Lab X.

Objetivos

Os objetivos deste trabalho são:

- Utilizar padrões de comportamento (i.e., *Mediator, Memento, Null Object, Observer*) para resolver casos práticos.
- Aplicar boas práticas de programação por padrões

Nota: Para além do código no github, inclua também um ficheiro PDF ou PNG com o diagrama de classes da solução final.

X.1

A empresa *Lei Lao* pretende criar um sistema de leilões online. Simule este cenário criando as seguintes entidades, bem como outras que entenda serem fundamentais para uma boa modulação:

- *Produto*, caraterizado por um código único atribuído automaticamente (*int*), descrição (*String*) e preço base (*double*). Cada produto pode estar num dos seguintes estados: stock, leilão, vendas. A passagem a leilão deve incluir o tempo de duração neste estado. Caso não seja licitado deverá ser reposto no stock; caso vendido passará para a lista de vendas.
- *Cliente*, caraterizado por nome (*String*). Pode consultar os produtos em leilão. Pode licitar um (ou mais) produto por um determinado valor, passando a receber informação sempre que esse produto receba uma oferta mais elevada. Deverá ser informado quando a licitação termina e o produto é vendido.
- *Gestor*, caraterizado por nome (*String*). Tem acesso à lista de produtos em stock, em leilão e vendidos. Deve receber informação sempre que uma licitação é feita, ou um produto é vendido.

Para simular a informação trocada com cada cliente pode usar a linha de comando ou Java Swing. Crie um programa *main* de teste para simular uma situação real (por exemplo, com 5 produtos, 3 clientes e 1 gestor).

X.2

Considere o código seguinte. Reescreva-o, usando o padrão *Null Object*, de modo a evitar os erros de execução.

```
abstract class Employee {
   protected String name;
   public abstract String getName();
}

class Programmer extends Employee {
   public Programmer(String name) {
        this.name = name;
   }
   @Override
```

```
    public String getName() {
        return name;
    }
}

class EmployeeFactory {
    public static final String[] names = { "Mac", "Linux", "Win" };

    public static Employee getEmployee(String name) {
        for (int i = 0; i < names.length; i++) {
            if (names[i].equalsIgnoreCase(name)) {
                return new Programmer(name);
            }
        }
        return null;
    }
}

public class NullDemo {
    public static void main(String[] args) {

        Employee emp = EmployeeFactory.getEmployee("Mac");
        Employee emp2 = EmployeeFactory.getEmployee("Janela");
        Employee emp3 = EmployeeFactory.getEmployee("Linux");
        Employee emp4 = EmployeeFactory.getEmployee("Mack");

        System.out.println(emp.getName());
        System.out.println(emp2.getName());
        System.out.println(emp3.getName());
        System.out.println(emp4.getName());
    }
}
```

X.3

Tendo por base o padrão *Mediator*, descreva um problema e apresente uma implementação em Java onde aplique este padrão.
Juntamente com o código entregue um ficheiro README onde descreva: 1) o problema; 2) a solução; 3) referências para recursos/fontes utilizados.