Communciation System Lab’s Report  
Homework CH3



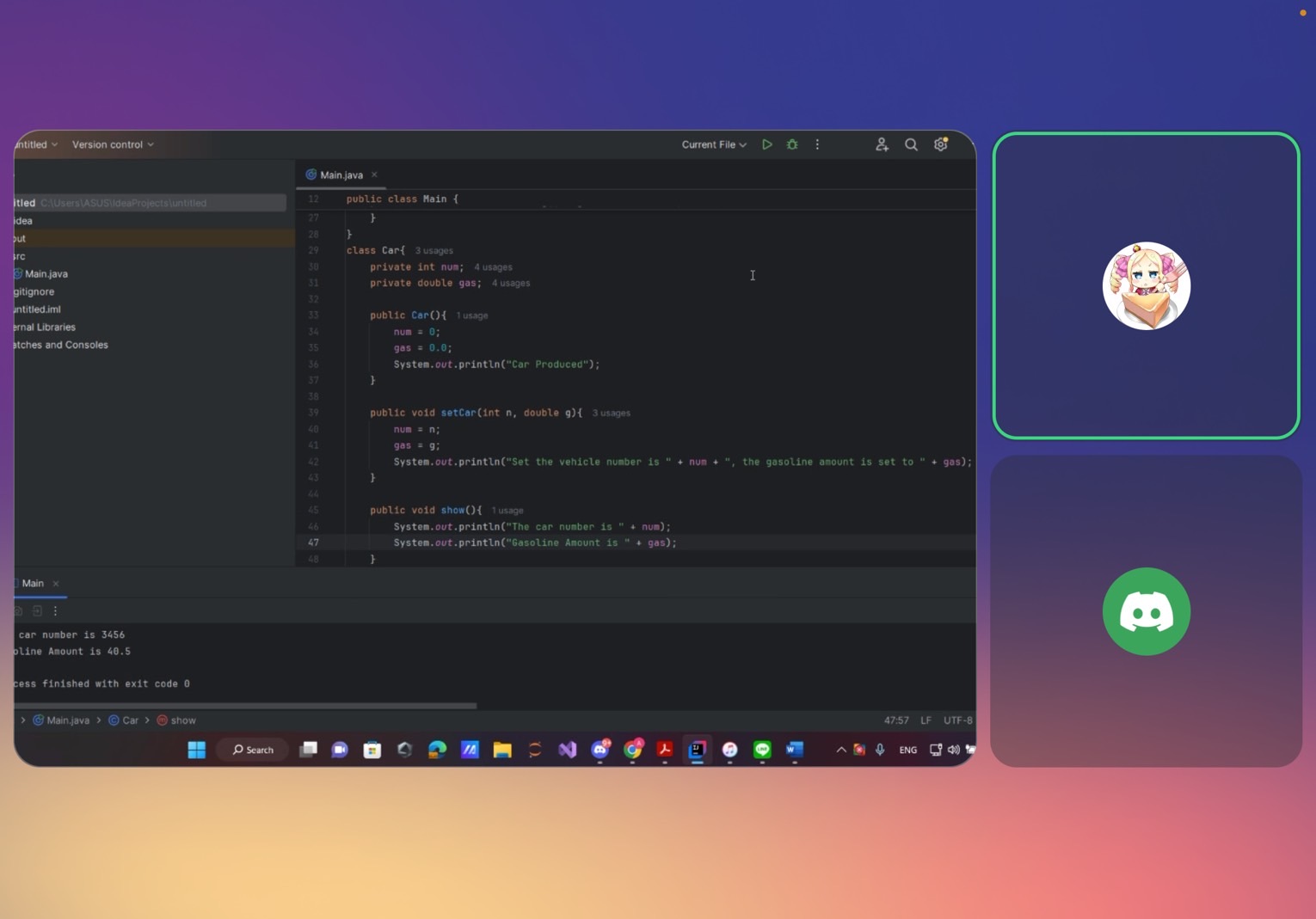
Name: 雷智豪

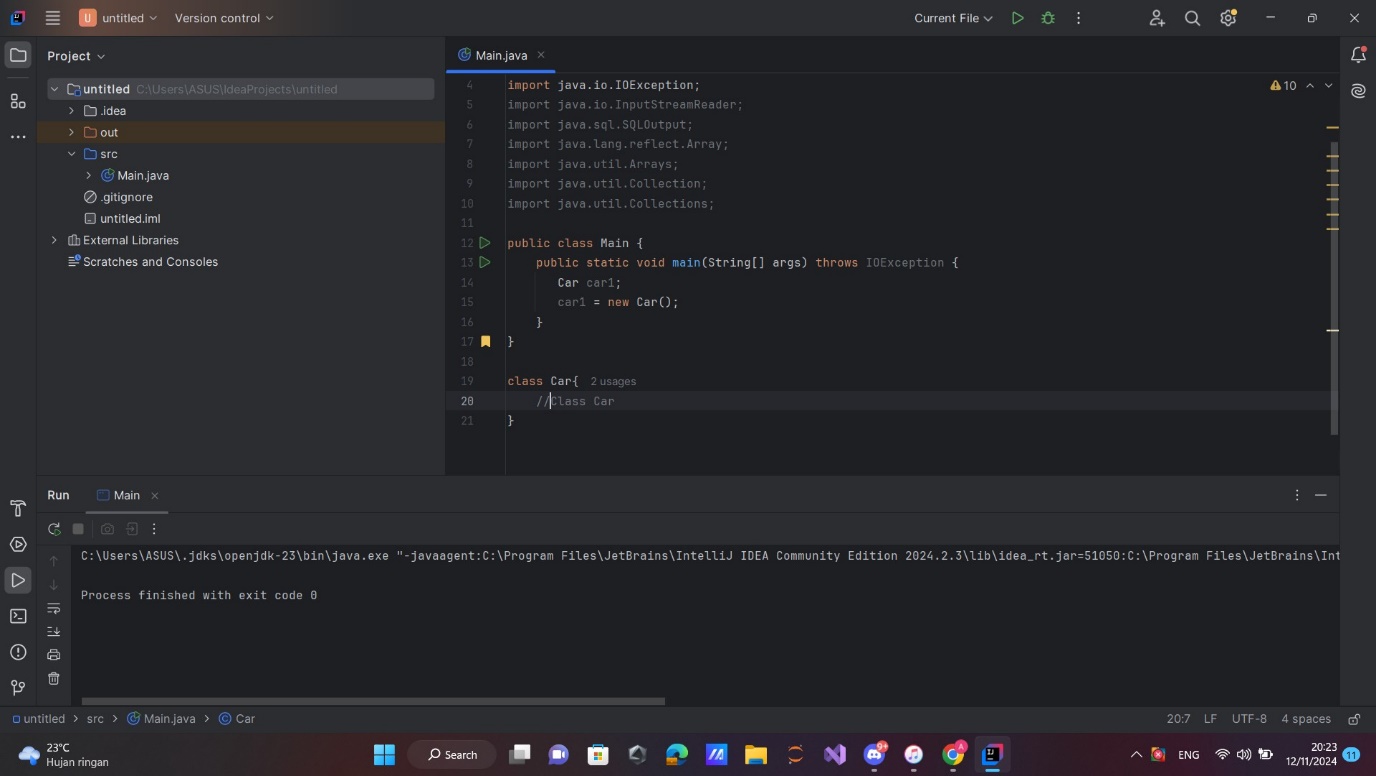
Class: 電子三甲

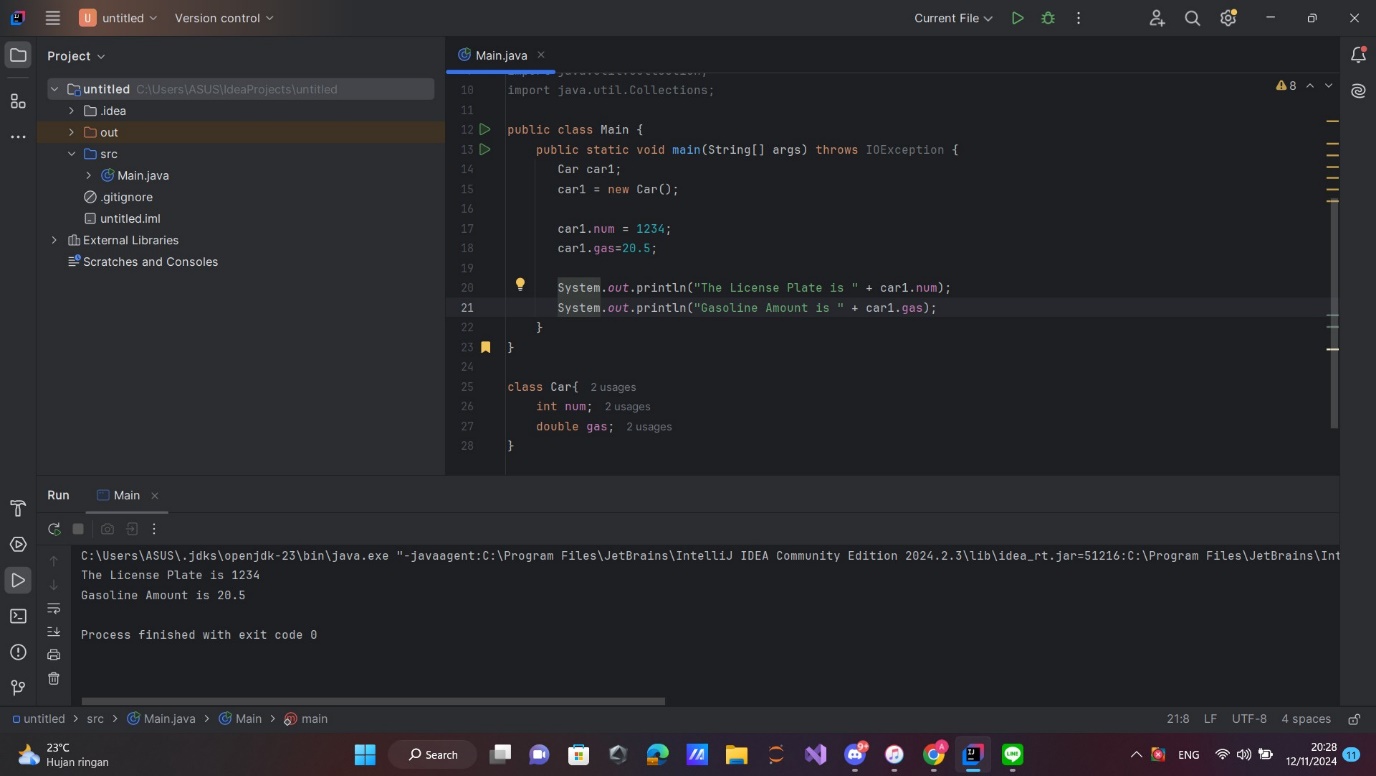
Student ID: 111360155

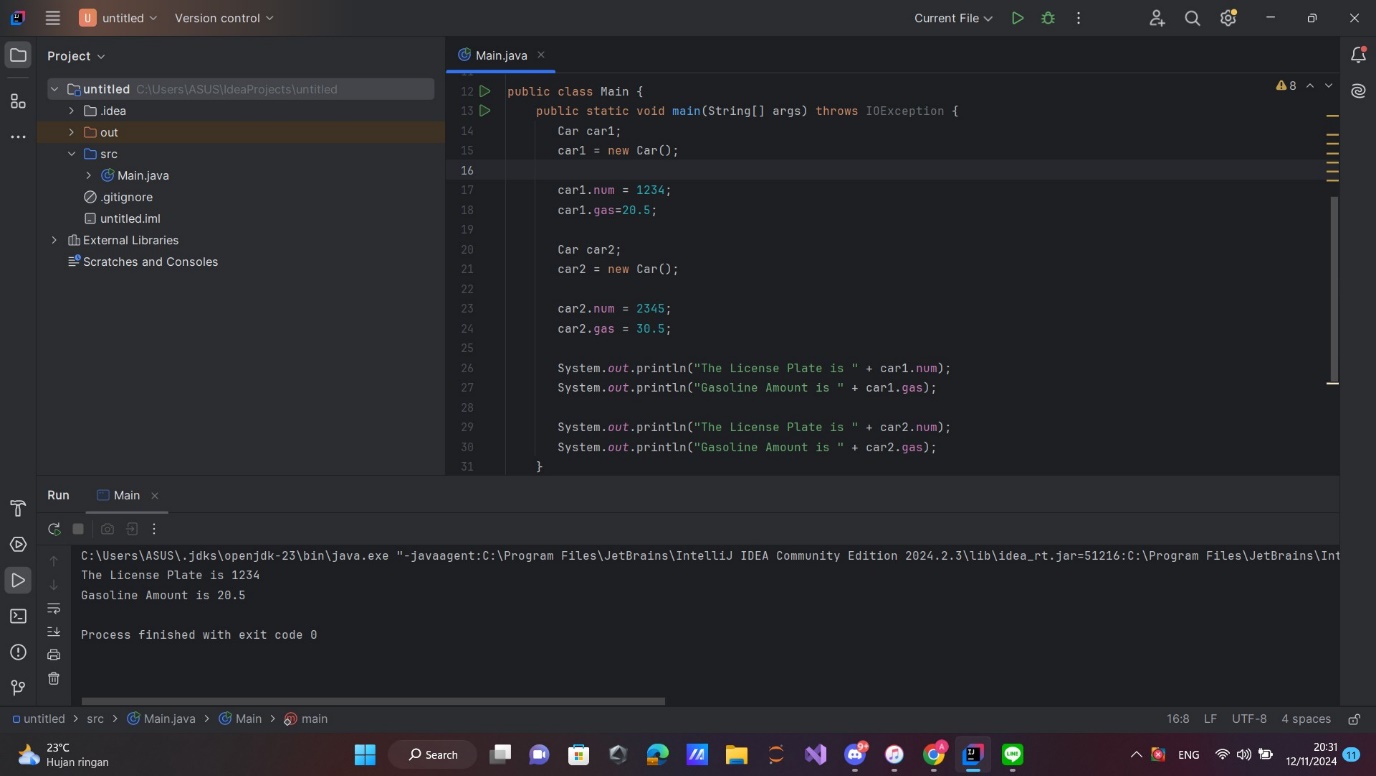
Group : 雷智豪 111360155

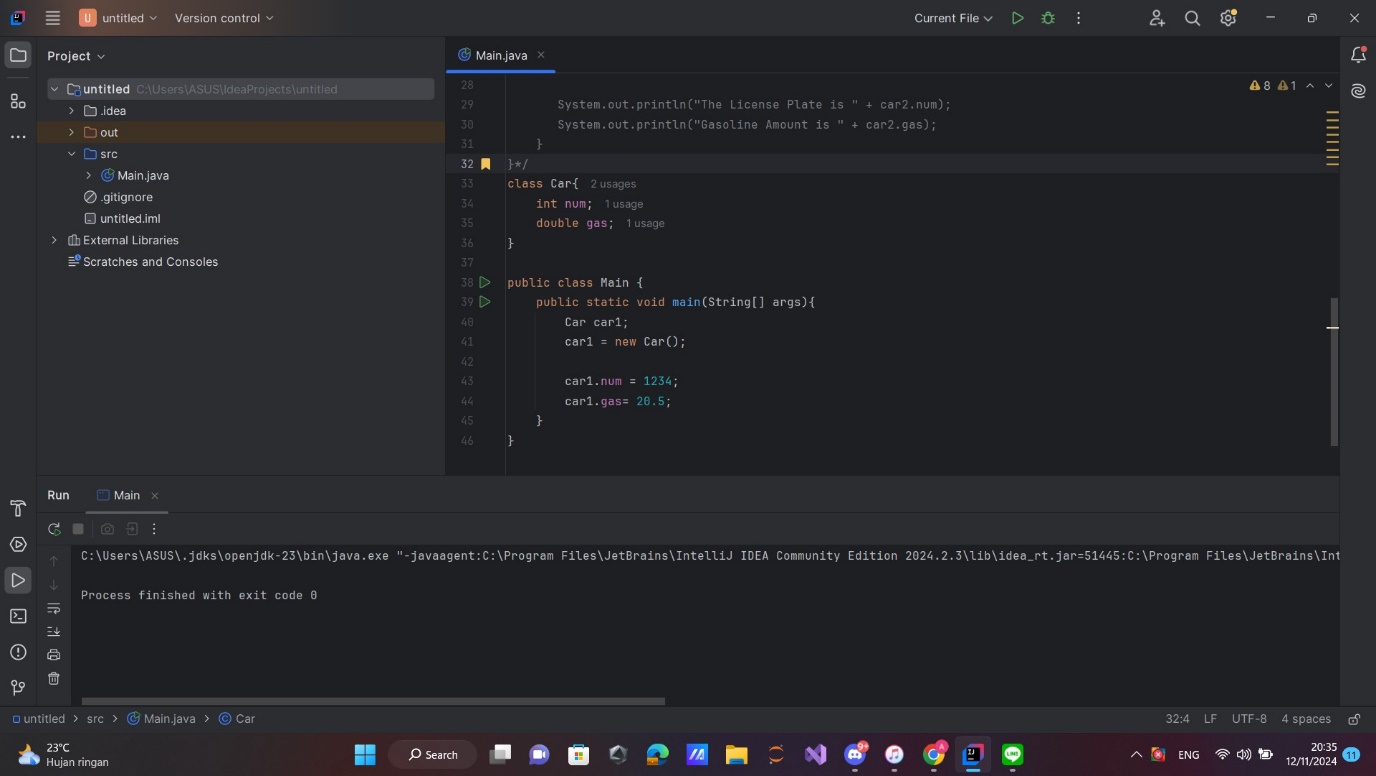
鄭光佑 109360105

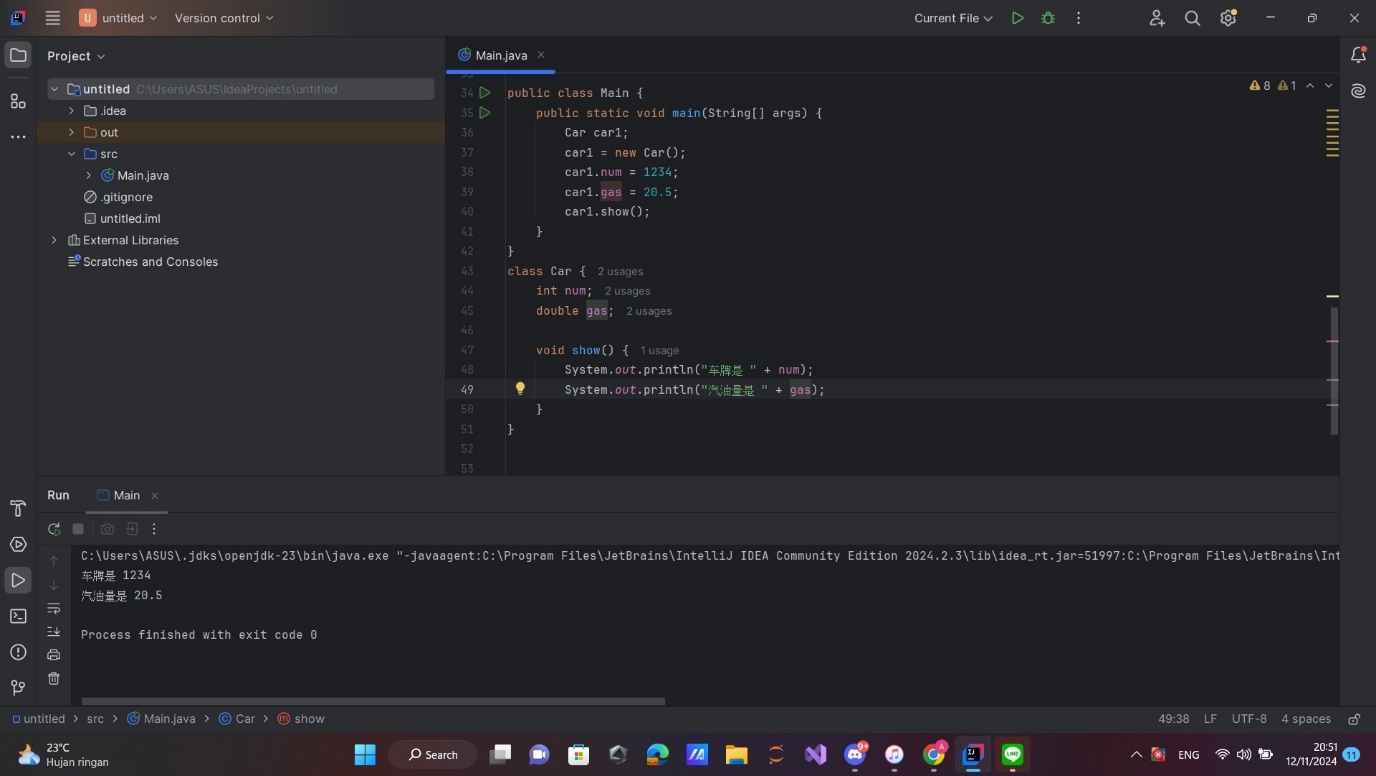


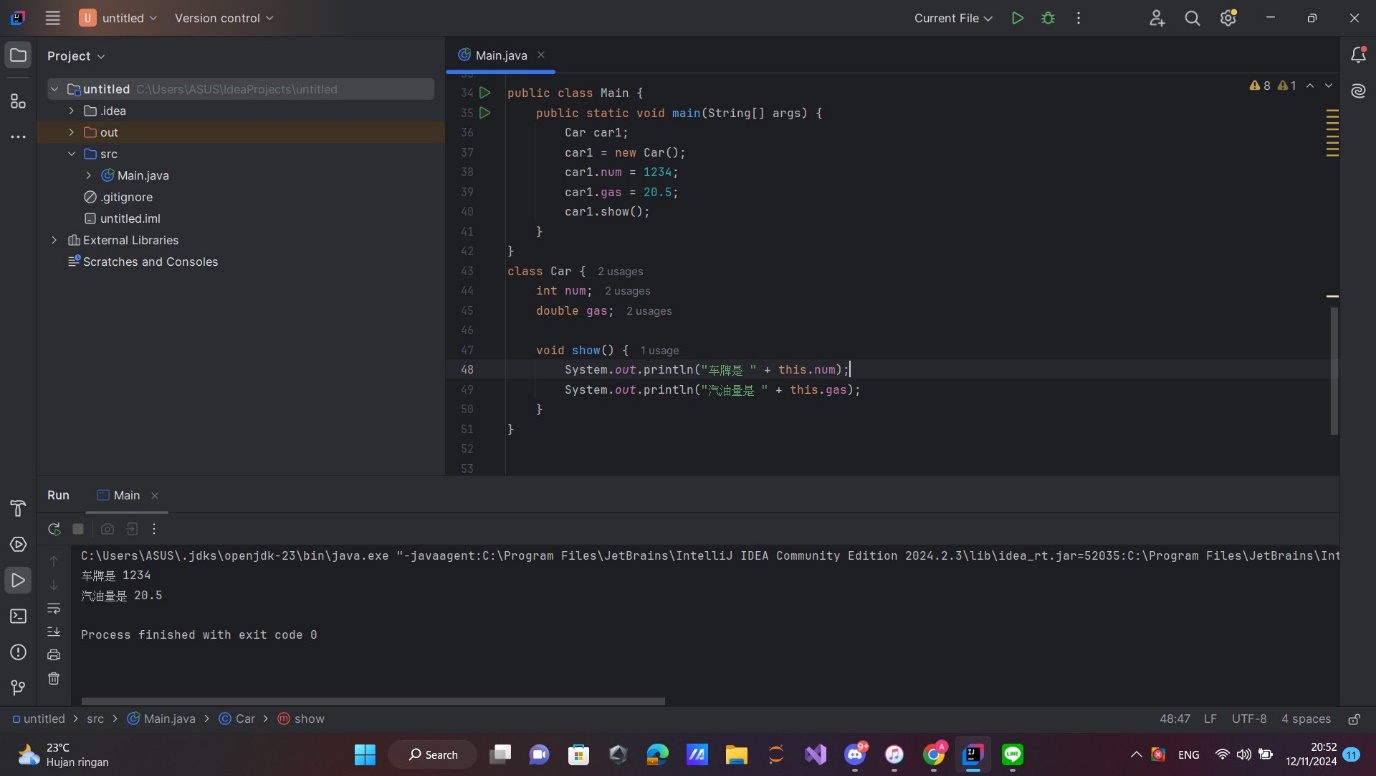


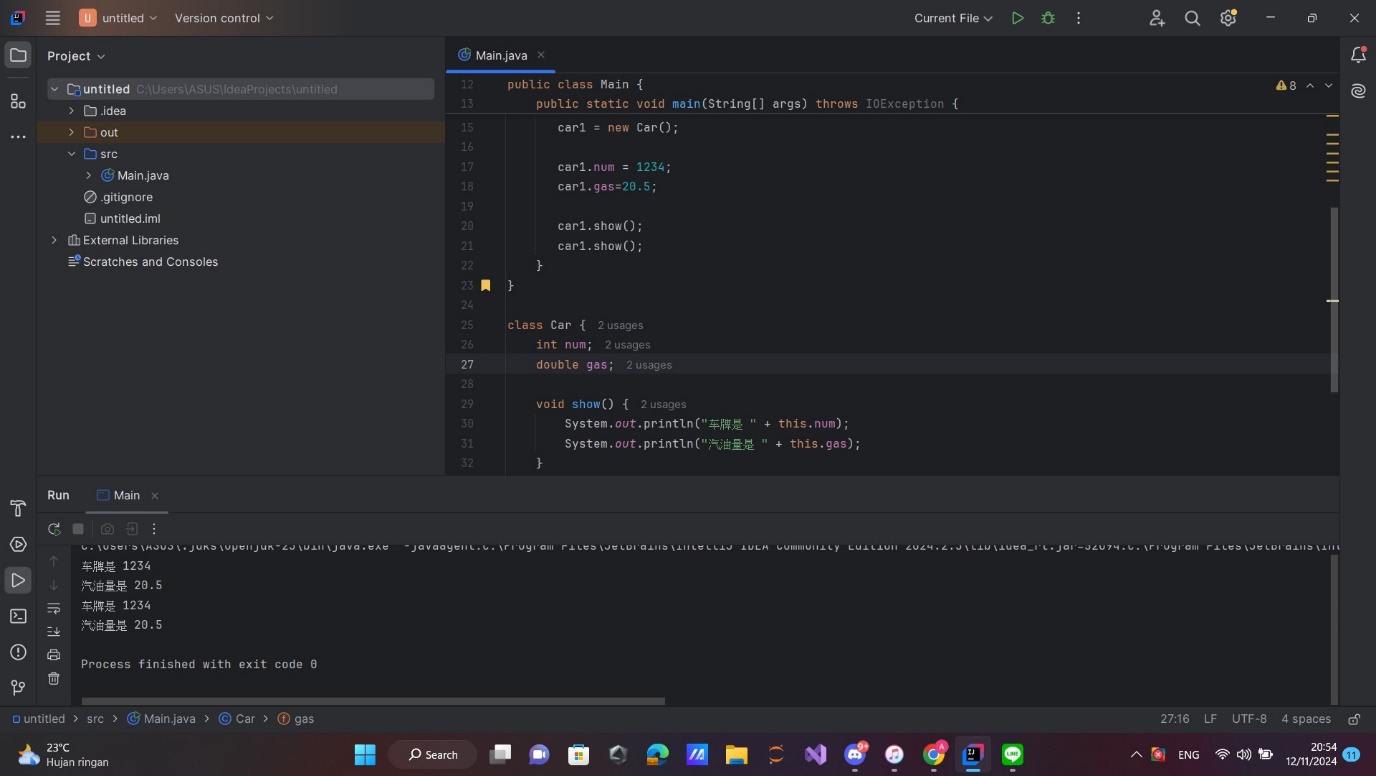


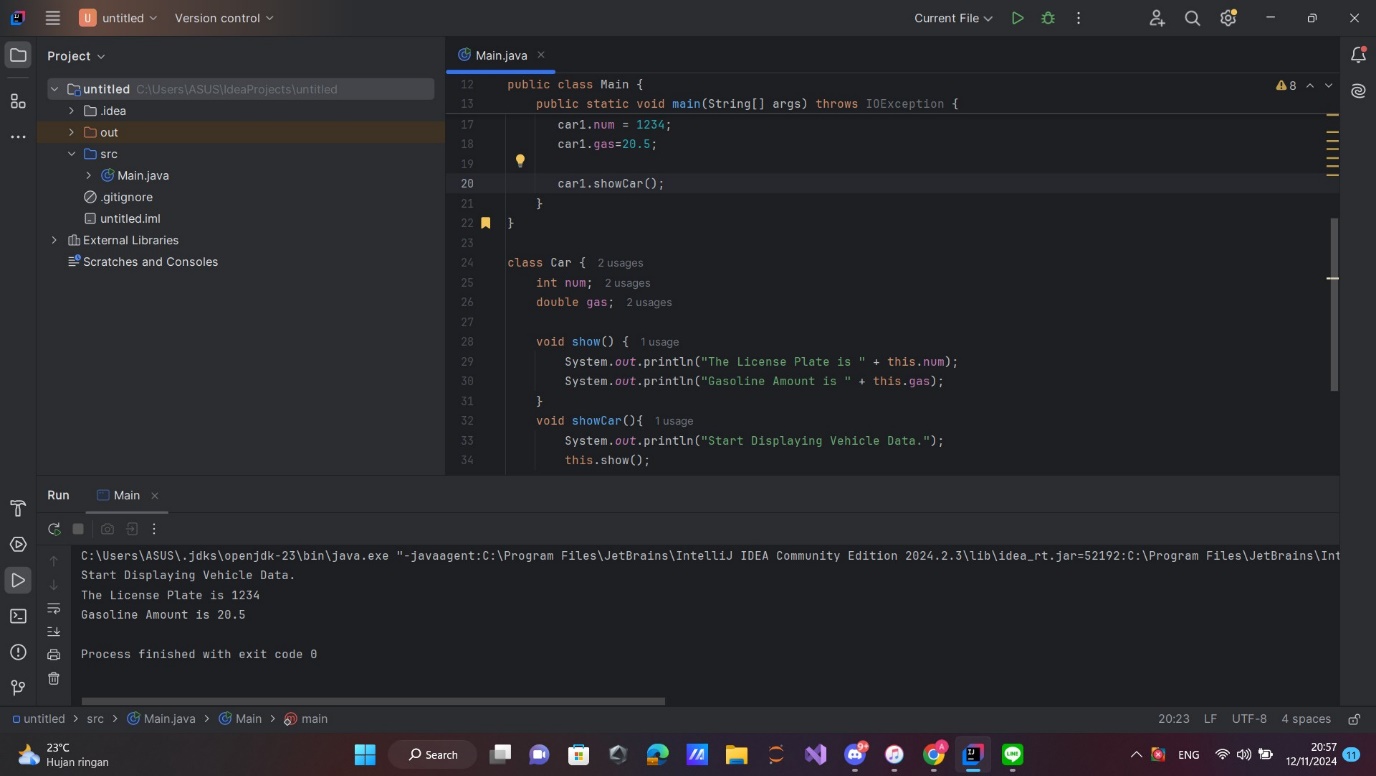


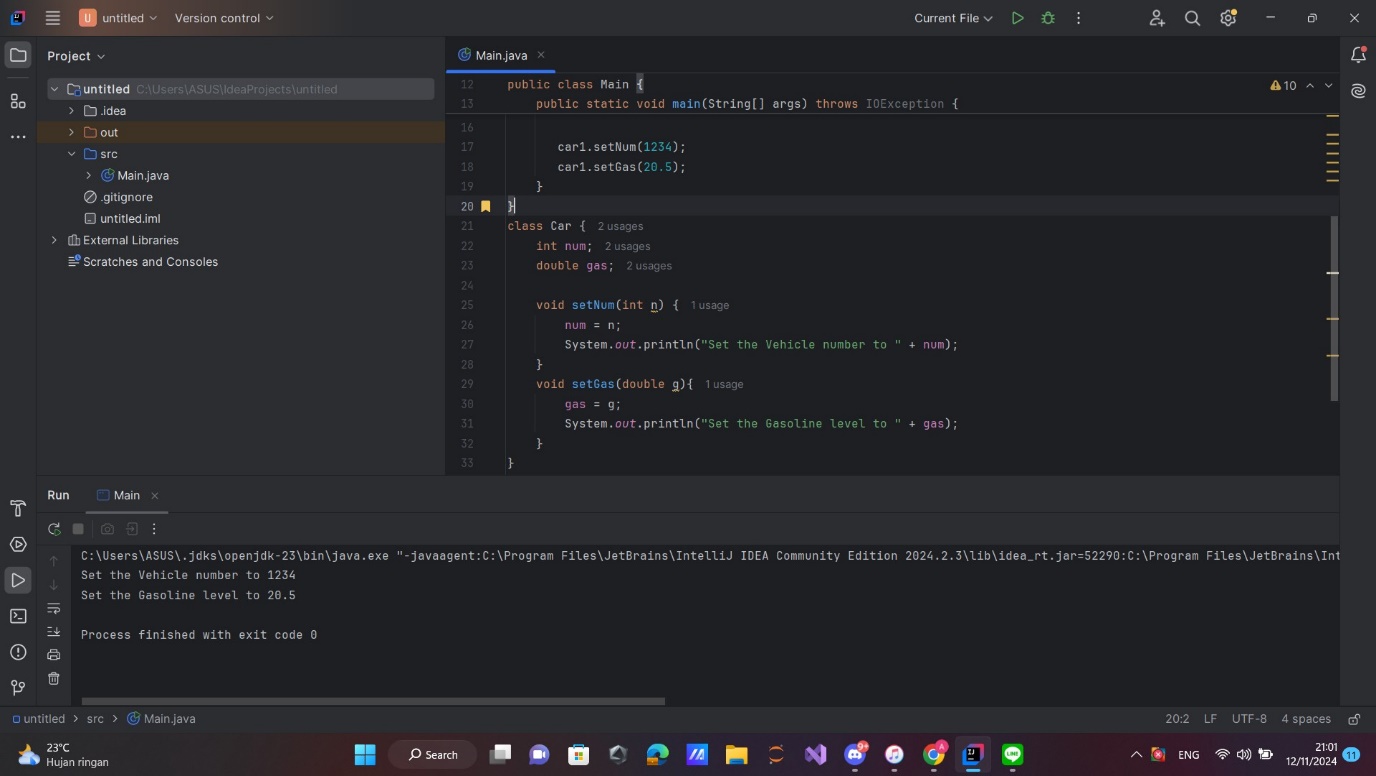


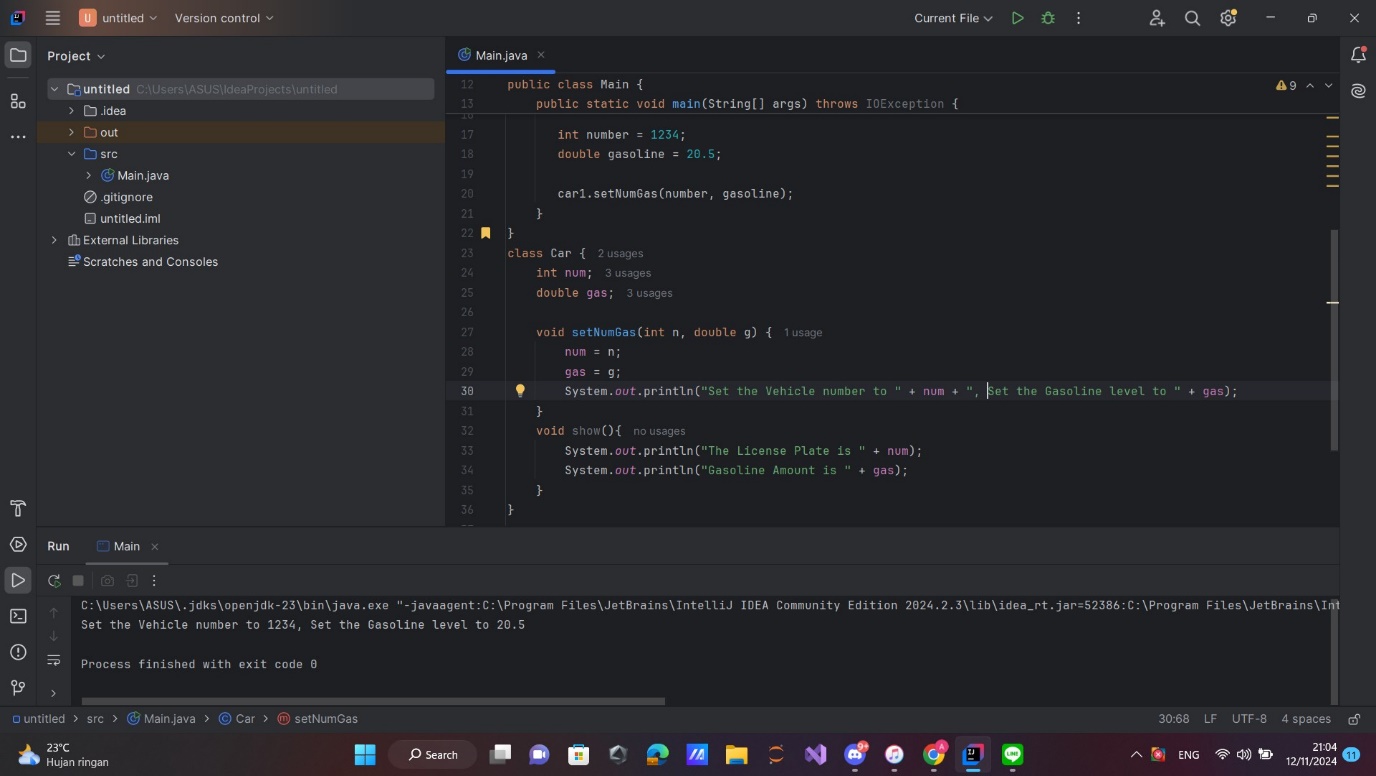


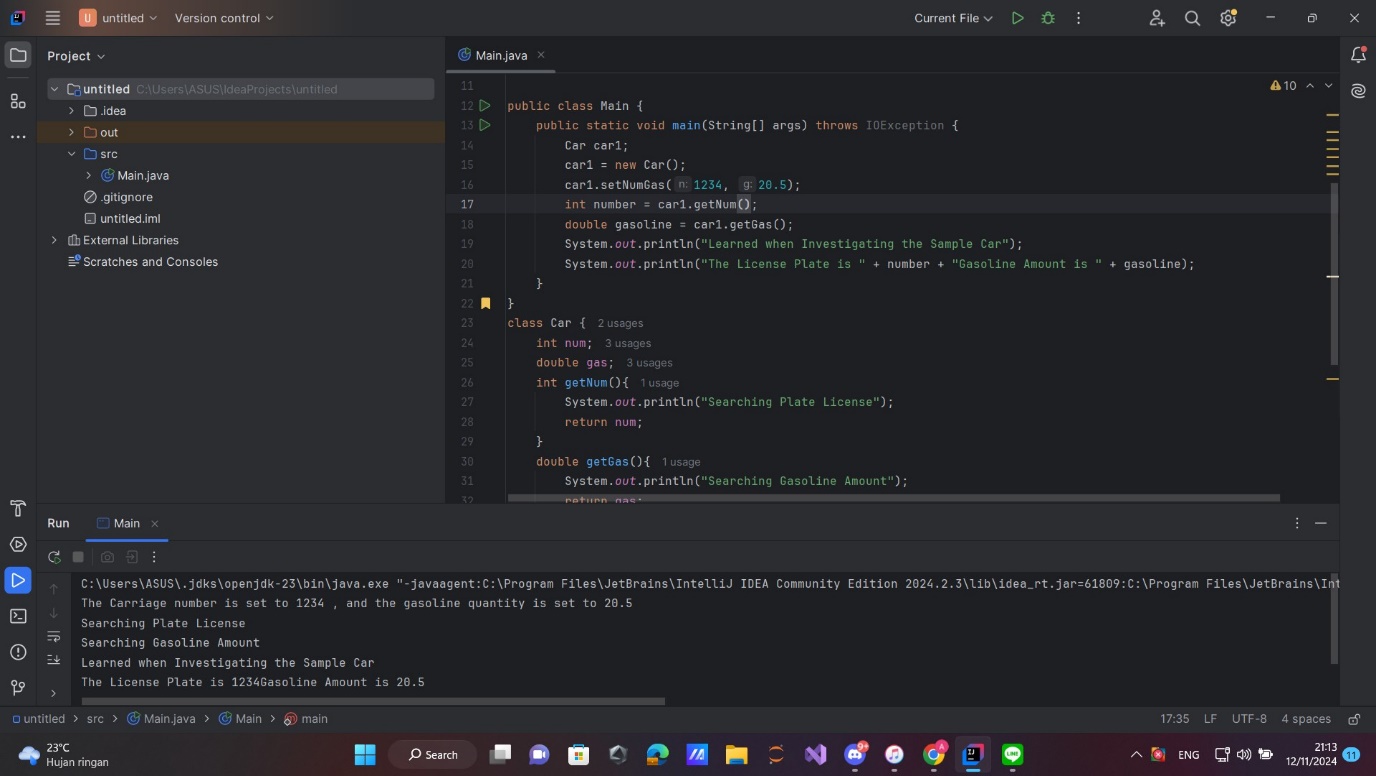


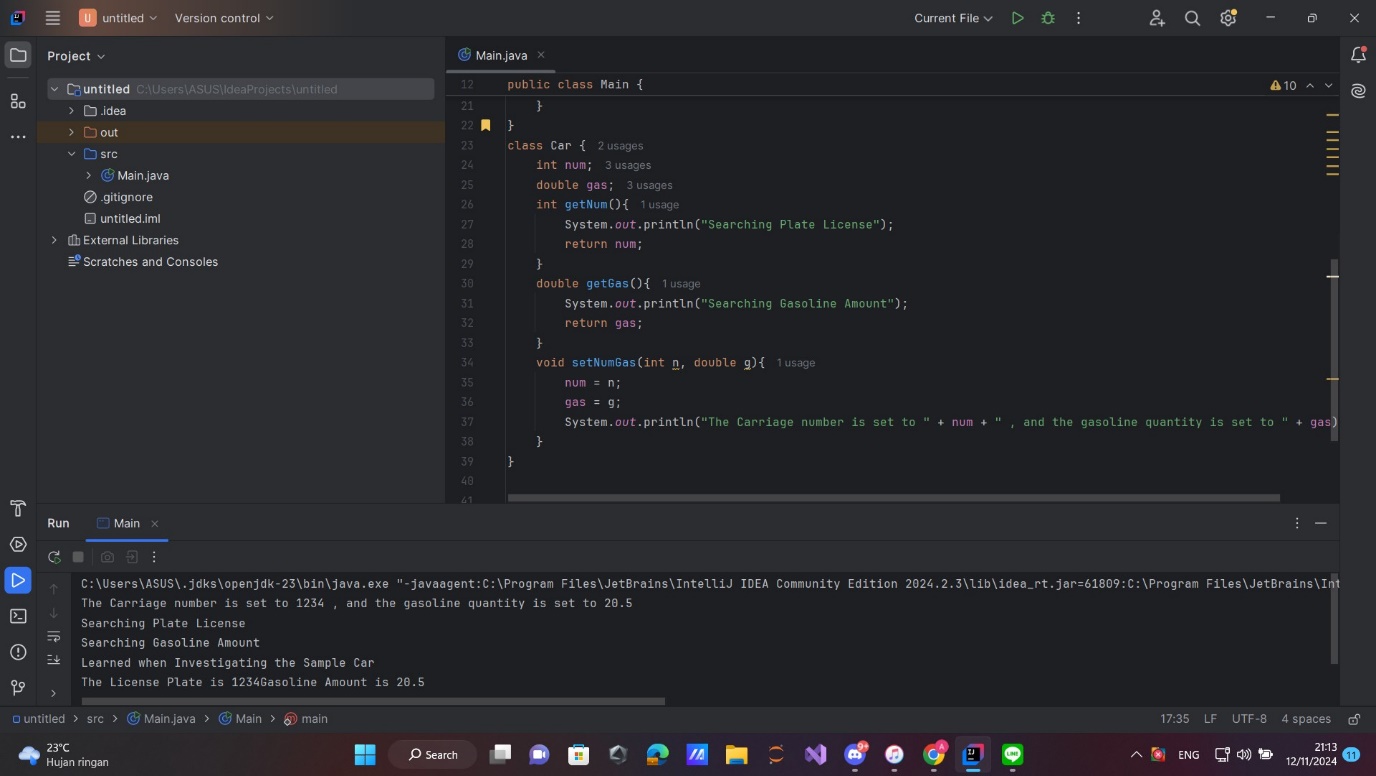


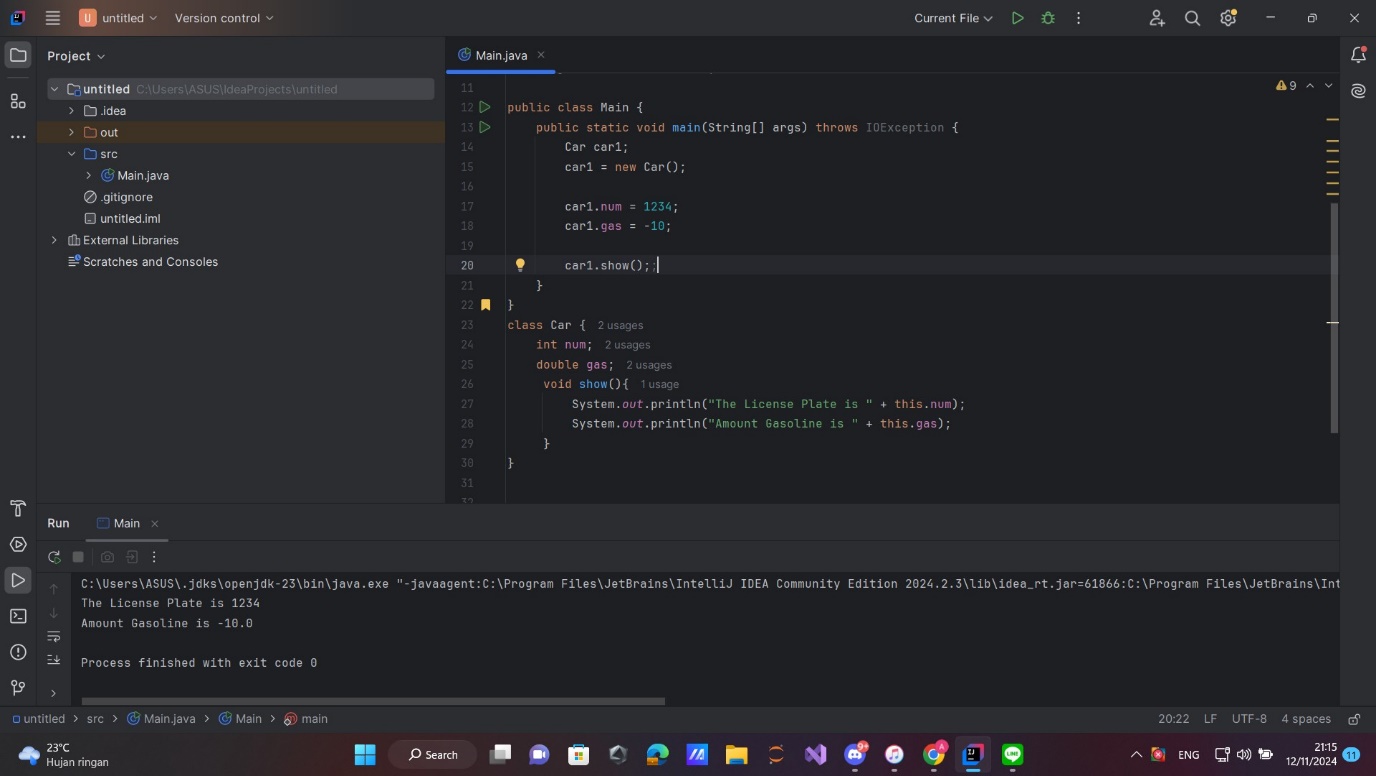


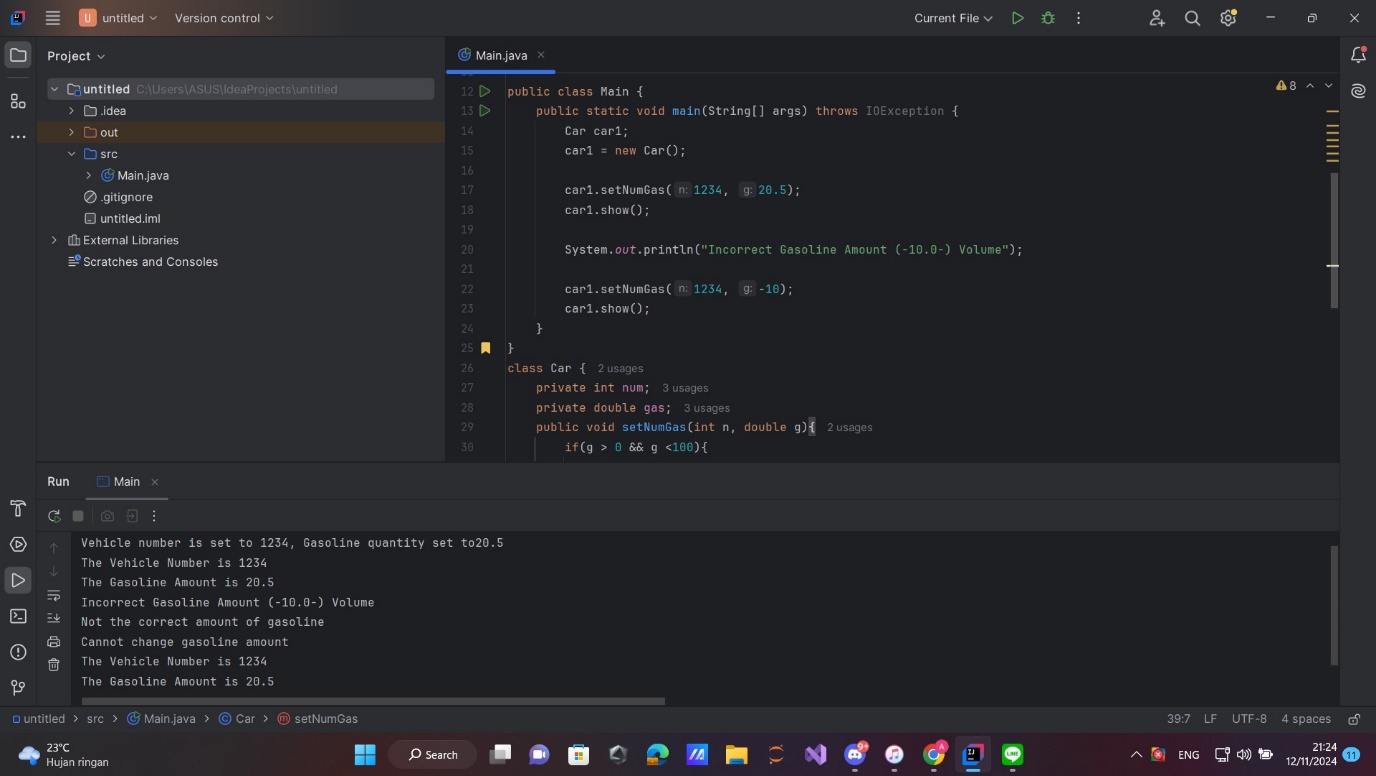


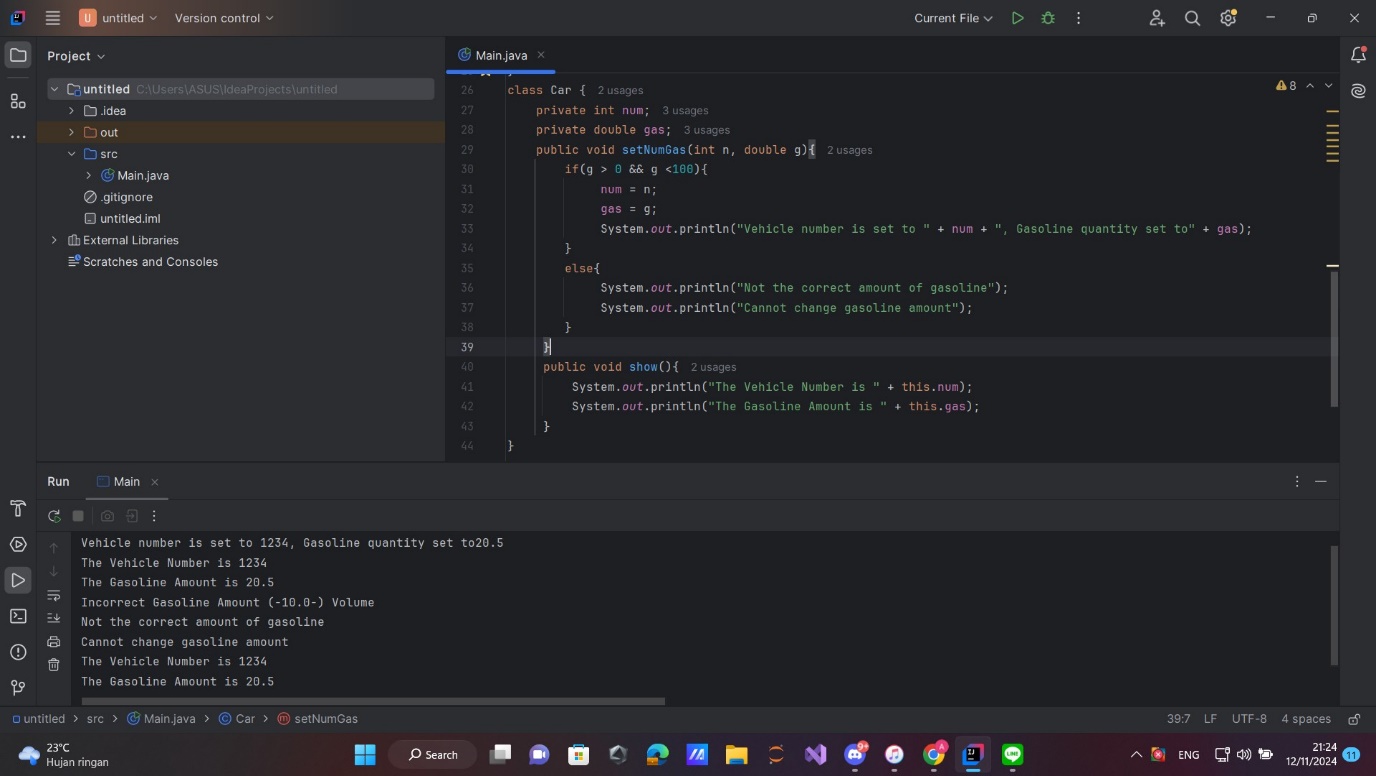


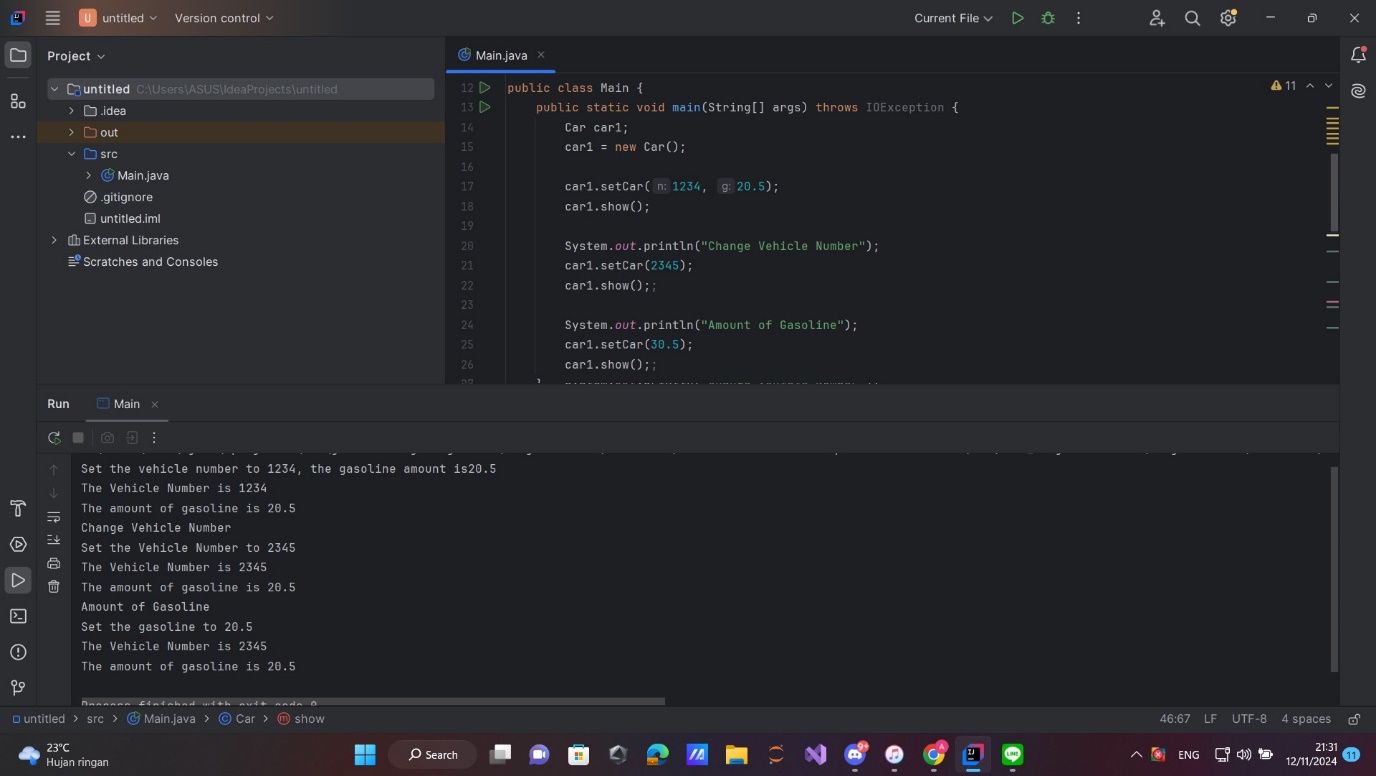


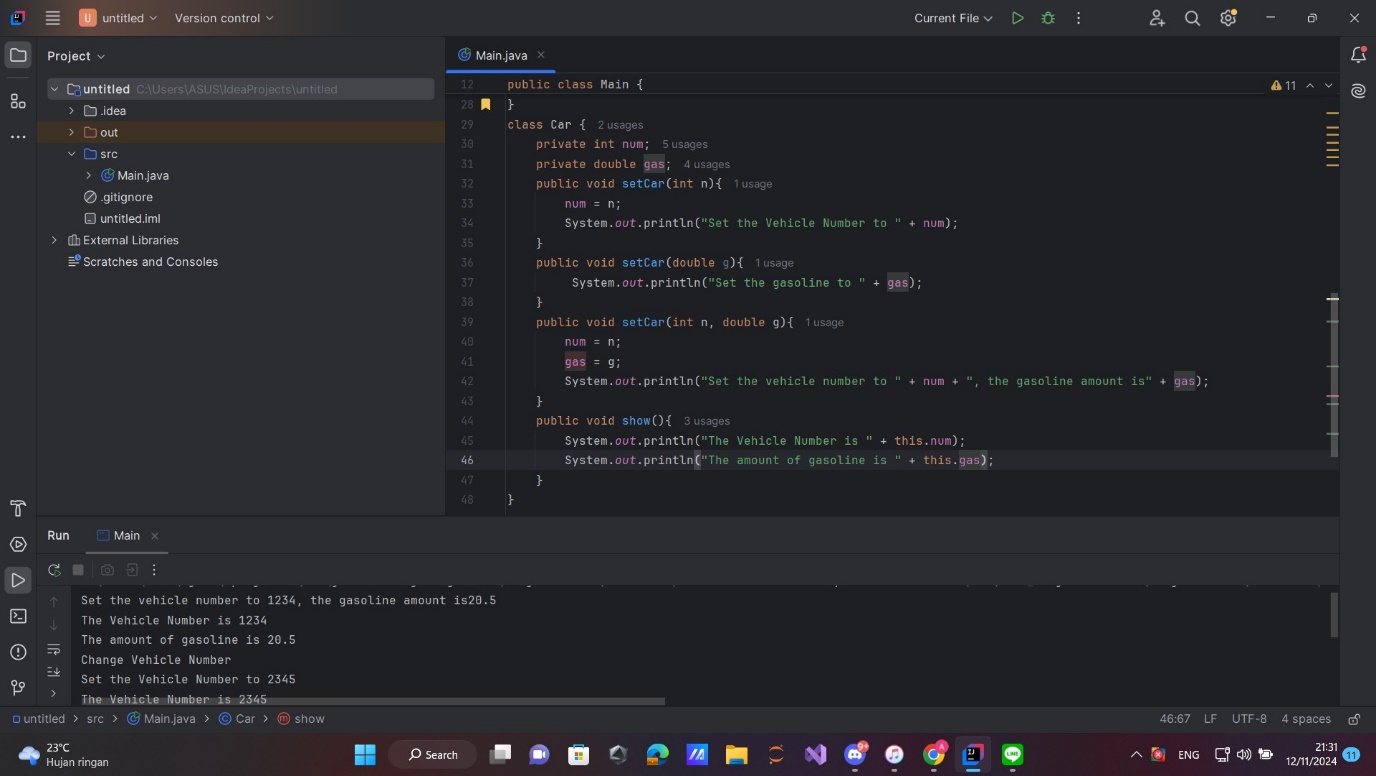


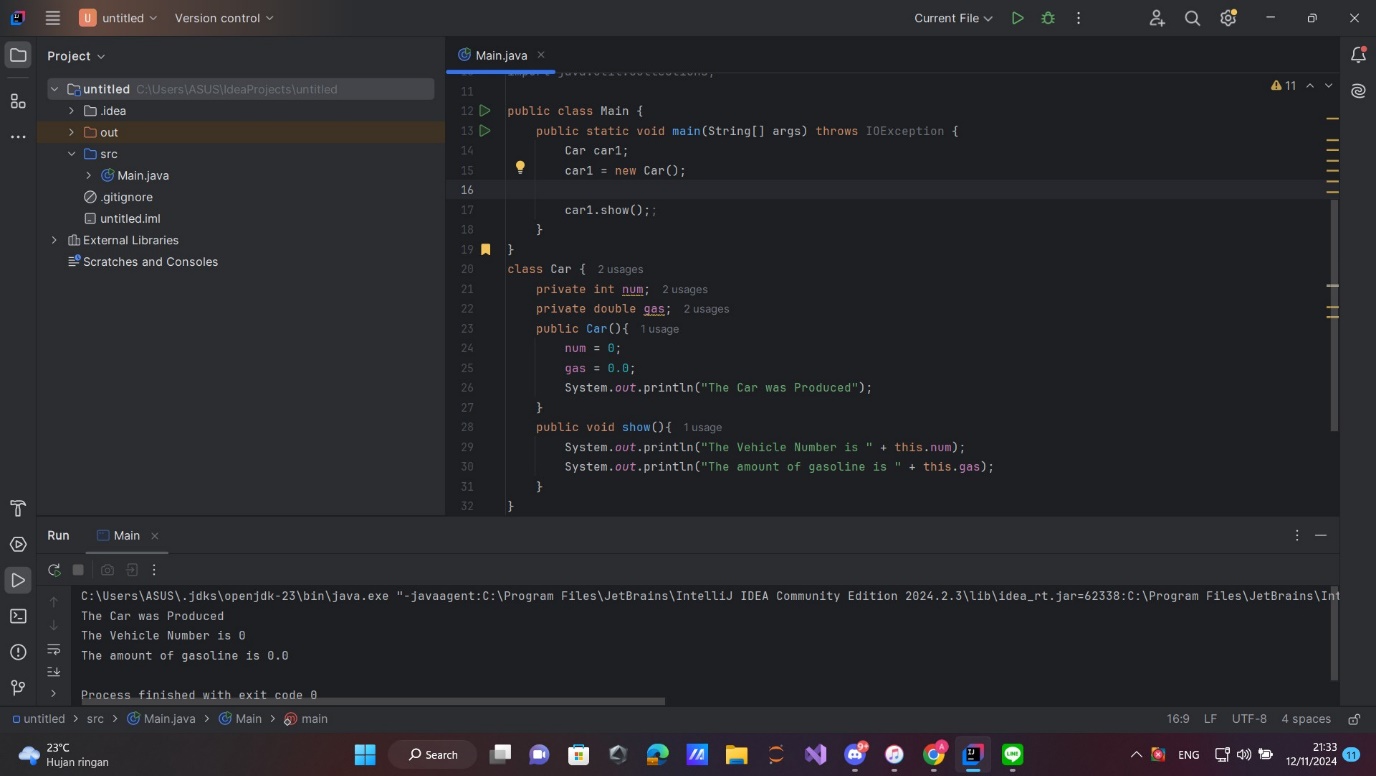


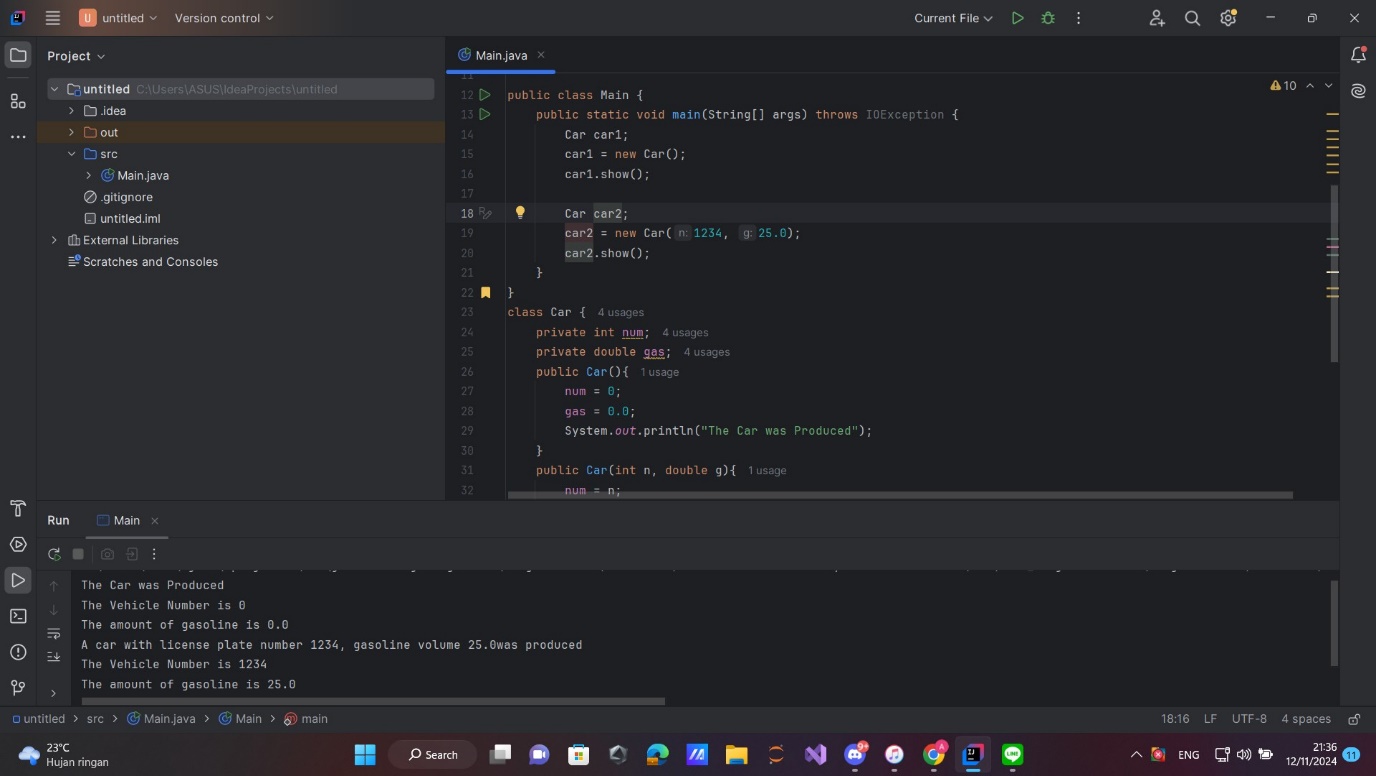


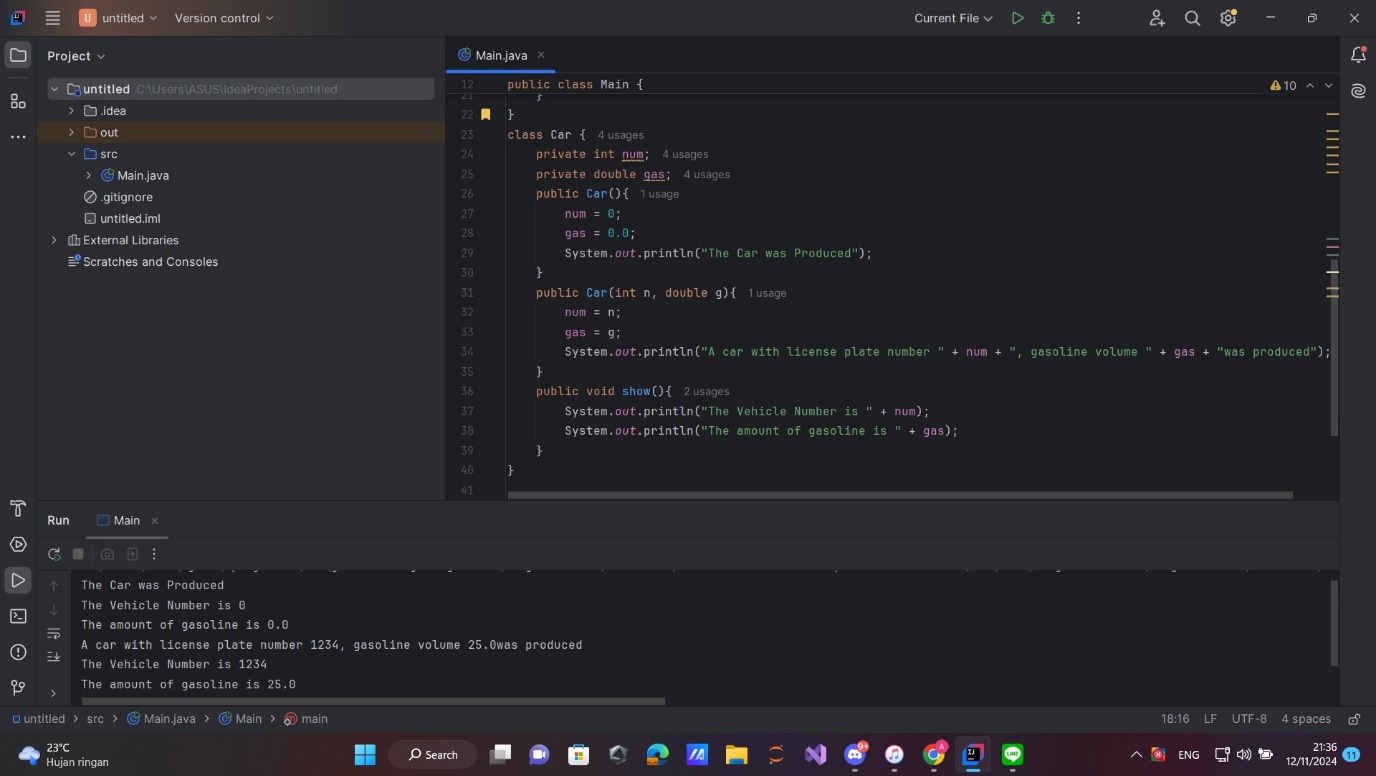


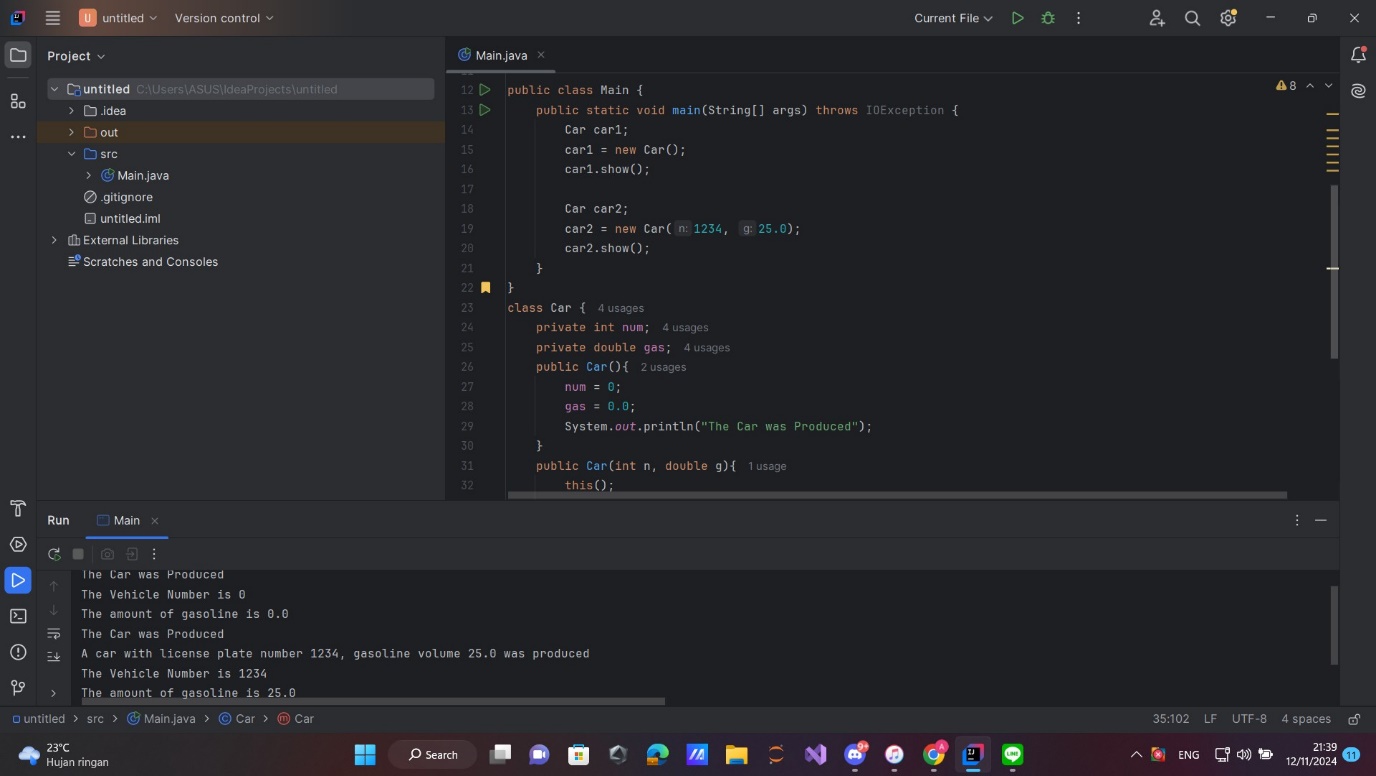


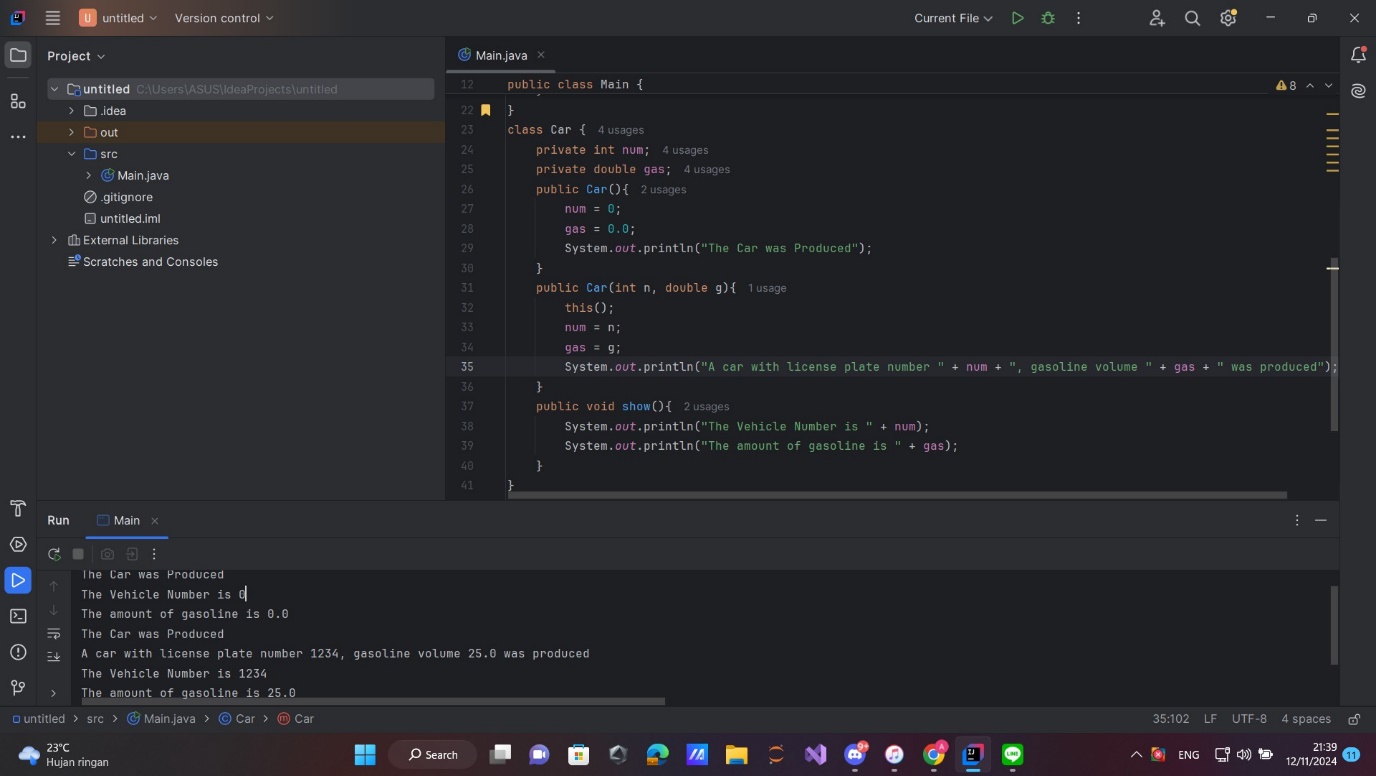


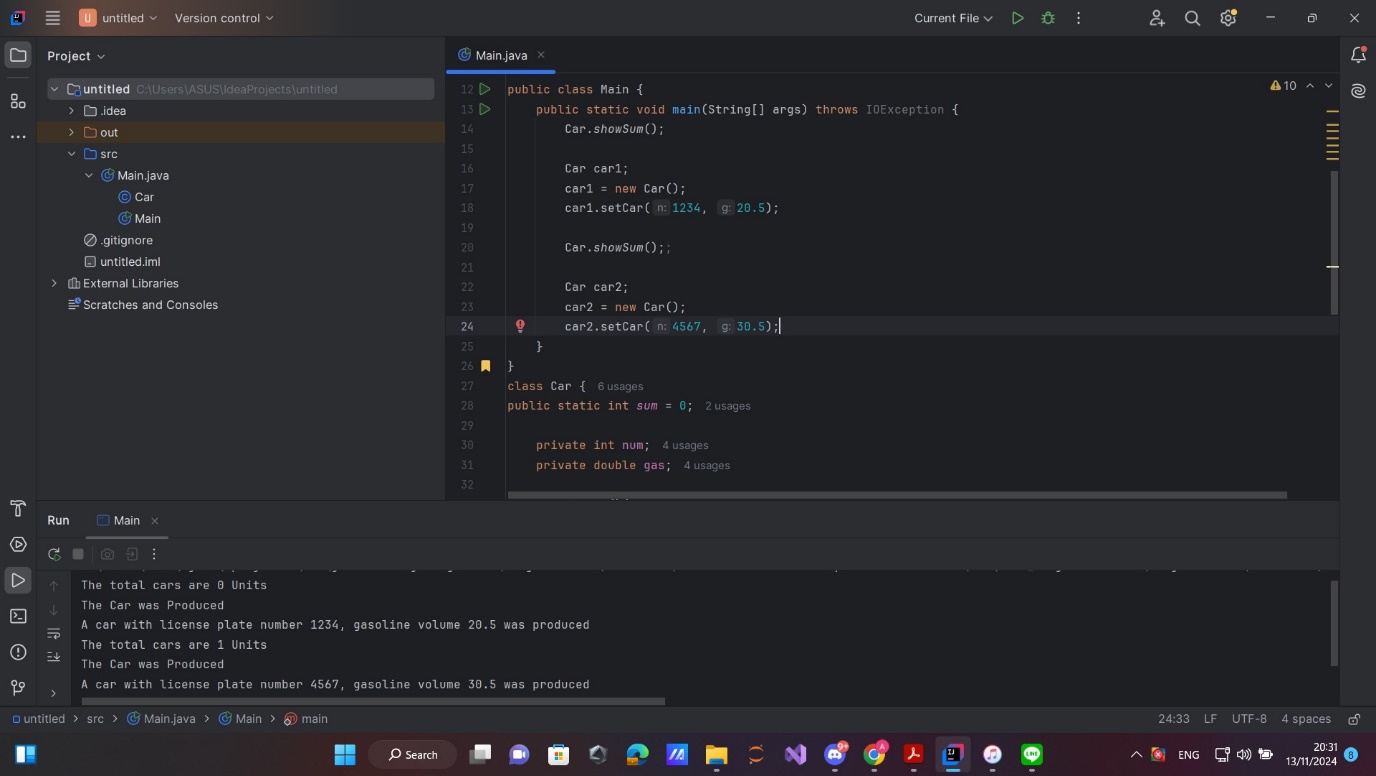


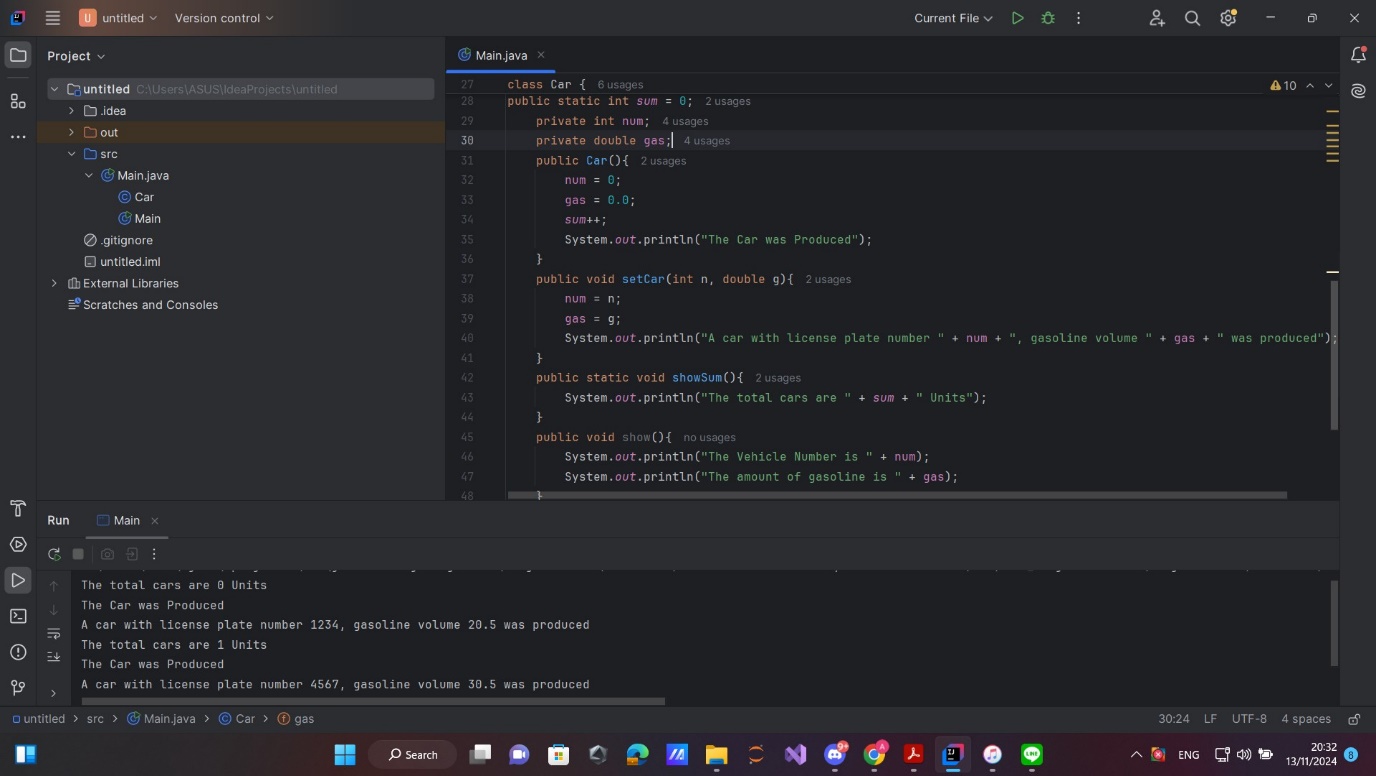


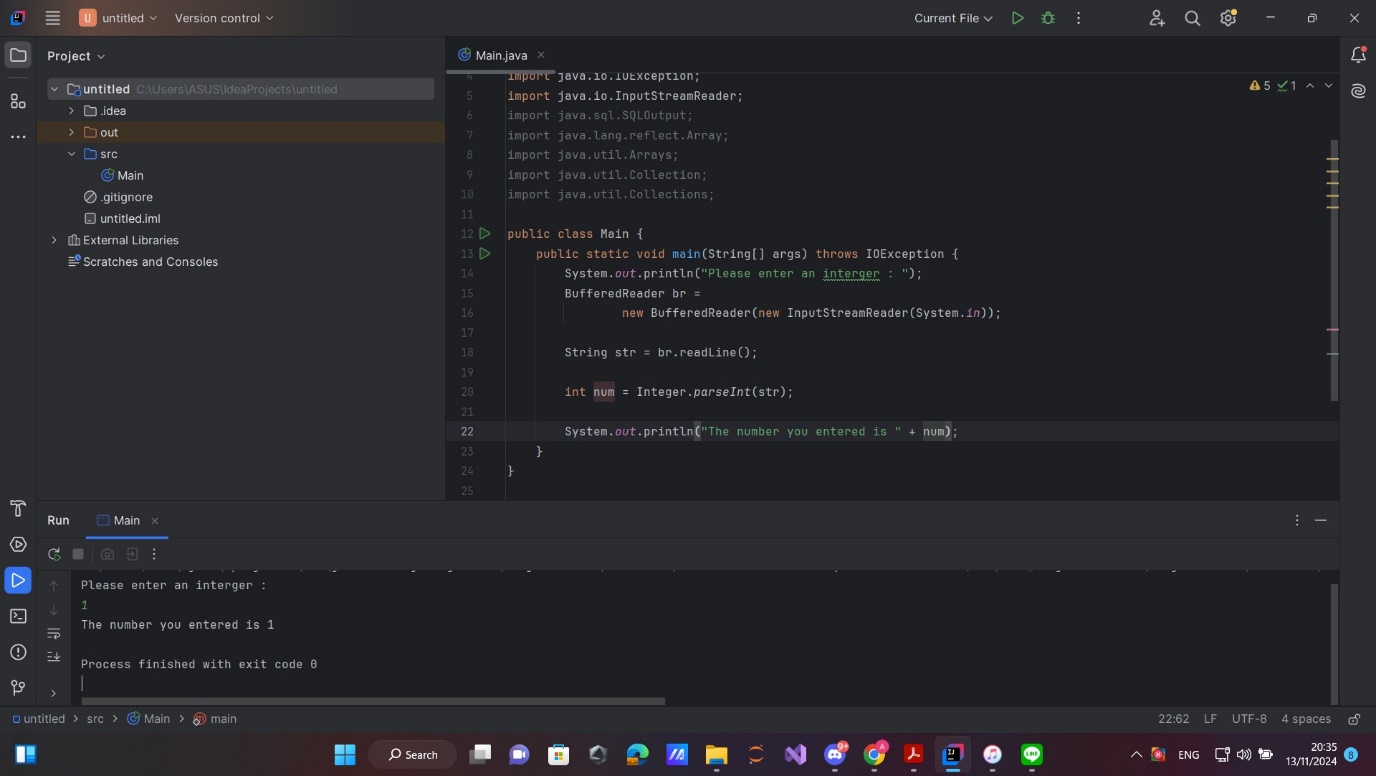


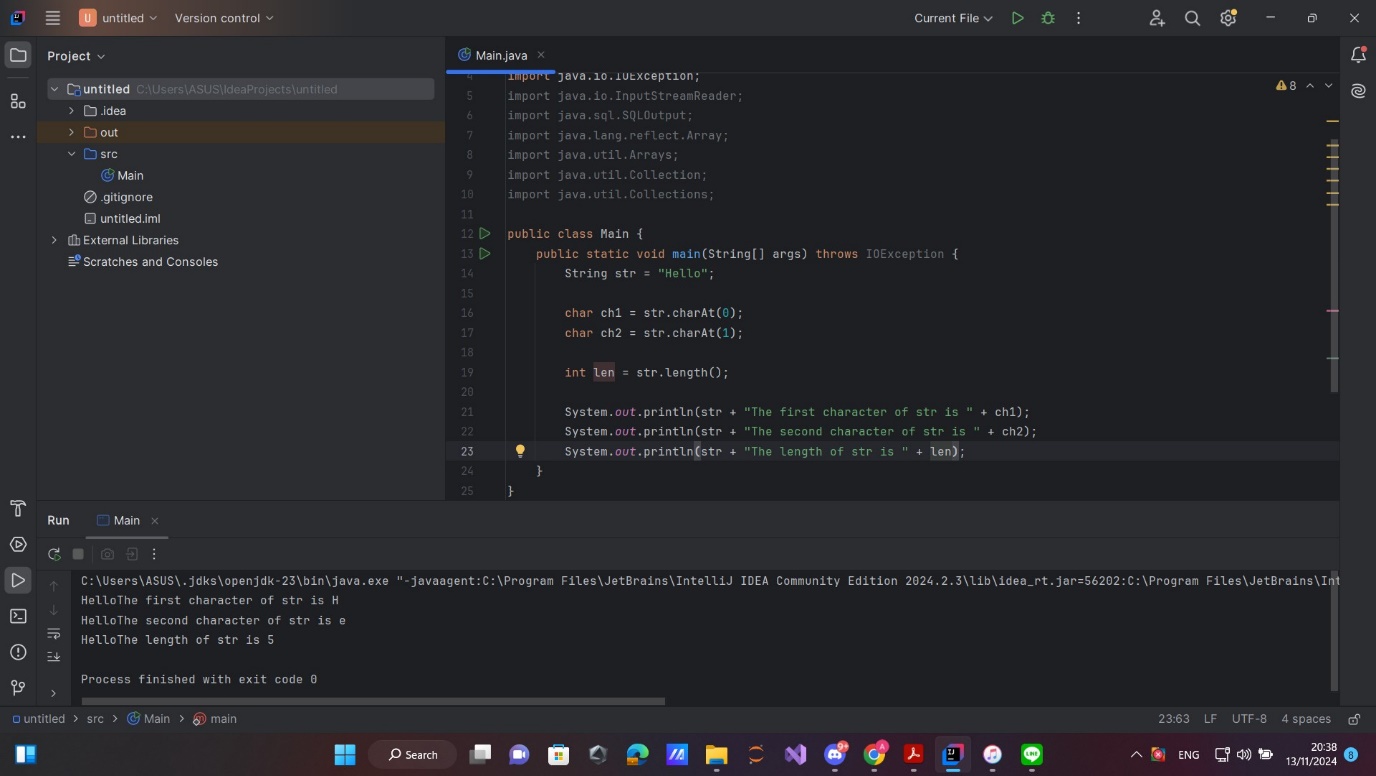


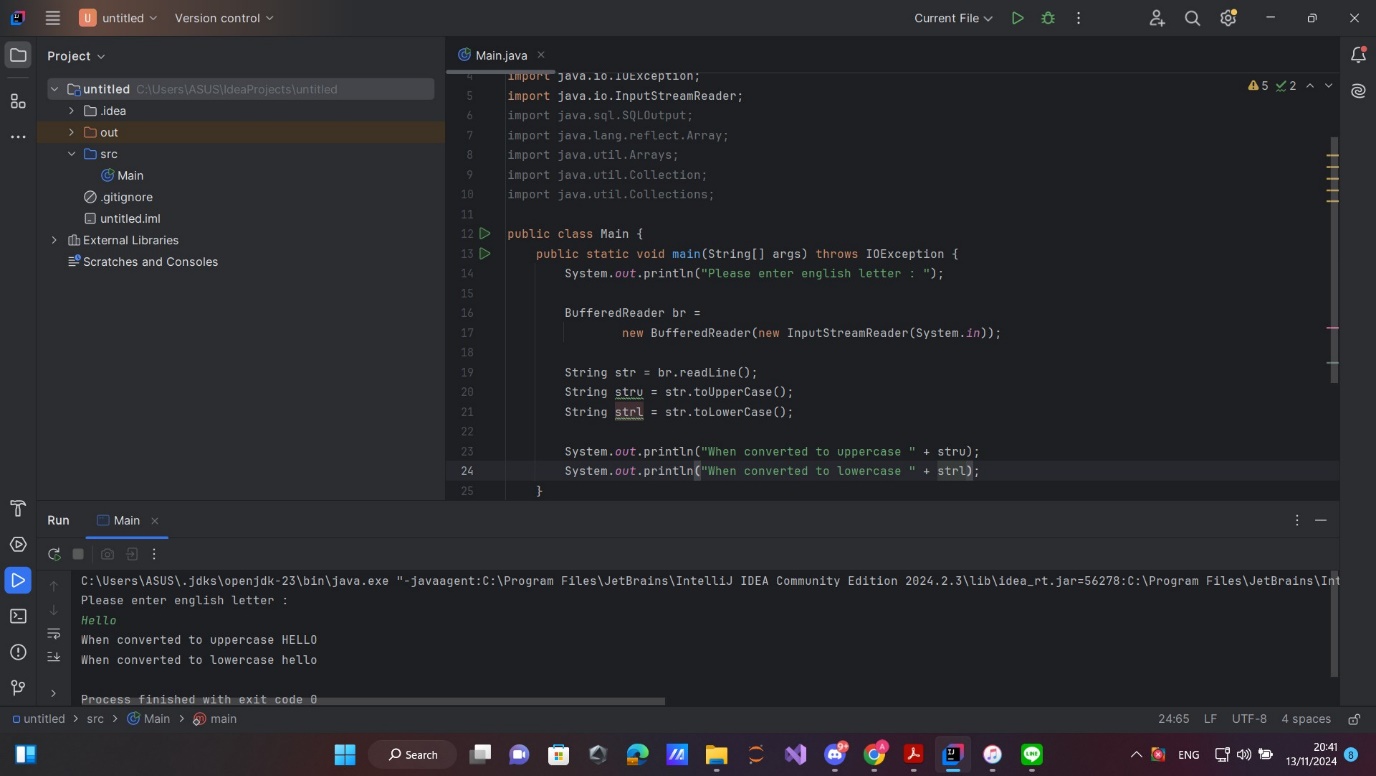


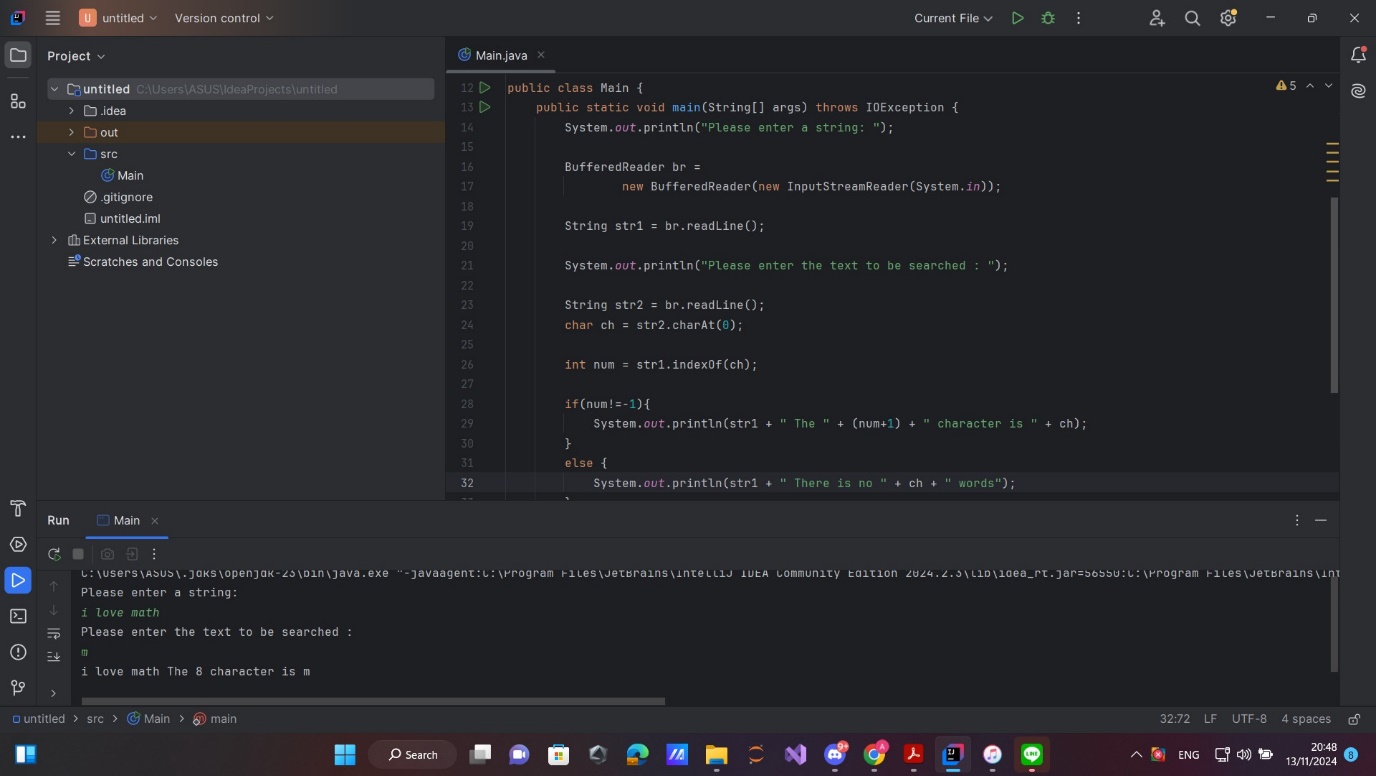


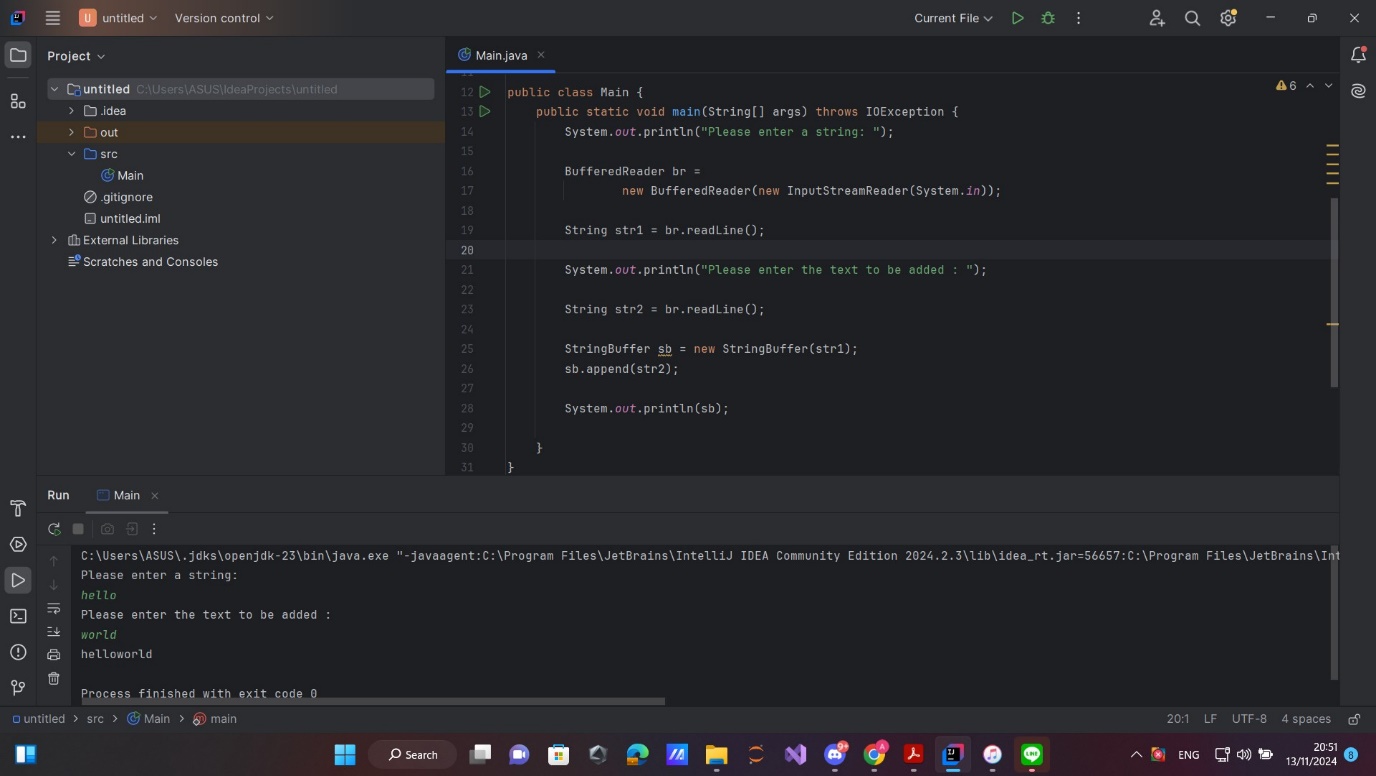


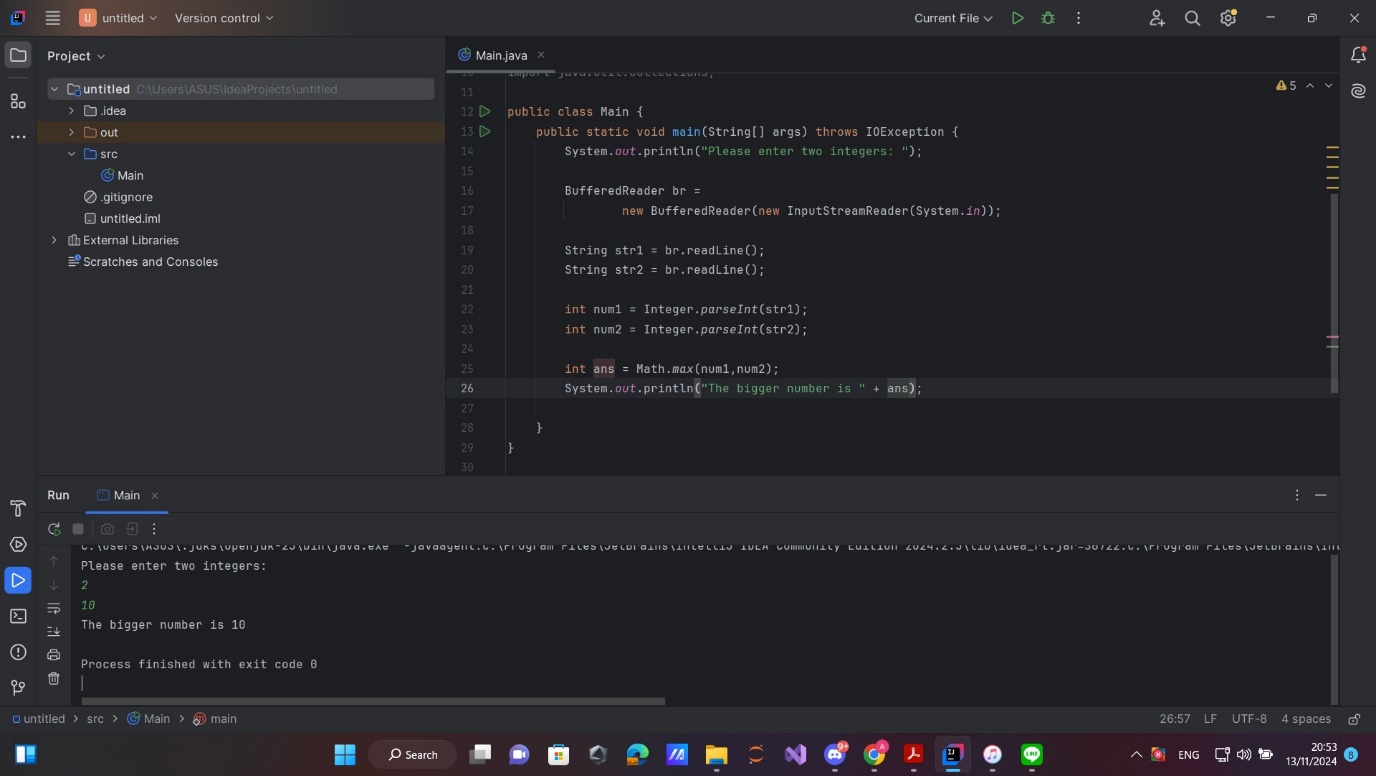


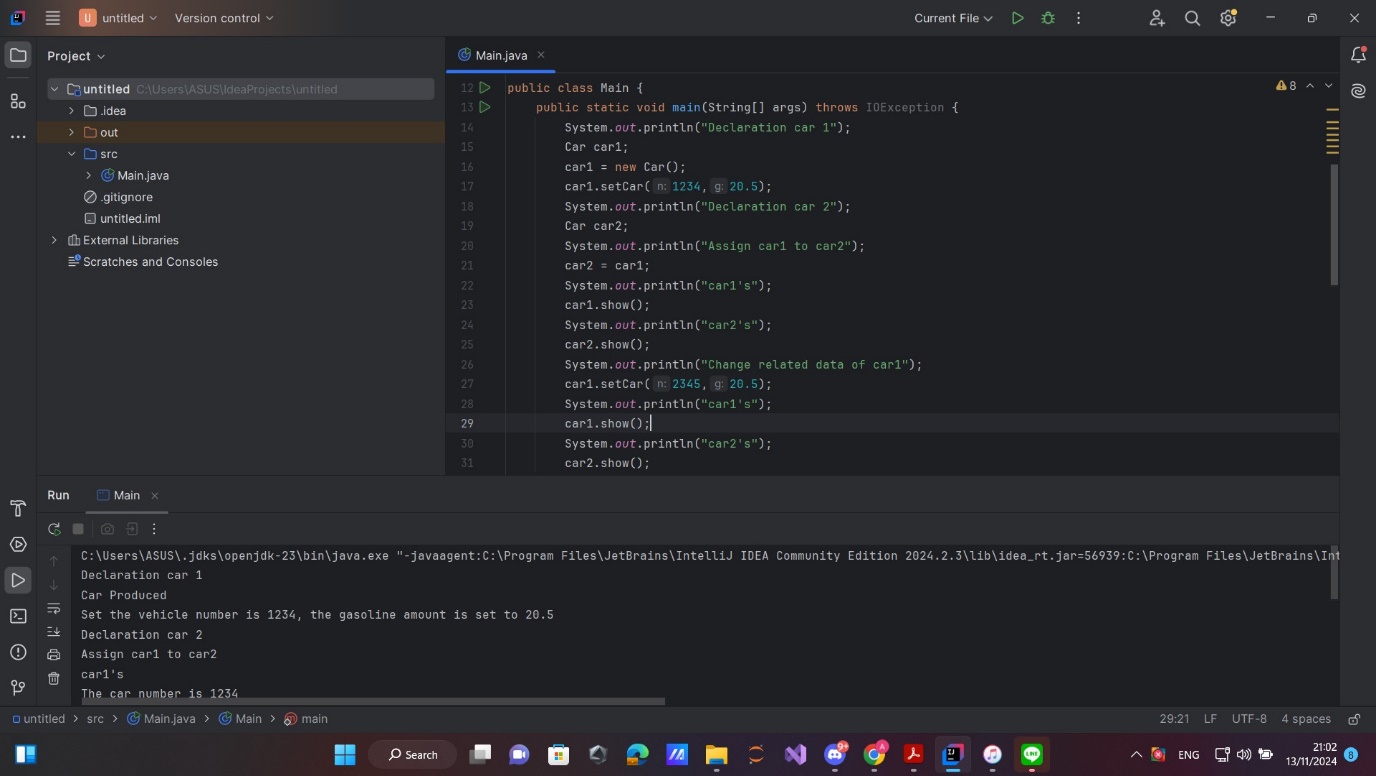


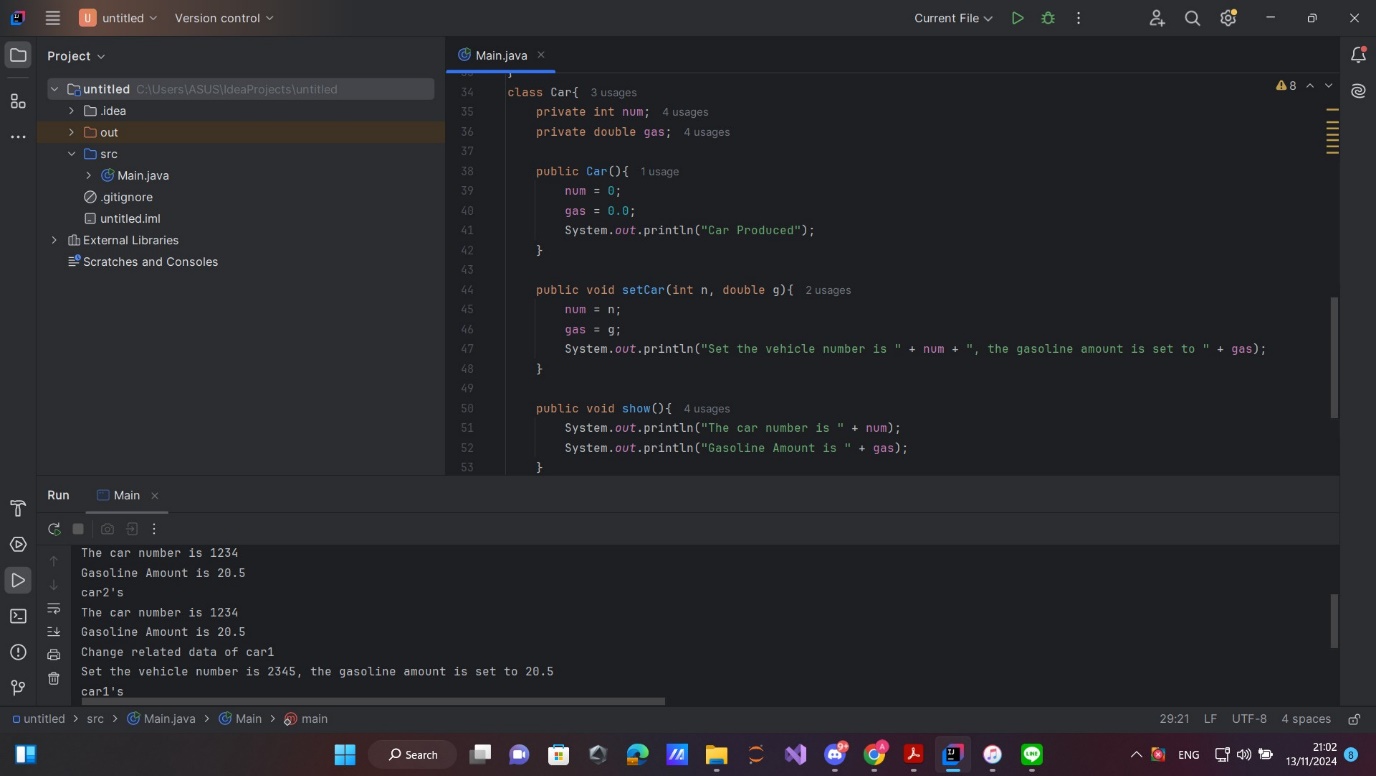




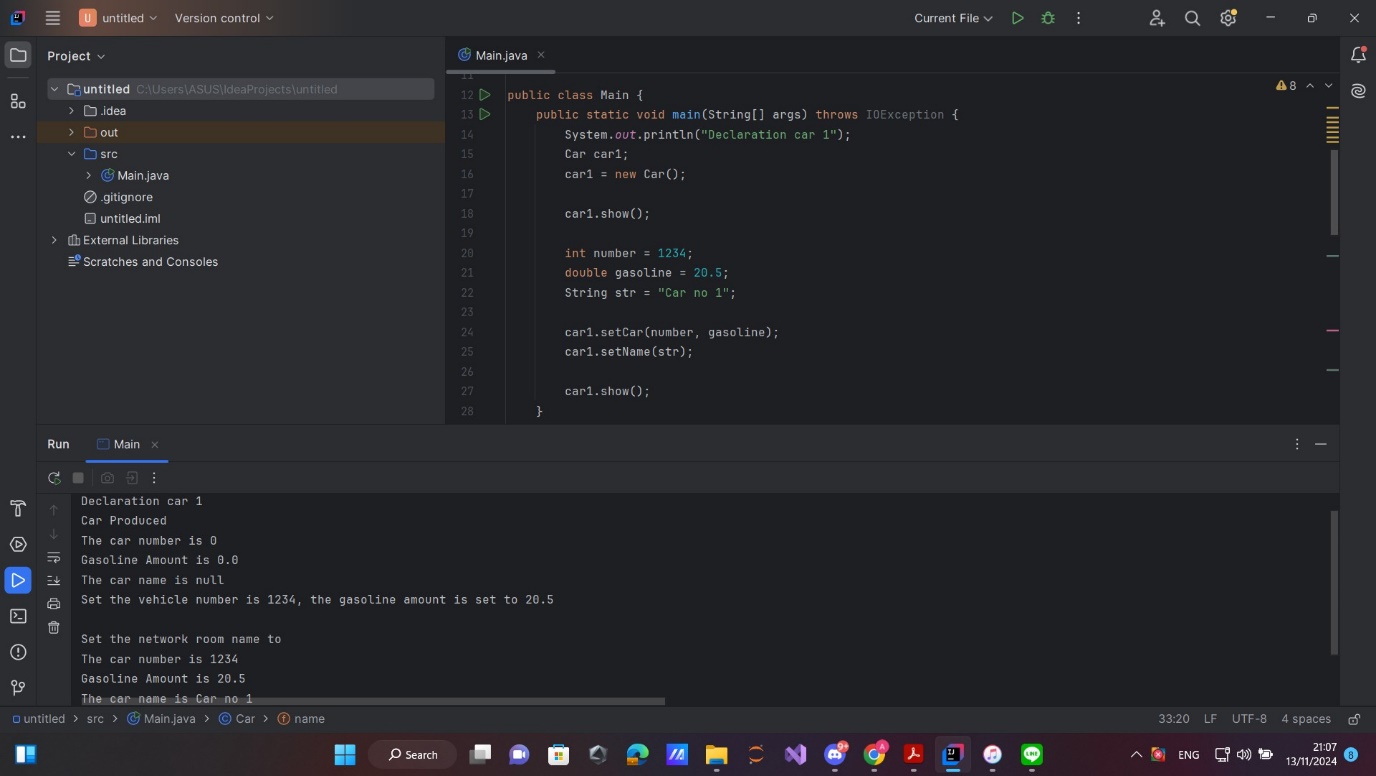


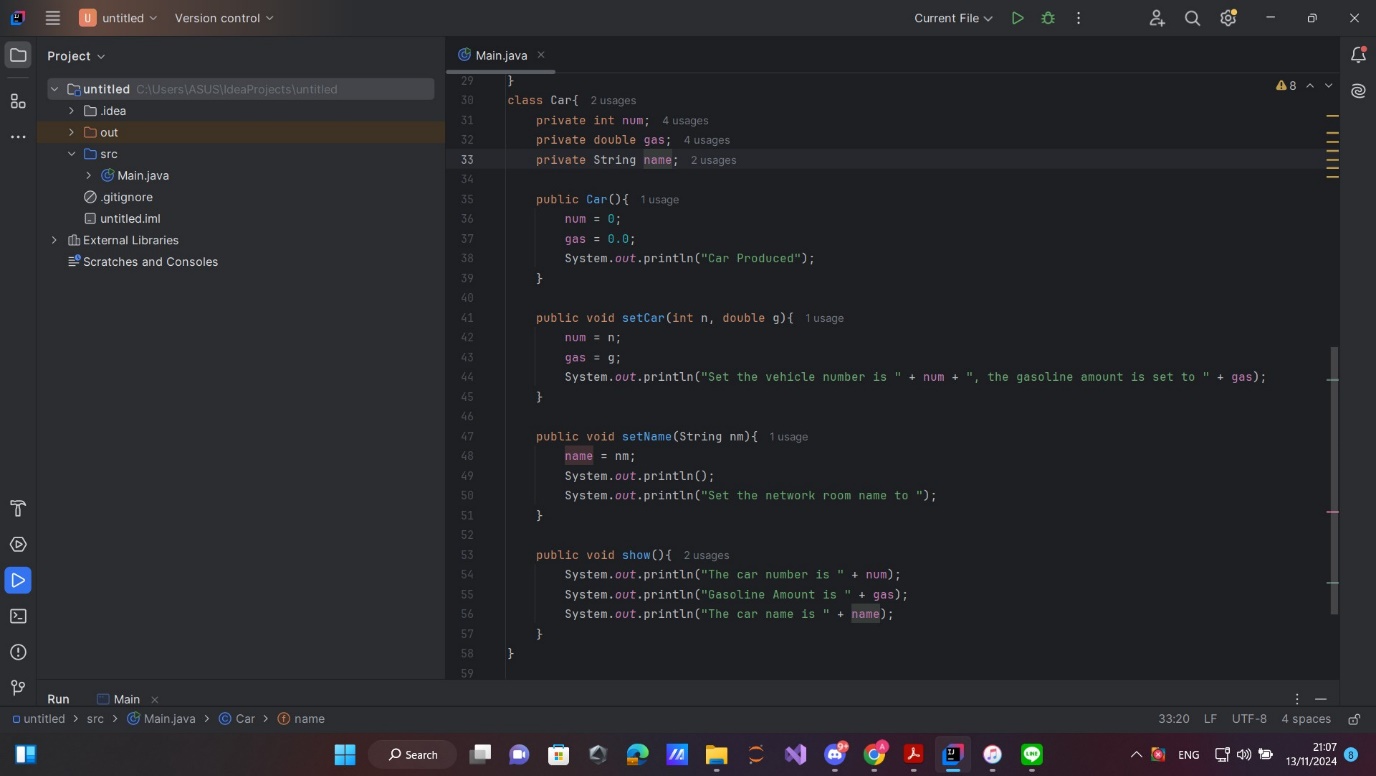


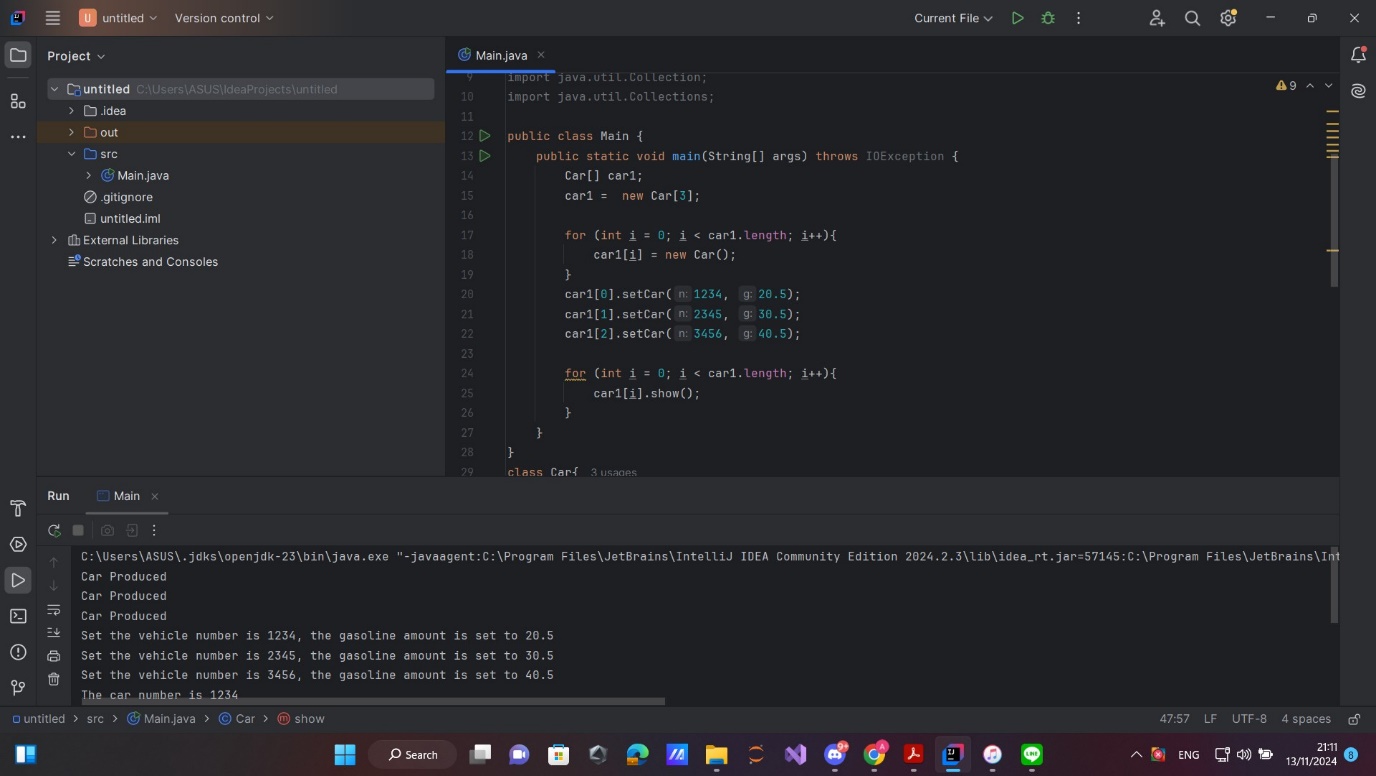


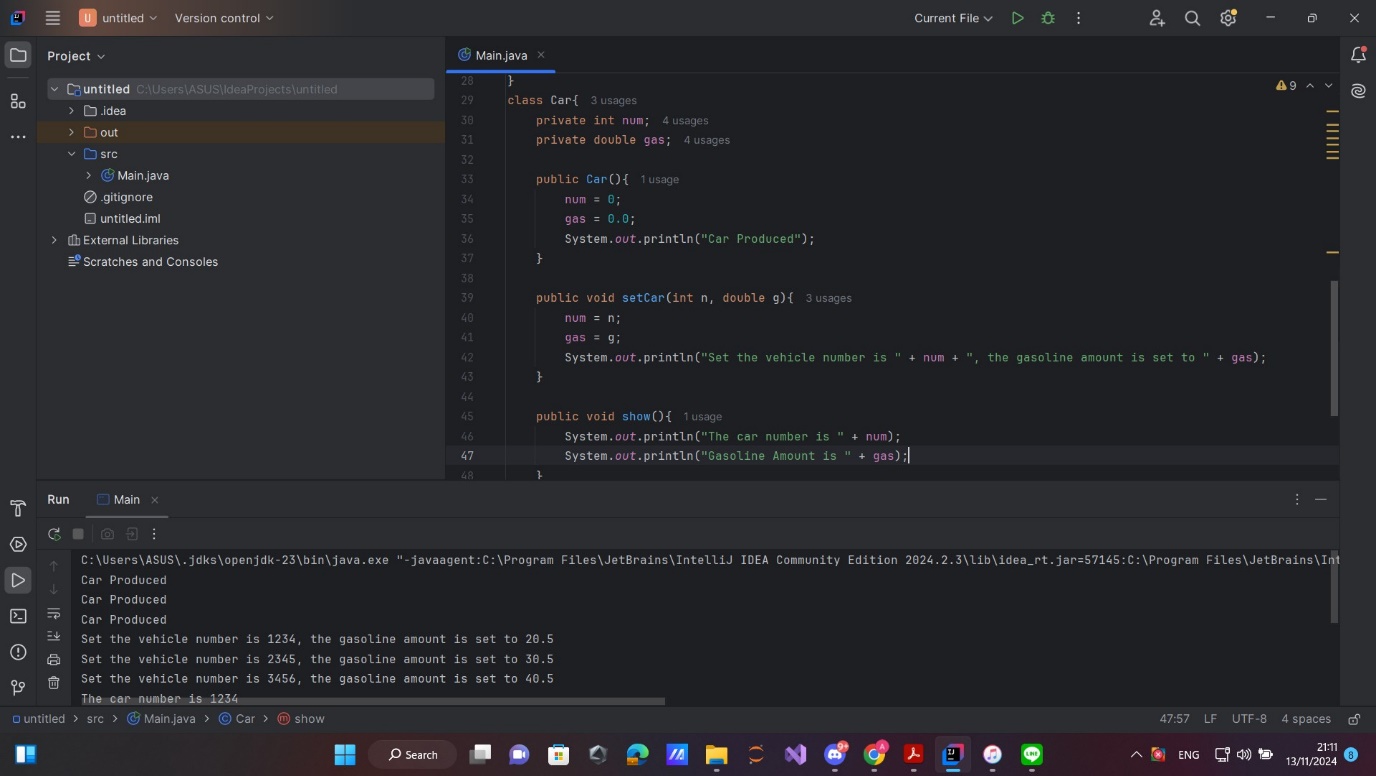


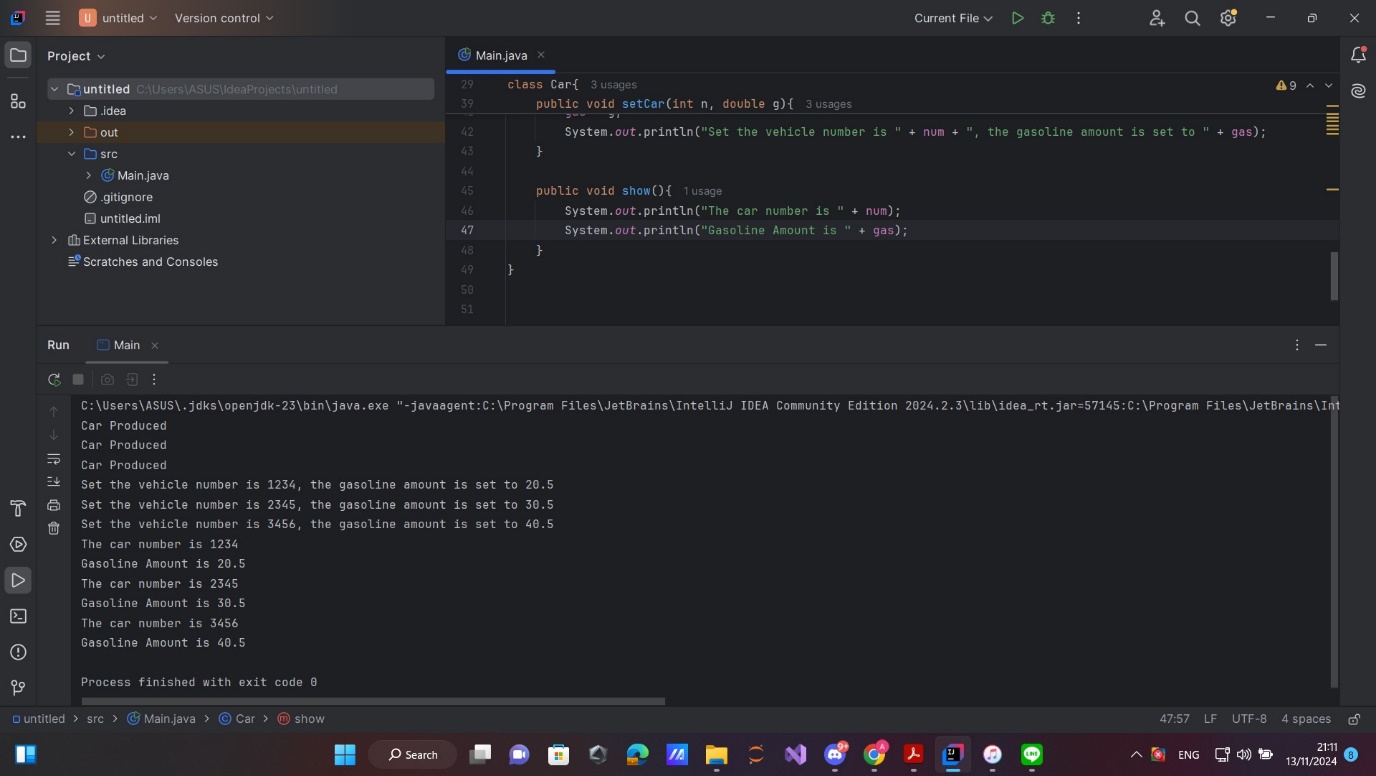












In conclusion, Chapter 3 provides foundational insights into Java classes, starting with their basic structure and role in object-oriented programming. Classes, defined by the class keyword, serve as blueprints for creating objects, which encapsulate fields (data) and methods (functionality). This chapter explains how to declare classes, create objects, and access members, covering topics like arguments, return values, and the use of this for internal field access. Methods within classes can perform operations on data, and constructors help initialize new objects, allowing multiple versions (overloading) to handle different initializations. Java also emphasizes encapsulation by using access modifiers like private, public, and protected to control access to class members, helping to prevent unintended interference with the class’s internal workings.

On the application side, the chapter explores the use of class libraries like String for text manipulation and Math for calculations. Java offers wrapper classes that let primitive types be managed as objects, enhancing flexibility in data handling. Methods can accept parameters either by value (for basic types) or by reference (for objects), allowing shared data across different methods. Additionally, object arrays enable storage and manipulation of multiple objects as a collection. The chapter highlights best practices for class functionality, such as separating instance-specific members from class-level members using the static keyword, reinforcing structured, reusable code design.

Question and answer (ChatGPT):

**1. What is the purpose of encapsulation in Java classes, and how is it achieved?**

Encapsulation in Java classes is a way to protect data by restricting access to specific parts of a class. This prevents external interference with the internal workings of the class. It is achieved by marking class members as private, making them accessible only within the class itself. Public methods, or "getters" and "setters," are then provided to allow controlled access or modification of these private members from outside the class. This practice ensures data integrity and promotes modular, maintainable code.

**2. How does Java handle method overloading within a class, and why is it useful?**

Method overloading in Java allows multiple methods in a single class to have the same name but different parameter lists. Java differentiates these methods based on the number or type of parameters passed to them. Overloading is useful because it enables methods to perform similar functions while handling different data inputs, making the code more readable and flexible. For example, a print method could be overloaded to accept either a String or an int, allowing it to print both text and numbers with the same method name.

**3. What is the difference between instance and class variables/methods, and when should each be used?**

Instance variables and methods are associated with specific objects created from a class, meaning each object has its own copy of these variables. They are used for attributes and behaviors that vary between instances. In contrast, class variables and methods (marked with static) are shared across all instances of a class, meaning they belong to the class itself rather than any individual object. Class variables are useful for storing shared information, and class methods are helpful for operations that don’t require specific object data, such as utility or helper functions.