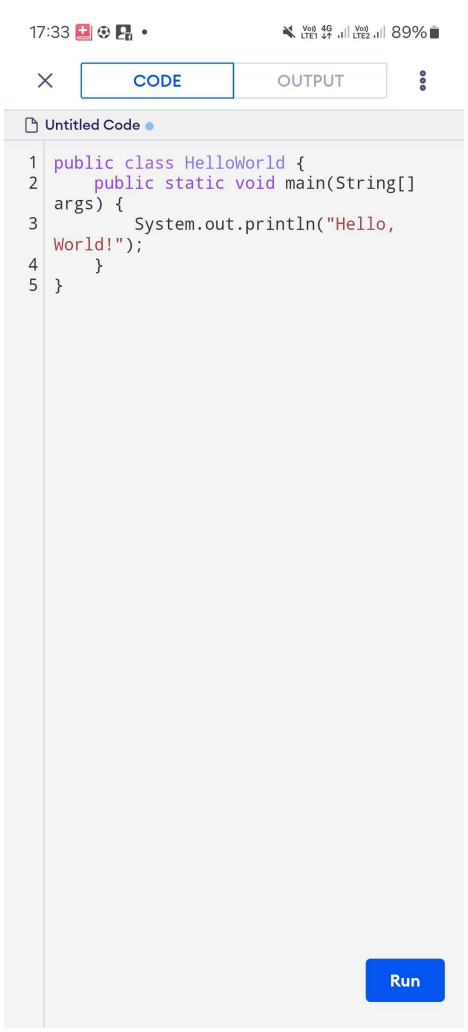
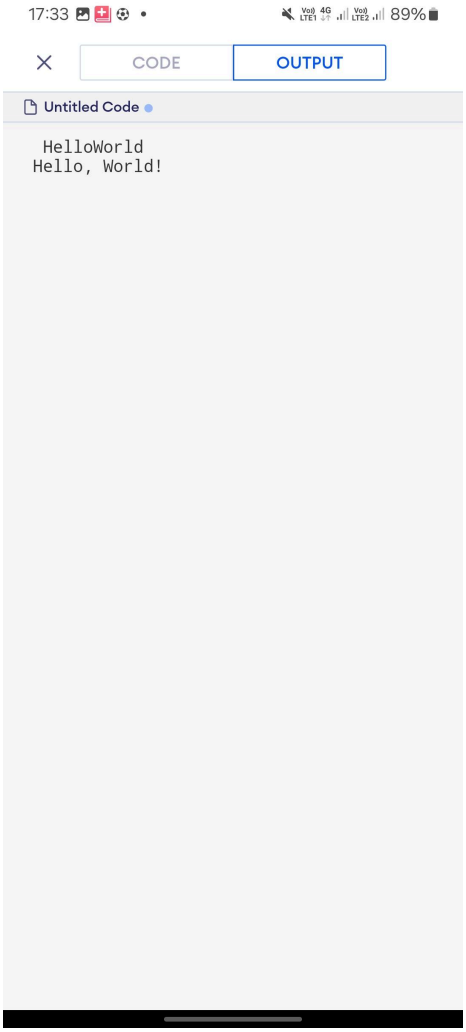
NAME: Nweze Chibuikem Ogbonna
REG NO.: 2023/253020
DEPT: Computer Science
COURSE: COS 261

Cos 261 test

1. Write a Java program to print "Hello, World!".



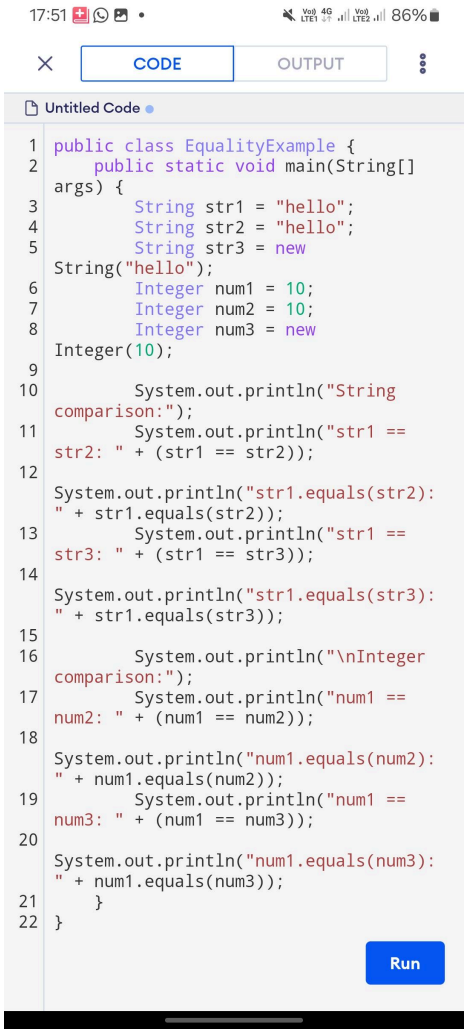


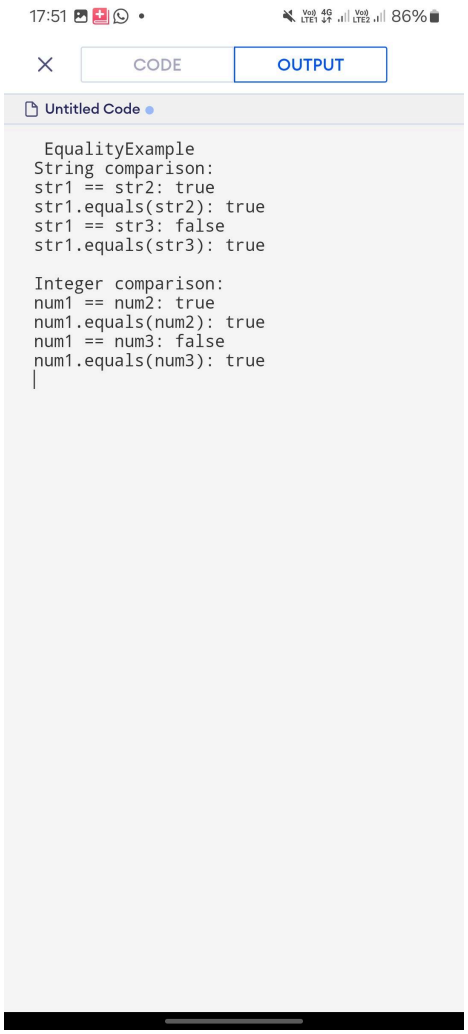
2. Explain the difference between `==` and `.equals()` in Java. Show with code examples and outputs.

   a) == :
   - Compares the memory locations of two objects (for reference types) or the actual values (for primitive types).
   - For objects, it checks if both references point to the same object in memory.
   - For primitives (like int , boolean , etc.), it compares the values directly.

b) .equals() :

- Compares the actual content or values of two objects.
- By default, .equals() behaves like == for reference types unless overridden in a class.
- Many Java classes (like String , Integer, etc.) override .equals() to compare values meaningfully.

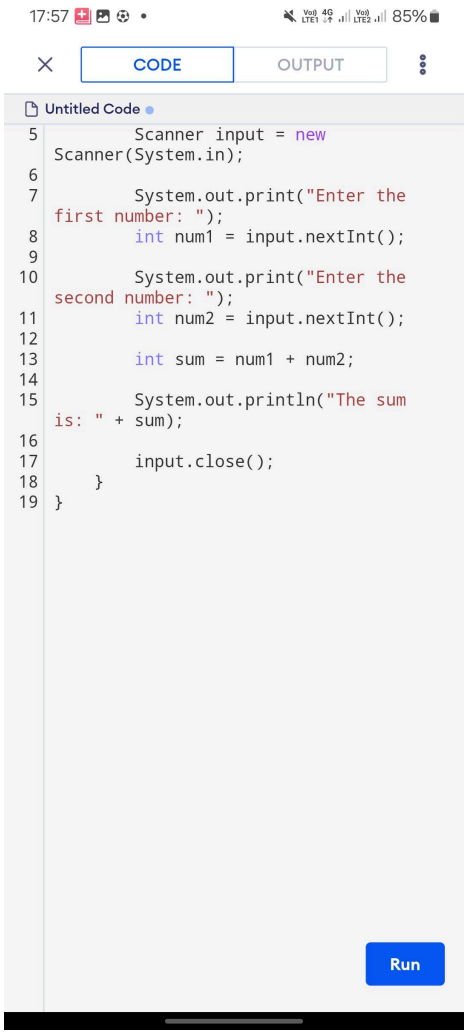


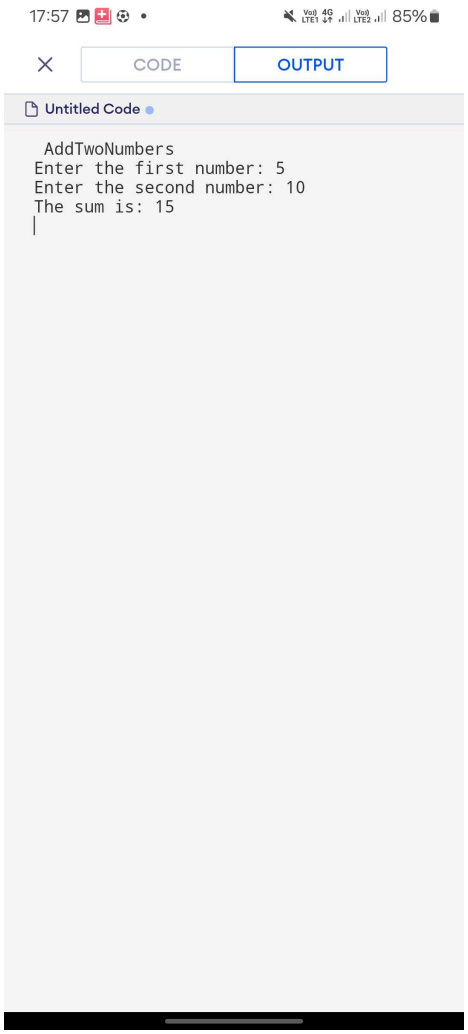


3. What is the use of the `main` method in Java?
   The main method in Java is the entry point of any standalone Java application. The Java Virtual Machine (JVM) starts the execution of a program by calling the main method. It has a specific signature: public static void main(String[] args).

   ● public: It must be public so that the JVM can access it from outside the class.
   ● static: It must be static so that the JVM can call it without creating an instance of the class.
   ● void: It doesn't return any value.
   main: This is the predefined name that the JVM looks for.
   ● String[] args: This is an array of strings that allows you to pass command-line arguments to the program

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


17:57
85%
CODE
OUTPUT
Untitled Code
Scanner input = new Scanner(System.in);
System.out.print("Enter the first number: ");
int num1 = input.nextInt();
System.out.print("Enter the second number: ");
int num2 = input.nextInt();
int sum = num1 + num2;
System.out.println("The sum is: " + sum);
input.close();
}
}
Run



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

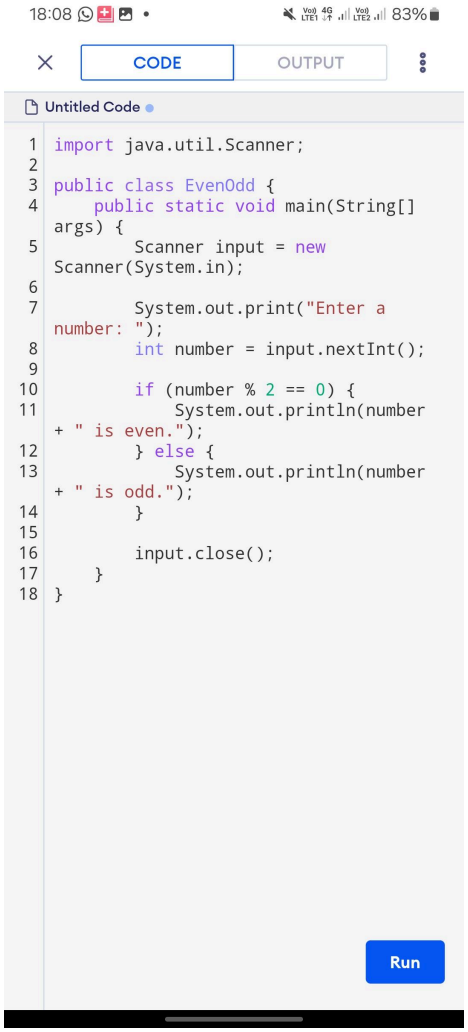
   a) int :
   - A basic data type that represents a 32-bit signed integer.
   - Holds a numeric value directly.
   - Not an object, so it doesn't have methods.
   - More efficient in terms of memory and performance.

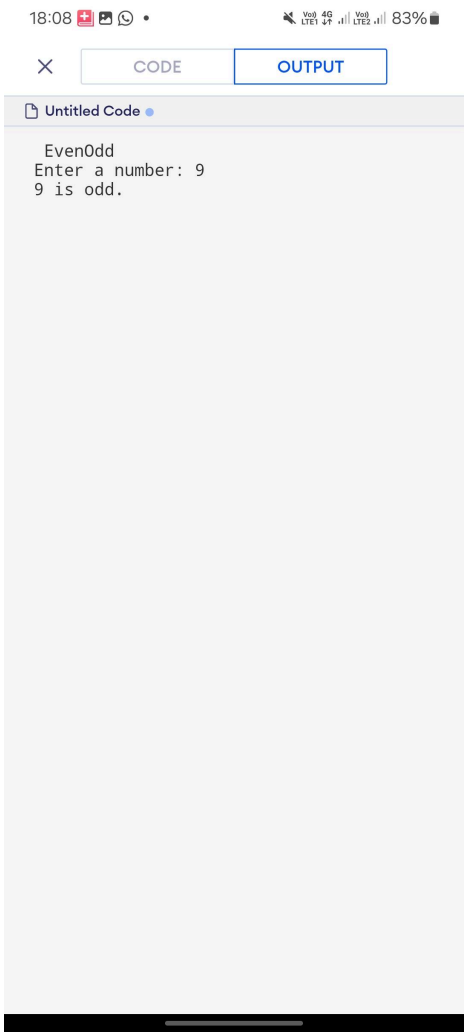
   b). Integer :
   - A class that wraps the int primitive type, providing object-oriented features.
   - Can hold null values, unlike int.
   - Offers methods like parseInt(), valueOf(), and more.
   - Useful for generics, collections, and situations where objects are required.

   c). String :
   - A class that represents a sequence of characters.
   - Immutable, meaning its value cannot be changed once created.

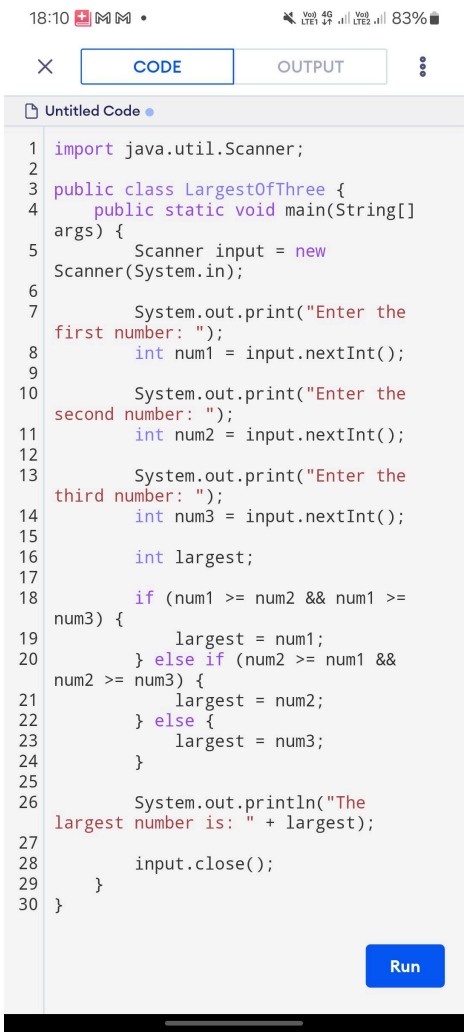
- Offers various methods for string manipulation, like length(), substring(), and equals().

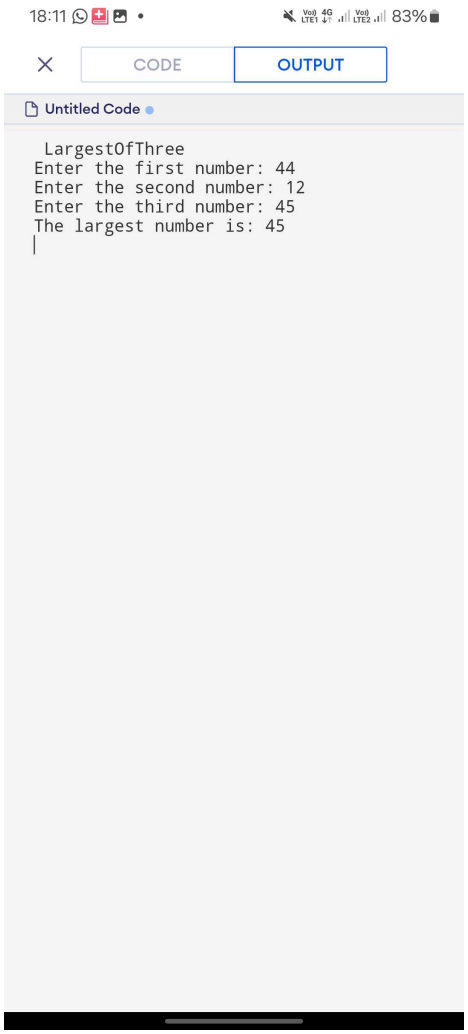
6. Write a program to check if a number is even or odd.





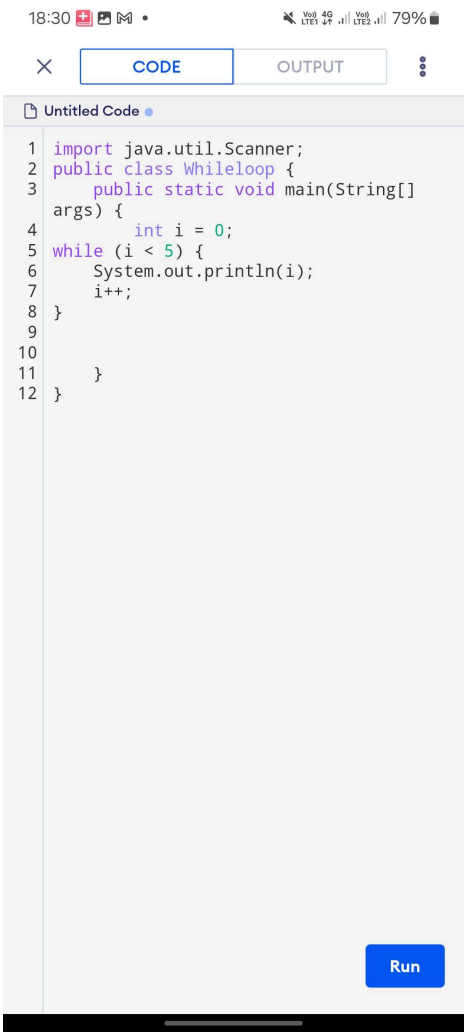
7. Write a program to find the largest among three numbers.

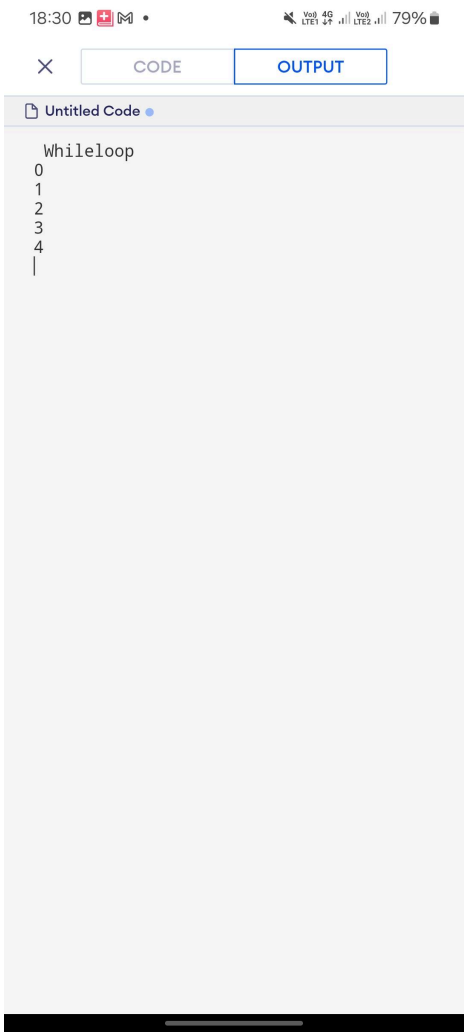

18:10
83%
CODE
OUTPUT
Untitled Code
import java.util.Scanner;
public class LargestOfThree {
public static void main(String[] args) {
Scanner input = new Scanner(System.in);
System.out.print("Enter the first number: ");
int num1 = input.nextInt();
System.out.print("Enter the second number: ");
int num2 = input.nextInt();
System.out.print("Enter the third number: ");
int num3 = input.nextInt();
int largest;
if (num1 >= num2 && num1 >= num3) {
largest = num1;
} else if (num2 >= num1 && num2 >= num3) {
largest = num2;
} else {
largest = num3;
}
System.out.println("The largest number is: " + largest);
input.close();
}
}
Run



8. Explain the difference between `while`, `for`, and `do-while` loops in Java.

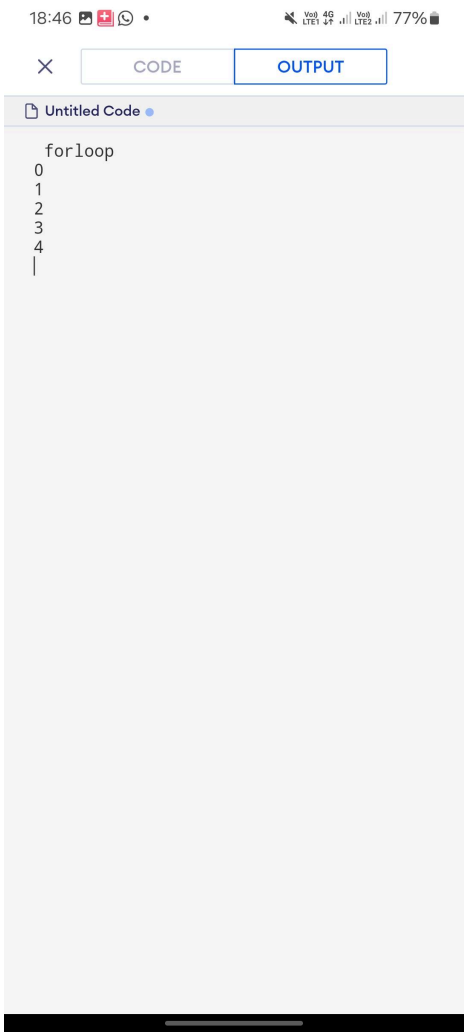
   ● while loop: The while loop executes a block of code as long as a specified condition is true. The condition is checked before each iteration. If the condition is initially false, the loop body will not execute at all.


18:30
79%
CODE
OUTPUT
Untitled Code
1 import java.util.Scanner;
2 public class Whileloop {
3 public static void main(String[]
args) {
4 int i = 0;
5 while (i < 5) {
6 System.out.println(i);
7 i++;
8 }
9
10
11 }
12 }
Run

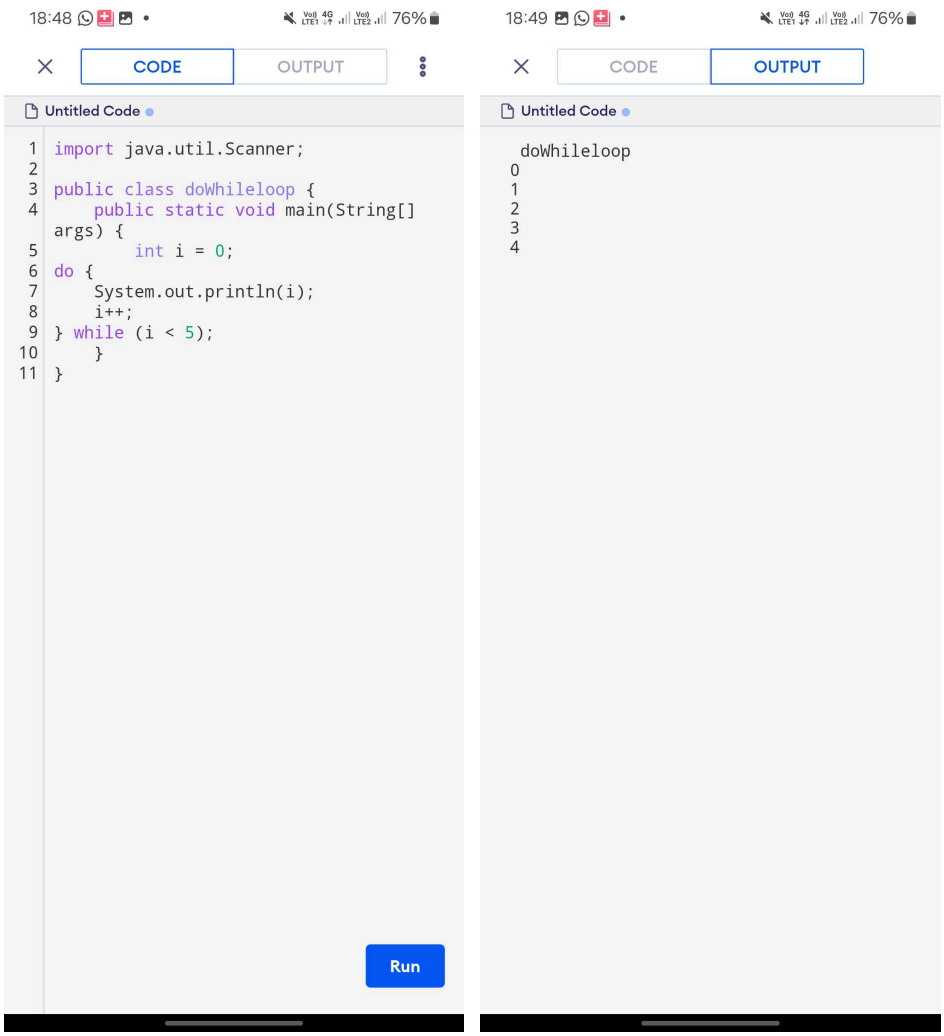


● for loop: The for loop provides a more structured way to iterate a specific number of times or over a collection. It typically consists of three parts (initialization, condition, increment/decrement) separated by semicolons, all within the parentheses

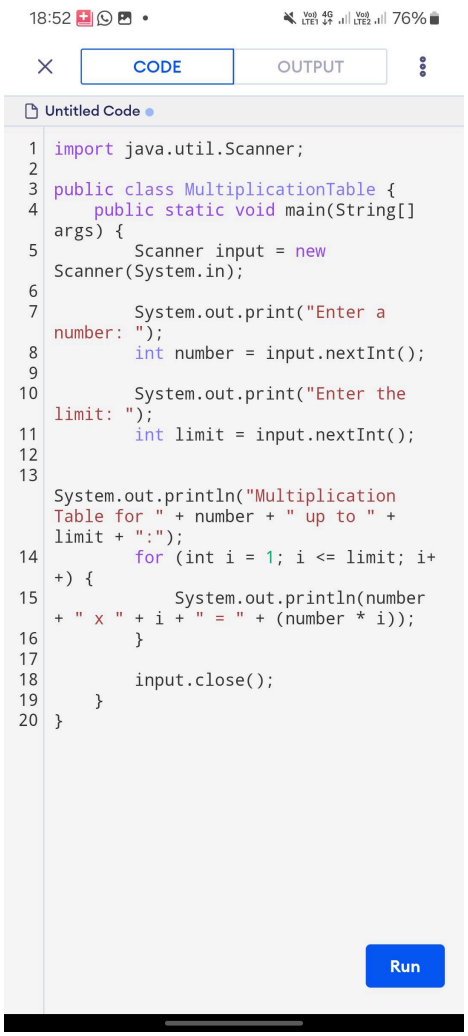

18:46
77%
CODE
OUTPUT
Untitled Code
1 import java.util.Scanner;
2
3 public class forloop {
4 public static void main(String[]
args) {
5 for (int i = 0; i < 5; i++) {
6 System.out.println(i);
7 }
8 }
9 }
Run

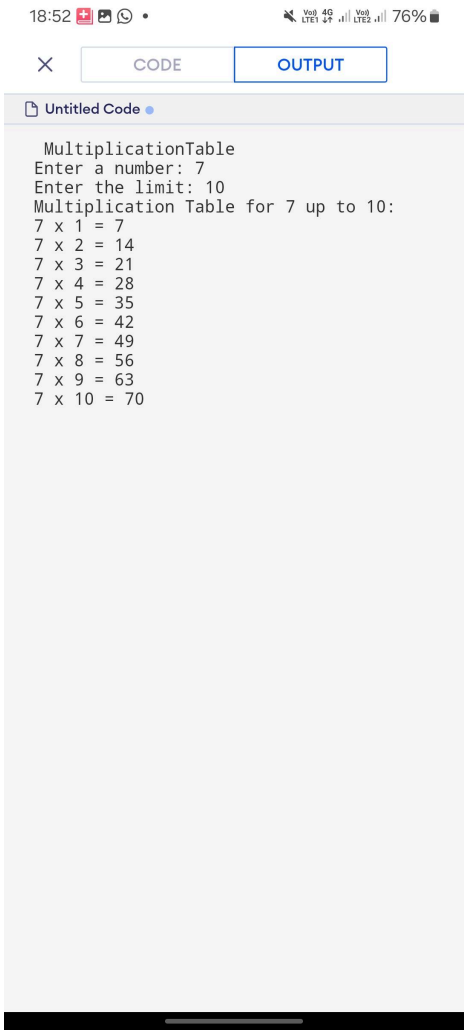


● do-while loop: The do-while loop is similar to the while loop, but the condition is checked after the loop body is executed. This guarantees that the loop body will execute at least once, even if the condition is initially false.



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


18:52
76%
CODE
OUTPUT
Untitled Code
import java.util.Scanner;
public class MultiplicationTable {
public static void main(String[] args) {
Scanner input = new Scanner(System.in);
System.out.print("Enter a number: ");
int number = input.nextInt();
System.out.print("Enter the limit: ");
int limit = input.nextInt();
System.out.println("Multiplication Table for " + number + " up to " + limit + ":");
for (int i = 1; i <= limit; i++) {
System.out.println(number + " x " + i + " = " + (number * i));
}
input.close();
}
}
Run

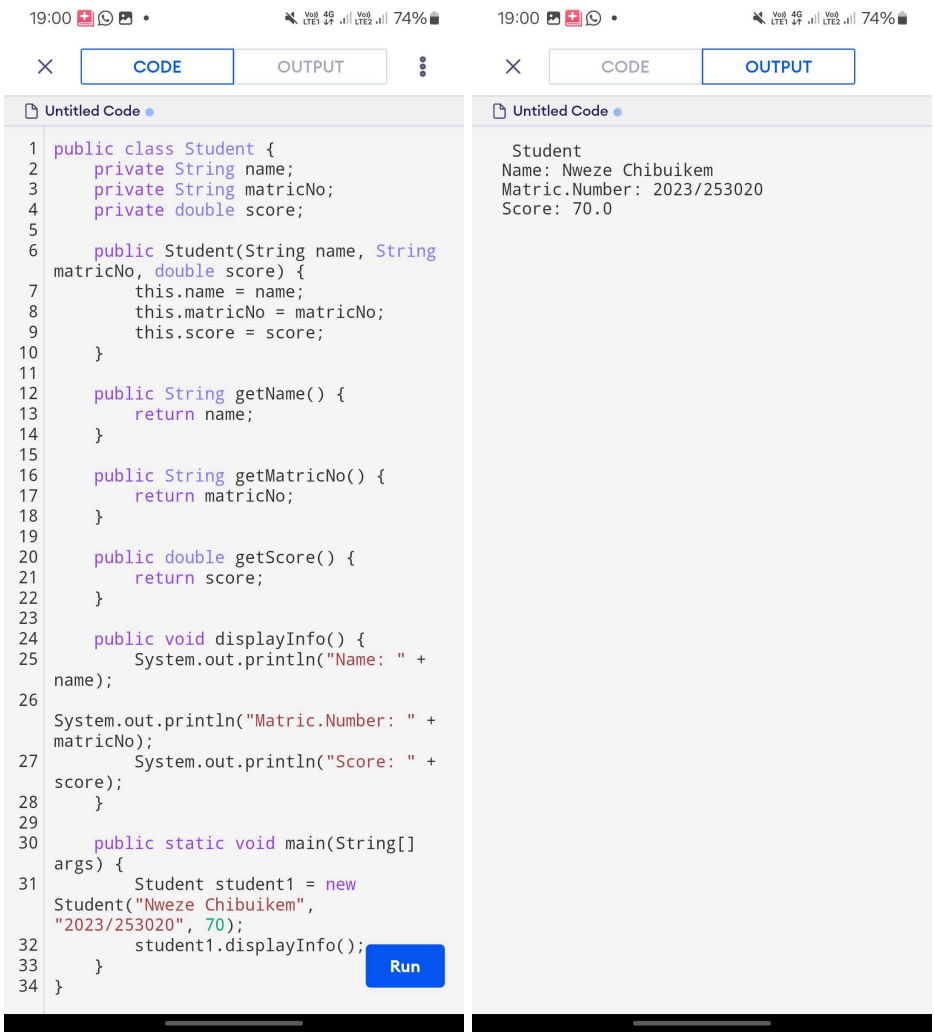


10. Explain the four pillars of OOP in Java.

    The four pillars of OOP in java are;

a) Encapsulation
b) Abstraction
c) Inheritance
d) Polymorphism

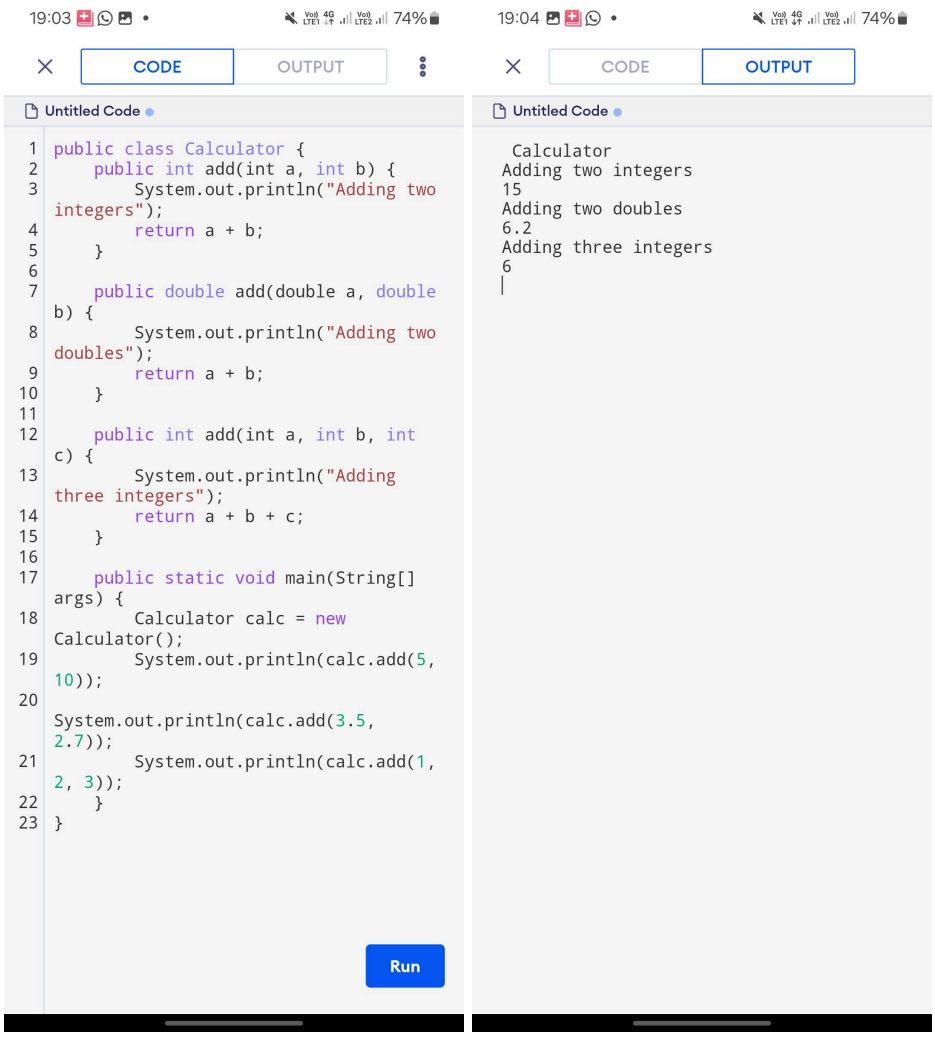
11. Create a class Student with properties name, matricNo, and score, and add methods to display the student's info.



12. What is method overloading? Give a code example.

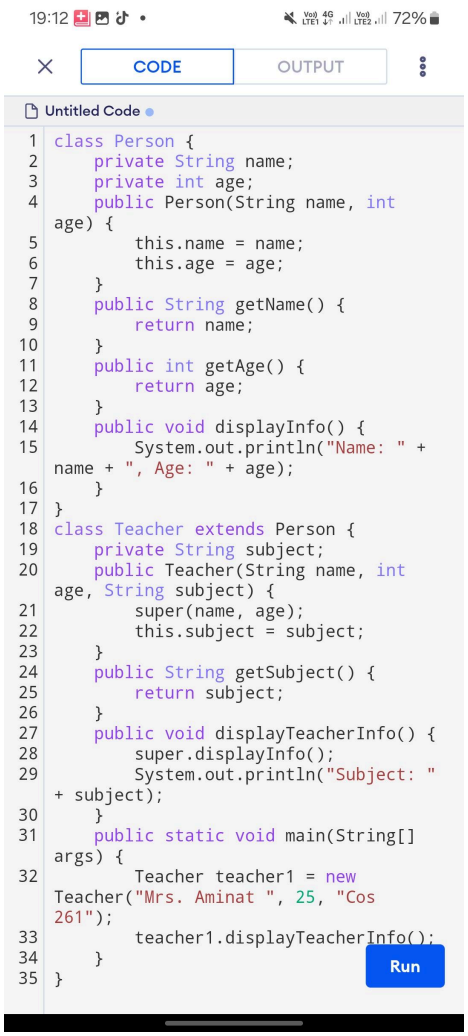
    Method overloading is a feature in Java that allows a class to have multiple methods with the same name but different parameter lists.

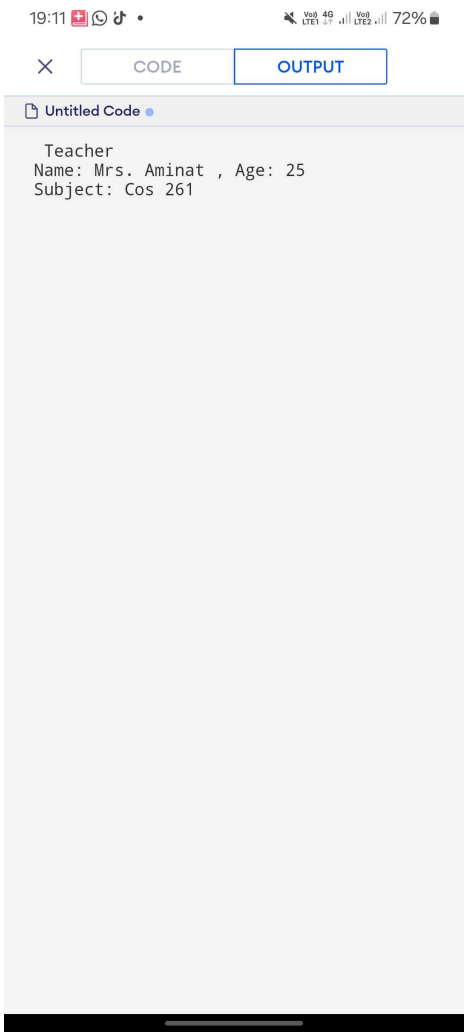
    Code example



13. What is inheritance? Create a base class Person and a subclass Teacher.

    Inheritance; is a mechanism in OOP where a new class (subclass or derived class) acquires the properties (fields) and behaviors (methods) of an existing class (superclass or base class).


19:12
72%
CODE
OUTPUT
Untitled Code
class Person {
private String name;
private int age;
public Person(String name, int age) {
this.name = name;
this.age = age;
}
public String getName() {
return name;
}
public int getAge() {
return age;
}
public void displayInfo() {
System.out.println("Name: " + name + ", Age: " + age);
}
}
class Teacher extends Person {
private String subject;
public Teacher(String name, int age, String subject) {
super(name, age);
this.subject = subject;
}
public String getSubject() {
return subject;
}
public void displayTeacherInfo() {
super.displayInfo();
System.out.println("Subject: " + subject);
}
public static void main(String[] args) {
Teacher teacher1 = new Teacher("Mrs. Aminat ", 25, "Cos 261");
teacher1.displayTeacherInfo();
}
}
Run



14. What does it mean to write "clean code"? Give 3 practices that make code clean and maintainable.

    What is Clean Code?

    Clean code is code that is easy to read, understand, and maintain. It follows best practices, is well-organized, and minimizes complexity.

    3 Practices for Clean and Maintainable Code:

a) Use Descriptive Naming:
   Use clear and concise names for variables, functions, and classes to improve readability and understanding.

b) Keep Functions Short and Focused:

Break down large functions into smaller ones, each with a single responsibility, to improve modularity and reusability.

c) Follow the DRY Principle (Don't Repeat Yourself):
Avoid duplicating code or logic by extracting common functionality into reusable functions or classes, reducing maintenance effort and errors.

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

Reasons we should avoid writing very long methods in Java programs for several reasons:

● Readability: Long methods are harder to read and understand, making it difficult for developers to comprehend the code's logic and intent.

● Maintainability: When methods are lengthy, making changes or fixing bugs becomes more complicated, increasing the risk of introducing new errors.

● Reusability: Long methods often perform multiple tasks, making it challenging to reuse them in other parts of the program.

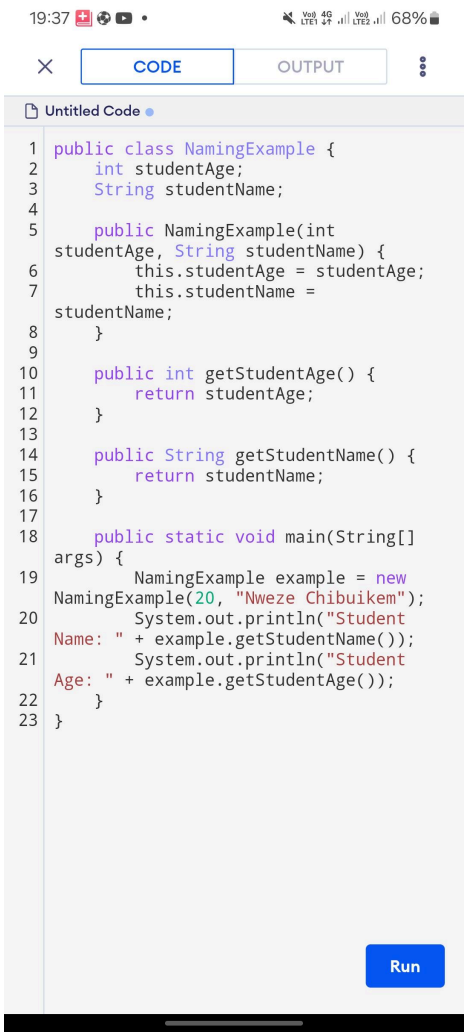
● Testability: Testing long methods can be more complex, as they may have multiple execution paths and edge cases.

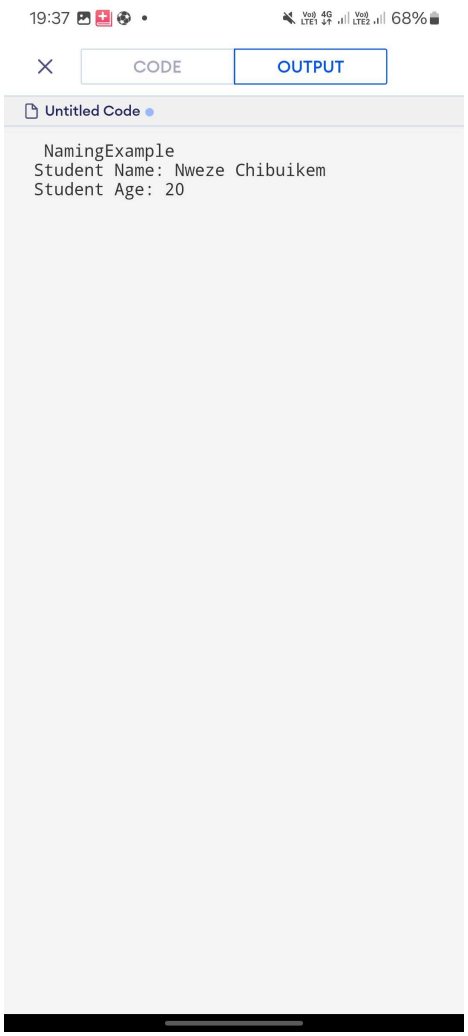
● Debugging: Identifying and fixing issues in long methods can be time-consuming due to the complexity and scope.

16. What naming conventions should be followed in Java for: Classes, Variables, Methods. Give examples with screenshot of code and output.

Naming conventions in java;

a) Classes: Use PascalCase. This means the first letter of each word in the class name is capitalized. Nouns or noun phrases are typically used for class names.
Example: public class Student, public class Car, public class ArrayList
b) Variables: Use camelCase. The first letter of the first word is lowercase, and the first letter of each subsequent word is capitalized. Nouns are typically used for variable names. Be descriptive.
Example: int studentId, String firstName, double accountBalance
c) Methods: Use camelCase. The first letter of the first word is lowercase, and the first letter of each subsequent word is capitalized. Verbs or verb phrases are typically used for method names, indicating the action the method performs.
Example: public void calculateAverage(), public String getName(), public boolean isEligible()

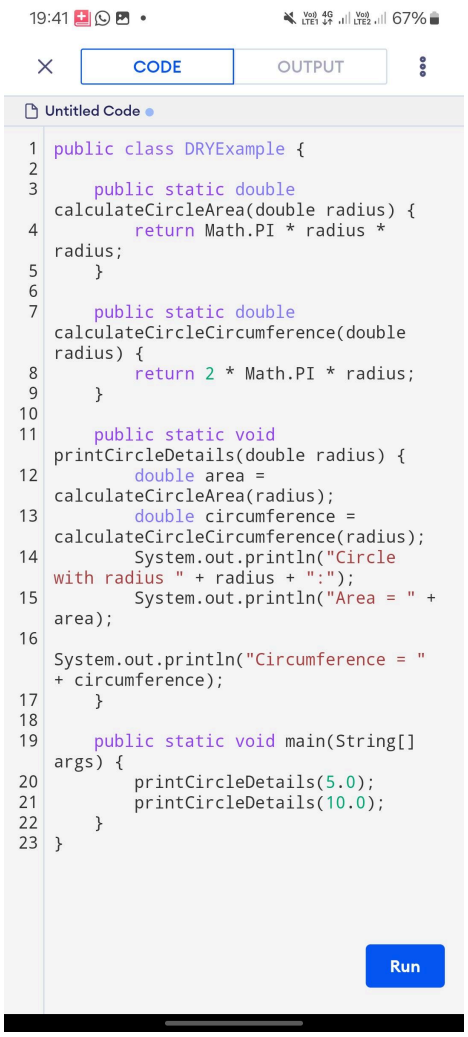

19:37
68%
CODE
OUTPUT
Untitled Code
public class NamingExample {
int studentAge;
String studentName;
public NamingExample(int studentAge, String studentName) {
this.studentAge = studentAge;
this.studentName = studentName;
}
public int getStudentAge() {
return studentAge;
}
public String getStudentName() {
return studentName;
}
public static void main(String[] args) {
NamingExample example = new NamingExample(20, "Nweze Chibuikem");
System.out.println("Student Name: " + example.getStudentName());
System.out.println("Student Age: " + example.getStudentAge());
}
}
Run

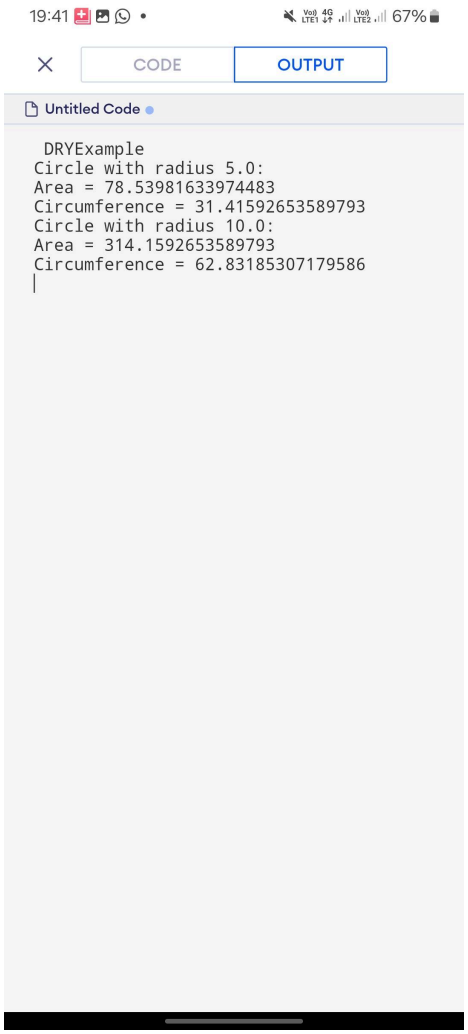


17. What is the importance of breaking your Java program into methods?

● Modularity: Methods break down a large, complex program into smaller, manageable, and logical units. Each method performs a specific task.
● Reusability: Once a method is written, it can be called multiple times from different parts of the program or even from other classes, reducing code duplication.
● Readability: Well-named methods make the code easier to understand. The overall flow of the program becomes clearer as you read the sequence of method calls rather than a long block of code.
● Maintainability: Changes or bug fixes are often localized to specific methods, making it easier to maintain and debug the code without affecting other parts of the program.
● Testability: Individual methods can be tested in isolation, making it easier to ensure the correctness of each component of the program.
● Abstraction: Methods allow you to hide the implementation details of a particular task. You only need to know what the method does, not how it does it.

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





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
    - Encapsulation
    - organisation
    - Abstraction
    - Maintainability
    - Extensibility
    - Reusability

20. Why is testing important during program development?

    Importance of testing during program development are;
    - ensuring correctness
    - improving reliability

- validating requirement
- preventing regression
- improving performance

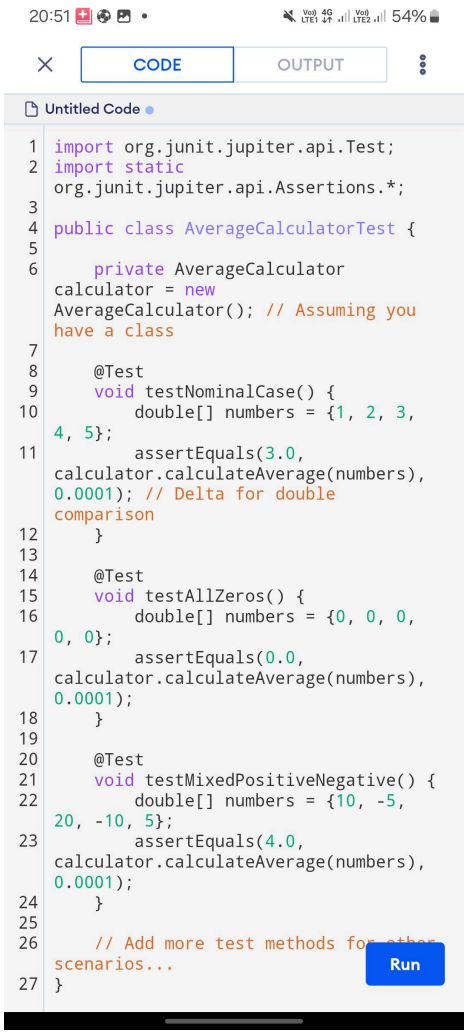
21. What is the difference between syntax error, runtime error.

a) Syntax Error:
   - Occurs when the code violates the programming language's syntax rules.
   - Typically caught by the compiler or interpreter before the code runs.
   - Examples: missing semicolons, mismatched brackets, typos in keywords.

b) Runtime Error:
   - Occurs when the code encounters an issue during execution, often due to unexpected input or environmental factors.
   - Can cause the program to crash or terminate abruptly.
   - Examples: division by zero, null pointer exceptions, out-of-memory errors.

c) Logic Error:
   - Occurs when the code runs without syntax or runtime errors but produces incorrect or unexpected results.
   - Often due to flawed algorithm design, incorrect assumptions, or misunderstandings of the problem.
   - Examples: incorrect calculations, wrong output formatting, or unexpected behavior.

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



23. Why should Java developers write comments in their code?

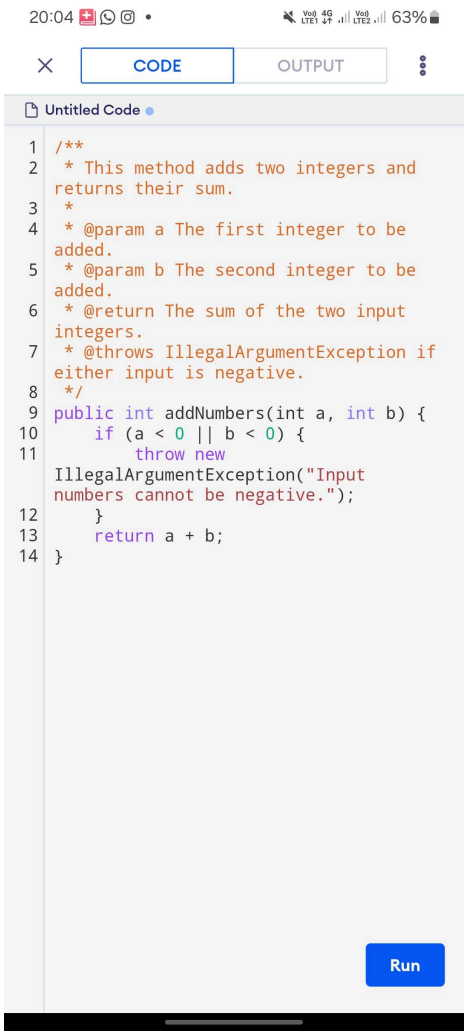
    Java developers write comments in their code for;
    - Understanding
    - Maintainability
    - clarity
    - collaboration
    - documentation generation

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

    The difference are:
    - In Purpose: Regular comments are for explaining the code to developers; JavaDoc comments are for documenting the API for users of the code.

- In Processing: Regular comments are ignored by the JavaDoc tool; JavaDoc comments are processed to generate documentation.

25. Write a sample Java method with JavaDoc comments.



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

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

The importance in team projects
- Auditing
- Rollback
- Tracking Changes
- Collaboration

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

    What is Code Refactoring?

    Code refactoring is the process of improving the structure, design, and readability of existing code without changing its functionality or behavior. It's like cleaning and organizing your room without throwing away any of your belongings.

    Why Refactor Code?

    Refactoring helps make your code:

    1. Easier to understand: Cleaner code is simpler to read and comprehend.
    2. More maintainable: Refactored code is easier to modify and update.
    3. More efficient: Improved code can run faster or use fewer resources.

    Common Refactoring Techniques:

    1. Renaming variables: Giving variables clear and descriptive names.
    2. Extracting methods: Breaking down long methods into smaller, focused ones.
    3. Removing duplicates: Eliminating repeated code or logic.

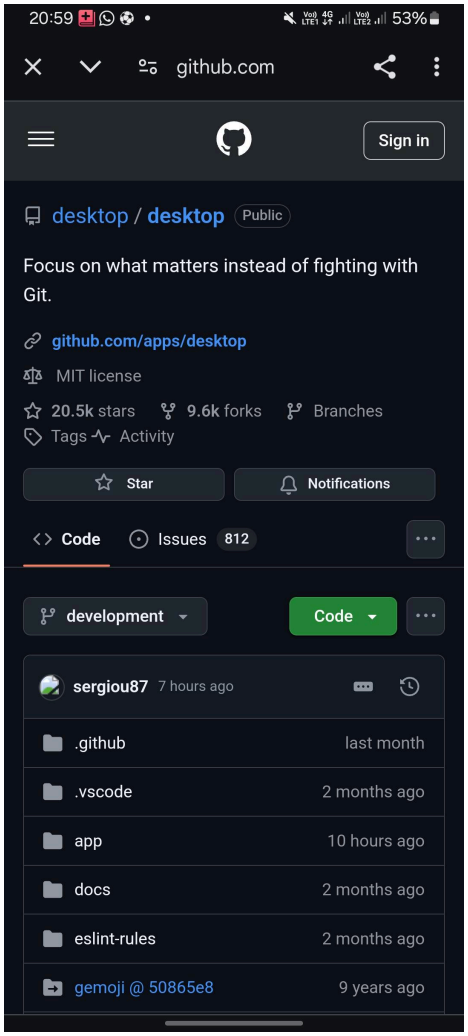
    Best Practices:

    1. Refactor incrementally: Make small changes and test regularly.
    2. Use version control: Track changes and revert if needed.
    3. Test thoroughly: Ensure refactored code works as expected.

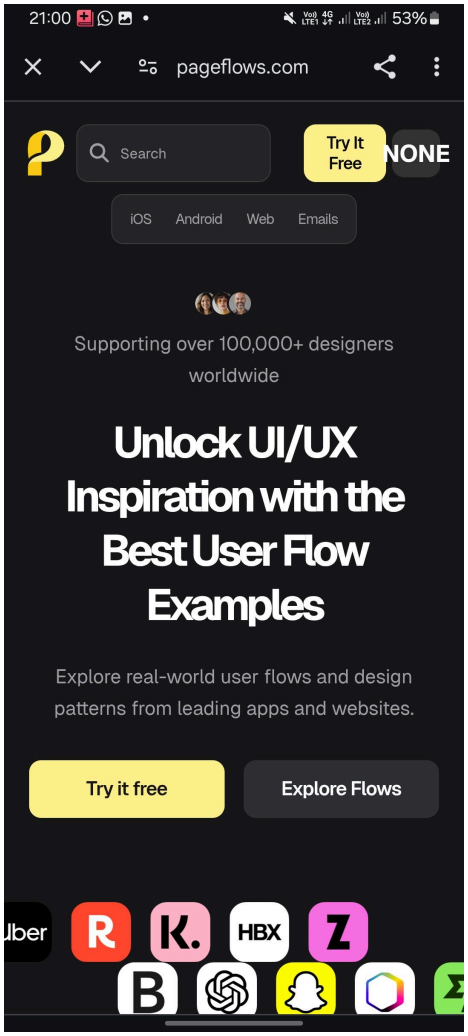
    By refactoring your code, you'll make it more efficient, readable, and maintainable, which will save you time and effort in the long run.

28. What tools can Java developers use to collaborate on large projects? Attach screenshots of 3 examples.

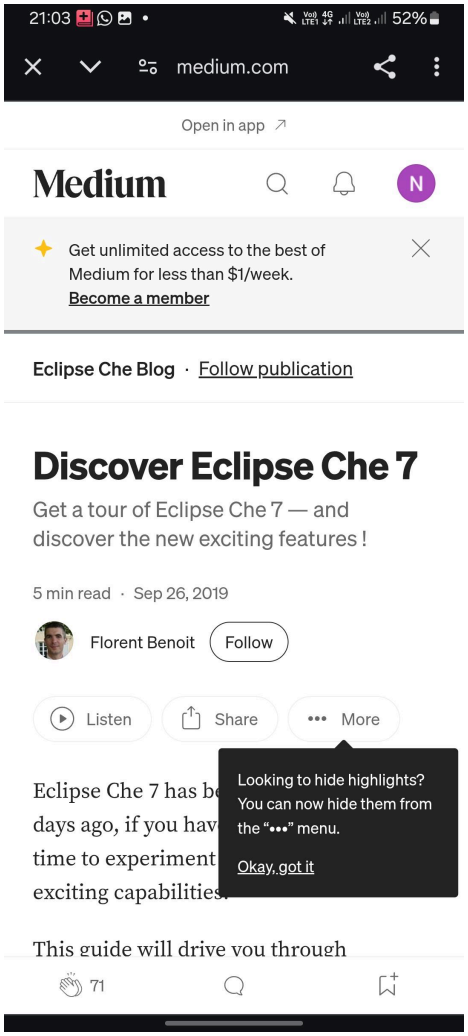
a) **GitHub:** A cloud-based platform that facilitates version control using Git. It offers features like pull requests, code reviews, and issue tracking, making it ideal for collaborative development.



b) **Jira:** Developed by Atlassian, Jira is a project management tool that supports Agile methodologies. It provides features such as Scrum and Kanban boards, issue tracking, and customizable workflows.



c) **Eclipse Che:** An open-source cloud IDE that allows multiple developers to work on the same codebase simultaneously. It supports collaborative editing and integrates with various development tools



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

    5 best practices in java program are:

a) Write clean and readable code
b) Follow naming Convention
c) Embrace object-oriented principles
d) Write unit tests
e) Handle exceptions properly

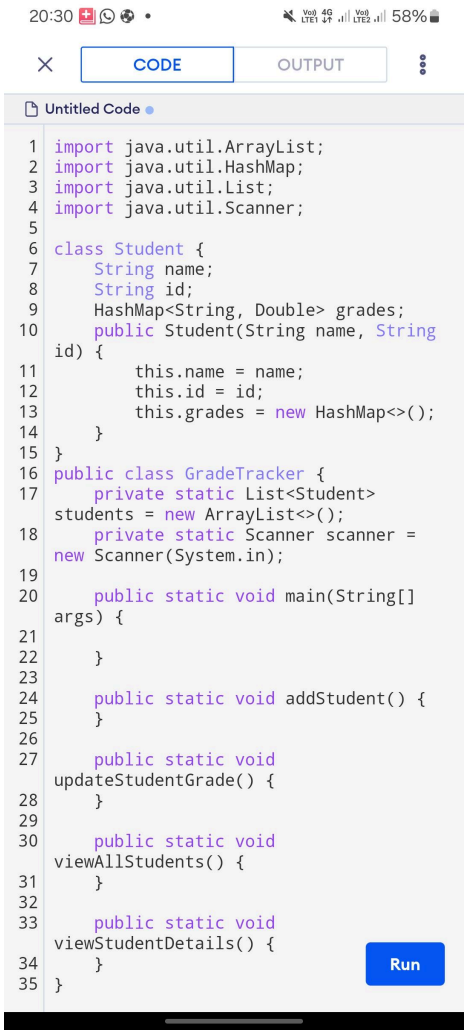
30. What is code readability, and why is it more important than "smart" code?

    Code readability refers to how easy it is for humans to understand the purpose, logic, and flow of the code

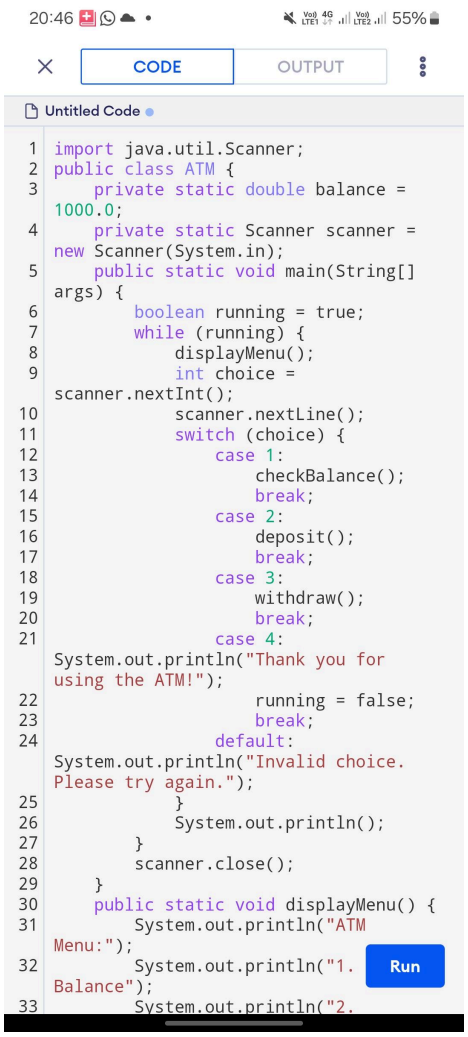
Code readability is more important than “smart” code is;

- Easier maintenance
- Fewer errors
- Better collaboration
- Long term Benefits

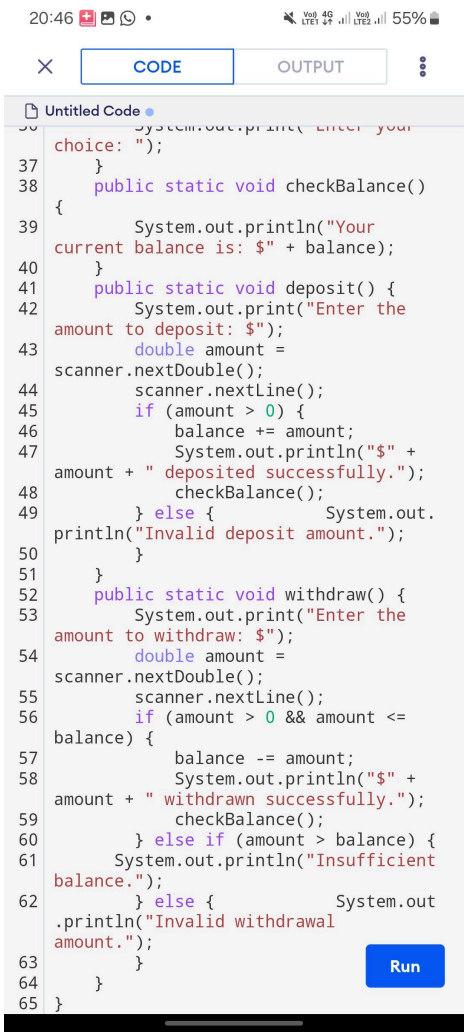
31. Build a command-line application that keeps track of student grades and allows adding, updating, and viewing records.

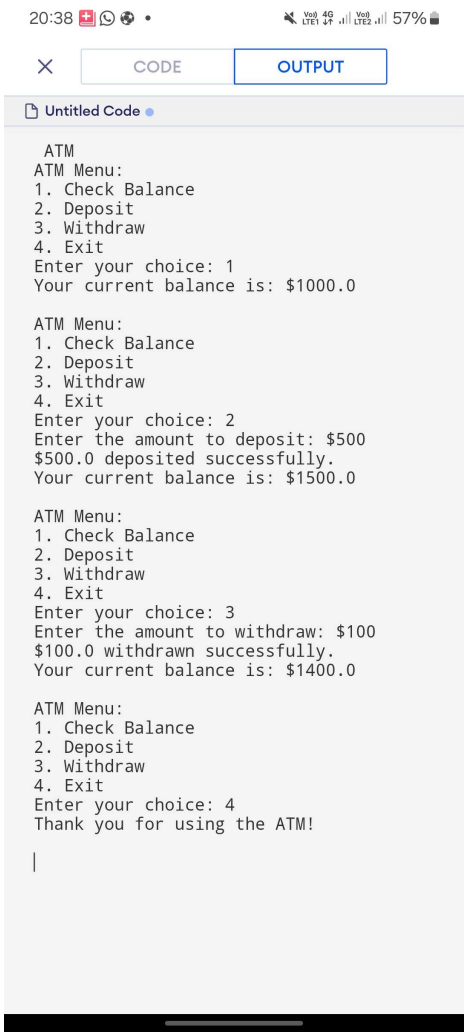


32. Write a program that simulates a basic ATM system (check balance, deposit, withdraw).




```java
import java.util.Scanner;
public class ATM {
    private static double balance = 1000.0;
    private static Scanner scanner = new Scanner(System.in);
    public static void main(String[] args) {
        boolean running = true;
        while (running) {
            displayMenu();
            int choice = scanner.nextInt();
            scanner.nextLine();
            switch (choice) {
                case 1:
                    checkBalance();
                    break;
                case 2:
                    deposit();
                    break;
                case 3:
                    withdraw();
                    break;
                case 4:
System.out.println("Thank you for using the ATM!");
                    running = false;
                    break;
                default:
System.out.println("Invalid choice. Please try again.");
            }
            System.out.println();
        }
        scanner.close();
    }
    public static void displayMenu() {
        System.out.println("ATM Menu:");
        System.out.println("1. Balance");
        System.out.println("2.
```


20:46
55%
CODE
OUTPUT
Untitled Code
choice: ");
}
public static void checkBalance()
{
System.out.println("Your
current balance is: $" + balance);
}
public static void deposit() {
System.out.print("Enter the
amount to deposit: $");
double amount =
scanner.nextDouble();
scanner.nextLine();
if (amount > 0) {
balance += amount;
System.out.println("$" +
amount + " deposited successfully.");
checkBalance();
} else { System.out.
println("Invalid deposit amount.");
}
}
public static void withdraw() {
System.out.print("Enter the
amount to withdraw: $");
double amount =
scanner.nextDouble();
scanner.nextLine();
if (amount > 0 && amount <=
balance) {
balance -= amount;
System.out.println("$" +
amount + " withdrawn successfully.");
checkBalance();
} else if (amount > balance) {
System.out.println("Insufficient
balance.");
} else { System.out
.println("Invalid withdrawal
amount.");
}
Run
}
}


20:38
57%
CODE
OUTPUT
Untitled Code
ATM
ATM Menu:
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter your choice: 1
Your current balance is: $1000.0
ATM Menu:
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter your choice: 2
Enter the amount to deposit: $500
$500.0 deposited successfully.
Your current balance is: $1500.0
ATM Menu:
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter your choice: 3
Enter the amount to withdraw: $100
$100.0 withdrawn successfully.
Your current balance is: $1400.0
ATM Menu:
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter your choice: 4
Thank you for using the ATM!