**DESIGN PATTERNS AND PRINCIPLES**

**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.

**DESCRIPTION:**

This program demonstrates the Builder Design Pattern using a Computer class, where objects can be created with a flexible and readable configuration. The Builder pattern is especially useful for constructing complex objects with many optional attributes.

**PROGRAM:**

public class BuilderPatternExample {

public static void main(String[] args) {

Computer basicComputer = new Computer.Builder()

.setCPU("Intel i3")

.setRAM("8GB")

.setStorage("256GB SSD")

.build();

System.out.println(basicComputer);

System.out.println("---------------------------");

Computer gamingComputer = new Computer.Builder()

.setCPU("AMD Ryzen 9")

.setRAM("32GB")

.setStorage("1TB NVMe SSD")

.setGraphicsCard("NVIDIA RTX 4080")

.setOperatingSystem("Windows 11")

.build();

System.out.println(gamingComputer);

}

}

class Computer {

private String CPU;

private String RAM;

private String storage;

private String graphicsCard;

private String operatingSystem;

private Computer(Builder builder) {

this.CPU = builder.CPU;

this.RAM = builder.RAM;

this.storage = builder.storage;

this.graphicsCard = builder.graphicsCard;

this.operatingSystem = builder.operatingSystem;

}

public static class Builder {

private String CPU;

private String RAM;

private String storage;

private String graphicsCard;

private String operatingSystem;

public Builder setCPU(String CPU) {

this.CPU = CPU;

return this;

}

public Builder setRAM(String RAM) {

this.RAM = RAM;

return this;

}

public Builder setStorage(String storage) {

this.storage = storage;

return this;

}

public Builder setGraphicsCard(String graphicsCard) {

this.graphicsCard = graphicsCard;

return this;

}

public Builder setOperatingSystem(String operatingSystem) {

this.operatingSystem = operatingSystem;

return this;

}

public Computer build() {

return new Computer(this);

}

}

@Override

public String toString() {

return "Computer Configuration:\n" +

"CPU: " + CPU + "\n" +

"RAM: " + RAM + "\n" +

"Storage: " + storage + "\n" +

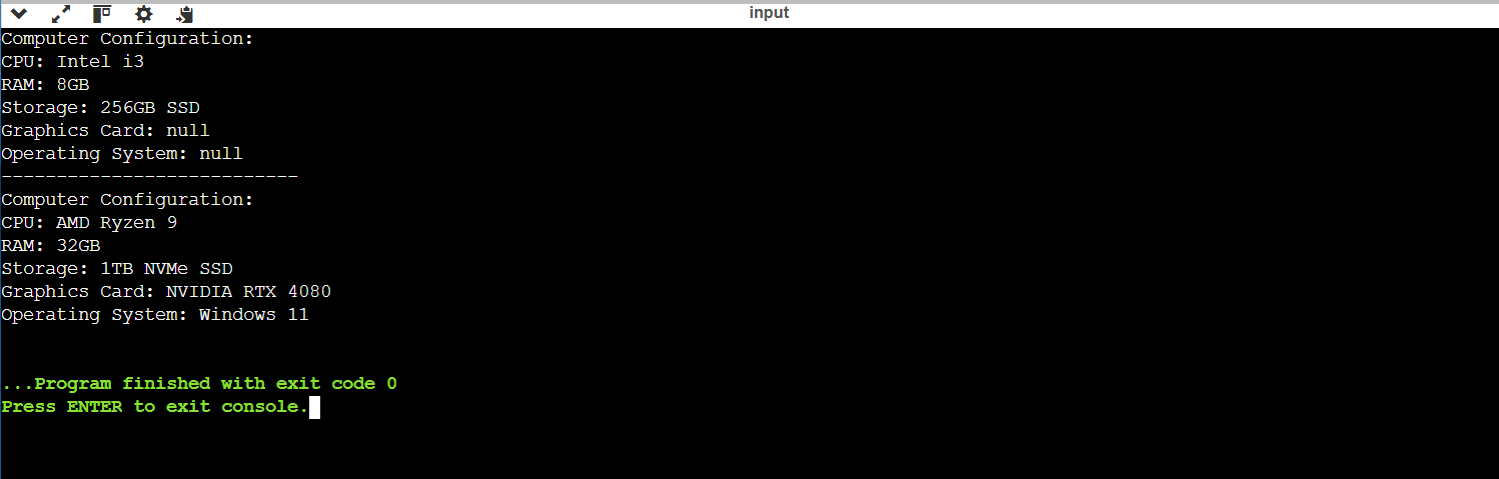
"Graphics Card: " + graphicsCard + "\n" +

"Operating System: " + operatingSystem;

}

}

**OUTPUT:**

****