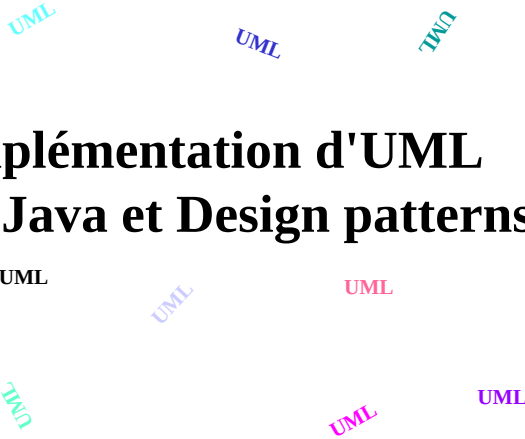




Implémentation d'UML avec Java et Design patterns

Dominique Ribouchon – Février 2005

Objectifs du cours

- Etudier quelques exemples d'implémentation de spécifications UML en Java, avec utilisation de design patterns:
 - diagramme de classes: généralisation
 - diagramme d'états-transition

Plan du cours

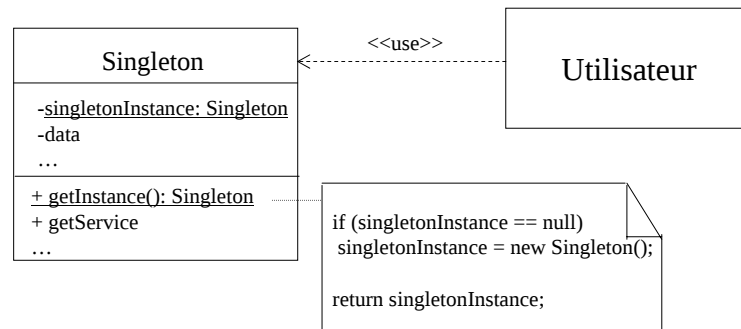
- Les designs pattern
- Généralisation avec le pattern Delegation
- Le diagramme d'états-transitions avec le pattern State:
 - Principes généraux
 - Instanciation d'un objet à états
 - Etat composite

Designs patterns

- Solutions communes à des problèmes de conception récurrents
- A l'échelle d'une ou peu de classes
- "Patron de solution" documenté
- Ex: "Gamma" patterns
*(Design patterns Element of Reusable Object-Oriented Software -
Erich Gamma - R. Helm - R. Johnson – J. Vlissides
Addison-Wesley)*

Exemple: le Singleton

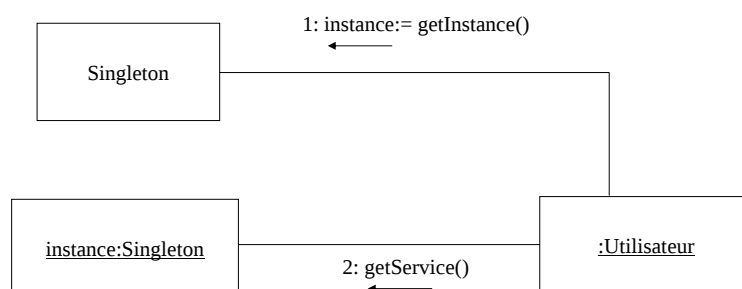
- Diagramme de classes



5

Le Singleton (2)

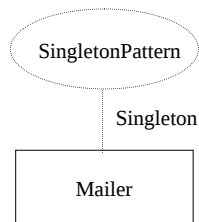
- Diagramme de collaboration



6

Le singleton (3)

Lien avec le pattern



"la classe Mailer joue le rôle de Singleton dans la collaboration paramétrée SingletonPattern"

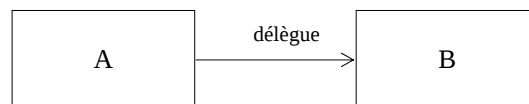
7

Plan du cours

- Les designs pattern
- Généralisation avec le pattern Delegation
- Le diagramme d'états-transitions avec le pattern State:
 - Principes généraux
 - Instanciation d'un objet à états
 - Etat composite

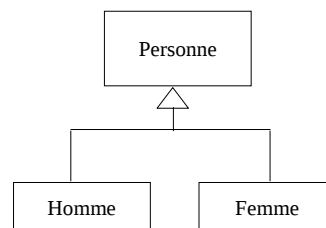
8

Le pattern délégation



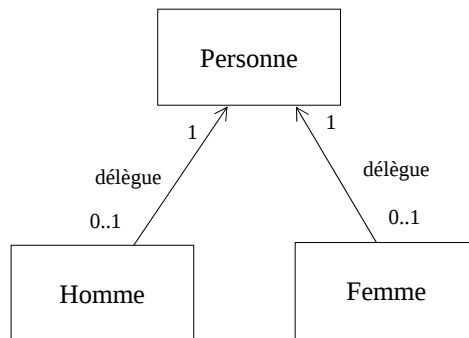
9

Généralisation Héritage ou délégation?



10

Généralisation implémentée par Délégation



=> Composition

11

Délégation - Exemple

- Marie, une femme:



12

Implémentations Java

- Héritage: héritage Java (extends)

```
public class Personne {  
    private int age;  
    public Personne() {age=0;}  
    public int getAge() {return age;}  
}  
  
public class Femme extends Personne {  
    private int nbMaternites;  
    public Femme() {nbMaternites=0;}  
    public int getNbMaternites{return nbMaternites;}  
}
```

Utilisation de l'objet marie, instance de femme:
marie Femme = new Femme();
....
ageMarie = marie.getAge();

13

Implémentations Java (2)

- Délégation: attribut de type classe

```
public class Personne {  
    private int age;  
    public Personne() {age=0;}  
    public int getAge() {return age;}  
}  
  
public class Femme {  
    private int nbMaternites;  
    private Personne personne;  
    public Femme() {  
        nbMaternites=0;  
        personne = new Personne();  
    }  
    public int getNbMaternites{return nbMaternites;}  
    public int getAge(){return personne.getAge();}  
}
```

Utilisation de l'objet marie, instance de femme: **idem héritage**

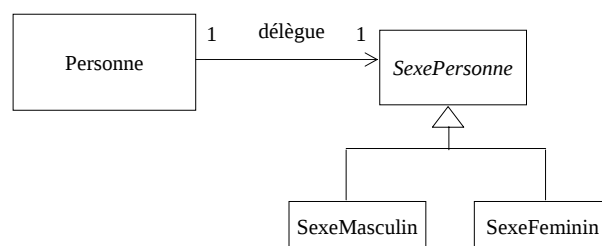
14

Comparaison

- Héritage:
 - Avantages: extensibilité, gestion automatique des liens
 - Inconvénients: rigidité des objets (des liens)
- Délégation:
 - Avantages: objets modifiables dynamiquement, implémentation de la généralisation dans un langage sans héritage
 - Inconvénients: codage plus lourd (gestion manuelle des liens)

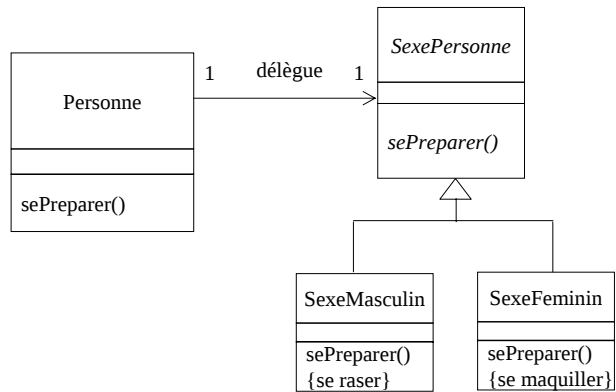
15

Une variante du pattern délégation



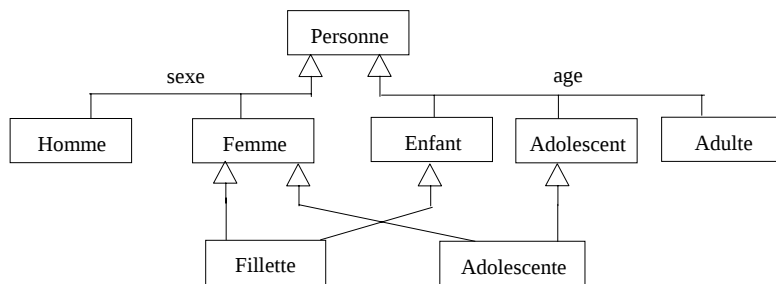
16

Délégation et polymorphisme



17

Généralisation multiple



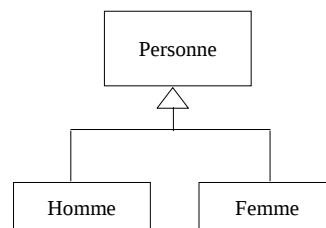
18

Généralisation multiple (2)

- Comment transformer la généralisation multiple grâce à la délégation?
- Quel problème présentait la première solution?

19

Autre solution pour implémenter le diagramme suivant?



20

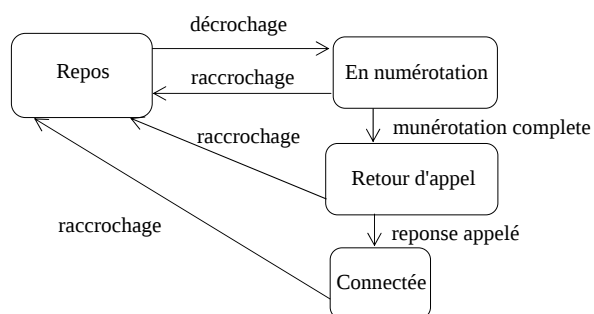
Plan du cours

- Les designs pattern
- Généralisation avec le pattern Delegation
- Le diagramme d'états-transitions avec le pattern State:
 - Principes généraux
 - Instanciation d'un objet à états
 - Etat composite

21

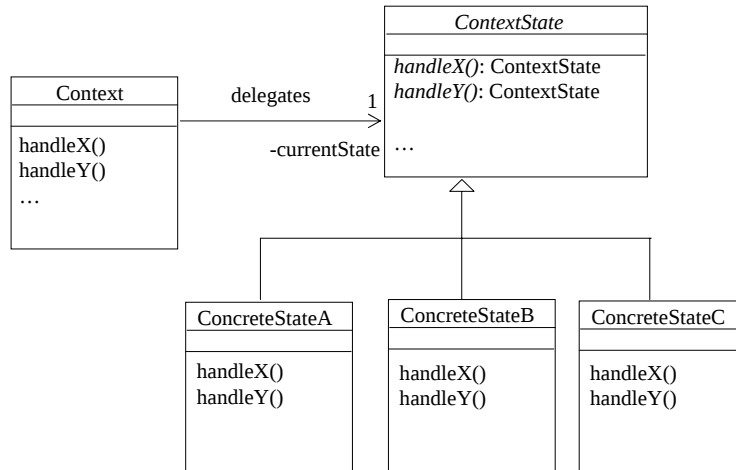
Diagramme d'états-transitions

Classe: Ligne téléphonique



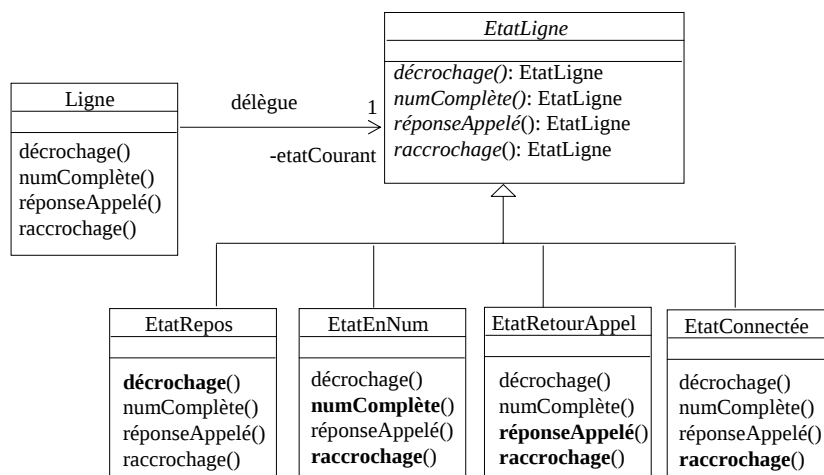
22

Le design pattern State



23

Le pattern State appliqué à la Ligne téléphonique



24

Avantages et inconvénients

- Avantages:
 - Lisibilité – les états et les transitions sont explicites
 - Extensibilité
- Inconvénient: code peu compact – augmentation du nombre de classes

25

Implémentation Java

```
public class Ligne {
    private int numeroLigne;
    private EtatLigne etatCourant;

    public void decrochage(){etatCourant = etatCourant.decrochage();}
    public void numComplete(){etatCourant = etatCourant.numComplete();}
    public void reponseAppele(){etatCourant= etatCourant.reponseAppele();}
    public void raccrochage(){etatCourant = etatCourant.raccrochage();}
    public void afficherEtat(){etatCourant.afficherEtat();}

    public Ligne(){
        numeroLigne=4;
        etatCourant = new EtatRepos();
    }
}
//*****
private abstract class EtatLigne{
    protected abstract EtatLigne decrochage();
    protected abstract EtatLigne numComplete();
    protected abstract EtatLigne reponseAppele();
    protected abstract EtatLigne raccrochage();
    protected abstract void afficherEtat();
}
```

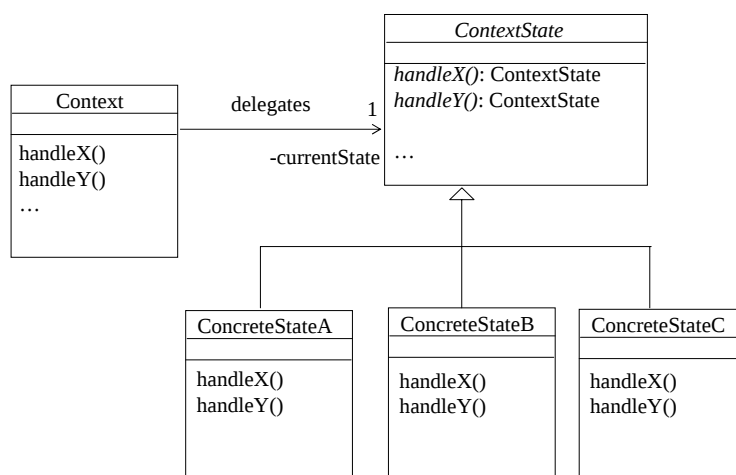
26

Implémentation Java(2)

```
private class EtatRepos extends EtatLigne{
    protected EtatLigne decrochage(){
        System.out.println( "Tonalité invitation à numéroté
                               sur la ligne " + numeroLigne);
        return new EtatEnNum();
    }
    protected EtatLigne numComplete(){
        System.out.println( "erreur: evt non attendu dans cet état");
        return this;
    }
    protected EtatLigne reponseAppelle(){
        System.out.println( "erreur: evt non attendu dans cet état");
        return this;
    }
    protected EtatLigne raccrochage(){
        System.out.println( "erreur: evt non attendu dans cet état");
        return this;
    }
    protected void afficherEtat(){
        System.out.println( "Etat: Repos");
    }
}
```

27

Placement des attributs



28

Exercice

- Faire le diagramme d'états-transition d'une personne sur le critère de l'âge:
 - Les états sont: Enfant, Adolescent, Adulte
 - Dans chacun de ces états, une personne prend en compte l'événement "dormir", mais le traite différemment:
 - Dans l'état "Enfant": écouter un conte
 - Dans l'état "Adolescent": écouter walk-man
 - Dans l'état "Adulte": lire roman
- Concevoir ce diagramme d'états-transitions avec le design pattern State

29

Plan du cours

- Les designs pattern
- Généralisation avec le pattern Delegation
- Le diagramme d'états-transitions avec le pattern State:
 - Principes généraux
 - Instanciation d'un objet à états
 - Etat composite

30

Exemple

Naissance d'une personne, Marie:



31

Possibilité d'instanciation

```
Class Personne {  
    public Personne() {  
        etatAgeCourant = new EtatEnfant();  
    }  
    private EtatAge etatAgeCourant;  
}
```

Quel est l'inconvénient de cette solution?

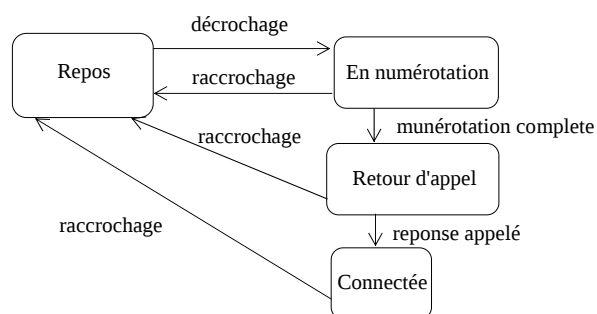
32

Plan du cours

- Les designs pattern
- Généralisation avec le pattern Delegation
- Le diagramme d'états-transitions avec le pattern State:
 - Principes généraux
 - Instanciation d'un objet à états
 - Etat composite

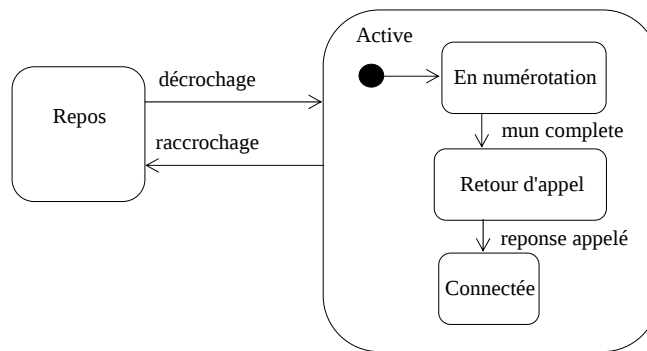
33

La ligne téléphonique sans état composite



34

La ligne téléphonique avec état composite



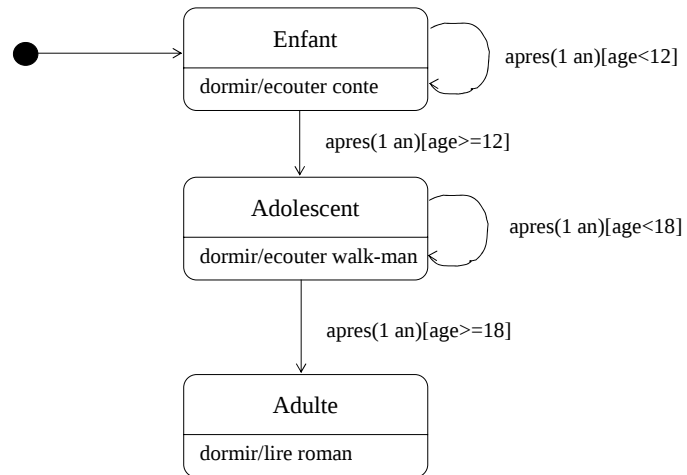
35

Exercice

- Comment concevoir un état composite dans l'esprit du design pattern State?

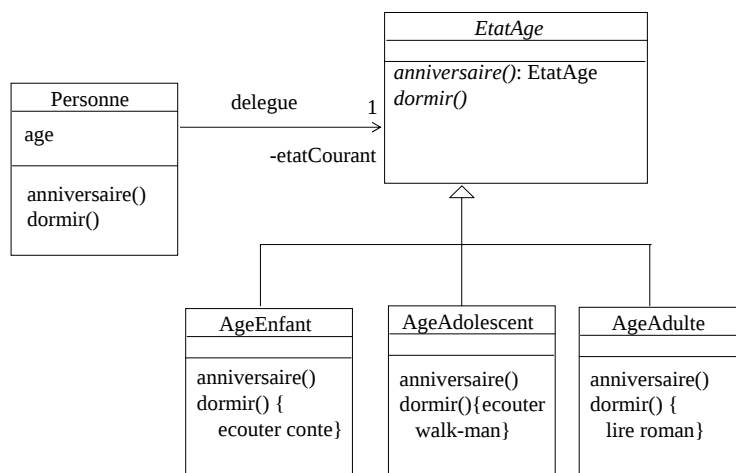
36

Correction exercice Age



37

Correction exercice Age (2)



38