	Course Name: Design Patterns/Thinking LAB	**EXPERIMENT NO. 3**	
	Course Code: 20CP210P **Faculty: Dr. Ketan Sabale**	**Branch: CSE**	**Semester: IV**
Submitted by: Rhythm Shah **Roll no:** 22BCP071			

Objective: To familiarize students with standard Creational design patterns.
Experiment: Explain the builder design pattern and write a program using any object-oriented programming language to demonstrate the working of builder design pattern.

Theory:

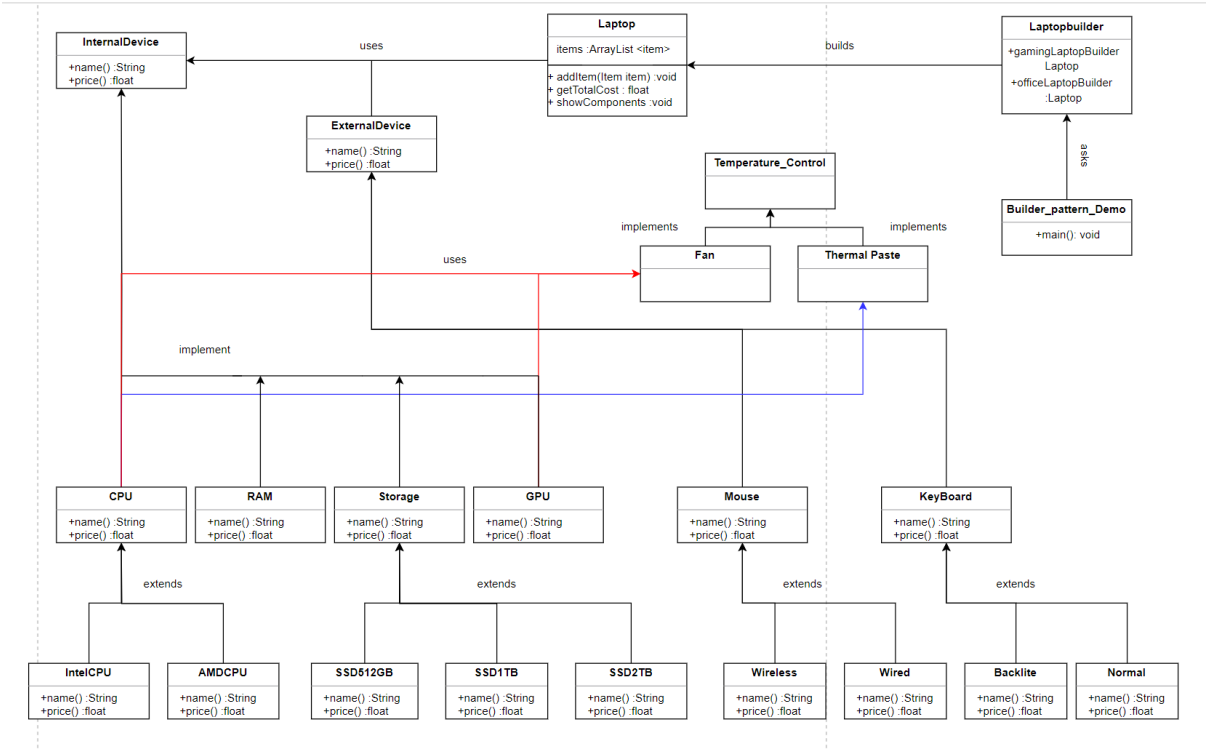
- Builder pattern builds a complex object using simple objects and using a step by step approach. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.
- A Builder class builds the final object step by step. This builder is independent of other objects.

Example(Implementation):

- Here to implement Builder Factory Design Pattern I have taken the example of Laptop_Builder. I am building laptop which gives the cost of all individual components and at last the total cost of the laptop.
- Here I have a class called ‘laptop’ which has 3 methods: ‘addinternalDevice() and addexternalDevice()’ which has void return type, ‘getCost’ which has return type of float and ‘showComponents()’ which has void return type.
- I have created 3 interfaces called InternalDevice ,ExternalDevice and Temperature_Control which has 2 methods: name() and price(). ‘name()’ will show the name of the device and ‘price()’ will show the price of the product. ‘Temperature_Control’ has 1 method ‘uses()’ which will show my component is using cooling device or not.

- I have also created the classes which implements the Device and Temperature_Control interface.
- Finally there is a builder class Laptop_Builder() which has 2 methods: OfficeLaptop_Builder() and GamingLaptop_Builder() in which return type is laptop.
- So in the main method I ask the user which type of Laptop they want to build Office or Gaming, and which components they want to add and at last it will show the Laptop Configuration and total cost of the laptop.

UML Diagram:



Code:

```
import java.util.ArrayList;
import java.util.Scanner;
interface Temperature_Control {
    void uses();
}
class Fan implements Temperature_Control {
    public void uses() {
        System.out.println("Uses Fan");
    }
}
class Thermal_Paste implements Temperature_Control {
    public void uses() {
        System.out.println("uses ThermalPaste");
    }
}
interface InternalDevice {
    String name();
    float price();
}
interface ExternalDevice {
    String name();
    float price();
}
abstract class CPU implements InternalDevice {
    public abstract String name();
    public abstract float price();
    public abstract void uses();
}
```

```
class IntelCPU extends CPU  {
    public String name() {
        return "Intel CPU";
    }
    public float price() {
        return 20000f; // Fixed price
    }
    public void uses()
    {
       System.out.println("It has fan and thermal paste for cooling");
    }
 }

class AMDCPU extends CPU {
    public String name() {
        return "AMD CPU";
    }
    public float price() {
        return 18000f; // Fixed price
    }
    public void uses()
    {
        System.out.println("It has fan and thermal paste for cooling");
    }
}
abstract class RAM implements InternalDevice {
}
abstract class Storage implements InternalDevice {
    public abstract String name();
```

```
    public abstract float price();
}
class SSD512GB extends Storage {
    public String name() {
        return "512GB SSD";
    }
    public float price() {
        return 15000f;
    }
}

class SSD1TB extends Storage {
    public String name() {
        return "1TB SSD";
    }
    public float price() {
        return 25000f;
    }
}
class SSD2TB extends Storage {
    public String name() {
        return "2TB SSD";
    }
    public float price() {
        return 45000f;
    }
}
abstract class GPU implements InternalDevice{
}
```

```
abstract class Mouse implements ExternalDevice {
    public abstract String name();
    public abstract float price();
}
class WiredMouse extends Mouse {
    public String name() {
        return "Wired Mouse";
    }
    public float price(){
        return 500f;
    }
}

class WirelessMouse extends Mouse {
    public String name() {
        return "Wireless Mouse";
    }
    public float price(){
        return 1000f;
    }
}

abstract class Keyboard implements ExternalDevice {
    public abstract String name();
    public abstract float price();
}

class BacklitKeyboard extends Keyboard {
    public String name() {
        return "Backlit Keyboard";
```

```
    }
    public float price(){
        return 2000f;
    }
}

class NormalKeyboard extends Keyboard {
    public String name() {
        return "Normal Keyboard";
    }
    public float price(){
        return 400f;
    }
}

class Laptop {
    ArrayList<InternalDevice> internalDevices = new ArrayList<>();
    ArrayList<ExternalDevice> externalDevices = new ArrayList<>();

    public void addInternalDevice(InternalDevice device) {
        internalDevices.add(device);
    }

    public void addExternalDevice(ExternalDevice device) {
        externalDevices.add(device);
    }

    public float getTotalCost() {
        float totalCost = 0;
        for (InternalDevice device : internalDevices) {
```

```
            totalCost += device.price();
        }
        for (ExternalDevice device : externalDevices) {
            totalCost += device.price();
        }
        return totalCost;
    }

    public void showComponents() {
        System.out.println("Internal and external components of the laptop are:");
        for (InternalDevice device : internalDevices) {
            System.out.println(device.name() + ": -> " + device.price());
        }
        for (ExternalDevice device : externalDevices) {
            System.out.println(device.name() + ": -> " + device.price());
        }
    }
}

class LaptopBuilder {
    public static Laptop officeLaptopBuilder(String cpuType, String storageType, String
mouseType, String keyboardType) {
        Laptop laptop = new Laptop();
        laptop.addInternalDevice(getCPU(cpuType));
        laptop.addInternalDevice(getStorage(storageType));
        laptop.addInternalDevice(new GPU() {
            public String name() {
                return "Intel Iris";
            }
            public float price() {
                return 10000f;
```

```
        }
    });
    laptop.addInternalDevice(new RAM() {
        public String name() {
            return "16GB";
        }
        public float price() {
            return 3000f;
        }
    });
    laptop.addExternalDevice(getMouse(mouseType));
    laptop.addExternalDevice(getKeyboard(keyboardType));
    return laptop;
}

public static Laptop gamingLaptopBuilder(String cpuType, String storageType, String
mouseType, String keyboardType) {
    Laptop laptop = new Laptop();
    laptop.addInternalDevice(getCPU(cpuType));
    laptop.addInternalDevice(getStorage(storageType));
    laptop.addInternalDevice(new GPU() {
        public String name() {
            return "RTX 4090";
        }
        public float price() {
            return 15000f;
        }
    });
    laptop.addInternalDevice(new RAM() {
        public String name() {
            return "64GB";
```

```
        }
        public float price() {
            return 8000f;
        }
    });
    laptop.addExternalDevice(getMouse(mouseType));
    laptop.addExternalDevice(getKeyboard(keyboardType));
    return laptop;
}

private static InternalDevice getCPU(String cpuType) {
    if (cpuType.equalsIgnoreCase("Intel")) {
        return new IntelCPU();
    } else if (cpuType.equalsIgnoreCase("AMD")) {
        return new AMDCPU();
    }
    return null;
}

private static InternalDevice getStorage(String storageType) {
    if (storageType.equalsIgnoreCase("512GB")) {
        return new SSD512GB();
    } else if (storageType.equalsIgnoreCase("1TB")) {
        return new SSD1TB();
    } else if (storageType.equalsIgnoreCase("2TB")) {
        return new SSD2TB();
    }
    return null;
}
```

```
    private static ExternalDevice getMouse(String mouseType) {
        if (mouseType.equalsIgnoreCase("Wired")) {
            return new WiredMouse();
        } else if (mouseType.equalsIgnoreCase("Wireless")) {
            return new WirelessMouse();
        }
        return null;
    }

    private static ExternalDevice getKeyboard(String keyboardType) {
        if (keyboardType.equalsIgnoreCase("Backlit")) {
            return new BacklitKeyboard();
        } else if (keyboardType.equalsIgnoreCase("Normal")) {
            return new NormalKeyboard();
        }
        return null;
    }
}

public class rhythmbuilder {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter laptop type (Office/Gaming): ");
        String laptopType = scanner.nextLine();

        System.out.println("Enter CPU type (Intel/AMD): ");
        String cpuType = scanner.nextLine();

        System.out.println("Enter storage type (512GB/1TB/2TB): ");
        String storageType = scanner.nextLine();
```

```
        System.out.println("Enter mouse type (Wired/Wireless): ");
        String mouseType = scanner.nextLine();

        System.out.println("Enter keyboard type (Backlit/Normal): ");
        String keyboardType = scanner.nextLine();

        Laptop laptop;
        if (laptopType.equalsIgnoreCase("Office")) {
            laptop = LaptopBuilder.officeLaptopBuilder(cpuType, storageType, mouseType,
keyboardType);
            System.out.println("Office Laptop Configuration:");
        } else if (laptopType.equalsIgnoreCase("Gaming")) {
            laptop = LaptopBuilder.gamingLaptopBuilder(cpuType, storageType, mouseType,
keyboardType);
            System.out.println("Gaming Laptop Configuration:");
        } else {
            System.out.println("Invalid laptop type!");
            return;
        }

        // Show Devices and Total Cost
        laptop.showComponents();
        System.out.println("Total Cost: $" + laptop.getTotalCost());

        // Call included method for AMD CPU
        if (cpuType.equalsIgnoreCase("AMD")) {
            AMDCPU amdCPU = new AMDCPU();
            amdCPU.uses();
        }
        else if(cpuType.equalsIgnoreCase("Intel")){
```

```
            IntelCPU intelCPU = new IntelCPU();
            intelCPU.uses();
        }
        else
        {
            System.out.println("Invalid Input");
        }
    }
}
```

Output:

```
Enter laptop type (Office/Gaming):
Gaming
Enter CPU type (Intel/AMD):
AMD
Enter storage type (512GB/1TB/2TB):
2tb
Enter mouse type (Wired/Wireless):
wireless
Enter keyboard type (Backlit/Normal):
backlit
Gaming Laptop Configuration:
Internal and external components of the laptop are:
AMD CPU: -> 18000.0
2TB SSD: -> 45000.0
RTX 4090: -> 15000.0
64GB: -> 8000.0
Wireless Mouse: -> 1000.0
Backlit Keyboard: -> 2000.0
Total Cost: $89000.0
It has fan and thermal paste for cooling
PS E:\Fourth sem\Design pattern lab>
```