# Exercise 3: Implementing the Builder Pattern

Computer.java:

package com.example;

public class Computer {

private final String cpu;

private final String ram;

private final String storage;

private final String gpu;

private Computer(ComputerBuilder builder) {

this.cpu = builder.cpu;

this.ram = builder.ram;

this.storage = builder.storage;

this.gpu = builder.gpu;

}

public String getCpu() {

return cpu;

}

public String getRam() {

return ram;

}

public String getStorage() {

return storage;

}

public String getGpu() {

return gpu;

}

*@Override*

public String toString() {

return "Computer [CPU=" + cpu + ", RAM=" + ram +

(storage != null ? ", Storage=" + storage : "") +

(gpu != null ? ", GPU=" + gpu : "") +"]";

}

// Builder Class

public static class ComputerBuilder {

private final String cpu;

private final String ram;

private String storage = null;

private String gpu = null;

public ComputerBuilder(String cpu, String ram, String storage) {

this.cpu = cpu;

this.ram = ram;

this.storage = storage;

}

public ComputerBuilder setGpu(String gpu) {

this.gpu = gpu;

return this;

}

public Computer build() {

return new Computer(this);

}

}

}

ComputerBuilderTest.java:

package com.example;

public class ComputerBuilderTest {

public static void main(String[] args) {

// Basic computer with just CPU and RAM

Computer basicComputer = new Computer.ComputerBuilder("Intel i5", "8GB DDR4","512GB SSD")

.build();

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

// Gaming computer with all features

Computer gamingComputer = new Computer.ComputerBuilder("AMD Ryzen 9", "32GB DDR4","1TB NVMe SSD")

.setGpu("NVIDIA RTX 3080")

.build();

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

// Office computer with some optional features

Computer officeComputer = new Computer.ComputerBuilder("Intel i7", "16GB DDR4","512GB SSD")

.build();

System.***out***.println("Office Computer: " + officeComputer);

}

}

Output:

