23CSE111

**OBJECT-ORIENTED PROGRAMMING**

**LAB MANUAL**

****

**Department of Computer Science Engineering**

**Amrita School of Computing**

**Amrita Vishwa Vidyapeetham, Amaravati Campus**

**VERIFIED BY: NAME : A. SAMHITHA**

**ROLLNO:AV.SC.U4CSE24011**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No. | Title | Date | Page No. | Signature |
| Week 1 |  | 27-01-25 |  |  |
| 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 |  | 10-02-25 |  |  |
| 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- 25 |  |  |
|  | 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 withdraw(): Whenever an amount is withdrawn, it has to be less than the current amount , else print (“Insufficient funds”) |  | 15 |  |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.

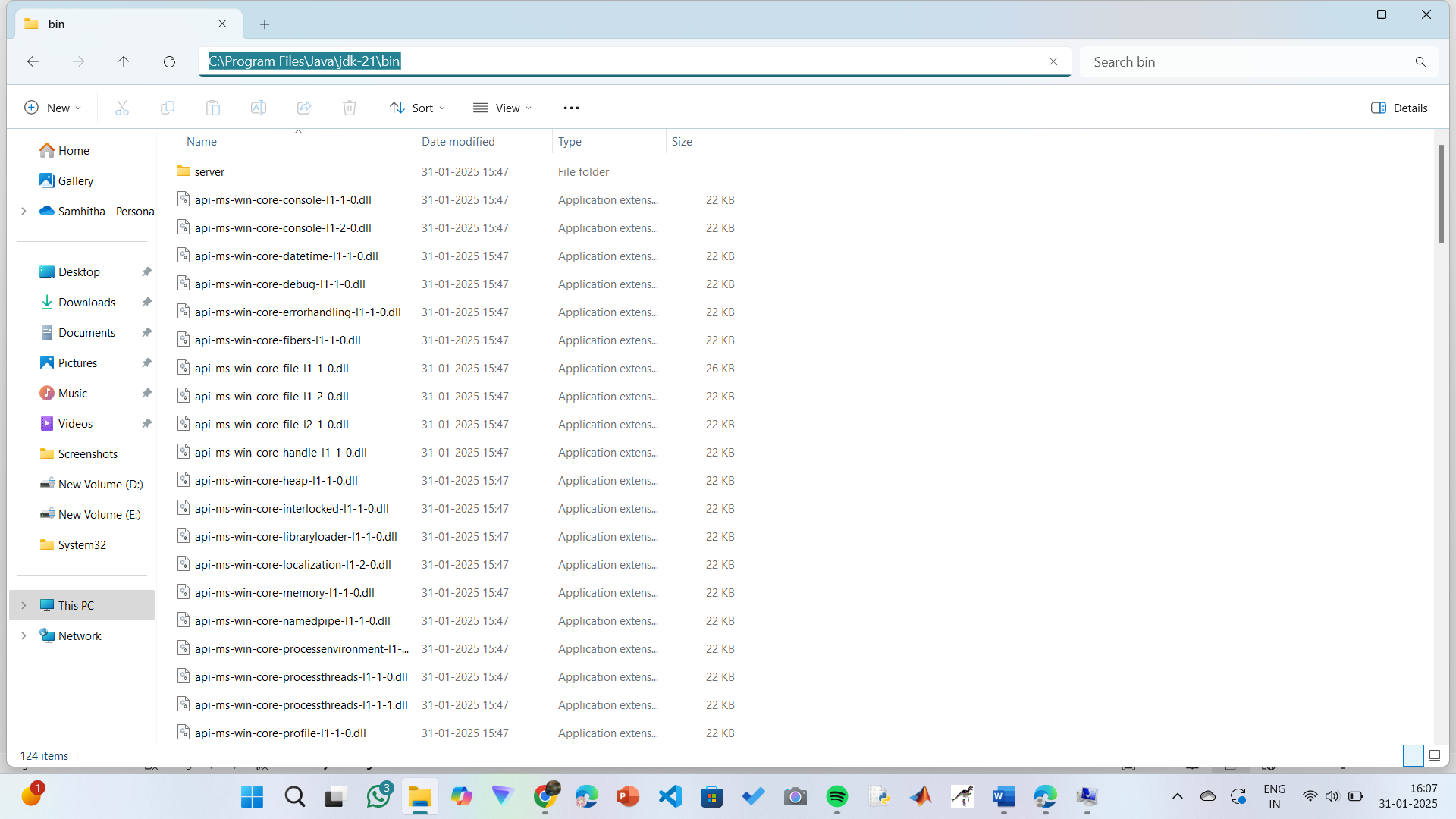


**Step 3: Setting up the path**

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.

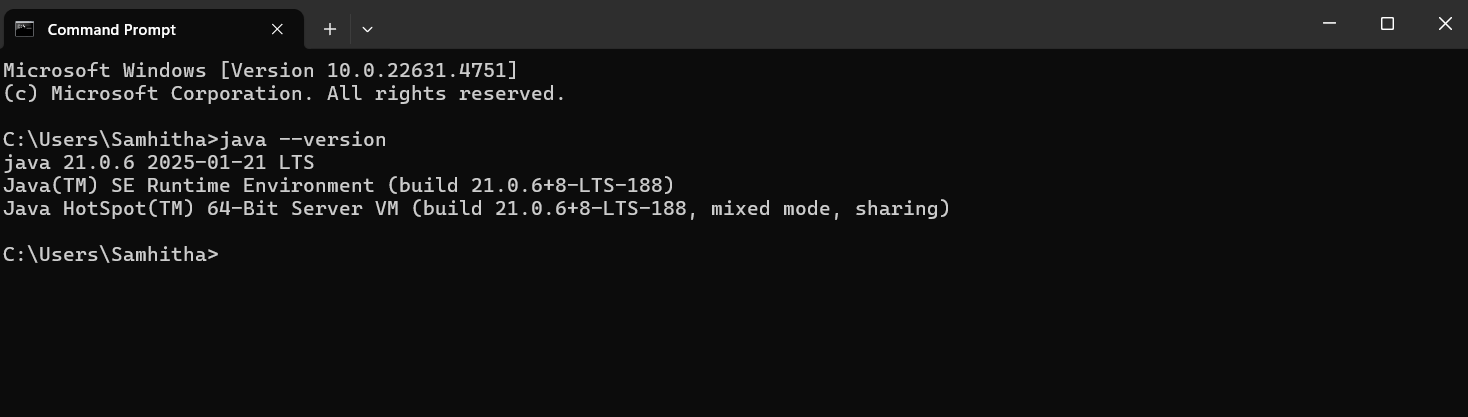
2)Click New and add: C:\Program Files\Java\jdk-21\bin

3)Click OK to save.



**Step 7:Verify Installation**

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



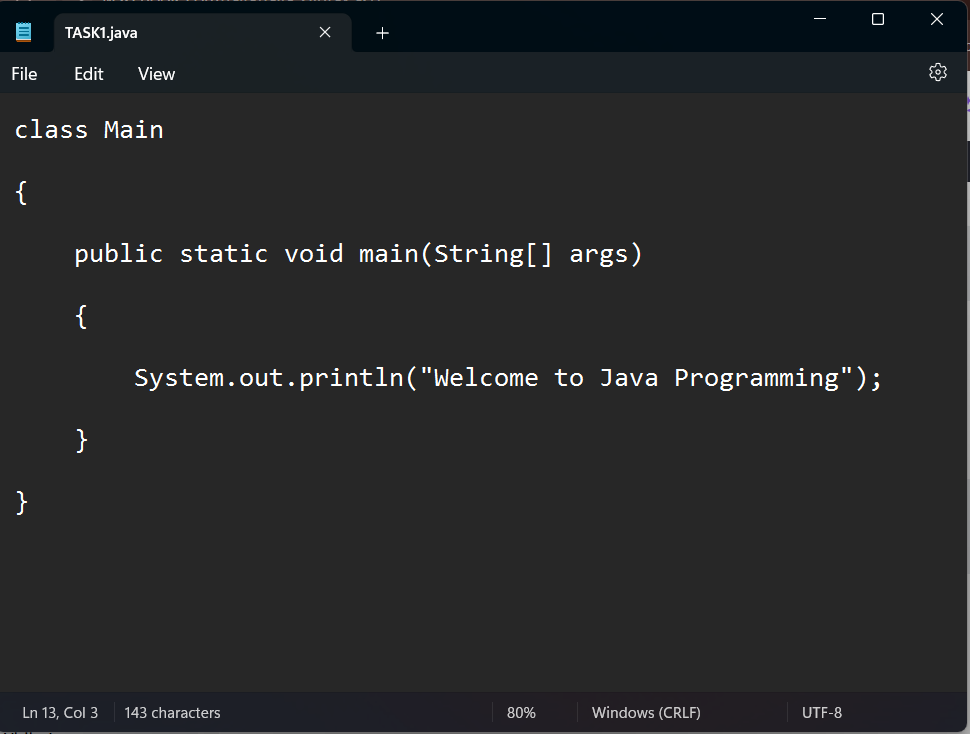
1. To check the java compiler type: **javac –version.**



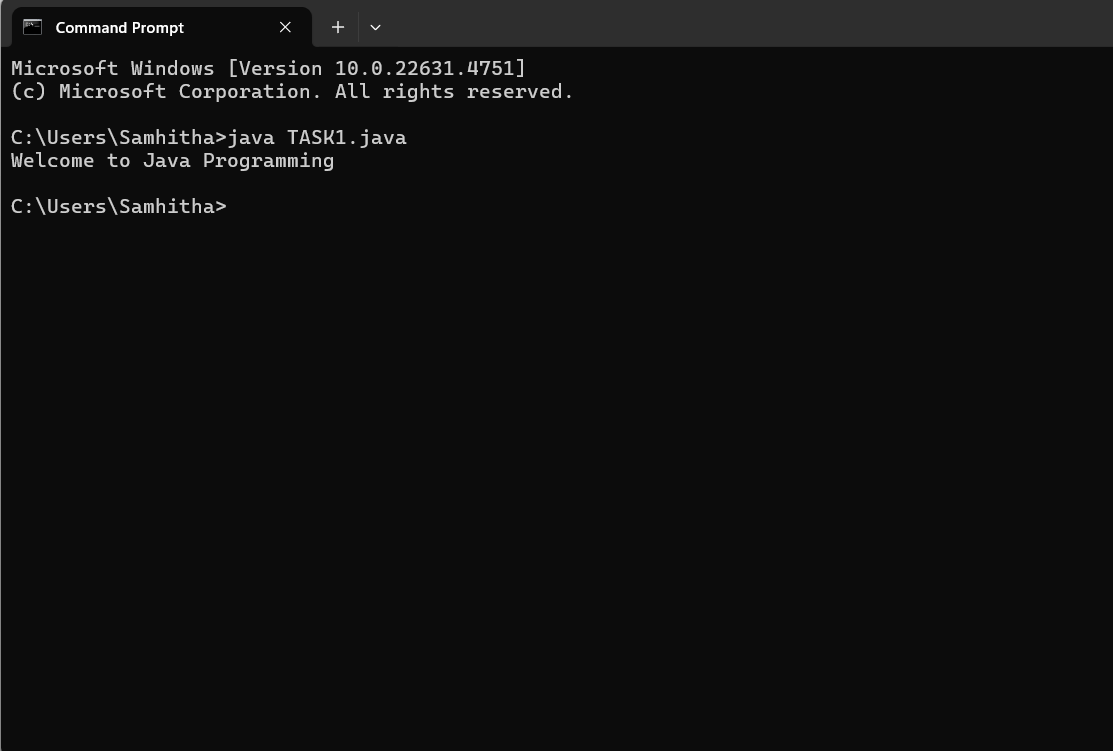
**PROGRAM-2:**

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

**CODE:**

****

**OUTPUT:**

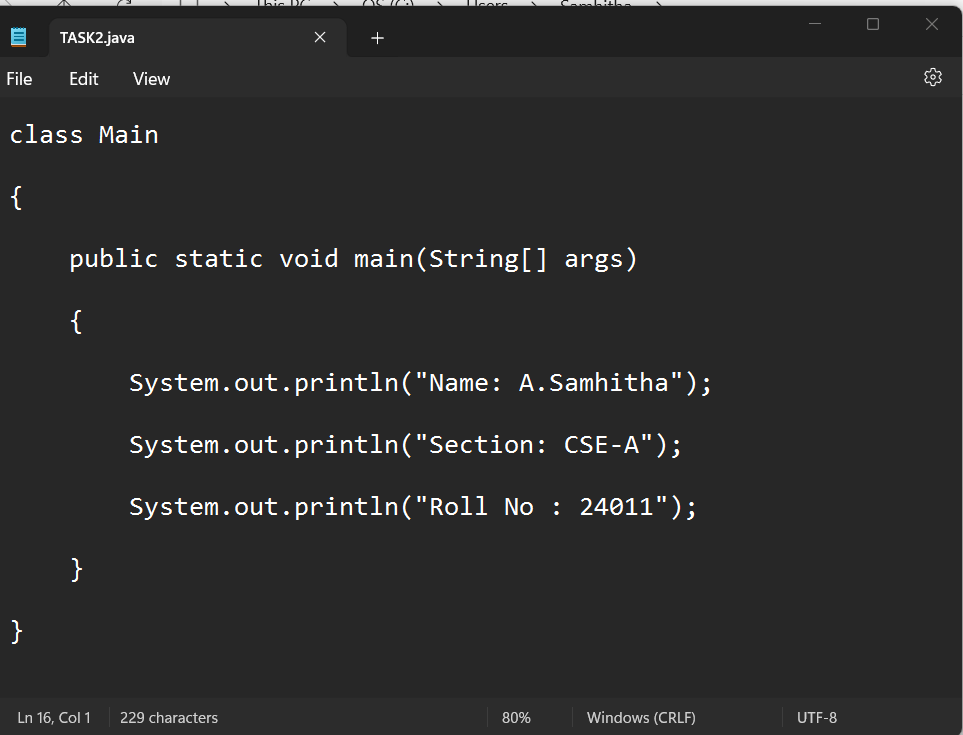
****

**ERRORS:** None found

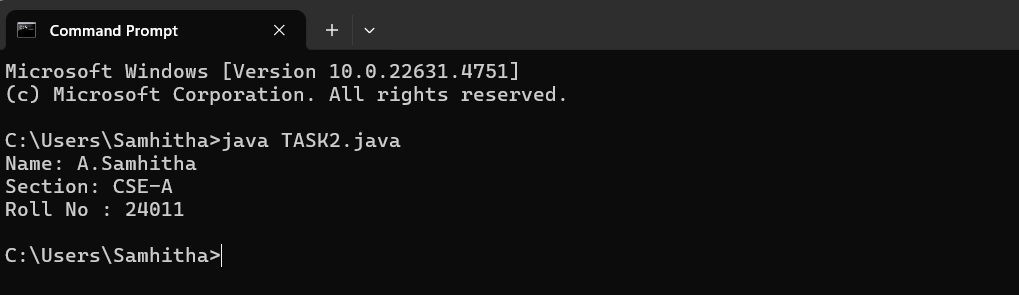
**PROGRAM-3:**

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

**CODE:**

****

**OUTPUT:**

****

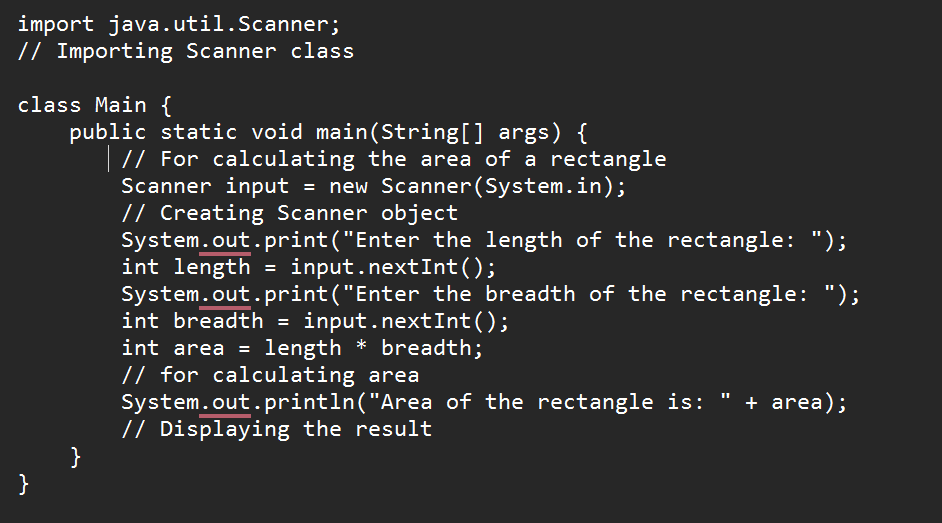
**ERRORS:** None Found

**WEEK-02**

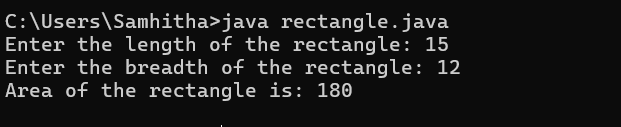
**PROGRAM-01:**

**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 |

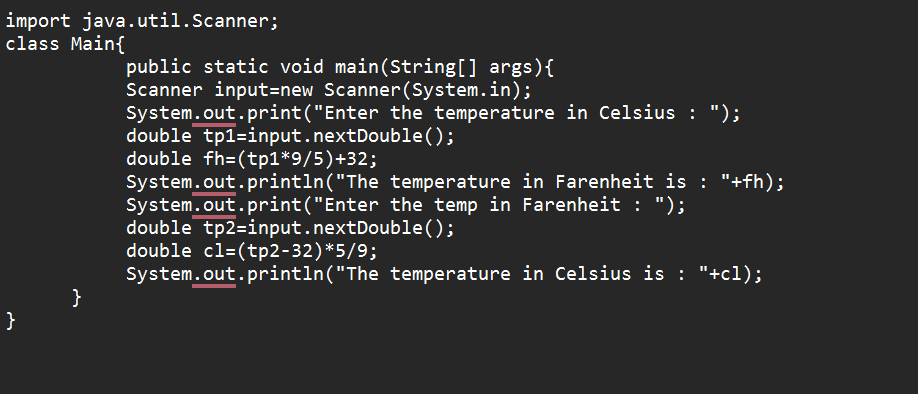
**Concepts to be known:**

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.

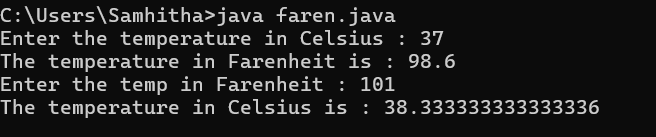
**PROGRAM-02:**

**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. |

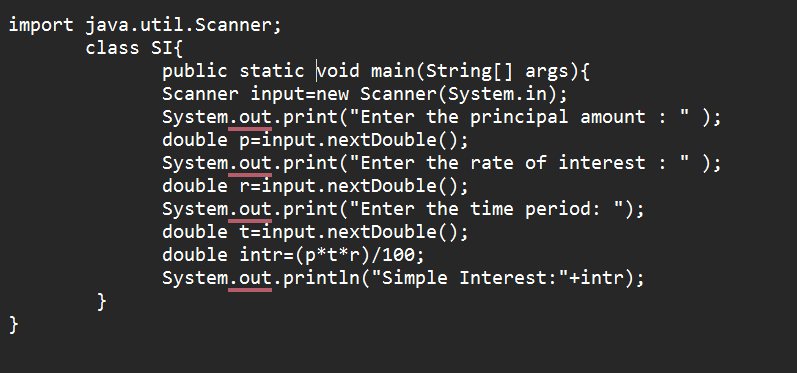
**Concepts to be known:**

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.

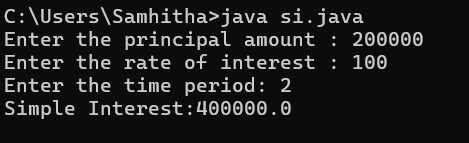
**PROGRAM-03:**

**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(); |

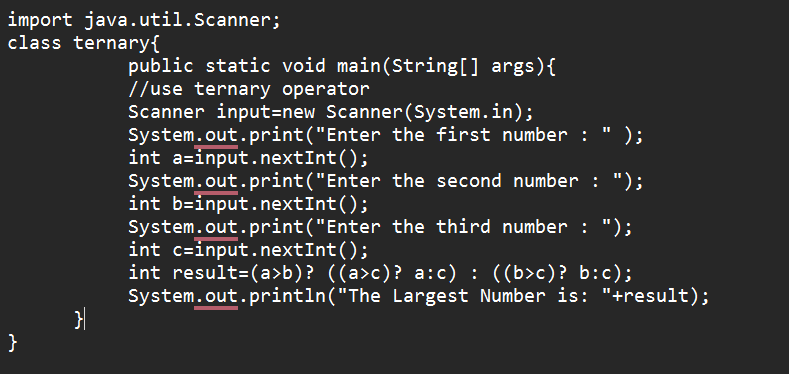
**Concepts to be known:**

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.

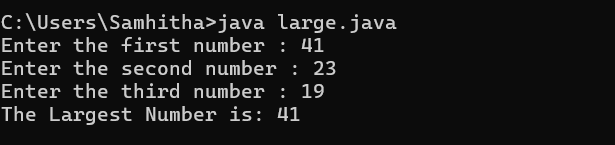
**PROGRAM-04:**

**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); |

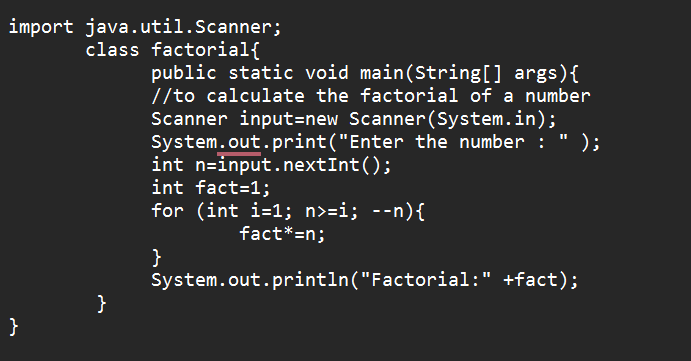
**Concepts to be known:**

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.

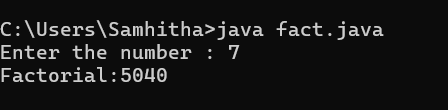
**PROGRAM-05:**

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

**CODE:**

****

**OUTPUT:**

****

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

**ERROR:**

**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.

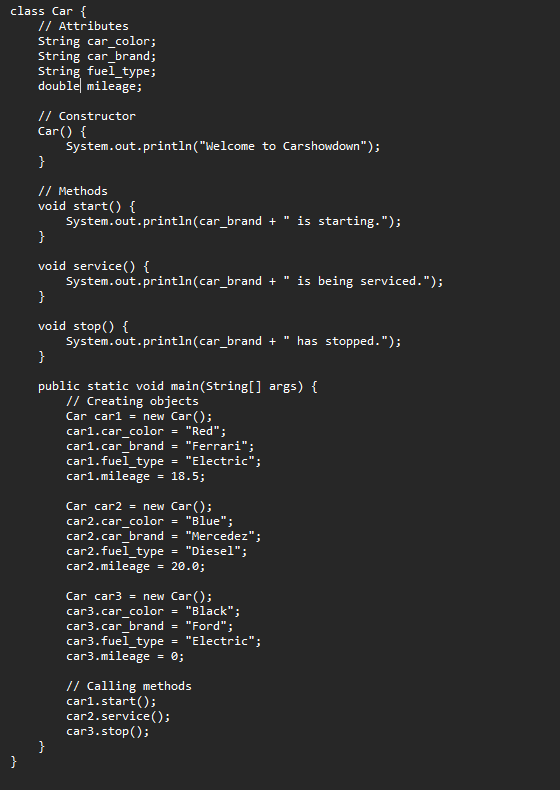
**WEEK-03**

**PROGRAM-01:**

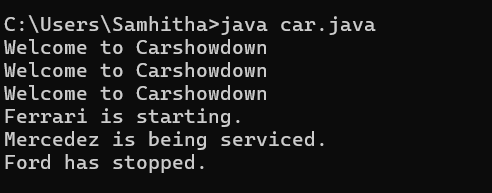
**AIM:** To create a java program with the following instructions:

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:**

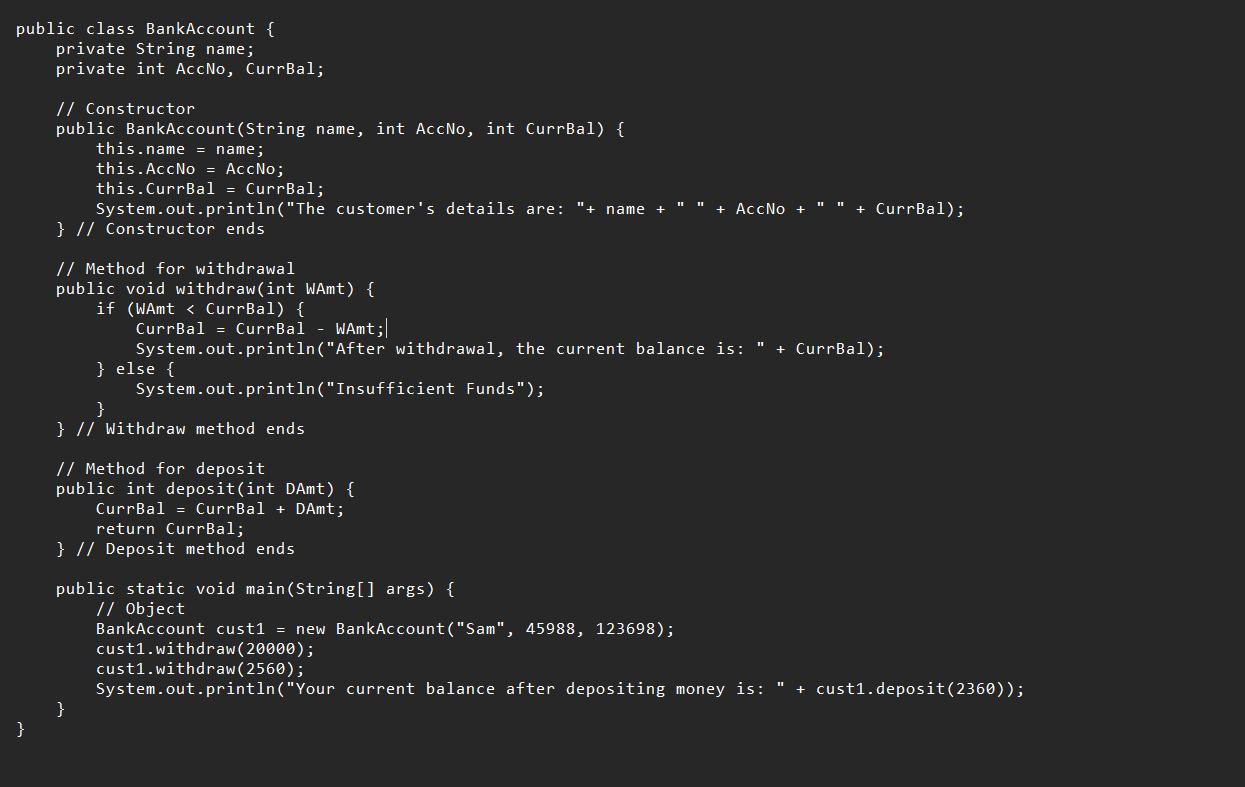
1. String car\_color; declares an instance variable car\_color of type String, which will store the color of the car.
2. Car() is a constructor that gets executed when an object of the Car class is created, printing "Welcome to Carshowdown".
3. this.car\_color = car\_color; is used inside a constructor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

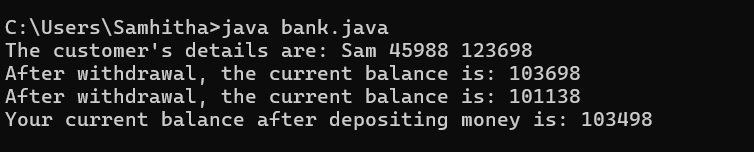
**AIM:** To write a java program to create a class named BankAccount, with 2 methods deposit() and withdraw().

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:**

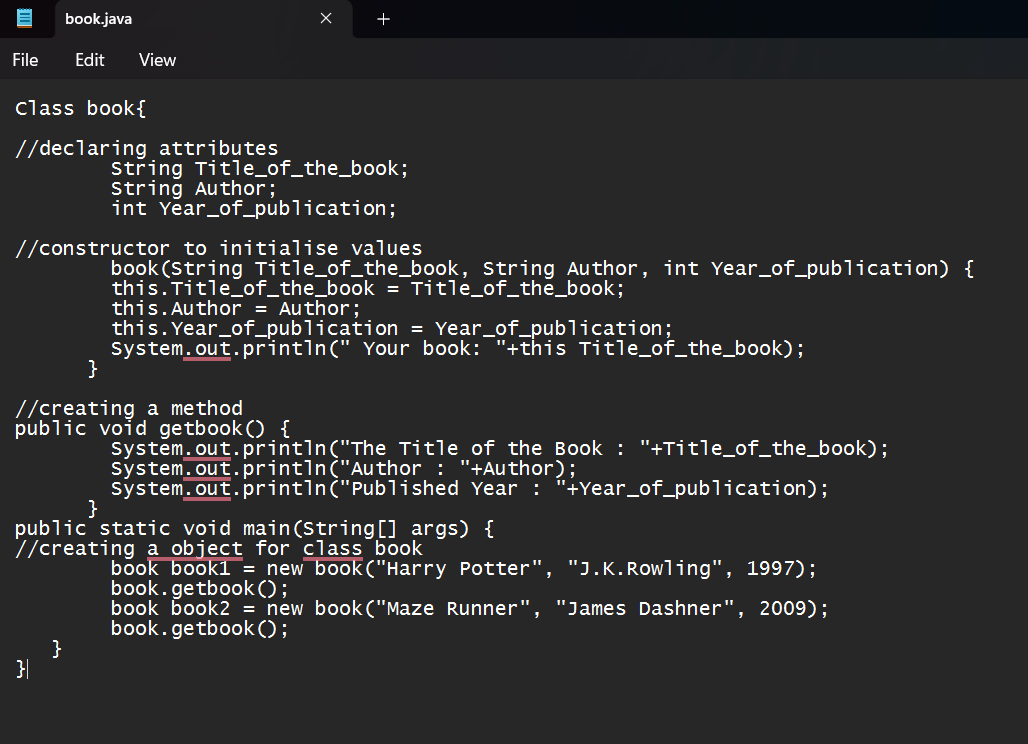
1. Classes and Objects – The program defines a BankAccount class and creates an object (cust1) to access methods and store account details.
2. Constructors – The constructor BankAccount(String name, int AccNo, int CurrBal) initializes the object with values when a new bank account is created.
3. Instance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

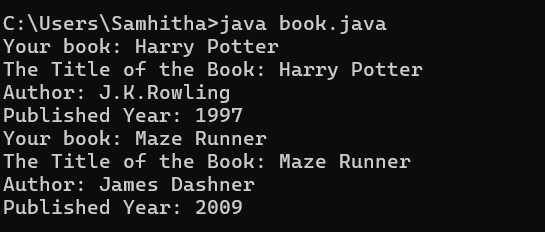
**AIM:** Write a java program with class named “book”, the class should contain various attributes such as title, author, year of publication it should also contain a constructor with parameters which initializes, title, author, and year of publication.

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.

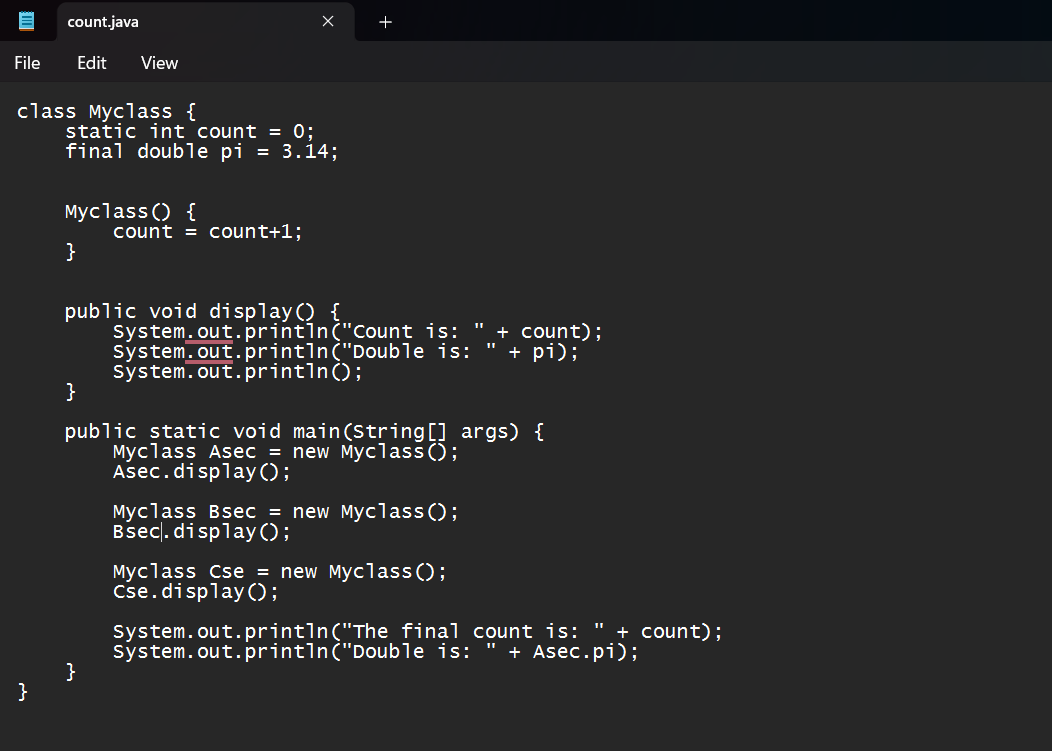
**CLASS DIAGRAM:**

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

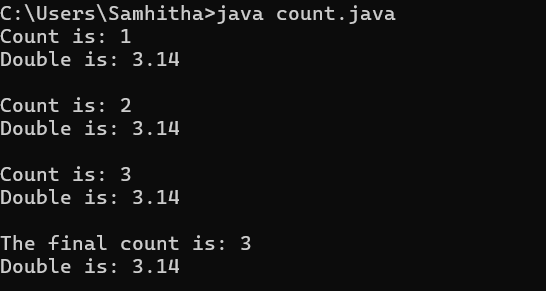
**PROGRAM-2:**

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

**CODE:**

****

**OUTPUT:**

****

**ERROR:**

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

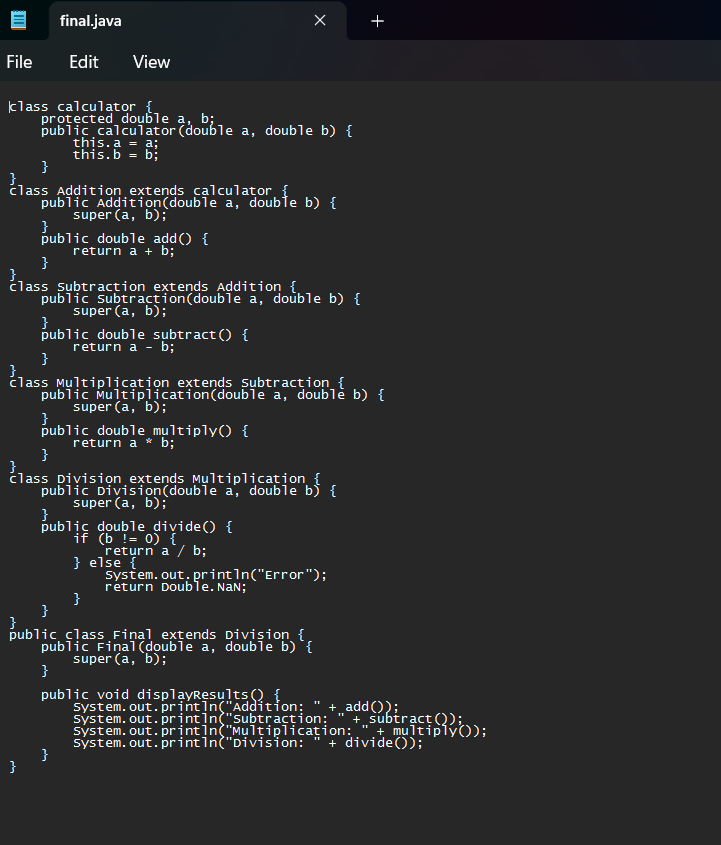
|  |
| --- |
| Myclass |
| -Count: int  -Pi: double |
| + myclass( )  + main(args: String[]): void |

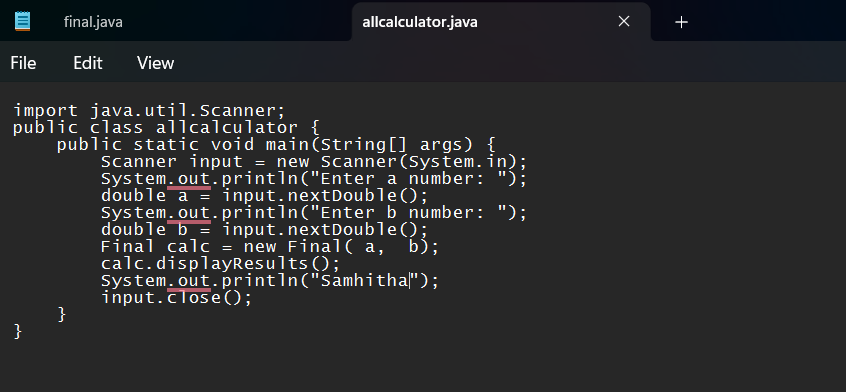
**WEEK-05**

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

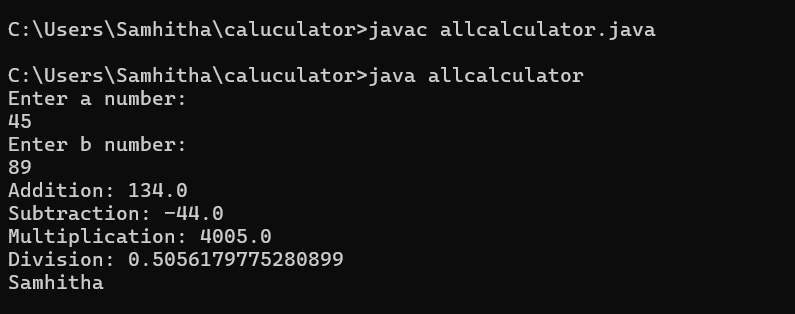
Hint: collect required variables using super class, create each class for a parameter and each class must contain a method.

**CODE:**





**OUTPUT:**



**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. 2. Not mentioning super to obtain the super class constructor. | 1. After declaring methods, we must provide the return method correctly. 2. To obtain the super class we need to mention super. |

**CLASS DIAGRAM:**

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

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

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

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

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

**PROGRAM-2:**

**AIM:** A vehicle rental company wants to develop a system that maintains information about different types of vechicles available for rent the company rents out cars and bikes, and they need a program to store details about each vehicle, such as brand and speed( should be in super class)

1. cars should have an additional property: no.of doors
2. Bikes should have a property indicating whether they have gears or not.
3. The system should also include a function to display details about each vehicle and indicate when a vehicle is starting.
4. Every class should have a constructor

**Question:**

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

**CONCEPTS TO BE KNOWN:**

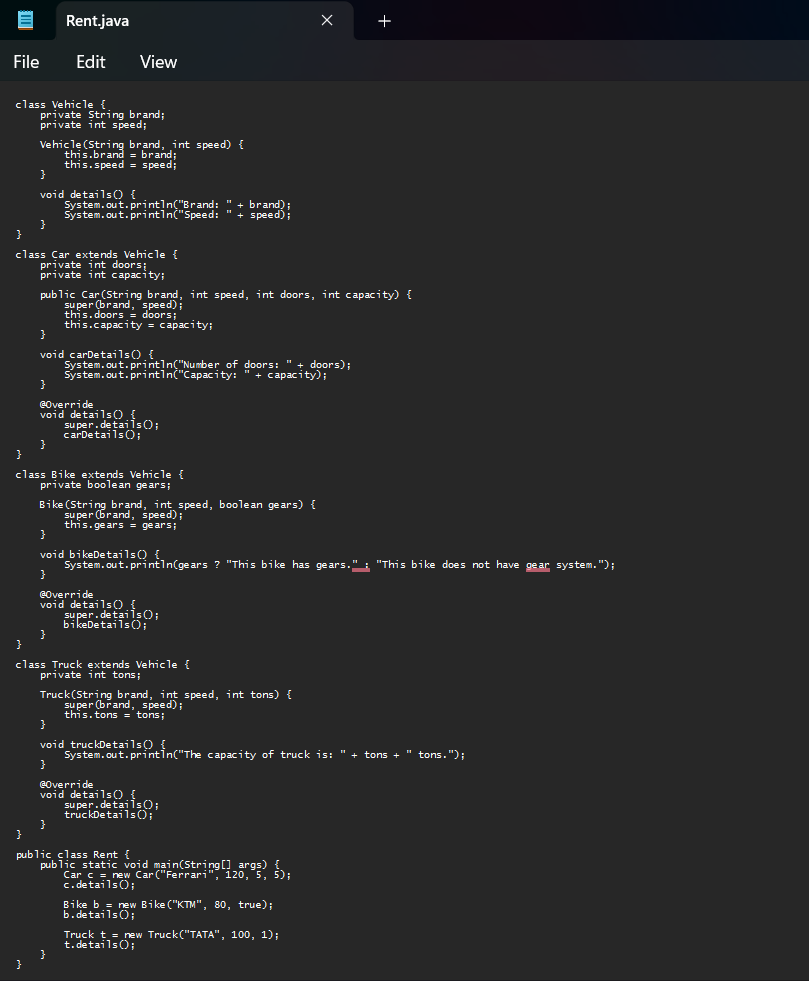
1. a constructor helps in initializing an object that doesn't exist.
2. a method performs functions on pre-constructed or already developed objects.
3. a double method can represent more decimal point numbers than float method.
4. the void keyword in java is used to specify that a method does not return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

**Answer for Q1:**

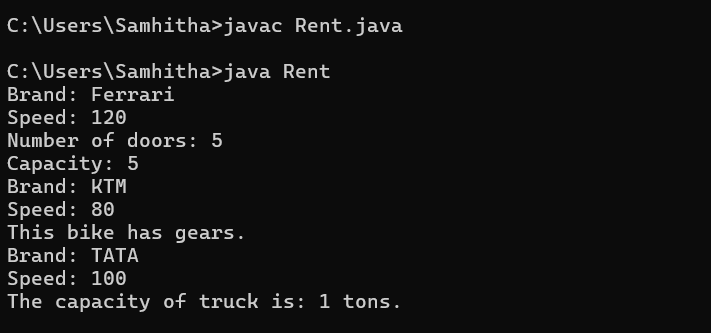
The oops concepts used in the above program are:

Inheritance, encapsulation, polymorphism, abstraction.

**CODE:**



**OUTPUT:**



**ERROR TABLE:**

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

**CLASS DIAGRAM:**

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

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

|  |
| --- |
| Car |
| -no.of.doors:int |
| +int (brand, speed,  No.of doors);  +display deatails(); |

|  |
| --- |
| Bikes |
| -has\_gears:bool |
| +int (brand, speed,  has gears);  +display deatails(); |

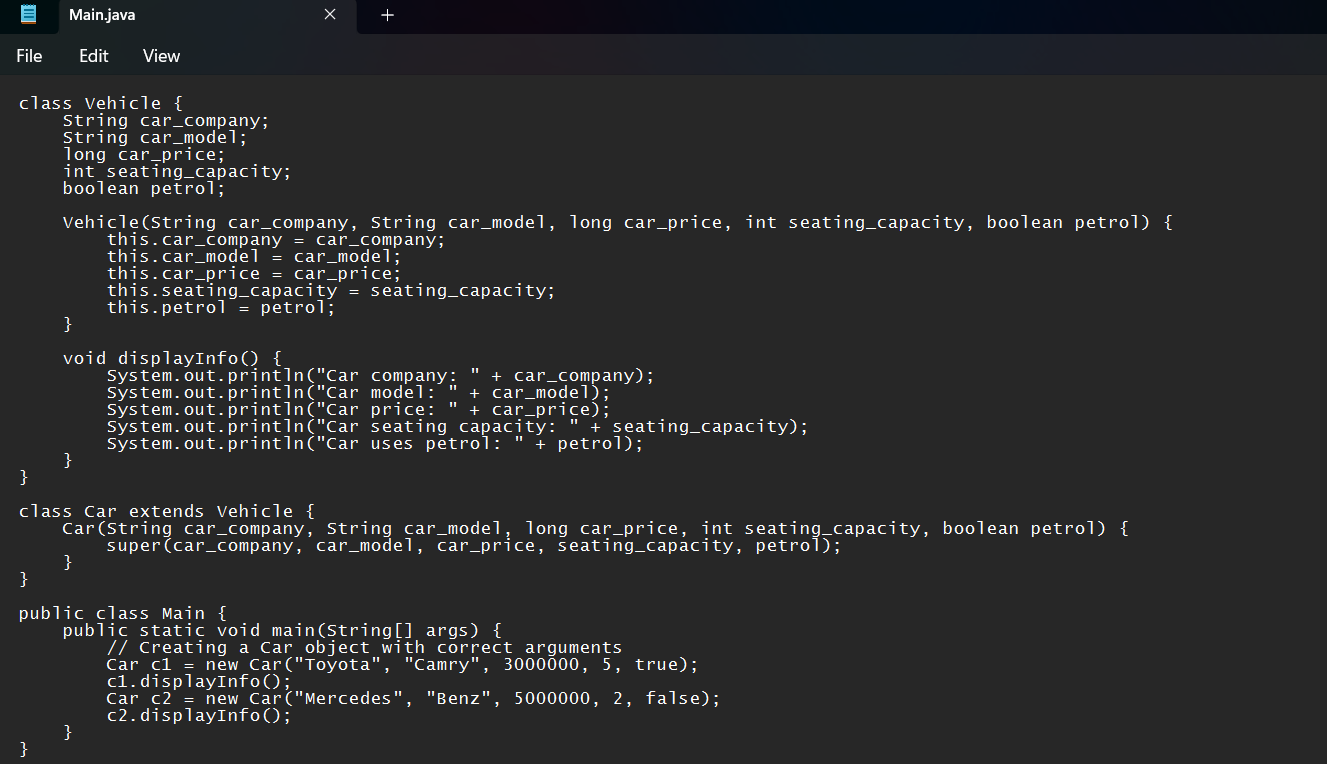
|  |
| --- |
| Truck |
| -Capacity:float |
| -Show truck detais();  +display deatails(); |

**WEEK-06**

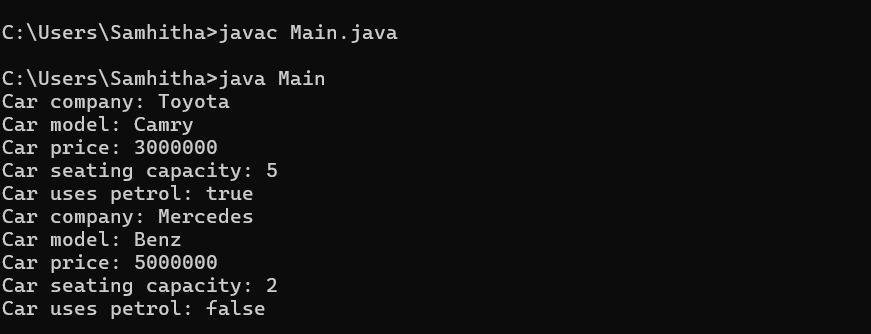
**PROGRAM-1:**

**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:**



**OUTPUT:**



**ERRORS:**

|  |  |
| --- | --- |
| **Code error** | **Code rectification** |
| 1. Incorrect class name for main method(Truck).  2. Inconsistent car model output in displayinfo(). | 1.Rename Truck to Main or place main inside car or vehicle.  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 * Speed: int |
| + vehicle(brand: string  Speed: int)  +start vehicle(): void  +displaydetails():void |

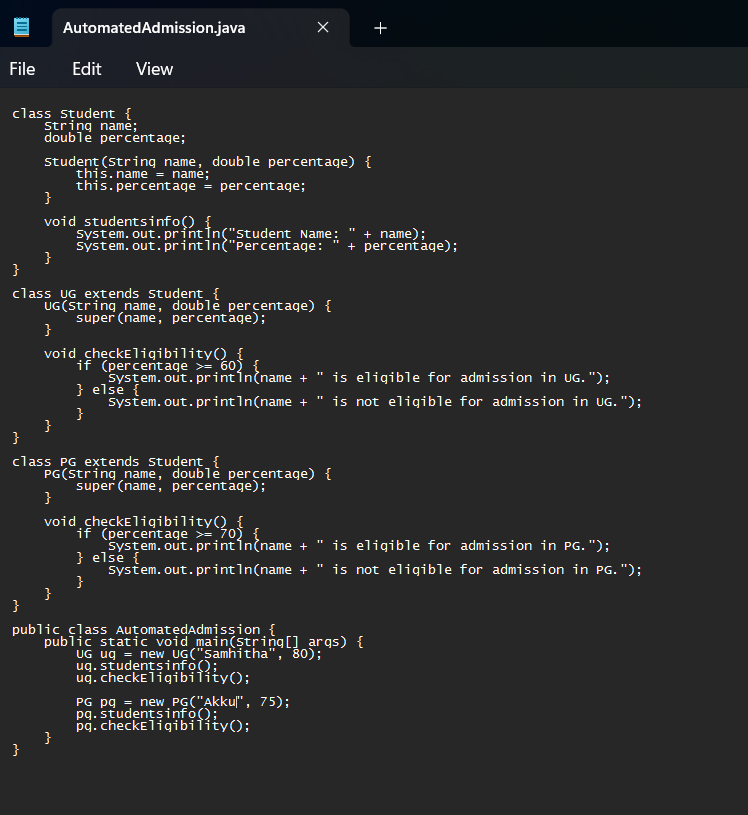
**PROGRAM-2:**

**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.

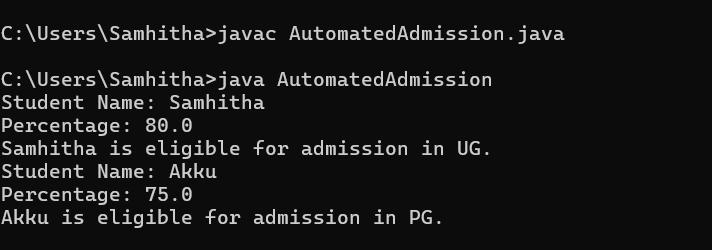
1. UG admission recquire a minimum of 60%.

2. PG admission recquire a minimum of 70%.

**CODE:**



**OUTPUT:**



**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.  **2.Program type input case sensitivity issue**: If the user enters ug or pg in lowercase, it may cause incorrect comparisons. | **1**.Add scanner.nextLine(); after nextDouble(); to consume the leftover newline.  **2.**Use program.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:**

|  |
| --- |
| **AutomatedAdmission** |
| * Scanner: scanner * Name: String * Percentage : double * Program: stirng |
| + main(args:String[]): void  +takeInput(): void  +checkEligibility(): void  +closeScanner(); void |

**PROGRAM-3:**

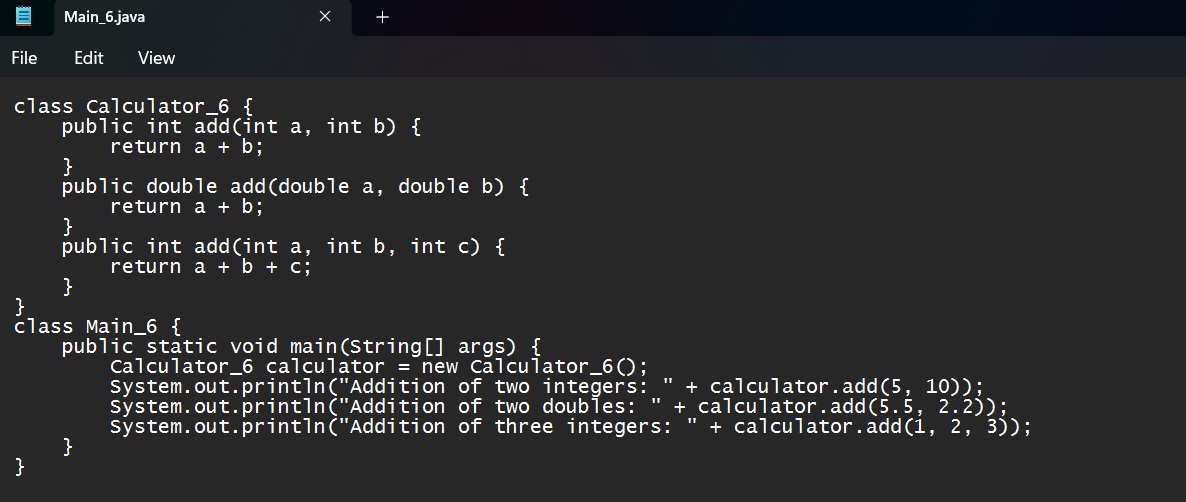
**AIM:** Create a calculator class with overloaded methods to perform addition of:

1. Add two integers

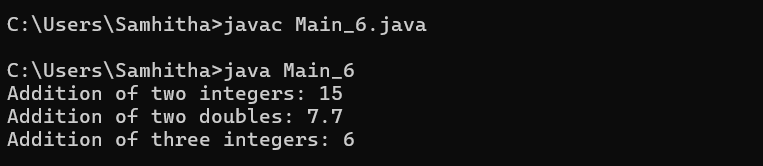
2. Add two doubles

3. Add three integers

**CODE:**



**OUTPUT:**

****

**ERRORS:**

|  |  |
| --- | --- |
| **Code error** | **Code rectification** |
| 1.Method parameters missing spaces. E.g.,”inta, intb”should be “int a, int b”  2.Inconsistent indentation in method bodies | 1**.** Add proper spacing between parameters: (int a, int b)  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.

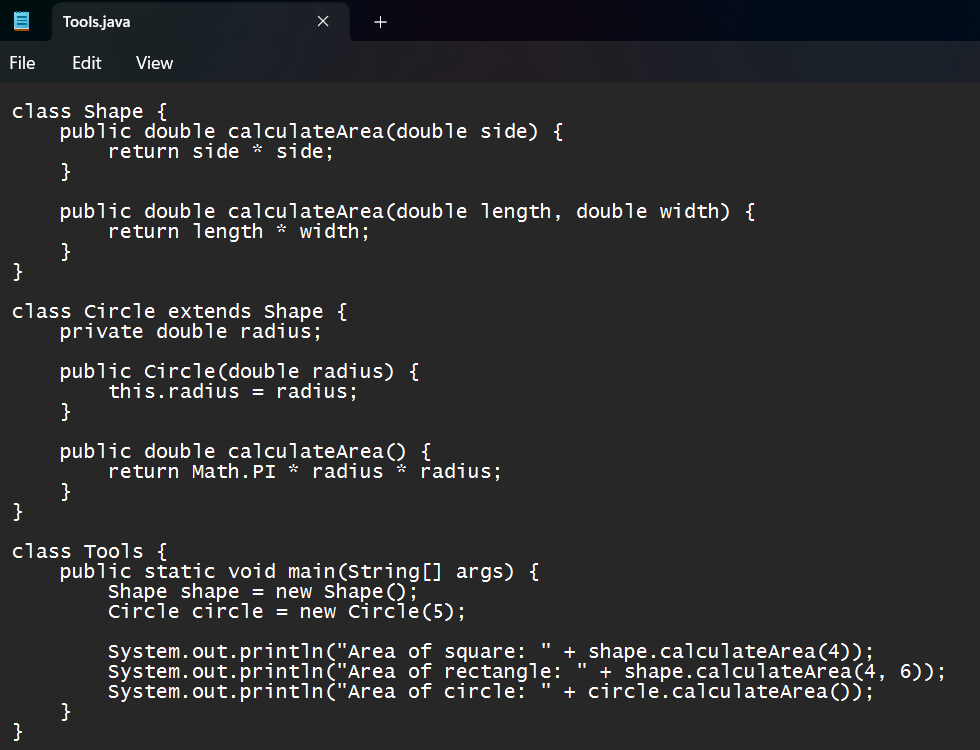
**CLASS DIAGRAM:**

|  |
| --- |
| **Calculator** |
| + add(int, int): int  +add(double, double): double  +add(int,int,int): int |

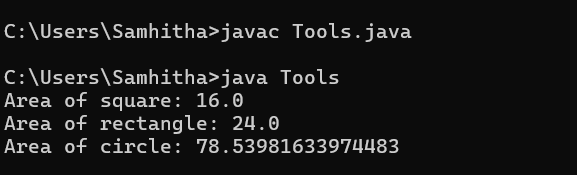
**PROGRAM-4:**

**AIM:** 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.

**CODE:**



**OUTPUT:**



**ERRORS:**

|  |  |
| --- | --- |
| **Code error** | **Code rectification** |
| 1. Method calls in main are missing an object reference (e.g., calculateArea(4) instead of s.calculateArea(4)).  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.  2. Ensure @Override is used, and the method signature should match correctly. |

**CLASS DIAGRAM:**

|  |
| --- |
| **SHAPE** |
| + CalculateArea(side:double): double  +CalculateArea(width: double, length: double): double |

|  |
| --- |
| **CIRCLE** |
| + CalculateArea(radius: double): double |

|  |
| --- |
| **Tools** |
| +main(args:String[]): Void |

**IMPORTANT POINTS:**

**1.Inheritance**: Circle class extends Shape, inheriting its methods.

**2.Method Overloading**: Shape has multiple calculateArea methods with different parameters.

**3.Method Overriding**: Circle overrides calculateArea from Shape to implement its own formula.

**4.Polymorphism**: The overridden method in Circle demonstrates runtime polymorphism.

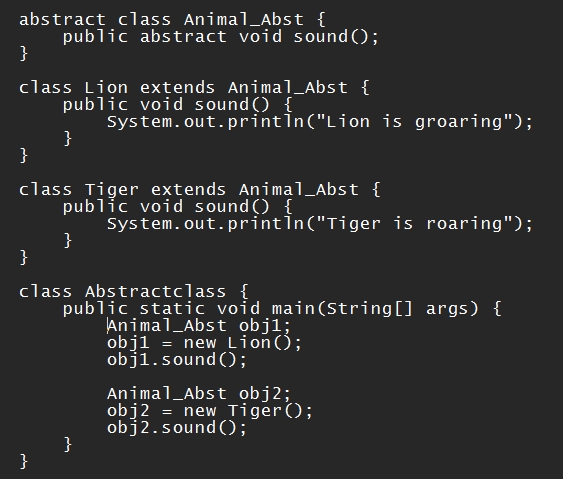
**5.Proper Object Reference**: Methods should be called using an object (s.calculateArea(4), c.calculateArea(2)).

**WEEK-07**

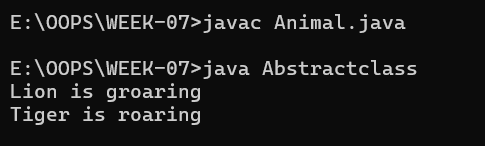
**PROGRAM-1:**

**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:**



**OUTPUT:**



**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.  2 Not overriding the sound() method in subclasses. | 1.Rectified as abstract void sound();  2. Added void sound() { ... } in each subclass. |

**CLASS DIAGRAM:**

|  |
| --- |
| **<<abstract>>**  **ANIMAL** |
| + sound(): void |

|  |
| --- |
| **TIGER** |
| + sound(): void |

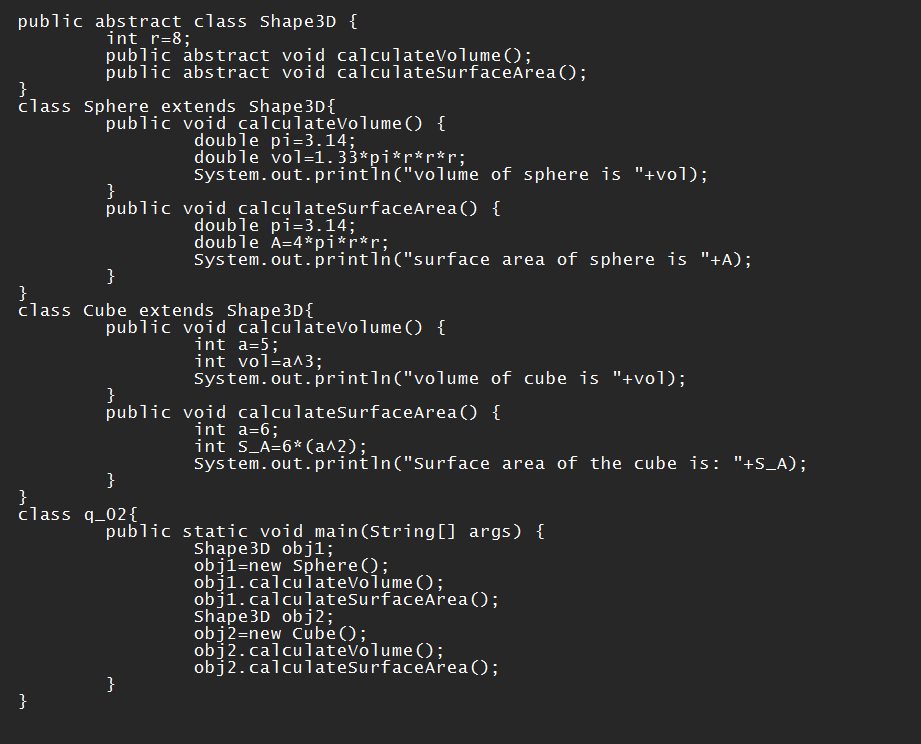
|  |
| --- |
| **LION** |
| + sound(): void |

|  |
| --- |
| **Animalsound** |
| +main(args:String[]): Void |

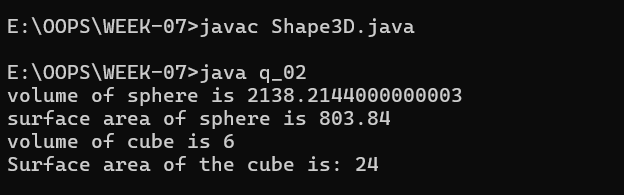
**PROGRAM-2:**

**AIM :** 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.

**CODE:**



**OUTPUT:**



**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 | | |  |  |  |  |  | | --- | --- | | 1. (4 / 3) used instead of (4.0 / 3.0) | | |  |  |  |  | | --- | | 1. 3.14 used as approximation for π |  |  | | --- | |  |  |  | | --- | |  | | 1. Changed return types of calculateVolume() and calculateSurface() to double 2. Used floating-point division to avoid integer division loss. 3. 3.Used Math.PI for more accurate calculations. |

**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

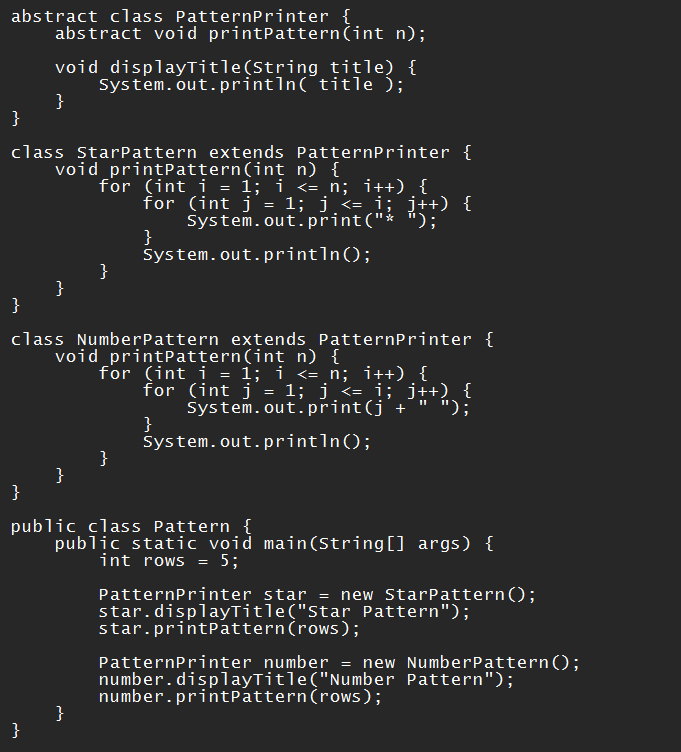
Create two subclasses:

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

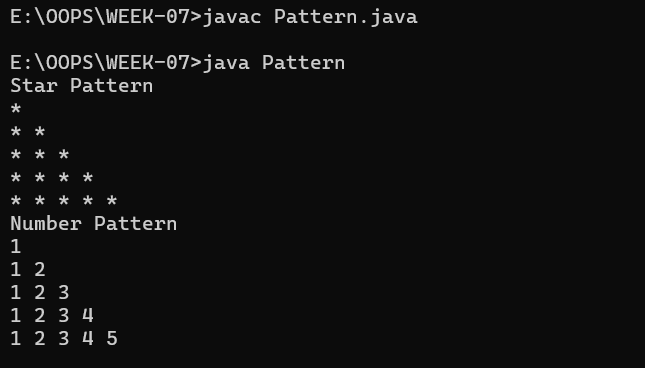
2.NumberPattern: Prints a right-angled triangle of increasing numbers

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

**CODE:**



**OUTPUT:**



**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). | | |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | | |  |  |  | | --- | --- | --- | | |  | | --- | | 2.displayTitle method not used before pattern printing |  |  | | --- | | 3.Forgot to implement printPattern(int n) in subclass | | | | |  |  |  |  | | --- | |  |  |  | | --- | |  | | 1.Use nested loops: outer loop for rows, inner loop for printing symbols or numbers.  2.Call displayTitle() before printing the pattern for proper formatting  3.Implemented the method in both subclasses |

**CLASS DIAGRAM:**

|  |
| --- |
| **<<abstract>> PatternPrinter** |
| +displayTitle(title)  +printPattern(n) |

|  |
| --- |
| **StarPattern** |
| +Printpattern (n) |

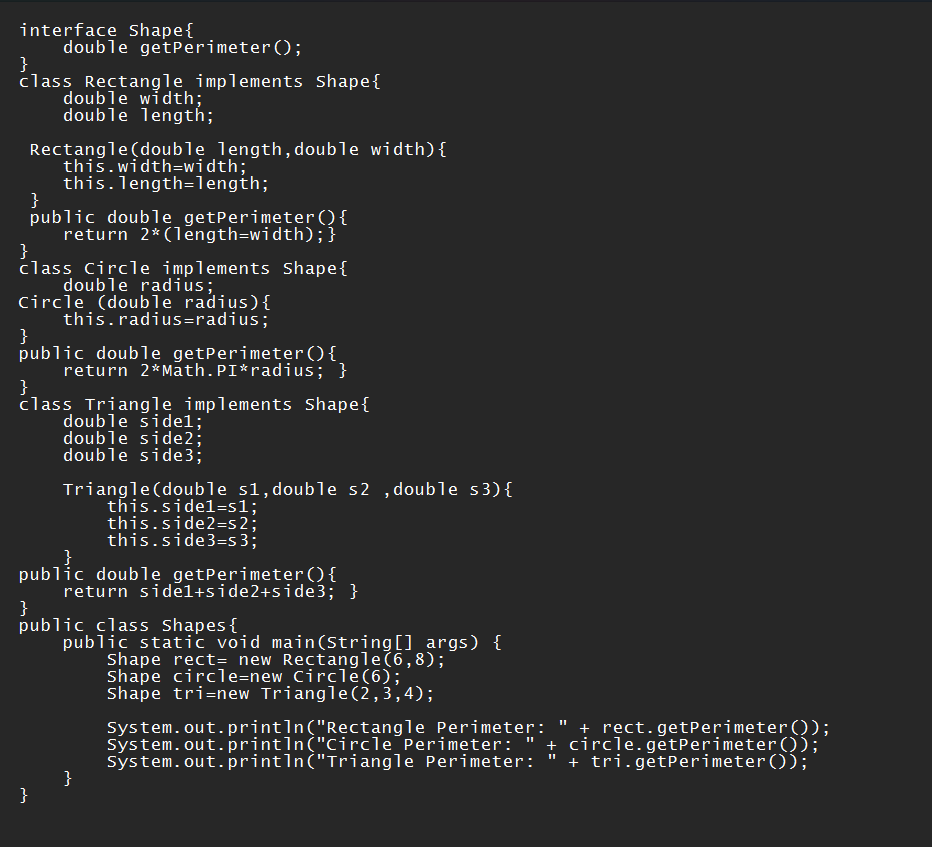
|  |
| --- |
| **NumberPattern** |
| +Printpattern (n) |

**WEEK-08**

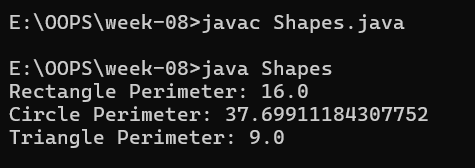
**PROGRAM-1:**

**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:**



**OUTPUT:**



**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** | 1. **Code rectification** |
| 1. Missed Math.PI in Circle perimeter,  2.Did not give meaningful class name. | 1. Rectified to Math.PI in Circle perimeter. 2. Rectified |

**CLASS DIAGRAM:**

|  |
| --- |
| **<<interface >> SHAPE** |
| + getPerimeter(): double |

|  |
| --- |
| **Circle** |
| - radius: double |
| + getPerimeter(): double |

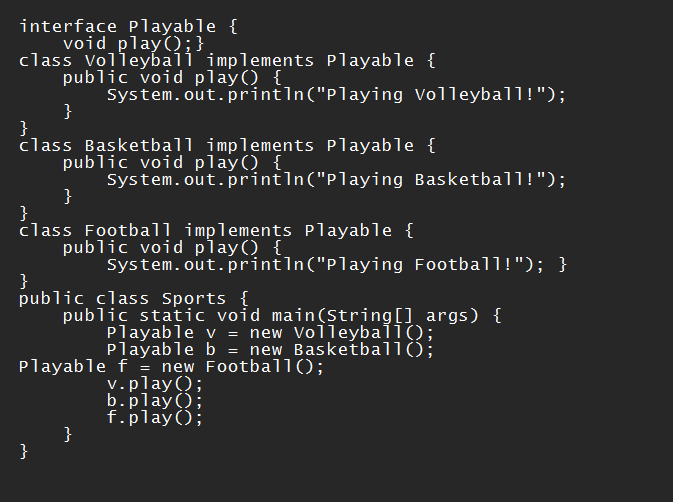
|  |
| --- |
| **Rectangle** |
| - length: double  - width: double |
| + getPerimeter(): double |

|  |
| --- |
| **Triangle** |
| - side1: double  - side 2: double  - side 3: double |
| + getPerimeter(): double |

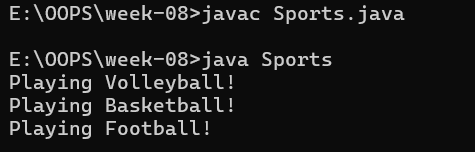
**PROGRAM-2:**

**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:**



**OUTPUT:**



**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);  2. forgot public in Method play()   |  | | --- | |  | | 1.Rectified to v.play();  2.used public void play() in all implementing classes |

**CLASS DIAGRAM:**

|  |
| --- |
| **<<interface>>**  **Playable** |
| + play(): void |

|  |
| --- |
| **Basketball** |
| + play(): void |

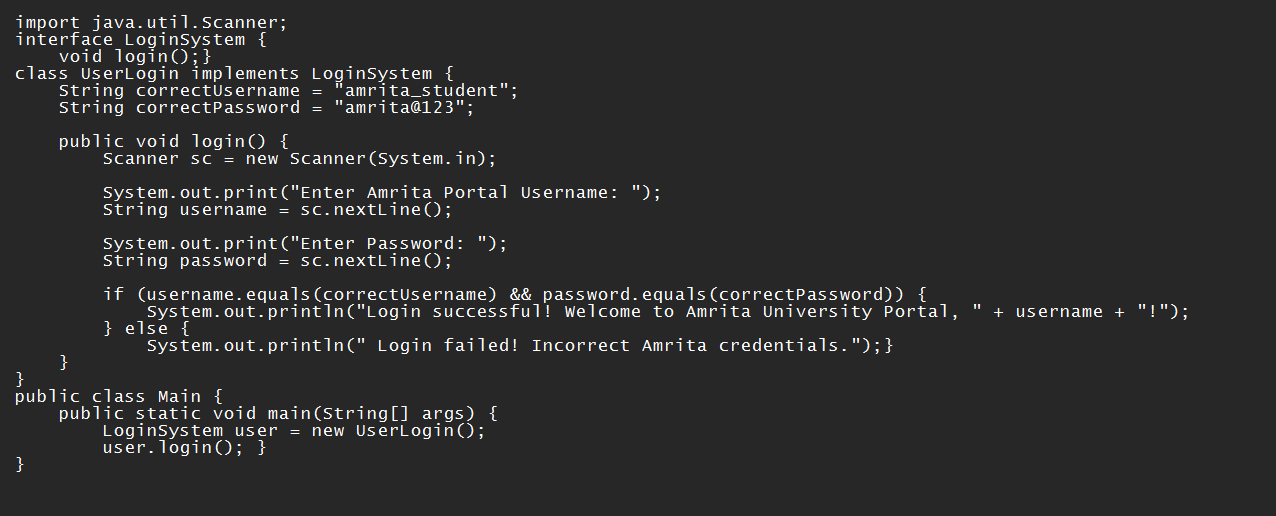
|  |
| --- |
| **Volleyball** |
| + play(): void |

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

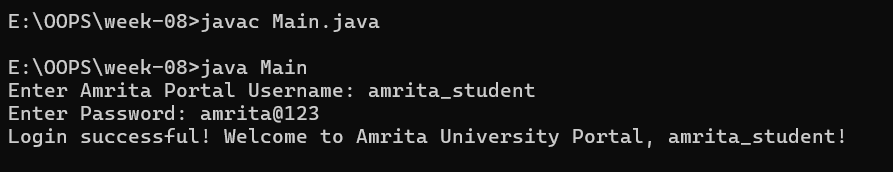
**PROGRAM-3:**

**AIM:** Write a java program to create a login system using interface.

**CODE:**



**OUTPUT:**



**IMPORTANT POINTS:**

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  2. Didn't import java.util.Scanner.  3. Called user.login() with parameters   |  | | --- | |  | | 1. Used username.equals(correctUsername) not ==  2. Added import java.util.Scanner;  3. Correct is user.login(); with no parameters |

|  |
| --- |
| **<<interface>>**  **LoginSystem** |
| + login(): void |

**CLASS DIAGRAM:**

|  |
| --- |
| **UserLogin** |
| - correctUsername: String  - correctPassword: String |

**WEEK-9**

**PROGRAM-1**

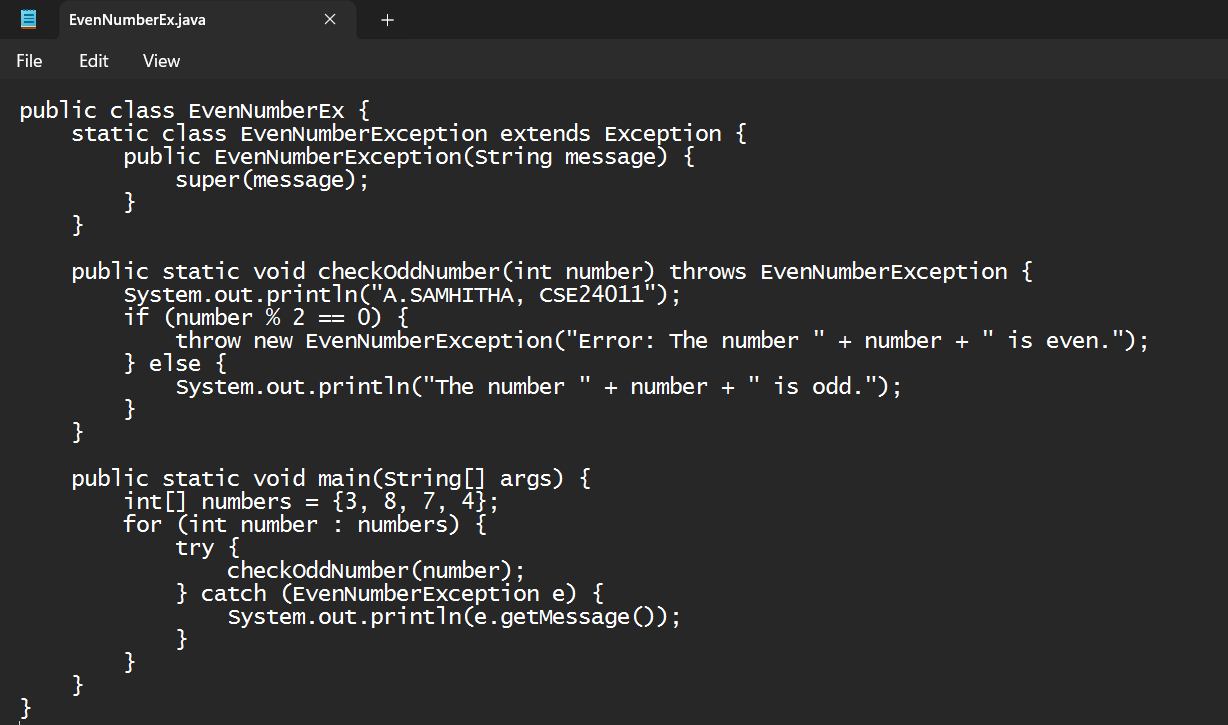
**AIM:** Write a java program to create a method that takes integers as parameters and throws an exception if the number is even.

**CLASS DIAGRAM:**

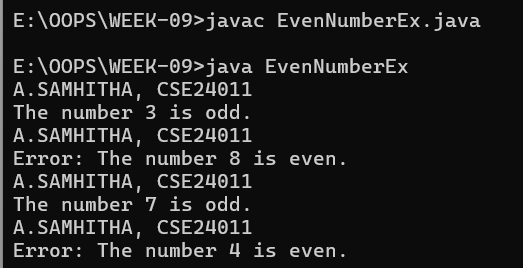
|  |
| --- |
| **EvenNumberExceptionDemo** |
| +checkoddNumber(int number): void+main(String[]args):void |

|  |
| --- |
| **EvenNumberException**  (extends Exception) |
| + EvenNumberException() |

**CODE:**

****

**OUTPUT:**

****

**ERRORS:**

|  |  |
| --- | --- |
| **Code Error** | **Code Rectification** |
| 1. Unhandeled exception type EvennumberException. 2. Syntax errors 3. Compilation error. | 1. If you checkoddNumber() without using try-catch or without declaring throws. 2. If missing curly barces or wrong method syntax. 3. If constructor of EvenNumberException is missing or incorrectly defined. |

**IMPORTANT POINTS:**

1) Created a custom exception by extending the Exception class.

2) Used throw keyword to manually throw the custom exception if the number is even.

3)Handled the exception using a try-catch block inside main() method.

4) Demonstrates user-defined exception handling.

5) Shows clear separation of concerns: checking number and exception message.

**PROGRAM-2**

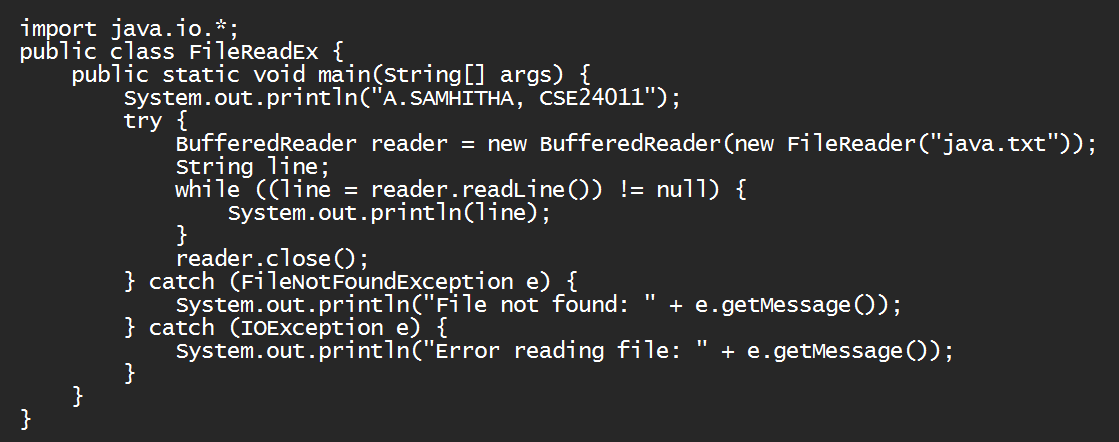
**AIM:** Write a java program to create a method that reads a file and throws an exception if the file is not found.

**CLASS DIAGRAM:**

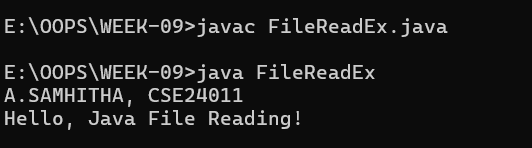
|  |
| --- |
| **FileReadExample** |
| + main(String[] args) : void |

|  |
| --- |
| **Uses** |
| -BuefferdReader  -FileReader  -FileNotFound  -IOException |

**CODE:**



**OUTPUT:**

****

**ERRORS:**

|  |  |
| --- | --- |
| **Code Error** | **Code Rectification** |
| 1. File Not Found 2. IOException 3. Syntax Error | 1. Occurs if the specified fiole path is wrong or file does not exist. 2. Occurs while reading file if an input/output error happens. 3. If missing semicolon; wronf try-catch syntax |

**IMPORTANT POINTS:**

1. Used BufferedReader and FileReader to read text files.
2. FileNotFoundException occurs if the file is missing.
3. IOException occurs for input/output errors during file reading.
4. try-catch block is used for proper exception handling.
5. Always close the reader after reading the file (reader.close()).

**PROGRAM-3**

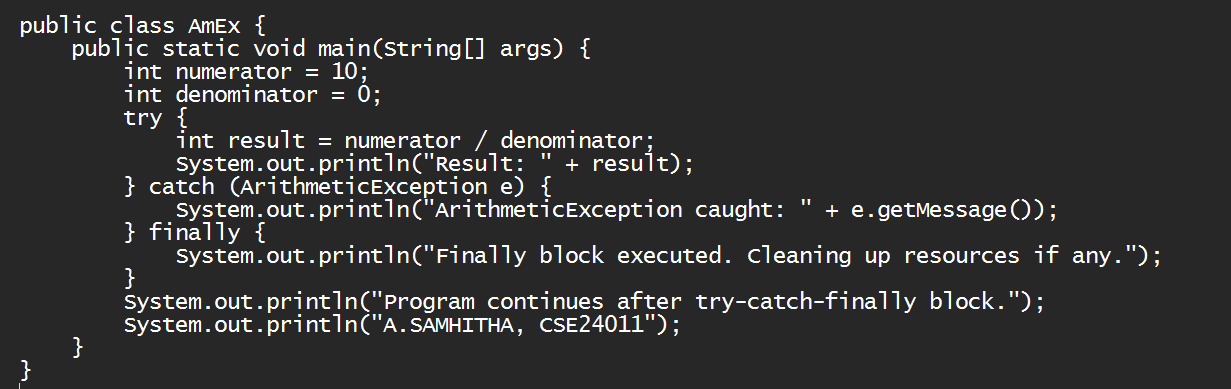
**AIM:** Write a java program to handle arthimatic exception using try, catch and finally.

**CLASS DIAGRAM:**

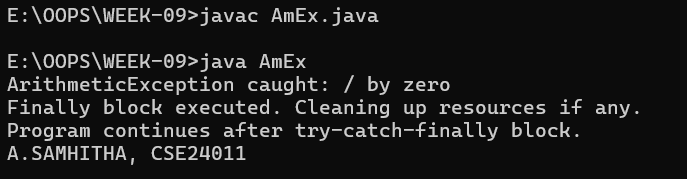
|  |
| --- |
| **FileReadExample** |
| +main(String[]args): void |
|  |

|  |
| --- |
| **Uses** |
| **-**BuefferdReader  -FileReader  -FileNotFound  -IOException |

**CODE:**



**OUTPUT:**



**ERRORS:**

|  |  |
| --- | --- |
| **Code Error** | **Code Rectification** |
| 1. File not found 2. IOException 3. Syntax error | 1. Occurs if the specified file path is wrong or file does not exist 2. Occurs while reading file if an input/output error happens 3. If missing semicolon ; wrong try-catch block syntax |

**IMPORTANT POINTS:**

1. Used BufferedReader and FileReader to read text files.
2. FileNotFoundException occurs if the file is missing.
3. IOException occurs for input/output errors during file reading.
4. try-catch block is used for proper exception handling.
5. Always close the reader after reading the file (reader.close()).

**PROGRAM-4**

**AIM:** Write a program to stimulate a university system using inner classes.

* Create an outer class named university with a variable universityName.
* Inside it, define two non-static inner classes.

1. Department- With variables like deptName, deptCode and a method to display department details.
2. Student- With variables like studentName, rollNumber and a method to display details.
3. Create an object for each class and call their methods to display their details along with their universityName.

**CLASS DIAGRAM:**

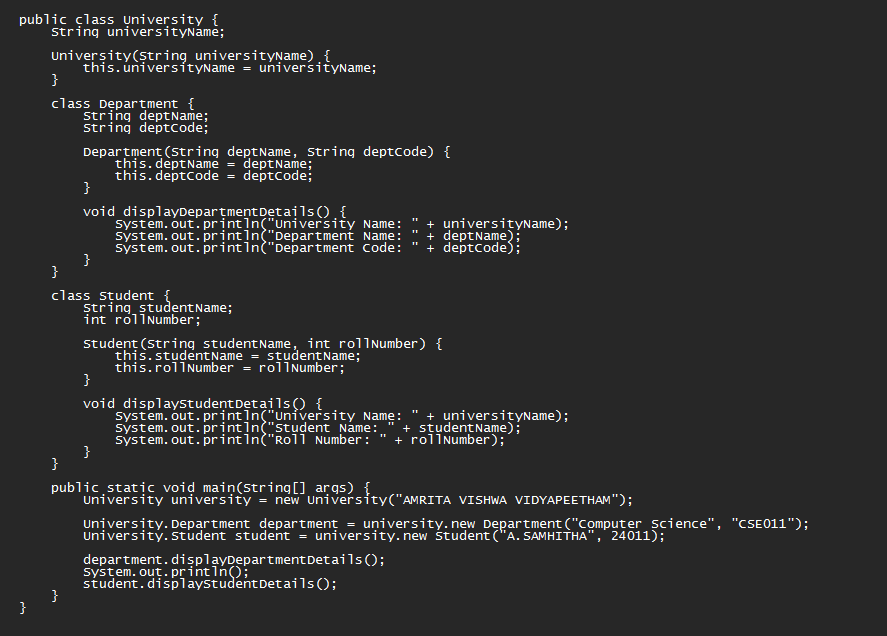
|  |
| --- |
| **University** |
| -universityName: String  + University(String name) |

|  |
| --- |
| **Innerclass** |
| - Department - deptName: String - deptCode: String  + displayDepartment(): void |

|  |
| --- |
| - Student  - studentName: String  - rollNumber: int + displayStudent(): void |

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

**CODE:**



**OUTPUT:**



**ERRORS:**

|  |  |
| --- | --- |
| **Code Error** | **Code Rectification** |
| 1. Syntax Error 2. Compilation Error 3. Runtime Error | 1.Wrong object creation for inner class  2.Accessing outerclass members wrongly  3.Null pointerException if outer object missing |

**IMPORTANT POINTS:**

1. Demonstrate inner class usage.
2. Inner classes access outer class members easily.
3. Separate objects for Department and Student.Good example of encapsulation

**WEEK-10**

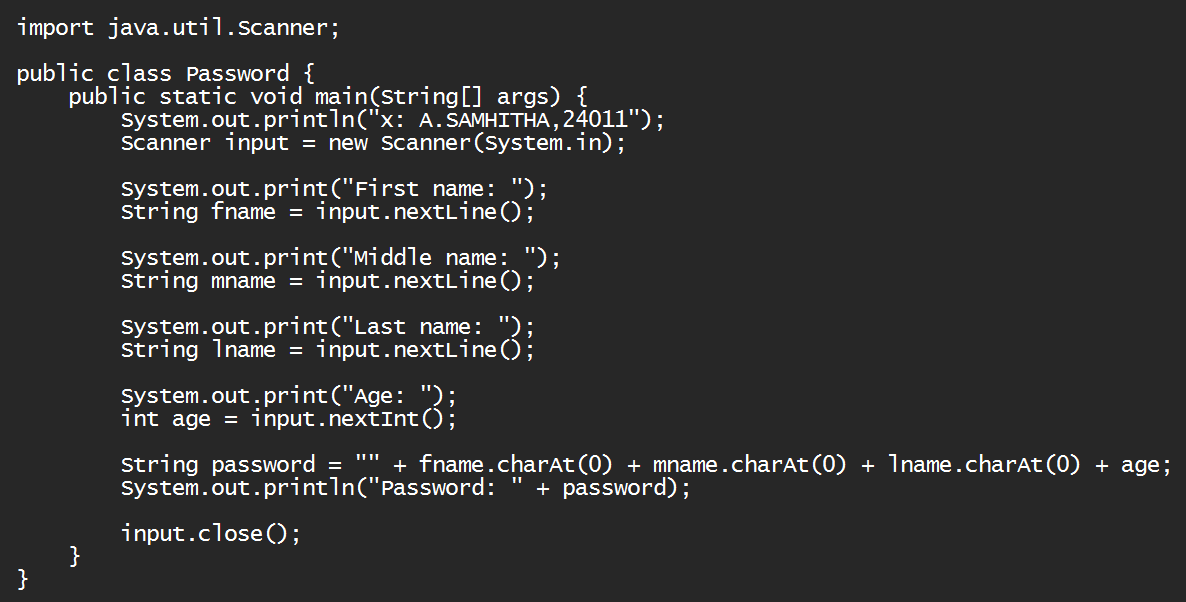
**PROGRAM-1:**

**AIM:** Write a java program to generate a password for a student using his/her initials and age. The password displayed should be the string consists of first character of first name, middle name, last name with age.

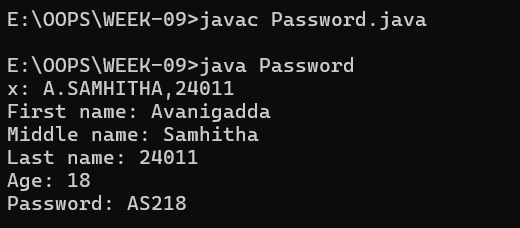
**CLASS DIAGRAM:**

|  |
| --- |
| **Password** |
| +first name: String  +middle name: String  +last name: String  +age: int |
| +password(): String |
| +main(String[]args): void |

**CODE:**

****

**OUTPUT:**



**ERRORS:**

|  |  |
| --- | --- |
| **Code Error** | **Code Rectification** |
| 1. Identifier expected after the token | Give a suitable identifier that will give the output |

**IMPORTANT POINTS:**

1. The program takes the first, middle, lastnames and age as input.
2. It creates a password using the first letter of each name plus the age.
3. The logic is organized using a class and method.
4. It uses the Scanner class to read input from the user.
5. The program follows object-oriented principles with clean structure and reusability.

**PROGRAM-2:**

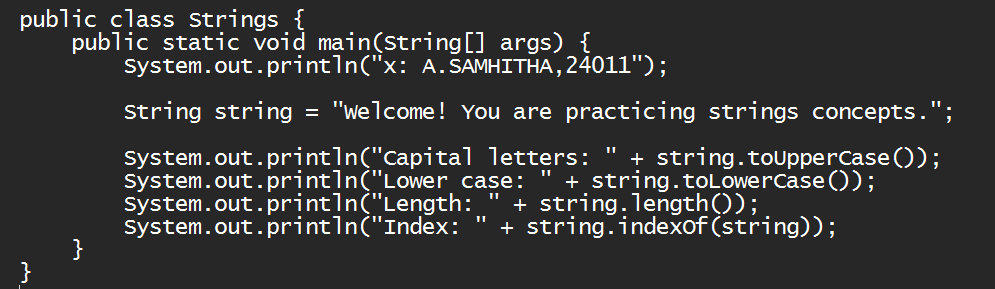
**AIM:** Design and implement a java program that will do the folleing operations to this string “Welcome! You are practicing String concepts”.

* Convert all alphabets to capital letters and print out the result.
* Convert all alphabets to lowercase letters and print out the result.
* Print out the length of the string.
* Print out the index of the concept.

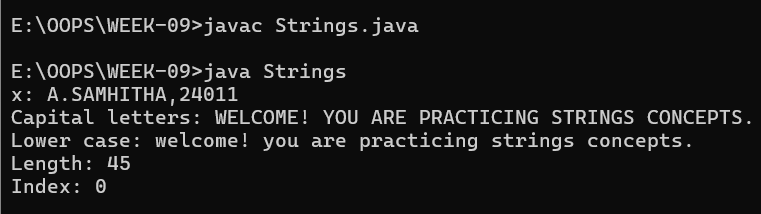
**CLASS DIAGRAM:**

|  |
| --- |
| **Strings** |
| +string: public  +string.uppercase()  +string.lowercase()  +string.length()  +string.index() |
| +void main(String[]args): void |

**CODE:**



**OUTPUT:**

****

**ERRORS:**

|  |  |
| --- | --- |
| **Code Error** | **Code Rectification** |
| 1. ToUpper is undefined for the type string. | * + - 1. Type toUpperCase() instead of to uppercase() |

**IMPORTANT POINTS:**

1. The program defines a string message and displays it.
2. It uses touppercase() and tolowercase() to change the case of the string.
3. Length() is used to find the length of the string.
4. Indexof() returns 0, because the string is searching for itself at index().

**PROGRAM-3:**

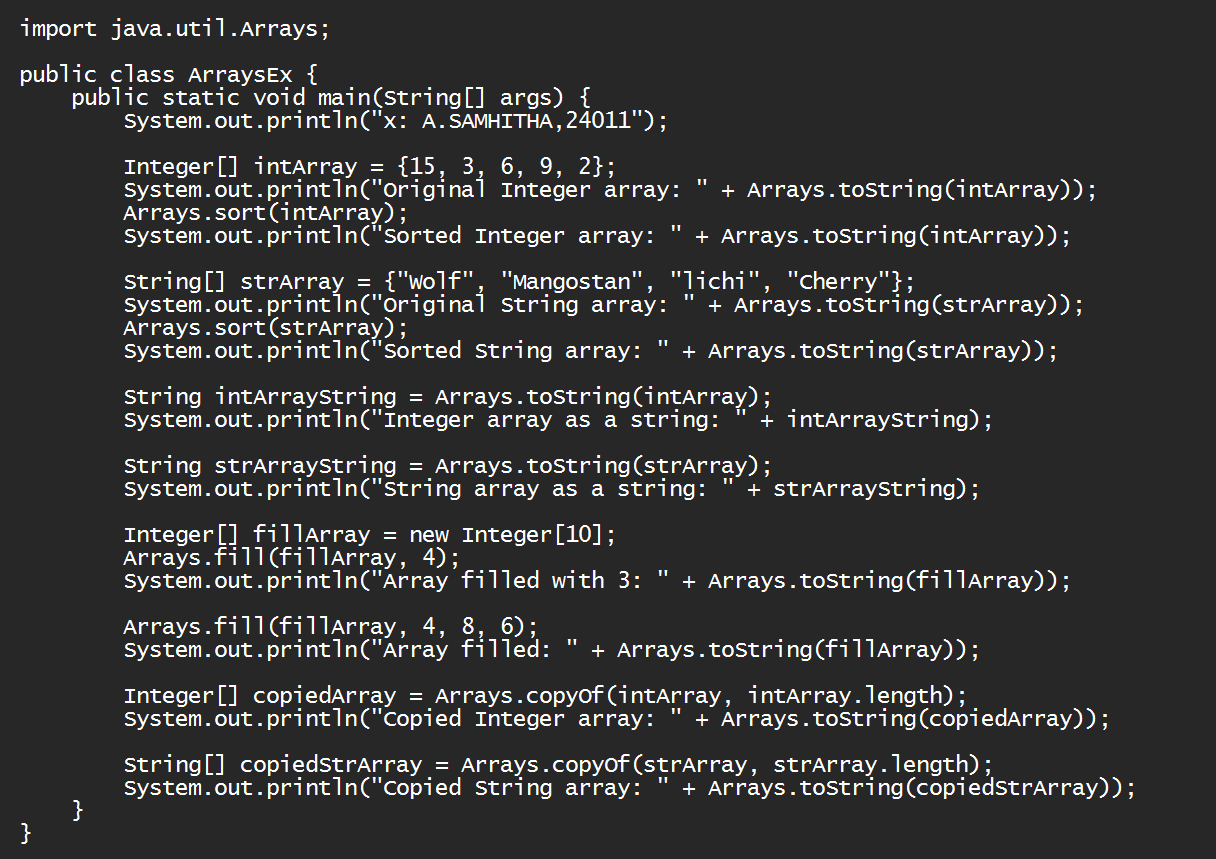
**AIM**: Implement a java program using the below array methods.

* Sorting the elements (numbers & strings) of an array.
* Convert the array elements into String.
* Fill the part of an array.
* Copy the elementsof one array into the other.

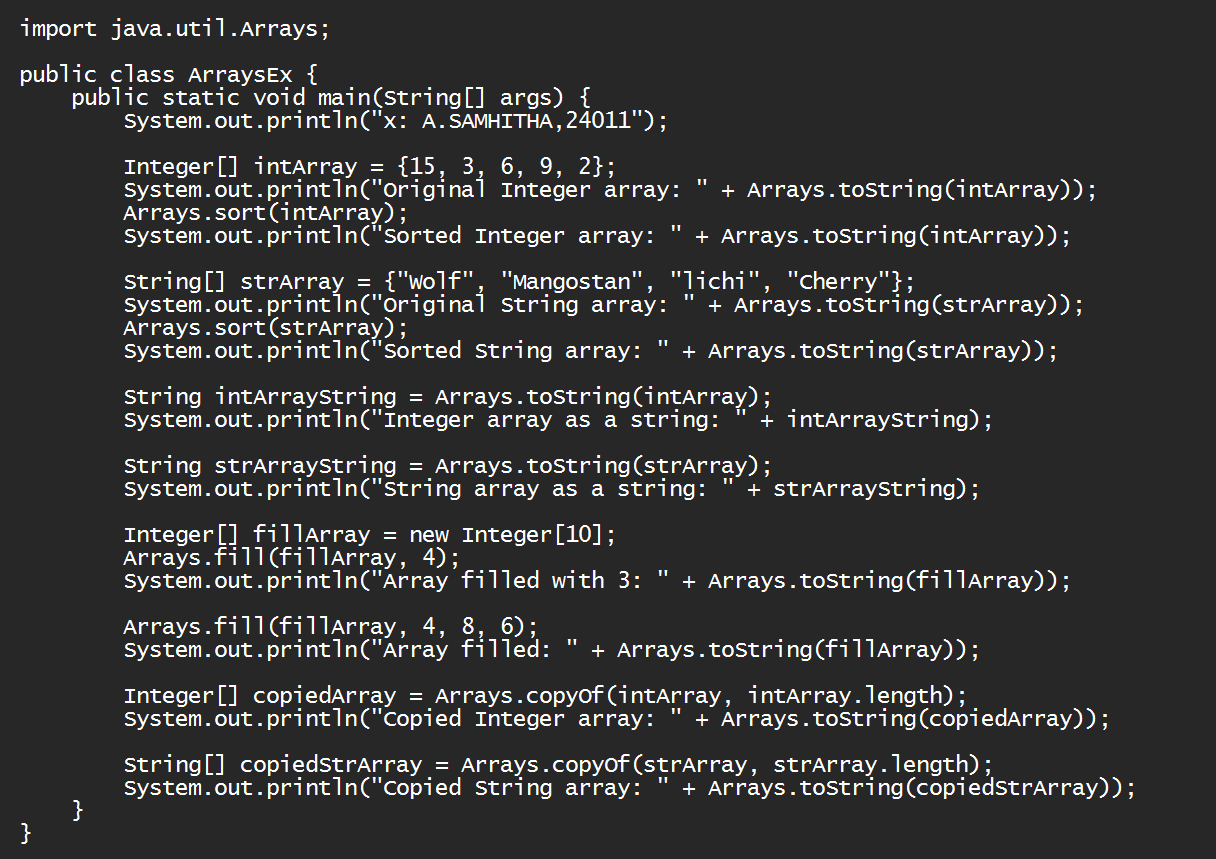
**CLASS DIAGRAM:**

|  |
| --- |
| **Arrays** |
| +sort(arr: T[]): void  +toString(arr: T[]): String  +fill(arr: T[], value: T): void  +fill(arr: T[], fromindex: int, toindex: int, value: T): void  +copyof(arr: T[], newLength: int): T[] |

**CODE:**



**OUTPUT:**

****

**ERRORS:**

|  |  |
| --- | --- |
| **Code Error** | **Code Rectification** |
| 1. 1. The method toString() in the object is not applicable for the arguments(integer[]) | * + - 1. Declare Array instead of array. |

**IMPORTANT POINTS:**

1. The code demonstrates sorting arrays to both integers and strings using Arrays.sort().
2. It converts arrays to string representations with Arrays.toString().
3. It uses Arrays.fill() to fill entire arrays or parts of arrays with specific values.
4. The code copies arrays using Arrays.copyof() to create new arrays with the same elements.

**PROGRAM-4:**

**AIM:** Implement a java program using the below Array list method.

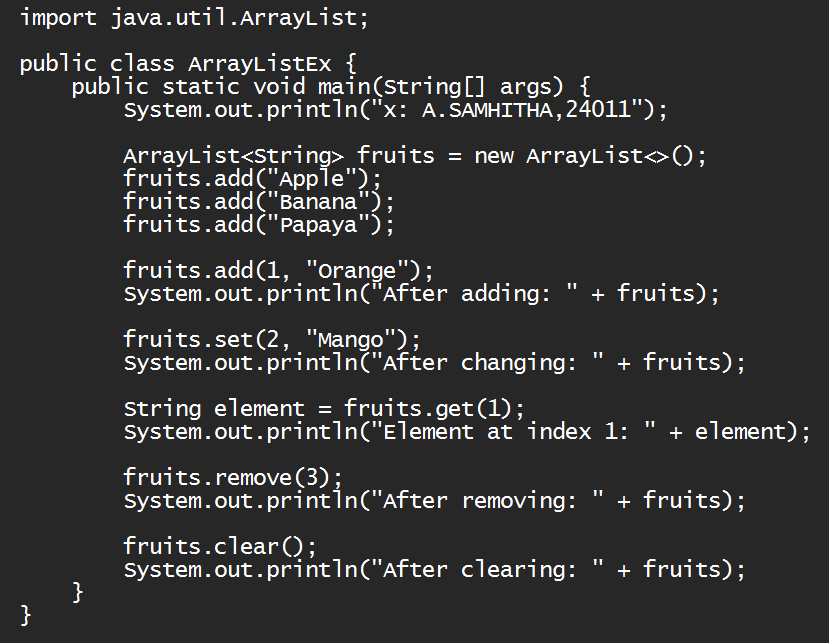
* Insert an element at particular index in the array list.
* Modify an element in the array list.
* Access an element from the array list.
* Remove an element from the Array list.
* Clear the elements from the array list.

**CLASS DIAGRAM:**

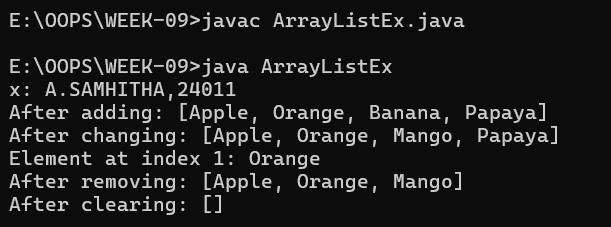
|  |
| --- |
| **ArrayListExamaple** |
| +fruits: ArrayList<String>  +main(args: String[]): void |

|  |
| --- |
| **ArrayList<T>** |
| +add(E e): Boolean  +add(int index, E element): void  +set(int index, E element): E  +get(int index): E  +remove(int index): E  +clear(): void |

**CODE:**



**OUTPUT:**

**ERRORS:**

|  |  |
| --- | --- |
| **Code Error** | **Code Rectification** |
| 1. 1. Array list not defined 2. 2. Exception main.java.lang.error; | * + - 1. Define the array list class.   2. Insert() to computer class instance creation expression. |

**IMPORTANT POINTS:**

1. The program demonstrates how to create and manipulate an arraylist in java.
2. Elements are instead at specific positons using the add(index, element) method.
3. The set(index, element) method is used to modify existing elements.
4. The entire listcan be cleared using the clear() method to reamove all elements.