Présentation Design Pattern Visiteur

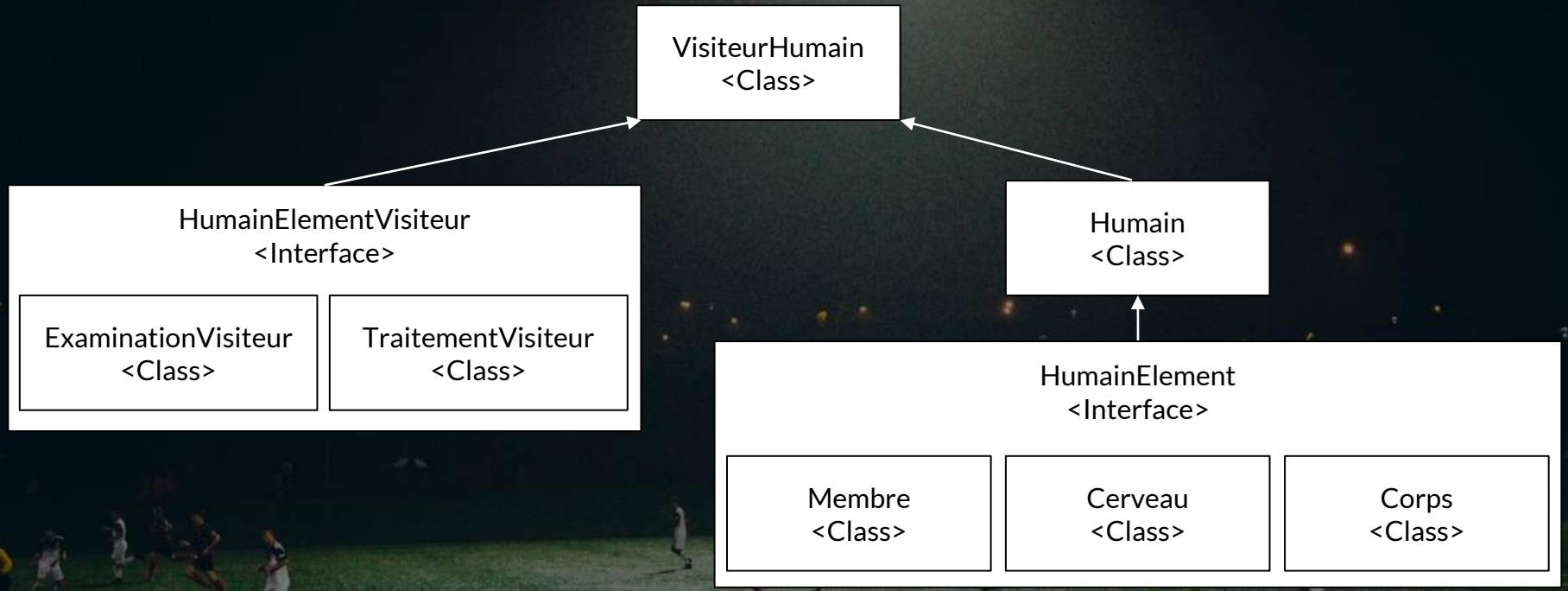
Par Dian LI et Ouaili CHERIKH

1. Introduction

Le design pattern “Visiteur” est un design pattern de type Comportemental, car il structure le comportement entre objets.

Il permet de faire une séparation entre des objets et des opérations faites sur ces objets. Ces opérations sont placées dans une nouvelle classe. Ce sera la classe “Visiteur”.

Schéma


VisiteurHumain
<Class>
HumainElementVisiteur
<Interface>
ExaminationVisiteur
<Class>
TraitementVisiteur
<Class>
Humain
<Class>
HumainElement
<Interface>
Membre
<Class>
Cerveau
<Class>
Corps
<Class>

Conception

Notre classe de départ

```java
// Classe Humain
class Humain {
    HumainElement[] elements;

    public HumainElement[] getElements() {
        return elements.clone(); // Retourne une copie du tableau de références
    }

    public Humain() {
        this.elements = new HumainElement[] {
                new Membre( name: "bras gauche"),
                new Membre( name: "bras droit"),
                new Membre( name: "jambe gauche"),
                new Membre( name: "jambe droite"),
                new Corps(),
                new Cerveau()
            };
    }
}
```

Conception

Notre classe de départ se décompose en plusieurs éléments

méthode “accept”

```java
// Interface HumainElement
interface HumainElement {
    void accept(HumainElementVisiteur visiteur);
}
```

```java
// Classe Membre
class Membre implements HumainElement {
    private String name;

    Membre(String name) { this.name = name; }

    String getName() { return this.name; }

    public void accept(HumainElementVisiteur visiteur) { visiteur.visit( membre: this); }
}

// Classe Cerveau
class Cerveau implements HumainElement {
    public void accept(HumainElementVisiteur visiteur) { visiteur.visit( cerveau: this); }
}

// Classe Corps
class Corps implements HumainElement {
    public void accept(HumainElementVisiteur visiteur) { visiteur.visit( corps: this); }
}
```

Conception

Notre interface Visiteur
avec les méthodes “visit”

```
// Interface HumainElementVisiteur
interface HumainElementVisiteur {
    void visit(Membre membre);
    void visit(Cerveau cerveau);
    void visit(Corps corps);
    void visitHumain(Humain humain);
}
```

Conception

Visiteur 1:
Examiner les éléments de l'Humain

```java
// Classe HumainElementExaminationVisiteur
class HumainElementExaminationVisiteur implements HumainElementVisiteur {
    public void visit(Membre membre) {
        System.out.println("Examiner "+ membre.getName());
    }

    public void visit(Cerveau cerveau) { System.out.println("Examiner Cerveau"); }

    public void visit(Corps corps) { System.out.println("Examiner Corps"); }

    public void visitHumain(Humain humain) {
        System.out.println("\nExaminer Humain");
        for(HumainElement element : humain.getElements()) {
            element.accept( visiteur: this);
        }
        System.out.println("Humain examiné");
    }
}
```

Conception

Visiteur 2:
Traiter les éléments de l'Humain

```
// Classe HumainElementTraitementVisiteur
class HumainElementTraitementVisiteur implements HumainElementVisiteur {
    public void visit(Membre membre) { System.out.println("Plâtrer le "+ membre.getName()); }

    public void visit(Cerveau cerveau) { System.out.println("Secouer le cerveau"); }

    public void visit(Corps corps) { System.out.println("Injecter le traitement dans le corps"); }

    public void visitHumain(Humain humain) {
        System.out.println("\nSoigner l'humain");
        for(HumainElement element : humain.getElements()) {
            element.accept( visiteur: this);
        }
        System.out.println("L'humain est soigné");
    }
}
```

Conception

La classe “main” pour la demo

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

        Humain humain = new Humain();

        HumainElementVisiteur examinationVisiteur = new HumainElementExaminationVisiteur();
        HumainElementVisiteur traitementVisiteur = new HumainElementTraitementVisiteur();

        examinationVisiteur.visitHumain(humain);
        traitementVisiteur.visitHumain(humain);
    }
}
```

```
MacBook-Pro-de-DIAN:visiteur dian$ javac VisiteurHumain.java
MacBook-Pro-de-DIAN:visiteur dian$ java VisiteurHumain

Examiner Humain
Examiner bras gauche
Examiner bras droit
Examiner jambe gauche
Examiner jambe droite
Examiner Corps
Examiner Cerveau
Humain examiné

Soigner l'humain
Plâtrer le bras gauche
Plâtrer le bras droit
Plâtrer le jambe gauche
Plâtrer le jambe droite
Injecter le traitement dans le corps
Secouer le cerveau
L'humain est soigné
```

DEMO

Conclusion

Avantages:

- Ajouter de nouvelles opérations sans altérer l'objet sur lequel elles sont effectuées
- Regrouper en une seule classe les opérations similaires

Inconvénients:

- Ajouter une extension à une classe nécessite de modifier toutes les classes "Visiteur" qui agissent sur elle
- La classe "Visiteur" n'a pas accès aux méthodes privées de la classe sur laquelle elle doit opérer

Merci de votre attention

Des questions? J'espère que non