**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.

Code-

package Design\_Patterns\_And\_Principles.BuilderPatternExample;

// Product class - Computer

 class Computer {

    private String cpu;

    private int ram;

    private int storage;

    // Private constructor that takes the Builder as a parameter

    private Computer(Builder builder) {

        this.cpu = builder.cpu;

        this.ram = builder.ram;

        this.storage = builder.storage;

    }

    // Builder class for Computer

    public static class Builder {

        private String cpu;

        private int ram;

        private int storage;

        public Builder() {

            // Set default values if needed

        }

        public Builder cpu(String cpu) {

            this.cpu = cpu;

            return this;

        }

        public Builder ram(int ram) {

            this.ram = ram;

            return this;

        }

        public Builder storage(int storage) {

            this.storage = storage;

            return this;

        }

        // Build method that returns an instance of Computer

        public Computer build() {

            return new Computer(this);

        }

    }

    // Getters for Computer attributes

    public String getCpu() {

        return cpu;

    }

    public int getRam() {

        return ram;

    }

    public int getStorage() {

        return storage;

    }

}

// Test class to demonstrate the Builder pattern

public class ComputerBuilderTest {

    public static void main(String[] args) {

        // Creating different configurations of Computer using the Builder pattern

        Computer computer1 = new Computer.Builder()

                .cpu("Intel i7")

                .ram(16)

                .storage(512)

                .build();

        Computer computer2 = new Computer.Builder()

                .cpu("AMD Ryzen 5")

                .ram(8)

                .storage(256)

                .build();

        // Displaying the configurations

        System.out.println("Configuration of Computer 1:");

        System.out.println("CPU: " + computer1.getCpu());

        System.out.println("RAM: " + computer1.getRam() + " GB");

        System.out.println("Storage: " + computer1.getStorage() + " GB");

        System.out.println("\nConfiguration of Computer 2:");

        System.out.println("CPU: " + computer2.getCpu());

        System.out.println("RAM: " + computer2.getRam() + " GB");

        System.out.println("Storage: " + computer2.getStorage() + " GB");

    }

}