Concepts avancés de la programmation orientée objet en Java

Branche SIT – Unité de valeur LO02

Guillaume Doyen

Contact : guillaume.doyen@utt.fr

Plan du cours

- **Les classes abstraites**
- **Les classes internes**
- **Les classes anonymes**
- **Les packages**

Les classes abstraites

- **Objectif**
  - Permettre à une classe de ne fournir qu'une implémentation partielle de ses méthodes
    - Certaines méthodes sont implantées
    - D'autres méthodes ne le sont pas, ce sont les méthodes abstraites
  - Les méthodes abstraites seront implantées dans des classes filles
    - Alternative à la redéfinition de méthodes
- **Principe**
  - Le modificateur **`abstract`**
    - Dans la déclaration d'une classe : indique que la classe est abstraite
    - Dans la déclaration d'une méthode : indique que la méthode est abstraite

```
public abstract class Vehicule {
    // Cette méthode est implantée
    public void setKilometres (int kilometres) { // ...  }
    // Cette méthode est abstraite
    public abstract void rouler ();
}
```

Classes abstraites et interfaces

- **Les interfaces**
  - Déclaration de méthodes implantées dans des classes
    - Définition implicite de méthodes abstraites, donc on ne mentionne pas le mot clé « abstract »
- **Classe abstraites ou interfaces ?**
  - Une classe abstraite permet de déclarer d'autres membres
  - Une classe abstraite fournit une implémentation partielle
  - Une interface représente une vue subjective d'une classe
  - Une classe peut hériter d'une seule classe abstraite mais implanter plusieurs interfaces
- **Classes abstraites et interfaces …**
  - Une classe qui n'implante pas toutes les méthodes d'une interface est abstraite
    - Elle délègue à ses classes filles l'implantation des méthodes restées abstraites

Plan du cours

- **Les classes abstraites**
- **Les classes imbriquées**
- **Les classes anonymes**
- **Les packages**

Les classes imbriquées

- **Objectif**
  - Permettre la définition de classes qui n'ont de sens que dans le contexte d'une autre classe
    - Regroupement logique des classes
    - Augmentation de l'encapsulation
    - Code plus facile à lire et à maintenir
- **Principe**
  - Une classe imbriquée est un membre d'une autre classe
    - Si le membre est statique, on parle de classe imbriquée statique
      - Usage rare
    - Si le membre n'est pas statique, on parle de classe interne
      - Usage fréquent
    - Les modificateurs de visibilité s'appliquent comme tout autre membre

Les classes internes

- **Principe**
  - Une classe interne est associée à une instance de la classe externe
  - Une classe interne ne peut pas définir de membres statiques
    - Car liée à une instance de la classe externe
  - En tant que membre de la classe externe, la classe interne a accès à tous les membres de la première

```
public class ClasseExterne {
    class ClasseInterne {
        // membres de la classe interne
    }
}
```

  - La création d'un objet de la classe interne fait référence à la classe externe

```
ClasseExterne refExterne = new ClasseExterne();
ClasseExterne.ClasseInterne refInterne = refExterne.new
    ClasseInterne();
```

Les classes imbriquées statiques

- **Principe**
  - Une classe imbriquée statique est associée à une classe externe
    - Elle ne peut donc pas accéder aux membres de la classe externe qui sont liés à une instance. Pour cela elle doit utiliser une référence comme tout autre classe.
  - Une classe imbriquée statique est finalement identique à une classe externe
    - Elle a été placée dans une autre classe pour des raisons d'organisation du code
  - Création d'une instance de classe imbriquée statique

```
ClasseExterne refExterne = new ClasseExterne();
ClasseExterne.ClasseInterne refInterne = new
    ClasseExterne.ClasseInterne();
```

Les classes locales

- **Principe**
  - Possibilité de déclarer une classe à l'intérieur d'une méthode pour un besoin particulier
    - Usage rare et montre une limite de la conception
    - Ne pas utiliser dans LO02… et par la suite

Les classes anonymes

- **Pourquoi ?**
  - Besoin ponctuel d'une instance d'une classe dont on va spécialiser le comportement
    - Besoin ponctuel : pas besoin de nommer la classe
    - Spécialisation du comportement : héritage
- **Principe**
  - Classe locale
  - Déclaration de la classe et instanciation en même temps
    - Extension de l'opérateur new
  - Utilisation principale : les contrôleurs dans les interfaces graphiques

```
Vehicule vehiculeDeClasseAnonyme = new Vehicule() {
    // Cette méthode est abstraite dans la classe Vehicule
    public  void rouler () {
        // code de la méthode dans la classe anonyme
    }
};
```