Université Nice Sophia Antipolis

Patrons de conception: Adaptateur

F. Mallet

miage.m1@gmail.com

http://deptinfo.unice.fr/~fmallet/

Université Nice Sophia Antipolis

Motivation

- Les besoins pour une bonne conception et du bon code :
  - Extensibilité
  - Flexibilité
  - Facilité à maintenir
  - Réutilisabilité
  - Les qualités internes
  - Meilleure spécification, construction, documentation

Université Nice Sophia Antipolis

Historique

- MVC
- Gang of Four : Erich Gamma, Richard Helm,Ralph Johnson, John Vlissides
  - Définition de 23 patterns
- Design Patterns – Elements of Reusable Object-Oriented Software, Addison Wesley, 1994
- Un Design Pattern nomme, abstrait et identifie les aspects essentiels d'une structuration récurrente, ce qui permet de créer une modélisation orientée objet réutilisable

Université Nice Sophia Antipolis

Classification

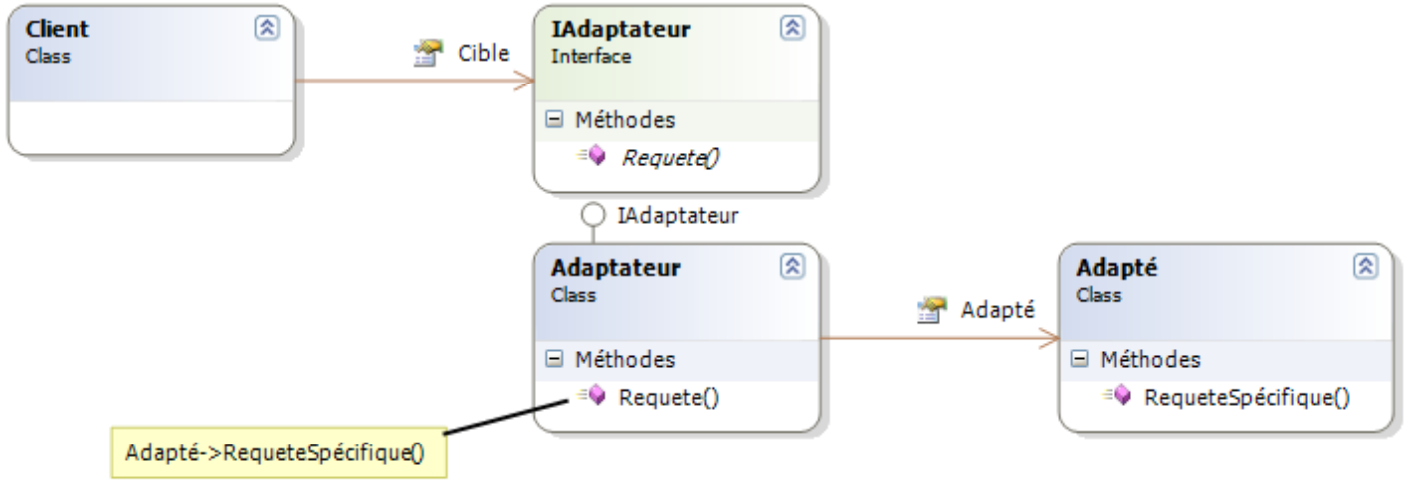
- Création
  - Comment un objet peut être créé
  - Indépendance entre la manière de créer et la manière d'utiliser
- Structure
  - Comment les objets peuvent être combinés
  - Indépendance entre les objets et les connexions
- Comportement
  - Comment les objets communiquent
  - Encapsulation de processus (ex : observer/observable)

Adaptateur

- Intention
  - Convertir l'interface d'une classe en une autre interface qui est attendue par un client.
  - Permet de faire collaborer des classes qui n'auraient pas pu le faire à cause de l'incompatibilité de leurs interfaces
- Exemple
  - Une classe de bibliothèque conçue pour la réutilisation ne peut pas l'être à cause d'une demande spécifique de l'application
  - Les Adapter de java : MouseAdapter, WindowAdapter,etc.
- *Synonymes : Wrapper, Mariage de convenance*

Structure

- Une cible (**`IAdaptateur`**) définit l'interface spécifique à l'application que le client utilise
- Le **`Client`** collabore avec les objets qui sont conformes à l'interface de **`IAdaptateur`**
- La classe à adapter (**`Adapté`**) est l'interface existante qui a besoin d'adaptation
- L'adaptateur (**`Adaptateur`**) adapte effectivement l'interface de **`Adapté`** à l'interface de **`IAdaptateur`** par traduction des accès

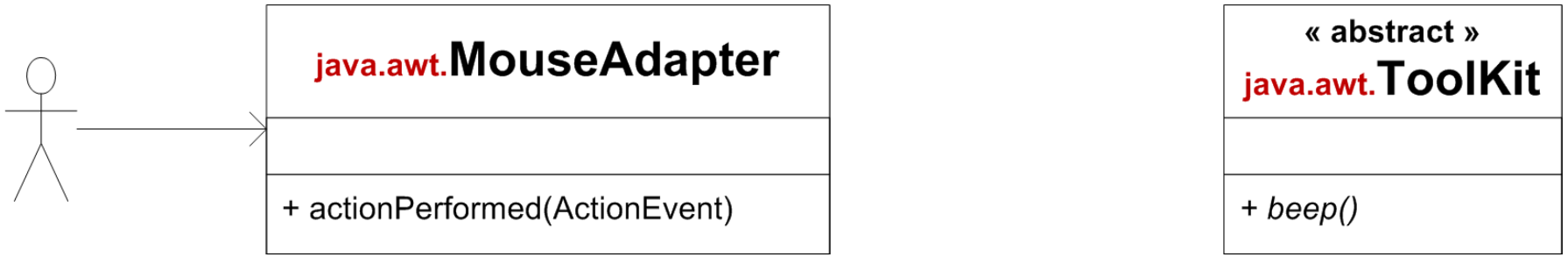


http://fr.wikipedia.org/wiki/Fichier:PatternAdapter.png

Implémentation

Adaptation de classe

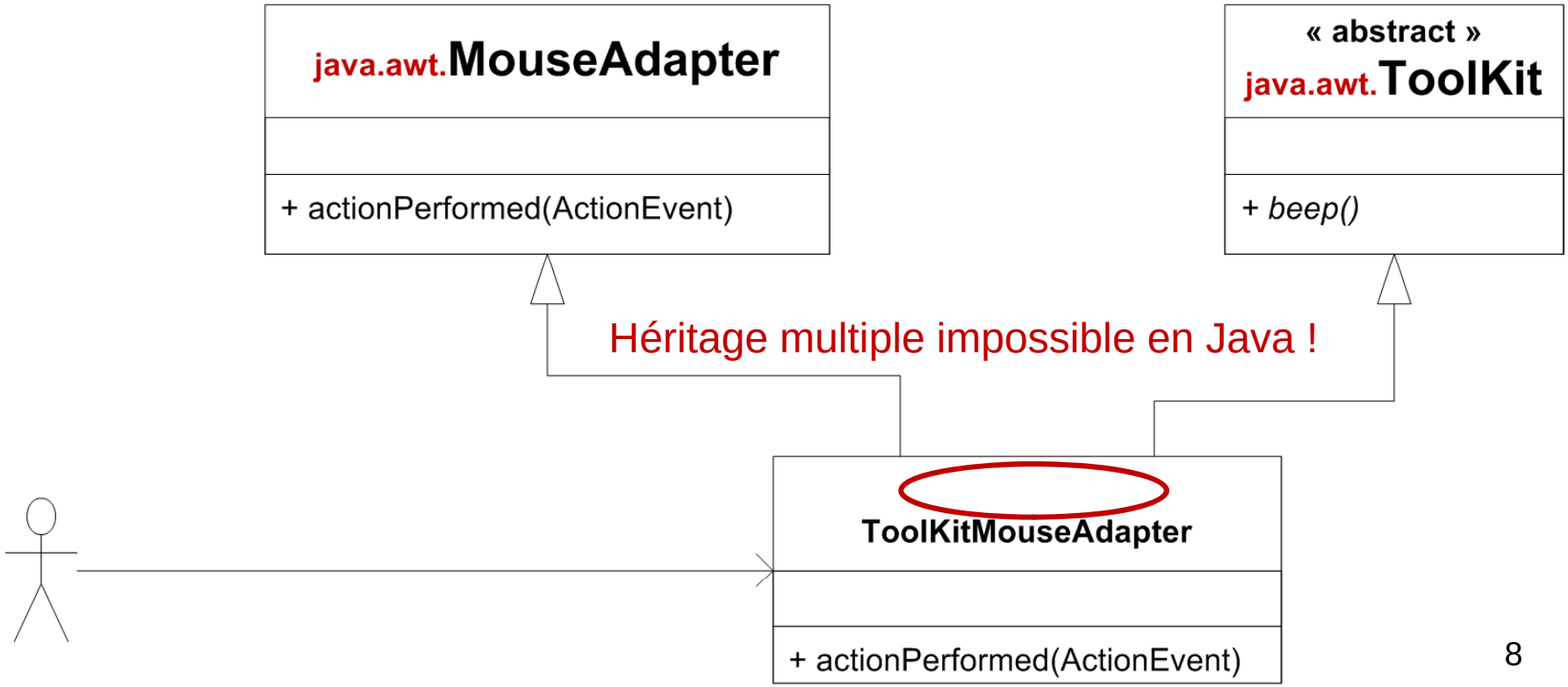
- **Par héritage (multiple)** de la classe à adapter
- en y ajoutant les méthodes de l'interface cible et en assurant les appels adéquats aux méthodes de la classe à adapter



Implémentation

Adaptation de classe

- **Par héritage (multiple)** de la classe à adapter
- en y ajoutant les méthodes de l'interface cible et en assurant les appels adéquats aux méthodes de la classe à adapter

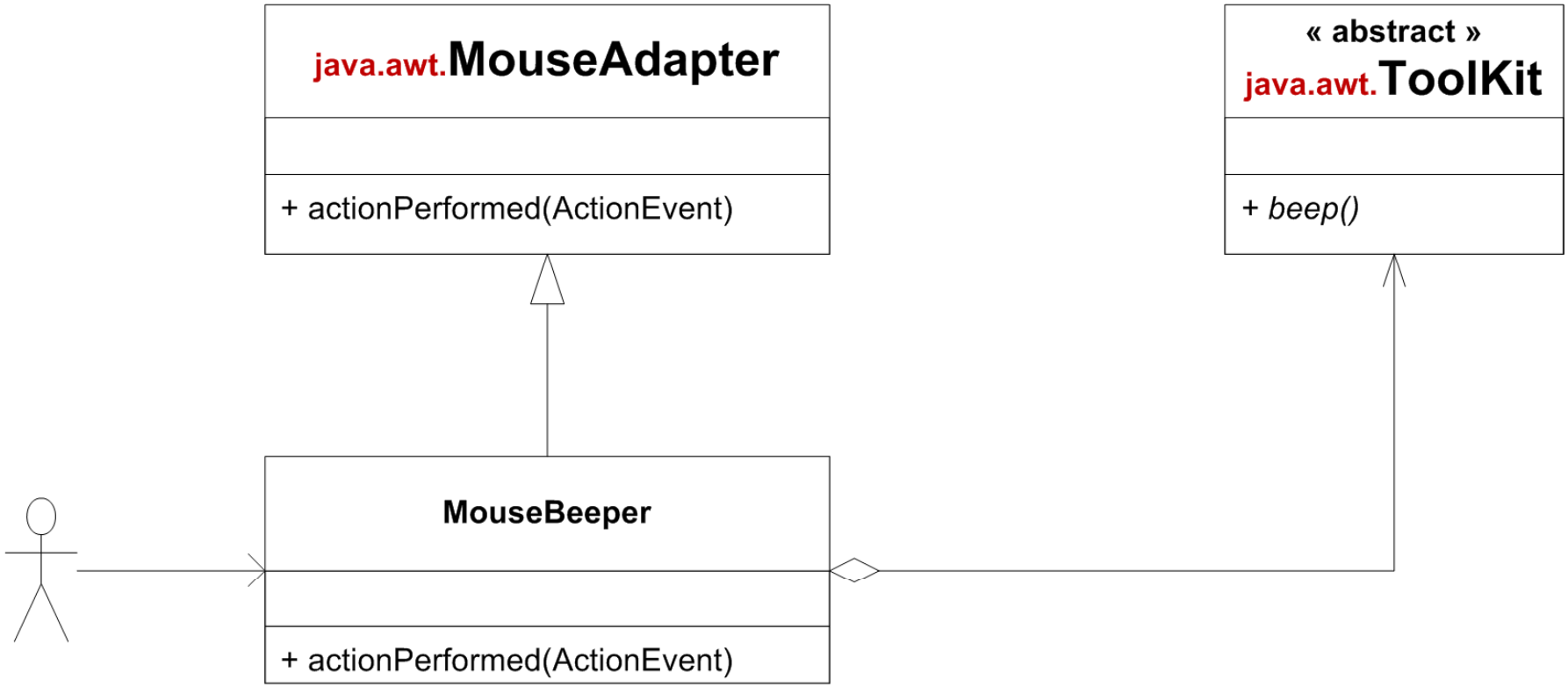


Université Nice Sophia Antipolis

Implémentation

Adaptation d'objet

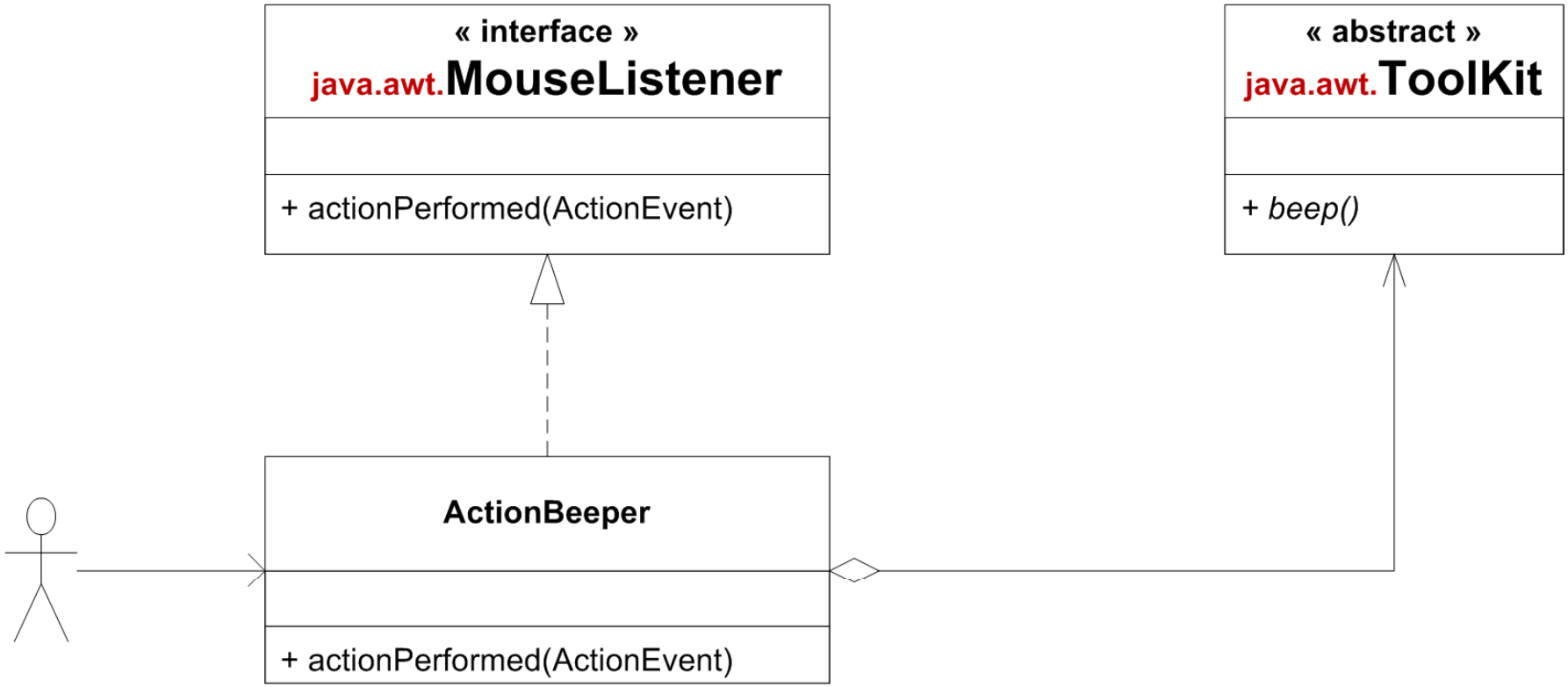
- **Par composition**
- faire correspondre les appels de méthode
- la classe à adapter est un champ de la classe qui adapte



Implémentation

Adaptation d'objet

- **Par composition**
- faire correspondre les appels de méthode
- la classe à adapter est un champ de la classe qui adapte



Université Nice Sophia Antipolis

Exemple

- Adapté `Toolkit.getDefaultToolkit().beep()` à une action de la souris, un clic de bouton, une fin de Timer, …

```
class MouseBeeper extends MouseAdapter {
  public void mouseClicked(MouseEvent e) {
     Toolkit.getDefaultToolkit().beep();
  }
}
class ActionBeeper implements ActionListener {
  public void actionPerformed(ActionEvent e) {
     Toolkit.getDefaultToolkit().beep();
  }
}
class TimerBeeper implements TimerListener {
  public void timerTriggered(TimerEvent e) {
     Toolkit.getDefaultToolkit().beep();
  }
}
```

Conséquences

- Pour la classe de l'objet qui adapte
  - Pas possible d'adapter une classe et ses sous-classes
  - Mais redéfinition possible du comportement (sous-classe)
- Pour l'objet qui adapte
  - Un **`Adaptateur`** peut travailler avec plusieurs Adaptées
  - Plus difficile de redéfinir le comportement d'Adaptée (sous-classer puis obliger **`Adaptateur`** à référencer la sous-classe)