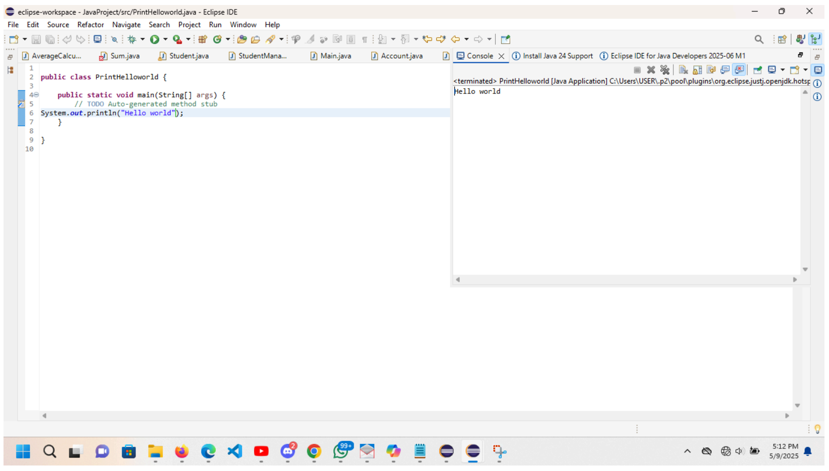
Name Ezeora Izuchukwu Micheal
Reg No 2023/255306
Course Cos 261

1.Write a java program Hello, world”


eclipse-workspace - JavaProject/src/PrintHelloworld.java - Eclipse IDE

```java

public class PrintHelloworld {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
System.out.println("Hello world");
    }

}

```

<terminated> PrintHelloworld [Java Application] C:\Users\USER\.p2\pool\plugins\org.eclipse.justj.openjdk.hotsp

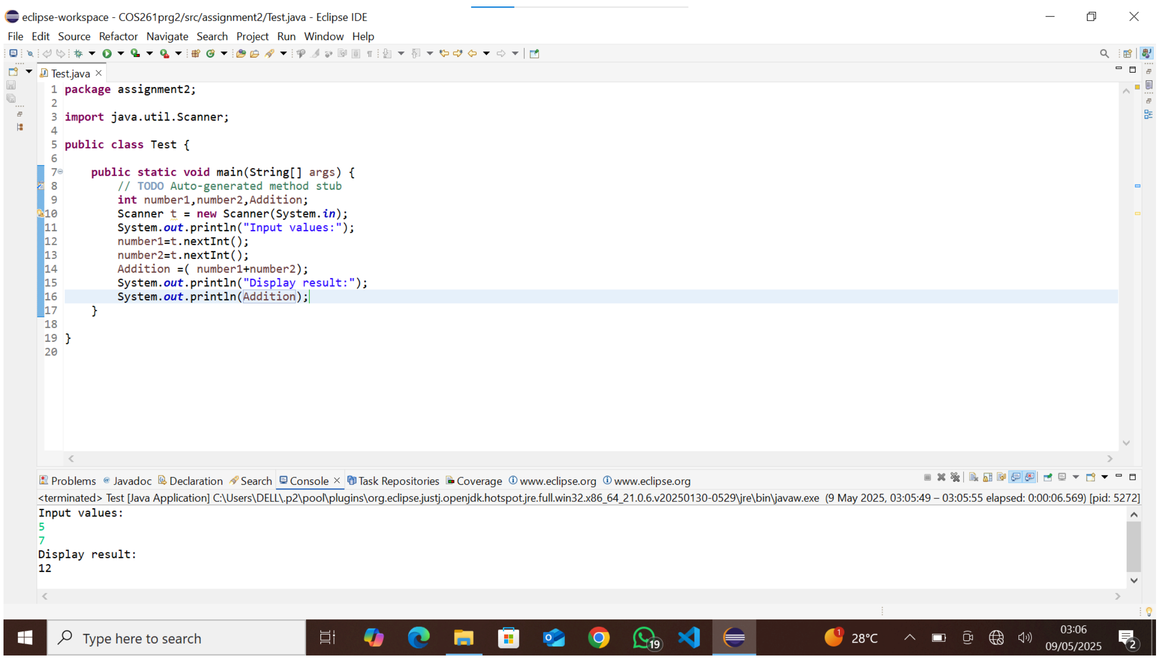
```
Hello world
```

2 **Explain The difference between== and .equal() in java.show with code examples and outputs.**

In Java, == and .equals() are used to compare objects, but they have different behaviors:

== Operator
The == operator checks for reference equality, meaning it compares the memory locations of two objects. If both objects point to the same memory location, it returns true.

.equals() Method
The .equals() method checks for content equality, meaning it compares the actual values of two objects. By default, .equals() behaves like ==, but many classes override this method to compare the contents of objects.


eclipse-workspace - COS261prg2/src/assignment2/Test.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Test.java
Problems Javadoc Declaration Search Console Task Repositories Coverage www.eclipse.org www.eclipse.org
Type here to search
28°C
03:06
09/05/2025

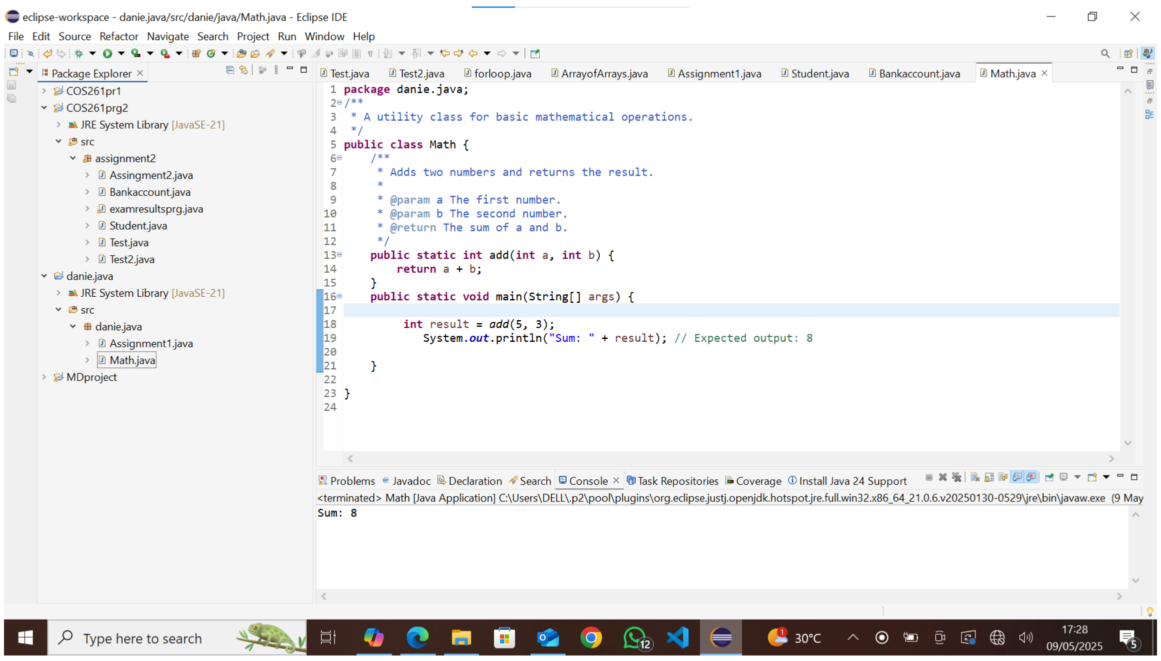
```java
package assignment2;

import java.util.Scanner;

public class Test {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		int number1,number2,Addition;
		Scanner t = new Scanner(System.in);
		System.out.println("Input values:");
		number1=t.nextInt();
		number2=t.nextInt();
		Addition =( number1+number2);
		System.out.println("Display result:");
		System.out.println(Addition);
	}

}
```

```
<terminated> Test [Java Application] C:\Users\DELL\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.6.v20250130-0529\jre\bin\javaw.exe  (9 May 2025, 03:05:49 – 03:05:55 elapsed: 0:00:06.569) [pid: 5272]
Input values:
5
7
Display result:
12
```

3 **What is the use if main method in java:**
The main method in Java is the entry point of a Java application. It's where the Java Virtual Machine (JVM) starts executing the program.

Key Uses of the Main Method
1. Entry Point: The main method is the starting point of a Java application.
2. Program Execution: The JVM calls the main method to begin executing the program.
3. Command-Line Arguments: The main method can accept command-line arguments through the String[] args parameter.

4.Write a Java program to add two numbers entered by the user.



```java
package danie.java;
/**
 * A utility class for basic mathematical operations.
 */
public class Math {
    /**
     * Adds two numbers and returns the result.
     *
     * @param a The first number.
     * @param b The second number.
     * @return The sum of a and b.
     */
    public static int add(int a, int b) {
        return a + b;
    }
    public static void main(String[] args) {

         int result = add(5, 3);
            System.out.println("Sum: " + result); // Expected output: 8

    }

}
```

```
<terminated> Math [Java Application] C:\Users\DELL\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.6.v20250130-0529\jre\bin\javaw.exe (9 May
Sum: 8
```

5.What is the difference between int,Integer,and String?

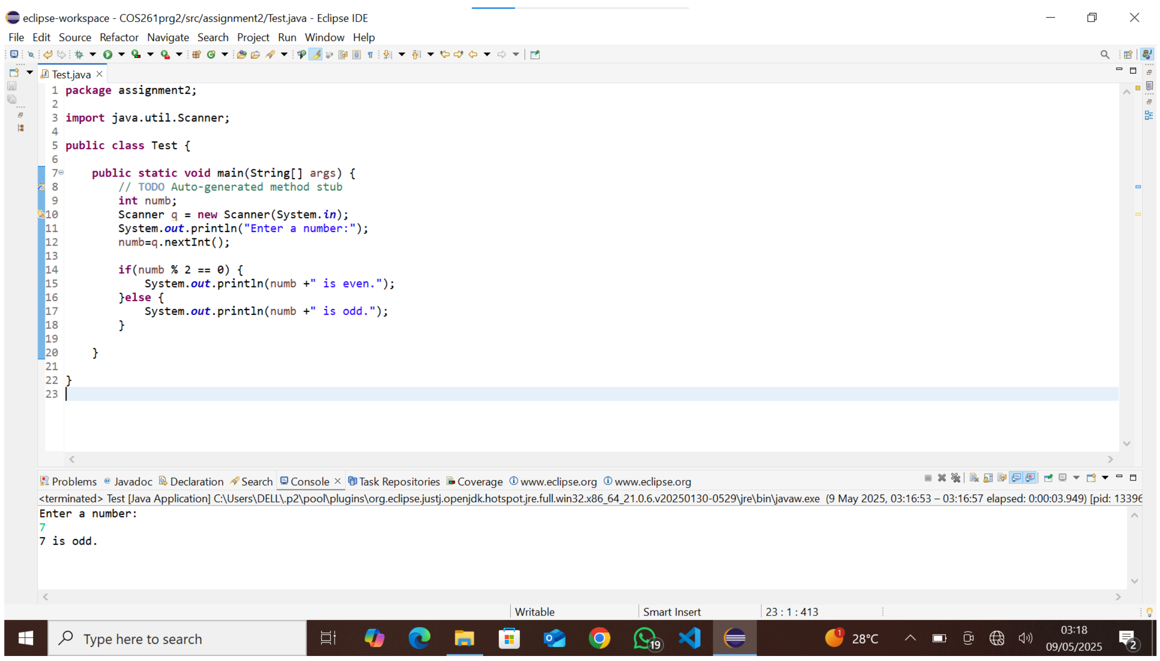
Here's a breakdown of the differences between int, Integer, and String:

int
1. Primitive Type: int is a primitive data type in Java.
2. Whole Number: int represents a whole number, like 1, 2, 3, etc.
3. Fixed Size: int is a 32-bit signed integer.

Integer
1. Wrapper Class: Integer is a wrapper class for the `int` primitive type.
2. Object: `Integer` is an object that wraps an int value.
3. Additional Methods: Integer provides additional methods, like parseInt()` and valueOf().

String
1. Sequence of Characters: String represents a sequence of characters, like "hello" or "world".
2. Object: String is an object in Java.
3. Immutability: String objects are immutable, meaning their contents cannot be changed after creation.
4. Type: int is a primitive type, while Integer and String are objects.
5. Purpose: int and Integer are used for numerical values, while String is used for text.
6. Behavior: int and Integer have different behaviors due to their primitive and object nature, respectively.

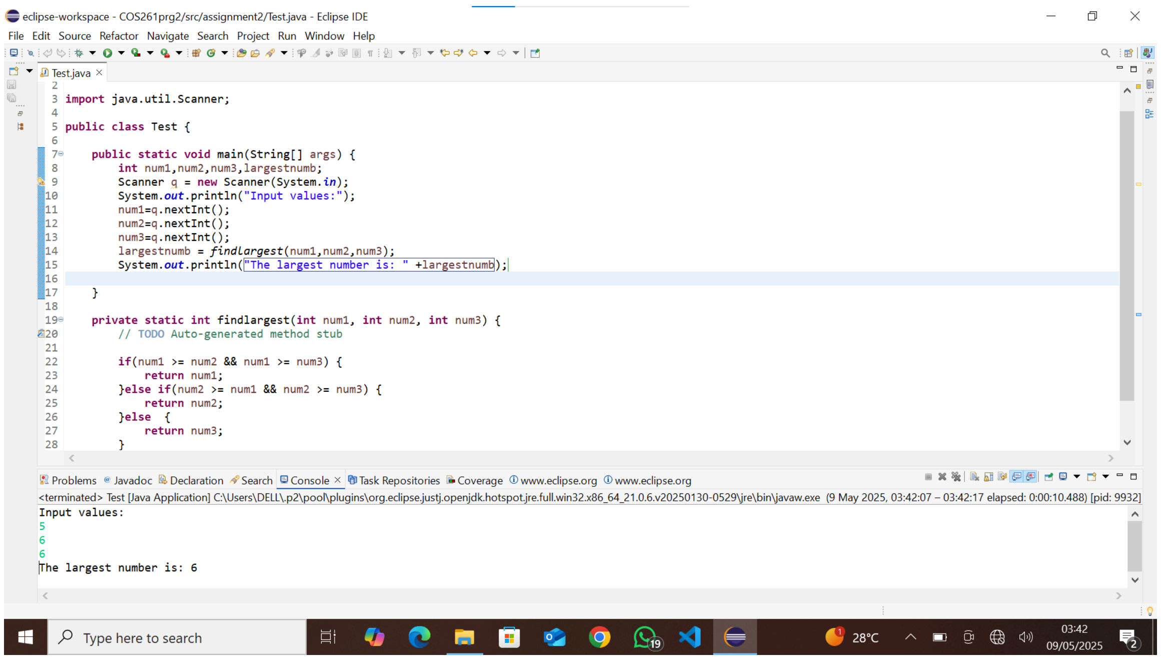
6 Write a program to check if a number is even or odd



```java
package assignment2;

import java.util.Scanner;

public class Test {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int numb;
        Scanner q = new Scanner(System.in);
        System.out.println("Enter a number:");
        numb=q.nextInt();

        if(numb % 2 == 0) {
            System.out.println(numb +" is even.");
        }else {
            System.out.println(numb +" is odd.");
        }

    }

}
```

```
<terminated> Test [Java Application] C:\Users\DELL\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.6.v20250130-0529\jre\bin\javaw.exe (9 May 2025, 03:16:53 – 03:16:57 elapsed: 0:00:03.949) [pid: 13396
Enter a number:
7
7 is odd.
```

7 Write a program to find the largest among three numbers



```java

import java.util.Scanner;

public class Test {

    public static void main(String[] args) {
        int num1,num2,num3,largestnumb;
        Scanner q = new Scanner(System.in);
        System.out.println("Input values:");
        num1=q.nextInt();
        num2=q.nextInt();
        num3=q.nextInt();
        largestnumb = findlargest(num1,num2,num3);
        System.out.println("The largest number is:  " +largestnumb);

    }

    private static int findlargest(int num1, int num2, int num3) {
        // TODO Auto-generated method stub

        if(num1 >= num2 && num1 >= num3) {
            return num1;
        }else if(num2 >= num1 && num2 >= num3) {
            return num2;
        }else  {
            return num3;
        }
```

```
<terminated> Test [Java Application] C:\Users\DELL\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.6.v20250130-0529\jre\bin\javaw.exe (9 May 2025, 03:42:07 – 03:42:17 elapsed: 0:00:10.488) [pid: 9932]
Input values:
5
6
6
The largest number is: 6
```

8 Explain The difference between while,for and do-while loop in java

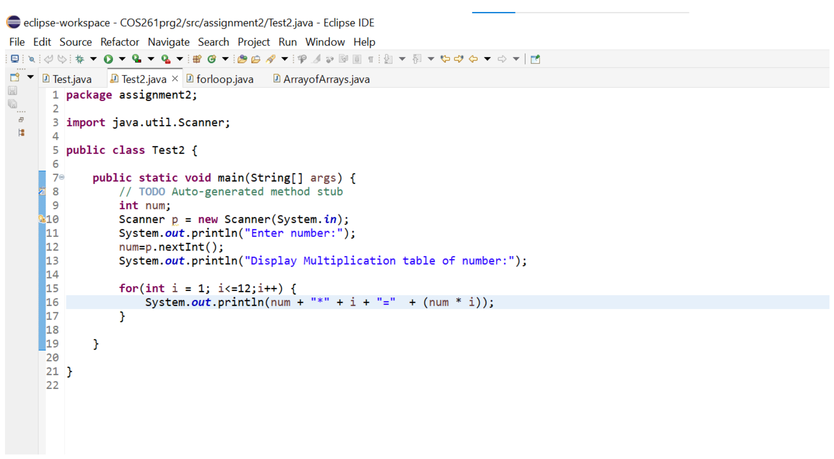
While Loop
1. Condition-Based: The while loop continues to execute as long as the condition is true.
2. Pre-Test Loop: The condition is checked before the loop body is executed.
3. Flexible: Can be used for any type of loop.

For Loop
1. Counter-Based: The for loop is often used when you know the number of iterations.

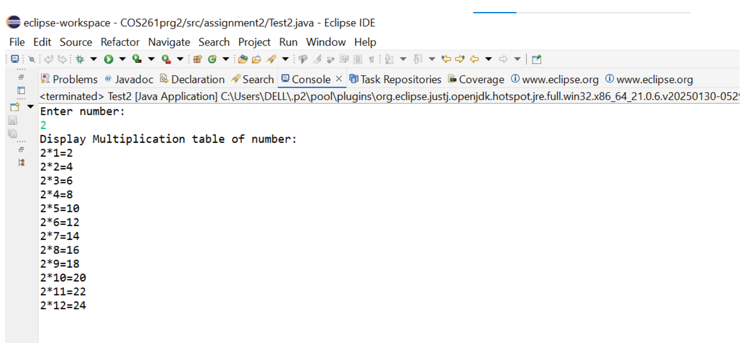
2. Initialization, Condition, and Increment: The for loop has three parts: initialization, condition, and increment.
3. Concise: Can be more concise than a while loop.

Do-While Loop
1. Post-Test Loop: The condition is checked after the loop body is executed.
2. Guaranteed Execution: The loop body is executed at least once.
3. Useful for Menus: Often used in menu-driven programs where you want to display the menu at least once.

9 Write a java program to print the multiplication table of any number


eclipse-workspace - COS261prg2/src/assignment2/Test2.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Test.java
Test2.java
forloop.java
ArrayofArrays.java

```
package assignment2;

import java.util.Scanner;

public class Test2 {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		int num;
		Scanner p = new Scanner(System.in);
		System.out.println("Enter number:");
		num=p.nextInt();
		System.out.println("Display Multiplication table of number:");

		for(int i = 1; i<=12;i++) {
			System.out.println(num + "*" + i + "="  + (num * i));
		}

	}

}
```


eclipse-workspace - COS261prg2/src/assignment2/Test2.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Problems
Javadoc
Declaration
Search
Console
Task Repositories
Coverage
www.eclipse.org
www.eclipse.org

```
<terminated> Test2 [Java Application] C:\Users\DELL\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.6.v20250130-052
Enter number:
2
Display Multiplication table of number:
2*1=2
2*2=4
2*3=6
2*4=8
2*5=10
2*6=12
2*7=14
2*8=16
2*9=18
2*10=20
2*11=22
2*12=24
```

10 Explain The four pillars of OOP in java

The four pillars of Object-Oriented Programming (OOP) in Java are:

1. Encapsulation
1. Hiding Internal Details: Encapsulation is the concept of hiding the internal details of an object from the outside world.
2. Data Hiding: It involves wrapping up data (attributes) and methods (functions that operate on that data) into a single unit, called a class.
3. Access Modifiers: Java provides access modifiers like public, private, and protected to control access to an object's internal state.

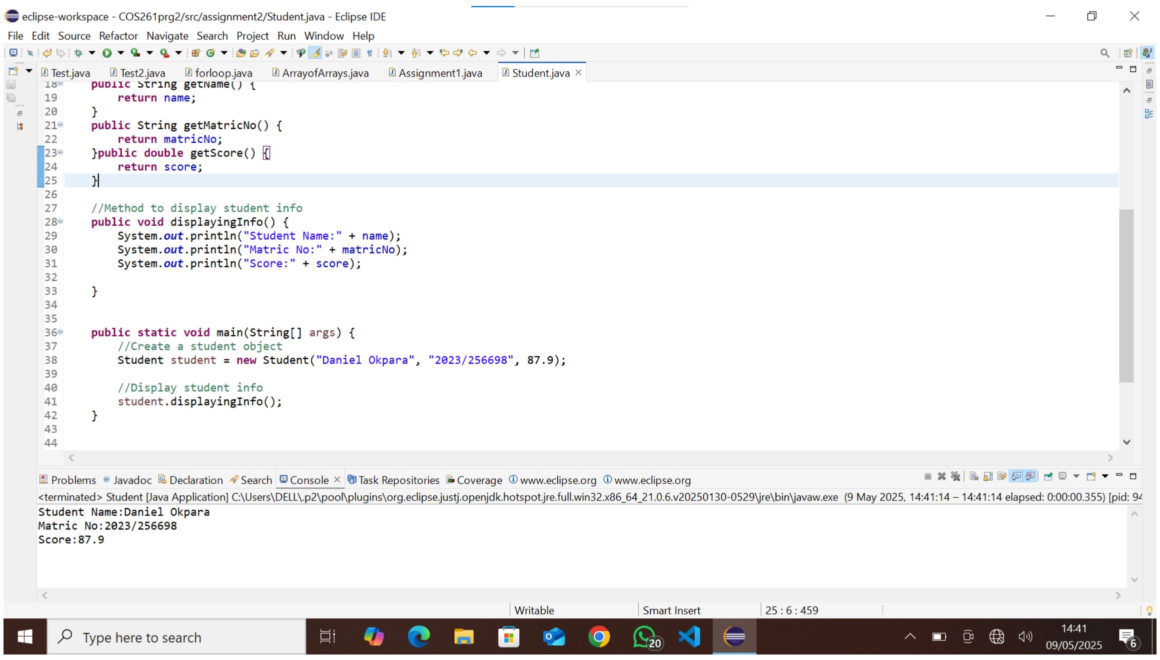
2. Abstraction

1. Showing Only Necessary Information: Abstraction is the concept of showing only the necessary information to the outside world while hiding the internal details.
2. Abstract Classes and Interfaces: Java provides abstract classes and interfaces to achieve abstraction.

3. Inheritance
1. Inheriting Properties and Behavior*: Inheritance is the concept of creating a new class based on an existing class.
2. Parent-Child Relationship: The existing class is called the parent or superclass, and the new class is called the child or subclass.
3. Code Reusability: Inheritance promotes code reusability by allowing subclasses to inherit properties and behavior from their parents.

4. Polymorphism
1. Multiple Forms: Polymorphism is the concept of an object taking on multiple forms.
2. Method Overloading and Overriding: Java supports method overloading (multiple methods with the same name but different parameters) and method overriding (a subclass provides a different implementation of a method already defined in its superclass).
3. Dynamic Method Dispatch: Polymorphism allows for dynamic method dispatch, where the correct method to be called is determined at runtime.

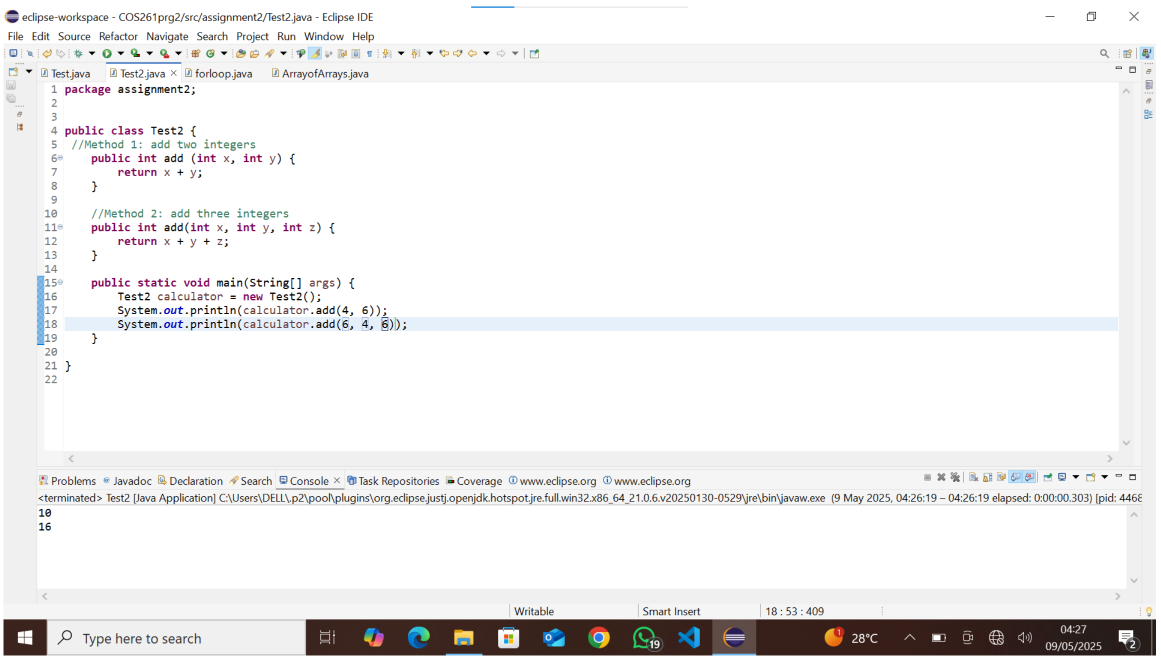
11 Create a class Student with properties name, matricNo, and score, and add methods to display the student's info.


eclipse-workspace - COS261prg2/src/assignment2/Student.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Test.java Test2.java forloop.java ArrayofArrays.java Assignment1.java Student.java
Problems Javadoc Declaration Search Console Task Repositories Coverage www.eclipse.org www.eclipse.org
Writable Smart Insert 25 : 6 : 459
Type here to search
14:41 09/05/2025

```java
    public String getName() {
        return name;
    }
    public String getMatricNo() {
        return matricNo;
    }public double getScore() {
        return score;
    }

    //Method to display student info
    public void displayingInfo() {
        System.out.println("Student Name:" + name);
        System.out.println("Matric No:" + matricNo);
        System.out.println("Score:" + score);

    }


    public static void main(String[] args) {
        //Create a student object
        Student student = new Student("Daniel Okpara", "2023/256698", 87.9);

        //Display student info
        student.displayingInfo();
    }

```

```
<terminated> Student [Java Application] C:\Users\DELL\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.6.v20250130-0529\jre\bin\javaw.exe  (9 May 2025, 14:41:14 – 14:41:14 elapsed: 0:00:00.355) [pid: 94
Student Name:Daniel Okpara
Matric No:2023/256698
Score:87.9
```

12 What is method overloading and give a code
Method Overloading is a feature in Java that allows a class to have multiple methods with the same name but different parameters. The method to be invoked is determined by the number and types of arguments passed to it.



```java
package assignment2;


public class Test2 {
 //Method 1: add two integers
    public int add (int x, int y) {
        return x + y;
    }

    //Method 2: add three integers
    public int add(int x, int y, int z) {
        return x + y + z;
    }

    public static void main(String[] args) {
        Test2 calculator = new Test2();
        System.out.println(calculator.add(4, 6));
        System.out.println(calculator.add(6, 4, 6));
    }

}
```

```
<terminated> Test2 [Java Application] C:\Users\DELL\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.6.v20250130-0529\jre\bin\javaw.exe  (9 May 2025, 04:26:19 – 04:26:19 elapsed: 0:00:00.303) [pid: 4468
10
16
```

13 What is inheritance? Create a base class Person and a sub classTeacher.
Inheritance is a fundamental concept in object-oriented programming (OOP) where a new class (the subclass or derived class) inherits the properties and behavior of an existing class (the superclass or base class). The subclass inherits all the fields and methods of the superclass and can also add new fields and methods or override the ones inherited from the superclass.

14 What those it mean to write a clean code? Give the practices that makes code clean and maintainable

Clean code refers to code that is well-organized, easy to understand, and maintainable. It follows best practices and principles that make it simple for developers to read, modify, and extend the codebase.

Practices for Clean Code
1. Meaningful Names:
   - Use descriptive and concise names for variables, methods, and classes.
   - Avoid abbreviations and acronyms unless they are widely recognized.
2. Single Responsibility Principle (SRP):
   - Each class or method should have a single responsibility and reason to change.
   - Avoid God objects or methods that do too much.
3. Separation of Concerns (SoC):
   - Separate different concerns, such as presentation, business logic, and data storage.
   - Use layers or modules to organize code.
4. Don't Repeat Yourself (DRY):
   - Avoid duplicated code by extracting common logic into reusable functions or classes.
   - Use inheritance or composition to share behavior.
5. Keep it Simple and Stupid (KISS):
   - Favor simple solutions over complex ones.
   - Avoid over-engineering or premature optimization.

6. Command-Query Separation (CQS):
   - Separate methods that perform actions (commands) from those that retrieve data (queries).
   - Avoid methods that both modify state and return values.
7. Code Formatting and Style:
   - Follow a consistent coding style and formatting guidelines.
   - Use whitespace, indentation, and line breaks to make code readable.
8. *Comments and Documentation:
   - Use comments to explain complex code or logic.
   - Write clear and concise documentation for classes, methods, and APIs.
9. Testing:
   - Write unit tests and integration tests to ensure code correctness.
   - Use testing frameworks and tools to automate testing.
10. Refactoring:
   - Regularly review and refactor code to improve its structure and maintainability.
   - Remove technical debt and improve code quality.

15. **Why should you avoid writing very long methods in Java programs?**

Avoiding long methods in Java programs is essential for several reasons:

Reasons to Avoid Long Methods
1. Readability: Long methods can be difficult to read and understand, making it harder for developers to maintain or modify the code.
2. Complexity: Long methods often perform multiple, unrelated tasks, increasing complexity and making it harder to debug.
3. Testability: Long methods can be challenging to test, as they may have many dependencies and side effects.
4. Reusability: Long methods are less likely to be reusable, as they often contain specific logic tied to a particular context.
5. Maintainability: Long methods can lead to technical debt, making it harder to modify or extend the codebase.

16 What naming convention should be in java for classes,Variables, Methods.
Give examples with screen shot of code and out put

Java Naming Conventions are essential for writing clean, readable, and maintainable code.

Classes
1. Use Nouns: Class names should be nouns or noun phrases.
2. PascalCase: Use PascalCase (also known as UpperCamelCase) for class names.
3. Descriptive: Class names should be descriptive and indicate the class's purpose.

Variables
1. Use Meaningful Names: Variable names should be descriptive and indicate the variable's purpose.
2. CamelCase: Use camelCase (also known as lowerCamelCase) for variable names.
3. Avoid Abbreviations: Avoid using abbreviations unless they are widely recognized.

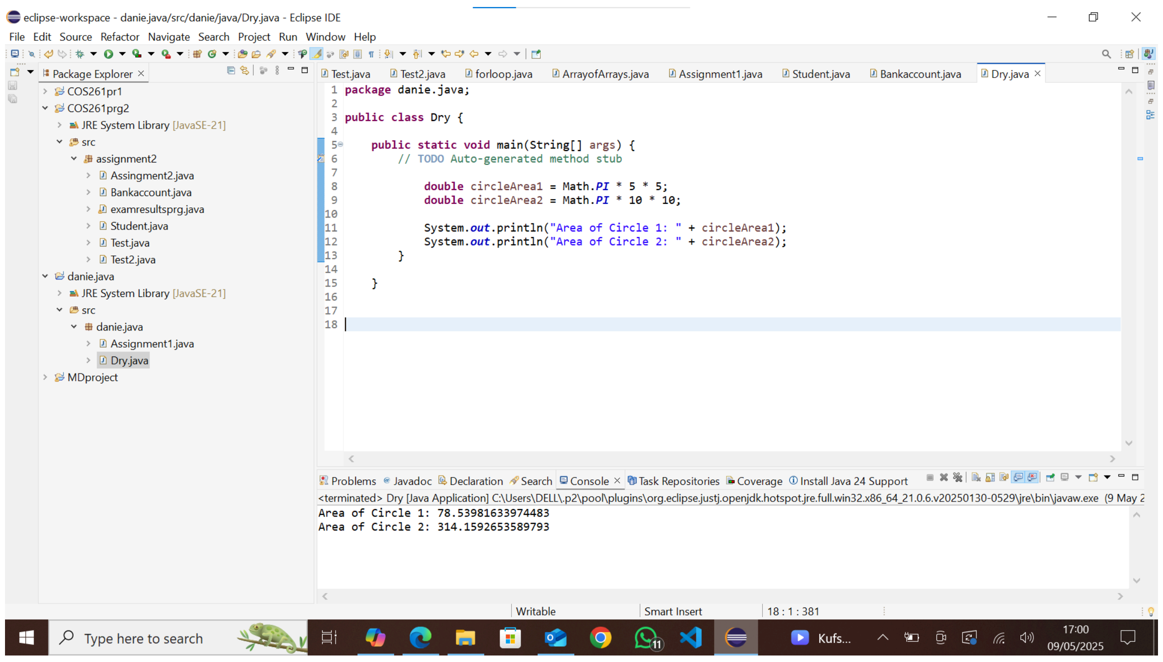
Methods
1. Use Verbs: Method names should be verbs or verb phrases.
2. CamelCase: Use camelCase for method names.
3. Descriptive: Method names should be descriptive and indicate the method's purpose.

17. **What is the importance of breaking your Java program into methods?**

Importance of Breaking Java Program into Methods
1. Modularity: Methods promote modular code, making it easier to manage and maintain.
2. Reusability: Methods can be reused in multiple parts of the program, reducing code duplication.
3. Readability: Breaking code into methods improves readability by providing clear, concise, and focused functionality.
4. Easier Debugging: Methods make it easier to identify and isolate issues.
5. Single Responsibility Principle (SRP): Each method can have a single responsibility, making code more organized.

18. Explain the concept of DRY (Don't Repeat Yourself) with a Java code example.

DRY (Don't Repeat Yourself) is a fundamental principle in software development that aims to reduce repetition of code. It states that every piece of knowledge must have a single, unambiguous representation within a system.



```java
package danie.java;

public class Dry {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

            double circleArea1 = Math.PI * 5 * 5;
            double circleArea2 = Math.PI * 10 * 10;

            System.out.println("Area of Circle 1: " + circleArea1);
            System.out.println("Area of Circle 2: " + circleArea2);
        }

    }


```

```
<terminated> Dry [Java Application] C:\Users\DELL\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.6.v20250130-0529\jre\bin\javaw.exe  (9 May 2
Area of Circle 1: 78.53981633974483
Area of Circle 2: 314.1592653589793
```

19. What are the benefits of using classes and objects instead of writing all logic in the main method?

Benefits of Using Classes and Objects

Benefits
1. Modularity: Classes and objects promote modularity, making it easier to manage complex systems.
2. Reusability: Classes can be reused in multiple contexts, reducing code duplication.
3. Encapsulation: Classes encapsulate data and behavior, improving data hiding and reducing coupling.
4. Abstraction: Classes can abstract away complex implementation details, making it easier to focus on essential features.
5. Easier Maintenance: With classes and objects, changes can be made in one place, reducing the risk of errors.
6. Improved Readability: Classes and objects make code more readable by organizing related data and behavior together.
7. Better Error Handling: Classes can handle errors and exceptions in a more structured way.

20. **Why is testing important during program development?**

Importance of Testing
1. Ensures Correctness: Testing verifies that the program works as expected and produces the desired output.
2. Catches Bugs Early: Testing helps identify and fix errors early in the development process, reducing costs and time.
3. Improves Quality: Thorough testing ensures the program is reliable, stable, and performs well.
4. Reduces Risk: Testing minimizes the risk of downstream problems, such as crashes or data loss.
5. Facilitates Maintenance: Well-tested code is easier to maintain and modify, as changes can be verified through testing.

21. What is the difference between syntax error, runtime error, and logic error?

Types of Errors
1. Syntax Error
1. Definition: Violation of programming language rules.
2. Examples: Missing semicolons, mismatched brackets.
3. Detection: Typically caught by compilers or IDEs.

2. Runtime Error
1. Definition: Errors that occur during program execution.
2. Examples*: Division by zero, null pointer exceptions.
3. Detection: Occur while the program is running.

3. Logic Error
1. Definition: Errors in program logic or algorithm.
2. Examples: Incorrect calculations, flawed logic.
3. Detection: Often requires manual testing and debugging.

22. **How would you test a method that calculates the average of five numbers?**

Testing a Method to Calculate Average
Test Cases
1. Valid input: Test with known values (e.g., 1, 2, 3, 4, 5) to verify the average calculation.
2. Edge cases: Test with extreme values (e.g., very large or small numbers).
3. Invalid input: Test with invalid inputs (e.g., non-numeric values).

Testing Steps
1. Prepare test data (input values and expected results).
2. Call the method with test data.
3. Compare actual results with expected results.

Example Test Cases
1. Input: 1, 2, 3, 4, 5 | Expected Output: 3.0
2. Input: 10, 20, 30, 40, 50 | Expected Output: 30.0
3. Input: -1, -2, -3, -4, -5 | Expected Output: -3.0

23 Why should java developers write comment in their code

1. Improved Readability: Comments explain code logic, making it easier for others (and yourself) to understand.
2. Better Maintainability: Comments help identify code purpose and functionality, simplifying updates and modifications.
3. Knowledge Sharing: Comments facilitate knowledge transfer among team members.
4. Debugging Assistance: Comments can highlight complex sections of code, aiding in debugging.

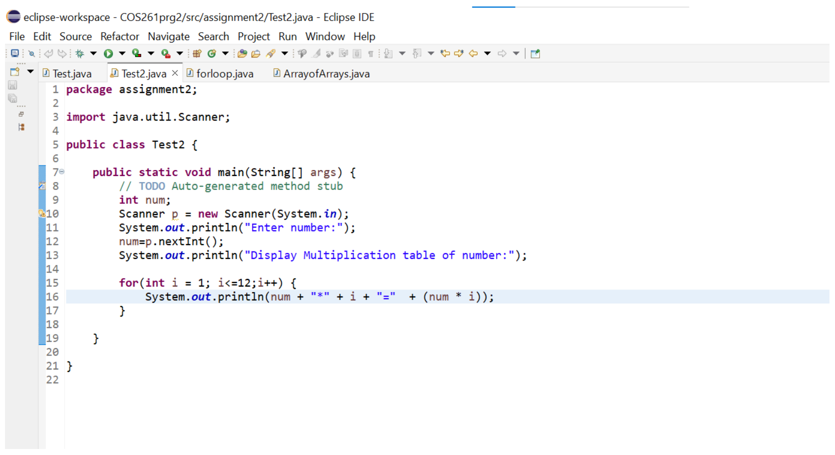
24. What are JavaDoc comments and how are they different from regular comments?

JavaDoc comments are used to generate documentation for Java code, providing information about classes, methods, and variables.

Format
JavaDoc comments start with /and end with /, and use tags like @param, @return, and @throws to describe code elements.

Difference from Regular Comments
1. Documentation generation: JavaDoc comments are used to generate HTML documentation.
2. Standardized format: JavaDoc comments follow a specific format and use specific tags.
3. Purpose: JavaDoc comments provide detailed information about code elements.

25. Write a sample Java method with JavaDoc comments


eclipse-workspace - COS261prg2/src/assignment2/Test2.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Test.java
Test2.java
forloop.java
ArrayofArrays.java

```java
package assignment2;

import java.util.Scanner;

public class Test2 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int num;
        Scanner p = new Scanner(System.in);
        System.out.println("Enter number:");
        num=p.nextInt();
        System.out.println("Display Multiplication table of number:");

        for(int i = 1; i<=12;i++) {
            System.out.println(num + "*" + i + "="  + (num * i));
        }

    }

}

```

26. What is version control and why is it important in team projects?

Version control is a system that tracks changes to code, documents, or other digital content over time.

Importance in Team Projects
1. Collaboration: Version control enables multiple team members to work on the same project simultaneously.
2. Change tracking: It tracks changes, allowing teams to identify who made changes and why.
3. Version management: Version control helps manage different versions of code or documents.
4. Conflict resolution: It assists in resolving conflicts that arise from multiple changes.
5. Backup and recovery: Version control provides a backup of the project, enabling recovery in case of data loss.

27 How would you explain the concept of "code refactoring" to a junior developer?

Code refactoring is the process of restructuring existing code to improve its quality, readability, and maintainability without changing its functionality.

Goals
1. Simplify code: Make code easier to understand and maintain.
2. Improve performance: Enhance code efficiency and speed.
3. Reduce complexity: Eliminate unnecessary complexity.

28 What tool can java developers use to collaborate on large projects. attach screen on 3 examples

Version Control Systems:
GitHub : A widely-used platform for version control and collaboration, offering features like pull requests, code review, and integrated issue tracking.
GitLab: Similar to GitHub, providing version control, collaboration, and continuous integration/continuous deployment (CI/CD) capabilities.

-Communication and Messaging:
Slack: A powerful messaging platform for real-time communication, file sharing, and integration with development tools.

Project Management:
Jira: A popular tool for project management, issue tracking, and agile project planning, offering customizable workflows and integration with other Atlassian products.

Real-time Code Collaboration:
Visual Studio Live Share: A tool for real-time collaboration on code, allowing multiple developers to work together on a shared codebase.
CodeSandbox Live: A platform for collaborative coding, offering features like live editing, voice chat, and group debugging.
Codeanywhere: A cloud-based development environment that enables real-time collaboration and pair programming.

29. Mention 5 best practices you follow when developing a Java program.

1. Follow Naming Conventions: Use meaningful and consistent names for variables, methods, and classes.
2. Write Clean and Modular Code: Break down code into smaller, reusable methods and classes.
3. Use Comments and Documentation: Add comments to explain complex code and use JavaDoc for documentation.
4. Handle Exceptions Properly: Use try-catch blocks to handle exceptions and provide meaningful error messages.
5. Test Thoroughly: Write unit tests to ensure code functionality and catch bugs early.

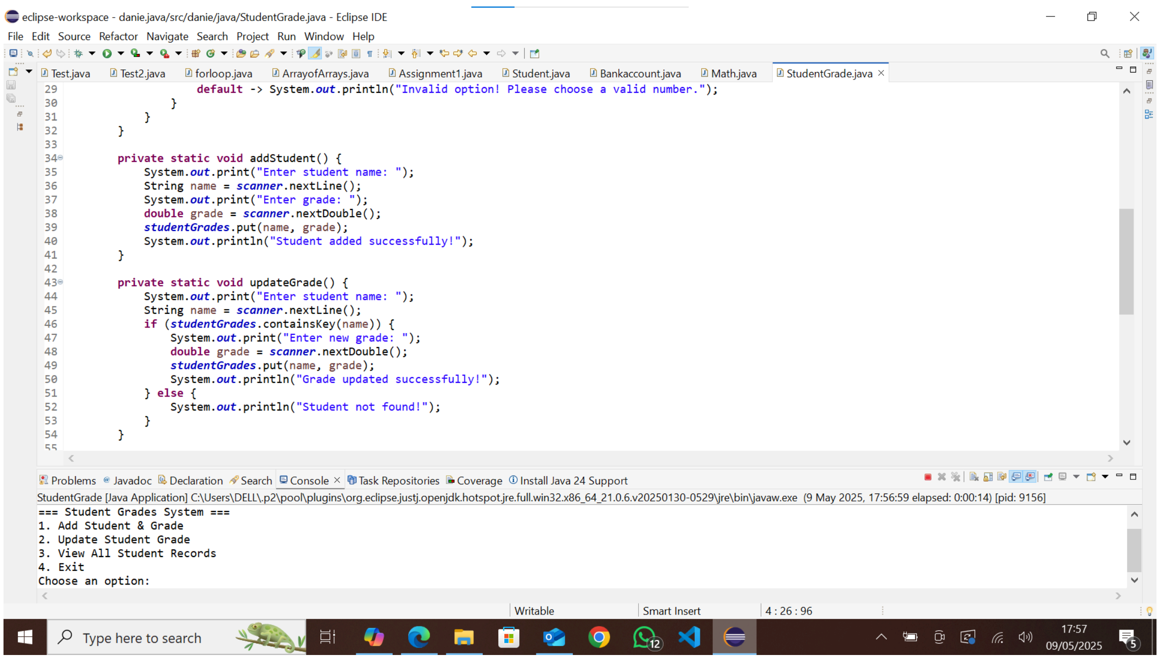
30. What is code readability, and why is it more important than “smart” code?

Code readability: refers to how easily someone can understand and maintain code.

Importance
1. Easier maintenance: Readable code is simpler to update and modify.
2. Faster onboarding: New team members can understand codebase more quickly.
3. Reduced errors: Readable code helps prevent misunderstandings.

Why Readability Trumps "Smart" Code
1. Maintainability: Readable code is more important than clever code.
2. Collaboration: Readable code facilitates teamwork.
3. Long-term benefits: Readability outweighs short-term cleverness.

31. Build a command-line application that keeps track of student grades and allows adding, updating, and viewing records.



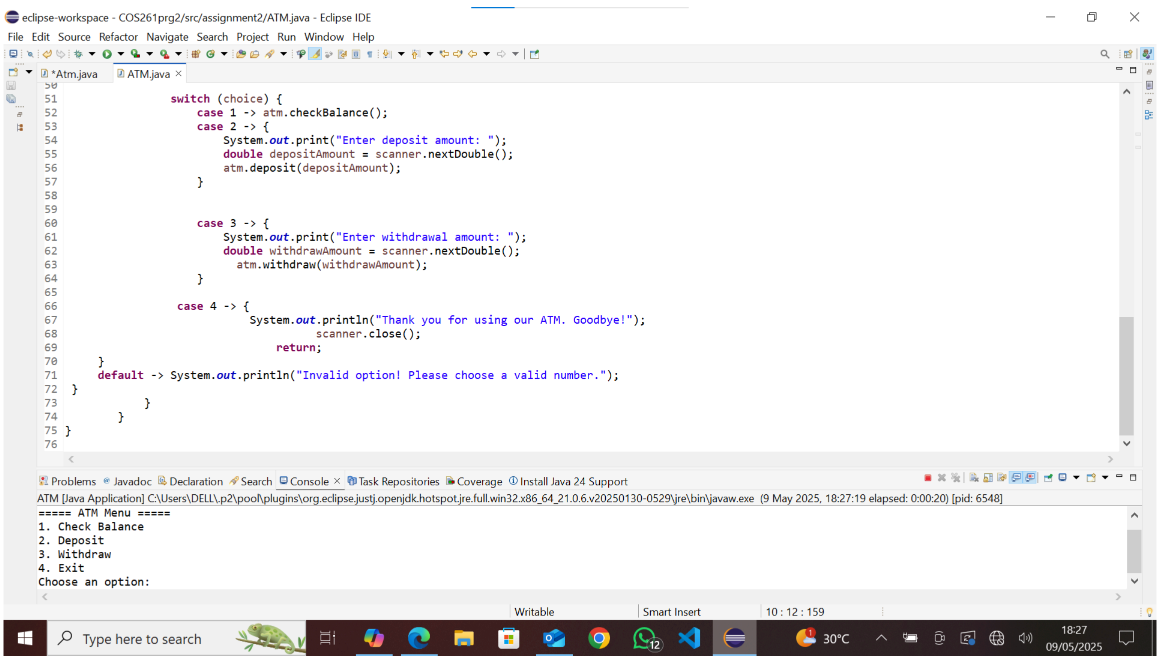
```java
                    default -> System.out.println("Invalid option! Please choose a valid number.");
                }
            }
        }

        private static void addStudent() {
            System.out.print("Enter student name: ");
            String name = scanner.nextLine();
            System.out.print("Enter grade: ");
            double grade = scanner.nextDouble();
            studentGrades.put(name, grade);
            System.out.println("Student added successfully!");
        }

        private static void updateGrade() {
            System.out.print("Enter student name: ");
            String name = scanner.nextLine();
            if (studentGrades.containsKey(name)) {
                System.out.print("Enter new grade: ");
                double grade = scanner.nextDouble();
                studentGrades.put(name, grade);
                System.out.println("Grade updated successfully!");
            } else {
                System.out.println("Student not found!");
            }
        }
```

StudentGrade [Java Application] C:\Users\DELL\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.6.v20250130-0529\jre\bin\javaw.exe (9 May 2025, 17:56:59 elapsed: 0:00:14) [pid: 9156]

```
=== Student Grades System ===
1. Add Student & Grade
2. Update Student Grade
3. View All Student Records
4. Exit
Choose an option:
```

32. Write a program that simulates a basic ATM system (check balance, deposit, withdraw).

Note: For all code/ program examples, create a github



```java
                switch (choice) {
                    case 1 -> atm.checkBalance();
                    case 2 -> {
                        System.out.print("Enter deposit amount: ");
                        double depositAmount = scanner.nextDouble();
                        atm.deposit(depositAmount);
                    }


                    case 3 -> {
                        System.out.print("Enter withdrawal amount: ");
                        double withdrawAmount = scanner.nextDouble();
                          atm.withdraw(withdrawAmount);
                    }

                 case 4 -> {
                            System.out.println("Thank you for using our ATM. Goodbye!");
                                      scanner.close();
                                return;
     }
     default -> System.out.println("Invalid option! Please choose a valid number.");
 }
            }
        }
}
```

ATM [Java Application] C:\Users\DELL\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.6.v20250130-0529\jre\bin\javaw.exe (9 May 2025, 18:27:19 elapsed: 0:00:20) [pid: 6548]

```
===== ATM Menu =====
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Choose an option:
```