06/10/2025

# POO : Polymorphisme

## Les interfaces

### Syntaxe de l'interface

Une interface est une manière de rassembler des classes selon certaines contraintes. La syntaxe est :

```java
public interface Recyclable {
  public void recycle();
   ...
}
```

> ⚠ Warning
>
> une interface n'est pas une classe, ça n'a riieeeeeeeeeeeeeeeeeeeeen à voir

Les interfaces ne prennent que des signatures de méthodes, pas de corps

### Implémentation de l'interface

Une classe peut implémenter une interface. La classe implémentant l'interface doit contenir toutes les méthodes de l'interface, et doit les implémenter

Par exemple, on peut avoir :

```java
public class Paper implements Recyclable {
  public class a() {}

  public class recycle() {}
}
```

> ⚠ Important
>
> Une interface peut être implémentée par plusieurs classes Réciproquement, une classe peut implémenter plusieurs interfaces, séparée par des virgules

> Tip
>
> On peut ajouter du code sur les interfaces, comme suit :

```java
public interface I {
  public double f();
  public double g();
  public default double m() {
    return this.f() + this.g() // N'appelle que les méthodes de I
  }
}
```

Cas d'usage assez spécifique, et nous ne sommes alors pas obligés de définir `m` dans les classes, mais elle peut :)

## Utilisation des interfaces

Pour résoudre le problème du cours, on peut utiliser l'interface comme ceci :

```java
// Ceci est licite
Paper p = new Paper();
Recyclable p2 = new Paper();
```

Les références définissent les méthodes autorisées, pas l'objet

Dans ce cas suivant :

```java
Paper p = new Paper();
p.tear(); //ok
p.recycle(); //ok

Recyclable r = p;
r.tear(); // pas ok
```

Une interface est également un type

## Différence avec une classe

Une interface est un type abstrait : elle n'a aucun sens sans contexte. L'interface est définie par les classes qui les implemente

Une interface n'a ni constructeur, ni attribut et ni corps de méthode

# Fonctionnement des interfaces

## Early binding vs Late binding

En C, le compilateur génère un appel à une fonction en particulier, et cela à la compilation : early binding
En Java, l'objet qui appelle la méthode exécutée n'est déterminé qu'au runtime : late binding.

dans le cas du late binding, le compilateur ne regarde que si les appels sont cohérents

Le late binding est un mécanisme fondamental de la POO

$\alpha, \beta$

## Exemple (voir les transparents de cours)