**Exercise 3: Implementing the Builder Pattern**

**Scenario:**

You are developing a system to create complex objects such as a Computer with multiple optional parts. Use the Builder Pattern to manage the construction process.

**Steps:**

1. **Create a New Java Project:**
   * Create a new Java project named **BuilderPatternExample**.
2. **Define a Product Class:**
   * Create a class **Computer** with attributes like **CPU**, **RAM**, **Storage**, etc.
3. **Implement the Builder Class:**
   * Create a static nested Builder class inside Computer with methods to set each attribute.
   * Provide a **build()** method in the Builder class that returns an instance of Computer.
4. **Implement the Builder Pattern:**
   * Ensure that the **Computer** class has a private constructor that takes the **Builder** as a parameter.
5. **Test the Builder Implementation:**
   * Create a test class to demonstrate the creation of different configurations of Computer using the Builder pattern.

**Computer Class**: The Computer class represents the product we want to create. It has various attributes like CPU, RAM, storage, etc., and a private constructor that takes a Builder instance.

**Builder Class**: The static nested Builder class has methods to set each attribute of the Computer. It also has a build() method that returns a new Computer instance.

**Testing the Builder Pattern**: In the BuilderPatternTest class, we create different configurations of the Computer using the Builder pattern and print their configurations.