UPMC
SORBONNE UNIVERSITÉS

SORBONNE
UNIVERSITÉS

Rappels, classes et héritage en Java

F. Kordon - Université P. & M. Curie - CC2016

Fabrice.Kordon@lip6.fr

LIP6

Classe et Objet?

Besoin d'origine, l'encapsulation

- Éléments publics, éléments privés (ou protégés)
- Structuration en graphe d'une application (opposition aux «couches»)
- Méthodes «de classe» ou «d'instance»

Comment penser

- Services offerts
- Données requises pour les implémenter

Héritage

- Enrichir ou modifier des objets
  - **Animal -> Mammifère -> Humain**
- Notion de «super-classe»
- Notion de hiérarchie de classe

Structure d'une classe

Trois parties (une par type de «membres»)

Attributs

- Déclaration des attributs
- publics, privés, protégés

Constructeurs

- Permettre l'élaboration de l'objet
- Les constructeurs peuvent s'appeler entre eux
- Il existe un constructeur par défaut

Méthodes

- Manipulation de la classe
- Services rendus aux autres classes «utilisatrices»

Visibilité (méthodes/attributs)

Entités «public», «protected» ou «private»

Structure d'une classe

Trois parties (une par type de «membres»)

Attributs

- Déclaration des attributs
- publics, privés, protégés

Constructeurs

- Permettre l'élaboration de l'objet
- Les constructeurs peuvent s'appeler entre eux
- Il existe un constructeur par défaut

Méthodes

- Manipulation de la classe
- Services rendus aux autres classes

Visibilité (méthodes/attributs)

Entités «public», «protected» ou «private»

toString = standard

Il est pratique de la surcharger

Étudiants, le retour


```java
public class Etudiants2 {

    // Attributs
    public String nom;
    public String prenom;
    private int numero;

    // Constructeur
    Etudiants2 (String n, String p) {
        nom = n;
        prenom = p;
    }

    // Méthodes
    public void setNumero (int n) { // nom normalisé
        numero = n;
    }

    public String toString ()  { // bonne pratique
        return nom + " " + prenom + " (" + numero + ")";
    }
}
```

Objectif de l'extension

Associer un binôme à l'étudiant

Étudiants et binômes


```java
public class EtudiantsBinomes extends Etudiants2 {

    // Attributs
    private Etudiants2 binome;

    // Constructeurs
    EtudiantsBinomes (String n, String p) {
        super(n, p);
    }

    EtudiantsBinomes (String n, String p, EtudiantsBinomes b) {
        super(n, p);
        binome = b;
    }

    // Méthodes
    public void setBinome (EtudiantsBinomes e) {
        binome = e;
    }

    public String toString ()  {
        if (binome == null) {
            return nom + " " + prenom + " (solitaire)";
        } else {
            return nom + " " + prenom + " avec " + binome.prenom + " " + binome.nom;
        }
    }
}
```

Tester les étudiants binômés


```java
public class TestEtudiantsBinomes {

    public static void main(String []args) {
        EtudiantsBinomes e1, e2, e3;

        // Créer deux étudiants
        e1 = new EtudiantsBinomes ("Jean", "d'Arme");
        e2 = new EtudiantsBinomes ("Paul", "Icier", e1);
        e3 = new EtudiantsBinomes ("Gaston", "Lagaffe");

        e1.setNumero (1);
        e2.setNumero (2);

        e1.setBinome (e2);

        // Afficher le résultat
        System.out.println ("Voici des binômes :");
        System.out.println ("  " + e1);
        System.out.println ("  " + e3);
    }

}
```

```
$ java TestEtudiantsBinomes
Voici des binômes :
  Jean d'Arme avec Icier Paul
  Gaston Lagaffe (solitaire)
```

En guise de conclusion…

L'héritage permet d'étendre une classe

- Modifier son comportement
  - **Constructeurs et méthodes**
- Rajouter des attributs et des méthodes

Il ne permet pas en pratique

- De supprimer des méthodes ou des attributs
  - **Possible de surcharger des méthodes avec un code «vide»**

Possibilité de définir des «généalogies complexes»

- Les utilitaires autour de Java ou de C++

Remarque : pas d'héritage multiple en Java