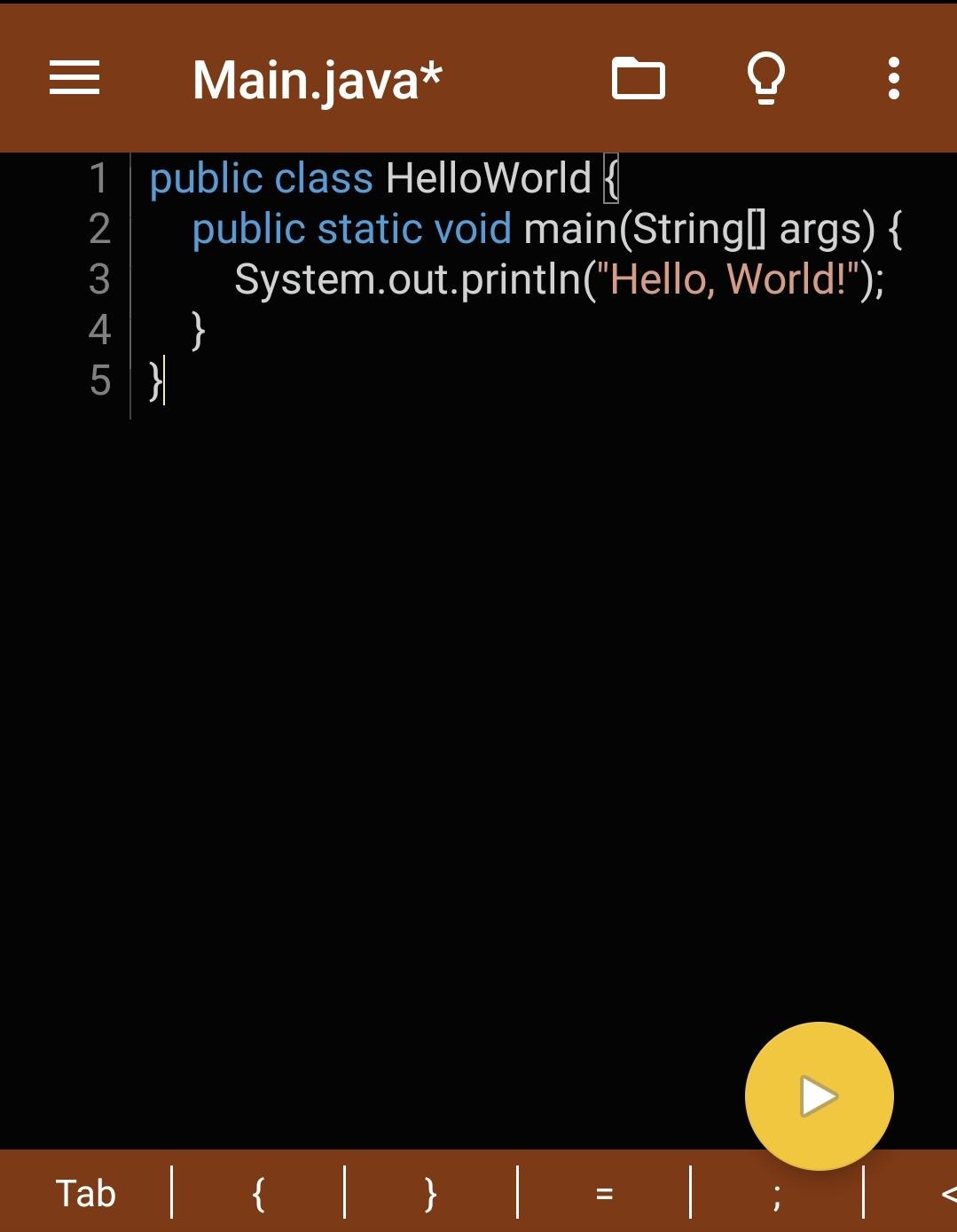
Basics and Syntax

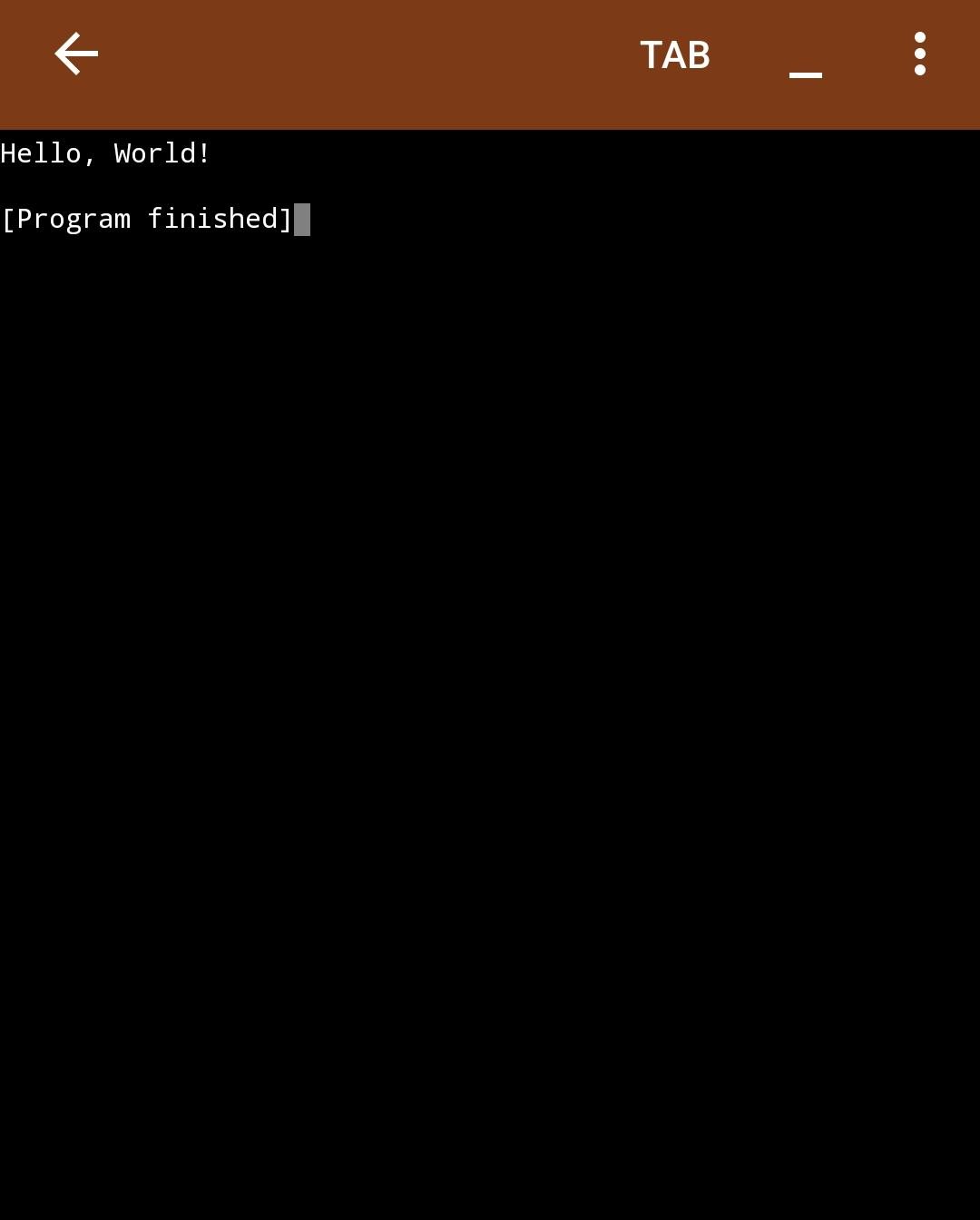
Question 1:

Printing Hello World

Program:



Output:



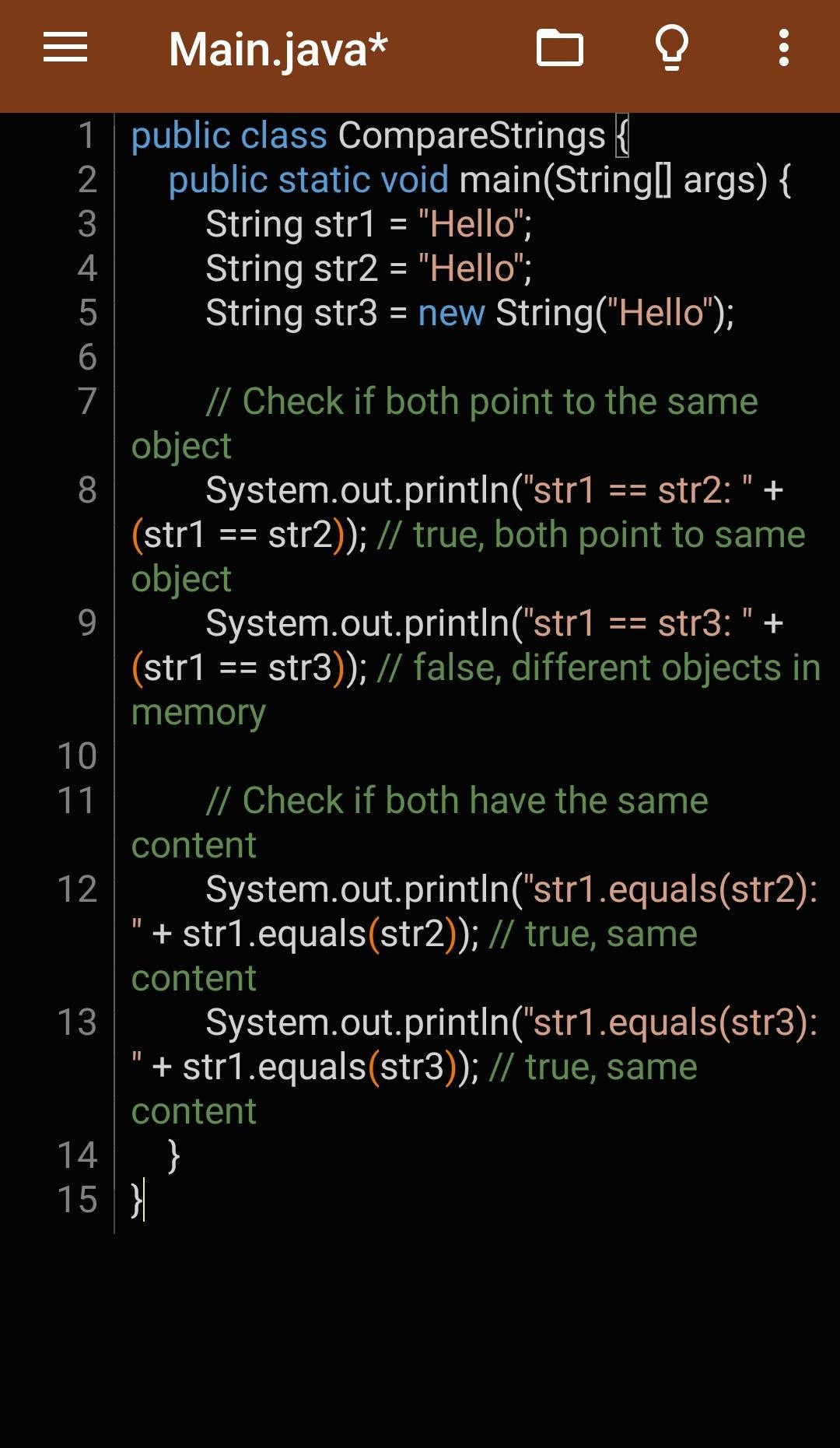
Question 2:

Difference Between == and .equals() (with code + output) What is the difference?

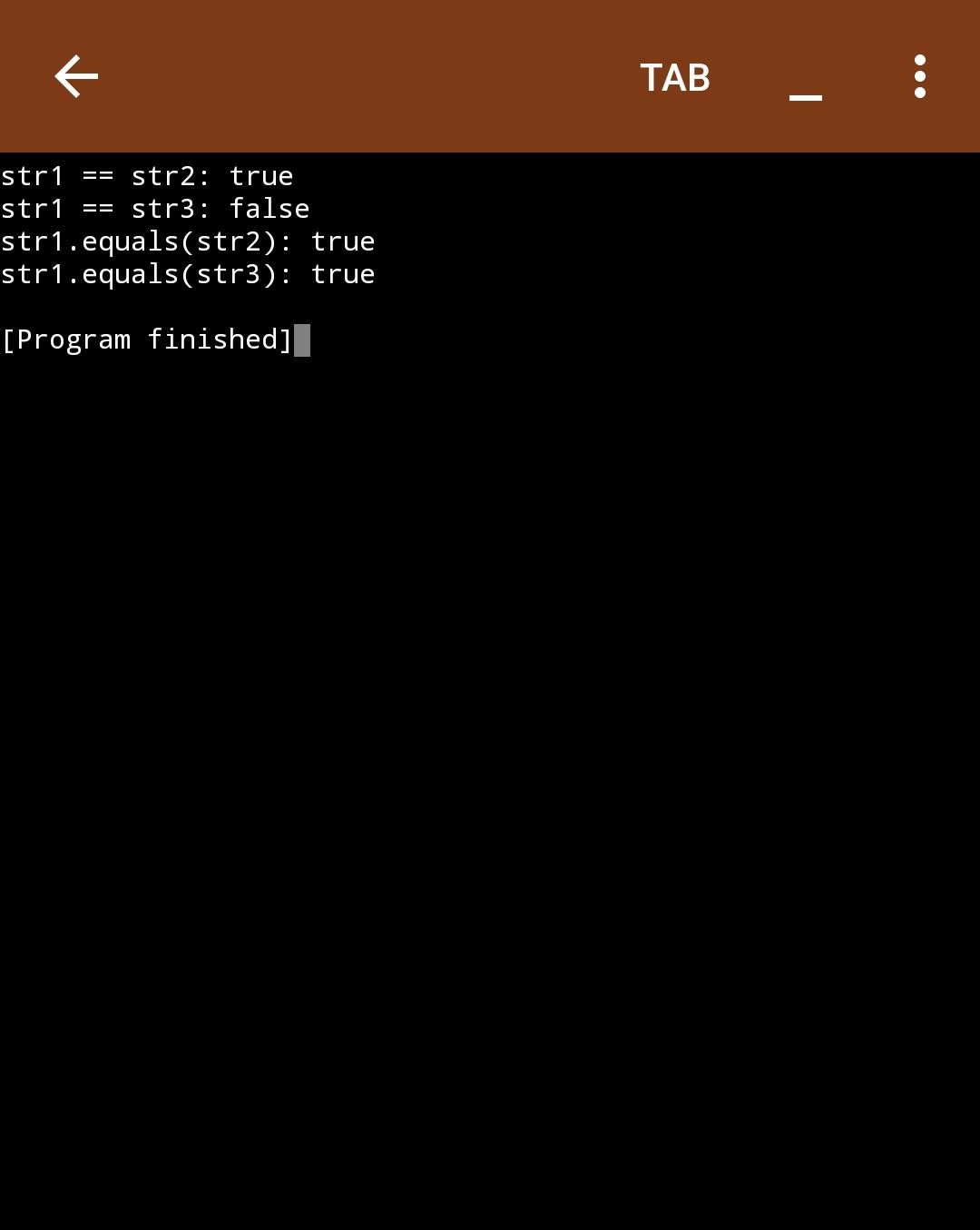
== checks if two things are the same object in memory (i.e., if they point to the same location in memory).

.equals() checks if two things have the same content (i.e., the actual values inside them).

Program:



Output:



Question 3:

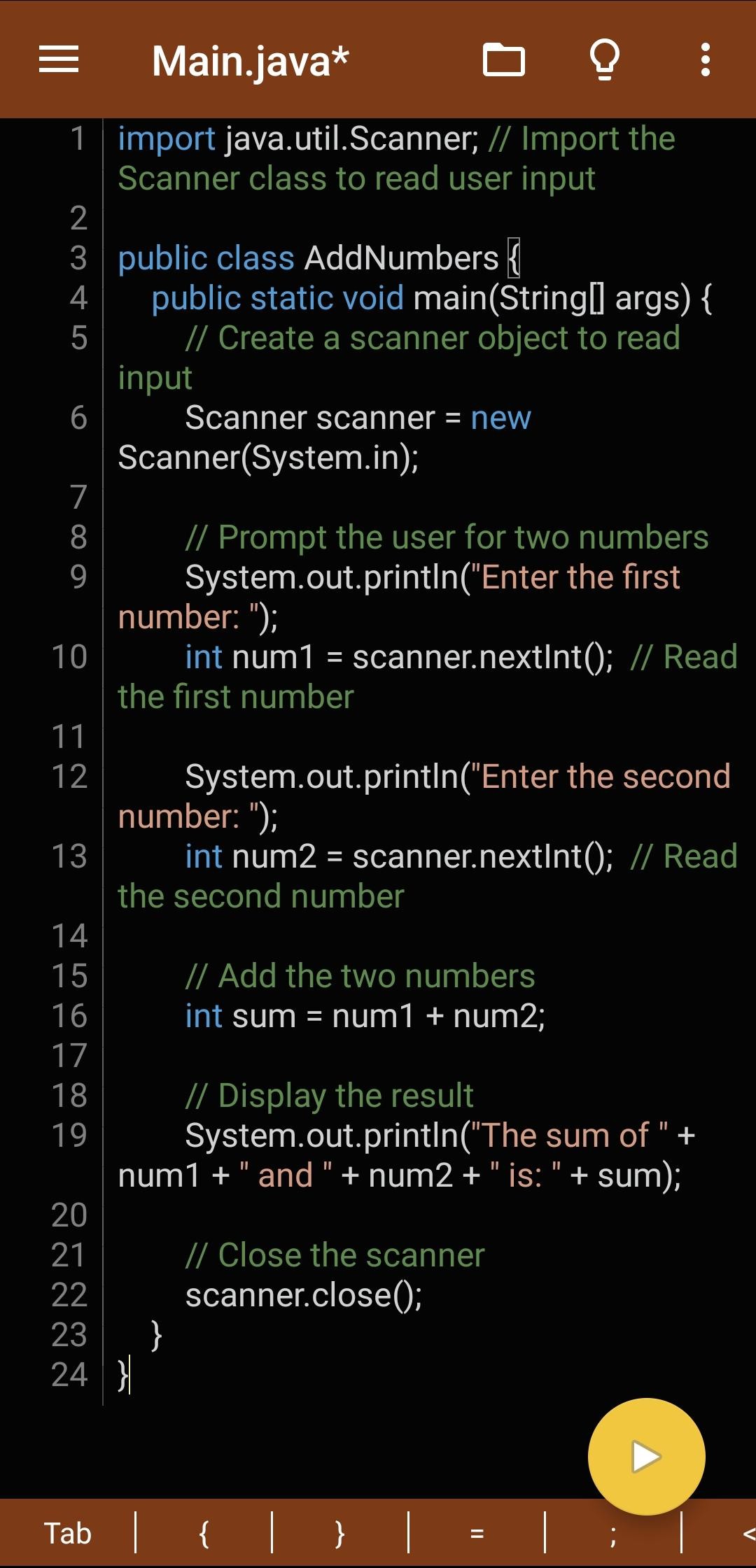
What is the main method?

In Java, the main method is the entry point for any Java application. When you run a Java program, the JVM (Java Virtual Machine) looks for the main method to start executing the code. It's where the program begins its execution.

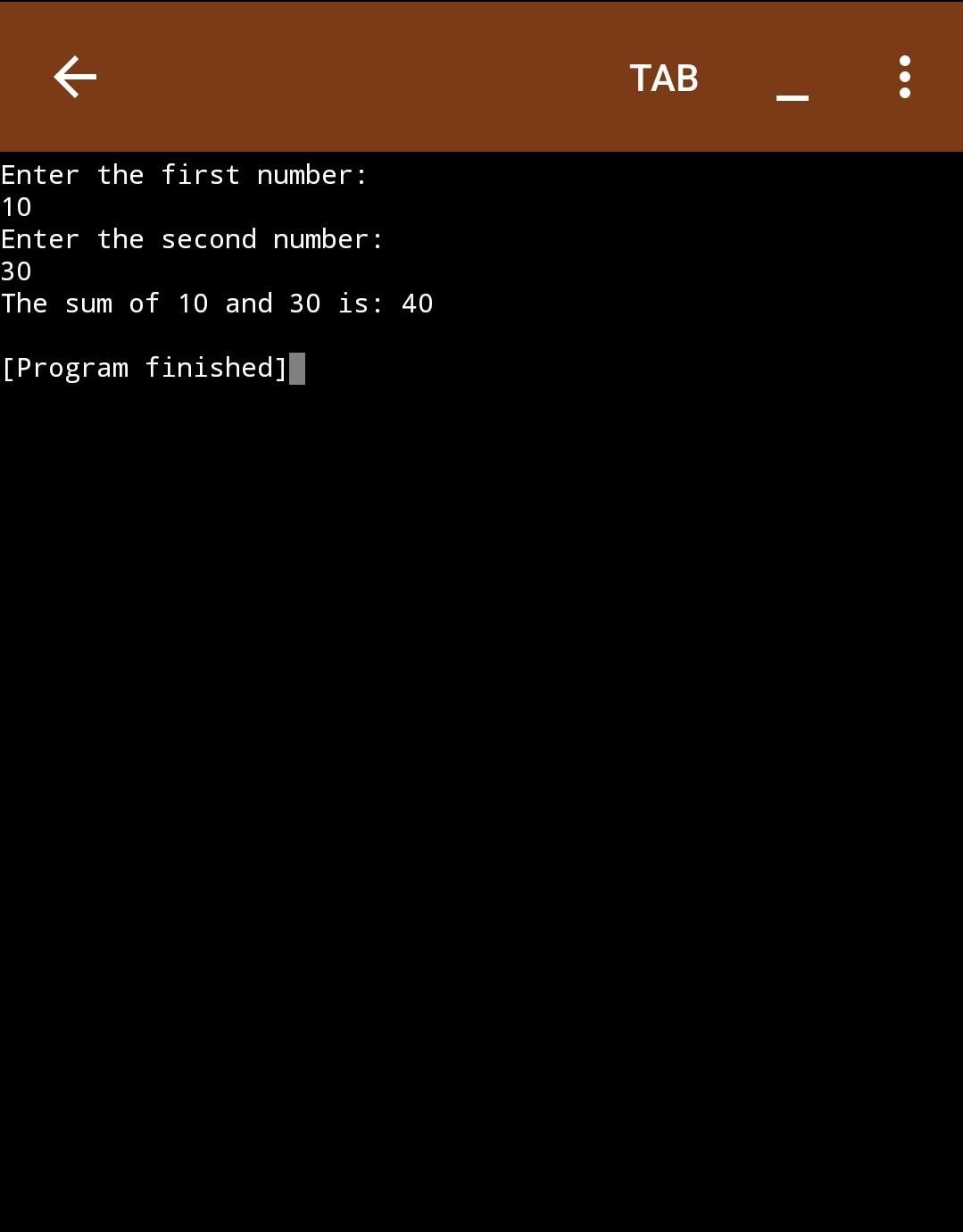
Question 4:

GUI that adds two numbers

Program:



Output:



Question 5:

Difference Between int, Integer, and String Explanation:

In Java, int, Integer, and String are all used to store data, but they are used in different ways. Here are their differences:

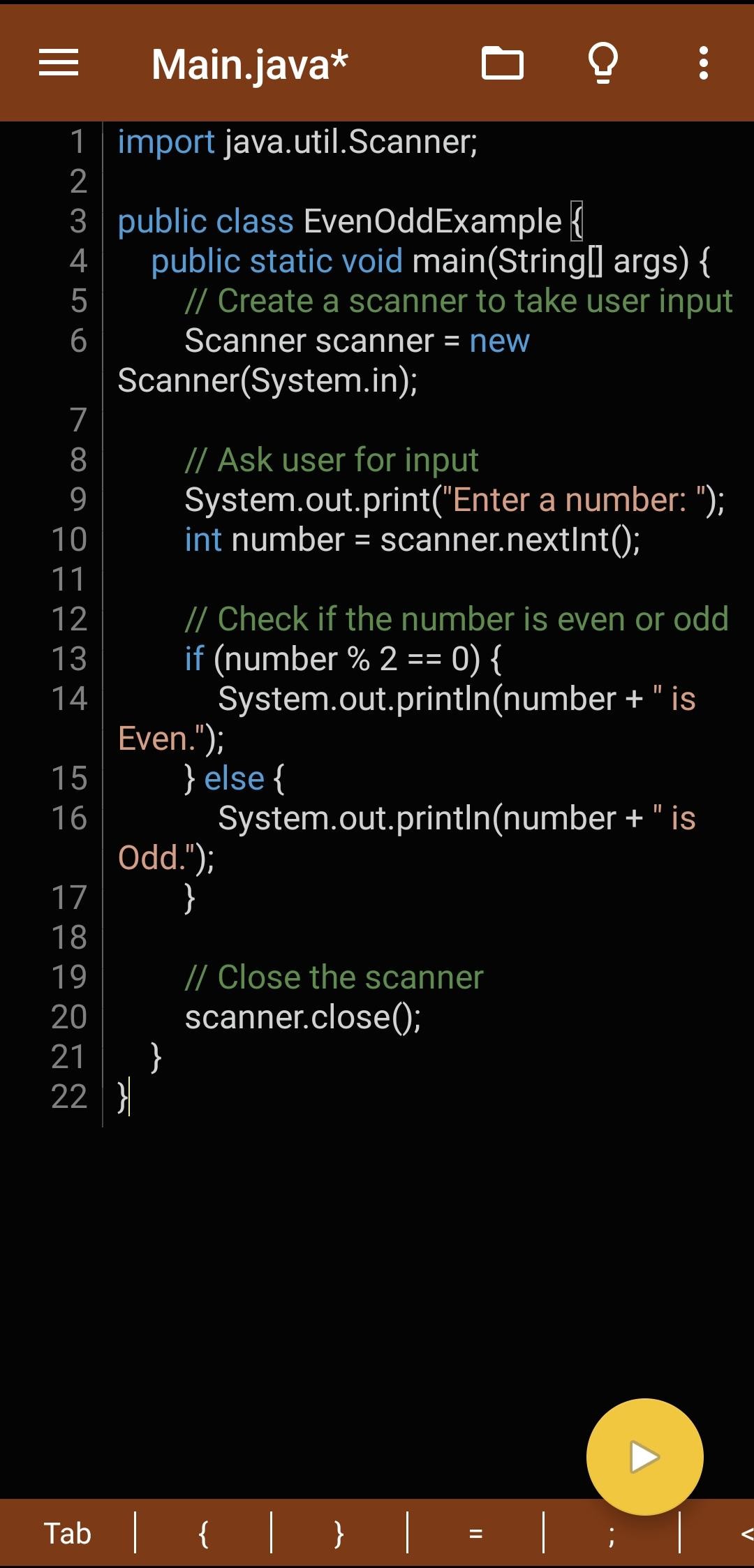
int: A primitive data type that stores whole numbers. It is not an object and is faster for simple arithmetic operations.

Integer: A wrapper class for the int primitive type. It allows you to store int values as objects. It is useful when you need to work with collections (like ArrayList) that require objects instead of primitives.

String: A class used to store a sequence of characters (text). It's not a primitive type but an object that stores a sequence of characters.

Control structures Question 6

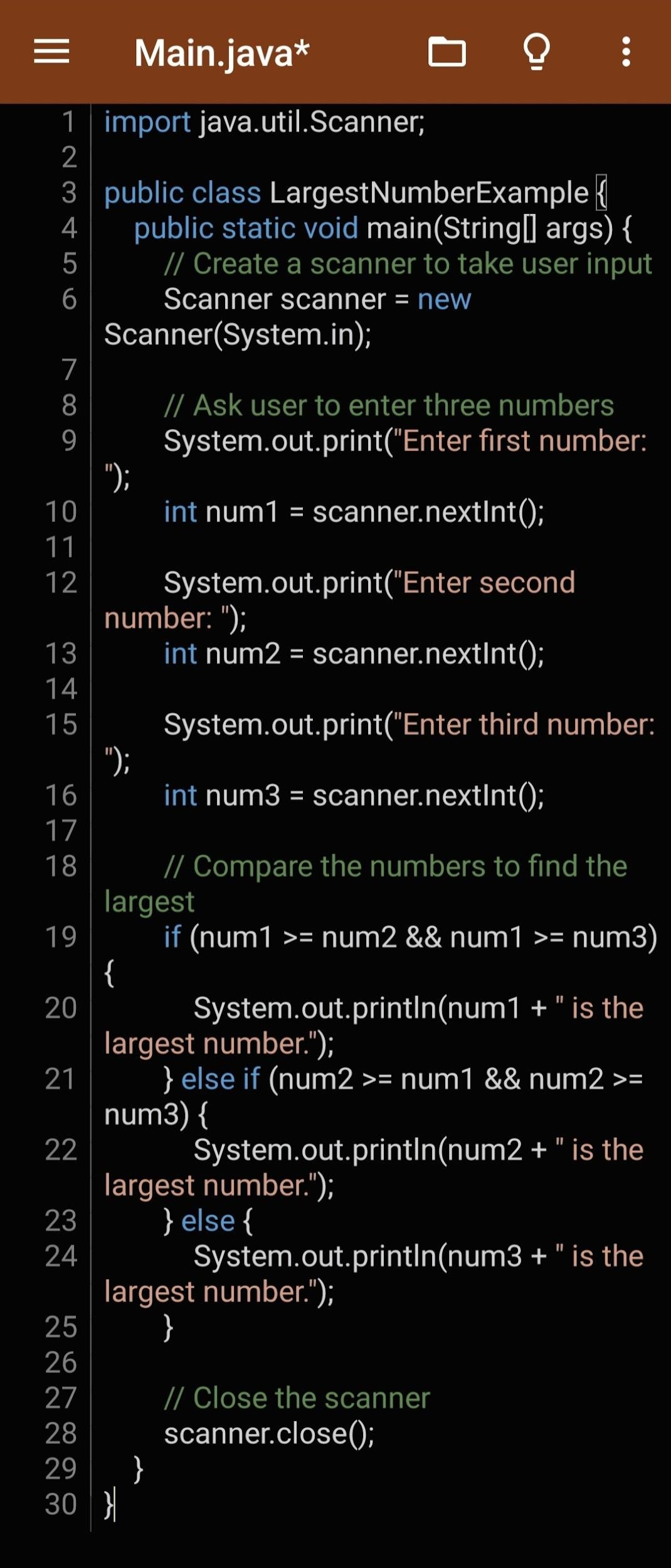
Code to check for even or odd number Program:



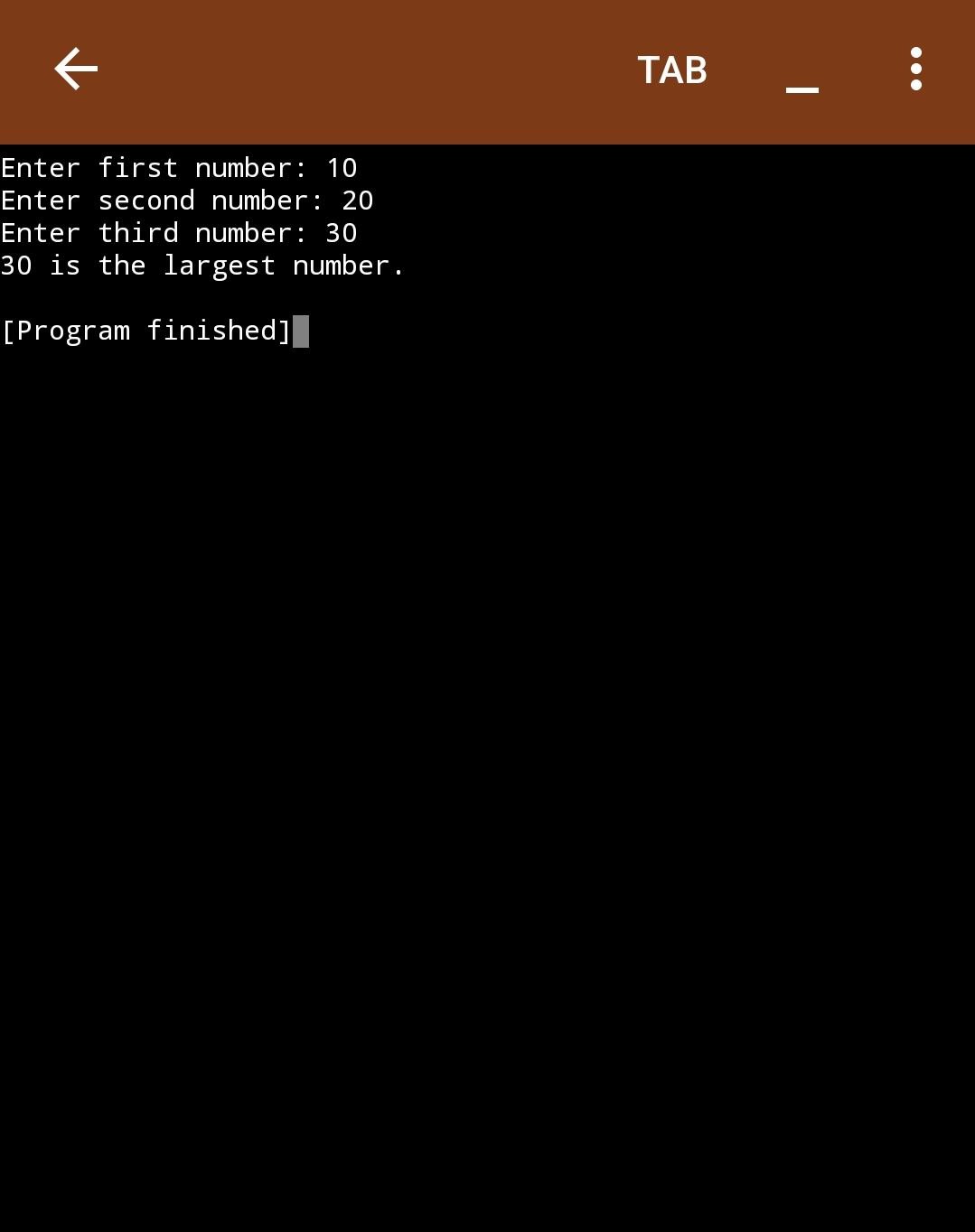
Output:



Question 7:

Code for largest number Program

Output:



Question 8:

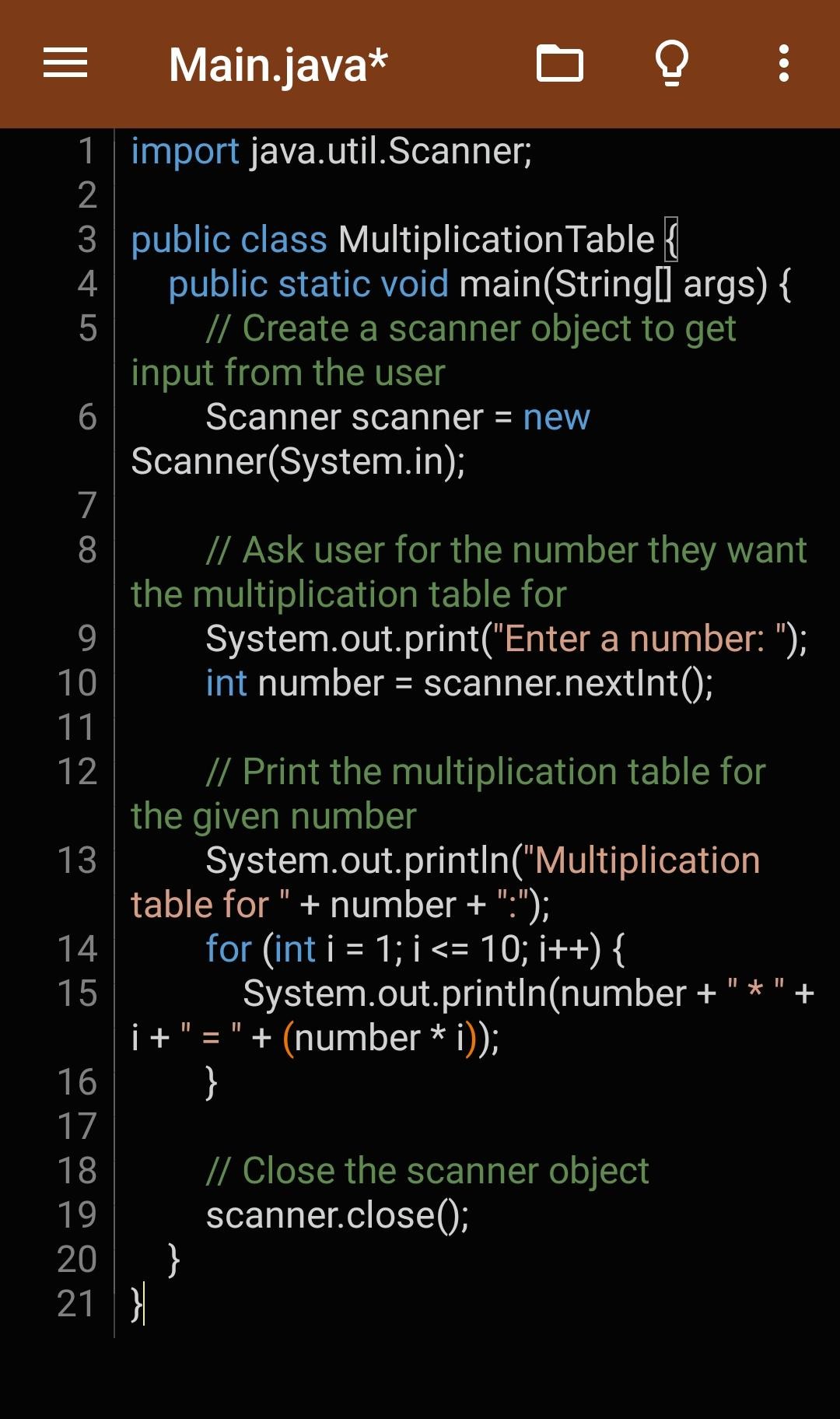
Difference between while, for, and do-while loops Explanation:

These are the three main types of loops in Java. Each has a slightly different behavior:

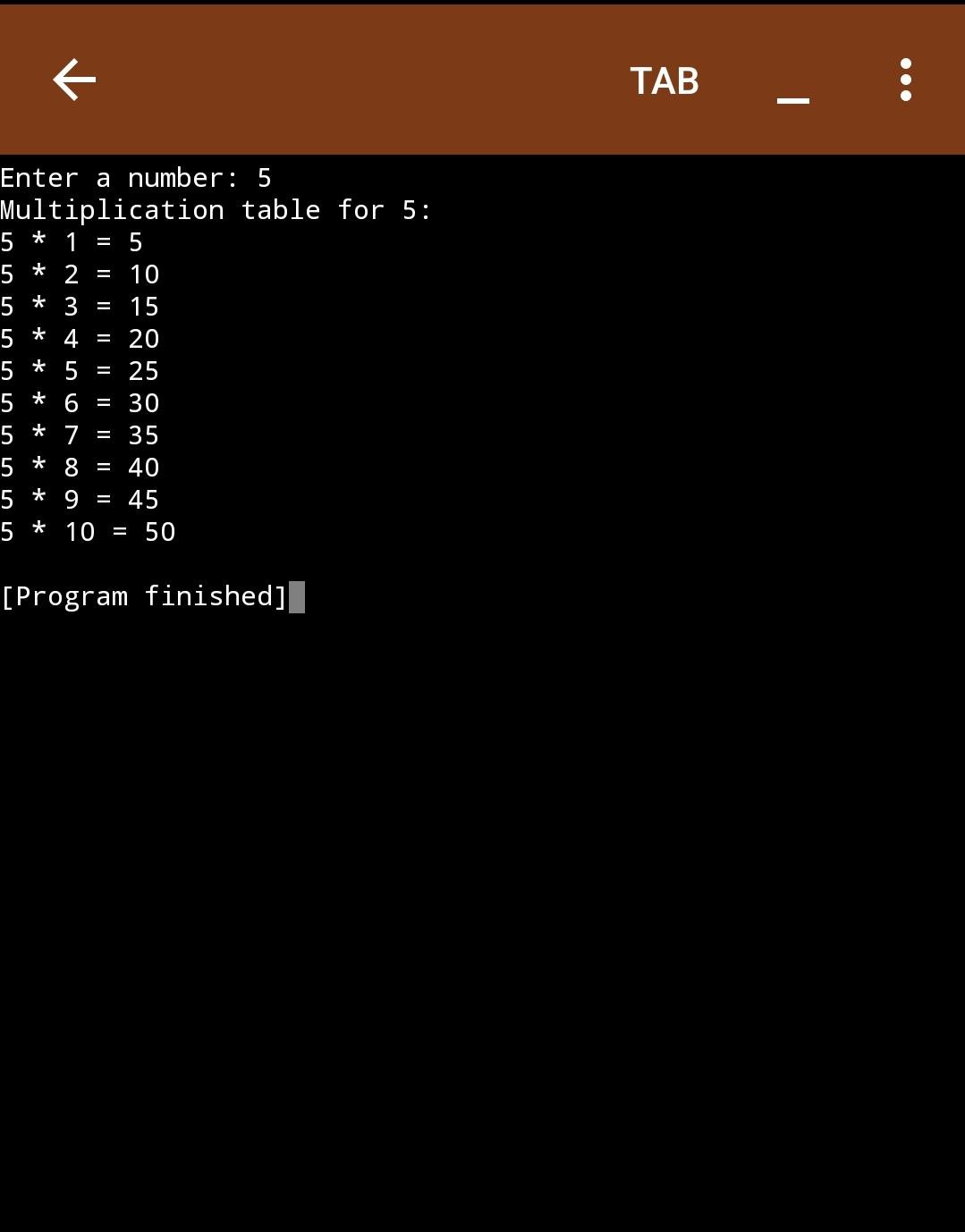
1. while loop: The condition is checked before executing the loop body. If the condition is false initially, the loop will not execute at all.
2. for loop: This is generally used when you know in advance how many times you need to iterate. The loop control variables (initialization, condition, and increment/decrement) are all written in one place.
3. do-while loop: The condition is checked after the loop body has been executed. This means the loop will always run at least once, even if the condition is false initially.

Question 9: Multiplication table code

Program



Output:



OOP Questions Question 10

4 pillars of OOP

The four pillars of Object-Oriented Programming (OOP) are core concepts that make programming in OOP languages more organized and manageable. They are:

1. Encapsulation
2. Abstraction
3. Inheritance
4. Polymorphism

Explanation…

1. Encapsulation:

Encapsulation is the concept of hiding the internal details of an object and only exposing a controlled interface. It protects the object’s data by restricting direct access to it and allows access only through getter and setter methods.

1. Abstraction:

Abstraction involves hiding the complex implementation details and showing only the essential features of an object. It allows a programmer to focus on high-level functionalities without needing to know the details of how they work.

1. Inheritance:

Inheritance allows a new class to inherit properties and methods from an existing class. It helps to create a hierarchical relationship between classes and promotes code reusability.

.

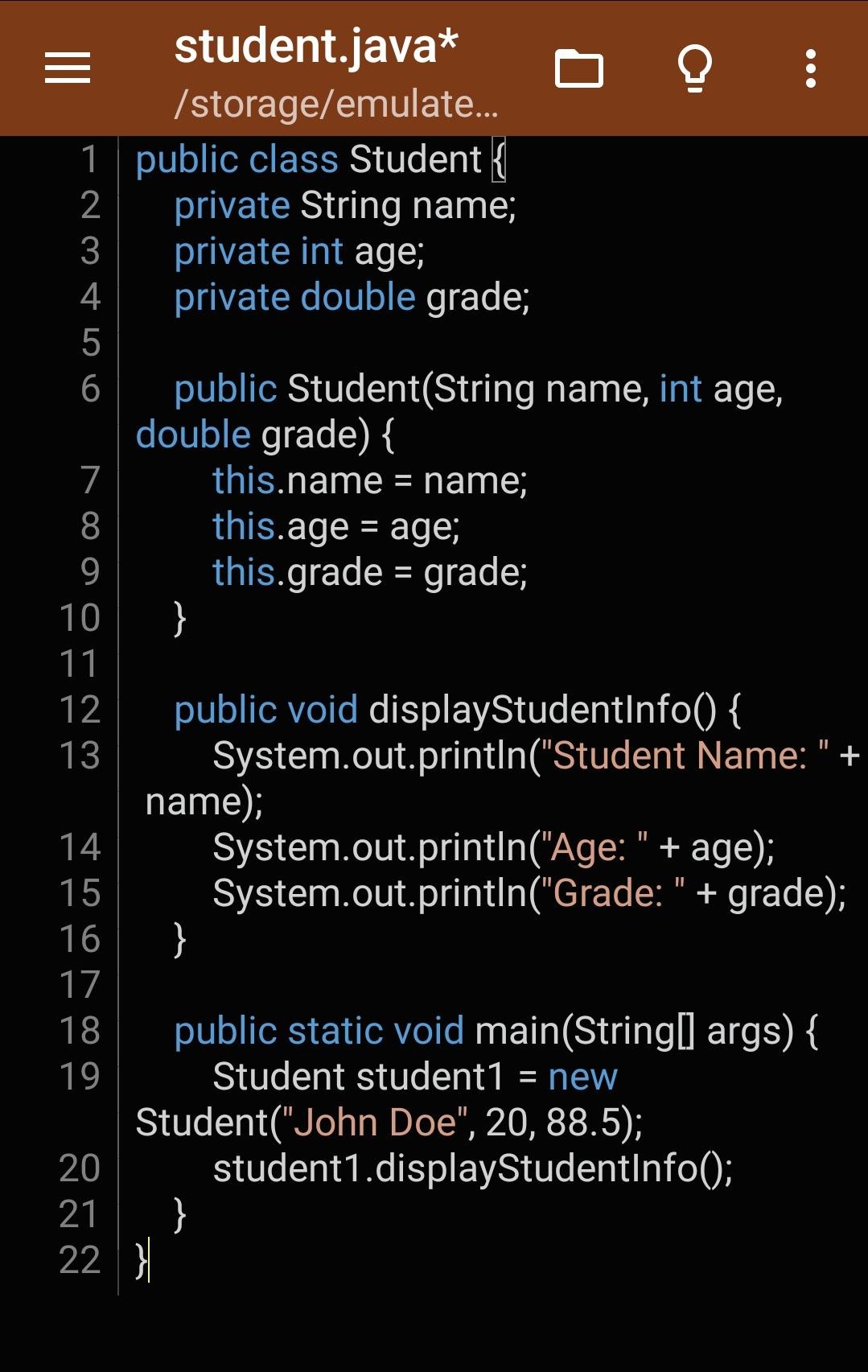
1. Polymorphism:

Polymorphism allows objects of different classes to be treated as objects of a common super class. It allows methods to have the same name but behave differently based on the object calling them.

Question 11

Class and Methods

Program:



Output:



Question 12:

What is Method Overloading (with code)?

Method overloading means creating multiple methods with the same name but different parameters (number or type). Java picks the right one based on the arguments you pass.

public class Calculator {

// First version: adds two integers public int add(int a, int b) {

return a + b;

}

// Second version: adds three integers public int add(int a, int b, int c) {

return a + b + c;

}

// Third version: adds two doubles public double add(double a, double b) {

return a + b;

}

public static void main(String[] args) { Calculator calc = new Calculator();

System.out.println("Add 2 and 3: " + calc.add(2, 3));

System.out.println("Add 1, 2, and 3: " + calc.add(1, 2, 3));

System.out.println("Add 2.5 and 3.5: " + calc.add(2.5, 3.5));

}

}

In this code:

add(int, int) add(int, int, int)

add(double, double)

All have same name but different parameters — that’s overloading.

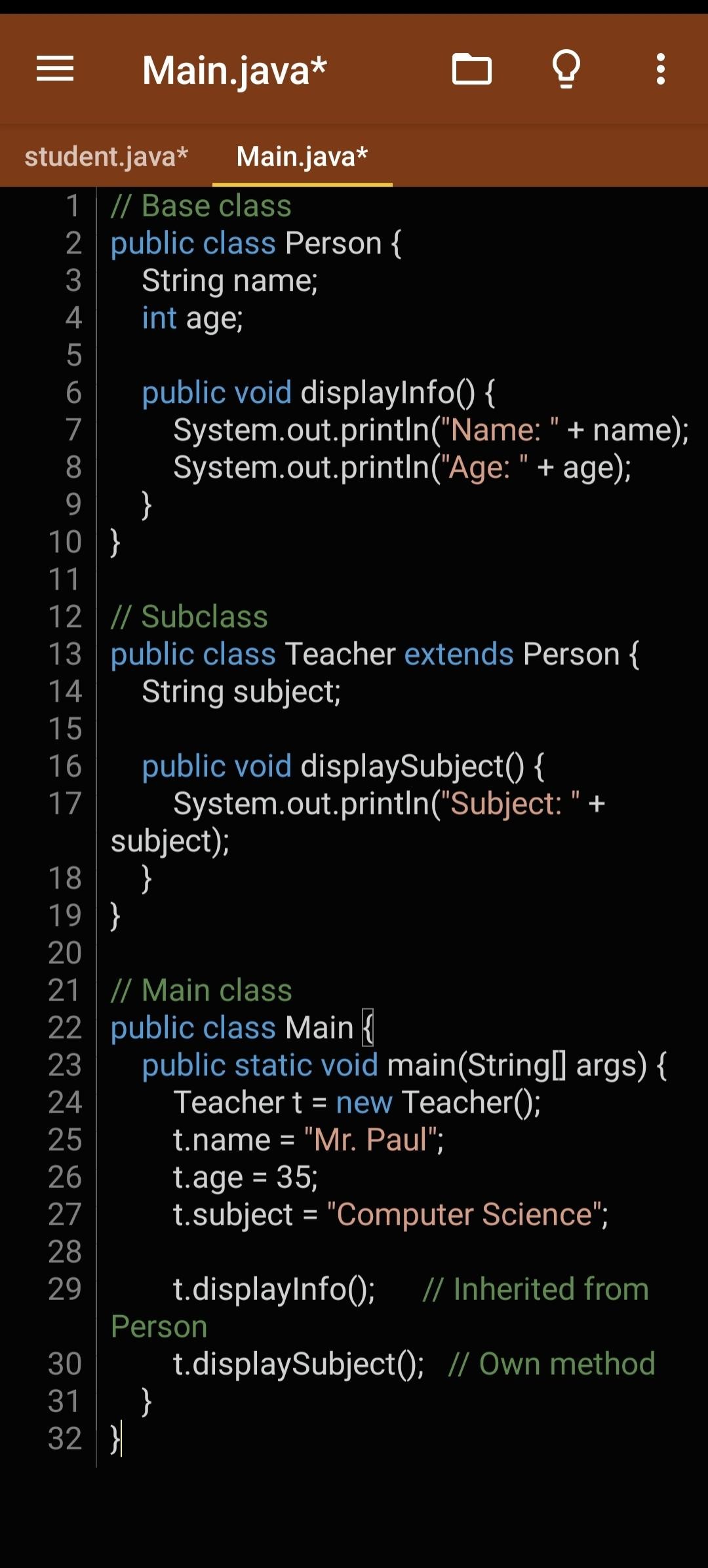
Question 13:

What is Inheritance? (with example using Person and Teacher)

Answer

Inheritance lets a class use the features (fields and methods) of another class. It helps reuse code and make programs more organized.

Example



General Practices Question 14:

What is clean code? (Give 3 good practices) Clean code means writing code that is:

* Easy to read
* Simple to understand
* Easy to maintain and debug

It helps other developers (and future you) understand your work without stress.

Three Good Practices of Writing Clean Code:

1. Use meaningful names:

Use names that clearly describe what a variable or function does. E.g., calculateAverage() is better than calcAvg() or cA().

1. Keep functions short and focused:

Each function should do one thing only. This makes code neat and easy to debug.

1. Use comments when necessary:

Explain why something is done, not what it’s doing — especially if it’s tricky.

Question 15:

Why long methods should be avoided

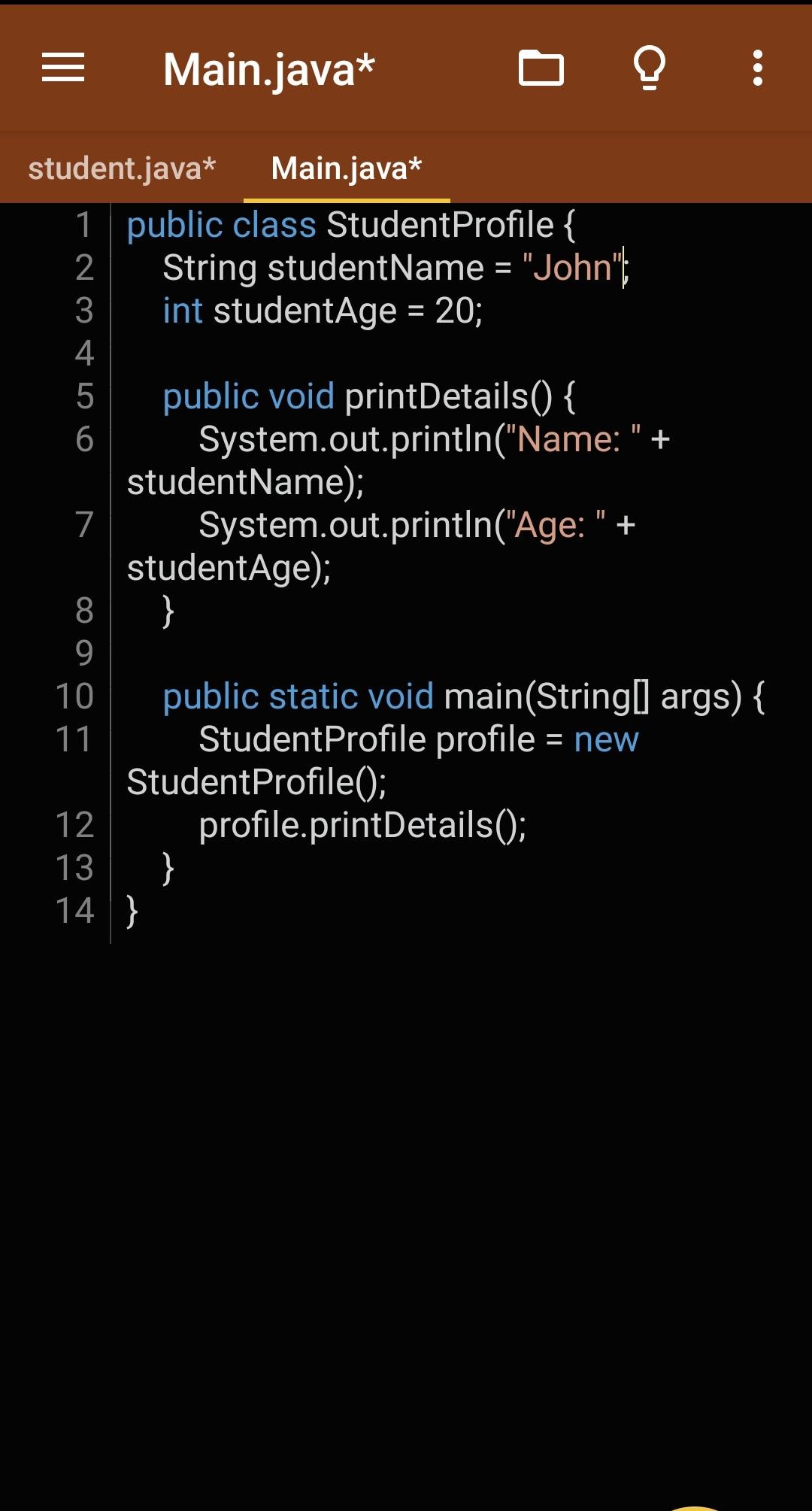
Answer

Long methods should be avoided in Java because they make the code hard to read, debug, and maintain. When a method is too long, it often tries to do too many things at once, which can confuse anyone reading the code and make testing more difficult. It's better to break your logic into smaller methods that each do one clear task.

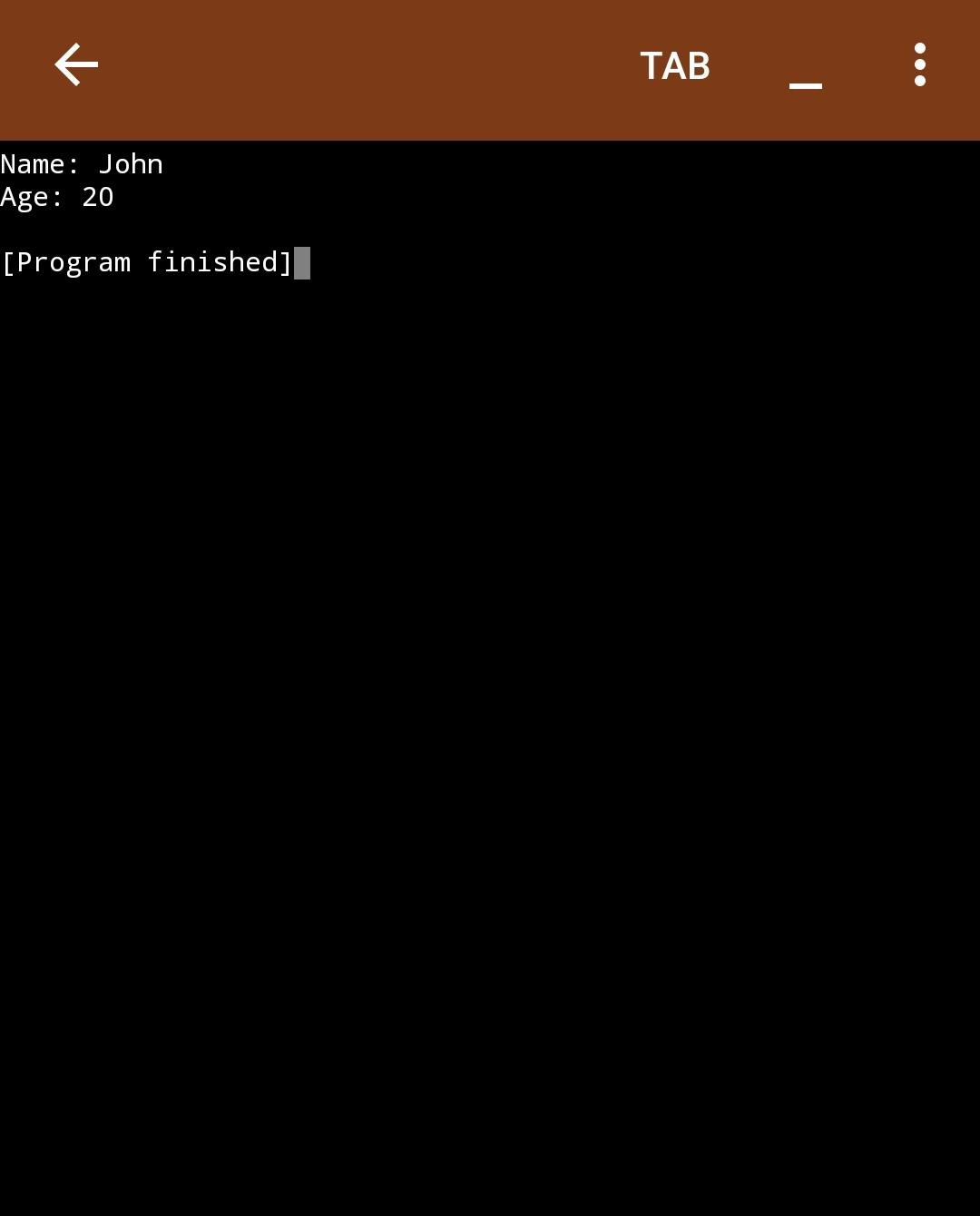
Question 16:

Naming conventions

Java follows standard naming conventions to make code easy to read and understand: Class names: Start with a capital letter (PascalCase). Example: StudentProfile Variable names: Start with a small letter (camelCase). Example: studentName

Method names: Also start with a small letter (camelCase). Example: printDetails() Program

Output



Question 17:

Why should we break code into methods?

Answer

Breaking code into methods helps keep your program organized. Each method does one job, so:

It’s easier to read and understand.

You can reuse the method without rewriting code.

It makes debugging easier—if something goes wrong, you know where to check

Question 18:

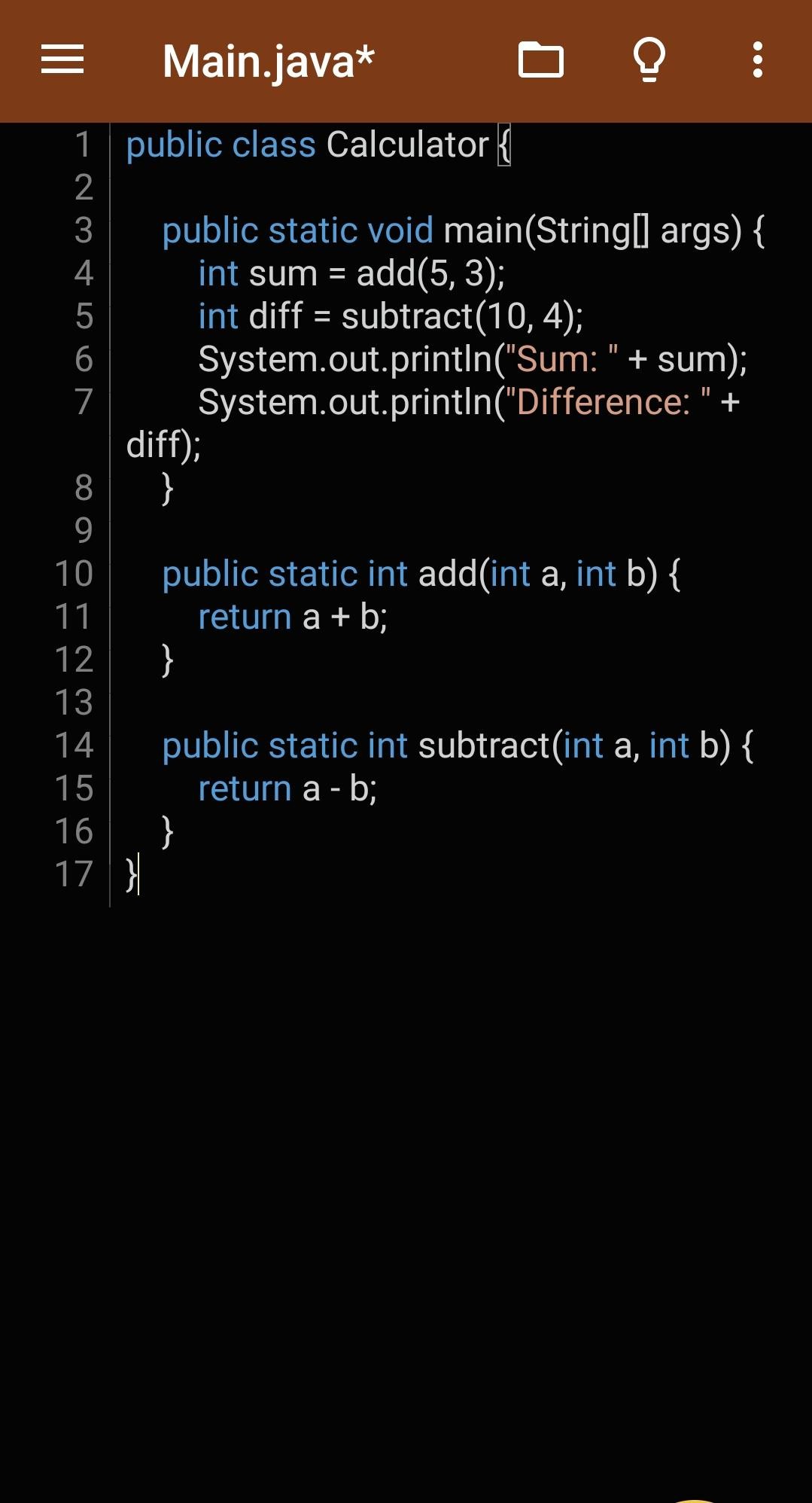
DRY with example

Answer

The DRY principle stands for Don't Repeat Yourself. It means you shouldn’t write the same code more than once. Instead, put it in a method and call it whenever needed.

By following DRY, you make your code cleaner, easier to maintain, and less error-prone. Example

Program



Output



Question 19:

Why use classes/objects over just the main method?

Answer

Using classes and objects makes your code more organized, modular, and easier to manage.

With the main method, all your logic is placed in one big block, which can be hard to manage and extend. But with classes and objects, you can divide the code into smaller chunks (classes), each responsible for a specific task.

For example:

Classes represent entities or concepts (like a Car, Student, etc.).

Objects are instances of those classes, where each object holds specific data (like the color of a car or the name of a student).

Using classes helps in reusability and makes your code easier to maintain and extend.

Testing and Debugging Question 20:

Why is testing important?

Answer

Testing is crucial because it ensures that your code works correctly and does what it's supposed to. Without testing, you may have bugs or errors that you don’t notice, which can lead to your program crashing or producing wrong results.

Testing helps you:

1. Catch errors early, preventing issues later.
2. Ensure code quality by verifying that all parts work as expected.
3. Save time in the long run, as fixing issues early is easier than later.

You test to make sure everything is functioning as it should and to avoid future problems in production.

Question 21:

Differences between Syntax, Runtime and Logic errors

Answer

Syntax Errors: These occur when you make a mistake in the code's structure, like missing semicolons, parentheses, or incorrect keywords. The compiler will catch these before running the program.

Runtime Errors: These happen when the program runs, and it encounters an unexpected problem, like dividing by zero or accessing an invalid array index.

Logic Errors: These occur when your code runs without errors but doesn't behave as expected. It’s not a mistake in syntax or runtime, but a flaw in the logic of the code.

Question 22:

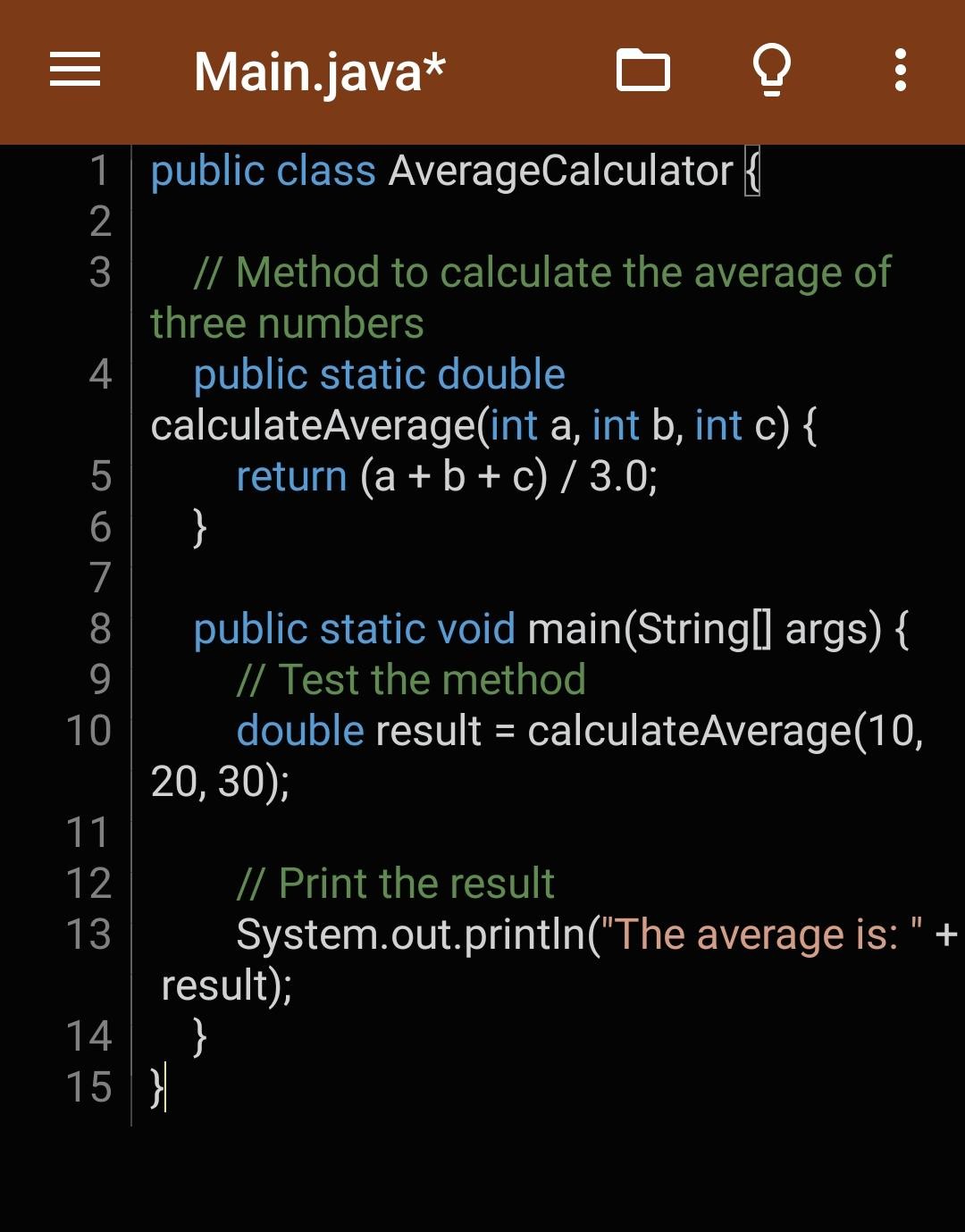
How to test average calculator method (with code)

Answer

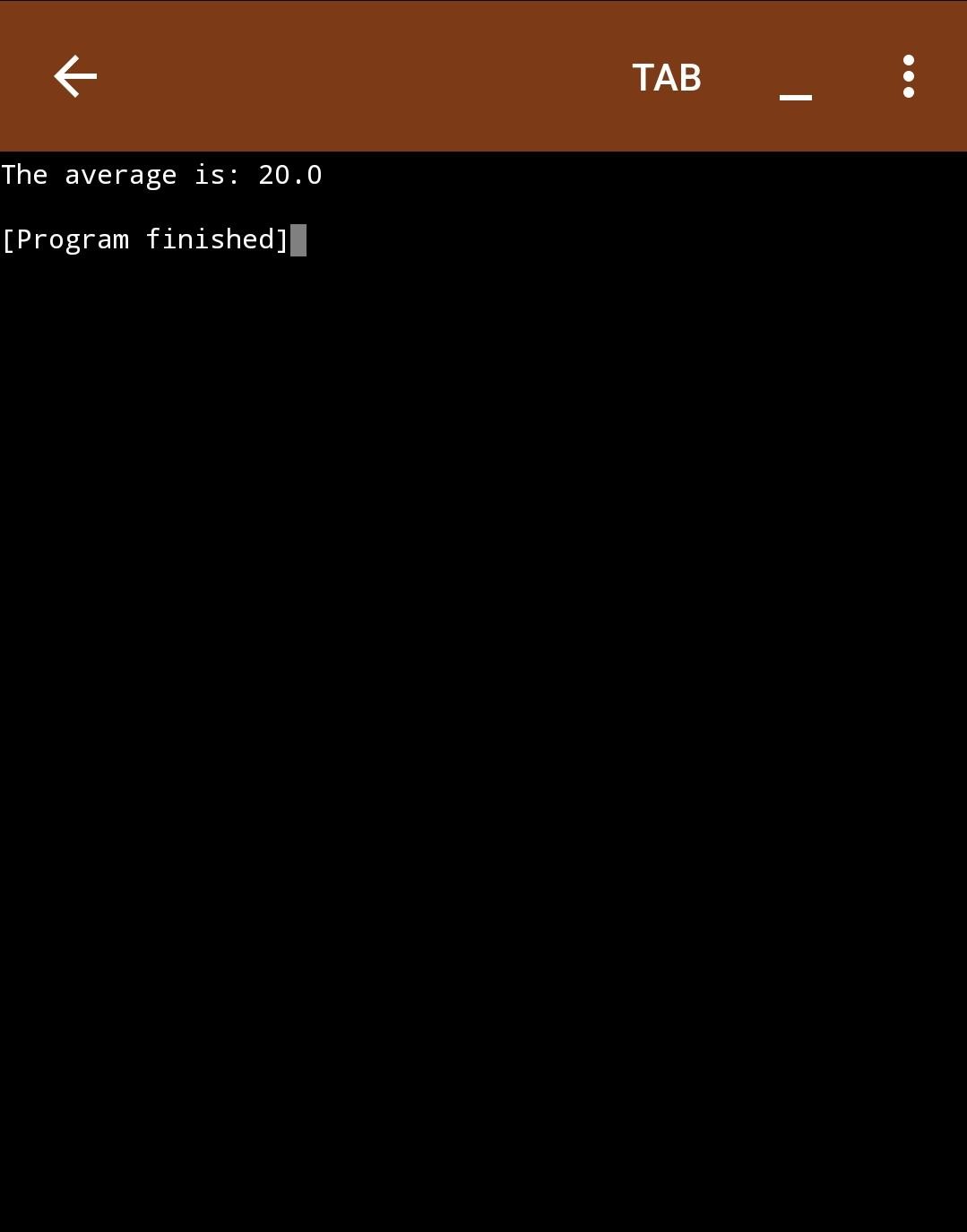
To test the average calculator method, you can:

1. Write the method that calculates the average.
2. Call it with some test values.
3. Print the result and verify if it matches the expected average.

Program:



Output:



Comments and Documentation Question 23:

Why write comments in your code?

Answer

We write comments in code to explain what the code does so that:

1. Others (or even you later) can understand the purpose of the code easily.
2. It makes the code more readable and maintainable.
3. Helps in debugging or improving the code later.

Question 24:

What is the difference between JavaDoc comments and regular comments?

Answer:

Java has three types of comments:

1. Single-line comments (//) – used for short notes.
2. Multi-line comments (/\* ... \*/) – used for longer explanations.
3. JavaDoc comments (/\*\* ... \*/) – used specifically to generate documentation for classes, methods, etc.

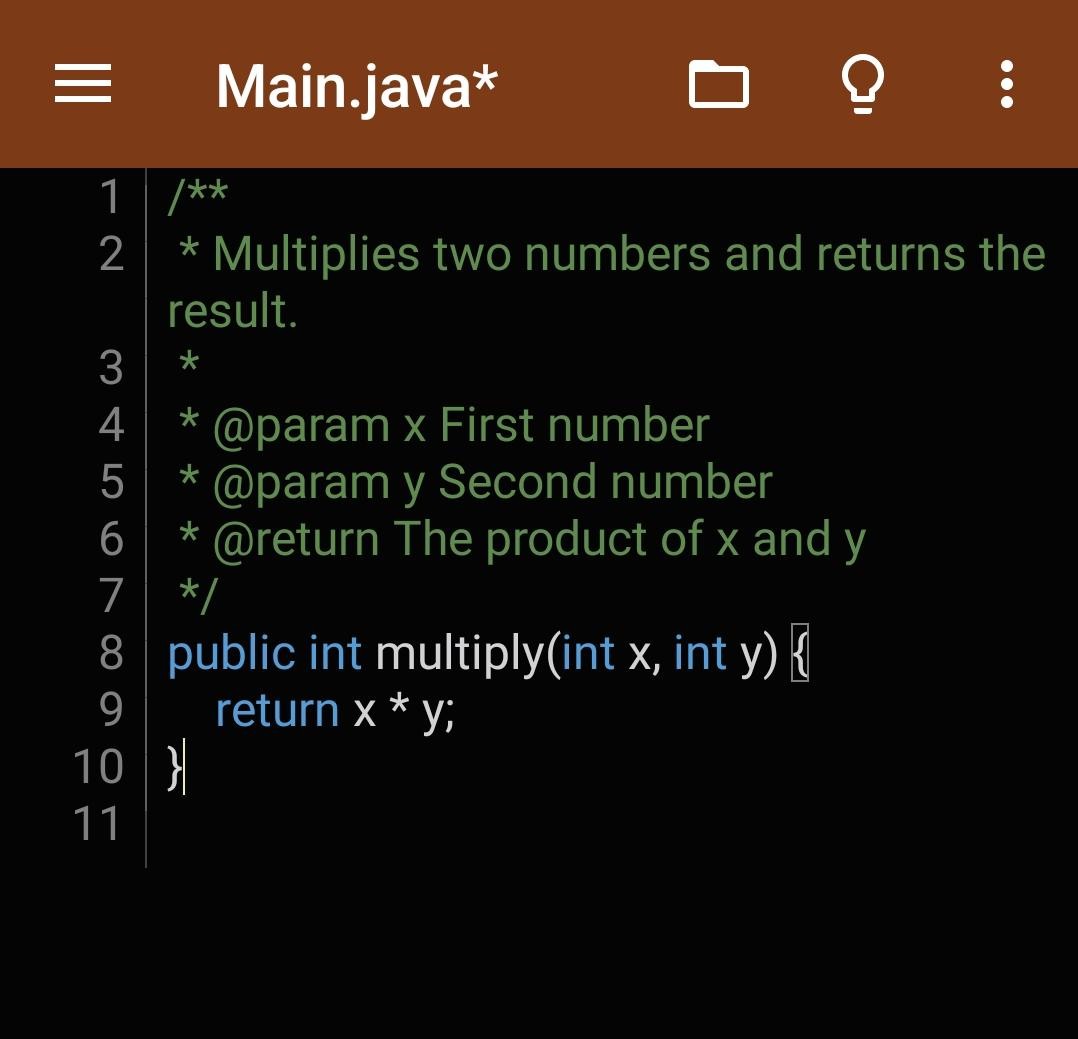
Key Difference:

Regular comments (//, /\* \*/) are just for the programmer to read.

JavaDoc comments (/\*\* \*/) are special because tools like javadoc can turn them into HTML documentation that explains how the code works.

Question 25:

Write a sample method and include JavaDoc comments to describe what it does. Example



Versioning and Collaboration Question 26:

What is version control & why is it important?

Version control is a system that helps developers track and manage changes to code over time. It lets you go back to previous versions, collaborate with others without overwriting each other’s work, and keep your work organized.

Importance:

Helps you undo mistakes by going back to earlier versions.

Makes teamwork easy — everyone works on their part without conflict. Keeps a history of what changed, when, and by whom.

Popular version control system: Git. Popular hosting platform: GitHub.

Question 27:

Explain “code refactoring”

Code refactoring means improving the internal structure of your code without changing what it does. It makes the code cleaner, easier to read, and easier to maintain.

Example:

Changing messy or repeated code into neat, reusable methods — but the app still works the same.

Why refactor?

Makes code easier to understand Reduces bugs

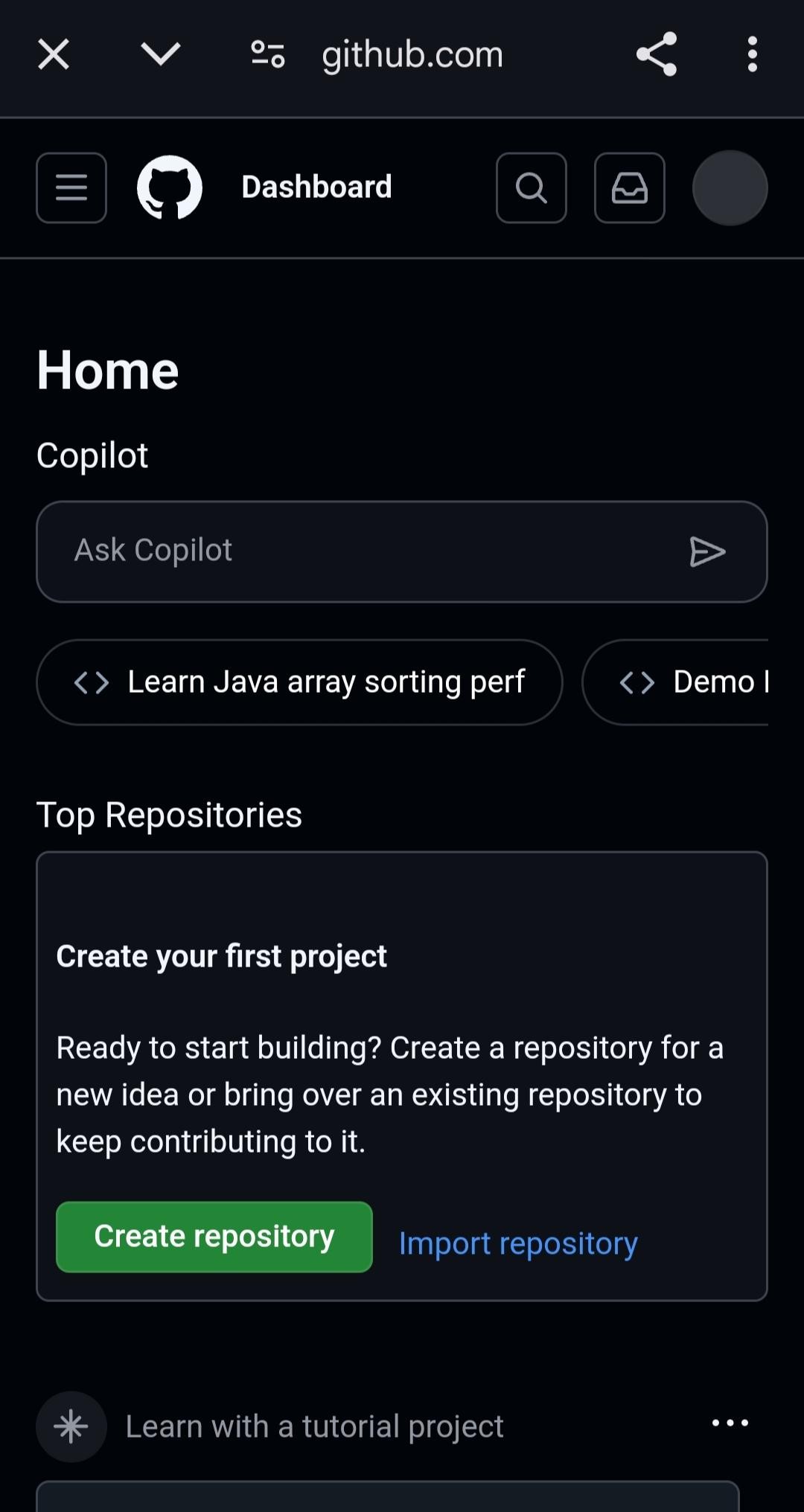
Prepares your code for future updates

Question 28:

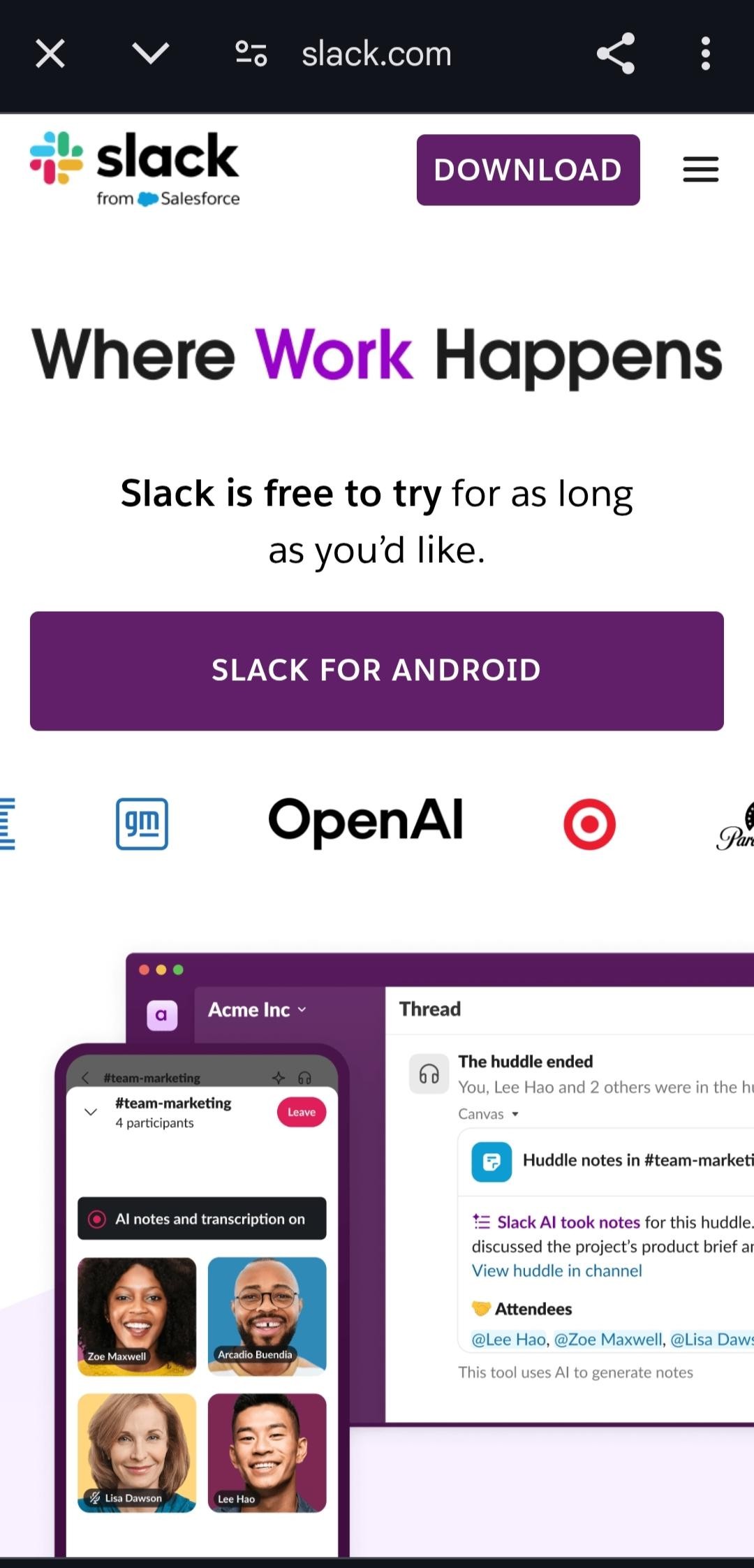
List 3 collaboration tools Java developers use (include screenshots) Here are three tools commonly used by Java developers to collaborate:

1. GitHub – For sharing and managing code in teams
2. Slack – For team communication and updates
3. Trello – For organizing tasks and workflow

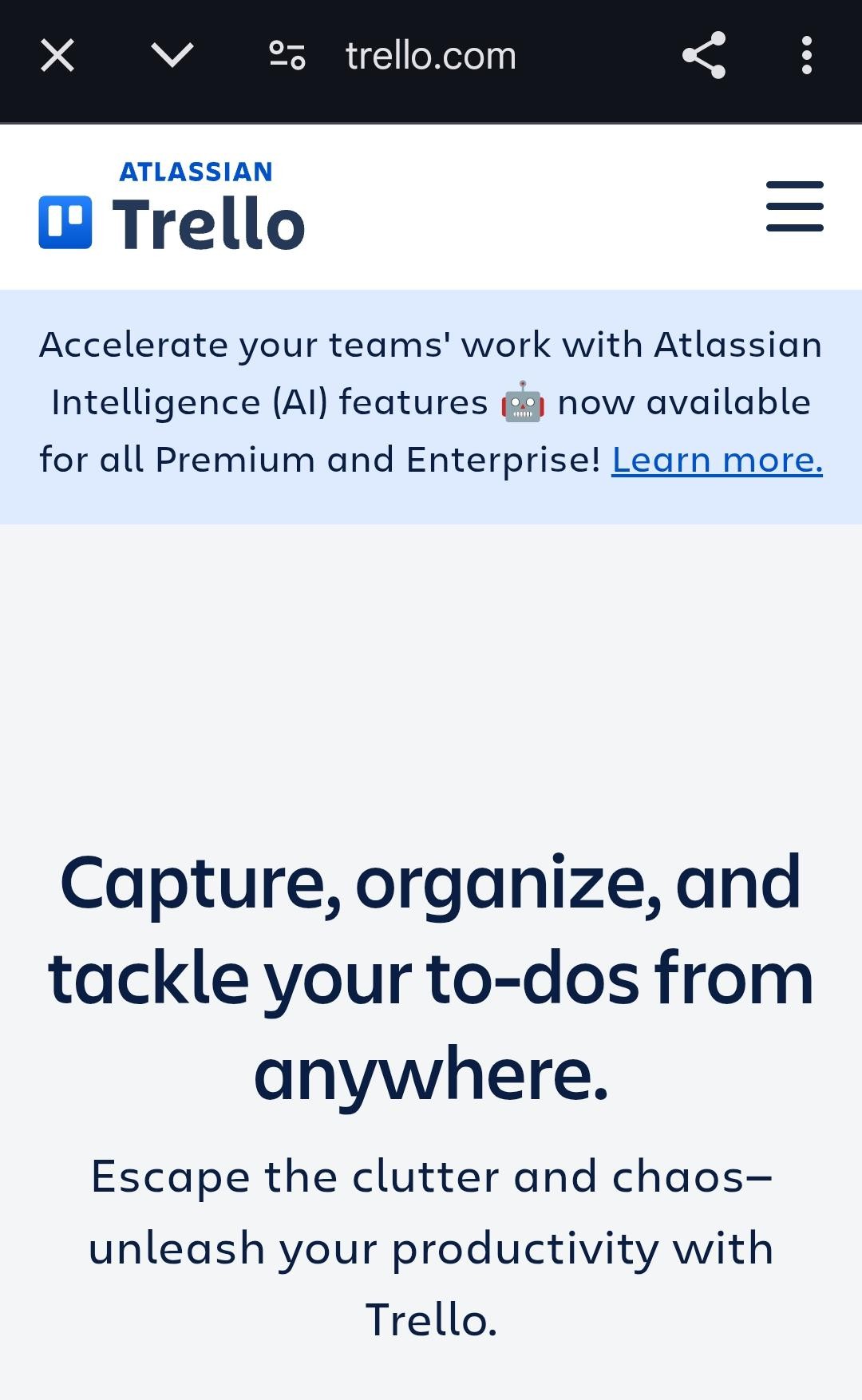
GitHub



Slack



Trello



Good Practices Summary Question 29:

List 5 best practices for Java development

Here are five best practices every Java developer should follow:

1. Use meaningful names

Give your variables, methods, and classes names that clearly describe their purpose.

1. Keep methods short and focused

Each method should do only one thing. This makes your code easier to understand and maintain.

1. Write comments only when necessary

Code should be self-explanatory. Use comments to explain why, not what.

1. Follow Java naming conventions For example:

Class names: StudentManager

Variable names: totalMarks Method names: calculateGrade()

1. Use proper exception handling

Always catch specific exceptions, not just general ones, and avoid empty catch blocks.

Question 30:

What is code readability and why does it matter?

Code readability means how easy it is for someone else (or even your future self) to read, understand, and work with your code.

Why it is important:

Makes debugging and maintenance easier.

Helps team members understand your code quickly. Reduces mistakes and confusion.

Encourages writing clean, organized, and consistent code.

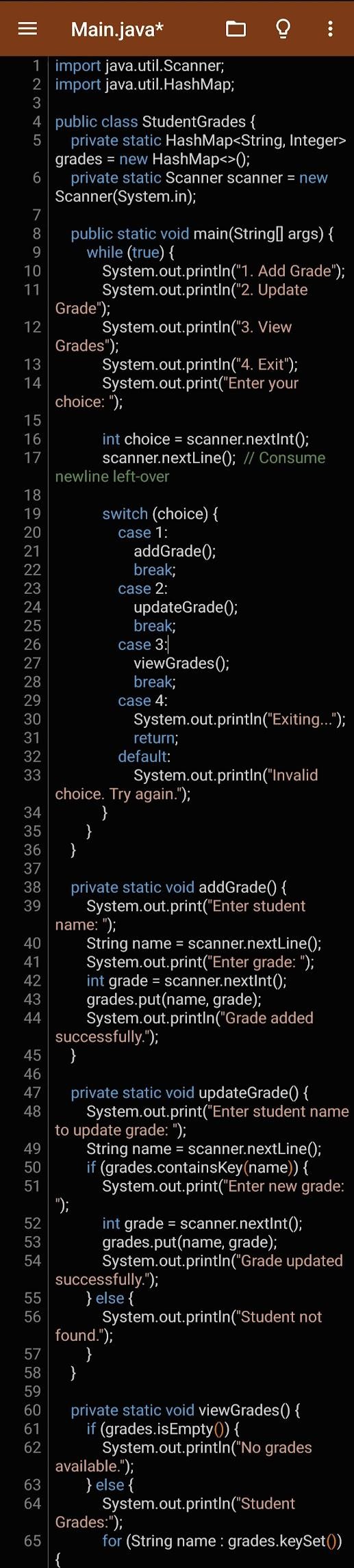
Readable code saves time and helps everyone write better software

Question 31:

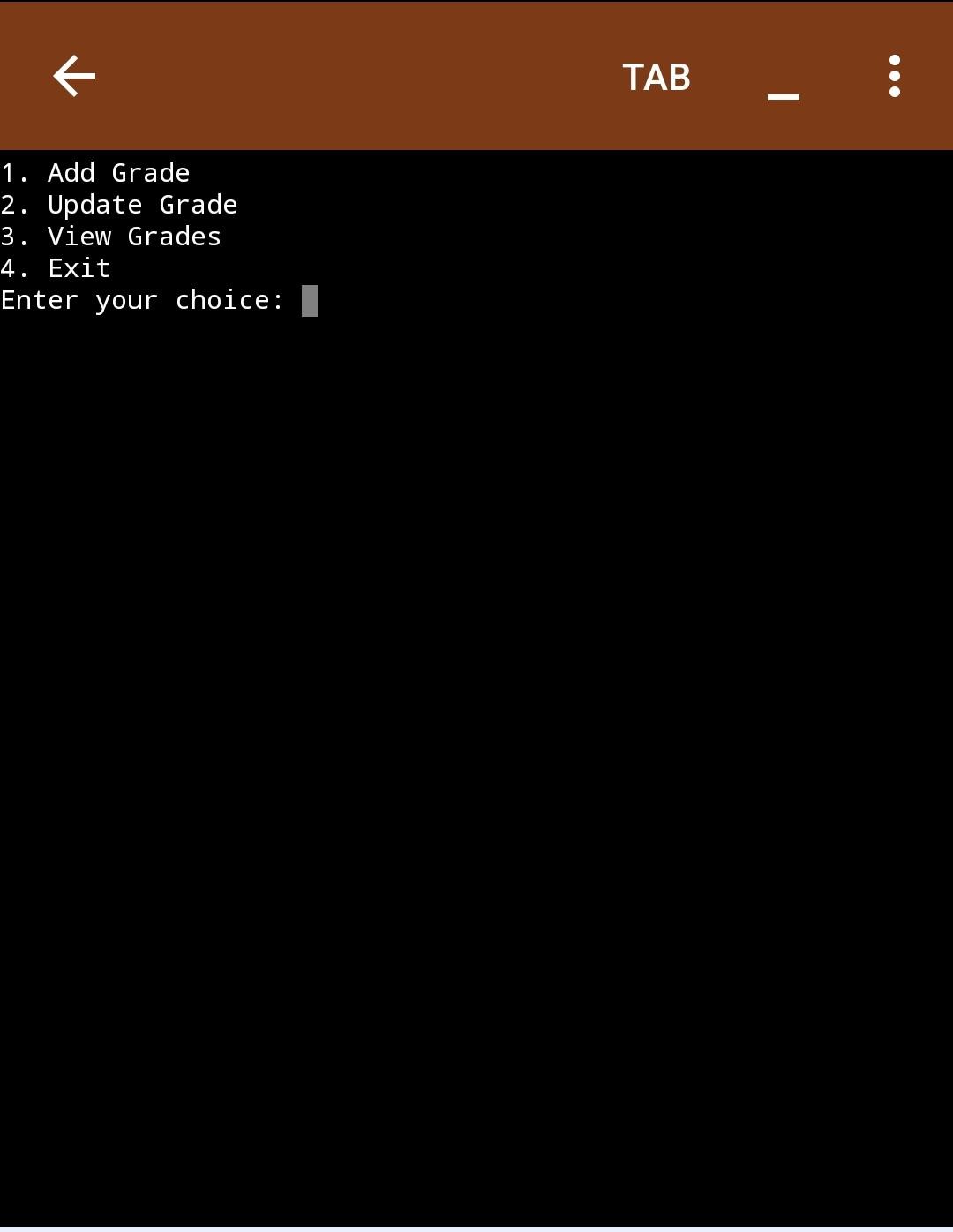
Mini Projects/Logic Building

CLI App for Student Grades (Add, Update, View)

Program



Output:

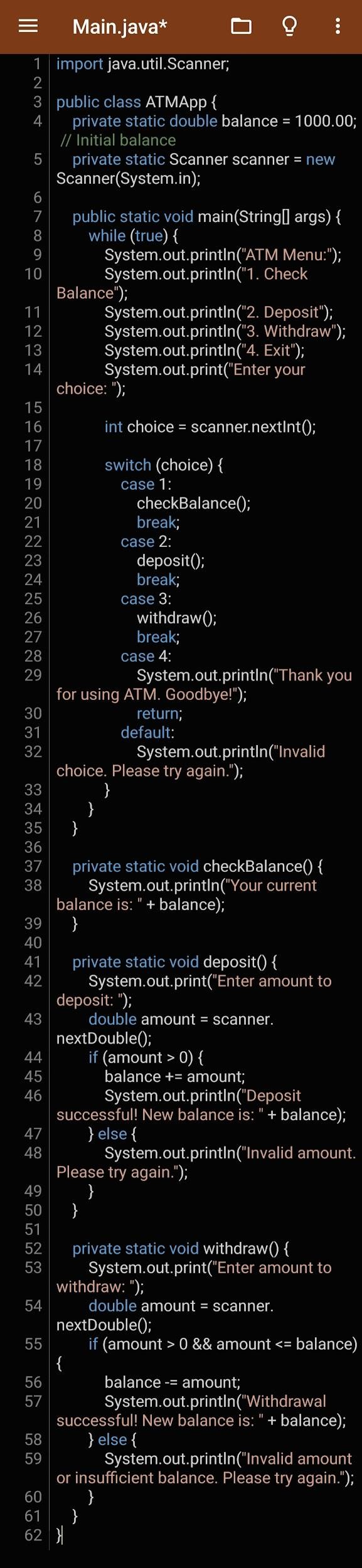


Question 32

Basic ATM App (Check balance, Deposit, Withdraw)

Here’s a simple example of how to implement a basic ATM app. It allows a user to check their balance, deposit money, and withdraw money.

Program



Output

