**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**

**ROLL NO: AV.SC.U4CSE24011**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No. | Title | Date | Page No. | Signature |
| Week 1 |  | 27-01-2025 |  |  |
| 1. | How to download and install Java Software. |  | 3 |  |
| 2. | Write a Java Program to print the message “Welcome to Java Programming”. |  | 6 |  |
| 3. | Write a Java program that prints: Name, Roll.no. , section of a student. |  | 7 |  |
| Week 2 |  | 10-02-2025 |  |  |
| 1. | Write a java program to calculate the area of a rectangle. |  | 8 |  |
| 2. | Write a java program to temperature from Celsius to Fahrenheit and vica-versa. |  | 9 |  |
| 3. | Write a java program to calculate the simple interest. |  | 10 |  |
| 4. | Write a java program to find the largest of three numbers, using ternary operator. |  | 11 |  |
| 5. | Write a java program to find the factorial of a number. |  | 12 |  |
| Week 3 |  | 24-02-2025 |  |  |
|  | To create a java program with the following instructions:   1. Create a class with name “Car” 2. Create 4 attributes, named: car\_color, car\_brand, fuel\_type, mileage 3. Create 3 methods, named: start(), service(), stop() 4. Create 3 objects, named: car1, car2, car3 5. Create a constructor, which should print, “Welcome to car garage”. |  | 13 |  |
|  | To write a java program to create a class named BankAccount, with 2 methods deposit() and withdraw().   1. deposit(): Whenever an amount is deposited, it has to be update the current amount. 2. withdraw(): Whenever an amount is withdrawn, it has to be less than the current amount , else print (“Insufficient funds”) |  | 15 |  |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**.**



**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.

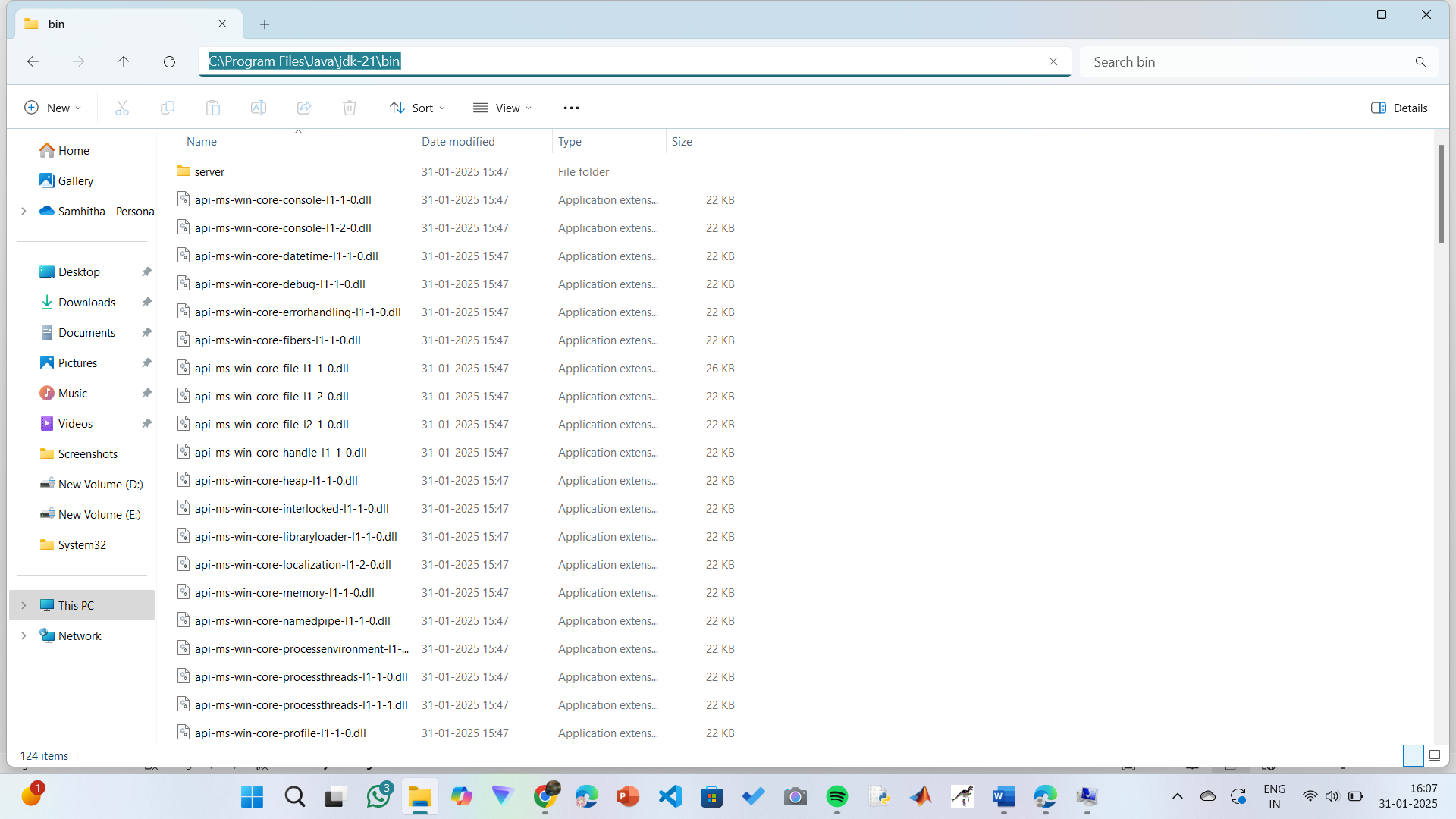


**Step 3: Setting up the path**

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.

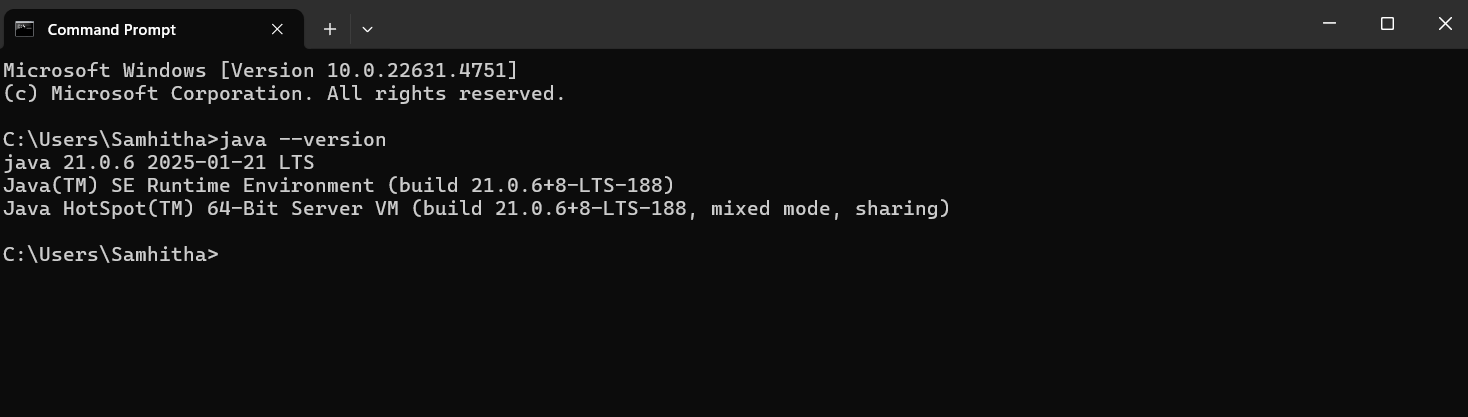
2)Click New and add: C:\Program Files\Java\jdk-21\bin

3)Click OK to save.



**Step 7:Verify Installation**

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



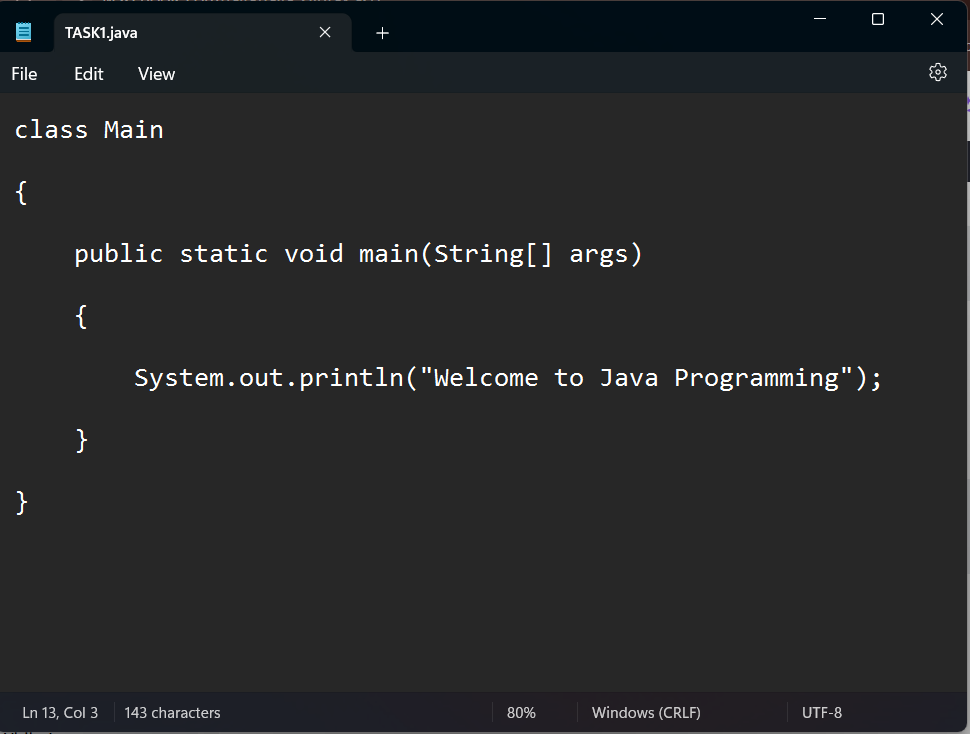
1. To check the java compiler type: **javac –version.**



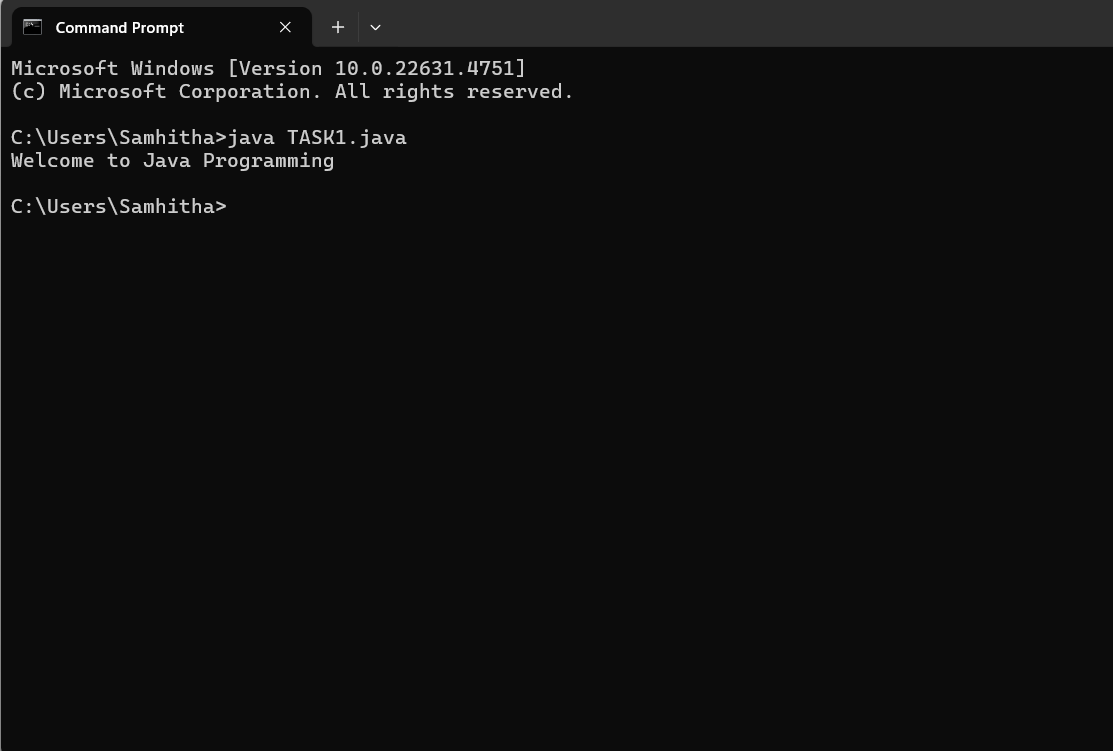
**PROGRAM-2:**

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

**CODE:**

****

**OUTPUT:**

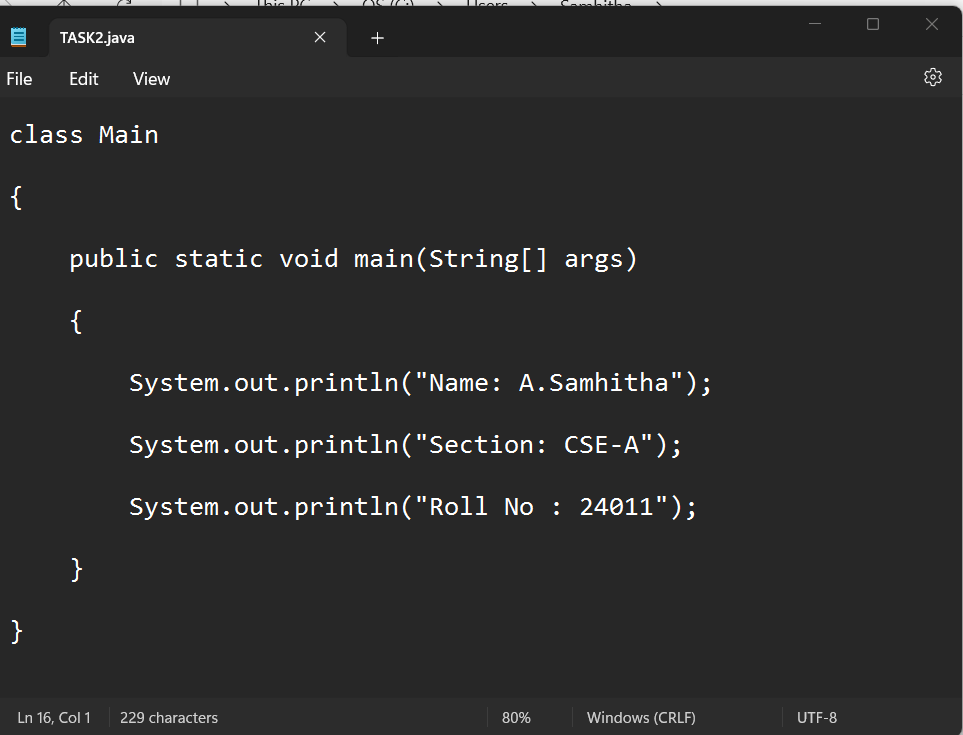
****

**ERRORS:** None found

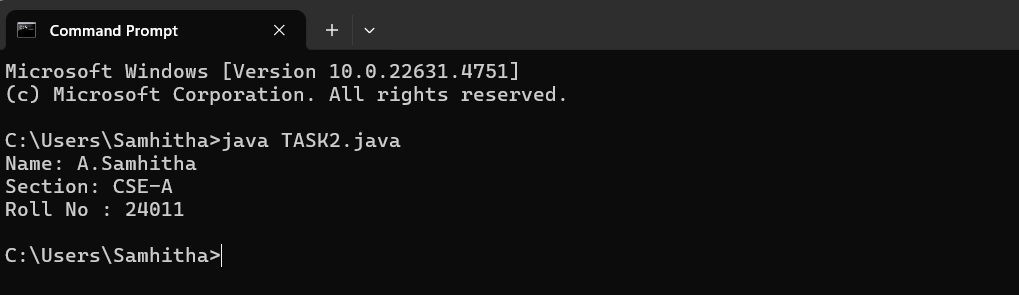
**PROGRAM-3:**

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

**CODE:**

****

**OUTPUT:**

****

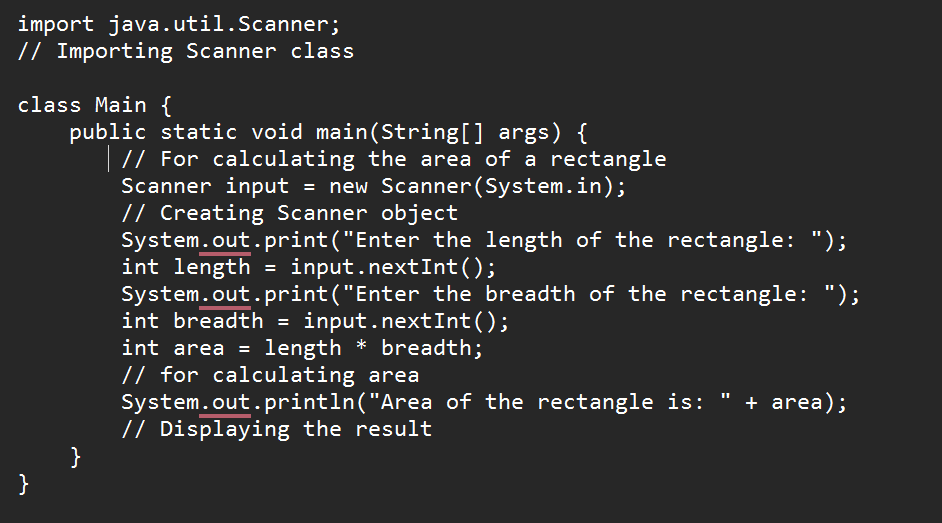
**ERRORS:** None Found

**WEEK-02**

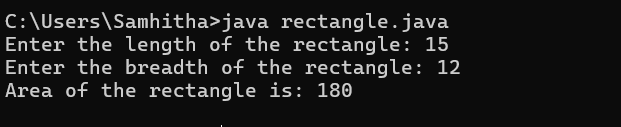
**PROGRAM-01:**

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

**CODE:**

****

**OUTPUT:**

****

**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 |

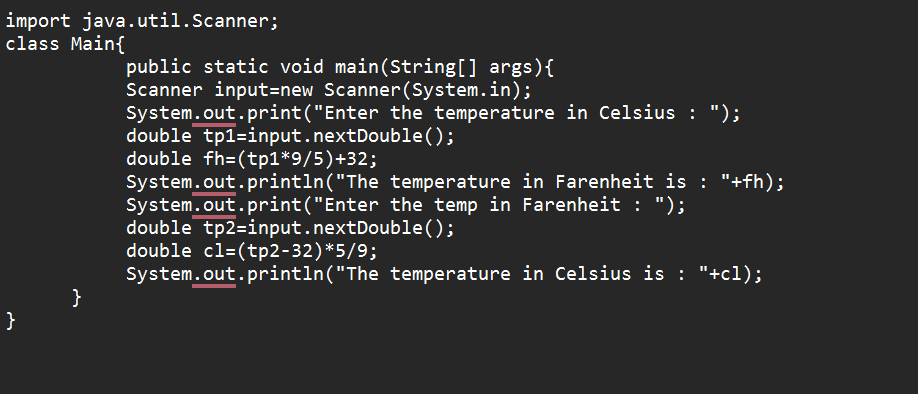
**Concepts to be known:**

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.

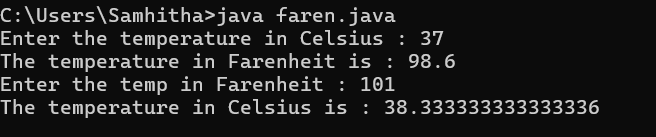
**PROGRAM-02:**

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

**CODE:**

****

**OUTPUT:**

****

**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. |

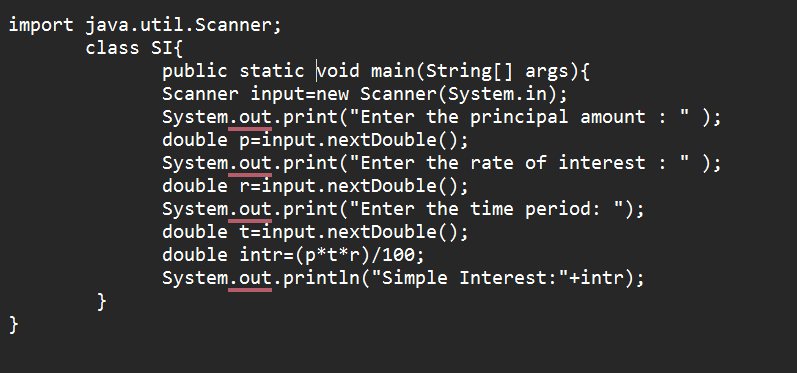
**Concepts to be known:**

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.

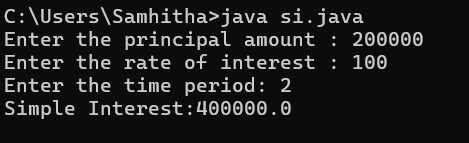
**PROGRAM-03:**

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

**CODE:**

****

**OUTPUT:**

****

**ERRORS:**

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

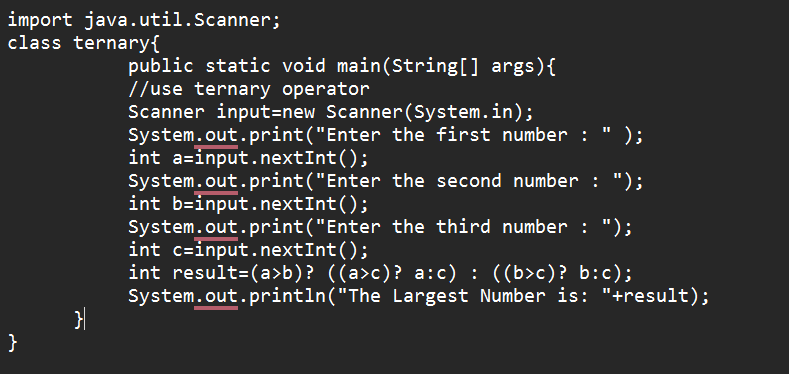
**Concepts to be known:**

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.

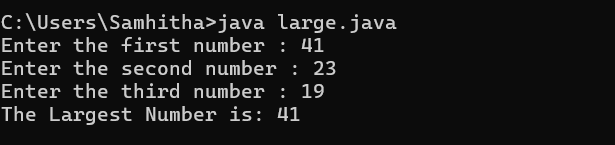
**PROGRAM-04:**

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

**CODE:**

****

**OUTPUT:**

****

**ERRORS:**

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

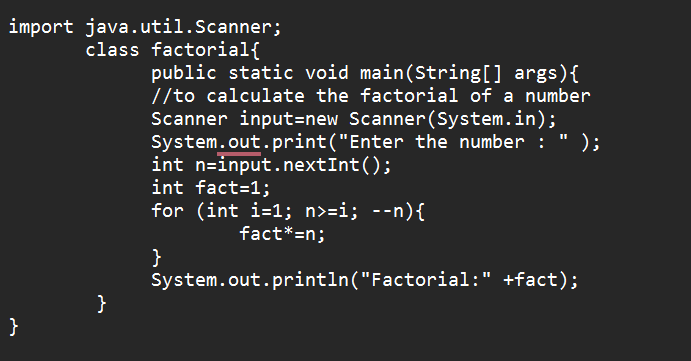
**Concepts to be known:**

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.

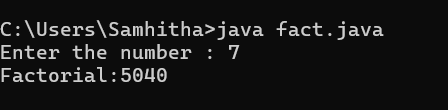
**PROGRAM-05:**

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

**CODE:**

****

**OUTPUT:**

****

**ERROR:**

|  |  |  |
| --- | --- | --- |
| **Sno.** | **Error** | **Error rectification** |
| 1. | error: ';' expected  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.

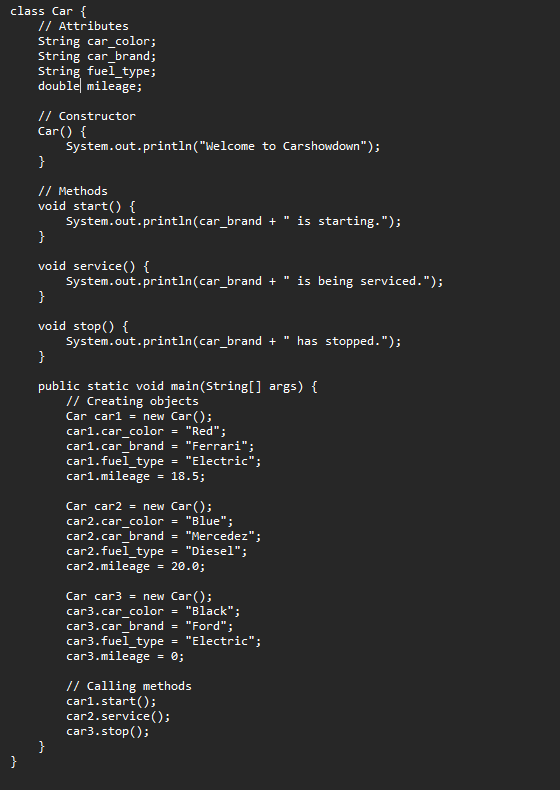
**WEEK-03**

**PROGRAM-01:**

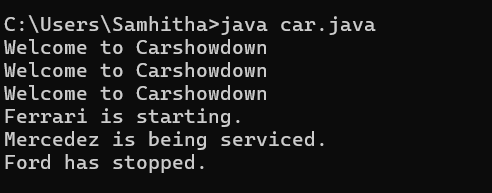
**AIM:** To create a java program with the following instructions:

1. Create a class with name “Car”
2. Create 4 attributes, named: car\_color, car\_brand, fuel\_type, mileage
3. Create 3 methods, named: start(), service(), stop()
4. Create 3 objects, named: car1, car2, car3
5. Create a constructor, which should print, “Welcome to car garage” .

**CODE:**

****

**OUTPUT:**

****

**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  + car\_brand: String  + fuel\_type: String  + mileage: int |
| + Car(): void  + start(): void  + service(): void  + stop(): void |

**Concepts to be known:**

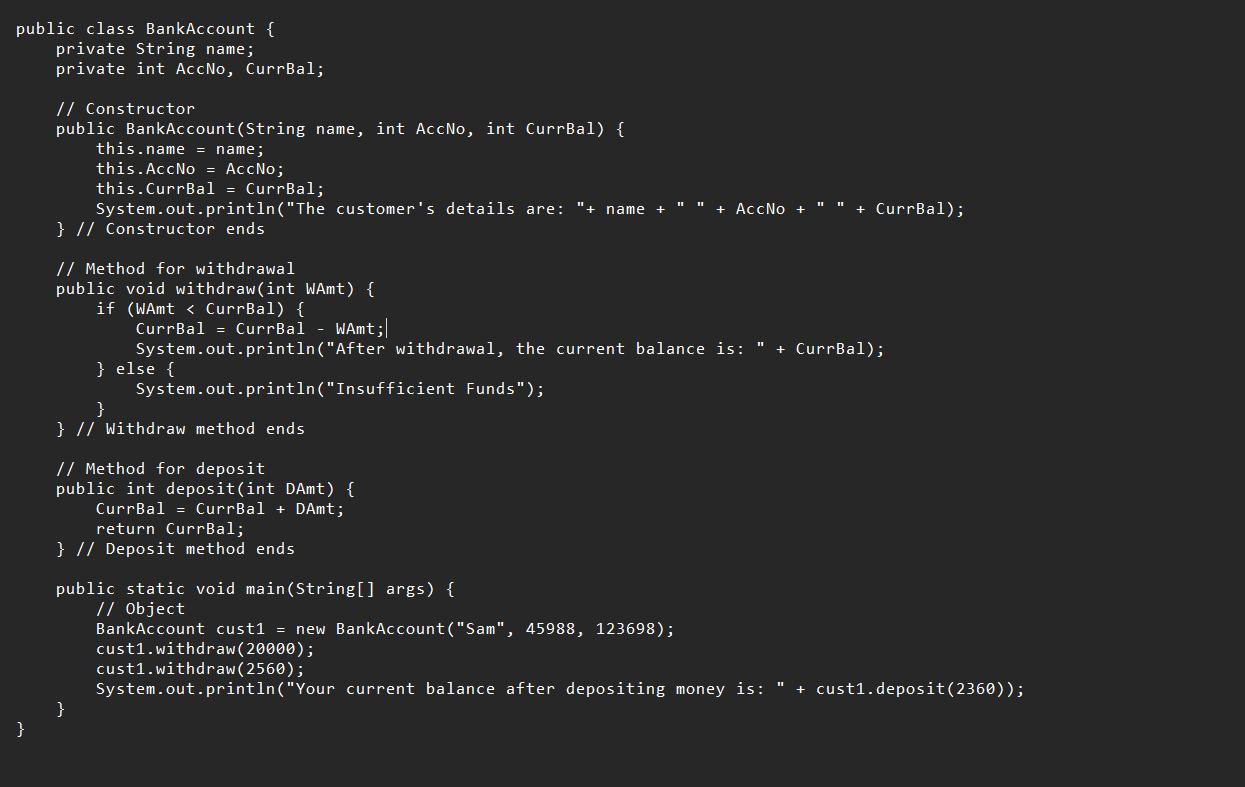
1. String car\_color; declares an instance variable car\_color of type String, which will store the color of the car.
2. Car() is a constructor that gets executed when an object of the Car class is created, printing "Welcome to Carshowdown".
3. this.car\_color = car\_color; is used inside a constructor 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:**

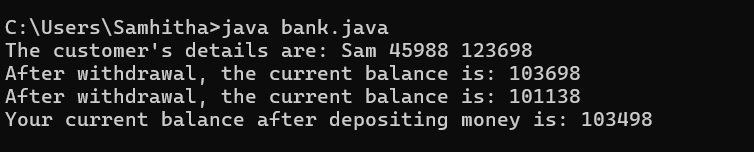
**AIM:** To write a java program to create a class named BankAccount, with 2 methods deposit() and withdraw().

1. deposit(): Whenever an amount is deposited, it has to be update the current amount.
2. withdraw(): Whenever an amount is withdrawn, it has to be less than the current amount , else print (“Insufficient funds”)

**CODE:**

****

**OUTPUT:**

****

**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **Sno.** | **Error** | **Error rectification** |
| 1. | error: ';' expected cust1.withdraw(3050) | Add a “;”  cust1.withdraw(3050); |
| 2. | error: cannot find symbol thisCurrBal=CurrBal; | Add a “.”  this.CurrBal=CurrBal; |

**CLASS DIAGRAM:**

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

**Concepts to be known:**

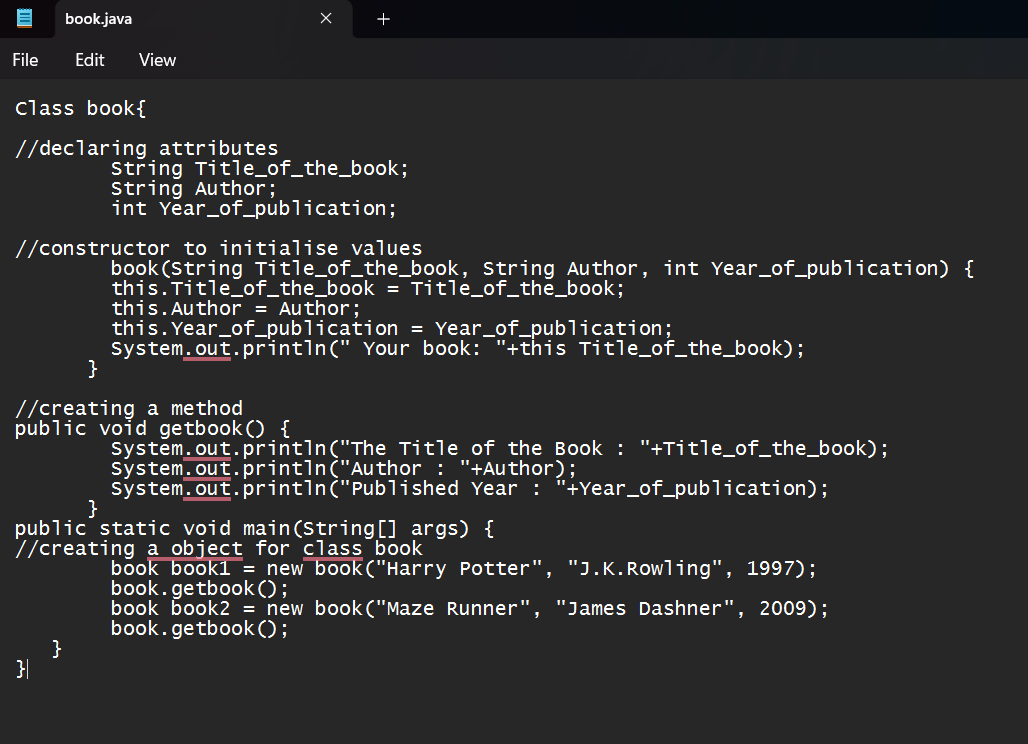
1. Classes and Objects – The program defines a BankAccount class and creates an object (cust1) to access methods and store account details.
2. Constructors – The constructor BankAccount(String name, int AccNo, int CurrBal) initializes the object with values when a new bank account is created.
3. Instance 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**

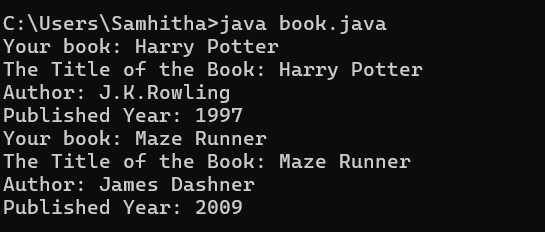
**AIM:** Write a java program with class named “book”, the class should contain various attributes such as title, author, year of publication it should also contain a constructor with parameters which initializes, title, author, and year of publication.

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

**CODE:**



**OUTPUT:**

****

**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:**

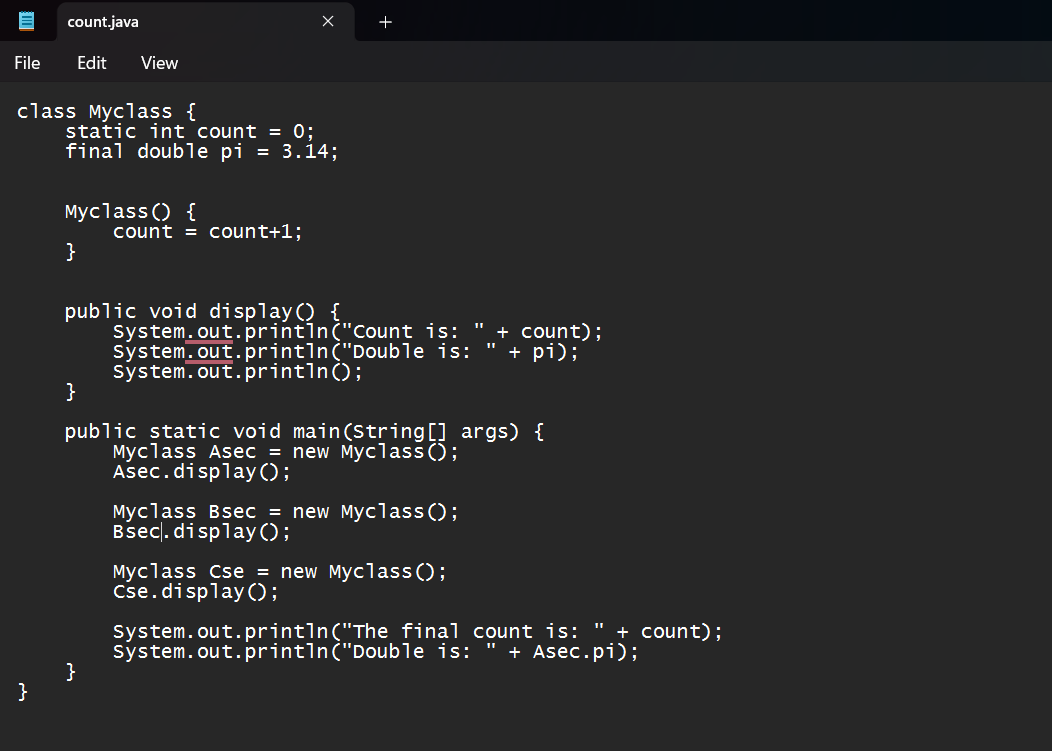
1. The class Book follows Java naming conventions and represents a real-world book entity.
2. Stores book details (Title\_of\_the\_book, Author, Year\_of\_publication).
3. Initializes object properties using the this keyword to avoid variable name conflicts.
4. Displays book details, demonstrating encapsulation 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:**

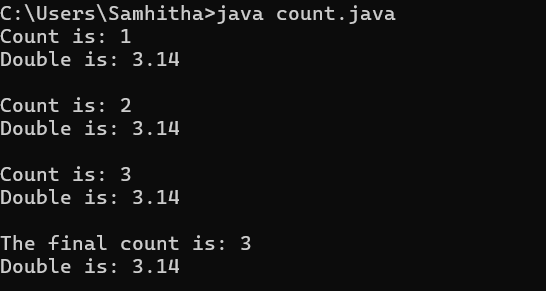
|  |
| --- |
| Book  -Title\_of\_the\_book: String  -Author: String  -Year\_of\_publication: int  + Book(title: String,Author: String;Year of publication: int  + getbook( ): void |

**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:**

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

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

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