Student: Kadulbekova Asem

Creational design patterns - Builder Pattern

# 1. Introduction

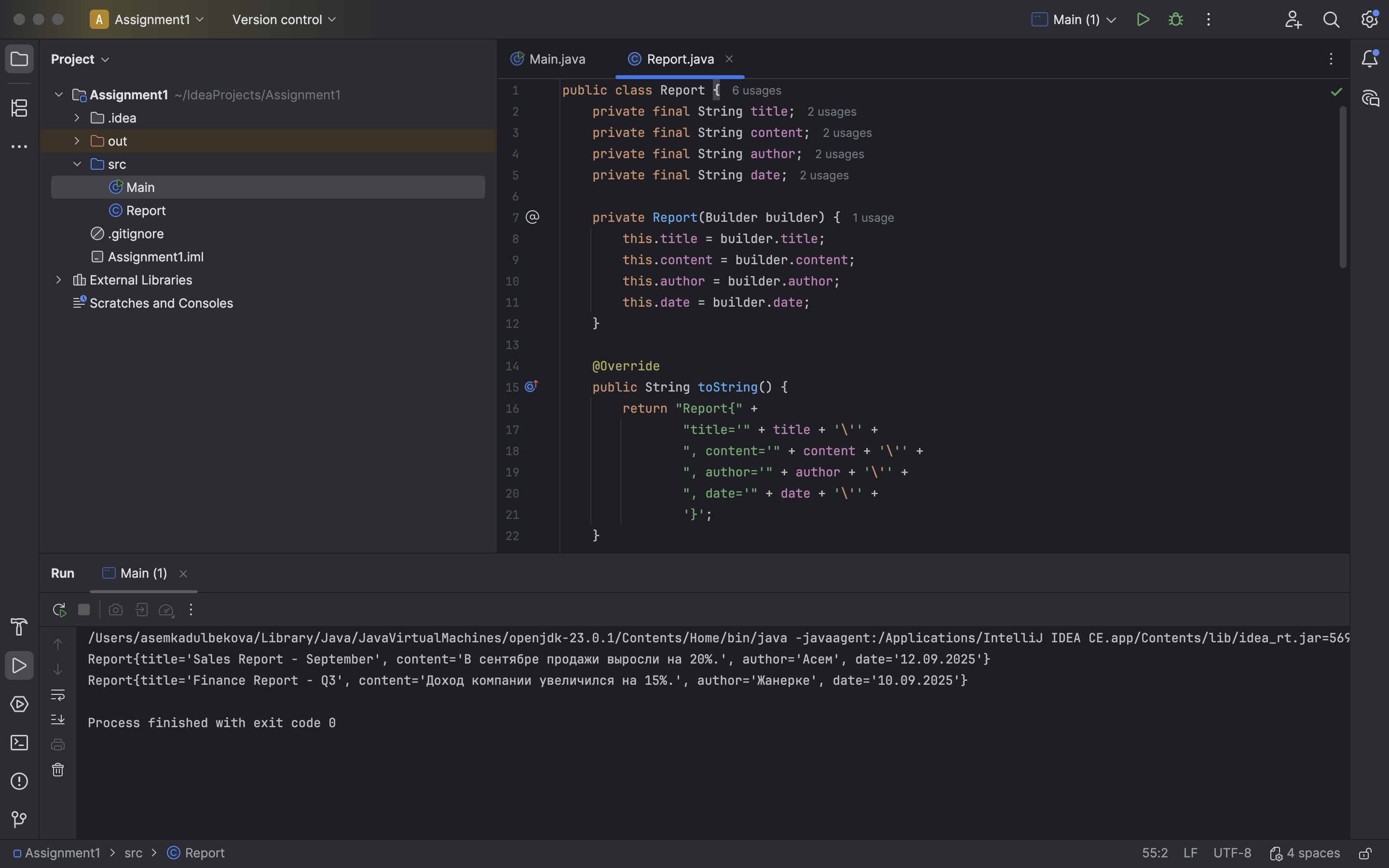
Design patterns are standardized solutions to common software development problems. They allow developers to write code that is easier to understand, maintain, and extend. In this assignment, I implemented the Builder Pattern, which separates the construction of an object from its representation. This makes it possible to build complex objects step by step in a clean and flexible way.

# 2. Motivation and Product Choice

I selected Report.Builder as the product to implement. Reports are a good example of objects with multiple optional attributes, such as a title, author, date, and content. Using constructors with many parameters would make the code difficult to read. The Builder Pattern solves this issue by allowing a clear and step-by-step object creation.

# 3. Implementation

## Report.java (Product + Builder)



## Main.java (Client)

# 4. Application of Clean Code Principles

1. Small functions – each method (setTitle, setAuthor, etc.) does one task only.  
2. Meaningful names – method names are clear and self-explanatory.  
3. Readability – the code reads like a narrative: new Report.Builder().setTitle(...).setAuthor(...).build().  
4. No duplication (DRY) – all construction logic is isolated inside the build() method.

# 5. Conclusion

The Builder Pattern allowed me to construct Report objects in a clear and flexible way. It improves readability and maintainability of the code and follows the principles of Clean Code. The only disadvantage is that it requires writing additional classes, but the benefits outweigh this cost.