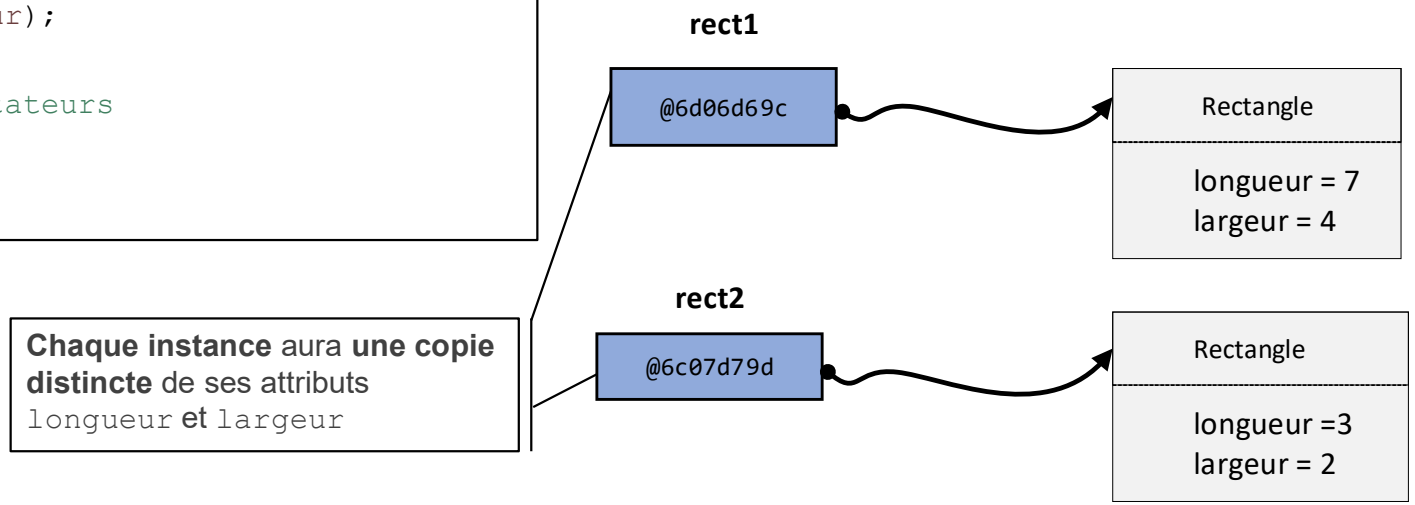
Les attributs et méthodes statiques

Rappel: attribut ou champ d'instance

- On a vu dans le cours précédent que chaque nouvel objet initialisé aura une copie distincte de ses attributs (ou variables d'instance).
- Exemple: Si on crée 2 objets de la classe `Rectangle rect1` et `rect2` comme ici-bas, alors `rect1` aura comme longueur 7 et comme largeur 4, alors que `rect2` aura comme longueur 3 et comme largeur 2.

```
public class Rectangle {
   // Attributs(ou variables d'instances)
   private int longueur;
   private int largeur;

   public Rectangle(int longueur, int largeur) {
      setLongueur(longueur);
      setLargeur(largeur);
}
  // Accesseurs et mutateurs
}
```

```
public static void main(String[] args) {
   Rectangle rect1, rect2;
   rect1 = new Rectangle(7, 4);
   rect2 = new Rectangle(3, 2);
}
```



Attribut statique (ou variable de classe)

- Si on veut partager un attribut entre toutes les instances d'une classe, on doit le définir comme statique à l'aide du mot clé `static`. **Une copie unique de cet attribut sera alors créée et partagée entre toutes les instances de cette classe**.
- Exemple: Supposons que nous voulions créer une variable qui va compter le nombre d'objets de type Rectangle qui ont été initialisés. Cette variable devra être partagée entre toutes les instances de Rectangle, permettant ainsi à chacune d'y accéder et de l'incrémenter lors de son initialisation.

```
public class Rectangle {
   // Attribut statique ou variable de classe
   public static int compteurRectangle = 0;

   // Attributs(ou champs d'instance)
   private int longueur;
   private int largeur;

   // Constructeur
   public Rectangle(int longueur, int largeur) {
      setLongueur(longueur);
      setLargeur(largeur);
      compteurRectangle ++; //une instance de plus
   }
   // Accesseurs et mutateurs
   …
 }
```

```
public static void main(String[] args) {
   Rectangle rect1, rect2;
   rect1 = new Rectangle(7, 4);
  //incrémente compteurRectangle
   rect2 = new Rectangle(3, 2);
   //incrémente compteurRectangle
    System.out.println(
          Rectangle. compteurRectangle);
     //affiche 2
}
```

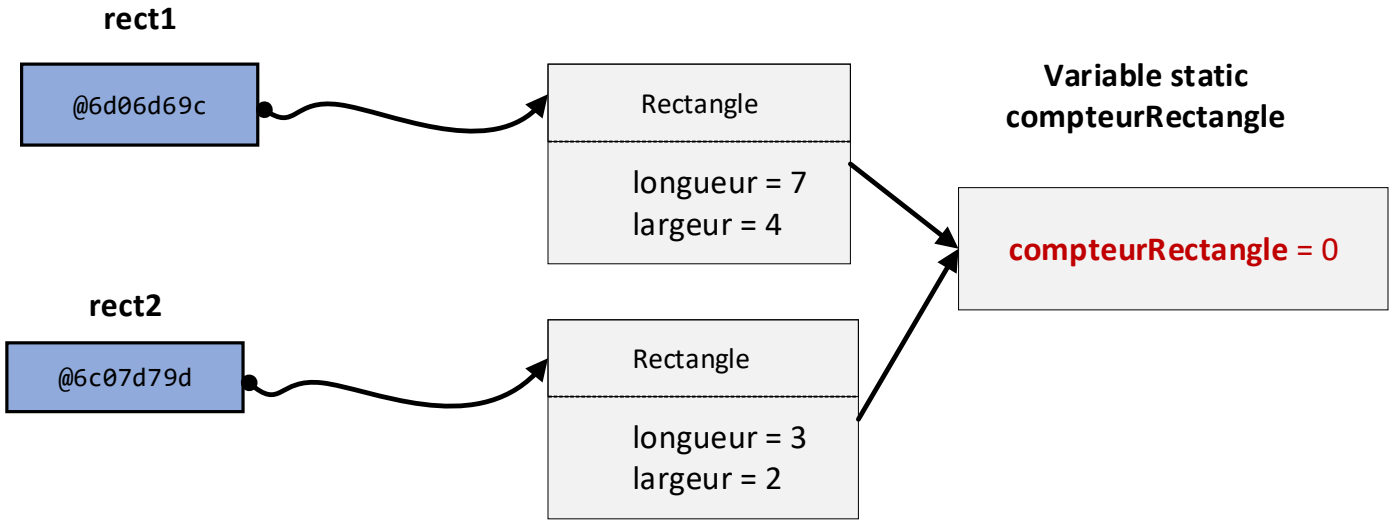
- En dehors de la classe, il faut préciser le nom de la classe suivi de la variable statique.

- Dans la classe, on référence directement la variable statique.
- `compteurRectangle` est incrémentée à chaque instanciation de la classe Rectangle.

Attribut statique (ou variable de classe)

Il y aura un attribut `longueur` et un attribut `largeur` pour chaque instance (objet) de la classe `Rectangle`, mais il n'y a qu'un seul champ statique `compteurRectangle`.

- L'attribut statique `compteurRectangle` appartient à la classe `Rectangle` et non pas aux objets (instances) de la classe `Rectangle`. **Il existe un seul exemplaire** de cet attribut statique que toutes les instances de la classe `Rectangle` se partagent. Cet attribut aura la même valeur pour tous les objets Rectangle.
- La création de cette variable statique permet d'économiser l'espace mémoire. En effet, on a une copie unique pour toutes les instances de `Rectangle`.



Accès à un attribut statique

- Les attributs statiques sont accessibles sans initialisation d'objet. **On y accède avec le nom de la classe suivi du point suivi du nom de l'attribut,** s'il n'est pas protégé.

```
Rectangle.compteurRectangle;
```

- Même si aucun objet n'est encore instancié, le champ statique est présent et peut être utilisé. **Un champ statique est disponible aussitôt que la classe est chargée en mémoire.**

- On peut accéder à un attribut statique en utilisant une référence d'objet, mais il n'est pas conseillé. En effet, il devient difficile dans ce cas de déterminer s'il s'agit d'une variable d'instance ou de classe.
- Exemple:

```
rect1.compteurRectangle = 45;
```

  - Attention! Bien que `compteurRectangle` soit modifié par l'instance `rect1`, **cette valeur est modifiée pour tous les objets de la classe puisqu'elle est unique.**

Accès à un attribut statique

- Dans le cas où l'attribut statique est privé, on lui ajoute un accesseur comme pour les attributs d'instance.
- Exemple:

```
public class Rectangle {
   // Attribut statique ou variable de classe
   private static int compteurRectangle = 0;

   // Attributs ou variables d'instance de la classe Rectangle
   private int longueur;
   private int largeur;

   // Constructeur
   public Rectangle(int longueur, int largeur) {
      setLongueur(longueur);
      setLargeur(largeur);
      compteurRectangle++; // une instance de plus
   }

   // Accesseur de l'attribut statique compteurRectangle
   public static int getCompteurRectangle() {
      return compteurRectangle;
   }
}
```

```
public static void main(String[] args) {
   Rectangle rect1, rect2;
   rect1 = new Rectangle(7, 4);
   rect2 = new Rectangle(3, 2);
   System.out.print("Nombre d'occurrences: "
+ Rectangle.getCompteurRectangle());
}
```

Méthode statique

- Une méthode statique, également appelée méthode de classe, est une méthode qui a un rôle indépendant d'une quelconque instance. **Elle appartient à la classe et non à une instance quelconque de la classe**.
- On utilise une méthode statique dans les cas suivants:
  - La méthode n'a **pas besoin d'accéder à l'état d'un objet** (attributs d'instance).
  - la méthode **n'accède qu'à des variables statiques** de la classe.
- Nous n'avons pas besoin de créer un objet pour appeler une méthode statique. On invoque une méthode statique à l'aide du nom de la classe suivi de l'opérateur « . » suivi du nom de la méthode.
- Exemple:

```
public class Rectangle {
   // Attribut statique ou variable de classe
   private static int compteurRectangle = 0;

  // Attributs d'instance
   private int longueur;
   private int largeur;

   public Rectangle(int longueur, int largeur) {
      setLongueur(longueur);
      setLargeur(largeur);
      compteurRectangle++; // une instance de plus
   }
   // méthode statique
   public static int getCompteurRectangle() {
      return compteurRectangle;
   }
}
```

```
public static void main(String[] args) {
   Rectangle rect1, rect2;
   rect1 = new Rectangle(7, 4);
   rect2 = new Rectangle(3, 2);
   System.out.print("Nombre d'occurrences: "
+ Rectangle.getCompteurRectangle());
}
```

Méthode statique

- **Une méthode statique ne peut accéder aux attributs et méthodes qui ne sont pas statiques**. Puisqu'une méthode statique n'appartient à aucune instance de la classe, elle ne peut invoquer ni un attribut d'instance ni une méthode d'instance.
- **Une méthode statique ne possède aucune référence `this`**, puisqu'elle n'est rattachée à aucun objet de la classe. D'ailleurs, elle existe avant même l'instanciation du tout premier objet de la classe.
- **Un constructeur n'est jamais statique**

```
12  public static Rectangle(int longueur, int largeur) {
13      setLongueur(longueur);
14      setLargeur(largeur);
15      compteurRectangle++;
16  }
17
18  public static int testMethodeStatic() {
19      longueur = 3;
20      this.compteurRectangle =10;
```

- Le message d'erreur de la ligne 12

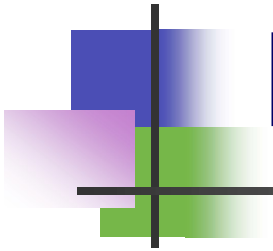
«Illegal modifier for the constructor in type Rectangle; only public, protected & private are permitted»

- Le message d'erreur de la ligne 19

«Cannot make a static reference to the non-static field longueur»

- Le message d'erreur de la ligne 20

«Cannot use this in a static context»



Méthode statique

- **Les méthodes statiques sont souvent utilisées dans des classes utilitaires** ou auxiliaires afin de pouvoir les utiliser sans créer d'objet.
  Exemple: les méthodes `Math.sqrt()` et `Math.pow()` sont des méthodes statiques.

- La méthode `main()` est la méthode principale ou le point d'entrée d'un programme. Elle est invoquée automatiquement lors de l'exécution du programme.

- La méthode `main()` est une méthode statique. Elle n'a besoin d'aucun objet pour être appelée. Lorsqu'un programme démarre, il n'existe aucun objet dont le programme a besoin.

Une constante statique

- C'est une constante commune à tous les objets de la classe.
- C'est un attribut déclaré avec le mot `static` et `final` (il ne peut être initialisé qu'une seule fois).
- Exemple: ***TYPE_FORME = "Rectangle"*** est une constante commune à tous les objets Rectangle.

```
public static final String TYPE_FORME = "Rectangle";
```

```
public class Rectangle {
   // Constante publique et statique (de classe)
   public static final String TYPE_FORME = "Rectangle";

  // Variable statique ou de classe
   private static int compteurRectangle = 0;

   // Attributs de classe
   private int longueur;
   private int largeur;

   // Constructeur
   public Rectangle(int longueur, int largeur) {
         setLongueur(longueur);
         setLargeur(largeur);
         compteurRectangle++; // une instance de plus
    }

public void afficher() {
   System.out.println(TYPE_FORME + "  de longueur "+
            longueur + " et de largeur "+ largeur);
   }
}
```

```
public static void main(String[] args) {
   Rectangle rect1;
   System.out.println(
        Rectangle.TYPE_FORME );
    // affiche Rectangle
    rect1 = new Rectangle(7, 4);
    rect1.afficher();
    // affiche: Rectangle de  longueur 7
    et de largeur 3
    …

}
```

Classe utilitaire

- Une classe utilitaire regroupe des fonctionnalités utiles au programme en cours de développement. Elle possède les caractéristiques suivantes:
  - Elle ne contient aucun attribut, ce qui implique l'absence de getters et de setters.
  - Elle se compose exclusivement de constantes et de méthodes statiques, permettant ainsi leur invocation depuis une autre classe sans nécessiter d'instancier un objet.
  - Elle ne contient généralement pas de constructeur public. Cependant, la classe peut avoir un constructeur privé pour empêcher explicitement toute instanciation depuis l'extérieur.

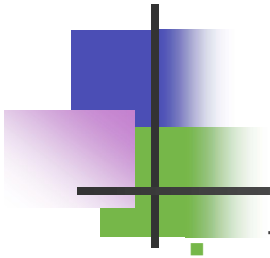
```
public class NomDeLaClasseUtilitaire {
    // Constantes, le cas échéant
    public static final int UNE_CONSTANTE = 42;

    // Constructeur privé pour empêcher l'instanciation
    private NomDeLaClasseUtilitaire() {
        // Constructeur privé pour empêcher l'instanciation
    }

    // Méthode utilitaire 1
    public static TypeRetour methodeUtilitaire1(TypeParametre1 parametre1, TypeParametre2
parametre2, …) {…}

    // Méthode utilitaire 2
    public static TypeRetour methodeUtilitaire2(TypeParametre1 parametre1, TypeParametre2
parametre2, …) {…}

    // Autres méthodes utilitaires
}
```

La classe utilitaire Math

- La classe `java.lang.Math` est une classe utilitaire en Java qui fournit des méthodes statiques pour effectuer des opérations mathématiques.
- Elle inclut des constantes et des méthodes telles que :
  - `public static final double PI`
  - `public static double sqrt(double a){ … }`
  - `public static double pow(double a, double b){…}`
- La classe Math a un constructeur qui n'est pas visible (privé)
- https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html

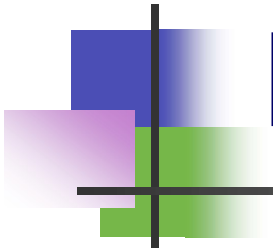
- Exemples d'utilisation de la classe Math:

```
public double perimetre() {
        return Math.PI * rayon * 2;
  }

 boolean estSurCercle(double x, double y) {
      return Math.pow((x - h), 2) + Math.pow((y - k), 2) == Math.pow(rayon, 2);
 }
```



Résumé

- Tout ce qui est déclaré **static**, appartient à la classe et non à une instance particulière de cette classe.
- Il est possible d'avoir des attributs statiques et des méthodes statiques.



Exercice

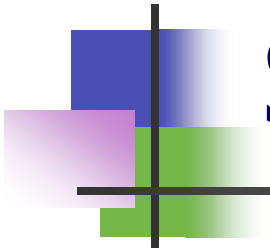
- Créez une classe utilitaire nommée `UtilPuissance`
- Ajoutez un constructeur privé par défaut qui ne fait rien
- Ajoutez une méthode publique statique `double calculerCarre(double nombre)` qui calcule et retourne le carré d'un nombre de type `double` reçu en paramètre.
- Ajoutez une méthode publique statique `long calculerCarre(int nombre)` qui calcule et retourne le carré d'un nombre de type `int` reçu en paramètre.
- Ajoutez une méthode publique statique `double calculerCube(double nombre)` qui calcule et retourne le cube d'un nombre de type double reçu en paramètre.
- Créez une classe nommée `MainUtilPuissance` comportant une méthode `main()` pour tester les 3 méthodes.
  - Calculez le carré de 78,5
  - Calculez le carré de 200
  - Calculez le cube de 15

Corrigé de l'exercice (1/2)

```
public class UtilPuissance {
    // Constructeur privé pour empêcher l'instanciation
    private UtilPuissance() {
    }

    // Méthode utilitaire 1 - Carré d'un double
    public static double calculerCarre(double nombre) {
        return nombre * nombre;
    }

    // Méthode surchargée 1 - Carré d'un int
    public static long calculerCarre(int nombre) {
        return nombre * nombre;
    }

    // Méthode utilitaire 2 - Cube d'un double
    public static double calculerCube(double nombre) {
       // return nombre * nombre * nombre;
       return Math.pow(nombre, 3);
}

}
```

Corrigé de l'exercice (2/2)

```
public class MainUtilPuissance {

    public static void main(String[] args) {
        System.out.println("le carre de 78.5 est  " + UtilPuissance.calculerCarre(78.5));
        System.out.println("le carre de 200 est " + UtilPuissance.calculerCarre(200));
        System.out.println("le cube de 15 est " + UtilPuissance.calculerCube(15));
    }

}
```



Sources

- Kypriano, S. Le modificateur static
- Apprendre Java et la programmation orientée objet: https://www.ukonline.be/cours/java/apprendre-java
- Tasso, A. (s.d.). Le livre de JAVA premier langage avec 109 exercices corrigés. Éditions EYROLLES.