**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.

package WithBuilder;

public class Computer {

private final String HDD;

private final String RAM;

private final String graphicsCard;

private final String bluetooth;

private Computer(String HDD, String RAM, String graphicsCard, String bluetooth) {

this.HDD = HDD;

this.RAM = RAM;

this.graphicsCard = graphicsCard;

this.bluetooth = bluetooth;

}

@Override

public String toString() {

return "Computer [CPU=" + HDD + ", RAM=" + RAM +

", GraphicsCardEnabled=" + graphicsCard +

", BluetoothEnabled=" + bluetooth + "]";

}

public static class ComputerBuilder {

private final String HDD;

private final String RAM;

private String graphicsCard;

private String bluetooth;

public ComputerBuilder(String HDD, String RAM) {

this.HDD = HDD;

this.RAM = RAM; }

public ComputerBuilder enableGraphicsCard(String graphicsCard) {

this.graphicsCard = graphicsCard;

return this;

}

public ComputerBuilder enableBluetooth(String bluetooth) {

this.bluetooth = bluetooth;

return this;

}

public Computer build() {

return new Computer(HDD, RAM, graphicsCard, bluetooth);

}

}

}

class TestBuilderPattern {

public static void main(String[] args) {

Computer gamingComputer = new Computer.ComputerBuilder("1 TB", "16 GB")

.enableGraphicsCard("ROG")

.enableBluetooth("SONY")

.build();

System.out.println(gamingComputer);

Computer basicComputer = new Computer.ComputerBuilder("500 GB", "8 GB")

.enableBluetooth("SONY")

.build();

System.out.println(basicComputer);

}}

OUTPUT:

