Programmation Orienté Objets en JAVA

Daniele Varacca
Departement d'Informatique
Université Paris Est

2014

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

La formule magique

public static void main (String [] args)
Aujourd'hui on expliquera les deux mots clés

- public
- static

Encapsulation

Révéler tout l'état interne d'un objet

- révéler des détails d'implémentation
- rendre plus difficile de changer l'implémentation
- avoir moins de contrôle sur comment on utilise un objet

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

Encapsulation

On veut contrôler l'accès aux objets
Tout ce qui concerne l'implémentation doit être caché
Mot clé private

- private int x
- private int f(int x) {...

Rules for private

Un champ ou une méthode déclarée private :

- les objets des autres classes n'ont pas accès
- les objets des sous-classes n'ont pas accès non plus!
- les *autres* objets de la *même* classe ont accès

Rules for public

Le contraire de private c'est public:

- tout le monde a accès aux champs et aux méthodes déclarés public

Classes public

Quand une classe A est déclarée public

- elle doit être dans un fichier séparé
- le fichier doit s'appeler `A.java`

Bonne pratique

Règles générales de bonne conduite:

- tous les champs sont déclarés private
- Si on veut lire le contenu d'un champ on peut créer un *getter* publique
- Le getter pour le champ Type toto s'appelle Type getToto()
- Si on veut aussi modifier le contenu d'un champ on peut créer un *setter* publique
- Le setter pour le champ Type toto s'appelle void setToto(Type t)

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

Bonne pratique

Règles générales de bonne conduite:

- les méthodes qui servent sont private
- les méthodes qu'on veut exposer pour l'extérieur sont publiques
- ce sont les méthodes publiques qui définissent la signature d'une classe

Entre le deux

Il existe un troisième mot clé de visibilité: protected

- les méthodes sont visibles à toute sous-classe
- cela permet d'utiliser super

Exemple

```
public class A{
  protected void f() {
    ...operations utilitaires...
  }
}
public class B extends A {
  public void g () {
      ...
      super.f()
  }
}
```

- A utiliser avec moderation
- Un cas spécial: clone()

Visibilité par défaut

Ehm, moi je n'ai jamais écrit public, mais ça marche quand même...

- La visibilité *par défaut* est *presque* comme public.
- La visibilité par défaut est comme publique à l'intérieur d'un *package*.

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

Packages

Façon de structurer le code.

- On déclare un package avec
  package toto;
- Et on l'utilise avec
  import toto .*;
- Des règles précises pour les fichiers:
  le package toto . fifi .doudou
  doit être en `/toto/fifi/doudou`

Eclispe gère cet aspect tout seul, mais attention!

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

Champs statiques

L'autre mot clé static :

- un champ statique n'appartient pas à un objet
- on peut dire qu'il est partagé par tous les objets d'une classe

```
public class A {
    public static int x=0;
}
...
A.x = 24;
```

Champs statiques

```
public class A {
   public static int instances=0;

   public A() {
      instances++;
   }
}

new A();
new A();
print(A.instances) // affiche 2
```

Champs statiques

On a vu déjà un champ statique:

```
System.out.println("Hello World");
```

- out est un champ statique de la classe System.
- il est de type PrintStream
- PrintStream contient une méthode void println (String x)

Champs statiques finaux

Un cas typique:

```
public class A {
   public static final int REPONSE=42;
}
```

- mot clé final : la variable ne peut pas être réaffectée
- il s'agit de la définition des constantes utiles au fonctionnement des objets de la classe

Méthodes statiques

Une méthode statique n'appartient pas à un objet

- elle s'appelle en utilisant le nom de la classe
- elle ne peut qu'utiliser de champs statiques
- (sauf si elle crée des objets elle-même)
- elle ne peut pas être redéfinie par une sousclasse

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

Méthodes statiques

Une méthode statique est appelée avec le nom de sa *classe*

```
public class A {
   public static int add2 (int x) {
        return x+2;
   }
}
...
int y = A.f(3);
```

- pas de polymorphisme
- le code à exécuter est décidé à la compilation
- (le compilateur peut optimiser le code)

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

Méthodes statiques

Utilisation des méthodes statiques

- pour des taches utilitaires (calculs mathématiques, algorithmes). Exemple Math.sqrt,
- pour la création d'objets: factory.

Factory

Créer des objets sans utiliser explicitement new:

```
public class A {
   public static A create () {
        return new A();
   }
}
A obj = A.create();
```

C'est quoi l'intérêt?

Factory

Changer l'implémentation par la suite sans changer trop de code

```
public class B extends A{
  ... meilleure implémentation ...
}
public class A {
   public static A create () {
        return new B();
   }
}
```

Factory

Créer un objet différent selon les paramètres:

```
public class B extends A{...}
public class C extends A{...}
public class A {
   public static A create (int x) {
        if (x==0) return new B();
        else return new C();
   }
}
```

Singleton

Un seul objet d'une certaine classe doit exister.

```
public class Single {
    private static Single instance;
    private Single() { }
    public static Single getInstance() {
        if (instance == null) {
            instance = new Single();
        }
        return instance;
    }
}
```

Le constructeur est déclaré private!

Autres sujets

- JVM
- La méthode equals

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