Interfaces

Un autre point de vue : les interfaces

- Plutôt qu'hériter classe abstr. : **implanter** une **interface**
- Rien à voir avec les interfaces graphiques
- « Etre positionnable »
  - Propriété *transverse* à d'autres objets : formes, images, icônes, etc.
  - Pas forcément implantée par toutes les formes

Une **interface** est un ensemble d'en-têtes de méthodes constituant une API, et assurant aux objets qui l'implantent la faculté d'être manipulables conformément à la propriété qu'ils annoncent posséder.

```
public interface Positionnable {
    public void setPosition(Point2D position);
    public Point2D getPosition();
}
```

Définition de type (comme une classe)

Définit uniquement une API, sans l'implanter

- On n'hérite pas d'une interface, on l'implante (**implements**)

Toute classe qui implante l'interface a l'obligation d'en implanter toutes les méthodes.

Interfaces : règles

Les règles suivantes s'appliquent :

— Une interface ne contient aucune implantation. Un point-virgule ' ;' remplace le corps de la méthode.
— Toutes les méthodes d'une interface sont implicitement abstraites, même si le modificateur **abstract** (ou **abstraite**) est omis.
— Toutes les méthodes d'une interface doivent être d'instance (⇒ pas de méthode statique dans une interface).
— Toutes les méthodes d'une interface sont implicitement publiques, même si le modificateur **public** est omis.
— Une interface n'a pas le droit de définir d'attributs, à l'exception de constantes déclarées à la fois **static** et **final**.
— Une interface n'a pas de constructeur.

Extrait de « Java in a Nutshell » par David Flanagan, 4ème édition en français (O'Reilly)

A retenir.
- Aucune implantation, même partielle
- On peut implanter plusieurs interfaces à la fois
- Implanter une interface engage à implanter toutes ses méthodes

Interfaces : utilité et utilisation

L'algo repose sur l'invocation des méthodes de positionnement : présence garantie grâce au **typage par l'interface** des paramètres.

- **Implanter une interface, c'est annoncer une propriété**
  - **Exemple : « Je suis positionnable »**
- **Rend l'objet disponible pour un algorithme pré-défini**
  - **L'algorithme n'a pas besoin de connaître l'objet qu'il manipule**
  - **Il sait le manipuler via les méthodes annoncées dans l'interface**

```
public void swapPositions(Positionnable o1, Positionnable o2) {
    Point2D tmp = o1.getPosition();
    o1.setPosition(o2.getPosition());
    o2.setPosition(tmp);
}
```

Tous les objets implantant Positionnable sont acceptés, et rien qu'eux

Ça marche avec n'importe quel objet, il suffit qu'il soit Positionnable

- **Exemple dans l'API java**

Tous les objets implantant l'interface java Comparable (voir API) peuvent être triés par la méthode standard de tri java sort().

```
public interface Comparable {
    public int compareTo(Object o);
}
```

Méthode de comparaison, à redéfinir par chaque objet

Trier les formes par aire croissante

Interface : on peut en implanter plusieurs

- Contrairement à l'héritage, qui n'est pas multiple …
- On peut posséder plusieurs propriétés
  - On implante alors plusieurs interfaces
  - Syntaxe : implements Interface1, Interface2, …, InterfaceN;
- Exemple
  - On peut être « représentable » (ou montrable) : Showable
  - Méthode pour montrer l'objet : Image getRepresentation()

```
public interface Showable {
    public Image getRepresentation();
}
```

  - Un Rectangle est à la fois positonnable *et* représentable
  - Un Humain est représentable, mais pas positionnable

```
public class Rectangle extends Forme implements
Positionnable, Showable {
   public double aire() {…}
   public double perimetre() {…}
   public Image getRepresentation() {…}
   public void setPosition(Point2D pos) {…}
   public Point2D getPosition() {…}
}
```

```
public class Humain extends Hominidé implements
Showable {
    public Image getRepresentation() {…}
}
```

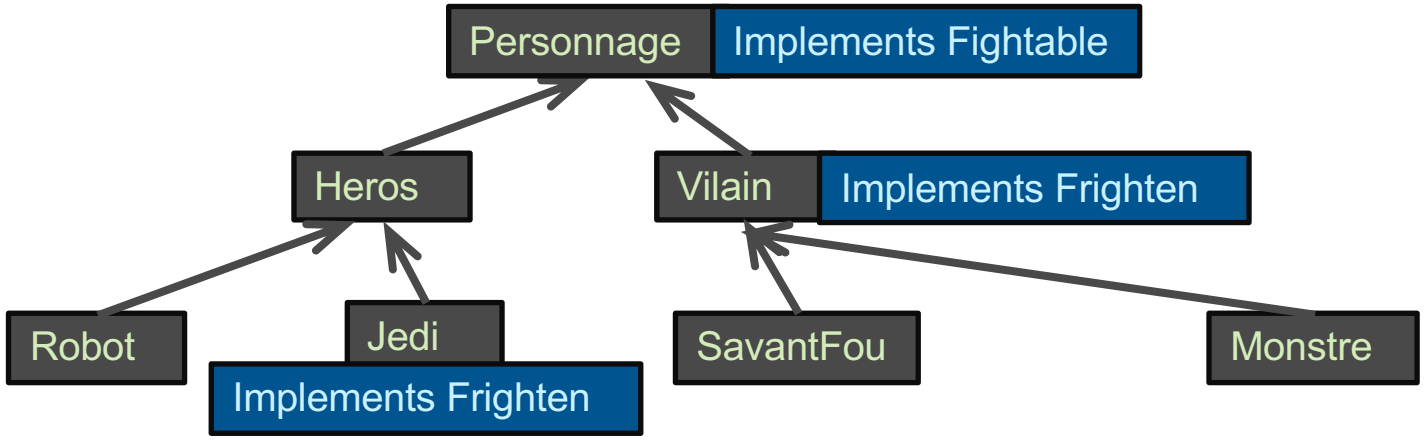
Montrer un Rectangle Positionnable et Comparable

Classe abstraite ou interface ? Comment choisir ?

- Héritage = relation « est un(e) »
  - Ex : un rectangle est une forme
  - A réserver : un objet en *spécialise* un autre, plus général
  - Méthodes abstraites
    - Celles que doivent posséder toutes les sous-classes,
    - mais qu'on ne sait pas implanter au niveau de la super-classe
    - Ex : aire d'une forme, mouvement d'une pièce d'échec, …
- Interface = « être capable de se comporter comme »
  - Indépendamment de ce qu'on « est » (i.e. de qui on hérite)
  - On peut implanter plusieurs types de comportement
  - Ex : être comparable, être positionnable, être cliquable (IHM)
  - Un programme saura manipuler n'importe quel objet s'il connaît l'interface qu'il implante
  - Ex : trier une liste d'objets Comparable : on les compare par la méthode dédiée, annoncée par l'interface

Exemple des Personnages

- Hiérarchie de clase des personnages



- Différentes capacités
  - Se battre : tout le monde — Interface Fightable
  - Effrayer: les vilains — Interface Frighten
  - Maîtriser la force : les Jedi — Interface Force