Programmation Orienté Objets en JAVA

Daniele Varacca
Departement d'Informatique
Université Paris Est

2014

UPEC
UNIVERSITÉ PARIS-EST CRÉTEIL VAL DE MARNE
Connaissance - Action

Encore des classes

- Classes publiques chacune dans son fichier
- Classes avec visibilité de package, plusieurs dans un fichier

Mais peut-on avoir une classe private ?

Classes internes

On peut définir une classe *dans une autre classe*. Cela s'appelle un *classe interne*.

Une classe interne B est déclarée dans le corps d'une autre classe A

- parce que B n'est utilisé que par A
- pour des raisons d'encapsulation

Un classe interne peut être private ! Personne ne la voit, sauf la classe englobante.

UPEC
Connaissance - Action
UNIVERSITÉ PARIS-EST CRÉTEIL VAL DE MARNE

Classes internes

Une instance d'une classe interne a accès aux champs privés de la classe englobante.
Tout objet d'une classe interne connaît l'objet englobant.

```
class A {
    private String s;
    class X {
        void f () {
            A.this.s="Pippo";
        }
    }

}
```

Classes internes

Si la classe interne est private on peut en créer une instance seulement dans un objet englobant.
Si la classe interne n'est pas private, on peut en créer une instance en dehors de l'objet englobant, mais il faut tout de même que cet objet existe

```
class A {
    class X {
    }
}

A a = new A();
A.X obj = a.new X();
```

Classes internes

Utilisation typique: structures des données.

```
class MyList<T> {
    private class Element {
        T value;
        Element next;
    }
    private Element first;
    public void add (T value) {
        Element e = new Element();
        e.value=value;
        e.next = first;
        first = e;
    }
}
```

Autres modificateurs

Une classe être aussi

- static (peu utilisé)
- protected (très peu utilisé)
- final : cela veut dire qu'on ne peut pas hériter d'elle.

Classe anonyme

Une classe anonyme, est une classe qui n'a pas de nom. Elle est créée à la volée, soit en héritant d'une autre classe, soit en implémentant une interface

```
interface A {
   int f (int x);
}

A obj = new A() {
   public int f (int x) {
      return x+3;
   }
}
```

Classe anonyme

Déjà vu! La classe existe pour la JVM mais on ne peut pas l'appeler dans le code.

- Utile pour redéfinir quelques méthodes ou pour implémenter des petites interfaces
- On ne peut pas implémenter plusieurs interfaces

Classe locale

Pour cela on peut aussi créer des classes locales à une méthode (ou au main)

```
interface A {
   int f (int x);
}
interface B {
   int g (int x);
}
...
class X implements A,B {
   public int f (int x) {
                  return x+3;
   }
   public int g (int x) {
                  return x+5;
   }
}
```

Enumération

Un nombre fini de valeurs distinctes

```
public enum Day {
    SUNDAY, MONDAY, TUESDAY, WEDNESDAY,
    THURSDAY, FRIDAY, SATURDAY
}
```

Enumération

```
Day day;
    switch (day) {
    case MONDAY:
    System.out.println("Mondays are bad.");
    break;
    case FRIDAY:
    System.out.println("Fridays are better.");
    break;
    default:
    System.out.println("The rest is ok");
    break;
}
```

Énumération

Les énumérations sont des classes, qui héritent de `java.lang.Enum`.
Elles permettent d'écrire des boucle for:

```
for (Day d : Day.values()) {
    System.out.println("I do not like " + d);
}
```

Énumération

Elle peuvent contenir des méthodes, des champs, des constructeurs.

```
public enum Couronne {
    PARIS (75),
    SSD(93),
    VDM(94),
    HDS (92);

    private final int dept;
    Couronne(int dept) {
        this.dept = dept;
    }
    public int carPlate() { return dept; }
}
```

(on ne peut pas appeler le constructeur)

Énumération

```
for (Couronne c : Couronne.values()) {
    System.out.println(
        "The cars from " + c +
        " have number "+
         c.carPlate()
    );
}
```