

Classes internes

Université Française d'Égypte

Version O 3.5.2 – 5/7/13

Richard Grin

2 types de classes internes

- Classes définies à l'extérieur de toute méthode
- Classes définies à l'intérieur d'une méthode

R. Grin

Classes internes

2

Classes internes non incluses dans une méthode

R. Grin

Classes internes

3

Emplacement du code

- A l'intérieur d'une classe **englobante**, au même niveau que les autres membres :

```
public class ClasseE {
    private int x;

    class ClasseI {
        . . .
    }

    public String m() { . . . }
    . . .
}
```

Code de la classe interne

R. Grin

Classes internes

4

Modificateurs

- Une telle classe peut être : **private**, **package**, **protected**, **public**
- Elle peut aussi être **abstract** ou **final**

R. Grin

Classes internes

5

Nommer une classe interne

- **ClasseE.ClasseI**

R. Grin

Classes internes

6

2 types de classes internes définies à l'extérieur d'une méthode

- Classes **static** : une instance n'est pas liée à une instance de la classe englobante
- Classes non **static** : une instance est liée à une instance de la classe englobante

R. Grin

Classes internes

7

Classes internes **static**

- A peu près le même rôle que les classes non internes
- Interne pour indiquer que la classe n'a de sens qu'en relation avec la classe externe

R. Grin

Classes internes

8

Visibilité pour les classes internes **static**

- Une classe interne **static** a accès à toutes les variables **static** de la classe englobante, même les variables **static private**
- La classe englobante a accès à tous les membres de la classe interne, qu'ils soient **static** ou non, même s'ils sont **private**

R. Grin

Classes internes

9

Classes internes non **static**

- Une instance d'une classe interne non **static** ne peut exister que liée à une instance de la classe englobante
- Le code de la classe interne peut désigner cette instance de la classe englobante par **ClasseE.this**

R. Grin

Classes internes

10

Visibilité pour les classes internes non **static**

- Une classe interne non **static** et la classe englobante partagent tous leurs membres (même privés)

R. Grin

Classes internes

11

Nommages particuliers liés aux classes internes

- Si un membre de **ClasseE** n'est pas caché, la classe interne peut le nommer simplement par son nom
- Si un membre est caché, elle le nomme en le préfixant par « **ClasseE.this** »

R. Grin

Classes internes

12

Création d'une instance d'une classe interne non **static**

- Dans le code d'une autre classe, une instance de **ClasseI** liée à **instanceClasseE** est créée par `instanceClasseE. new ClasseI(...)`

R. Grin

Classes internes

13

Exemple de classe interne non **static**

```
public class Tableau<T>
    implements Iterable<T> {
    private Object[] valeurs;
    private int nbElements;
    public Tableau(int n) {
        valeurs = new Object[n];
    }
    . . .
    public Iterator<T> iterator() {
        return new IterTableau();
    }
    private class IterTableau implements Iterator<T> {
        // Définition de la classe interne dans le transparent suivant
    }
}
```

Classe qui enveloppe un tableau

Elle fournit un itérateur pour parcourir ce tableau ; cet itérateur est associé à une instance particulière de **Tableau**

R. Grin

Classes internes

14

Définition de la classe **IterTableau**

```
private class IterTableau implements Iterator<T> {
    private int indiceCourant;

    public boolean hasNext() {
        return indiceCourant < nbElements;
    }

    public T next() {
        return (T)valeurs[indiceCourant++];
    }
}
```

Accès direct aux attributs privés de la classe englobante

R. Grin

Classes internes

15

Utilisation de **Tableau**

```
Tableau<Integer> t1 = new Tableau<Integer>(10),
t2 = new Tableau<Integer>(5);
// Ajout de quelques éléments dans t1 et t2
. . .
// Affichage des éléments de t1 et t2
for(int i : t1) {
    System.out.println(i);
}
for(int i : t2) {
    System.out.println(i);
}
```

R. Grin

Classes internes

16

Avantage des classes internes

- **IterTableau** peut accéder aux variables privées de la classe **Tableau** ⇒ la structure de **Tableau** est encapsulée
- Avec une classe externe, il aurait fallu ajouter une méthode **get(int i)** à **Tableau**, pour que **IterTableau** puisse obtenir le *i*^{ème} élément

R. Grin

Classes internes

17

Classe interne **static** ou non ?

- Le critère de choix est le suivant :
 - ne choisir non **static** que si la classe interne doit accéder à une variable d'instance de la classe englobante
 - sinon, choisir **static**

R. Grin

Classes internes

18

Classes internes locales (incluses dans une méthode)

R. Grin

Classes internes

19

Classes internes locales, définies à l'intérieur d'une méthode

- 2 types de telles classes :
 - classes avec un nom
 - classes anonymes

R. Grin

Classes internes

20

Utilisation des classes anonymes

- Une classe interne anonyme sert à (re)définir « à la volée » une ou plusieurs méthodes d'une classe (ou d'une interface)

R. Grin

Classes internes

21

Création d'une instance d'une classe interne anonyme

- Pas de constructeur puisque pas de nom
- La création fait appel au constructeur de la classe mère dont la signature correspond à **listeParamètres** :

```
Cercle c = new Cercle(p, r) {
    public void dessineToi(Graphics g) { ... }
};
```

Définition de la classe anonyme

R. Grin

Classes internes

22

Classe anonyme qui implémente une interface

- **ClasseMère** est remplacé par une interface qu'implémente la classe anonyme ; dans ce cas la liste des paramètres doit être vide :

```
new Interface() {
    // Définition de la classe
    . . .
}
```

- Par exemple,
`new Iterator<T>() { . . . }`

R. Grin

Classes internes

23

Exemple de classe anonyme

```
public class Tableau<T> implements Iterable<T> {
    . . .
    public Iterator<T> iterator() {
        return
            new Iterator<T>() {
                private int indiceCourant;

                public boolean hasNext() {
                    return indiceCourant < valeurs.length;
                }
                . . .
            };
    } // Fin méthode iterator()
}
```

Définition classe anonyme

R. Grin

Classes internes

24

Avantages des classes anonymes

- Si le code de la classe anonyme est court, elle améliore la lisibilité car le code de la classe est proche de l'endroit où il est utilisé

R. Grin

Classes internes

25

Inconvénients des classes anonymes

- Si le code est long, la classe anonyme va au contraire nuire à la lisibilité
- Pas possible de créer plusieurs instances de la classe

R. Grin

Classes internes

26

Contexte d'exécution des classes internes locales

- Les classes internes locales ont accès
 - aux variables d'instance et de classe de la classe englobante (même privées)
 - aux paramètres **final** et aux variables locales **final** de la méthode

R. Grin

Classes internes

27