Builder Pattern Implementation in Java with Clean Code Principles

# 1. Introduction

This project demonstrates the Builder Design Pattern in Java. The Builder pattern is used for step-by-step construction of complex objects. It helps avoid telescoping constructors and improves the readability and maintainability of the code.

# 2. Implementation

- The Car class is implemented as immutable (final fields).  
- A nested static class CarBuilder is provided for step-by-step object creation.  
- Mandatory fields are validated during the build() method call.  
- The toString() method is overridden to display object information in a clear format.

# 3. Clean Code Principles Applied

- Meaningful Names: all variables and methods have clear, descriptive names (e.g., brand, setAutomatic).  
- Small Functions: setter methods perform only one specific action.  
- Consistency: uniform camelCase naming convention is used throughout the code.  
- Immutability: once created, a Car object cannot be modified.

# 4. Example

Car car = new Car.CarBuilder()  
 .setBrand("Toyota")  
 .setModel("Camry")  
 .setYear(2022)  
 .setColor("Black")  
 .setAutomatic(true)  
 .build();  
  
System.out.println(car);

# 5. Conclusion

The implementation of the Builder pattern provides flexibility in object creation and improves code clarity. Applying Clean Code principles ensures better readability, maintainability, and overall software quality.