**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.

Solution:

1st file: Computer.java

public class Computer {

private String CPU;

private String RAM;

private String storage;

private String graphicsCard;

private String keyboard;

private String mouse;

private Computer(Builder builder) {

this.CPU = builder.CPU;

this.RAM = builder.RAM;

this.storage = builder.storage;

this.graphicsCard = builder.graphicsCard;

this.keyboard = builder.keyboard;

this.mouse = builder.mouse;

}

public static class Builder {

private String CPU;

private String RAM;

private String storage;

private String graphicsCard;

private String keyboard;

private String mouse;

public Builder(String CPU, String RAM) {

this.CPU = CPU;

this.RAM = RAM;

}

public Builder setStorage(String storage) {

this.storage = storage;

return this;

}

public Builder setGraphicsCard(String graphicsCard) {

this.graphicsCard = graphicsCard;

return this;

}

public Builder setKeyboard(String keyboard) {

this.keyboard = keyboard;

return this;

}

public Builder setMouse(String mouse) {

this.mouse = mouse;

return this;

}

public Computer build() {

return new Computer(this);

}

}

@Override

public String toString() {

return "Computer [CPU=" + CPU + ", RAM=" + RAM + ", Storage=" + storage +

", GraphicsCard=" + graphicsCard + ", Keyboard=" + keyboard + ", Mouse=" + mouse + "]";

}

}

2nd file : TestBuilderPattern.java

public class TestBuilderPattern {

public static void main(String[] args) {

Computer basicComputer = new Computer.Builder("Intel i3", "8GB").build();

Computer gamingComputer = new Computer.Builder("Intel i9", "32GB")

.setStorage("1TB SSD")

.setGraphicsCard("NVIDIA RTX 4080")

.setKeyboard("Mechanical")

.setMouse("Gaming")

.build();

System.out.println("Basic Computer: " + basicComputer);

System.out.println("Gaming Computer: " + gamingComputer);

}

}

Output:

