Ecole Supérieure
d'Informatique et du Numérique
COLLEGE OF ENGINEERING & ARCHITECTURE

Chap3: Classes en JAVA

Classes

Plan

Après la création et l'utilisation d'une classe principale (Main), L'objectif de ce chapitre est de vous présenter la création des classes utilitaires

- Création et manipulation des variables d'instance / variables de classe
- Création et manipulation des méthodes d'instance / méthodes de classe
- La surcharge des méthodes
- Création et manipulation des Constructeurs
- L'utilisation des getters et des setters

La classe (rappel Déf)

- Une classe est un type de données abstrait.
- Les classes réunissent les caractéristiques communes à un ensemble d'objets
- Une classe en java ou en C++ joue un rôle dual. *C'est à la fois une intension (**type**) et une extension (**moule**)*

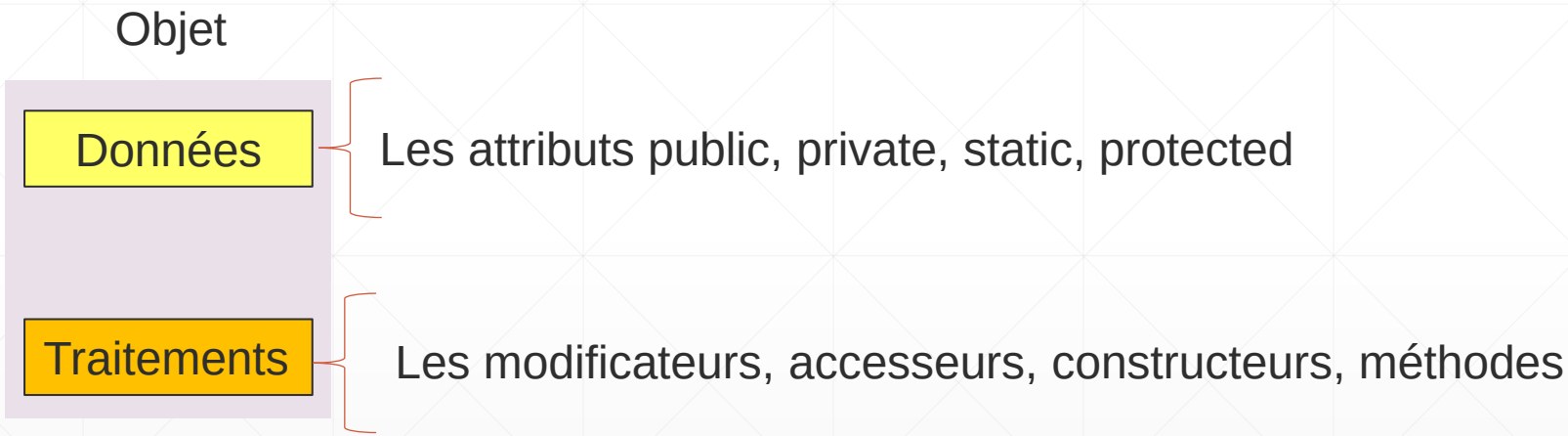
Objet (Définition)

- En Java tout est objet
- Un objet réel est l'autre nom d'une instance de classe
- Si les classes sont une représentation abstraite d'un objet, une instance en est une représentation concrète
- Une application est un ensemble d ’objets communiquant par un système d’envoi de messages (activation de méthodes)

Objet (Définition)

- Un objet peut être construit à partir d'objet déjà existants
  - héritage
- Chaque objet a un type
  - instance d 'une classe -> type
- Tout objet de même type (*y compris par inclusion*) peut recevoir les mêmes messages
  - substitution, polymorphisme

Objet et encapsulation



Les attributs (Définition)

- Les attributs sont les caractéristiques individuelles qui différencient un objet d'un autre. Ils définissent l'état d'un objet.

  - Soit une classe appelée Motocycle avec les attributs suivants:
    - couleur
    - style
    - marque

Il existe deux types d'attributs:
- Attribut d'instance (lié à l'objet)
- Attribut de classe

Visibilité des attributs

- La visibilité des attributs peut être modifiée selon la décision du développeur
- On distingue trois droits d'accès: publique, privé, protégé

- **public** attribut public accessible sans méthode pour toutes les parties du programme
- **private** attribut privé nécessite des méthodes getter/ setter pour récupérer et modifier sa valeur accessible seulement par la classe qui l'a défini
- **protected** attribut accessible seulement par la classe qui la définit et la méthode et ses classes filles

Visibilité des attributs

La modification de la visibilité des attributs permet de:

- Cacher la complexité de l'objet
- Restreint l'accès à certains attributs et méthodes
- Permet la modification de certaines méthodes sans perturber l'utilisateur

Les méthodes (Définition)

- Les méthodes d'une classe déterminent les actions sur ses instances :
  - C'est souvent la seule voie pour agir sur l'état d'un objet
- Exemple de méthodes pour notre classe MotoCycle:
  - démarrer le moteur, arrêter le moteur, accélérer
- La définition du comportement d'un objet passe donc par la création de méthodes. Celles-ci ressemblent aux fonctions des autres langages, *mais sont définies dans une classe*

Il existe deux types de méthodes :
- **Méthode d'instance**, son traitement dépend de l'objet
- **Méthode de classe**, son traitement dépend de la classe

Remarque: en JAVA toutes les méthodes sont déclarées dans les classes

Création d'une classe

- La création d'une class JAVA se fait en utilisant le mot clé **class** et le nom de la classe comme suite:

```
public class Etudiant {

}
```

Attributs d’instance /de classe

- Il existe deux types d’attributs (les variables) en java :

➢ Les attributs d’instances

➢ Les attributs de classe

Les variables membres = var de classe et les variables d’instance

Types d’attributs

- Comme dans les autres langages de programmation JAVA permet la déclaration des attributs de la manière suivante:

[DroitDacces] typeDeAttribut nomAttribut ;

Par exemple: pour la déclaration d’une chaîne de caractères

```
public String nomEtudiant;
```

Pour la déclaration d’un attribut entier

```
public int ageEtudiant;
```

Variable d'instance

- Une variable est dite variable d'instance si elle est déclarée en dehors du corps d'une méthode et sa déclaration n'est pas précédée par le mot clé **static** comme suite:

```
public class Etudiant {
    // les variables d'instance
    int CNE;
    String nomEtudiant, prenomEtudiant;
    Date dateNaissance;

}
```

Par convention ces attributs sont déclarés juste après la déclaration de la classe

Variable de classe

- Une variable de classe est une variable dont la valeur est partagée par toutes les instances d'une classe.
- On déclare une variable de classe par le mot clés **static:**

```
public class Etudiant {
    //déclaration des attributs
    public int id;
    public String nom;
    // attributs privés
    private double moyenne;
    // attribut de classe
    public static int nbrEtdInfo;
    // constante
    public final static int NBRMAXETD = 24;
}
```

Un attribut de classe garde la même valeur pour toutes les instances de la classe

Les méthodes de base

- Java a défini un ensemble de classe de base pour assurer les fonctionnalités de base
  - Les **constructeurs** : la création des objets
  - Les **accesseurs**: pour récupérer les valeurs des attributs privés
  - Les **modificateurs**: pour modifier les valeurs des attributs privés
  - Les méthodes **toString**(): pour afficher l'état d'un objet

La signature d'une méthode consiste en la combinaison du type de résultat renvoyé par la méthode, le nom de la méthode, et la liste des paramètres

Les méthodes de base (Constructeur)

- Une définition de classe contient des constructeurs
- Un constructeur est une méthode particulière qui détermine la manière dont un objet est initialisé lors de sa création

- Les constructeurs diffèrent des méthodes ordinaires sur deux points:
  - les constructeurs portent **toujours le même nom que la classe**
  - Les constructeurs n'ont pas de type de valeur retournée

- Définir un constructeur dans une classe permet de mettre à jour les variables d'instances ou d'appeler les méthodes construites sur ces variables à la création de l'objet

Les méthodes de base (Constructeur)

- Un constructeur vide est déclaré par défaut dans une classe
- On peut déclarer plusieurs constructeurs avec changement de signature:
  - Constructeur vide
  - Constructeur full-paramètres

Les méthodes de base (Constructeur)

Un constructeur d'une classe est une méthode utilisée par la MVJ lors de la création des objets instances de cette classe.

Ils sont appelés automatiquement par la MVJ quand un objet est créé par l'opérateur **new**.

Java effectue trois tâches lorsque l'opérateur **new** est utilisé pour créer un objet:

1. Allouer la mémoire
2. Initialiser les variables d'instance de cet objet, soit par des valeur initiales soit par les valeurs par défaut (0 pour les nombres, false pour boolean, null pour les objets et '\0' pour char).
3. Appelé l'un des constructeurs de la classe

Les méthodes de base (Constructeur)

Un constructeur diffère d'une méthode d'instance en trois points :

1. Il porte toujours le nom de la classe
2. Il n'a pas un type de retour
3. Il ne renvoi aucune valeur (pas de mot clé return dans le corps du constructeur)

Auto-référencement

- Pour référencer l'objet courant dans le corps d'une de ses méthodes par exemple pour accéder à ses variables d'instance on peut les préfixer par le mot clés "this ",

- ***this*** permet de désigner l'objet courant

- Souvent "**this**" est inutile. En effet, les variables d'instances et les appels de méthodes définis dans la classe courante sont implicitement référencées par leur nom "***this***"

Les méthodes de base (Constructeur)

- Exemple d'un constructeur vide et constructeur full-paramètres

```
// constructeur vide
public Etudiant() {

}

// constructeur fullparamètres
public Etudiant(int id, String nom, double moyenne) {
    this.id = id;
    this.nom = nom;
    this.moyenne = moyenne;
}
```

Remarque: **Les attributs de classe et les constantes sont indépendants du constructeur**

Les méthodes de base (Constructeur)

- L'instanciation des objets se fait avec l'instruction **new**

```
public class Main {
    public static void main(String[] args) {
        // constructeur vide
        Etudiant e1 = new Etudiant();
        // constructeur full-params
        Etudiant e2 = new Etudiant(1, "Durant", 15.2);
    }
}
```

Accès aux variables de classe et d'instance

- L'accès aux variables de classe passe directement par la classe comme suite:

- **NomClasse.nomVarClasse**

```
public static void main(String[] args) {
    System.out.println(" var classe "+ Etudiant.maxAge);
    System.out.println(" var classe "+ Etudiant.nombreEtudiant);
}
```

Accès aux variables de classe et d'instance

- Si la variable est publique (visible pour toutes les autres classes), un attribut d'instance peut être récupérée comme suite:
  - **nomObjet.nomVarInstance**

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

    Etudiant etd1 = new Etudiant();
    System.out.println("nom etd1: " + etd1.nomEtudiant);
    System.out.println("prenom etd1: " + etd1.prenomEtudiant);
```

Instanciation d'un objet

Accès à une variable d'instance public

Accès aux variables de classe et d'instance

- Exemple

```
// acces aux attributs d'instance
System.out.println("Le nom de e1 " + e1.nom);

// access aux attributs de classe
System.out.println("le nombre max des etds Ing3 = " + Etudiant.nbrMaxEtd );
```

- Résultat

```
Le nom de e1 Durant
le nombre max des etds Ing3 = 24
```

Les méthodes de base (accesseur)

- Les accesseurs ou bien les getters sont les méthodes utilisées pour la récupération de la valeur d'un attribut privé
- Les accesseurs sont des méthodes d'instances

- **Exemple**:

```
public   typeAttribut   getNomAttribut(){
            return this.attribut;
}
```

```
// méthode getter de l'attribut Moyenne
public double getMoyenne() {
        return moyenne;
}
```

Les méthodes de base (modificateur)

- Les modificateurs ou bien les setters sont les méthodes utilisées pour assigner une valeur à un attribut privé.
- Les modificateurs sont des méthodes d'instances

Exemple

```
public   void   setNomAttribut( typeAttribut  var2){
          this.nomAttribut = var2 ;
}
```

```
// méthode setter de l'attribut Moyenne
public void setMoyenne(double moyenne) {
        this.moyenne = moyenne;
}
```

Exercice d’application

- Nous souhaitons gérer la maintenance du matériel informatique existant dans la salle des TP. Pour ce faire, nous souhaitons enregistrer la configuration de chaque ordinateur. Un ordinateur est caractérisé par un numéro d’inventaire, marque, processeur (privé) et une mémoireRAM (4 ou 8Go).

➢ Demander à l’utilisateur de faire la saisie des informations d’un seul Ordinateur

➢ Afficher la configuration complète de cet ordinateur

➢ Adapter la classe Ordinateur pour générer automatiquement le numéro d’inventaire

Les méthodes de base (toString())

- ToString est une méthode utilisée pour afficher les avaleurs assignés aux attributs d'un objet
- Cette méthode est appelée par défaut si on affiche l'objet de recrutement sur la console en utilisant l'instruction **System.out.println(objet)**

Exemple

```
public String toString() {
        return "Etudiant [id=" + id + ", nom=" + nom + ", moyenne=" + moyenne + "]";
}
```

Les méthodes de base (toString())

- Exemple

```
public static void main(String[] args) {
    // constructeur full-params
    Etudiant e2 = new Etudiant(1, "Durant", 15.2);
    System.out.println(e2);
}
```

```
Etudiant [id=1, nom=Durant, moyenne=15.2]
```

Les méthodes d’instance

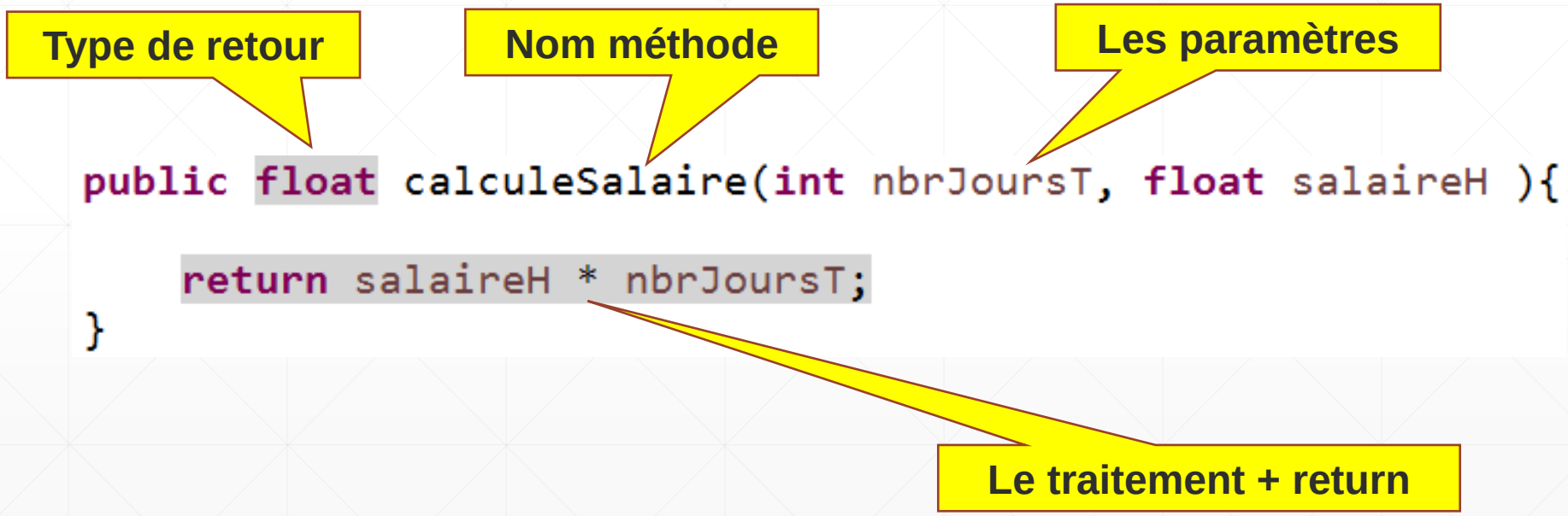
- Les méthodes d’instance en JAVA décrivent le comportement d’un objet, les tâches qu’il peut assurer.

- La déclaration d’une méthode d’instance comporte 4 parties:

1. Le nom de la méthode
2. La liste des paramètres
3. Le type d’objet ou primitif retourné par la méthode
4. Le corps de la méthode

Les méthodes d'instance

- Exemple d'une méthode de calcul du salaire d'un employé



La signature d'une méthode combine son nom et la liste des paramètres qu'elle reçoit

Les méthodes d'instance

- On utilise le mot clé **void** pour spécifier qu'une méthode ne retourne aucun résultat

```
// affichage d'un étudiant
public void afficherEtudiant(){
    System.out.println("nom "+ nombreEtudiant + " prenom " + prenomEtudiant );
}
```

- L'appel d'une méthode d'instance se fait comme suite

```
Etudiant etd1 = new Etudiant();
etd1.afficherEtudiant();
```

Les méthodes de classes

- La déclaration d'une méthode de classe est faite par l'utilisation du mot clé **static**

```
static void methodeClass(){
    System.out.println("je suis une méthode de classe");
}
```

- L'appel d'une méthode de classe se fait de la manière suivante
nomClasse.MethodeClasse

```
public static void main(String[] args) {
    Etudiant.methodeClass();
}
```

Donnez le nom d'une méthode de classe ? (Penser à la classe Integer)

Le passage d’argument

- Le passage des arguments se fait de la manière suivante:

```
// afficher le resultat
public void afficherRes(int resultat){
    System.out.println("le resultat est " + resultat);
}
// calcul de la somme
public void somme(int a , int b){
    int resultat = 0;
    resultat = a + b;
    afficherRes(resultat);
}
```

La surcharge des méthodes

En survolant l'API java vous avez sans doute rencontré des méthodes qui portent le même nom et non la même signature : un nombre différent d'arguments ou un type différent d'arguments.

Ex : println(int a), println(float x), println(String s)…

C'est ce qu'on appelle la surcharge de méthodes.

Avantages :

- Éliminer la définition de plusieurs méthodes qui font la même chose .
- Permettre de définir des méthodes qui agissent différemment selon les arguments reçus.

Attention !

Le type de retour doit rester le même, une tentation de le changer génère une erreur de compilation.

La surcharge des méthodes

- On peut définir la même méthode avec plusieurs signatures

```
public float calculeSalaire(int nbrJoursT, float salaireH ){

    return salaireH * nbrJoursT;
}

public float calculeSalaire(int nbrJoursT, float salaireH, int nbrH ){

    return salaireH * nbrJoursT * nbrH;
}
```

Exercice d’application (partie 2)

- Nous souhaitons garder la traçabilité des interventions de maintenance réalisées. Pour ce faire, nous souhaitons créer la classe maintenance qui permet de faire le lien entre le nom du technicien et les informations de l’ordinateur à maintenir.

➢ Créer la classe Maintenance (id, nomTechnicien, **Ordinateur**)

➢ Créer plusieurs instances de la classe maintenance

➢ Afficher les informations des différentes interventions réalisées