

# Interfaces/langage en Java

# Motivation

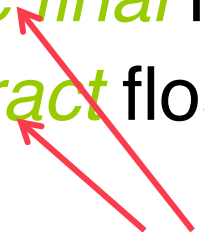
- Types **plus abstraits** que les classes
  - plus réutilisables
- Technique pour **masquer l'implémentation**
  - découplage public/privé : type/implémentation
- Favorise l'écriture de **code plus général**
  - écrit sur des types plus abstraits
- Relations de **spécialisation multiple**
  - entre les interfaces
  - entre les classes et les interfaces
  - Meilleure organisation des types

# Définition

- Interface
  - méthodes d'instances publiques et abstraites
    - public abstract
  - variables de classes constantes et publiques
    - public final static

# Syntaxe

```
public interface Iquadrilatre {  
    public static final int nbCotes = 4;  
    public abstract float perimetre();  
}
```



obligatoires ...*peuvent être omis*

Ecriture plus courante :

```
public interface Iquadrilatre {  
    int nbCotes = 4;  
    float perimetre();  
}
```

# Syntaxe

```
public interface Iquadrilatre {  
    int nbCotes = 4;  
    float perimetre();  
}
```

Remarque : pas de constructeur

- ce n'est pas un oubli !
- Il n'y aura pas de constructeur par défaut généré

# Spécialisation (*extends*)

```
interface Irectangle extends Iquadrilatere
{
    float angle = 90;
    float angle();
    float largeur();
    float hauteur();
}
```

# Implémentation - classe concrète

toutes les opérations sont implémentées

```
interface Iquadrilatere{...}
```

```
interface Irectangle extends Iquadrilatere{...}
```

```
public class Rectangle implements Irectangle {  
    private float largeur, hauteur;  
    public Rectangle(){  
    public Rectangle(float l, float h){largeur=l;hauteur=h;}  
    public float perimetre(){return 2*largeur()+2*hauteur();}  
    public float angle(){return Irectangle.angle;}  
    public float largeur(){return largeur;}  
    public float hauteur(){return hauteur;}  
}
```