

**b e p i a**  
Institut de design, management  
et d'innovation en Suisse

POO

**Hes-SO**  
Haute Ecole Spécialisée  
de Suisse occidentale

## Design Pattern Visitor

- Motif de conception *Visitor*
  - Création d'une classe externe qui agit sur les données d'autres classes
  - Utile pour effectuer une certaine opération sur toutes ou la plupart des instances d'un petit nombre de classes

06/04/08

1

---

---

---

---

---

---

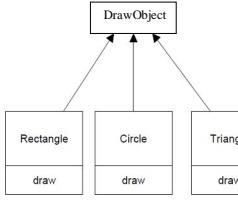
**b e p i a**  
Institut de design, management  
et d'innovation en Suisse

POO

**Hes-SO**  
Haute Ecole Spécialisée  
de Suisse occidentale

## Design Pattern Visitor

- Motivation



Chaque classe a un code similaire pour se dessiner

06/04/08

2

---

---

---

---

---

---

**b e p i a**  
Institut de design, management  
et d'innovation en Suisse

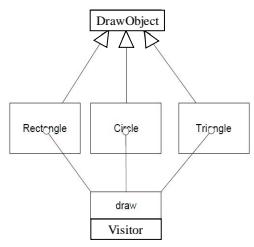
POO

**Hes-SO**  
Haute Ecole Spécialisée  
de Suisse occidentale

## Design Pattern Visitor

- Motivation

On déplace chacune des méthodes `draw()` dans une classe **Visitor**



06/04/08

3

---

---

---

---

---

---

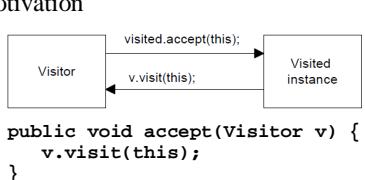
**b e p i r**  
Bibliothèque de projets didactiques  
et d'application en Informatique

POO

**Hes-SO**  
Haute Ecole Spécialisée  
de Suisse Occidentale

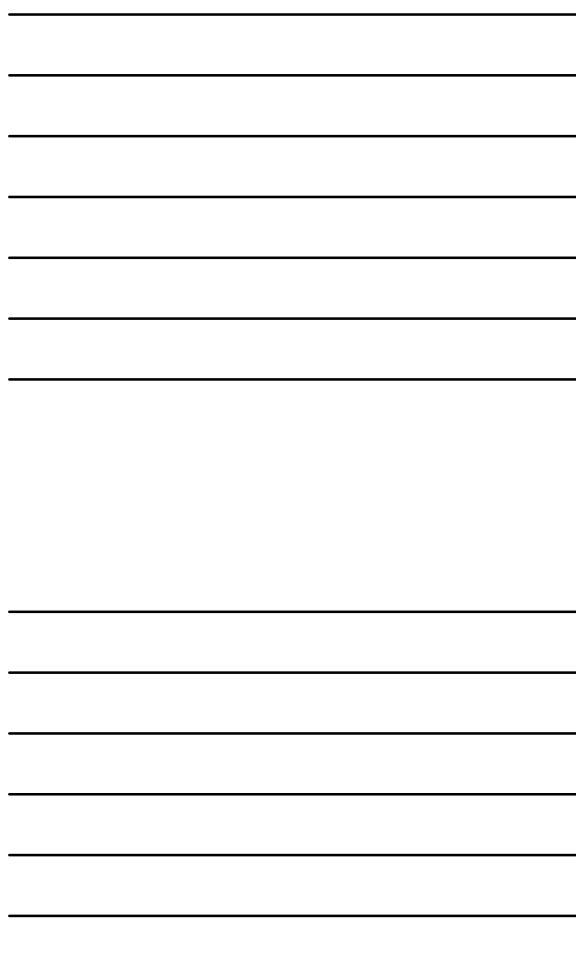
## *Design Pattern Visitor*

- Motivation
  - Que veut dire visiter?
    - Une classe externe ne peut agir sur une autre classe qu'à travers les méthodes publiques de celle-ci
    - Visiter c'est appeler une méthode **accept()**, prédefinie dans la classe à visiter, avec en paramètre l'objet **Visitor**
    - La méthode **accept()** appelle la méthode **visit()** de la classe **Visitor** avec une référence sur l'objet courant



06/04/08

5



# Design Pattern Visitor

## Design Pattern Visitor

- Exemple
  - Calcul du nombre de jours de vacances de tous les employés

```
abstract public class Visitor {  
    abstract public void visit(Employee emp);  
}
```
  - Remarque : on aurait aussi pu utiliser une interface

06/04/08

7

---

---

---

---

---

---

---

## Design Pattern Visitor

- Exemple
  - Visiteur

```
public class VacationVisitor extends Visitor {  
    protected int totalDays;  
    public VacationVisitor() { totalDays = 0; }  
    public void visit(Employee emp) {  
        totalDays += emp.getVacDays();  
    }  
    public int getTotalDays() { return totalDays; }  
}
```

06/04/08

8

---

---

---

---

---

---

---

## Design Pattern Visitor

- Exemple
  - Code client

```
VacationVisitor vac = new VacationVisitor();  
for (int i=0;i<employees.length;i++)  
    employees[i].accept(vac);  
System.out.println(vac.getTotalDays());
```

06/04/08

9

---

---

---

---

---

---

---

**b e p i a**  
Bureau des projets d'enseignement  
et d'innovation en Informatique

**POO**

**Hes-SO**  
Haute Ecole Spécialisée de Suisse Occidentale

## Design Pattern Visitor

- Résumé de la visite
  1. On itère à travers les employés
  2. On appelle la méthode **accept()** de chaque employé
  3. L'instance de **Employee** appelle la méthode **visit()** de **Visitor**
  4. L'objet **Visitor** récupère les jours de vacances et les ajoute au total
  5. Le résultat est disponible pour affichage

06/04/08

10

---



---



---



---



---



---



---



---



---



---

**b e p i a**  
Bureau des projets d'enseignement  
et d'innovation en Informatique

**POO**

**Hes-SO**  
Haute Ecole Spécialisée de Suisse Occidentale

## Design Pattern Visitor

- Exemple

```
public class Boss extends Employee {
    int bonusDays;
    public Boss(String nom, float sal,
               int vac, int sick) {
        super(nom, sal, vac, sick);
    }
    public void accept(Visitor v) { v.visit(this); }
    public int getBonusDays() { return bonusDays; }
}
abstract public class Visitor {
    abstract public void visit(Employee emp);
    abstract public void visit(Boss boss);
}
```

06/04/08

11

---



---



---



---



---



---



---



---



---



---

**b e p i a**  
Bureau des projets d'enseignement  
et d'innovation en Informatique

**POO**

**Hes-SO**  
Haute Ecole Spécialisée de Suisse Occidentale

## Design Pattern Visitor

- Exemple

```
public class BonusVacationVisitor extends Visitor {
    protected int totalDays;
    public BonusVacationVisitor() { totalDays = 0; }
    public void visit(Boss boss) {
        totalDays += boss.getVacDays();
        totalDays += boss.getBonusDays();
    }
    public void visit(Employee emp) {
        totalDays += emp.getVacDays();
    }
    public int getTotalDays() { return totalDays; }
}
```

06/04/08

12

---



---



---



---



---



---



---



---



---

**b e p i r**  
Bibliothèque de projets didactiques  
et d'application en Informatique

POO

**Hes-SO**  
Haute Ecole Spécialisée  
de Suisse Occidentale

---

---

---

---

---

---

---

---

---

---

**b e p i a**  
Bibliothèque de langages d'applications  
et d'interfaces de données

**POO**

**Hes-SO**  
Haute école spécialisée  
de Suisse occidentale

## *Design Pattern Visitor*

- Exemple

```
VacationVisitor vac = new VacationVisitor();
BonusVacationVisitor bvac =
    new BonusVacationVisitor();
for (Employee collab : employees) {
    collab.accept(vac);
    collab.accept(bvac);
}
System.out.println(vac.getTotalDays());
System.out.println(bvac.getTotalDays());
```

---

---

---

---

---

---

**b e s p i a**  
Bibliothèque de langages d'application  
et d'entretien de données

POO

**Hes-SO**  
Haute école spécialisée de Suisse occidentale

## *Design Pattern Visitor*

- Exemple
  - Il faut également implémenter dans la classe **VacationVisitor** la méthode ajoutée à **Visitor**

```
public void visit(Boss boss) {  
    visit((Employee)boss);  
}
```
- Remarque
  - Le motif de conception *Visitor* est lié aux motifs *Iterator* et *Composite* car un itérateur permet de parcourir des collections ou des arbres