



# PROGRAMMATION ORIENTÉE OBJET JAVA

LICENCE 2 INGÉNIERIE - INFORMATIQUE

2020– 2021

Marie NDIAYE



# LES MODIFICATEURS D'ACCÈS

# VISIBILITÉ DES MEMBRES (MODIFICATEUR D'ACCÈS)

- Java permet plusieurs degrés d'encapsulation pour les membres (**variables** et **méthodes**) et les **constructeurs** d'une classe.
  - **private**
    - Entité privée : inaccessible depuis les autres classes (même filles)
  - **public**
    - Entité publique : accessible à tous
  - **protected**
    - Entité protégée : accessible aux membres du même package et leurs sous-classes (devient private dans les sous-classes)

## EXAMPLE

```
public class Livre {  
    private String titre;  
    private Lecteur emprunteur;  
    ...  
    private boolean estEmprunte() {  
        if (emprunteur == null) return false;  
        else return true;  
    }  
  
    public Date emprunte(Lecteur lec) {  
        if ( this.estEmprunte() == true)  
            return null;  
        if ( lec.empruntPossible() ) {  
            emprunteur = lec;  
            lec.ajouteEmprunt( this );  
            return new Date();  
        } else return null;  
    }  
}
```

```
public class Lecteur {  
    private Livre[ ] emprunts;  
    private int nbEmprunts;  
    ...  
    public boolean empruntPossible() {  
        if (nbEmprunts < 5) return true;  
        else return false;  
    }  
  
    public void ajouteEmprunt(Livre liv)  
    {  
        emprunts[nbEmprunts] = liv;  
        nbEmprunts ++;  
    }  
}
```

# AUTRES MODIFICATEURS D'ACCÈS : FINAL

- **Attribut** (ou variable):
  - Constante d'instance (affectation d'une valeur une seule fois)
  - Ex : **final** String LANG\_PAR\_DEFAULT = "lg\_Fr"
- **Méthode**:
  - Méthode non surchargeable et non redéfinissable.
- **Classe**
  - Ne peut avoir de sous-classe
  - Ex : **public final class** Integer

## AUTRES MODIFICATEURS D'ACCÈS : **STATIC**

- **Attribut** : variable de classe

- Création d'un unique exemplaire de l'entité
- Ex : `public static final double PI ??`

- **Méthode** : méthode de classe

- Méthode sans envoi de messages.
- Appliquée à la classe et non à un objet
- Ex : `public static double sqrt (double a)`

# VARIABLES DE CLASSE

- Certaines variables sont partagées par toutes les instances d'une classe. Ce sont les variables de classe (modificateur **static**)
- Si une variable de classe est initialisée dans sa déclaration, cette initialisation est exécutée une seule fois quand la classe est chargée en mémoire

# EXEMPLE

```
public class Employe {  
  
    private String nom, prenom;  
  
    private double salaire;  
  
    private static int nbEmployes = 0;  
  
    // Constructeur  
  
    public Employe(String n, String p) {  
  
        nom = n;  
  
        prenom = p;  
  
        nbEmployes++;  
  
    }  
  
    ... }  
}
```



# MÉTHODE DE CLASSE

- Une méthode de classe (modificateur **static**) exécute une action indépendante d'une instance particulière de la classe
- Une méthode de classe peut être considérée comme un message envoyé à une classe
- Exemple :

```
public static int getNbEmployes() {  
    return nbEmployes; // nbEmployes déclaré static  
}
```

## DÉSIGNER UNE MÉTHODE DE CLASSE 1/2

- Pour désigner une méthode static depuis une autre classe, on la préfixe par le nom de la classe

```
int n = Employee.getNbEmploye();
```

```
double r = Math.sqrt(144.0);
```

- On peut aussi la préfixer par une instance quelconque de la classe (à éviter car cela nuit à la lisibilité : on ne voit pas que la méthode est static)

```
int n = e1.getNbEmploye();
```

## DÉSIGNER UNE MÉTHODE DE CLASSE 2/2

- Comme une méthode de classe exécute une action indépendante d'une instance particulière de la classe, elle ne peut utiliser de référence à une instance courante (this)
- Il serait, par exemple, interdit d'écrire

```
static double tripleSalaire() {  
    return salaire * 3; // salaire: variable d'instance  
}
```

# LA MÉTHODE MAIN

- Elle est nécessairement **static**.
- La méthode `main()` est exécutée au début du programme. Aucune instance n'est donc déjà créée lorsque la méthode `main()` commence son exécution. Ça ne peut donc pas être une méthode d'instance.

# GRANULARITÉ DE LA PROTECTION DES ATTRIBUTS D'UNE CLASSE

- En Java, la protection des attributs se fait classe par classe, et pas objet par objet
- Un objet a accès à tous les attributs d'un objet de la même classe, même les attributs privés

# PROTECTION DE L'ÉTAT INTERNE D'UN OBJET

- Autant que possible l'état d'un objet (les variables) doit être private.
- Si on veut autoriser la lecture d'une variable, on lui associe un accesseur (**getXX**), auquel on donne le niveau d'accessibilité que l'on veut.
- Si on veut autoriser la modification d'une variable, on lui associe un modificateur (**setXX**), qui permet la modification tout en contrôlant la validité de la modification.