## Méthodes statiques

Similairement, si l'on ajoute `static` à une méthode on peut alors y accéder *sans aucun* objet

```
class A {
   static void methode1() {
      System.out.println("Méthode 1");
   }
   void methode2() {
      System.out.println("Méthode 2");
   }
}
class ExempleMethodeStatique {
   public static void main(String[] args) {
      A.methode1(); // OK
      A.methode2(); // Non !
      A v = new A();
      v.methode1(); // OK, alternative
      v.methode2(); // OK (comme d'habitude)
   }
}
```

## Restrictions sur les méthodes statiques

Puisqu'une méthode statique peut être appelée avec ou sans objet :

- Le compilateur ne peut pas être sûr que l'objet `this` existe pendant l'exécution de la méthode
- Il ne peut donc pas admettre l'accès aux variables/méthodes d'instance (car elles dépendent de `this`)

Conclusion pour les accès dans la même classe :

- Une méthode statique peut *seulement* accéder à d'autres méthodes statiques et à des variables statiques

## Restrictions sur les méthodes statiques (2)

```
class A {
   int i;
   static int j;
   void methode1() {
      System.out.println(i); // OK
      System.out.println(j); // OK
      methode2();            // OK
   }
   static void methode2() {
      System.out.println(i); // Faux
      System.out.println(j); // OK
      methode1();            // Faux
      methode2();            // OK (sauf recursion infinie)
      A v = new A();
      v.methode1();          // OK
   }
}
```

## Utilité des méthodes statiques

☞ Méthodes qui ne sont pas liées à un objet

Exemple :

- Classe mettant à disposition des utilitaires mathématiques divers
- La création d'un objet de type `MathUtils` est artificielle
- La classe sert seulement à stocker des méthodes utilitaires

```
class MathUtils {
   public final static double PI = 3.14159265358979323846;
   public static double auCube(double d) {
      return d*d*d;
   }
}
```

## Utilité des méthodes statiques (2)

Utilisation de la classe `MathUtils` :

- Calculer $y = \pi \cdot x^3$ pour $x = 5.7$;
- On peut accéder aux variables/méthodes statiques sans construire d'objet

```
class Calcul {
   public static void main(String[] args) {
      double x = 5.7;
      double y = MathUtils.PI * MathUtils.auCube(x);
      System.out.println(y);
   }
}
```

## Méthodes et variables statiques

Eviter la prolifération de `static` !
On l'utilise seulement dans des situations très particulières :

- définition d'une constante : attribut `final static`
  (situation très courante)
- utilisation d'une valeur commune : attribut `static`
  (plus rare)
- méthodes utilitaires qu'il est artificiel de lier à un objet :
  méthode `static`, *invocable sans objet*
  (plus rare aussi)

Exemples de méthodes statiques :

- `Math.sqrt`
- la méthode `main`

## Méthodes auxiliaires de `main`

Nous comprennons maintenant pourquoi les méthodes auxiliaires de la méthode `main` sont statiques
(mais pas les méthodes dans les classes)

La méthode `main` a un en-tête fixe :

```
public static void main(String[] args)
```

Puisque la méthode `main` est obligatoirement statique :

- elle ne peut pas accéder à l'objet `this`
- elle ne peut pas accéder à des variables/méthodes d'instance
- elle peut seulement accéder à des variables/méthodes statiques

En dehors de cela, la classe de la méthode `main` est comme n'importe quelle classe.

Elle peut avoir des constructeurs, des méthodes et des variables