**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.

# THE CODE:

public class Computer

{

public string CPU { get; }

public string RAM { get; }

public string Storage { get; }

public string GraphicsCard { get; }

public string Motherboard { get; }

public string PowerSupply { get; }

private Computer(Builder builder)

{

CPU = builder.CPU;

RAM = builder.RAM;

Storage = builder.Storage;

GraphicsCard = builder.GraphicsCard;

Motherboard = builder.Motherboard;

PowerSupply = builder.PowerSupply;

}

public class Builder

{

public string CPU { get; }

public string RAM { get; }

public string Storage { get; private set; }

public string GraphicsCard { get; private set; }

public string Motherboard { get; private set; }

public string PowerSupply { get; private set; }

public Builder(string cpu, string ram)

{

CPU = cpu;

RAM = ram;

}

public Builder SetStorage(string storage)

{

Storage = storage;

return this;

}

public Builder SetGraphicsCard(string graphicsCard)

{

GraphicsCard = graphicsCard;

return this;

}

public Builder SetMotherboard(string motherboard)

{

Motherboard = motherboard;

return this;

}

public Builder SetPowerSupply(string powerSupply)

{

PowerSupply = powerSupply;

return this;

}

public Computer Build()

{

return new Computer(this);

}

}

public void ShowSpecs()

{

Console.WriteLine("Computer Configuration:");

Console.WriteLine($"CPU: {CPU}");

Console.WriteLine($"RAM: {RAM}");

Console.WriteLine($"Storage: {Storage ?? "Not specified"}");

Console.WriteLine($"Graphics Card: {GraphicsCard ?? "Not specified"}");

Console.WriteLine($"Motherboard: {Motherboard ?? "Not specified"}");

Console.WriteLine($"Power Supply: {PowerSupply ?? "Not specified"}");

}

}

class Program

{

static void Main(string[] args)

{

var basicComputer = new Computer.Builder("Intel i5", "8GB")

.Build();

var gamingComputer = new Computer.Builder("AMD Ryzen 9", "32GB")

.SetStorage("2TB SSD")

.SetGraphicsCard("NVIDIA RTX 4080")

.SetMotherboard("ASUS ROG Strix")

.SetPowerSupply("850W Gold")

.Build();

var workstation = new Computer.Builder("Intel Xeon", "64GB")

.SetStorage("1TB HDD")

.SetPowerSupply("750W Bronze")

.Build();

Console.WriteLine("=== Basic Computer ===");

basicComputer.ShowSpecs();

Console.WriteLine("\n=== Gaming Computer ===");

gamingComputer.ShowSpecs();

Console.WriteLine("\n=== Workstation Computer ===");

workstation.ShowSpecs();

}

}

# THE OUPUT:

