UPMC
SORBONNE UNIVERSITÉS

SORBONNE
UNIVERSITÉS

Rappels, interfaces et classes abstraites en Java

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

Fabrice.Kordon@lip6.fr

LIP6

Vous avez dit «interface»?

Contrat pour le/les objet(s) qui l'implémentent

- Pas de constructeur
- Juste des méthodes (implicitement abstraites)
- Seulement des attributs **statiques**

- Très utile pour les API
- Une classe peut implémenter plusieurs interfaces
  - **Rappel, elle ne peut hériter que d'une seule classe**

Exemple

```
public interface Vehicules {

    void demarrer();
    void arreter();
}
```

Première implémentation de l'interface : Velo


```java
public class Velo implements Vehicules {

    boolean deCourse;

    public Velo(boolean c) { // Le constructeur
        this.deCourse = c;
    }

    public void demarrer() {
        if (deCourse) {
            System.out.println("Le vélo de course démarre");
        } else {
            System.out.println("Le vélo démarre");
        }
    }

    public void arreter() {
        if (deCourse) {
            System.out.println("Le vélo de course s'arrête");
        } else {
            System.out.println("Le vélo s'arrête");
        }
    }
    //Autres méthodes propres à Velo
}
```

```java
public interface Vehicules {

    void demarrer();
    void arreter();
}
```

Seconde implémentation de l'interface : Voiture


```java
public class Voiture implements Vehicules {

    String couleur;

    public Voiture(String c) { // Le constructeur
        this.couleur = c;
    }

    public void demarrer() {
        System.out.println("La voiture " + couleur + " démarre");
    }

    public void arreter() {
        System.out.println("La voiture " + couleur + " s'arrête");
    }
    //Autres méthodes propres à Voiture
}
```

Important

La concrétisation s'utilise comme une classe!

Tester Velo et Voitures


```java
public class TestVehicules {
    public static void main(String []args) {
        Velo v1, v2;
        Voiture a1, a2;

        v1 = new Velo (true);
        v2 = new Velo (false);

        a1 = new Voiture ("Jaune");
        a2 = new Voiture ("Rosso ALFA");

        v1.demarrer();
        v2.demarrer();
        a1.demarrer();
        a2.demarrer();
        v1.arreter();
        v2.arreter();
        a1.arreter();
        a2.arreter();
    }
}
```

```
$ java TestVehicules
Le vélo de course démarre
Le vélo démarre
La voiture Jaune démarre
La voiture Rosso ALFA démarre
Le vélo de course s'arrête
Le vélo s'arrête
La voiture Jaune s’arrête
La voiture Rosso ALFA s’arrête
```

Et les classes abstraites?

Entre les classes et les interfaces

- Permettre une implémentation partielle
- Empêcher une utilisation «telle que»
- Imposer une implémentation de classe parente

Exemple

```
public abstract class AbstractMoteur {
    public boolean arrete;
    public String marque;
    public int numSerie;

    // Constructeur
    AbstractMoteur (String m, int s) {
        marque = m;
        numSerie = s;
        arrete = true;
    }

    // Méthodes
    public void allumer () {
        arrete = false;
    }

    public void eteindre () {
        arrete = true;
    }

    public abstract void reparer ();
}
```

En guise de conclusion

Ensemble de mécanismes très utile

- Permet une sorte d'héritage multiple

Usage intensif dans les «utilitaires»

- Valable en Java
- Valable en C++
- et aussi d'autres langages

Un outil à maîtriser