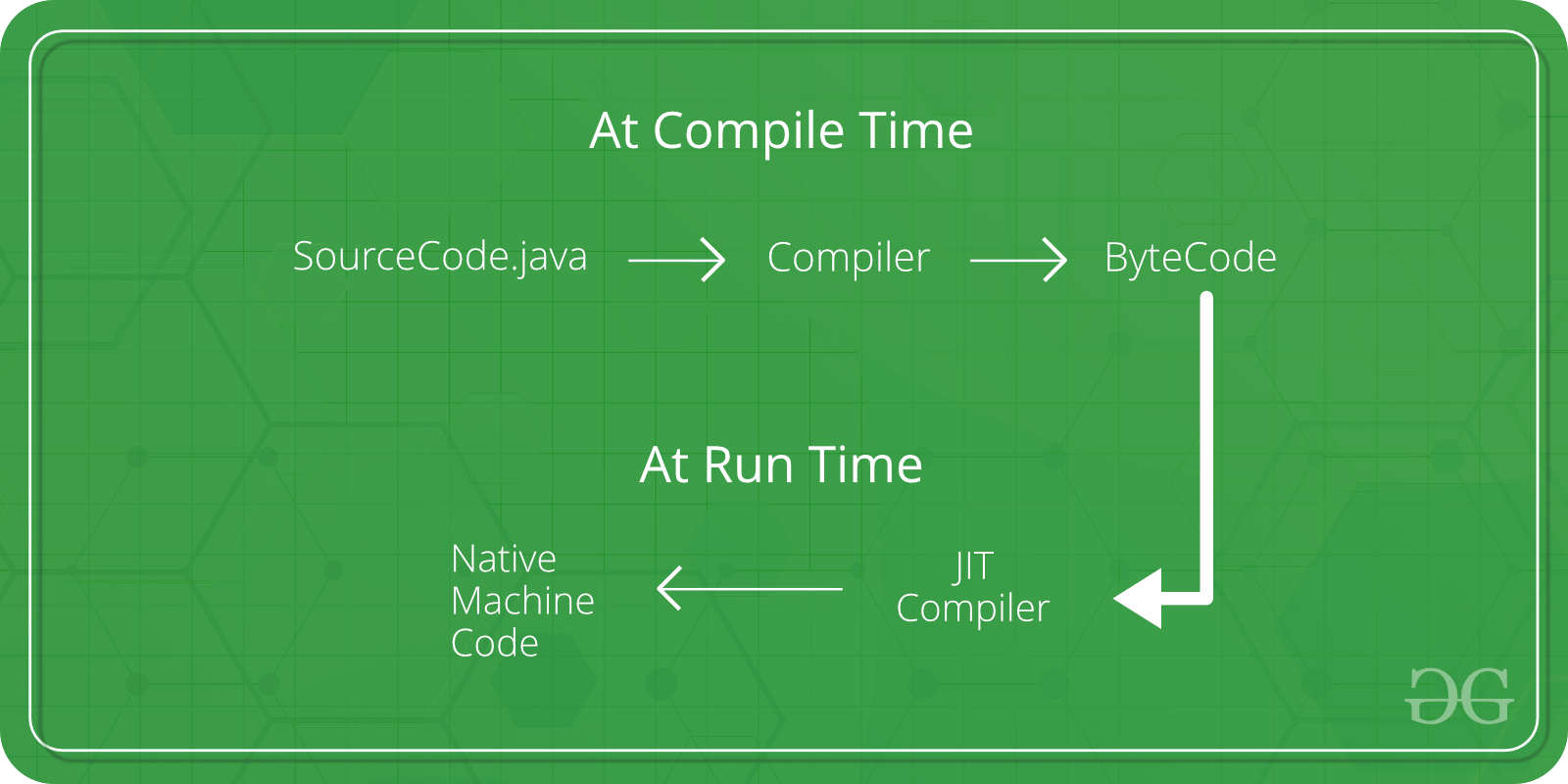
**JAVA Assignments(Day - 2)**

1. **Explore JIT**

****

The way of converting bytecode to native machine language for execution has a huge impact on its speed of it. These Bytecode have to be interpreted or compiled to proper machine instructions depending on the instruction set architecture. In order to improve performance, JIT compilers interact with the Java Virtual Machine (JVM) at run time and compile suitable bytecode sequences into native machine code. While using a JIT compiler, the hardware is able to execute the native code, as compared to having the JVM interpret the same sequence of bytecode repeatedly and incurring overhead for the translation process. This subsequently leads to performance gains in the execution speed, unless the compiled methods are executed less frequently.

1. **Difference between access specifier and access modifier.**

**Note:** There is no difference between access specifier and access modifier in Java. They both mean the same. Access modifier is the new and official term used instead of access specifier. Java provides four access modifiers to set access levels for classes, variables, methods and constructors.

## **1) Access Modifiers**

**Access modifiers are the keywords which are used with classes, variables, methods, and constructors to control their access level.**

**There are four access modifiers in Java.**

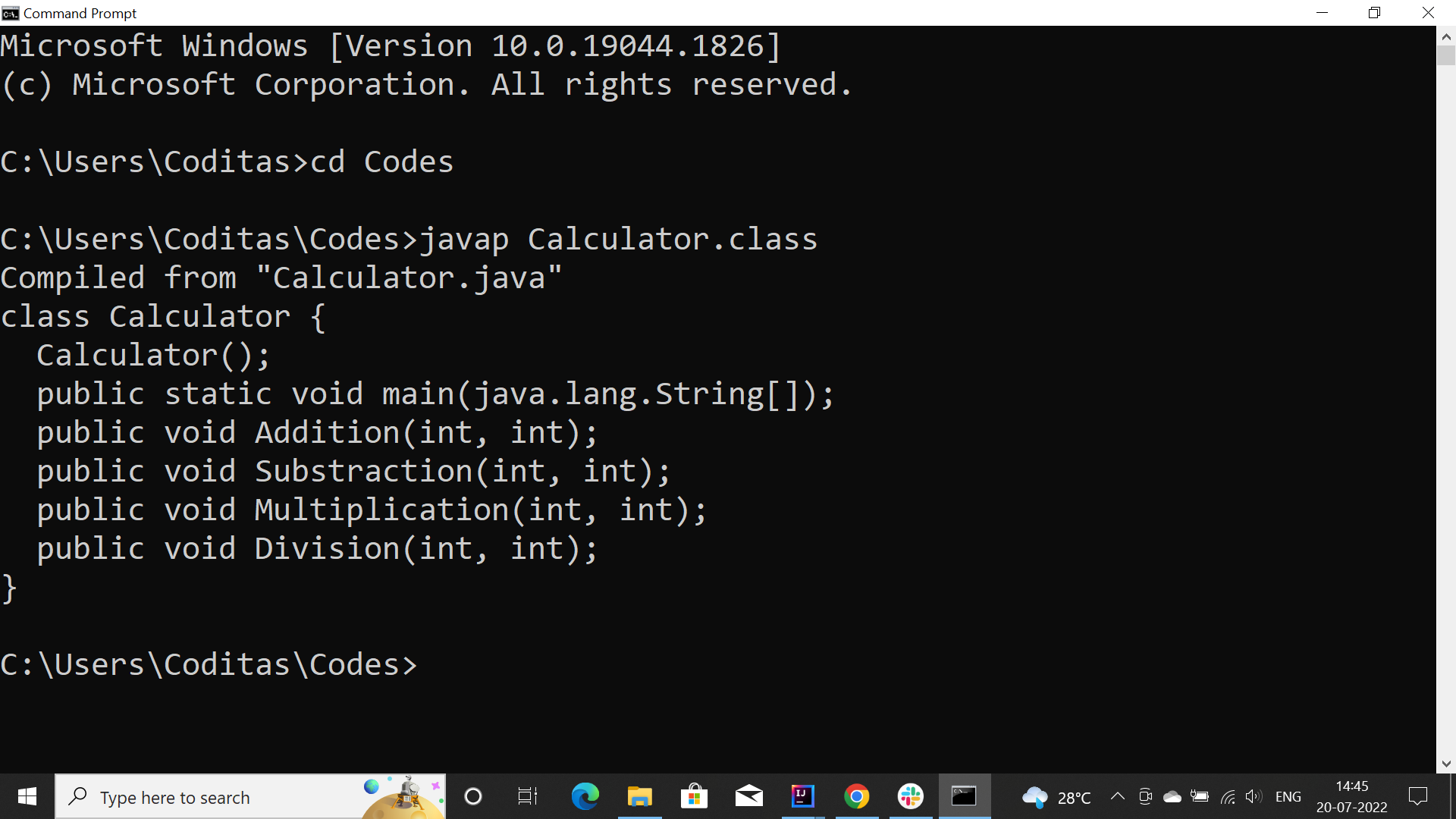
* Default  
  **When no access modifier is specified, java is treated as a default modifier. The scope of the default modifier is limited to within the package.**
* Public  
  **The scope of public modifier is to access everywhere and even outside the package.**
* Private  
  **The scope of private modifier is to access within the class itself.**
* Protected  
  **The scope of the protected modifier is limited within the package and all subclasses.**

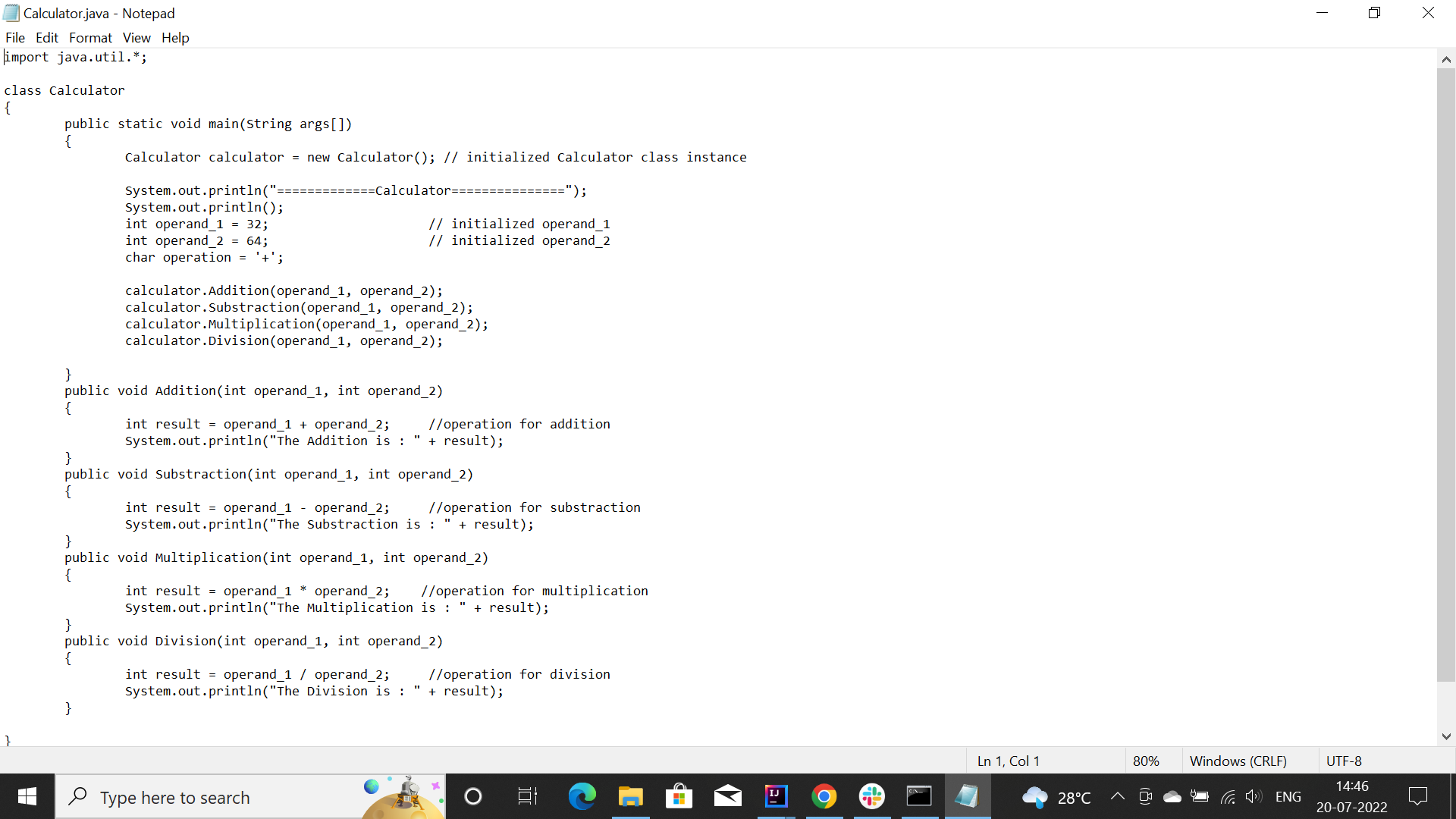
## **Non-access Modifiers**

**Non-access modifiers are those keywords that do not have anything related to the level of access but they provide a special functionality when specified.**

* Final  
  **Final keyword can be used with variable, method or class. It prevents from its content from being modified. When declared with class, it prevents the class from being extended.**
* Static  
  **The static modifier is used with class variables and methods which can be accessed without an instance of a class. Static variables have only single storage. All objects share the single storage of static variable. They can be accessed directly without any object. Static methods can also be declared. the main() method is the popular static method. Static blocks can also be declared and are executed before main() method.**
* Abstract  
  **abstract can be used with class and methods. An abstract class can never be instantiated and its purpose is only to be extended. Abstract methods are declared without body and end with a semicolon. If a class contains an abstract method, it should also be specified as abstract. A class which extends an abstract class must implement all of its abstract methods.**
* Synchronized  
  **It indicates that the method can be accessed only by one thread at a time.**
* Transient  
  **An instance variable is marked as transient to indicate the JVM to skip the particular variable when serializing the object containing it.**
* Volatile  
  **Volatile keyword is used to mark a java variable as "being stored in main memory". It means that every read of a volatile variable will be read from the computer's main memory and not from the CPU cache and writes to a volatile variable will be written to main memory and not just to the CPU cache.**
* Strictfp  
  **Strictfp keyword in java ensures that you will get the same result on every platform if you perform operations in the floating-point variable.**

1. **Explore the class file of any existing code and check it.**

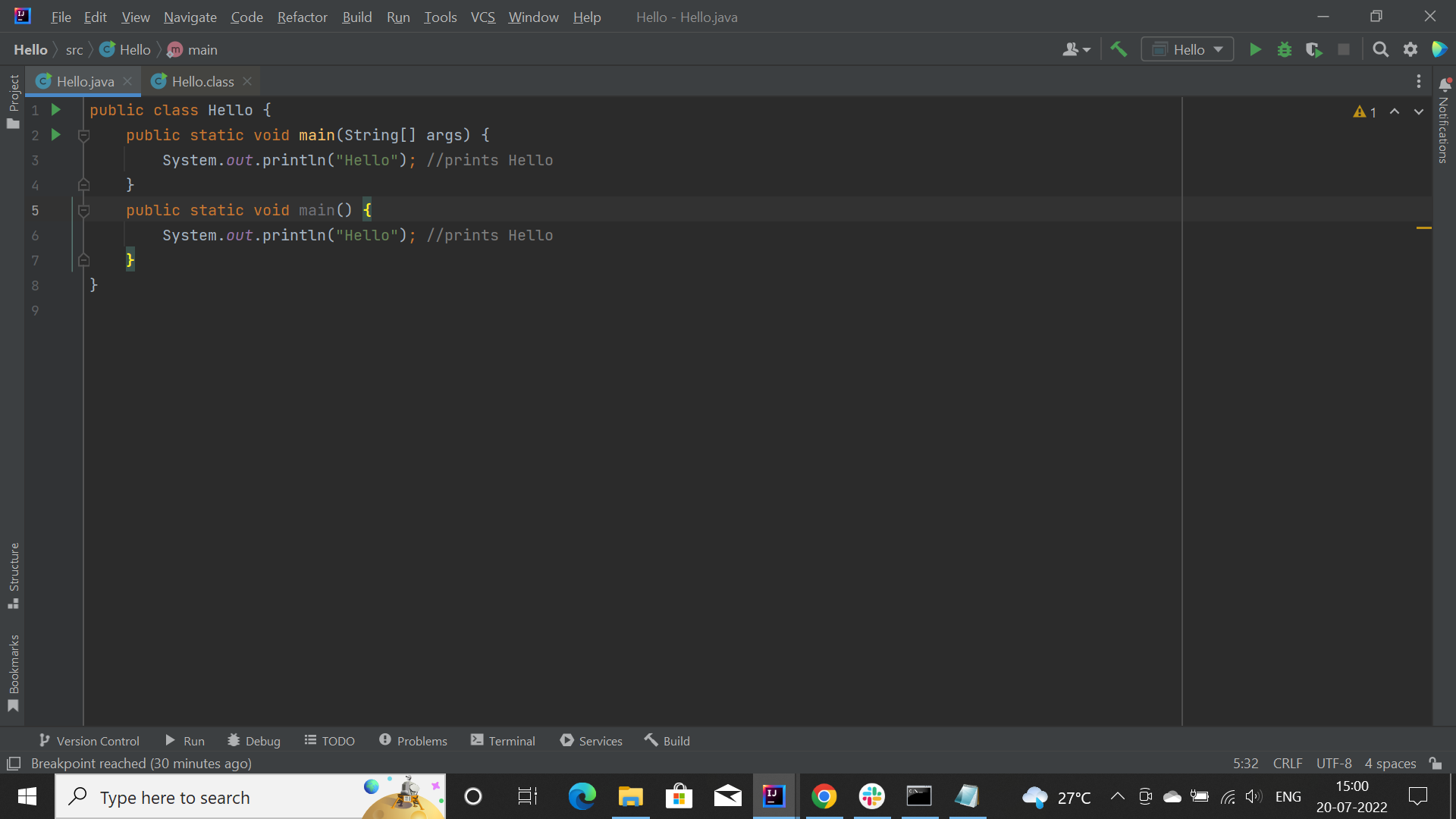
****

****

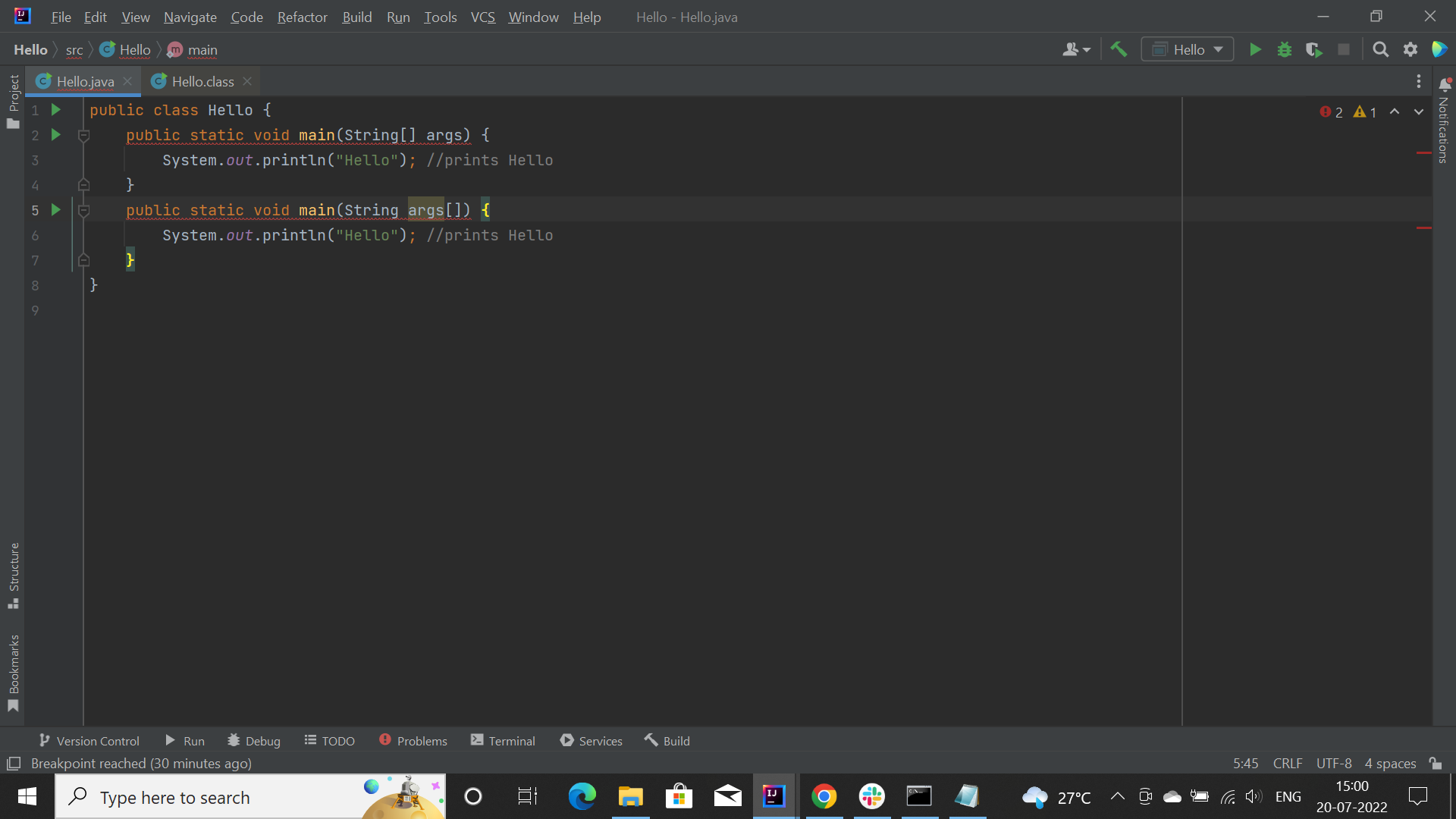
1. **Can we have multiple main methods in class ?**

**Case : Multiple main methods in same class**

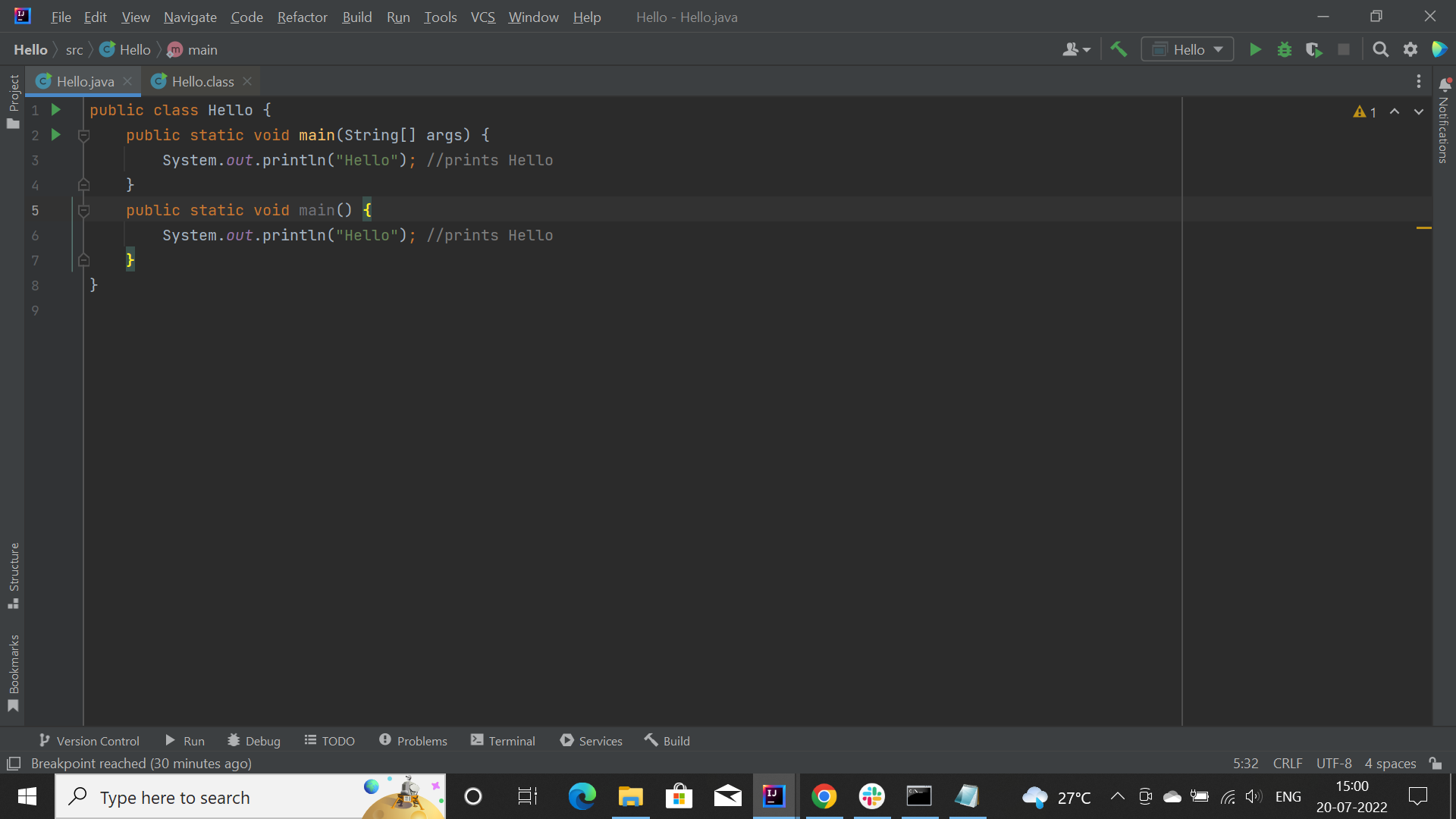
**Yes, we can create multiple main methods. But, Java’s main method accepts string array as an argument. It is also called a command-line argument and it can pass from the command line in the main method.**

****

**Hence, we can overload main method.**

****

1. **Can we overload/ override main method ?**

****

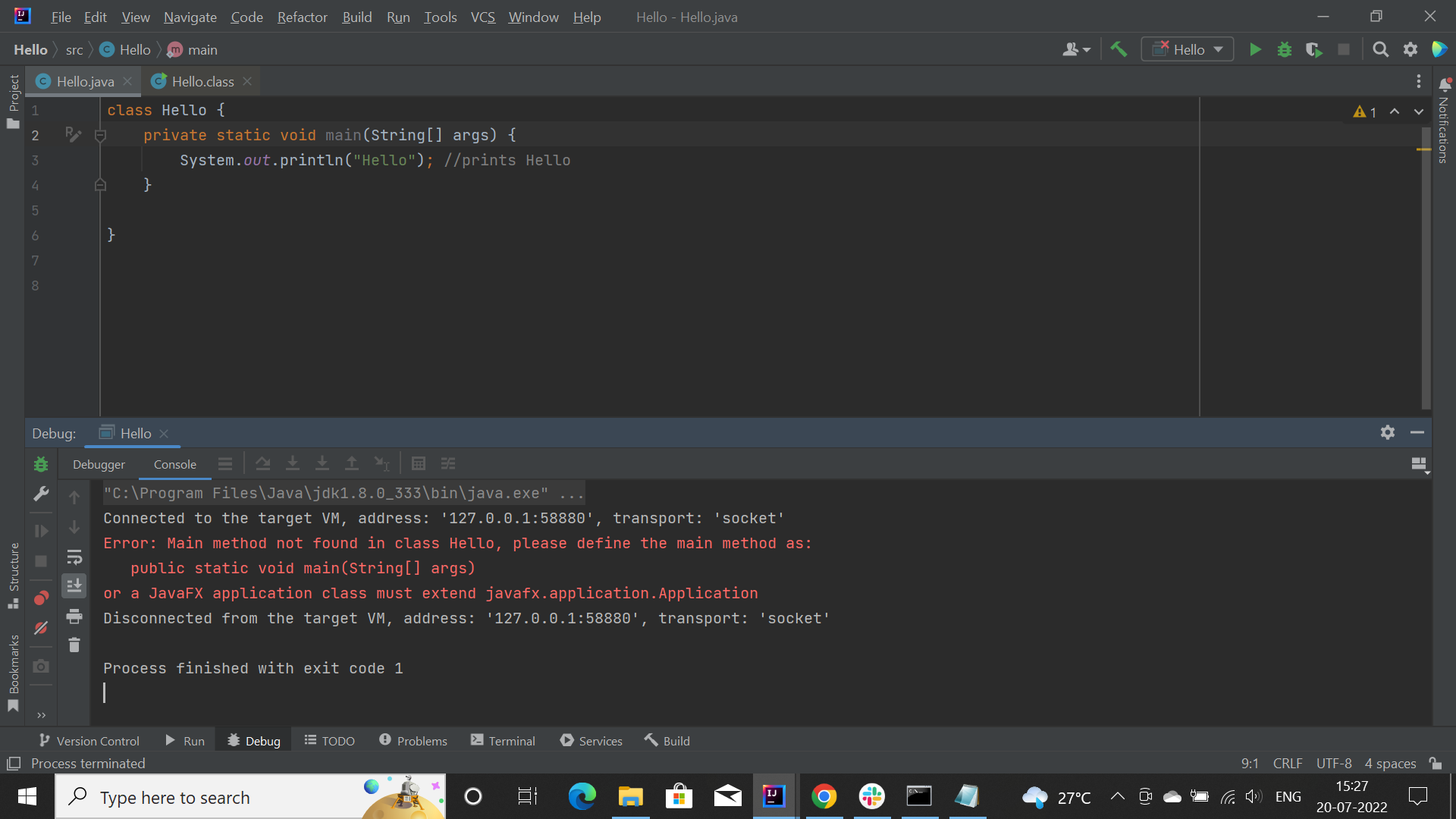
**Yes, we can overload main method**

**Override:**

No, we cannot override main method of java because a static method cannot be overridden. The static method in java is associated with class whereas the non-static method is associated with an object.

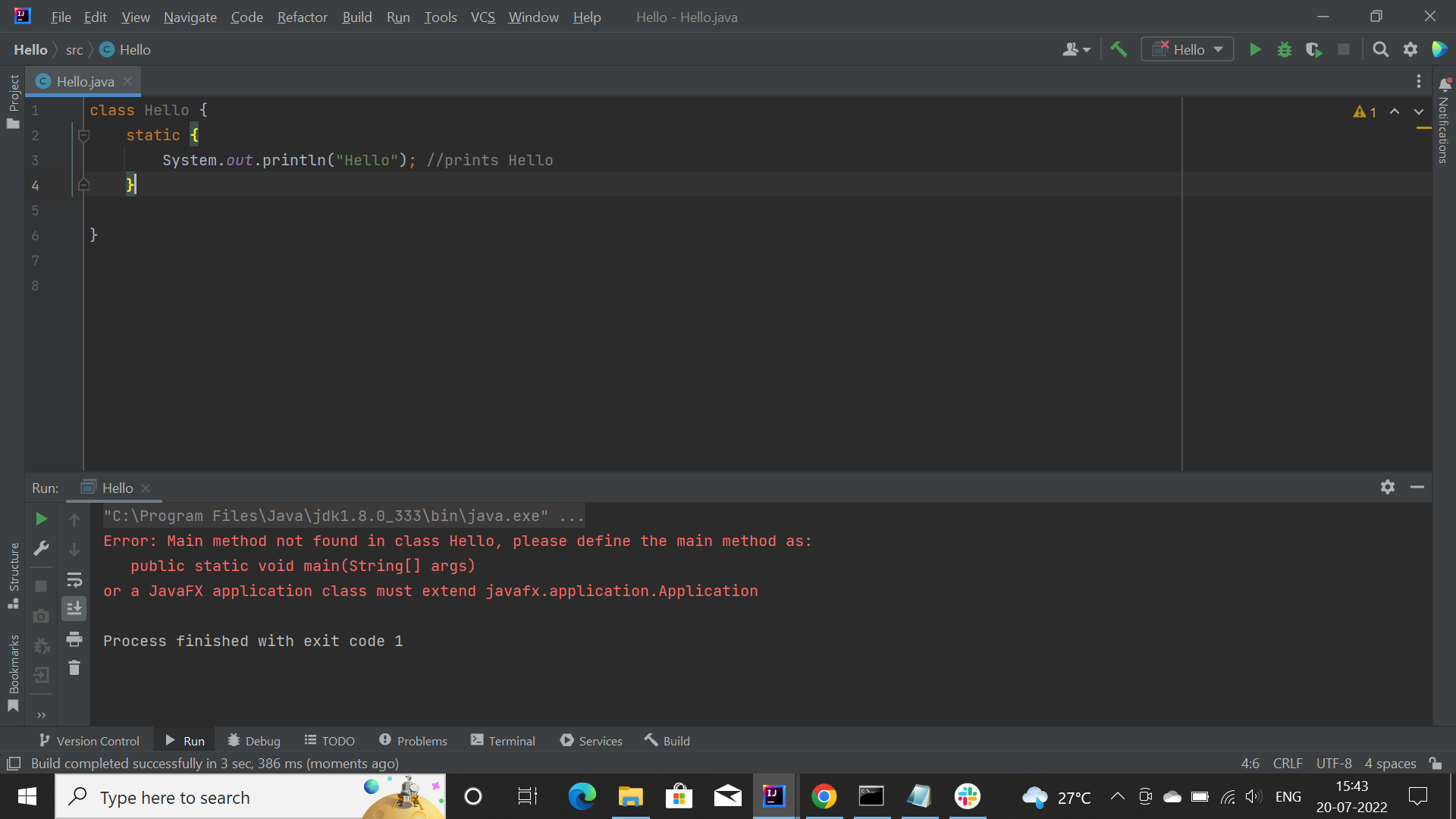
1. **Can write main method as private, protected and default ?**

Yes, You can define the main method in your program with private, protected or, default (none) modifier, the program gets compiled without compilation errors. But, at the time of execution JVM does not consider this as the entry point of the program. It searches for the main method which is public, static, with return type void, and a String array as an argument.



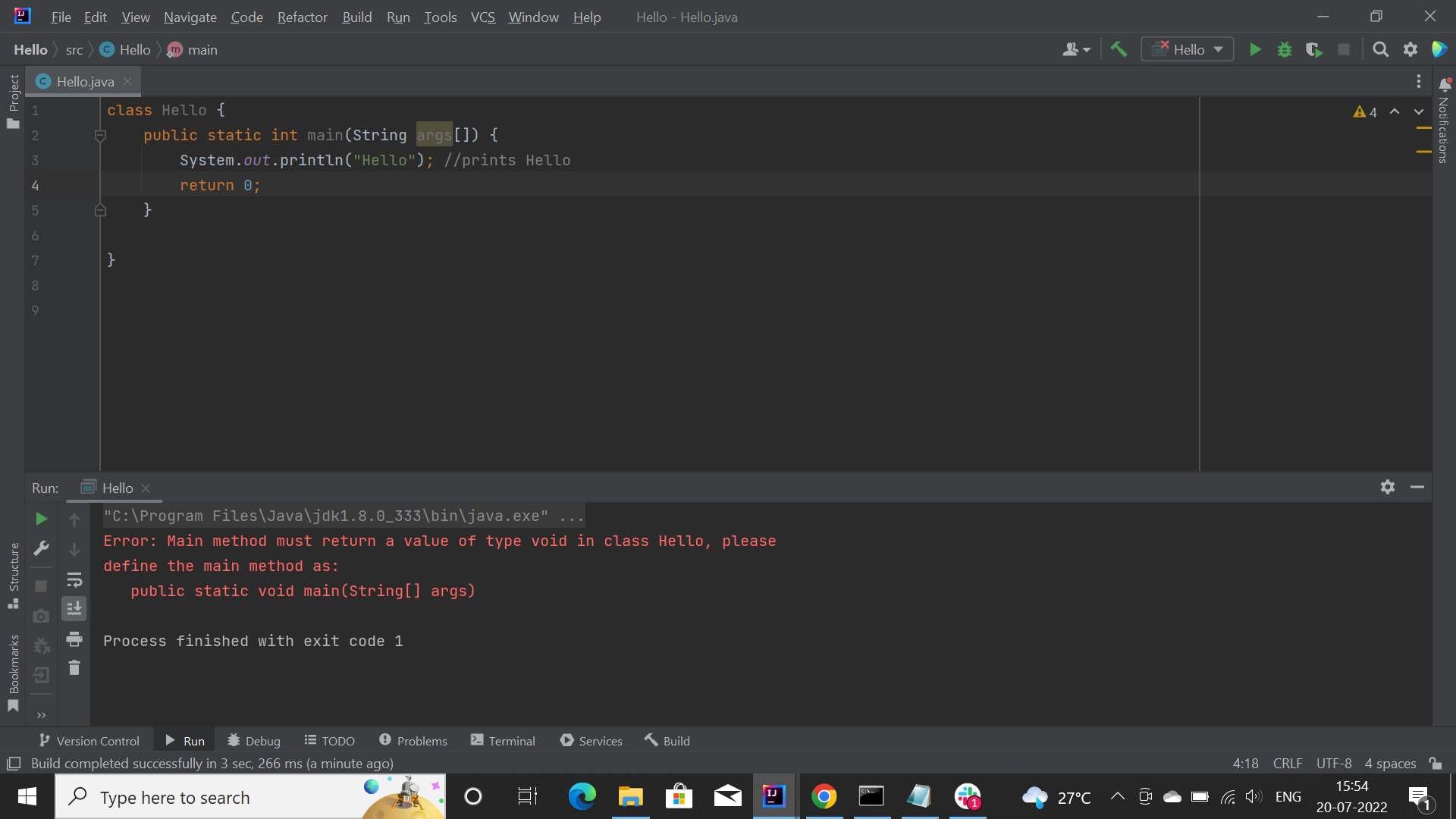
The problem can be solved by extending class with javafx application class as shown

1. **Without the main method, can we execute our code ? How ?**

****

Before, version 7.7, we were able to execute the code using static block without main method. After which, main method is required

1. **Can we change the return type of main method ?**

****

You can return an int with System.exit().

Returning anything other than an integer doesn't make much sense, as the OS expects an integer. In case nothing is returned the default is 0, which means OK. Other values typically are used to signal errors or special conditions.

1. **Explore strictfp**

Java strictfp keyword ensures that you will get the same result on every platform if you perform operations in the floating-point variable. The precision may differ from platform to platform. That is why the Java programming language has provided the strictfp keyword, so that you get the same result on every platform. So, now you have better control over the floating-point arithmetic.