TP2: Excercising Design Patterns

Table of Contents

1. Objectives

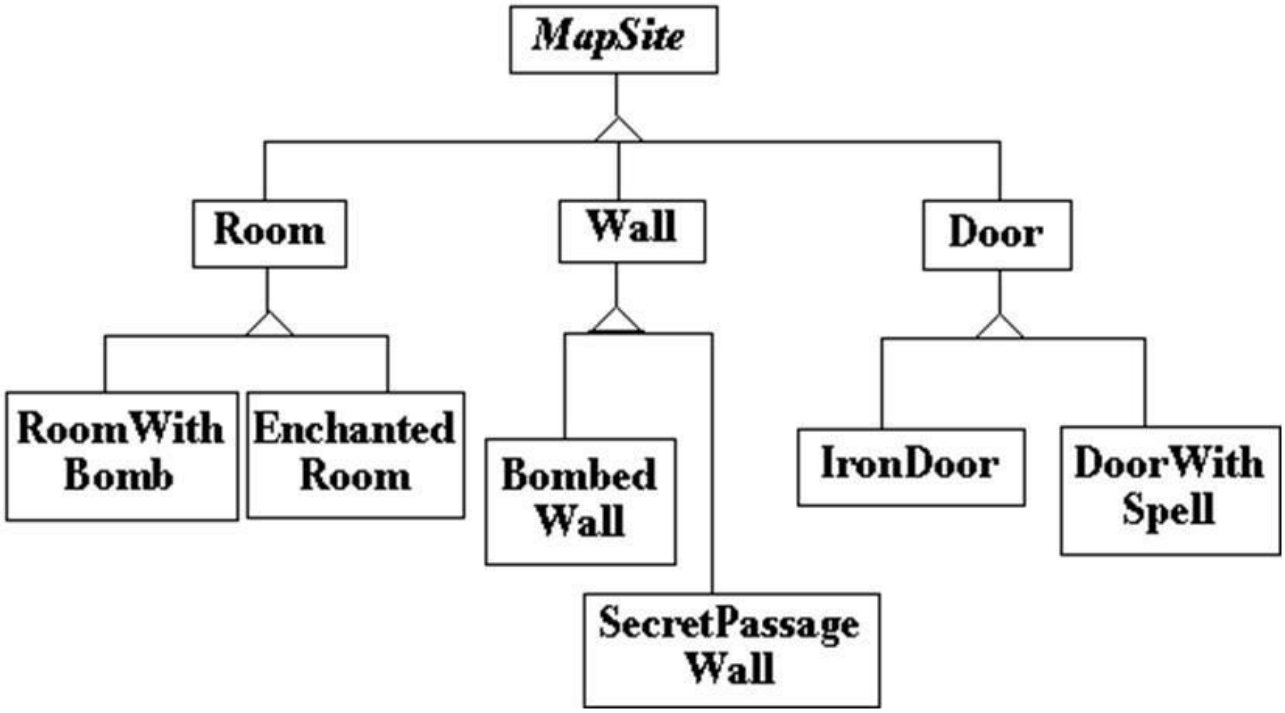
This practical class has three main objectives:

1. Recognise the need for a design pattern, and which design pattern is needed
2. Comment an existing design
3. Refactor existing code to make it comply with a design pattern

2. Recognizing design patterns

2.1. Recognizing a design from code:

Given the class hierarchy below:



- Guess which design pattern is used in the following java code:

Correction The factory method pattern is used here

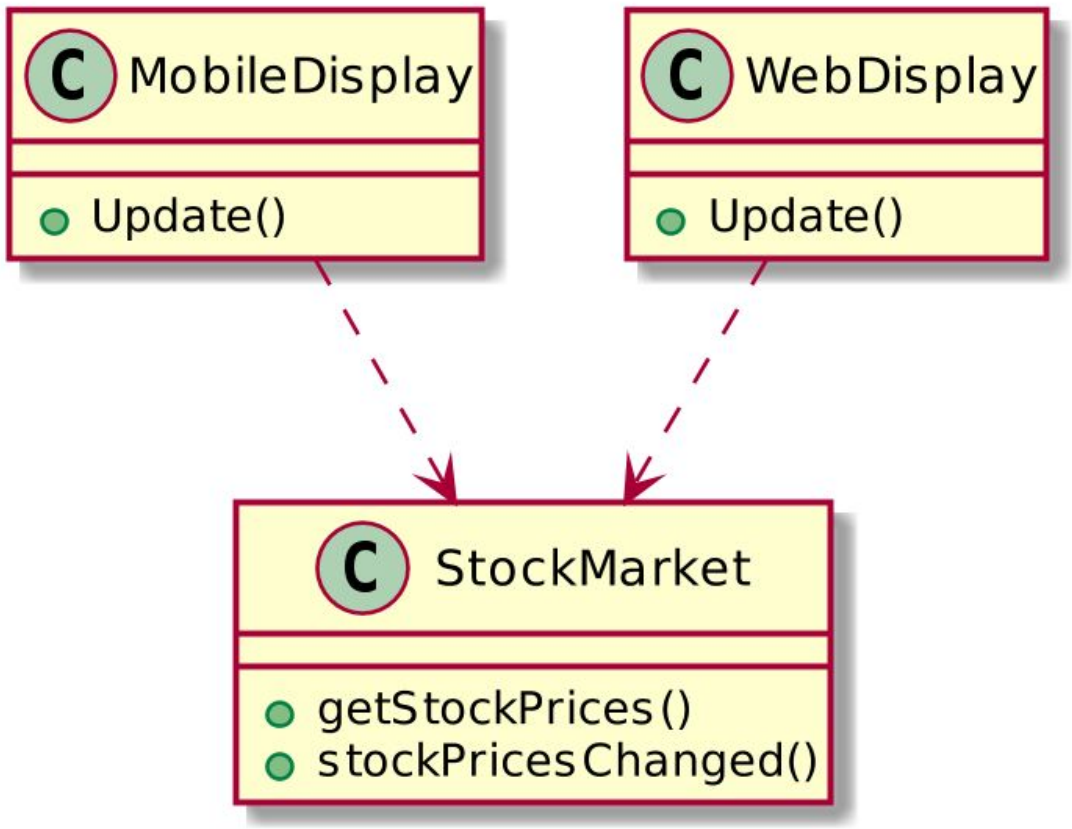
- Propose an extension of this code to add the "Enchanted Maze" class.

```
class MazeGame {
  public Maze makeMaze() { return new Maze(); }
  public Room makeRoom(int n ) { return new Room( n ); }
  public Wall makeWall() { return new Wall(); }
  public Door makeDoor(Room r1, Room r2) { return new Door(r1, r2); }

  public Maze CreateMaze() {
    Maze aMaze = makeMaze();
    Room r1 = makeRoom( 1 );
    Room r2 = makeRoom( 2 );
    Door theDoor = makeDoor( r1, r2);
    aMaze.addRoom( r1 );
    aMaze.addRoom( r2 );
    r1.setSide( North, makeWall() );
    r1.setSide( East, theDoor );
    r1.setSide( South, makeWall() );
    r1.setSide( West, makeWall() );
```

```
    r2.setSide( North, makeWall() );
    r2.setSide( East, makeWall() );
    r2.setSide( South, makeWall() );
    r2.setSide( West, theDoor );

    return aMaze;
  }
}
class BombedMazeGame extends MazeGame {
  public Room makeRoom(int n ) {
    return new RoomWithABomb( n );
  }
  public Wall makeWall() {
    return new BombedWall();
  }
  public Door makeDoor() {
    return new IronDoor();
  }
}

class EnchantedMazeGame extends MazeGame {

}
```

2.2. Recognizing from a diagram

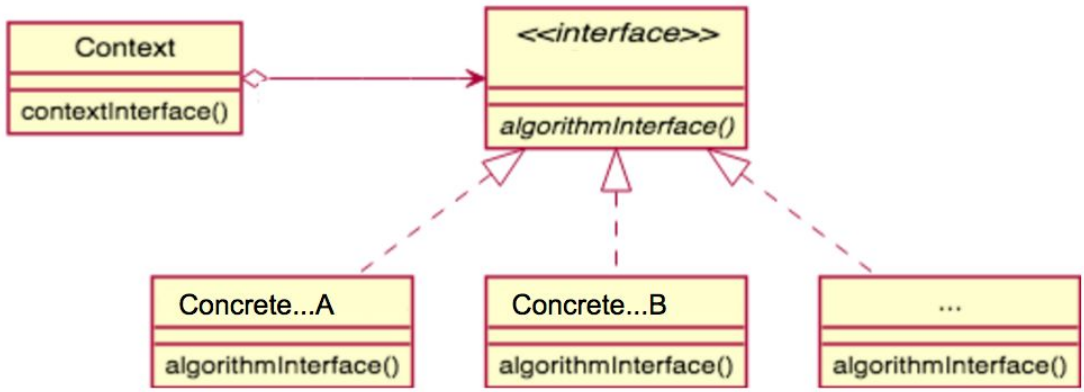
Can you guess which pattern is used in the following design:



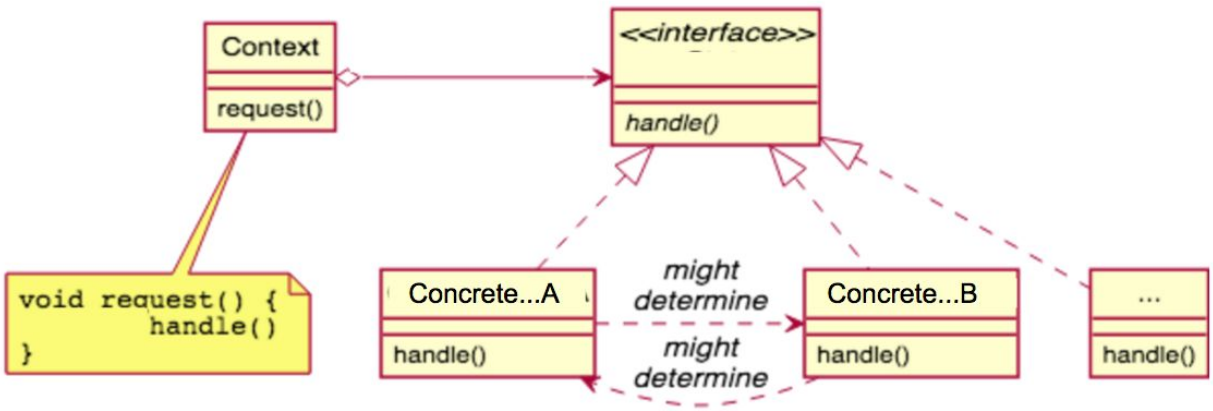
2.3. Differentiate 2 similars Patterns

(Exercice coming from a very good question of one your camarade in the class).
The 2 following patterns look very similar:

Pattern A:



Pattern B:



- Which ones are they ?
- What is the main difference between them ?
- Propose a real word example for both of them.

3. Embrace Peer Review and Feedbacks !

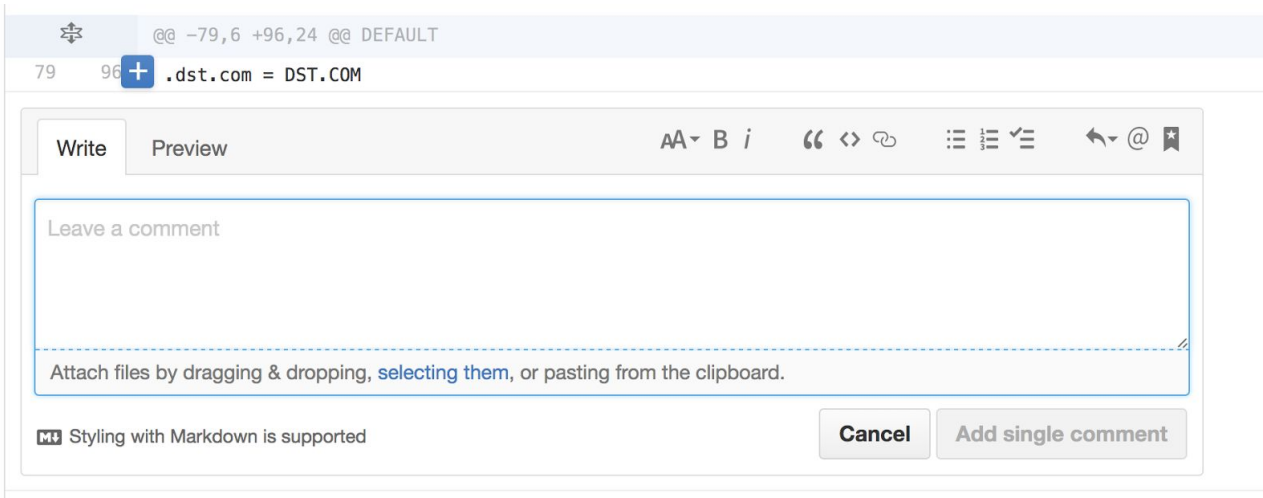
Peer Review is extremely important to improve your skills and produce better results.
Here you will do a code review of another team's code for the 1st TP Practical course.
You should provide at least 10 constructive feedbacks, comments on the other group's report.
Feedbacks can be about code style, architecture improvements, bugs, duplication, non respect of fundamental principles seen in the 2nd lecture...
Here are examples of feedbacks:

- "You should factorize your code here".
- Make an interface for this/these classes.
- A space is missing between the "sort(int[] arr)" and "{"
- variable name not clear
- ...

The best is to use github to make the comment.
Each team can upload its 1st practical work report on github and the other team can make comment.

To access commits it's here:

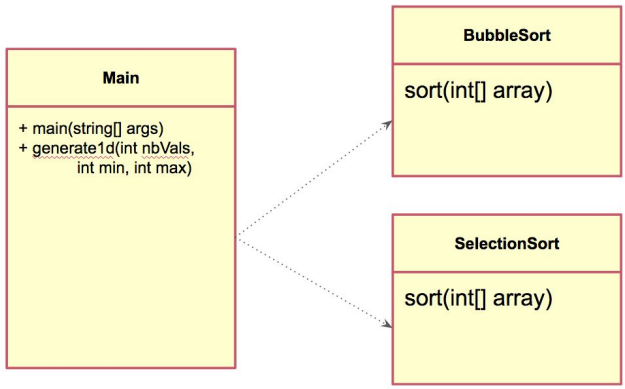
When you are on a commit page, you can click on the "+" button on the right of the line number, there you can make a comment:



Note: If there is a bug, you can also open an issue on the github page.

4. Make design generics.

In the first practical class we ended up with the following class diagram:



Your main code looked like this:

```
public class Main {
  public static void main(String[] args) {
    BubbleSort bubbleSort = new BubbleSort();
    System.out.println("Lauching benchmarks for bubbleSort");
    for (int i = 1000; i < 100000; i+=1000) {
      int[] arr = generate1d(i, 0, i);
      long start = new Date().getTime();
      bubbleSort.sort(arr);
      long end = new Date().getTime();
      System.out.println((i) + "," + (start - end));
    }
    SelectionSort selectionSort = new SelectionSort();
    System.out.println("Lauching benchmarks for selectionSort");
    for (int i = 1000; i < 100000; i+=1000) {
      int[] arr = generate1d(i, 0, i);
      long start = new Date().getTime();
      selectionSort.sort(arr);
      long end = new Date().getTime();
      System.out.println((i) + "," + (start - end));
    }
  }
  public static int[] generate1d(int nbVals, int min, int max) {
    ...
    return res;
  }
}
```

Comment on this design:

- Is this design **generic** ?
- Is it open for extension ? Why ?

Suggest a better model

- You can add other classes, interfaces, etc..

5. Refactoring

5.1. Refactoring a code base:

Modify the TP1 code to follow the design created in the previous question.

5.2. Extending a model

5.2.1. Strategy Pattern

- Verify you have changed the previous model following the Strategy Pattern.

5.2.2. Factories

We would like to support the creation of multiple sorting algorithms

- What modifications should we bring to model in order to support this functionality ?
- What type of factory can we use to create different types of sorting algorithm ?
- after making the appropriate diagram, implement the right factory pattern for this.

N.B.: One clean practice is to use an Enum for the name list of Sorting Algorithms.

=> Never hard-code* a String value ! If you reuse the same string value you don't want to rewrite the same string, but you want the string value to be in one place; the Enum.

*Hard-code: fix (data or parameters) in a program in such a way that they cannot be altered without modifying the program.

https://en.wikipedia.org/wiki/Hard_coding

6. Adapter (Bonus)

We have the following list “doubly chained”:

```
class DList {
  public void insert (DNode pos, Object o) { ... }
  public void remove (DNode pos) { ... }
  public void insertHead (Object o) { ... }
  public void insertTail (Object o) { ... }
  public Object removeHead () { ... }
  public Object removeTail () { ... }
  public Object getHead () { ... }
  public Object getTail () { ... }
}
```

And an interface Stack:

```
interface Stack {
  void push(Object o);
  Object pop();
  Object top();
}
```

We would like to use a DList as a stack.

- Make an Adapter so we can use the Dlist as a Stack.

Tip: The adapter should extends **DList** and implements the **Stack**