# Exercise 3: Implementing the Builder Pattern

### Step 1: Create a New Java Project

Using an IDE like IntelliJ IDEA or Eclipse, create a new Java project named BuilderPatternExample.

### Step 2: Define a Product Class

Create a class Computer with attributes like CPU, RAM, Storage, etc.

// Computer.java

public class Computer {

// Attributes

private String CPU;

private String RAM;

private String storage;

private String graphicsCard;

private boolean isBluetoothEnabled;

private boolean isWiFiEnabled;

// Private constructor

private Computer(Builder builder) {

this.CPU = builder.CPU;

this.RAM = builder.RAM;

this.storage = builder.storage;

this.graphicsCard = builder.graphicsCard;

this.isBluetoothEnabled = builder.isBluetoothEnabled;

this.isWiFiEnabled = builder.isWiFiEnabled;

}

// Getters

public String getCPU() {

return CPU;

}

public String getRAM() {

return RAM;

}

public String getStorage() {

return storage;

}

public String getGraphicsCard() {

return graphicsCard;

}

public boolean isBluetoothEnabled() {

return isBluetoothEnabled;

}

public boolean isWiFiEnabled() {

return isWiFiEnabled;

}

// Builder Class

public static class Builder {

private String CPU;

private String RAM;

private String storage;

private String graphicsCard;

private boolean isBluetoothEnabled;

private boolean isWiFiEnabled;

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 setBluetoothEnabled(boolean isBluetoothEnabled) {

this.isBluetoothEnabled = isBluetoothEnabled;

return this;

}

public Builder setWiFiEnabled(boolean isWiFiEnabled) {

this.isWiFiEnabled = isWiFiEnabled;

return this;

}

public Computer build() {

return new Computer(this);

}

}

}

### Step 3: Implement the Builder Class

The Builder class is implemented as a static nested class inside the Computer class. It has methods to set each attribute and a build() method to create an instance of Computer.

### Step 4: Ensure Private Constructor

The Computer class has a private constructor that takes the Builder as a parameter. This ensures that the Computer objects can only be created through the Builder.

### Step 5: Test the Builder Implementation

Create a test class to demonstrate the creation of different configurations of Computer using the Builder pattern.

// Main.java

public class Main {

public static void main(String[] args) {

// Creating different configurations of Computer

Computer gamingComputer = new Computer.Builder()

.setCPU("Intel i9")

.setRAM("32GB")

.setStorage("1TB SSD")

.setGraphicsCard("NVIDIA RTX 3080")

.setBluetoothEnabled(true)

.setWiFiEnabled(true)

.build();

Computer officeComputer = new Computer.Builder()

.setCPU("Intel i5")

.setRAM("16GB")

.setStorage("512GB SSD")

.setBluetoothEnabled(false)

.setWiFiEnabled(true)

.build();

// Display the configurations

System.out.println("Gaming Computer Configuration:");

System.out.println("CPU: " + gamingComputer.getCPU());

System.out.println("RAM: " + gamingComputer.getRAM());

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

System.out.println("Graphics Card: " + gamingComputer.getGraphicsCard());

System.out.println("Bluetooth Enabled: " + gamingComputer.isBluetoothEnabled());

System.out.println("WiFi Enabled: " + gamingComputer.isWiFiEnabled());

System.out.println("\nOffice Computer Configuration:");

System.out.println("CPU: " + officeComputer.getCPU());

System.out.println("RAM: " + officeComputer.getRAM());

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

System.out.println("Bluetooth Enabled: " + officeComputer.isBluetoothEnabled());

System.out.println("WiFi Enabled: " + officeComputer.isWiFiEnabled());

}

}

This completes the implementation of the Builder Pattern for the Computer class. The Main class demonstrates how to create different configurations of Computer using the Builder pattern.