

DUT MMI – IUT de Marne-la-Vallée

18/03/2016

M2202 - Algorithmique

## ***Cours 5***

# ***Programmation objets : les principes***

# Sources

---

- Cours d'Anne Tasso à l'IUT de Marne-la-Vallée
- *Le livre de Java premier langage*, d'A. Tasso

# Plan du cours 5 – Objets

---

- Concept de la programmation orientée objets
- Protection des données et encapsulation

# Plan du cours 5 – Objets

---

- Concept de la programmation orientée objets
- Protection des données et encapsulation

# Concept de la programmation orientée objets

## Objectif : modéliser des objets informatiques

- décrire leurs **caractéristiques**
- décrire leurs **comportements**

## Avantages :

- code modulaire, décomposé en “classes” d'objets
- code sécurisé, possible de réutiliser certains objets dans d'autres programmes

types d'objets Java que le programmeur peut définir



→ **simplifie la vie du programmeur**

- Exemples :
- les actions Javascript sur les composants d'une page web
  - la gestion des notes d'une classe

# Modélisation objets

Quelles caractéristiques / attributs ? **“propriétés”**

*Composants d'une page web*



*Etudiants d'une classe*

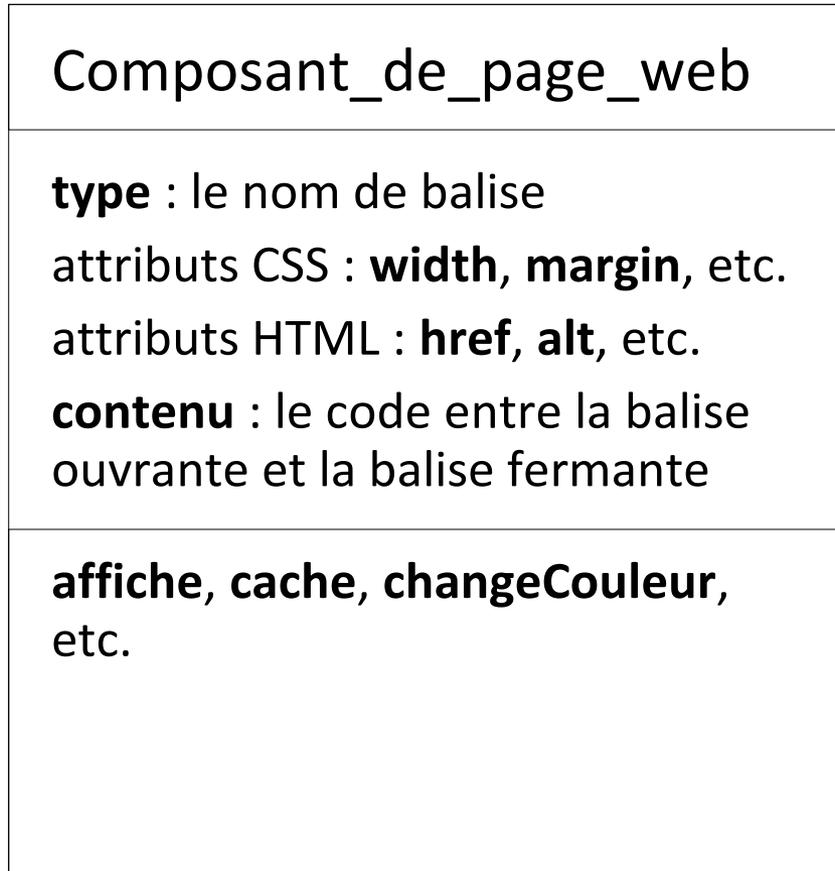


Quels comportements / actions ? **“méthodes”**

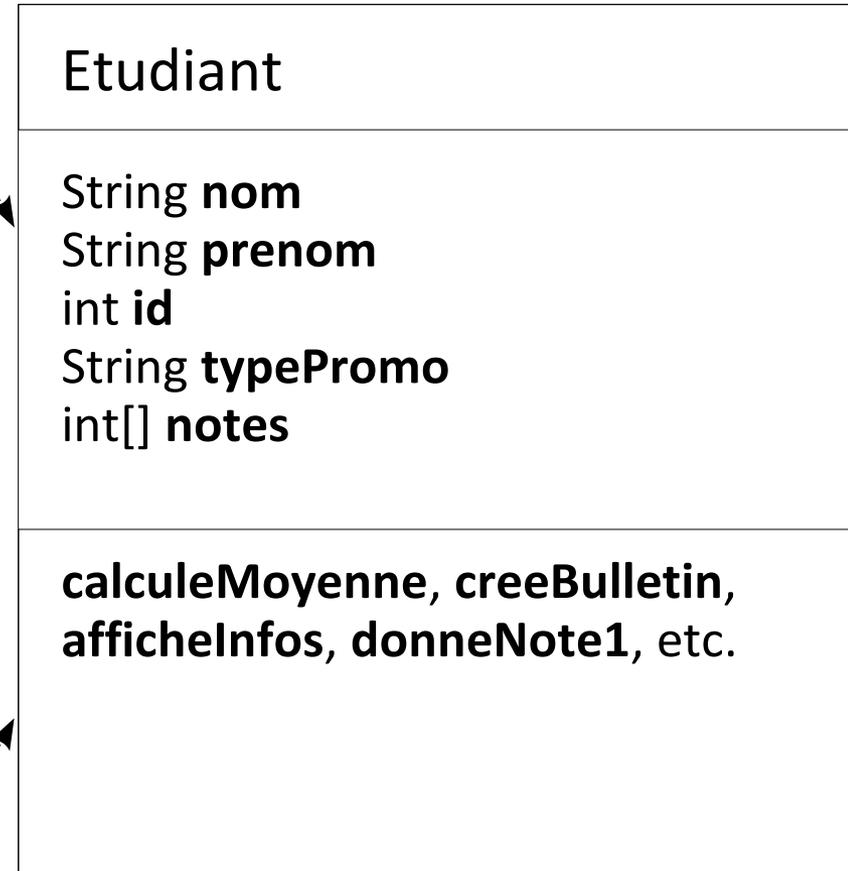
# Modélisation objets

Quelles caractéristiques / attributs ? “**propriétés**”

*Composants d'une page web*



*Etudiants d'une classe*



Quels comportements / actions ? “**méthodes**”

# Modélisation objets

---

La différence entre un objet et une classe ?

# Modélisation objets

La différence entre l'objet et sa classe est la même qu'entre :

- qu'entre une **variable** et son **type**
- qu'entre une **occurrence** et son **entité** (en bases de données)
- qu'entre la **ligne de données d'une table**, et la **classe** qui correspond à la table (en bases de données).

Ainsi, l'objet est un élément spécifique d'une classe.

Utiliser la variable `etudiant1` comme un objet de la classe `Etudiant` :

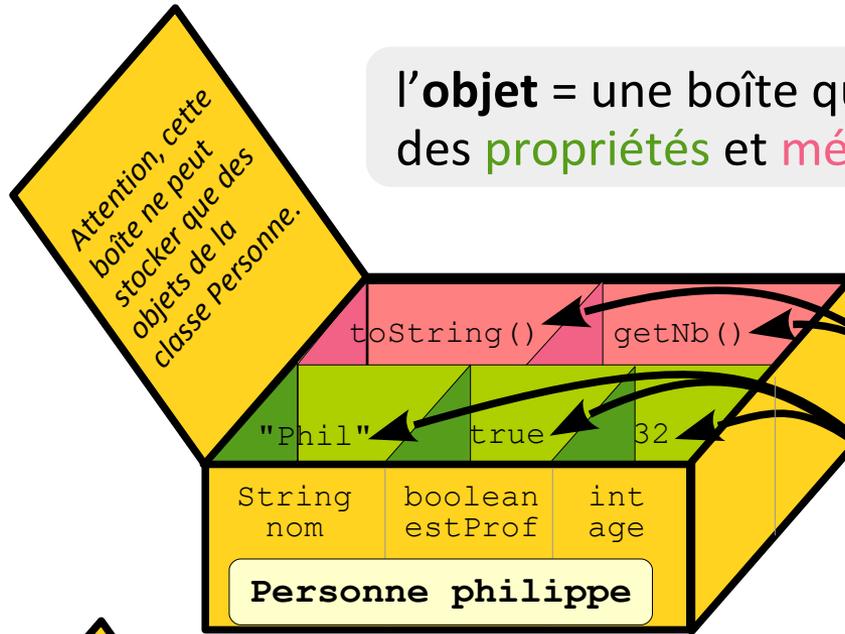
```
//Déclaration (nom de variable précédé de son type) :  
Etudiant etudiant1;  
  
//Utilisation de propriétés de la classe Etudiant :  
System.out.println(etudiant1.prenom+" "+etudiant1.nom);  
  
//Appel de méthodes de la classe Etudiant :  
int moyenne = etudiant1.calculeMoyenne();
```

# Modélisation objets – en résumé

Java 1er semestre	Java objet	Bases de données
type	classe	entité
valeur	objet = instance	occurrence
variable	variable de classe = propriété	attribut
fonction	méthode	-

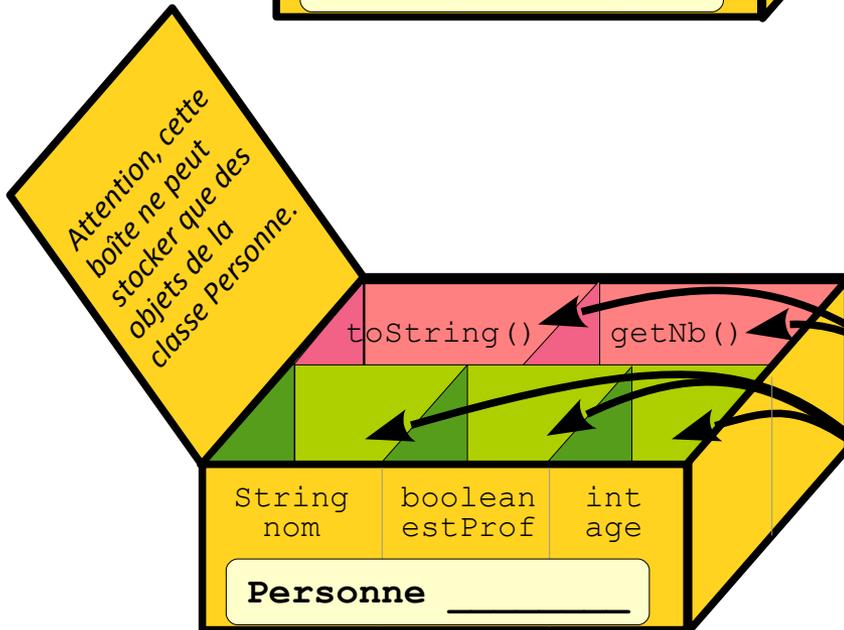
# Modélisation objets – en résumé

l'objet = une boîte qui contient des propriétés et méthodes



2 méthodes de l'objet philippe de la classe Personne

3 propriétés de l'objet philippe de la classe Personne



la classe =  
ce format de boîte

2 méthodes de la classe Personne

3 propriétés de la classe Personne

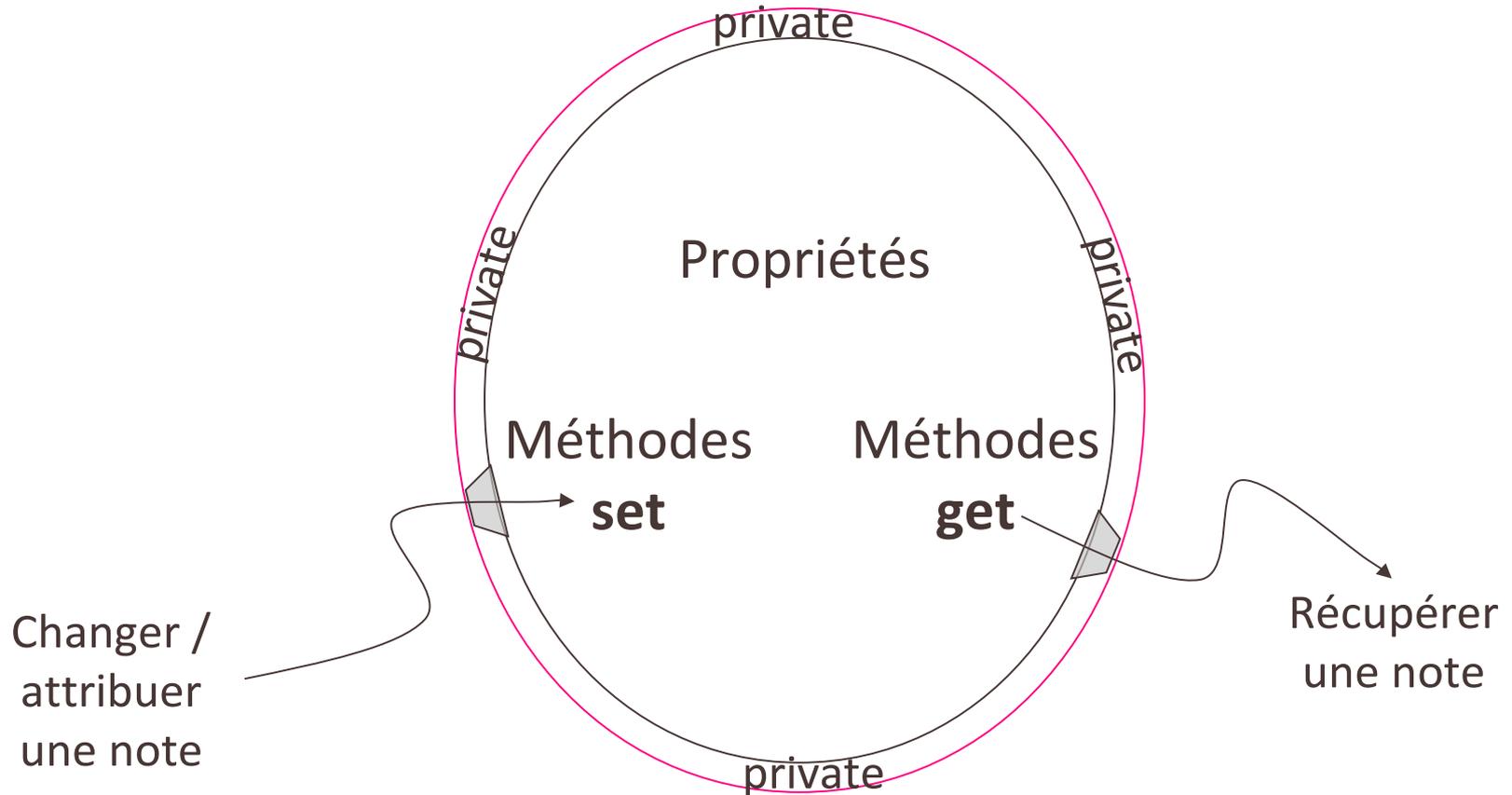
# Plan du cours 5 – Objets

---

- Concept de la programmation orientée objets
- Protection des données et encapsulation

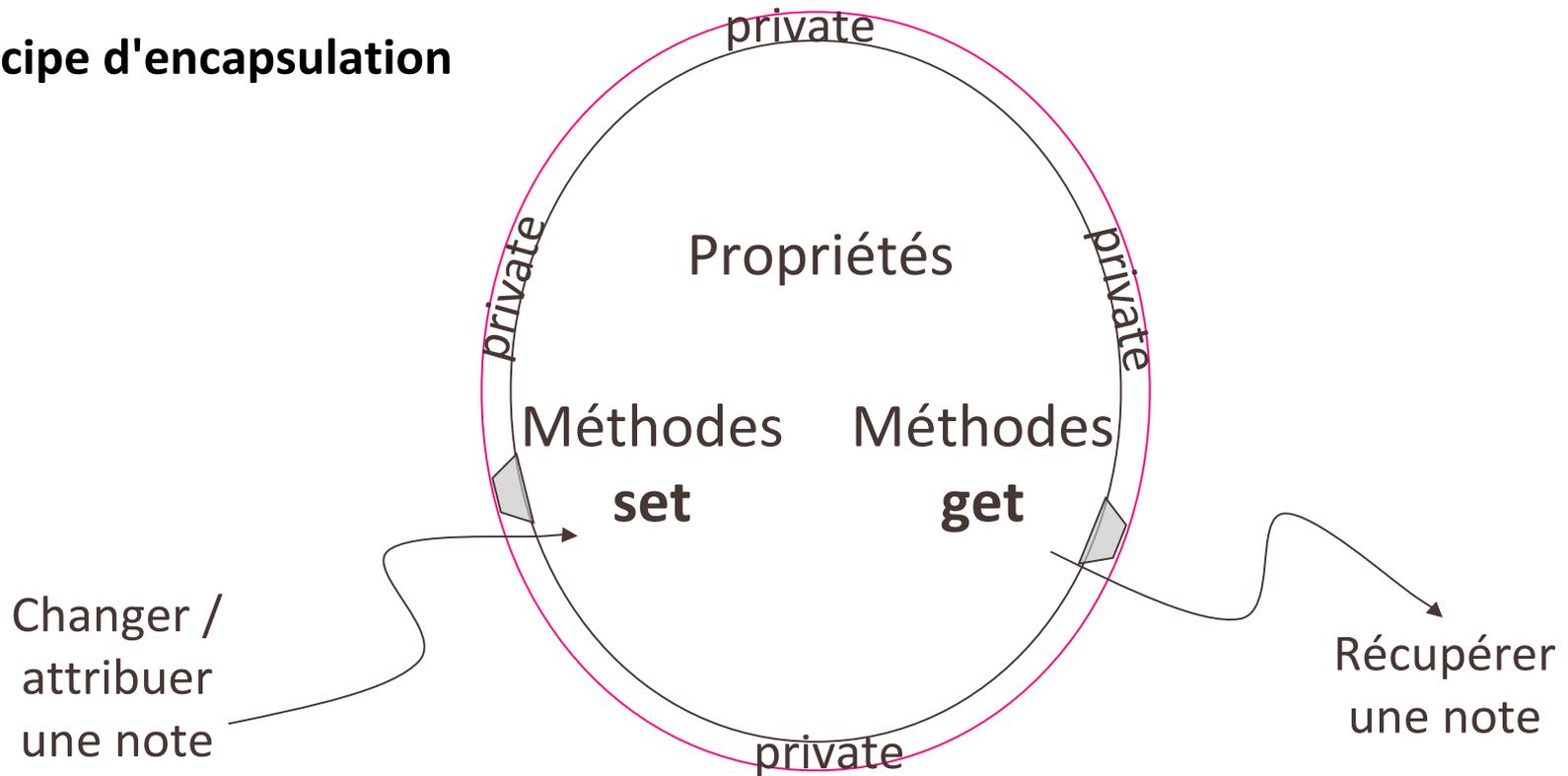
# Protection des données

## Principe d'encapsulation



# Protection des données

## Principe d'encapsulation



Utilisation d'objets de la classe `Etudiant` depuis la classe `GestionEtudiants` :

```
public static void entreNote1(String nomEtudiant){  
    etudiant Etudiant1 = trouveEtudiantParNom(nomEtudiant);  
    //sans encapsulation (public)  
    etudiant1.note1 = 12;  
    System.out.println(  
        etudiant1.note1);  
}  
    //avec encapsulation (private)  
    etudiant1.setNote1(12);  
    System.out.println(  
        etudiant1.getNote1());  
}
```

# Protection des données

## Constructeur

Sert à créer un objet

→ fournir une valeur à ses propriétés (initialisation)

Le constructeur par défaut initialise :

→ les entiers à 0

→ les flottants à 0.0

→ les chaînes de caractères à `Null`

Pour changer les valeurs des propriétés d'un objet par la suite :

- les modifier depuis une autre classe ?
- les modifier depuis une méthode de la classe de l'objet ?

# Protection des données

**Différents niveaux de protection** (pour les propriétés et pour les méthodes)

- `public` : accessible pour tous les objets de l'application
- `private` : accessible que pour les méthodes de la même classe
- `protected` : accessible que depuis les méthodes de la même classe ou d'une sous-classe

**Partage d'une donnée entre objets d'une même classe :**

- `static`
- sinon les données sont spécifique à un objet donné de la classe

*Similaire à variables de classe / variables locales, au niveau des objets plutôt que des fonctions*