

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

Cos 261 test

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

The screenshot shows a mobile device interface with two open applications side-by-side. The left application is a code editor titled 'Untitled Code' containing the following Java code:

```
1 public class HelloWorld {  
2     public static void main(String[]  
3         args) {  
4             System.out.println("Hello,  
5                 World!");  
6         }  
7     }
```

The right application is a terminal window titled 'Untitled Code' showing the output of the code execution:

```
HelloWorld  
Hello, World!
```

Both applications have a top status bar showing the time (17:33), signal strength, battery level (89%), and connectivity (LTE1, LTE2). The code editor has tabs for 'CODE' and 'OUTPUT', and the terminal has tabs for 'CODE' and 'OUTPUT'.

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.

The screenshot shows a Java code editor interface with the following details:

- Top status bar: 17:51, battery 4G, signal strength, 86%.
- Toolbar: X, CODE (highlighted), OUTPUT, More.
- Title bar: Untitled Code.
- Code area:

```
1 public class EqualityExample {
2     public static void main(String[]
args) {
3         String str1 = "hello";
4         String str2 = "hello";
5         String str3 = new
String("hello");
6         Integer num1 = 10;
7         Integer num2 = 10;
8         Integer num3 = new
Integer(10);
9
10        System.out.println("String
comparison:");
11        System.out.println("str1 ==
str2: " + (str1 == str2));
12
System.out.println("str1.equals(str2):
" + str1.equals(str2));
13        System.out.println("str1 ==
str3: " + (str1 == str3));
14
System.out.println("str1.equals(str3):
" + str1.equals(str3));
15
16        System.out.println("\nInteger
comparison:");
17        System.out.println("num1 ==
num2: " + (num1 == num2));
18
System.out.println("num1.equals(num2):
" + num1.equals(num2));
19        System.out.println("num1 ==
num3: " + (num1 == num3));
20
System.out.println("num1.equals(num3):
" + num1.equals(num3));
21    }
22 }
```
- Run button: A blue button labeled "Run".

A screenshot of a mobile device's screen displaying a code editor. The top status bar shows the time as 17:51, signal strength, battery level at 86%, and connectivity to VoIP, 4G, LTE1, and LTE2. Below the status bar is a navigation bar with 'CODE' and 'OUTPUT' tabs, where 'CODE' is selected. The main area is titled 'Untitled Code'. It contains two snippets of Java code:

```
EqualityExample
String comparison:
str1 == str2: true
str1.equals(str2): true
str1 == str3: false
str1.equals(str3): true

Integer comparison:
num1 == num2: true
num1.equals(num2): true
num1 == num3: false
num1.equals(num3): true
```

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 + 4G • LTE1 4G LTE2 85% ■

X CODE OUTPUT •

Untitled Code ●

```
5     Scanner input = new
6     Scanner(System.in);
7     System.out.print("Enter the
8         first number: ");
9         int num1 = input.nextInt();
10    System.out.print("Enter the
11        second number: ");
12        int num2 = input.nextInt();
13    int sum = num1 + num2;
14    System.out.println("The sum
15        is: " + sum);
16    input.close();
17 }
18 }
```

Run

A screenshot of a mobile application interface. At the top, there is a status bar with the time "17:57", signal strength, battery level at "85%", and other icons. Below the status bar is a navigation bar with three tabs: "CODE" (which is selected and highlighted in blue), "OUTPUT", and a third tab which is mostly obscured by a large white area. The main content area is titled "Untitled Code" and contains the following text:

```
AddTwoNumbers
Enter the first number: 5
Enter the second number: 10
The sum is: 15
```

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.

The screenshot shows a mobile application interface for writing and running Java code. At the top, there is a status bar with the time "18:08", signal strength, battery level at 83%, and other icons. Below the status bar is a navigation bar with tabs for "CODE" (which is selected) and "OUTPUT". To the right of the tabs is a three-dot menu icon. The main area is titled "Untitled Code" with a blue circular icon. The code itself is a Java program:

```
1 import java.util.Scanner;
2
3 public class EvenOdd {
4     public static void main(String[] args) {
5         Scanner input = new Scanner(System.in);
6
7         System.out.print("Enter a
number: ");
8         int number = input.nextInt();
9
10        if (number % 2 == 0) {
11            System.out.println(number
+ " is even.");
12        } else {
13            System.out.println(number
+ " is odd.");
14        }
15
16        input.close();
17    }
18 }
```

At the bottom of the code editor is a blue "Run" button. Below the editor is a black horizontal bar.

18:08 + 🔍 🔍 • 🔍 VoIP 4G LTE1 4G LTE2 83%

X CODE

OUTPUT

Untitled Code

```
EvenOdd  
Enter a number: 9  
9 is odd.
```

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

18:10 • 4G • 83%

X CODE OUTPUT

Untitled Code

```
1 import java.util.Scanner;
2
3 public class LargestOfThree {
4     public static void main(String[]
5 args) {
6         Scanner input = new
7 Scanner(System.in);
8
9         System.out.print("Enter the
10 first number: ");
11         int num1 = input.nextInt();
12
13         System.out.print("Enter the
14 second number: ");
15         int num2 = input.nextInt();
16
17         System.out.print("Enter the
18 third number: ");
19         int num3 = input.nextInt();
20
21         int largest;
22
23         if (num1 >= num2 && num1 >=
24 num3) {
25             largest = num1;
26         } else if (num2 >= num1 &&
27 num2 >= num3) {
28             largest = num2;
29         } else {
30             largest = num3;
31         }
32
33         System.out.println("The
34 largest number is: " + largest);
35
36         input.close();
37     }
38 }
```

Run

A screenshot of a mobile device's screen. At the top, there is a status bar with icons for signal strength, battery level (83%), and time (18:11). Below the status bar is a navigation bar with three tabs: 'CODE' (which is greyed out), 'OUTPUT' (which is highlighted in blue), and a third tab that is mostly obscured. The main area is a code editor titled 'Untitled Code'. It contains a Java program named 'LargestOfThree'. The code reads three numbers from the user and prints out the largest one. The output window shows the execution of the code and its results.

```
LargestOfThree
Enter the first number: 44
Enter the second number: 12
Enter the third number: 45
The largest number is: 45
```

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 4G 79%

X 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

18:30 4G VoIP LTE1 LTE2 79%

The screenshot shows a mobile application interface. At the top, there are status icons for signal strength, battery level at 79%, and connectivity. Below the status bar is a navigation bar with three items: a close button ('X'), a 'CODE' tab, and an 'OUTPUT' tab. The 'OUTPUT' tab is currently selected and highlighted in blue. Below the navigation bar is a title bar labeled 'Untitled Code'. The main content area contains the following text:

```
Whileloop
0
1
2
3
4
```

- 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 + ⊖ •

VoIP 4G LTE1 77% VoIP LTE2 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

18:46 • 4G • 77%

A screenshot of a mobile phone's screen displaying a code editor. The top status bar shows the time as 18:46, signal strength, battery level at 77%, and network connectivity. Below the status bar is a navigation bar with three buttons: a close button (X), a 'CODE' button, and a 'OUTPUT' button. The 'OUTPUT' button is highlighted with a blue border. The main area of the screen is titled 'Untitled Code' with a small blue circular icon. The code itself is a single-line script named 'forloop' containing the numbers 0, 1, 2, 3, and 4, each on a new line.

```
forloop
0
1
2
3
4
```

- 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.

The image displays two side-by-side screenshots of a mobile application interface for writing and running Java code. Both screens show the same code structure: a do-while loop that prints integers from 0 to 4. The left screen shows the code in the 'CODE' tab, and the right screen shows the output in the 'OUTPUT' tab.

Left Screen (CODE Tab):

```
1 import java.util.Scanner;
2
3 public class doWhileloop {
4     public static void main(String[]
5 args) {
6     int i = 0;
7     do {
8         System.out.println(i);
9         i++;
10    }
11 }
```

Right Screen (OUTPUT Tab):

```
doWhileloop
0
1
2
3
4
```

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

18:52 76% •

The screenshot shows a mobile application interface for writing and running Java code. At the top, there are status icons for signal strength, battery level at 76%, and connectivity. Below the status bar is a navigation bar with tabs for 'CODE' (which is selected) and 'OUTPUT'. The main area is titled 'Untitled Code' and contains the following Java code:

```
1 import java.util.Scanner;
2
3 public class MultiplicationTable {
4     public static void main(String[]
5         args) {
6         Scanner input = new
7             Scanner(System.in);
8
9         System.out.print("Enter a
10    number: ");
11        int number = input.nextInt();
12
13        System.out.print("Enter the
14    limit: ");
15        int limit = input.nextInt();
16
17        System.out.println("Multiplication
18            Table for " + number + " up to " +
19            limit + ":" );
20        for (int i = 1; i <= limit; i+
21        +) {
22            System.out.println(number
23            + " x " + i + " = " + (number * i));
24        }
25
26        input.close();
27    }
28 }
```

At the bottom of the code editor is a blue 'Run' button.

The screenshot shows a mobile device interface. At the top, there are status icons for battery level (76%), signal strength, and network connectivity (VoIP 4G, LTE1, LTE2). Below the status bar is a header with tabs: 'CODE' (selected) and 'OUTPUT'. The main area is titled 'Untitled Code' and contains the following Java code:

```
MultiplicationTable
Enter a number: 7
Enter the limit: 10
Multiplication Table for 7 up to 10:
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70
```

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.

The image shows two side-by-side screenshots of a mobile application interface. Both screens have a top status bar showing signal strength, battery level at 74%, and the time as 19:00.

Left Screen (CODE):

Untitled Code

```
1 public class Student {  
2     private String name;  
3     private String matricNo;  
4     private double score;  
5  
6     public Student(String name, String  
7         matricNo, double score) {  
8         this.name = name;  
9         this.matricNo = matricNo;  
10        this.score = score;  
11    }  
12  
13    public String getName() {  
14        return name;  
15    }  
16  
17    public String getMatricNo() {  
18        return matricNo;  
19    }  
20  
21    public double getScore() {  
22        return score;  
23    }  
24  
25    public void displayInfo() {  
26        System.out.println("Name: " +  
27            name);  
28        System.out.println("Matric.Number: " +  
29            matricNo);  
30        System.out.println("Score: " +  
31            score);  
32    }  
33  
34 }
```

A blue "Run" button is located at the bottom right of the code editor.

Right Screen (OUTPUT):

Untitled Code

```
Student  
Name: Nweze Chibukem  
Matric.Number: 2023/253020  
Score: 70.0
```

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

The image shows two side-by-side screenshots of a mobile application interface. Both screens have a top status bar showing signal strength, battery level at 74%, and the time (19:03 on the left, 19:04 on the right). Below the status bar is a header with 'CODE' and 'OUTPUT' tabs, both of which are currently selected.

Left Screen (Code View):

```
1 public class Calculator {  
2     public int add(int a, int b) {  
3         System.out.println("Adding two  
4             integers");  
5         return a + b;  
6     }  
7     public double add(double a, double  
8         b) {  
9         System.out.println("Adding two  
10            doubles");  
11         return a + b;  
12     }  
13     public int add(int a, int b, int  
14         c) {  
15         System.out.println("Adding  
16             three integers");  
17         return a + b + c;  
18     }  
19     public static void main(String[]  
20         args) {  
21         Calculator calc = new  
22             Calculator();  
23         System.out.println(calc.add(5,  
24             10));  
25         System.out.println(calc.add(3.5,  
26             2.7));  
27         System.out.println(calc.add(1,  
28             2, 3));  
29     }  
30 }
```

Right Screen (Output View):

```
Calculator  
Adding two integers  
15  
Adding two doubles  
6.2  
Adding three integers  
6
```

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 + 4G 72%

X CODE OUTPUT

Untitled Code

```
1 class Person {
2     private String name;
3     private int age;
4     public Person(String name, int
age) {
5         this.name = name;
6         this.age = age;
7     }
8     public String getName() {
9         return name;
10    }
11    public int getAge() {
12        return age;
13    }
14    public void displayInfo() {
15        System.out.println("Name: " +
name + ", Age: " + age);
16    }
17 }
18 class Teacher extends Person {
19     private String subject;
20     public Teacher(String name, int
age, String subject) {
21         super(name, age);
22         this.subject = subject;
23     }
24     public String getSubject() {
25         return subject;
26     }
27     public void displayTeacherInfo() {
28         super.displayInfo();
29         System.out.println("Subject: " +
+ subject);
30     }
31     public static void main(String[]
args) {
32         Teacher teacher1 = new
Teacher("Mrs. Aminat ", 25, "Cos
261");
33         teacher1.displayTeacherInfo();
34     }
35 }
```

Run

The screenshot shows a mobile application interface. At the top, there are status icons for battery level (72%), signal strength, and network connectivity (VoLTE 4G). Below the status bar, there are two tabs: "CODE" and "OUTPUT". The "OUTPUT" tab is selected, indicated by a blue border. The content area is titled "Untitled Code" and contains the following text:
Teacher
Name: Mrs. Aminat , Age: 25
Subject: Cos 261

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 VoIP 4G LTE1 VoIP LTE2 68%

X CODE OUTPUT

Untitled Code

```
1 public class NamingExample {
2     int studentAge;
3     String studentName;
4
5     public NamingExample(int
studentAge, String studentName) {
6         this.studentAge = studentAge;
7         this.studentName =
studentName;
8     }
9
10    public int getStudentAge() {
11        return studentAge;
12    }
13
14    public String getStudentName() {
15        return studentName;
16    }
17
18    public static void main(String[]
args) {
19        NamingExample example = new
NamingExample(20, "Nweze Chibukem");
20        System.out.println("Student
Name: " + example.getStudentName());
21        System.out.println("Student
Age: " + example.getStudentAge());
22    }
23 }
```

Run

The screenshot shows a mobile application interface. At the top, there is a status bar with the time '19:37', signal strength, and battery level '68%'. Below the status bar, there are two tabs: 'CODE' and 'OUTPUT'. The 'CODE' tab is selected, showing a file named 'Untitled Code' with the following content:

```
NamingExample
Student Name: Nweze Chibukem
Student Age: 20
```

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.

The screenshot shows a Java code editor interface. At the top, there are status bars for battery level (67%), signal strength, and network connection. Below the status bar, there are tabs labeled "CODE" and "OUTPUT". The "CODE" tab is selected. The code itself is titled "Untitled Code".

```
1 public class DRYExample {
2     public static double calculateCircleArea(double radius) {
3         return Math.PI * radius *
4             radius;
5     }
6     public static double calculateCircleCircumference(double
7         radius) {
8         return 2 * Math.PI * radius;
9     }
10    public static void printCircleDetails(double radius) {
11        double area =
12            calculateCircleArea(radius);
13        double circumference =
14            calculateCircleCircumference(radius);
15        System.out.println("Circle
16 with radius " + radius + ":");
17        System.out.println("Area = " +
18            area);
19    }
20    public static void main(String[]
21        args) {
22        printCircleDetails(5.0);
23        printCircleDetails(10.0);
24    }
}
```

At the bottom right of the code editor is a blue "Run" button.

A screenshot of a mobile device's screen. At the top, there is a status bar with icons for signal strength, battery level (67%), and time (19:41). Below the status bar is a navigation bar with three items: 'X', 'CODE' (which is currently selected), and 'OUTPUT'. Underneath the navigation bar is a code editor window titled 'Untitled Code'. The code in the editor is as follows:

```
DRYExample
Circle with radius 5.0:
Area = 78.53981633974483
Circumference = 31.41592653589793
Circle with radius 10.0:
Area = 314.1592653589793
Circumference = 62.83185307179586
```

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?

The screenshot shows a mobile application interface with a status bar at the top indicating time (20:51), signal strength, battery level (54%), and connectivity (VoIP, LTE1, LTE2). Below the status bar is a header with 'CODE' and 'OUTPUT' tabs, and a three-dot menu icon. The main area is titled 'Untitled Code'. It contains Java code for testing an AverageCalculator class using JUnit Jupiter. The code includes three test methods: testNominalCase(), testAllZeros(), and testMixedPositiveNegative(). Each test uses assertEquals() to verify the average calculation against expected values within a delta of 0.0001. A 'Run' button is visible at the bottom right of the code area.

```
1 import org.junit.jupiter.api.Test;
2 import static org.junit.jupiter.api.Assertions.*;
3
4 public class AverageCalculatorTest {
5     private AverageCalculator
6         calculator = new
7             AverageCalculator(); // Assuming you
8             have a class
9
10    @Test
11    void testNominalCase() {
12        double[] numbers = {1, 2, 3,
13            4, 5};
14        assertEquals(3.0,
15            calculator.calculateAverage(numbers),
16            0.0001); // Delta for double
17        comparison
18    }
19
20    @Test
21    void testAllZeros() {
22        double[] numbers = {0, 0, 0,
23            0, 0};
24        assertEquals(0.0,
25            calculator.calculateAverage(numbers),
26            0.0001);
27    }
28
29    @Test
30    void testMixedPositiveNegative() {
31        double[] numbers = {10, -5,
32            20, -10, 5};
33        assertEquals(4.0,
34            calculator.calculateAverage(numbers),
35            0.0001);
36    }
37
38    // Add more test methods for other
39    scenarios...
40 }
```

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

Java developers write comments in their code for;

- Understanding
- Maintainability
- clarity
- collaboration
- documentation generation

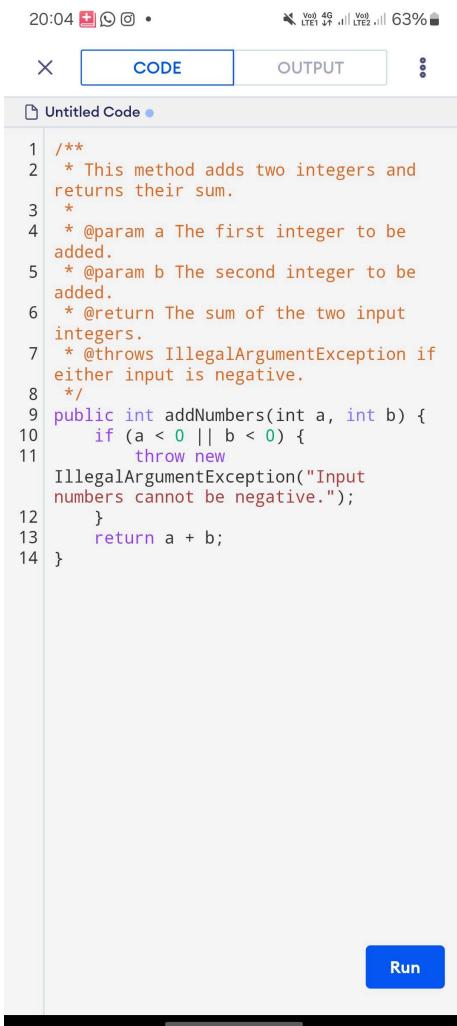
24. What are JavaDoc comments and how are they different from regular comments?

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.



The screenshot shows a code editor window titled "Untitled Code". The "CODE" tab is selected. The code is a Java method named "addNumbers" that adds two integers. It includes JavaDoc comments with @param and @return annotations, and a @throws annotation for negative input. The code editor has a "Run" button at the bottom.

```
20:04 20:46 63%  
X CODE OUTPUT :  
  
Untitled Code  
1  /**  
2   * This method adds two integers and  
3   * returns their sum.  
4   *  
5   * @param a The first integer to be  
6   * added.  
7   * @param b The second integer to be  
8   * added.  
9   * @return The sum of the two input  
10  * integers.  
11  * @throws IllegalArgumentException if  
12  * either input is negative.  
13  */  
14  public int addNumbers(int a, int b) {  
15      if (a < 0 || b < 0) {  
16          throw new  
17          IllegalArgumentException("Input  
18          numbers cannot be negative.");  
19      }  
20      return a + b;  
21  }
```

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.

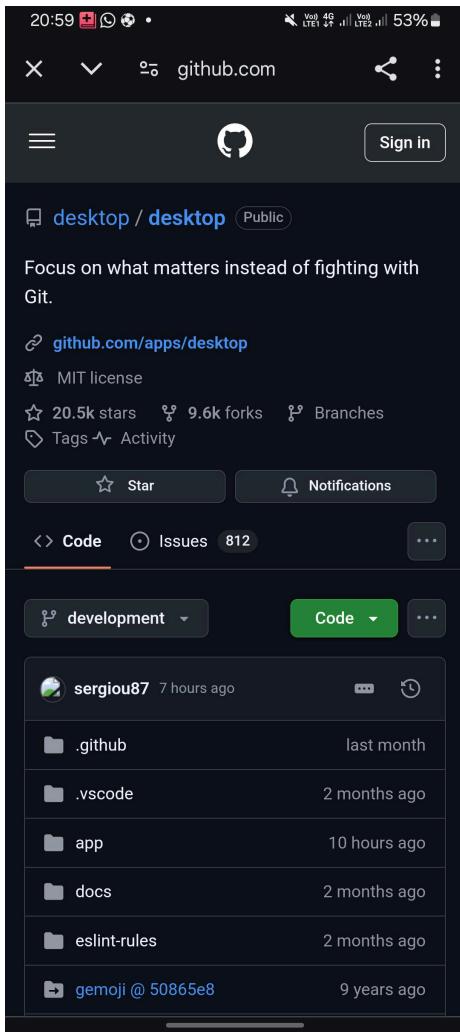
Best Practices:

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.

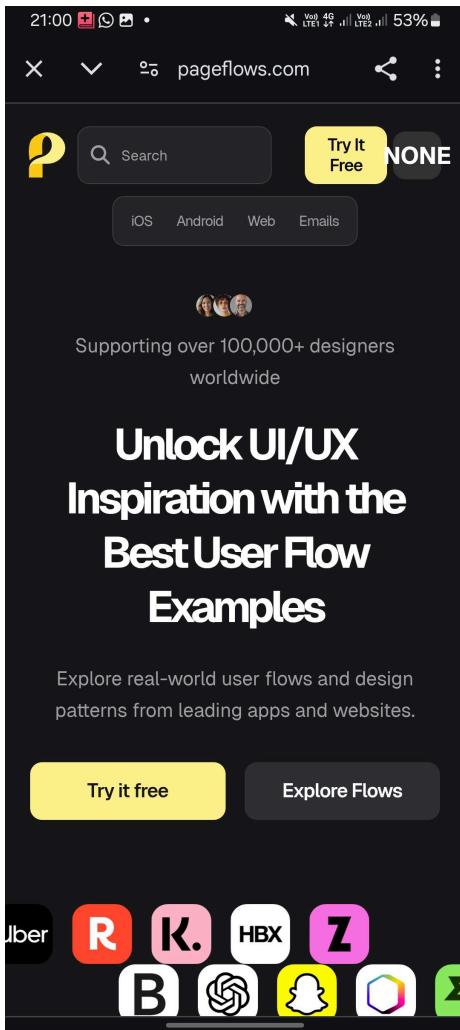
By refactoring your code, you'll make it more efficient, readable, and maintainable, which will save you time and effort in the long run.

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

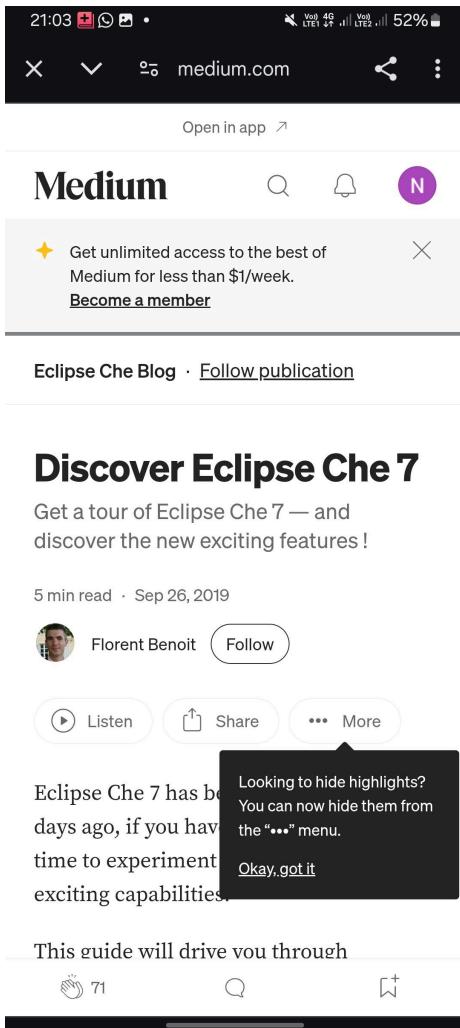
- a) **GitHub:** A cloud-based platform that facilitates version control using Git. It offers features like pull requests, code reviews, and issue tracking, making it ideal for collaborative development.



- 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.

The screenshot shows a Java code editor interface. At the top, there are status bars for battery level (58%), signal strength, and network connection. Below the status bar is a navigation bar with tabs for 'CODE' (which is selected) and 'OUTPUT'. The main area is titled 'Untitled Code'. The code itself is a Java program named 'GradeTracker'. It starts with imports for ArrayList, HashMap, List, and Scanner. The 'Student' class has fields for name and id, and a constructor that initializes a grades map. The 'GradeTracker' class has static fields for a list of students and a scanner. It contains four static methods: addStudent(), updateStudentGrade(), viewAllStudents(), and viewStudentDetails(). A 'Run' button is located at the bottom right of the code area.

```
1 import java.util.ArrayList;
2 import java.util.HashMap;
3 import java.util.List;
4 import java.util.Scanner;
5
6 class Student {
7     String name;
8     String id;
9     HashMap<String, Double> grades;
10    public Student(String name, String
11        id) {
12        this.name = name;
13        this.id = id;
14        this.grades = new HashMap<>();
15    }
16    public class GradeTracker {
17        private static List<Student>
18        students = new ArrayList<>();
19        private static Scanner scanner =
20        new Scanner(System.in);
21        public static void main(String[]
22            args) {
23        }
24        public static void addStudent() {
25        }
26        public static void
27        updateStudentGrade() {
28        }
29        public static void
30        viewAllStudents() {
31        }
32        public static void
33        viewStudentDetails() {
34        }
35    }

```

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

20:46 VoIP 4G LTE1 VoIP LTE2 55%

X CODE OUTPUT

Untitled Code

```
1 import java.util.Scanner;
2 public class ATM {
3     private static double balance =
4         1000.0;
5     private static Scanner scanner =
6         new Scanner(System.in);
7     public static void main(String[]
8         args) {
9         boolean running = true;
10        while (running) {
11            displayMenu();
12            int choice =
13                scanner.nextInt();
14            scanner.nextLine();
15            switch (choice) {
16                case 1:
17                    checkBalance();
18                    break;
19                case 2:
20                    deposit();
21                    break;
22                case 3:
23                    withdraw();
24                    break;
25                case 4:
26                    System.out.println("Thank you for
27                     using the ATM!");
28                    running = false;
29                    break;
30                default:
31                    System.out.println("Invalid choice.
32                     Please try again.");
33            }
34            System.out.println();
35        }
36        scanner.close();
37    }
38    public static void displayMenu() {
39        System.out.println("ATM
40             Menu:");
41        System.out.println("1. Run
42             Balance");
43        System.out.println("2.
44             Withdraw");
45    }
46    public static void checkBalance() {
47        System.out.println("Your current balance is
48             $" + balance);
49    }
50    public static void deposit() {
51        System.out.println("Enter the amount to deposit:
52             ");
52        double amount =
53            scanner.nextDouble();
54        balance += amount;
55        System.out.println("Deposit successful. Your new
56             balance is $" + balance);
57    }
58    public static void withdraw() {
59        System.out.println("Enter the amount to withdraw:
60             ");
61        double amount =
62            scanner.nextDouble();
63        if (amount > balance) {
64            System.out.println("Insufficient funds. Your
65             balance is $" + balance);
66        } else {
67            balance -= amount;
68            System.out.println("Withdrawal successful. Your new
69             balance is $" + balance);
70        }
71    }
72}
```

20:46 VoIP 4G LTE1 VoIP LTE2 55%

X CODE OUTPUT

```
Untitled Code
36     System.out.print("Enter your
37     choice: ");
38     public static void checkBalance()
39     {
40         System.out.println("Your
41         current balance is: $" + balance);
42     }
43     public static void deposit() {
44         System.out.print("Enter the
45         amount to deposit: $");
46         double amount =
47             scanner.nextDouble();
48         scanner.nextLine();
49         if (amount > 0) {
50             balance += amount;
51             System.out.println("$" +
52             amount + " deposited successfully.");
53             checkBalance();
54         } else {
55             System.out.
56             println("Invalid deposit amount.");
57         }
58     }
59     public static void withdraw() {
60         System.out.print("Enter the
61         amount to withdraw: $");
62         double amount =
63             scanner.nextDouble();
64         scanner.nextLine();
65         if (amount > 0 && amount <=
balance) {
66             balance -= amount;
67             System.out.println("$" +
amount + " withdrawn successfully.");
68             checkBalance();
69         } else if (amount > balance) {
70             System.out.println("Insufficient
balance.");
71         } else {
72             System.out
73             .println("Invalid withdrawal
amount.");
74         }
75     }
76 }
```

Run

20:38 4G • 57%

X 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!
```