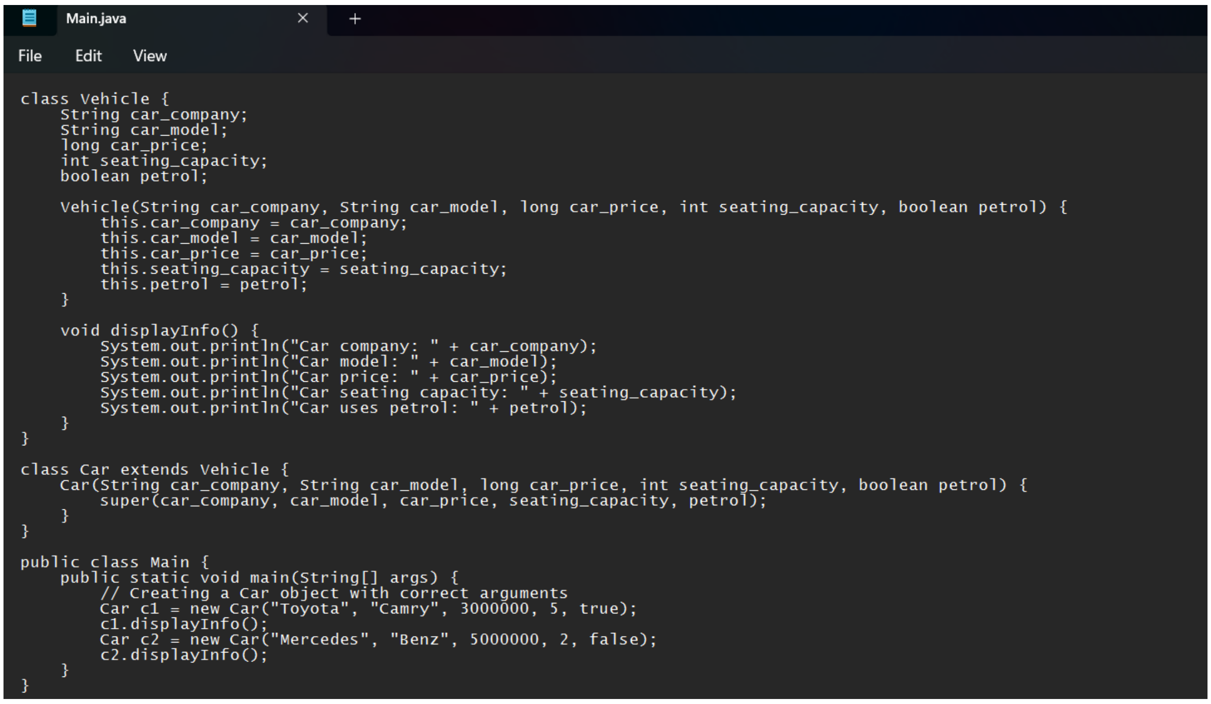


**WEEK-06**

**PROGRAM-1:**

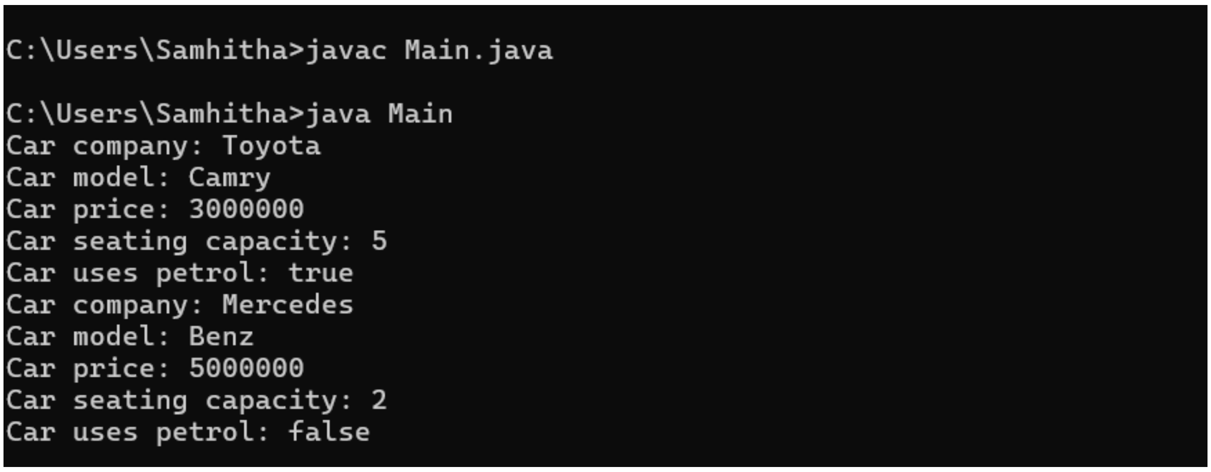
**AIM:** Write a java program to create a vehicle class with a method displayinfo(). Override this method in the car subclass to provide specific information about car (car company, seating capacity, petrol or not).

## CODE:



```java
class Vehicle {
    String car_company;
    String car_model;
    long car_price;
    int seating_capacity;
    boolean petrol;

    Vehicle(String car_company, String car_model, long car_price, int seating_capacity, boolean petrol) {
        this.car_company = car_company;
        this.car_model = car_model;
        this.car_price = car_price;
        this.seating_capacity = seating_capacity;
        this.petrol = petrol;
    }

    void displayInfo() {
        System.out.println("Car company: " + car_company);
        System.out.println("Car model: " + car_model);
        System.out.println("Car price: " + car_price);
        System.out.println("Car seating capacity: " + seating_capacity);
        System.out.println("Car uses petrol: " + petrol);
    }
}

class Car extends Vehicle {
    Car(String car_company, String car_model, long car_price, int seating_capacity, boolean petrol) {
        super(car_company, car_model, car_price, seating_capacity, petrol);
    }
}

public class Main {
    public static void main(String[] args) {
        // Creating a Car object with correct arguments
        Car c1 = new Car("Toyota", "Camry", 3000000, 5, true);
        c1.displayInfo();
        Car c2 = new Car("Mercedes", "Benz", 5000000, 2, false);
        c2.displayInfo();
    }
}
```

## OUTPUT:



```
C:\Users\Samhitha>javac Main.java

C:\Users\Samhitha>java Main
Car company: Toyota
Car model: Camry
Car price: 3000000
Car seating capacity: 5
Car uses petrol: true
Car company: Mercedes
Car model: Benz
Car price: 5000000
Car seating capacity: 2
Car uses petrol: false
```

## ERRORS:

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

**IMPORTANT POINTS:**

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

**2.Constructor Chaining:**The Car class calls the parent constructor using super(car_model, color, fuel_type); to initialize inherited attributes.

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

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

**CLASS DIAGRAM:**

| **Vehicle** |
|---|
| - Brand: String<br>- Speed: int |
| + vehicle(brand: string<br>Speed: int)<br>+start vehicle(): void<br>+displaydetails():void |

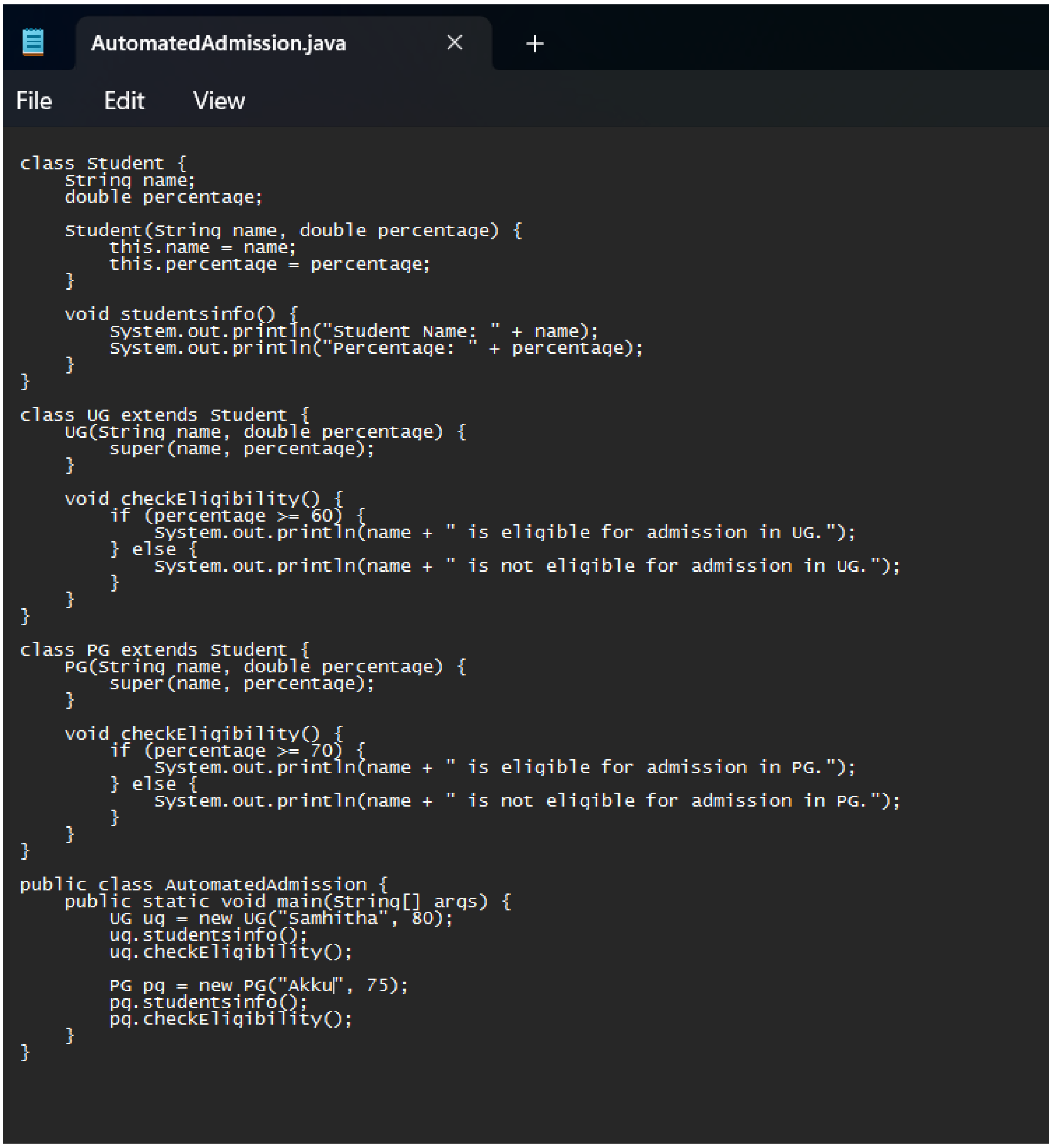
## PROGRAM-2:

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

1. UG admission recquire a minimum of 60%.

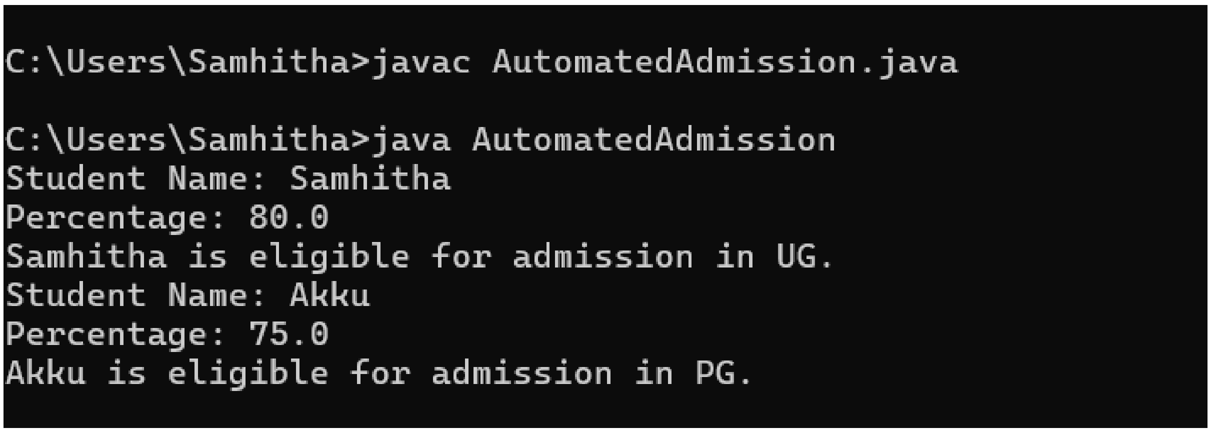
2. PG admission recquire a minimum of 70%.

## CODE:



```java
class Student {
    String name;
    double percentage;

    Student(String name, double percentage) {
        this.name = name;
        this.percentage = percentage;
    }

    void studentsinfo() {
        System.out.println("Student Name: " + name);
        System.out.println("Percentage: " + percentage);
    }
}

class UG extends Student {
    UG(String name, double percentage) {
        super(name, percentage);
    }

    void checkEligibility() {
        if (percentage >= 60) {
            System.out.println(name + " is eligible for admission in UG.");
        } else {
            System.out.println(name + " is not eligible for admission in UG.");
        }
    }
}

class PG extends Student {
    PG(String name, double percentage) {
        super(name, percentage);
    }

    void checkEligibility() {
        if (percentage >= 70) {
            System.out.println(name + " is eligible for admission in PG.");
        } else {
            System.out.println(name + " is not eligible for admission in PG.");
        }
    }
}

public class AutomatedAdmission {
    public static void main(String[] args) {
        UG ug = new UG("Samhitha", 80);
        ug.studentsinfo();
        ug.checkEligibility();

        PG pg = new PG("Akku", 75);
        pg.studentsinfo();
        pg.checkEligibility();
    }
}
```

## OUTPUT:



```
C:\Users\Samhitha>javac AutomatedAdmission.java

C:\Users\Samhitha>java AutomatedAdmission
Student Name: Samhitha
Percentage: 80.0
Samhitha is eligible for admission in UG.
Student Name: Akku
Percentage: 75.0
Akku is eligible for admission in PG.
```

## ERRORS:

| Code error | Code rectification |
|---|---|
| **1.Scanner nextLine() issue after nextDouble():** After scanner.nextDouble(), the newline character remains in the buffer, causing nextLine() to be skipped.<br><br>**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.<br><br>**2.**Use program.toUpperCase() to ensure case-insensitive comparison. |

## IMPORTANT POINTS:

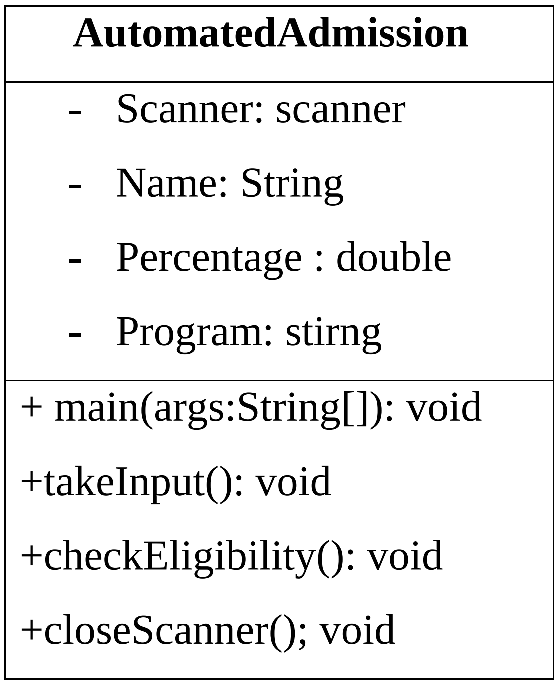
**1.User Input Handling:** Uses Scanner to take user input for name, percentage, and program type.

**2.Decision Making with Conditions:** Uses if-else statements to check eligibility criteria.

**3.String Handling:** Converts program input to uppercase (toUpperCase()) to handle case variations.

**4.Closing Scanner:** Properly closes scanner using scanner.close(); to prevent resource leaks.

## CLASS DIAGRAM:

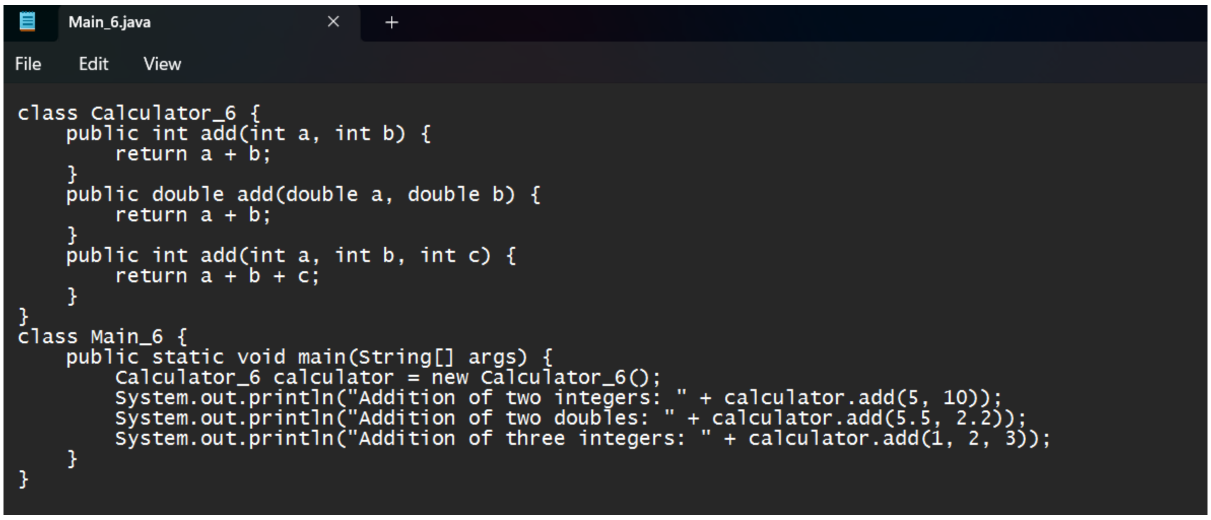


**<u>PROGRAM-3:</u>**

**<u>AIM</u>:** Create a calculator class with overloaded methods to perform addition of:

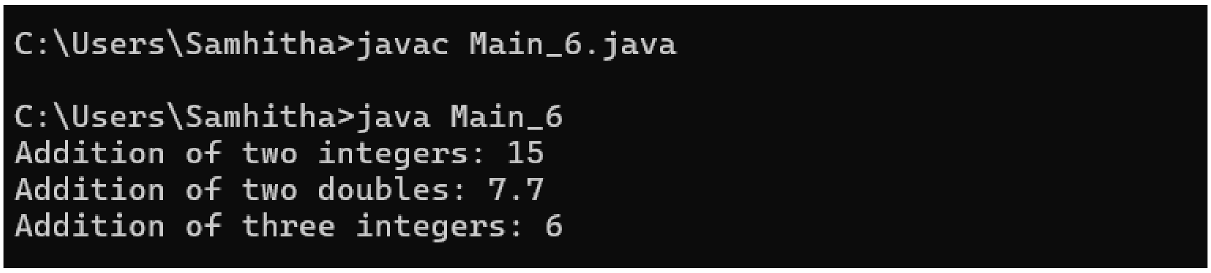
1. Add two integers
2. Add two doubles
3. Add three integers

**<u>CODE:</u>**



```java
class Calculator_6 {
    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;
    }
}
class Main_6 {
    public static void main(String[] args) {
        Calculator_6 calculator = new Calculator_6();
        System.out.println("Addition of two integers: " + calculator.add(5, 10));
        System.out.println("Addition of two doubles: " + calculator.add(5.5, 2.2));
        System.out.println("Addition of three integers: " + calculator.add(1, 2, 3));
    }
}
```

**OUTPUT:**



```
C:\Users\Samhitha>javac Main_6.java

C:\Users\Samhitha>java Main_6
Addition of two integers: 15
Addition of two doubles: 7.7
Addition of three integers: 6
```

**ERRORS:**

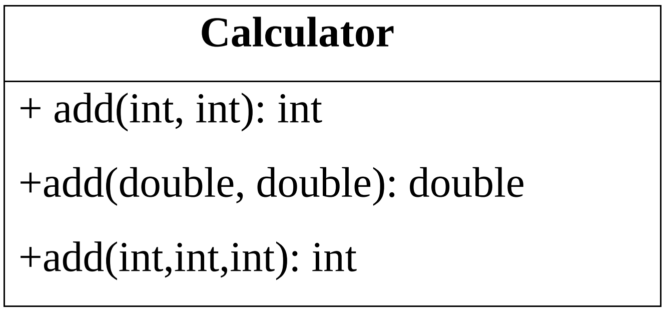
| Code error | Code rectification |
|---|---|
| 1.Method parameters missing spaces. E.g.,"inta, intb"should be "int a, int b" | 1. Add proper spacing between parameters: (int a, int b) |
| 2.Inconsistent indentation in method bodies | 2. Fix indentation: Consistent 4 space o indentation. |

**IMPORTANT POINTS:**

**1.Method Overloading:** The add method is overloaded with different parameter types and counts, demonstrating compile-time polymorphism.

**2.Automatic Method Selection:** Java selects the appropriate add method based on the argument types during compilation.

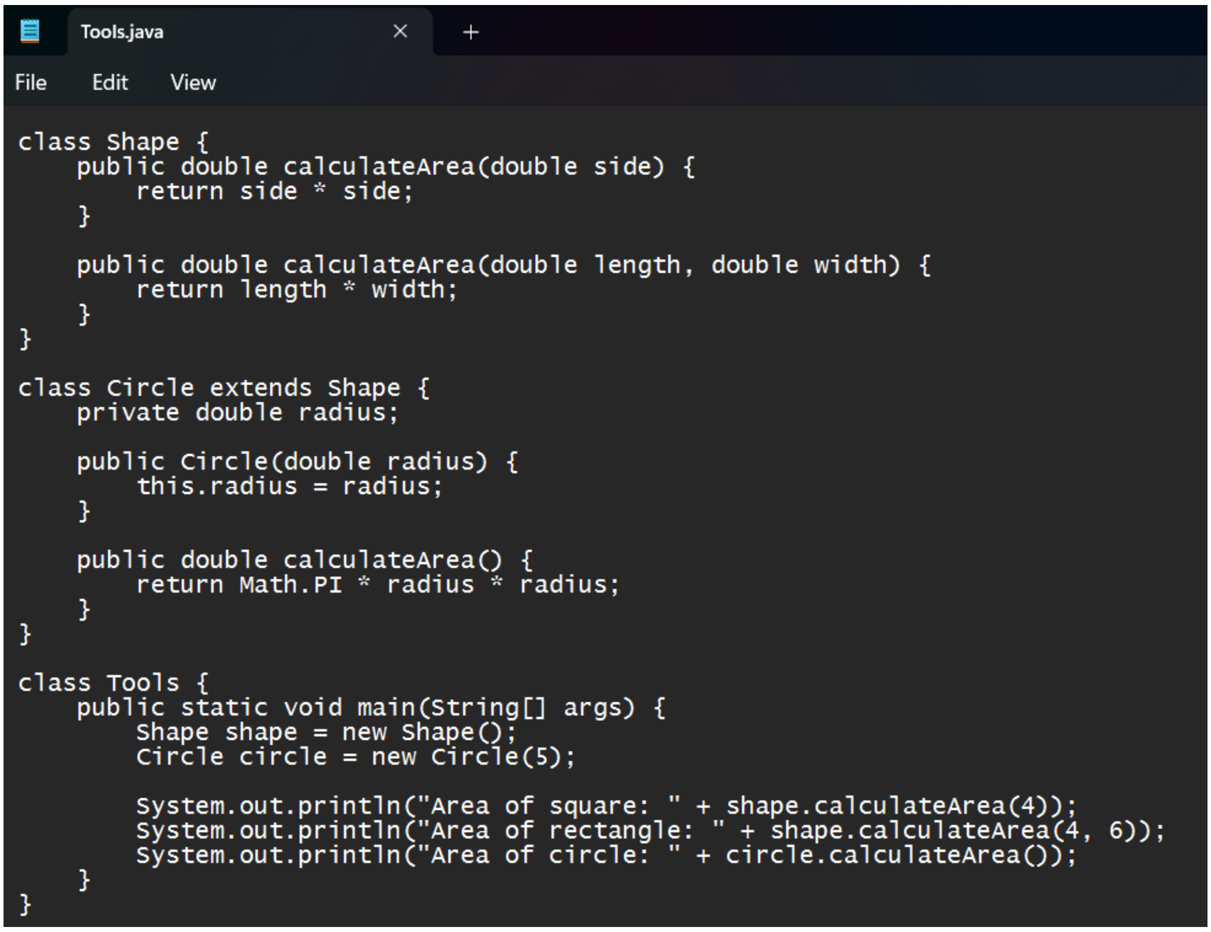
**<u>CLASS DIAGRAM:</u>**



**<u>PROGRAM-4:</u>**

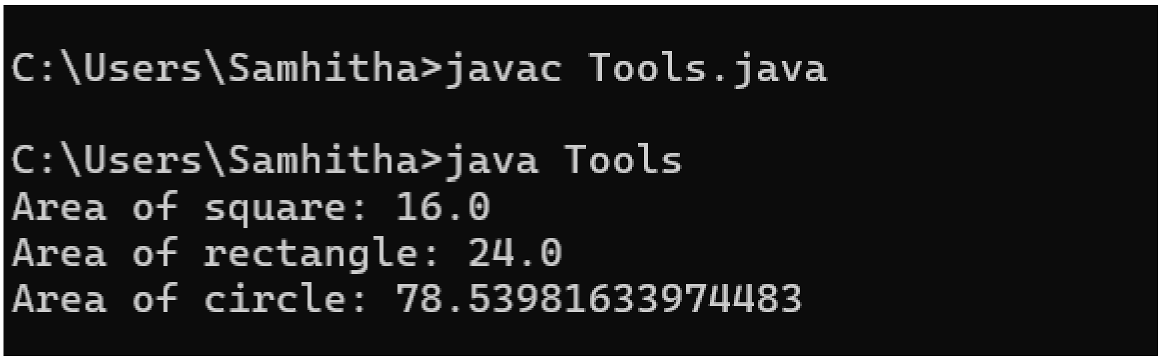
**<u>AIM</u>:** Create a shape class with a method to calculate area i.e., overloaded for different shapes eg: Squares, Recatangle. Then create a subclass circle that overrides the calculateArea() method for a circle.

**<u>CODE:</u>**



```
class Shape {
    public double calculateArea(double side) {
        return side * side;
    }

    public double calculateArea(double length, double width) {
        return length * width;
    }
}

class Circle extends Shape {
    private double radius;

    public Circle(double radius) {
        this.radius = radius;
    }

    public double calculateArea() {
        return Math.PI * radius * radius;
    }
}

class Tools {
    public static void main(String[] args) {
        Shape shape = new Shape();
        Circle circle = new Circle(5);

        System.out.println("Area of square: " + shape.calculateArea(4));
        System.out.println("Area of rectangle: " + shape.calculateArea(4, 6));
        System.out.println("Area of circle: " + circle.calculateArea());
    }
}
```

**OUTPUT:**



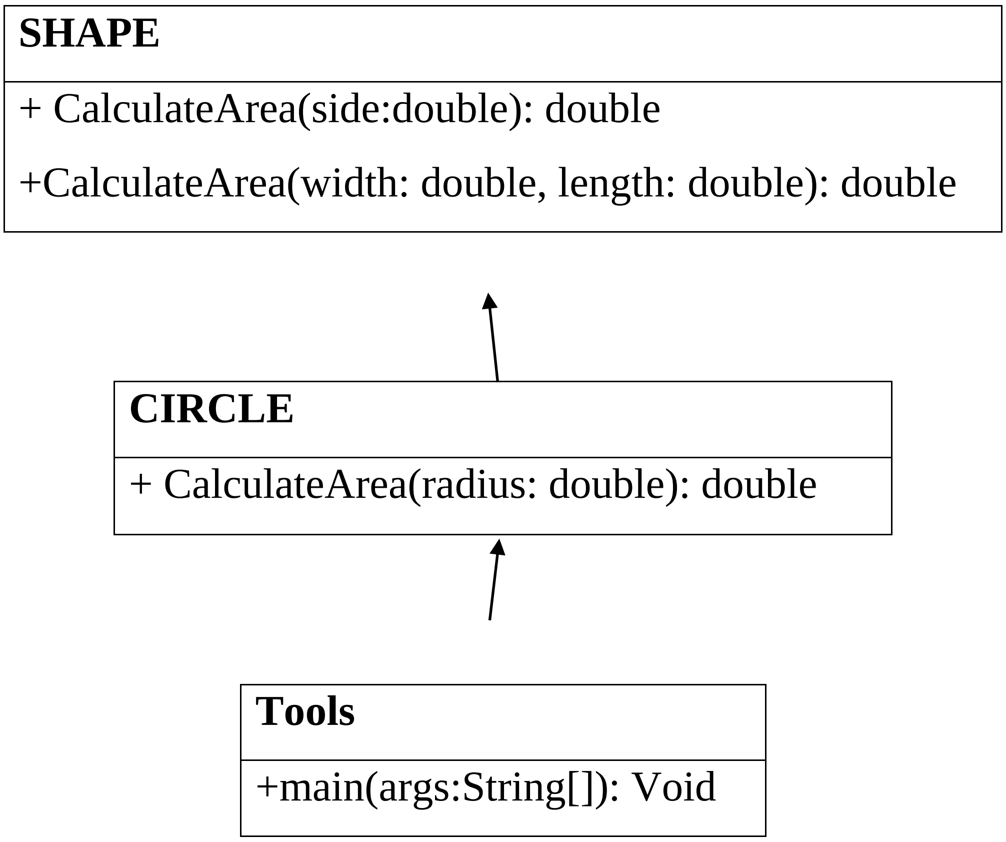
```
C:\Users\Samhitha>javac Tools.java

C:\Users\Samhitha>java Tools
Area of square: 16.0
Area of rectangle: 24.0
Area of circle: 78.53981633974483
```

**ERRORS:**

| Code error | Code rectification |
|---|---|

| | |
|---|---|
| 1. Method calls in main are missing an object reference (e.g., calculateArea(4) instead of s.calculateArea(4)).<br>2. Circle class method does not override theparent class method properly. | 1. Use s.calculateArea(4) and c.calculateArea(2) to call the method correctly.<br>2. Ensure @Override is used, and the method signature should match correctly. |

**<u>CLASS DIAGRAM:</u>**


SHAPE
+ CalculateArea(side:double): double
+CalculateArea(width: double, length: double): double
CIRCLE
+ CalculateArea(radius: double): double
Tools
+main(args:String[]): Void

**<u>IMPORTANT POINTS:</u>**

**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.

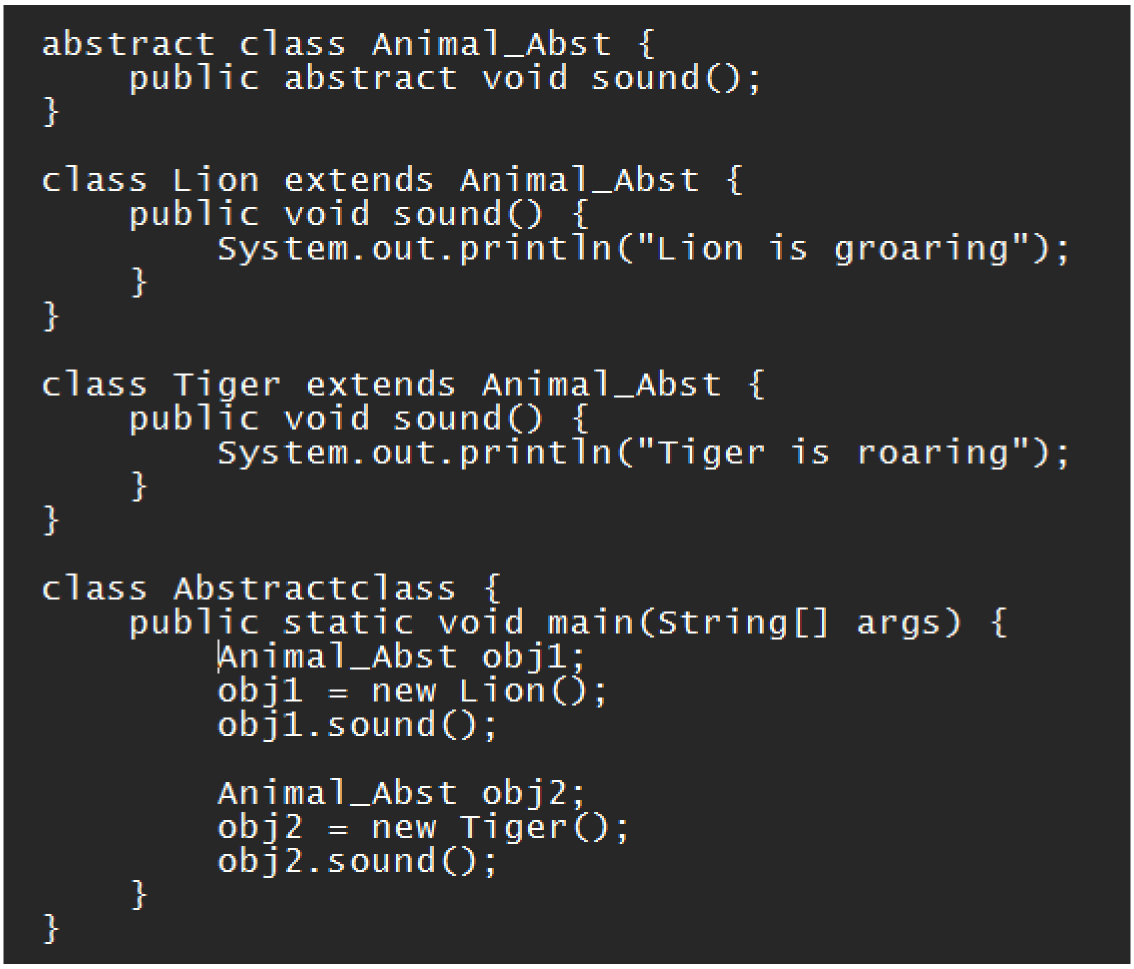
**5.Proper Object Reference**: Methods should be called using an object (s.calculateArea(4), c.calculateArea(2)).

## WEEK-07

## PROGRAM-1:

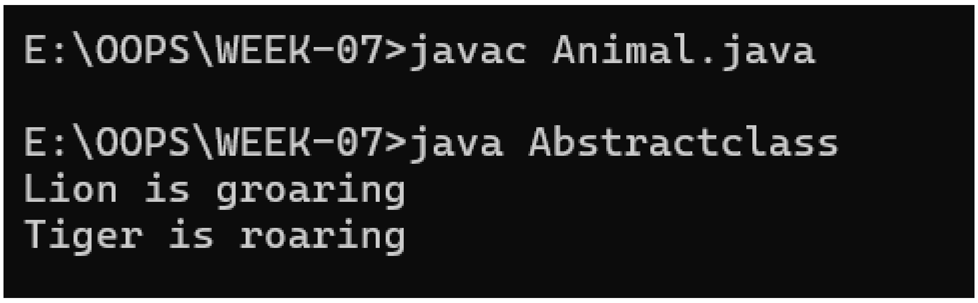
**AIM :** create a Java program to create an abstrad cass animal with an abstract method called sound ().Create a subclass Lion and tiger that extend the Animal class and implement the sound () method to make a specific sound for each animal.

## CODE:



```java
abstract class Animal_Abst {
    public abstract void sound();
}

class Lion extends Animal_Abst {
    public void sound() {
        System.out.println("Lion is groaring");
    }
}

class Tiger extends Animal_Abst {
    public void sound() {
        System.out.println("Tiger is roaring");
    }
}

class Abstractclass {
    public static void main(String[] args) {
        Animal_Abst obj1;
        obj1 = new Lion();
        obj1.sound();

        Animal_Abst obj2;
        obj2 = new Tiger();
        obj2.sound();
    }
}
```

## OUTPUT:



```
E:\OOPS\WEEK-07>javac Animal.java

E:\OOPS\WEEK-07>java Abstractclass
Lion is groaring
Tiger is roaring
```

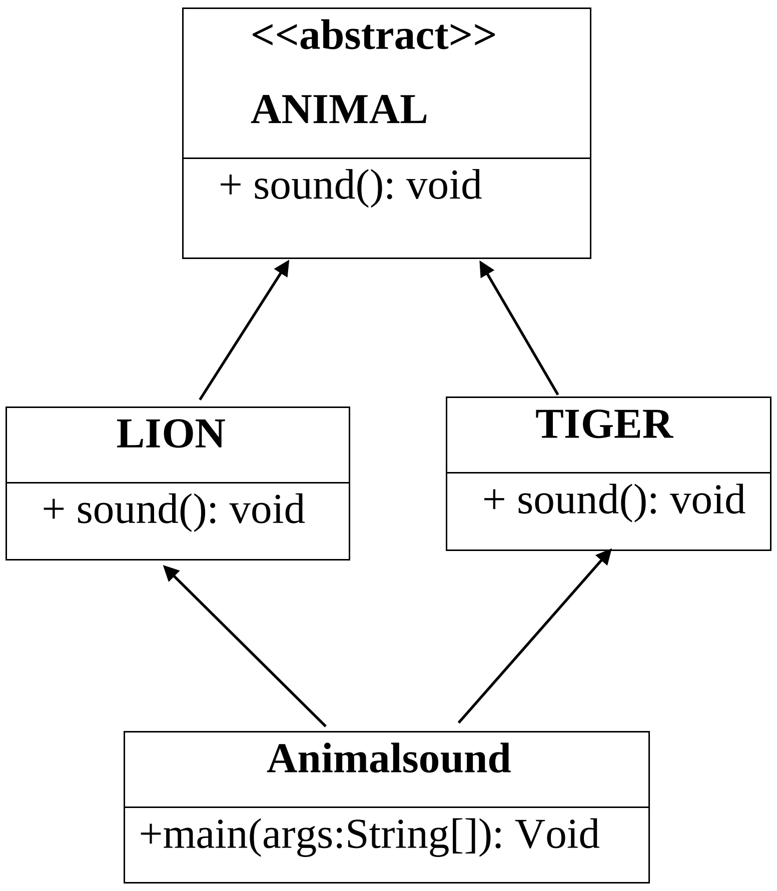
**IMPORTANT POINTS:**

1. abstract class Animal: Can't be directly used to create objects.
2. abstract void sound(): Forces subclasses to implement this method.
3. Lion and Tiger both override sound().
4. Animal a = new Lion(); uses runtime polymorphism.

**ERRORS:**

| Code error | Code rectification |
|---|---|
| 1. Forgetting to use abstract keyword for the sound() method.<br><br>2 Not overriding the sound() method in subclasses. | 1. Rectified as abstract void sound();<br><br>2. Added void sound() { ... } in each subclass. |

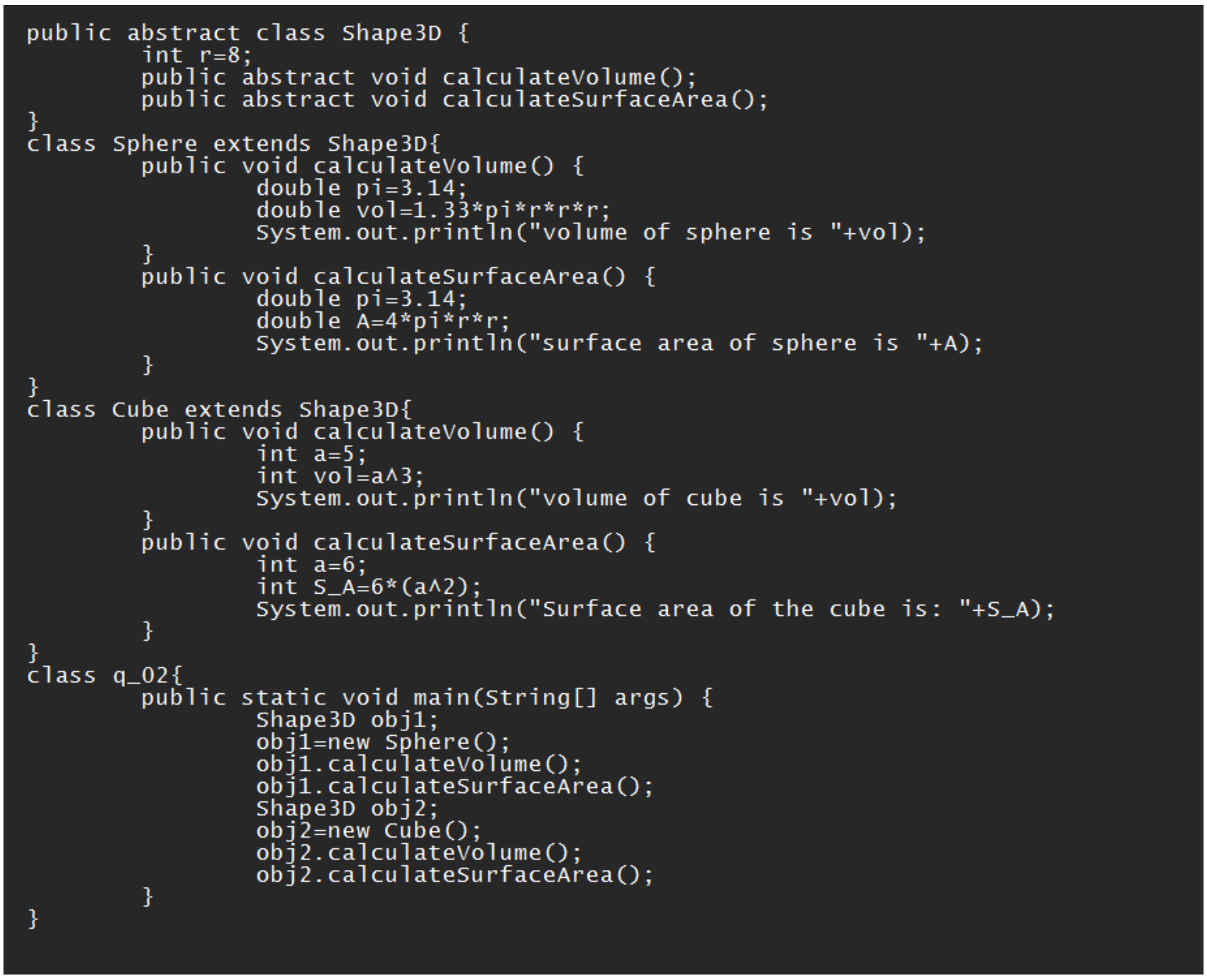
## **CLASS DIAGRAM:**



## <u>PROGRAM-2:</u>

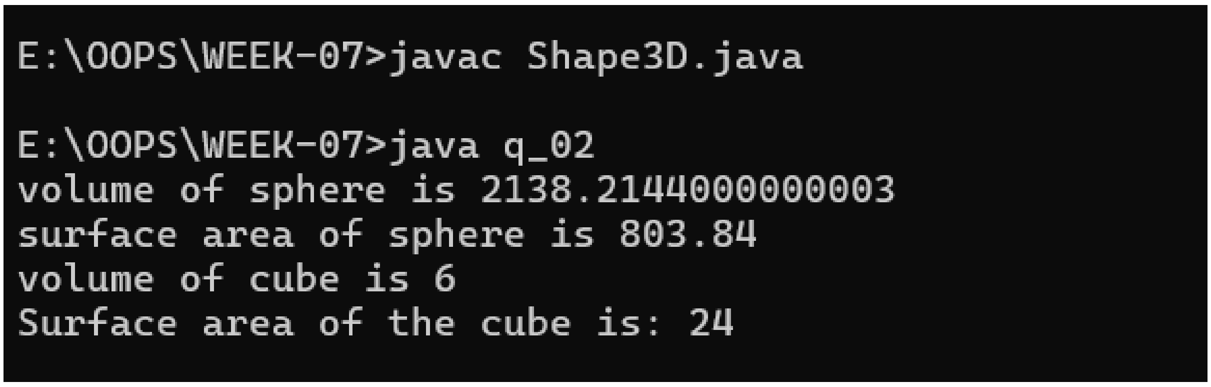
**<u>AIM :</u>** Write a Java program to create an abstract class shape 3D with abstract methods calculate volume ()and calculate surface Area ()create  subclasses Sphere and cube that extend the Spape 3D clas and implement the respective methods to calculate ine volume and surface area of each  shape.

## <u>CODE:</u>



```java
public abstract class Shape3D {
        int r=8;
        public abstract void calculateVolume();
        public abstract void calculateSurfaceArea();
}
class Sphere extends Shape3D{
        public void calculateVolume() {
                double pi=3.14;
                double vol=1.33*pi*r*r*r;
                System.out.println("volume of sphere is "+vol);
        }
        public void calculateSurfaceArea() {
                double pi=3.14;
                double A=4*pi*r*r;
                System.out.println("surface area of sphere is "+A);
        }
}
class Cube extends Shape3D{
        public void calculateVolume() {
                int a=5;
                int vol=a^3;
                System.out.println("volume of cube is "+vol);
        }
        public void calculateSurfaceArea() {
                int a=6;
                int S_A=6*(a^2);
                System.out.println("Surface area of the cube is: "+S_A);
        }
}
class q_02{
        public static void main(String[] args) {
                Shape3D obj1;
                obj1=new Sphere();
                obj1.calculateVolume();
                obj1.calculateSurfaceArea();
                Shape3D obj2;
                obj2=new Cube();
                obj2.calculateVolume();
                obj2.calculateSurfaceArea();
        }
}
```

## <u>OUTPUT:</u>



```
E:\OOPS\WEEK-07>javac Shape3D.java

E:\OOPS\WEEK-07>java q_02
volume of sphere is 2138.2144000000003
surface area of sphere is 803.84
volume of cube is 6
Surface area of the cube is: 24
```

**IMPORTSNT POINTS:**

**1.Abstract Class Used**:Shape3D is an abstract class with abstract methods – it can't be directly used to create objects.

**2.Method Overriding**:Sphere and Cube both override calculateVolume() and calculateSurface() with their own formulas.

**3.Return Type: double** :Volume and surface area can be decimal, so methods return double, not int.

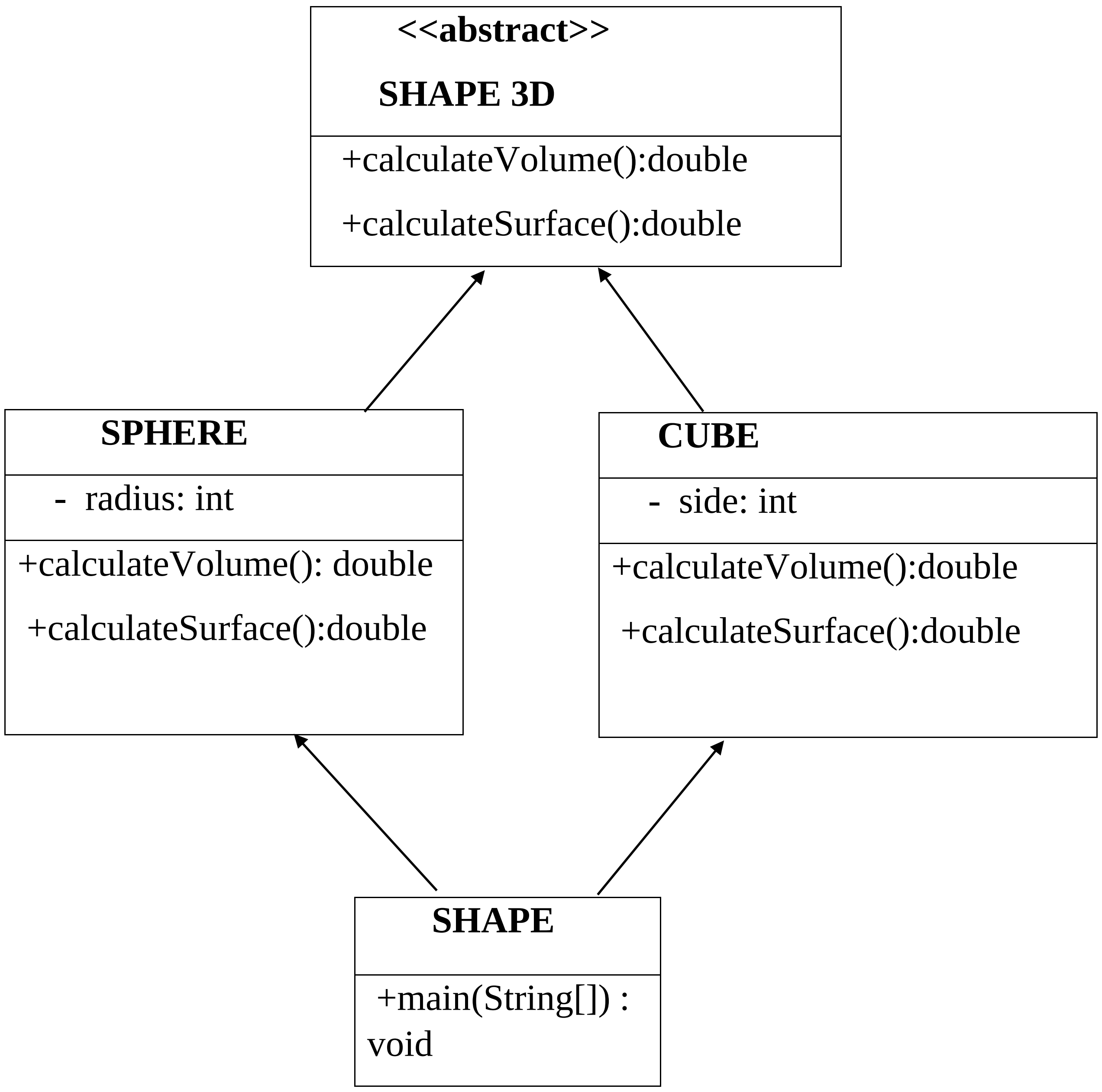
**4.Use of Math.PI and Math.pow()**: More accurate than hardcoding 3.14 and r*r*r. It's a good practice for real calculations.

**ERRORS:**

| **Code error** | **Code rectification** |
|---|---|
| 1. int used instead of double for Volume surface<br><br>2. (4 / 3) used instead of (4.0 / 3.0) | 4. Changed return types of calculateVolume() and calculateSurface() to double<br>5. Used floating-point division to avoid integer division loss.<br>6. 3.Used Math.PI for more accurate calculations. |

| | |
|---|---|
| 3. 3.14 used as approximation for π | |

**CLASS DIAGRAM:**



<<abstract>>
SHAPE 3D
+calculateVolume():double
+calculateSurface():double
SPHERE
- radius: int
+calculateVolume(): double
+calculateSurface():double
CUBE
- side: int
+calculateVolume():double
+calculateSurface():double
SHAPE
+main(String[]) : void

**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)
- A concrete method to display the pattern title

Create two subclasses:

1.StarPattern: Prints a right-angled triangle of stars (*)

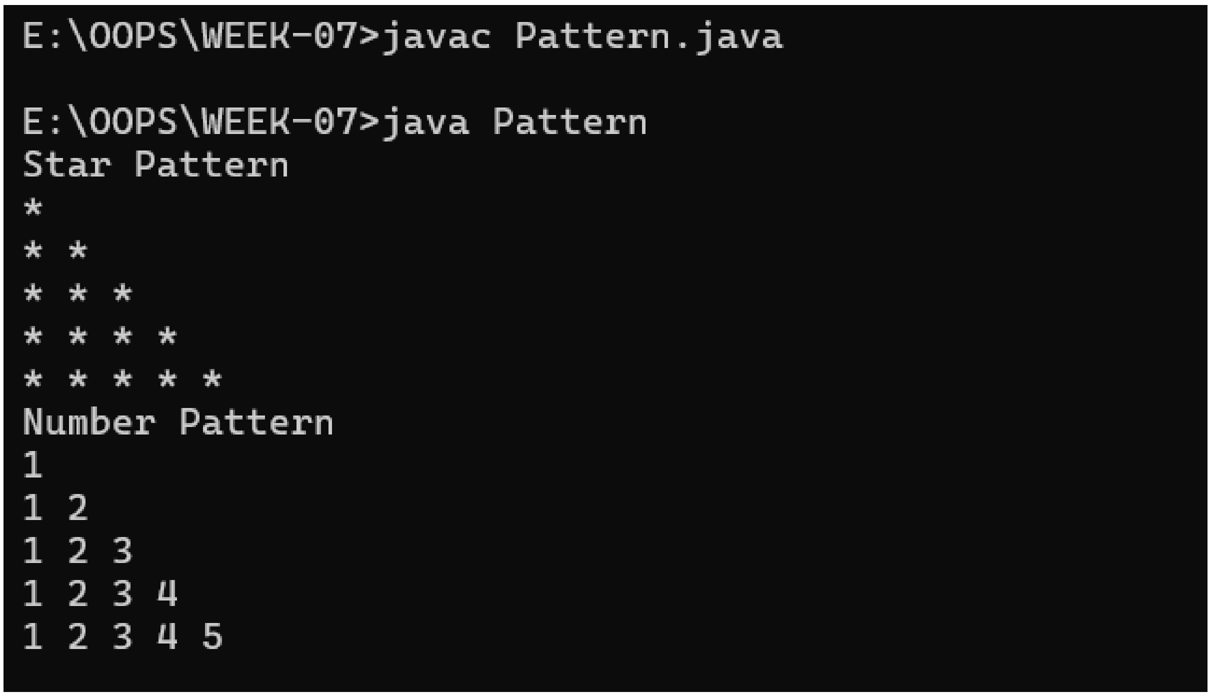
2.NumberPattern: Prints a right-angled triangle of increasing numbers

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

**CODE:**

```java
abstract class PatternPrinter {
    abstract void printPattern(int n);

    void displayTitle(String title) {
        System.out.println( title );
    }
}

class StarPattern extends PatternPrinter {
    void printPattern(int n) {
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= i; j++) {
                System.out.print("* ");
            }
            System.out.println();
        }
    }
}

class NumberPattern extends PatternPrinter {
    void printPattern(int n) {
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= i; j++) {
                System.out.print(j + " ");
            }
            System.out.println();
        }
    }
}

public class Pattern {
    public static void main(String[] args) {
        int rows = 5;

        PatternPrinter star = new StarPattern();
        star.displayTitle("Star Pattern");
        star.printPattern(rows);

        PatternPrinter number = new NumberPattern();
        number.displayTitle("Number Pattern");
        number.printPattern(rows);
    }
}
```

**OUTPUT:**



```
E:\OOPS\WEEK-07>javac Pattern.java

E:\OOPS\WEEK-07>java Pattern
Star Pattern
*
* *
* * *
* * * *
* * * * *
Number Pattern
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

**IMPORTANT POINTS:**

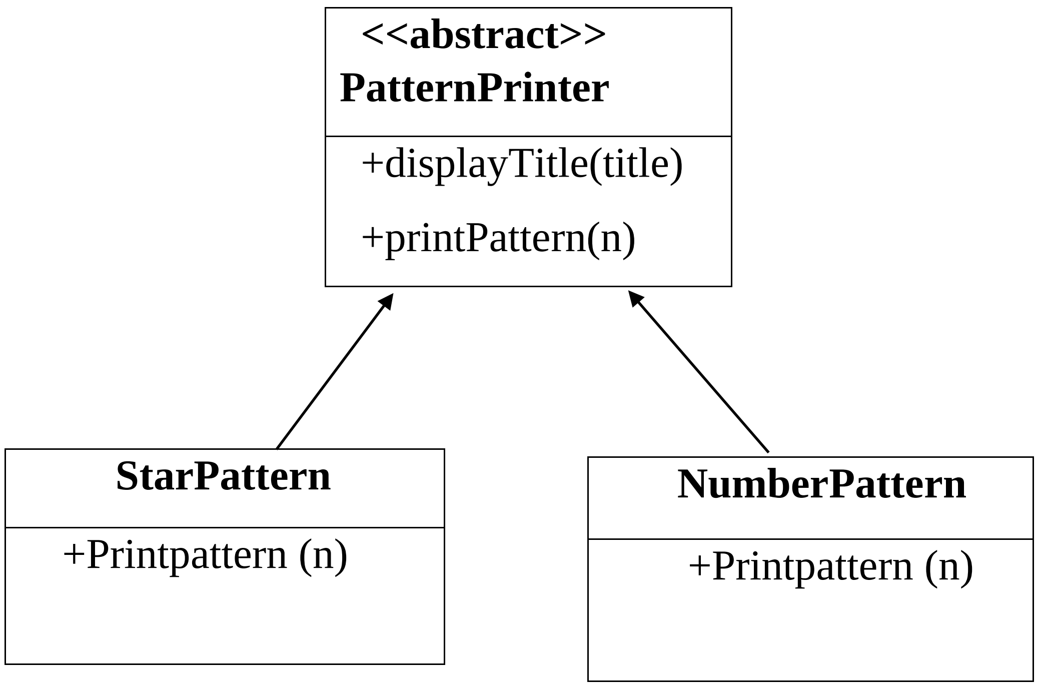
1. Abstract class PatternPrinter cannot be instantiated directly.
2. Abstract method printPattern(int n) must be implemented in all subclasses.
3. Concrete method displayTitle(String title) is reusable by both subclasses.
4. Use of inheritance: StarPattern and NumberPattern extend the abstract class.
5. In main(), objects are created from subclasses, not the abstract class.

**ERRORS:**

| Code error | 1. Code rectification |
|---|---|

| | |
|---|---|
| 1.Wrong loop logic ( printing * without loop).<br><br>2.displayTitle method not used before pattern printing<br><br>3.Forgot to implement printPattern(int n) in subclass<br>1. | 1.Use nested loops: outer loop for rows, inner loop for printing symbols or numbers.<br><br>2.Call displayTitle() before printing the pattern for proper formatting<br><br>3.Implemented the method in both subclasses |

**CLASS DIAGRAM:**

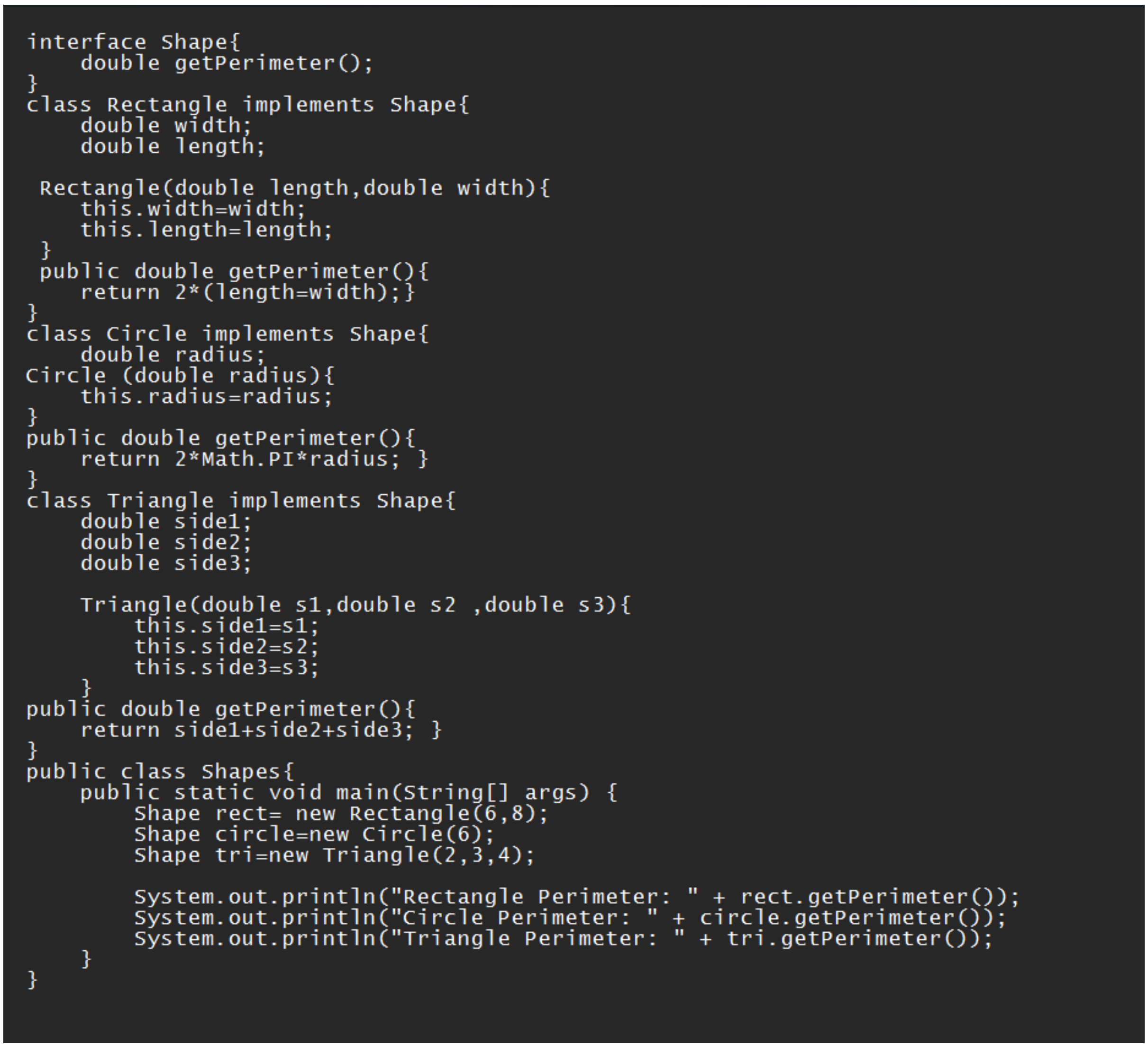


## WEEK-08

## PROGRAM-1:

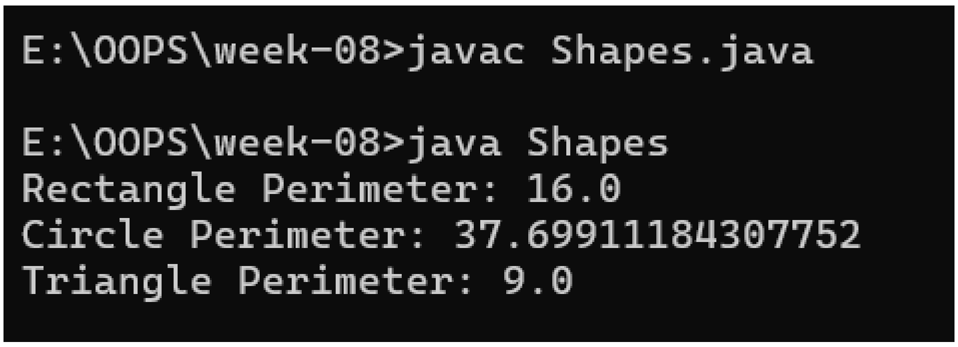
**AIM :** Write a java program shape with get perimeter() method and create three class rectangle circle triangle that implement the shape interface and implement the get perimeter for all the 3 classes.

**CODE:**



```
interface Shape{
    double getPerimeter();
}
class Rectangle implements Shape{
    double width;
    double length;

 Rectangle(double length,double width){
    this.width=width;
    this.length=length;
 }
 public double getPerimeter(){
    return 2*(length=width);}
}
class Circle implements Shape{
    double radius;
Circle (double radius){
    this.radius=radius;
}
public double getPerimeter(){
    return 2*Math.PI*radius; }
}
class Triangle implements Shape{
    double side1;
    double side2;
    double side3;

    Triangle(double s1,double s2 ,double s3){
        this.side1=s1;
        this.side2=s2;
        this.side3=s3;
    }
public double getPerimeter(){
    return side1+side2+side3; }
}
public class Shapes{
    public static void main(String[] args) {
        Shape rect= new Rectangle(6,8);
        Shape circle=new Circle(6);
        Shape tri=new Triangle(2,3,4);

        System.out.println("Rectangle Perimeter: " + rect.getPerimeter());
        System.out.println("Circle Perimeter: " + circle.getPerimeter());
        System.out.println("Triangle Perimeter: " + tri.getPerimeter());
    }
}
```

**OUTPUT:**



```
E:\OOPS\week-08>javac Shapes.java

E:\OOPS\week-08>java Shapes
Rectangle Perimeter: 16.0
Circle Perimeter: 37.69911184307752
Triangle Perimeter: 9.0
```

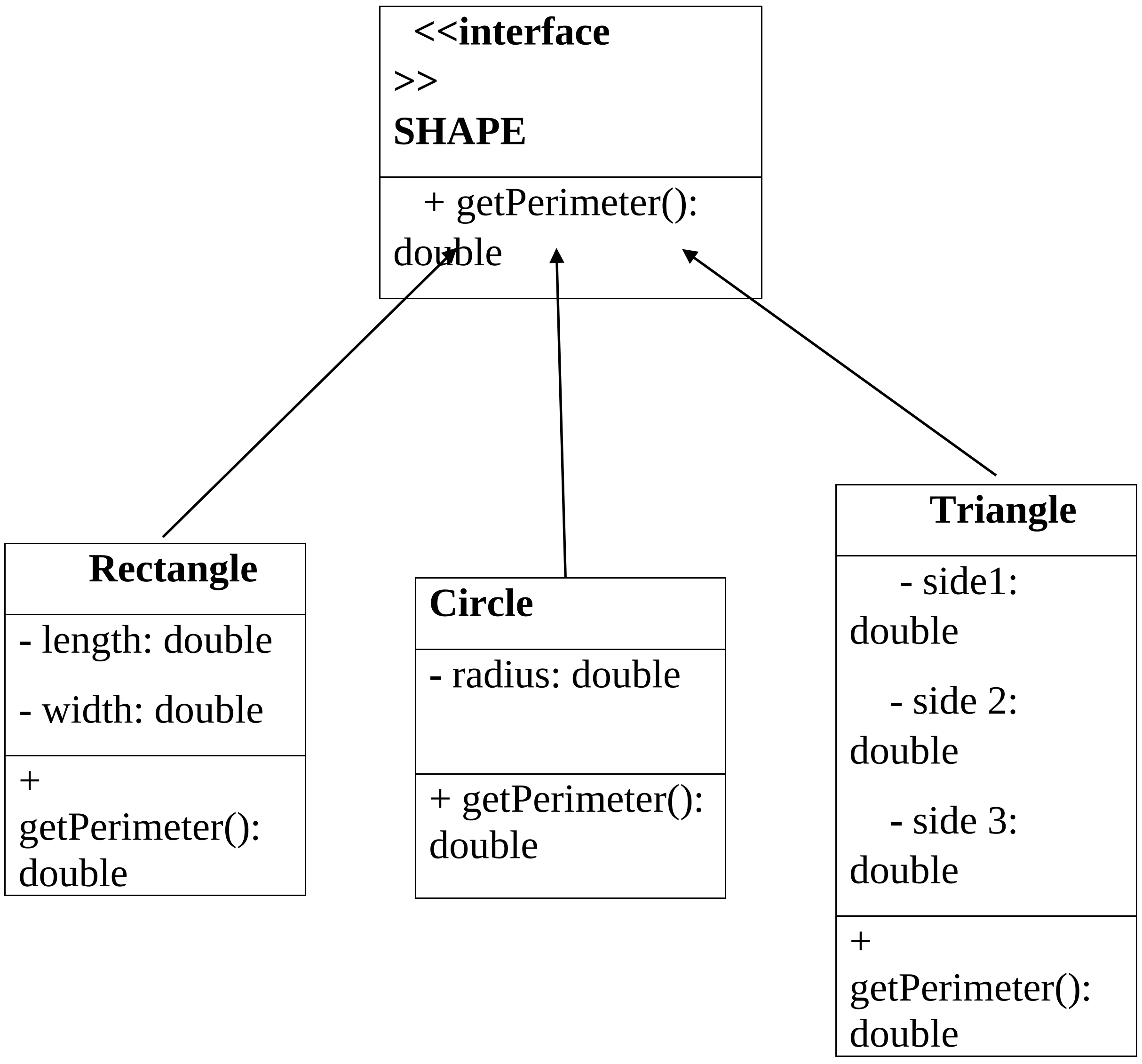
**IMPORTANT POINTS:**

1. Shape is an interface that defines the method getPerimeter() which must be implemented by all classes.
2. Rectangle, Circle, and Triangle classes implement the Shape interface and define their own perimeter logic.
3. Each class has a constructor to set the required dimensions like radius, sides, etc.
4. In main, different objects (Rectangle, Circle, Triangle) are referred using the Shape type — shows runtime polymorphism.

**ERRORS:**

| **Code error** | **2. Code rectification** |
|---|---|
| 1. Missed Math.PI in Circle perimeter,<br><br>2.Did not give meaningful class name. | 3. Rectified to Math.PI in Circle perimeter.<br><br>4. Rectified |

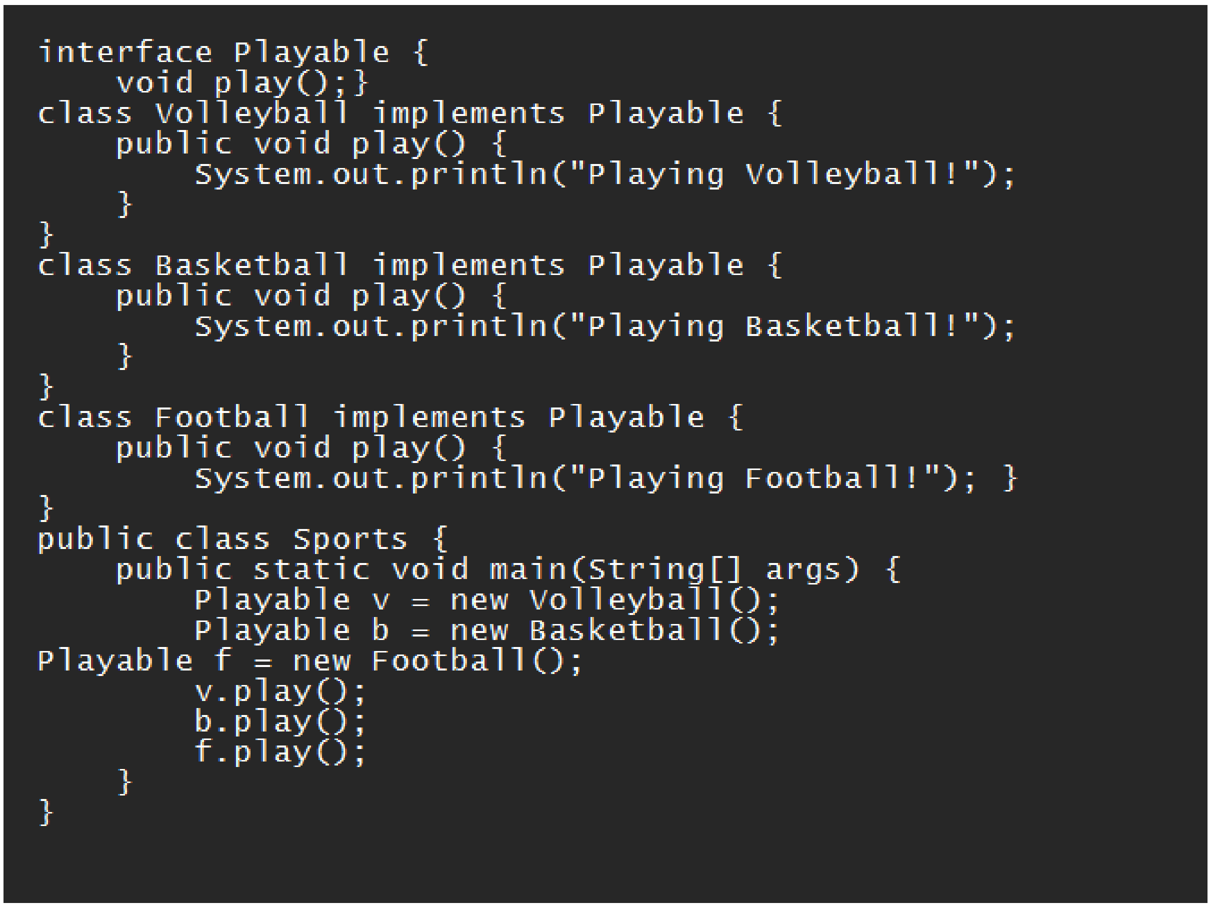
**CLASS DIAGRAM:**



## PROGRAM-2:

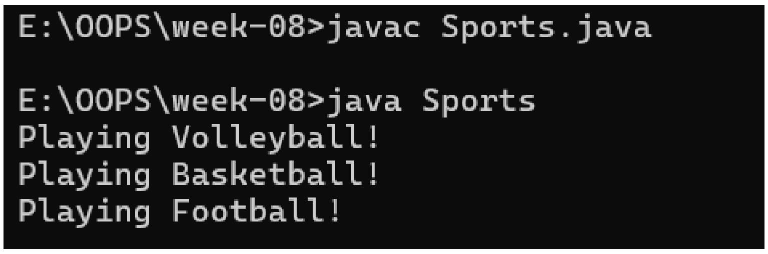
**AIM :** java program to create an interface playable with method play() that takes no argurments and return void create subclasses volleyball basketball football that implements playabale interface and override the play.

**CODE:**



```
interface Playable {
    void play();}
class Volleyball implements Playable {
    public void play() {
        System.out.println("Playing Volleyball!");
    }
}
class Basketball implements Playable {
    public void play() {
        System.out.println("Playing Basketball!");
    }
}
class Football implements Playable {
    public void play() {
        System.out.println("Playing Football!"); }
}
public class Sports {
    public static void main(String[] args) {
        Playable v = new Volleyball();
        Playable b = new Basketball();
Playable f = new Football();
        v.play();
        b.play();
        f.play();
    }
}
```

**OUTPUT:**



```
E:\OOPS\week-08>javac Sports.java

E:\OOPS\week-08>java Sports
Playing Volleyball!
Playing Basketball!
Playing Football!
```

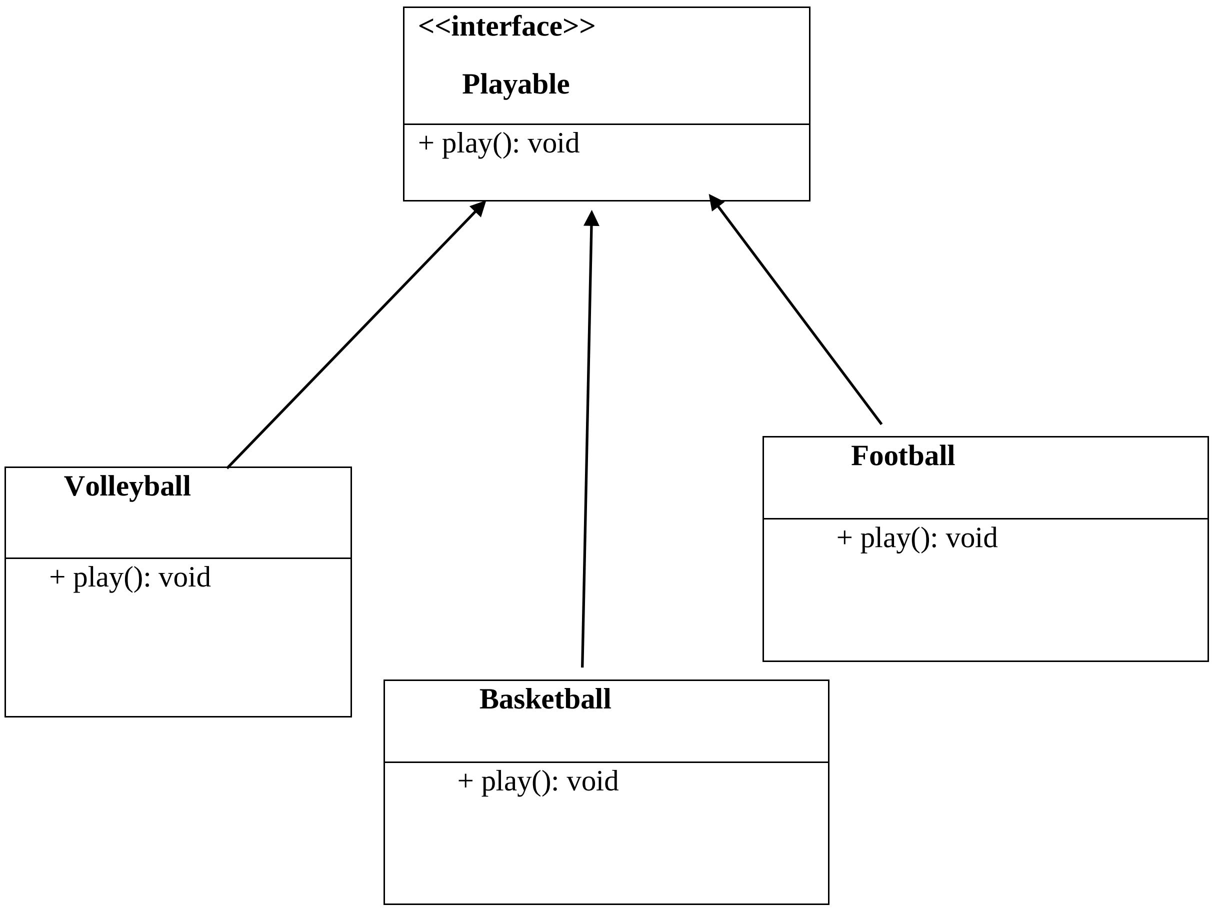
**IMPORTANT POINTS:**

1. Playable is an interface, so it only has abstract methods (by default).
2. All classes implement the Playable interface.
3. Each class overrides the play() method with its own message.
4. Playable is the reference type, but objects are of child classes.

**ERRORS:**

| **Code error** | **Code rectification** |
|---|---|
| 1.Wrote System.out.println(play);<br>2. forgot public in Method play() | 1.Rectified to v.play();<br>2.used public void play() in all implementing classes |

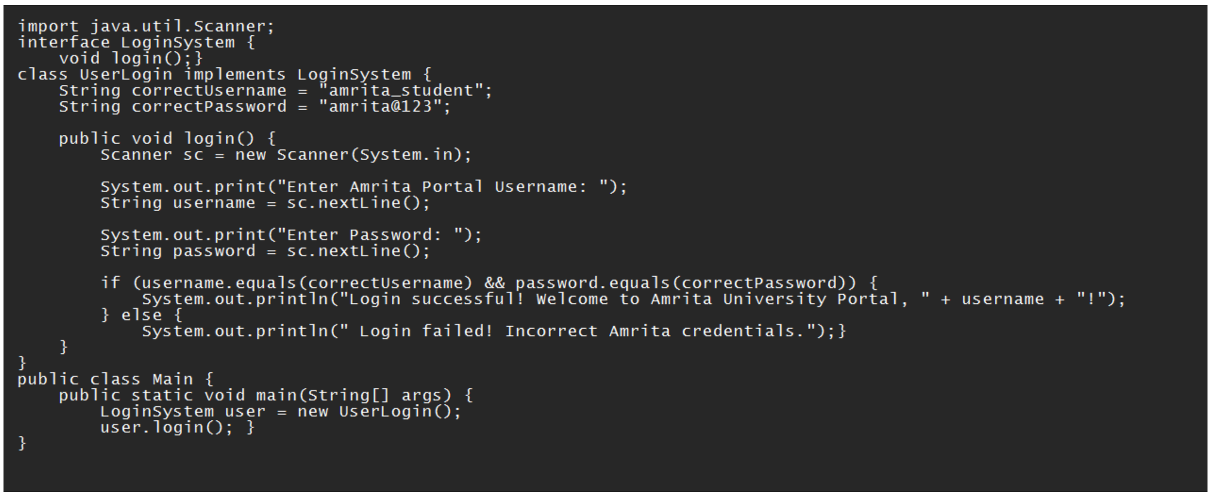
**CLASS DIAGRAM:**



**<u>PROGRAM-3:</u>**

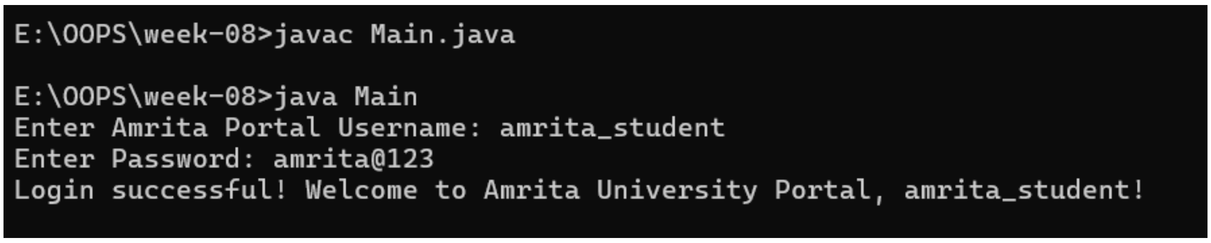
**<u>AIM</u>:** Write a java program to create a login system using interface.

**<u>CODE:</u>**



```java
import java.util.Scanner;
interface LoginSystem {
    void login();}
class UserLogin implements LoginSystem {
    String correctUsername = "amrita_student";
    String correctPassword = "amrita@123";

    public void login() {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter Amrita Portal Username: ");
        String username = sc.nextLine();

        System.out.print("Enter Password: ");
        String password = sc.nextLine();

        if (username.equals(correctUsername) && password.equals(correctPassword)) {
            System.out.println("Login successful! Welcome to Amrita University Portal, " + username + "!");
        } else {
            System.out.println(" Login failed! Incorrect Amrita credentials.");}
    }
}
public class Main {
    public static void main(String[] args) {
        LoginSystem user = new UserLogin();
        user.login(); }
}
```

**<u>OUTPUT:</u>**



```
E:\OOPS\week-08>javac Main.java

E:\OOPS\week-08>java Main
Enter Amrita Portal Username: amrita_student
Enter Password: amrita@123
Login successful! Welcome to Amrita University Portal, amrita_student!
```

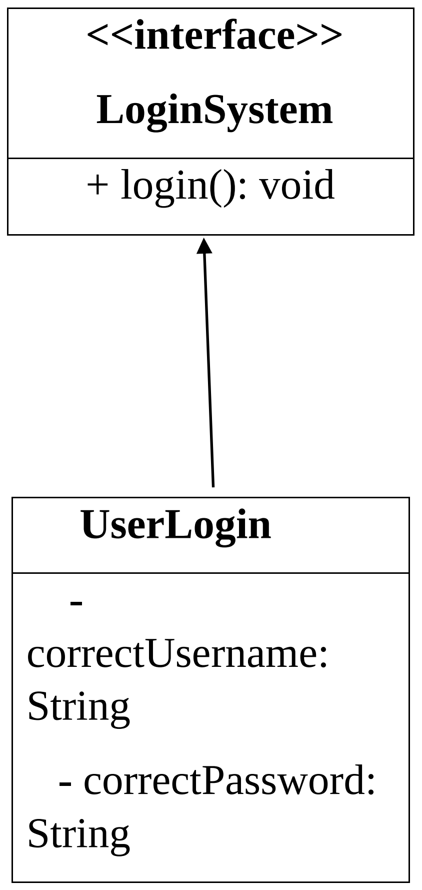
**<u>IMPORTANT POINTS:</u>**

1. LoginSystem is an interface with one abstract method login().
2. UserLogin implements LoginSystem and provides the body for login().
3. It uses Scanner to get username and password input from the user.
4. Validates credentials against hardcoded correct values.
5. Shows polymorphism: interface type LoginSystem refers to UserLogin object.

**ERRORS:**

| Code error | Code rectification |
|---|---|
| 1.Forgot to use .equals() for string comparison<br>2. Didn't import java.util.Scanner.<br>3. Called user.login() with parameters | 1. Used username.equals(correctUsername) not ==<br>2. Added import java.util.Scanner;<br>3. Correct is user.login(); with no parameters |

**CLASS DIAGRAM:**



<<interface>>
LoginSystem
+ login(): void
UserLogin
- correctUsername: String
- correctPassword: String