

DESIGN PATTERNS - CPIT252

# Assignment\_2

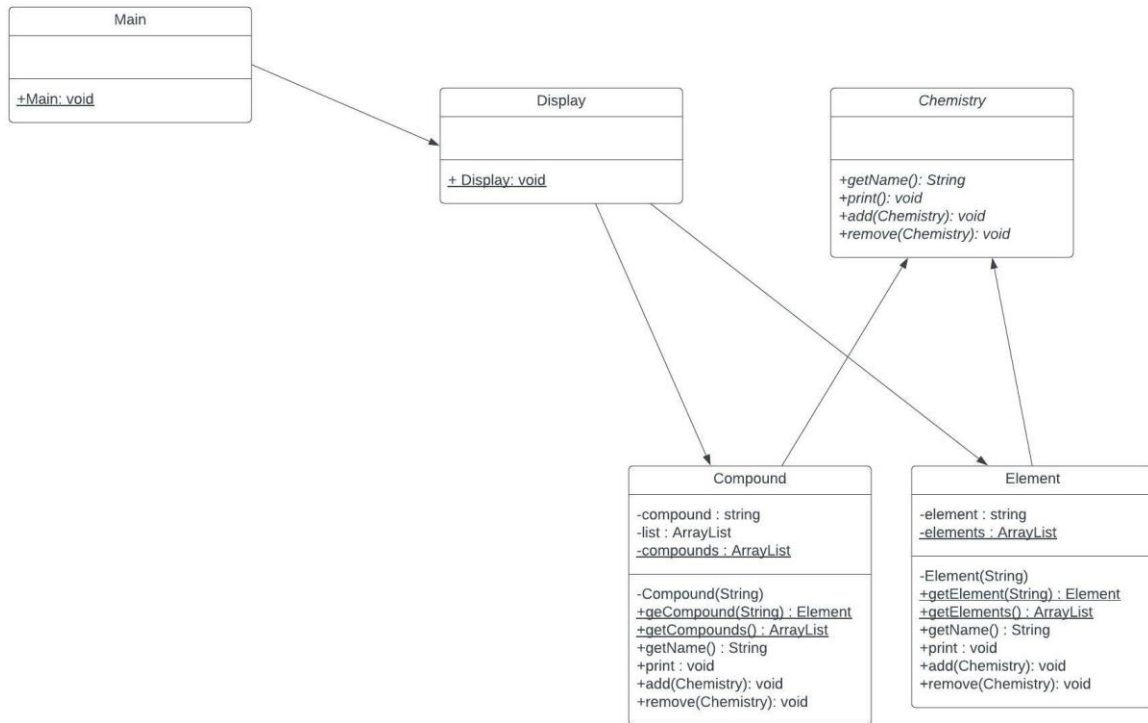
Shehab

---

## CONTENTS

UML and Brief Description .....	2
Main Class.....	3
Display .....	3
Switch .....	5
1> Add element .....	5
2> Add Compound .....	6
3> View all Elements.....	6
4> View all Compounds .....	7
5> Remove Element.....	7
6> Remove Compound .....	7
Chemistry Abstract Class .....	8
Element Class.....	8
Compound Class .....	9
Output .....	10
1# .....	10
2# .....	10
Why Composite? .....	11

## UML AND BRIEF DESCRIPTION



As we can see from the UML diagram the pattern that I've choose for this project is **Composite design pattern**. We have four classes without the "Main or Client". We will go through each class's implementation in the next section.

## MAIN CLASS

```
package com.CodeWithShehab;

public class Main {
    public static void main(String[] args) {
        /**
         * Elements examples >
         * Na
         * H
         * O
         * CL
         * Compounds examples >
         * H2o >>> HHO
         * NaCL
         */
        Display.displayMenu();
    }
}
```

Main class or client will use the static method inside “**Display**” class.

## DISPLAY

```
package com.CodeWithShehab;

import java.util.ArrayList;
import java.util.Scanner;

public class Display {

    private Display(){}

    public static void displayMenu() {
        Scanner scanner = new Scanner(System.in);
        String choice;
        boolean bool = false;
        System.out.println("*****");
        System.out.println("* Welcome to Chemistry Lab *");
        System.out.println("*****");
        while (!bool) {
            System.out.println("1> Add Element");
            System.out.println("2> Add Compound");
            System.out.println("3> View all Elements");
            System.out.println("4> View all Compounds");
            System.out.println("5> Remove Element");
            System.out.println("6> Remove Compound");
            System.out.println("!> any other number to exit");
            System.out.print("\t > Enter a choice$ ");
            choice = scanner.nextLine();
            switch (choice) {
                case "1":
                    System.out.print("Enter the name of the element: ");
                    Element.getElement(scanner.nextLine());
                    break;
                case "2":
                    System.out.print("Enter the name of the compound: ");
                    Compound compound = Compound.getCompound(scanner.nextLine());
                    System.out.println("How many elements in the compound?");
                    int y = Integer.parseInt(scanner.nextLine());
                    for (int i = 0; i < y; i++) {
                        System.out.println("Element number #" + (i+1) + " : ");
                        compound.add(Element.getElement(scanner.nextLine()));
                    }
                    System.out.println("How many compounds in the compound?");
                    int z = Integer.parseInt(scanner.nextLine());
            }
        }
    }
}
```

```

        for (int i = 0; i < z; i++) {
            System.out.println("Compound number #" + (i+1) + " : ");
            compound.add(Compound.getCompound(scanner.nextLine()));
        }
        break;
    case "3":
        System.out.println();
        ArrayList<Element> elements = Element.getElements();
        if (elements.size() == 0) {
            System.err.println("No elements in the System!");
            break;
        }
        for (int i = 0; i < elements.size(); i++) {
            System.out.print("#" + (i+1) + " ");
            elements.get(i).print();
        }
        System.out.println();
        break;
    case "4":
        System.out.println();
        ArrayList<Compound> compounds = Compound.getCompounds();
        if (compounds.size() == 0) {
            System.err.println("No compounds in the System!");
            break;
        }
        for (int i = 0; i < compounds.size(); i++) {
            System.out.print("#" + (i+1) + " ");
            compounds.get(i).print();
        }
        System.out.println();
        break;
    case "5":
        if (Element.getElements().size() == 0) {
            System.err.println("there is no elements in the system!");
            break;
        }
        System.out.println("Enter the name of the element you want to
remove");

        String name = scanner.nextLine();
        Element = null;
        for (int i = 0; i < Element.getElements().size(); i++) {
            if (Element.getElements().get(i).getName().equals(name)) {
                element = Element.getElements().get(i);
                Element.getElements().remove(i);
                System.out.println("Element has been removed!");
                break;
            }
        }
        if (element == null)
            System.err.println("element doesn't exists!");
        break;
    case "6":
        if (Compound.getCompounds().size() == 0) {
            System.err.println("there is no compounds in the system!");
            break;
        }
        System.out.println("Enter the name of the compound you want to
remove");

        String nam = scanner.nextLine();
        Compound compound1 = null;
        for (int i = 0; i < Compound.getCompounds().size(); i++) {
            if (Compound.getCompounds().get(i).getName().equals(nam)) {
                compound1 = Compound.getCompounds().get(i);
                Compound.getCompounds().remove(i);
                System.out.println("compound has been removed!");
                break;
            }
        }
    }
}

```

```

        }
        if(compound1 == null)
            System.err.println("compound doesn't exists!");
        break;
    default:
        bool = true;
        break;
    }
}
System.out.println("Thanks for using our system!");
}
}

```

“**Display**” class is the class which do interact with “**Compound**” and “**Element**”. This class prints a main menu where user must choose from **1** to **6** in a switch. And program ends if user entered any other number.

## SWITCH

### 1> ADD ELEMENT

Here user will be asked to enter the name of the element then, the input will be redirected getElement() method in “**Element**” class. This method contains a loop which is going to iterate size of the static array list in “**Element**” class times. And it will check if the element user tries to enter is already in the array list. If it is already there it will return it otherwise it will create a new object using the private constructor in the class, and that object will be added in the list increasing the size in the loop.

```


private Element(String element) {
    this.element = element;
    elements.add(this);
}

```

```

public static Element getElement(String name) {
    for (int i = 0; i < elements.size(); i++) {
        if(elements.get(i).getName().equals(name))
            return elements.get(i);
    }
    return new Element(name);
}

```



---

## 2> ADD COMPOUND

User first will be asked to enter the name of the compound and based on the name compound object will be created. Same as in “**Element**” if the compound exists it will be returned. Otherwise, the user will be asked how many elements this compound contains. Based on the answer provided there will be a for loop which will iterate that number of times. This loop will ask the user to enter the name of the elements these elements will be added in the static array list in “**Element**” class also they will be added using in the same compound object. Then the user will be asked how many compounds there in this compound are. Based on the answer another for loop will iterate adding all compounds to the static array list in the compound class.

```
case "2":
    System.out.print("Enter the name of the compound: ");
    Compound compound = Compound.getCompound(scanner.nextLine());
    System.out.println("How many elements in the compound?");
    int y = Integer.parseInt(scanner.nextLine());
    for (int i = 0; i < y; i++) {
        System.out.println("Element number #" + (i+1) + " : ");
        compound.add(Element.getElement(scanner.nextLine()));
    }
    System.out.println("How many compounds in the compound?");
    int z = Integer.parseInt(scanner.nextLine());
    for (int i = 0; i < z; i++) {
        System.out.println("Compound number #" + (i+1) + " : ");
        compound.add(Compound.getCompound(scanner.nextLine()));
    }
    break;
```

```
public static Compound getCompound(String compound){
    for (int i = 0; i < compounds.size(); i++) {
        if(compounds.get(i).getName().equals(compound))
            return compounds.get(i);
    }
    return new Compound(compound);
}
```

```
private Compound(String compound) {
    this.compound = compound;
    compounds.add(this);
}
```

---

## 3> VIEW ALL ELEMENTS

This method contains a loop that will iterate over all the elements stored in the static array list. Showing their names using print() and if there are no elements it will show an error message.

```
case "3":
    System.out.println();
    ArrayList<Element> elements = Element.getElements();
    if (elements.size() == 0) {
        System.err.println("No elements in the System!");
        break;
    }
    for (int i = 0; i < elements.size(); i++) {
        System.out.print("#" + (i+1) + " ");
        elements.get(i).print();
    }
    System.out.println();
    break;
```

---

#### 4> VIEW ALL COMPOUNDS

Same as -4- but it will also show what does this compound contains of.

```
case "3":
    System.out.println();
    ArrayList<Element> elements = Element.getElements();
    if (elements.size() == 0) {
        System.err.println("No elements in the System!");
        break;
    }
    for (int i = 0; i < elements.size(); i++) {
        System.out.print("#" + (i+1) + " ");
        elements.get(i).print();
    }
    System.out.println();
    break;
```

---

#### 5> REMOVE ELEMENT

In case of the static array list was empty an error message will appear. Otherwise, the name of the element that the user is trying to remove from the system will be asked to enter. A for loop will iterate number of times as the size of that static array list, searching for that element if the element does not exist an error will appear. Otherwise, a feedback message will appear telling the user that the element has been removed.

```
case "5":
    if (Element.getElements().size() == 0) {
        System.err.println("there is no elements in the system!");
        break;
    }
    System.out.println("Enter the name of the element you want to remove");
    String name = scanner.nextLine();
    Element element = null;
    for (int i = 0; i < Element.getElements().size(); i++) {
        if (Element.getElements().get(i).getName().equals(name)) {
            element = Element.getElements().get(i);
            Element.getElements().remove(i);
            System.out.println("Element has been removed!");
            break;
        }
    }
    if (element == null)
        System.err.println("element doesn't exists!");
    break;
```

---

#### 6> REMOVE COMPOUND

Same as -5-.

```
case "6":
    if (Compound.getCompounds().size() == 0) {
        System.err.println("there is no compounds in the system!");
        break;
    }
    System.out.println("Enter the name of the compound you want to remove");
    String nam = scanner.nextLine();
    Compound compound1 = null;
    for (int i = 0; i < Compound.getCompounds().size(); i++) {
        if (Compound.getCompounds().get(i).getName().equals(nam)) {
            compound1 = Compound.getCompounds().get(i);
            Compound.getCompounds().remove(i);
            System.out.println("compound has been removed!");
            break;
        }
    }
    if (compound1 == null)
        System.err.println("compound doesn't exists!");
    break;
```



## CHEMISTRY ABSTRACT CLASS

```
package com.CodeWithShehab;

public abstract class Chemistry {
    public abstract String getName();
    public abstract void print();
    public abstract void add(Chemistry chemistry);
    public abstract void remove(Chemistry chemistry);
}
```

This is the abstract class which both “**Element**” and “**Compound**” classes do implement. It can be an interface too.

## ELEMENT CLASS

```
package com.CodeWithShehab;

import java.util.ArrayList;

public class Element extends Chemistry{

    private String element;
    private static ArrayList<Element> elements = new ArrayList<>();

    private Element(String element) {
        this.element = element;
        elements.add(this);
    }

    public static Element getElement(String name) {
        for (int i = 0; i < elements.size(); i++) {
            if(elements.get(i).getName().equals(name))
                return elements.get(i);
        }
        return new Element(name);
    }

    public static ArrayList<Element> getElements() {
        return elements;
    }

    @Override
    public String getName() {
        return element;
    }

    @Override
    public void print() {
        System.out.println("Element = " + element);
    }

    @Override
    public void add(Chemistry chemistry) {}

    @Override
    public void remove(Chemistry chemistry) {}
}
```

Nothing to mention here just implementing the idea in a “**Composite**” way :D!

## COMPOUND CLASS

```
package com.CodeWithShehab;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

public class Compound extends Chemistry{

    private String compound;
    private List<Chemistry> list = new ArrayList<>();
    private static ArrayList<Compound> compounds = new ArrayList<>();

    private Compound(String compound) {
        this.compound = compound;
        compounds.add(this);
    }

    public static Compound getCompound(String compound){
        for (int i = 0; i < compounds.size(); i++) {
            if (compounds.get(i).getName().equals(compound))
                return compounds.get(i);
        }
        return new Compound(compound);
    }

    @Override
    public String getName() {
        return compound;
    }

    public static ArrayList<Compound> getCompounds() {
        return compounds;
    }

    @Override
    public void print() {
        System.out.println("Compound: " + compound);
        Iterator<Chemistry> iterator = list.iterator();
        System.out.println("\tContains of >");
        while (iterator.hasNext()) {
            Chemistry chemistry = iterator.next();
            System.out.print("\t\t");
            chemistry.print();
        }
    }

    @Override
    public void add(Chemistry chemistry) {
        list.add(chemistry);
    }

    @Override
    public void remove(Chemistry chemistry) {
        list.remove(chemistry);
    }
}
```

## OUTPUT

1#

```
*****
* Welcome to Chemistry Lab *
*****
1> Add Element
2> Add Compound
3> View all Elements
4> View all Compounds
5> Remove Element
6> Remove Compound
!> any other number to exit
  > Enter a choice$ 1

1> Add Element
2> Add Compound
3> View all Elements
4> View all Compounds
5> Remove Element
6> Remove Compound
!> any other number to exit
  > Enter a choice$ No elements in the System!
```

Error messages don't appear where they must be because of something related to IDE.

```
2> Add Compound
3> View all Elements
4> View all Compounds
5> Remove Element
6> Remove Compound
!> any other number to exit
  > Enter a choice$ No compounds in the System!

there is no elements in the system!
1> Add Element
2> Add Compound
3> View all Elements
4> View all Compounds
5> Remove Element
6> Remove Compound
!> any other number to exit
  > Enter a choice$ 1

1> Add Element
there is no compounds in the system!
2> Add Compound
3> View all Elements
4> View all Compounds
5> Remove Element
6> Remove Compound
!> any other number to exit
  > Enter a choice$
```

2#

```
3> View all Elements
4> View all Compounds
5> Remove Element
6> Remove Compound
!> any other number to exit
  > Enter a choice$ 1
Enter the name of the element: Na
1> Add Element
2> Add Compound
3> View all Elements
4> View all Compounds
5> Remove Element
6> Remove Compound
!> any other number to exit
  > Enter a choice$ 3

#1 Element = Na

1> Add Element
2> Add Compound
3> View all Elements
4> View all Compounds
5> Remove Element
6> Remove Compound
!> any other number to exit
  > Enter a choice$
```

```
5> Remove Element
6> Remove Compound
!> any other number to exit
  > Enter a choice$ 1
Enter the name of the element: CL
1> Add Element
2> Add Compound
3> View all Elements
4> View all Compounds
5> Remove Element
6> Remove Compound
!> any other number to exit
  > Enter a choice$ 3

#1 Element = Na
#2 Element = CL

1> Add Element
2> Add Compound
3> View all Elements
4> View all Compounds
5> Remove Element
6> Remove Compound
!> any other number to exit
  > Enter a choice$
```

```
> Enter a choice$ 3
Enter the name of the compound: NaCL
How many elements in the compound?
3
Element number #1 :
Na
Element number #2 :
CL
How many compounds in the compound?
1
1> Add Element
2> Add Compound
3> View all Elements
4> View all Compounds
5> Remove Element
6> Remove Compound
!> any other number to exit
  > Enter a choice$ 4

#1 Compound: NaCL
Contains of >
  Element = Na
  Element = CL
```

```
> Enter a choice$ 3

#1 Element = Na
#2 Element = CL

1> Add Element
2> Add Compound
3> View all Elements
4> View all Compounds
5> Remove Element
6> Remove Compound
!> any other number to exit
  > Enter a choice$
```

We notice the number of elements did not increase although I've reentered it. That is straightforward evidence that there aren't new objects that has been created. Not ignoring debugging :D!

## WHY COMPOSITE?

Since we are combining compounds and elements there is not any other design pattern that give you the ability where you can go far down as composite.

