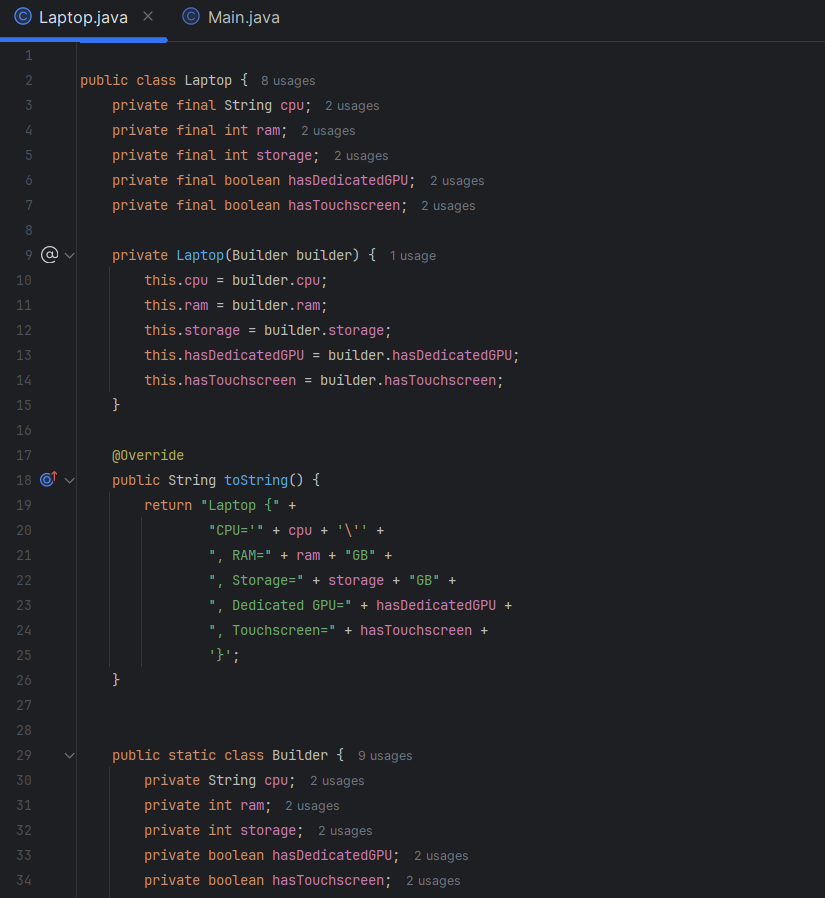
**REPORT**

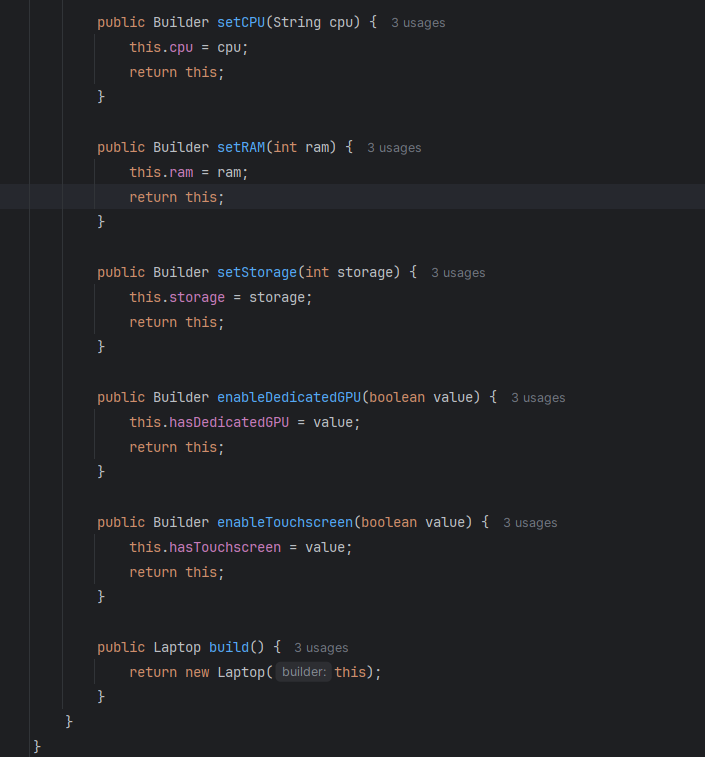
**Ayan Matayev**

**SE-2435**

In this work, I implemented a program in Java that uses the **Builder design pattern**. The main goal was to create a Laptop class with different configurations (like CPU, RAM, storage, GPU, and touchscreen) in a clean and flexible way.

The Laptop class is made private with final fields so the values cannot be changed after creation. To build an object, I created a static inner class called Builder. This builder allows setting parameters step by step using methods such as setCPU(), setRAM(), setStorage(), enableDedicatedGPU(), and enableTouchscreen(). After setting the values, the build() method creates the final Laptop object.





In the Main class, I tested the builder by creating three types of laptops:

**Gaming Laptop** – powerful CPU, 32GB RAM, 1TB storage, with a dedicated GPU but no touchscreen.

**Ultrabook** – balanced laptop with Intel i5, 16GB RAM, 512GB storage, no GPU, but with touchscreen.

**Budget Laptop** – low-cost option with AMD Ryzen 5, 8GB RAM, 256GB storage, no GPU, no touchscreen.

The program prints out the details of each laptop using the toString() method, which makes it easy to see the configuration. 

This work helped me understand how the **Builder pattern** makes code more readable and flexible, especially when creating objects with many optional parameters. It also follows clean code principles by avoiding long constructors with many arguments.