Java

Chapitre 1: Objets et Classes

Dr. Ahmad Khoureich Ka

UADB L2 D2AW

Notion d'objet

- Java est un langage orienté objet.
- Un programme Java est un ensemble d'objets interagissant entre eux pour résourdre un problème donné.
- Un objet est une entité ayant des propriétés et pouvant faire des actions.

Notion d'objet

Exemple d'objets

- **stylo**
  Propriétés : *couleur*, *longueur*, *poids*, *marque*, ...
  Actions : *écrire*, ...

- Le mot **travail**
  Propriétés : *type*, *synonymes*, ...
  Actions : *donner son nombre de caractères*, *se mettre en majuscule*, *se mettre en minuscule*, ...

- **écran**
  Propriétés : *taille*, *résolution*, *marque*, ...
  Actions : *afficher*, *s'allumer*, *s'éteindre*, *se mettre en veille*, ...

Exemple : un programme orienté objets

Écrire un programme permettant d'afficher le nombre de caractères contenus dans le mot **travail**.

Exemple : un programme orienté objets

Écrire un programme permettant d'afficher le nombre de caractères contenus dans le mot **travail**.

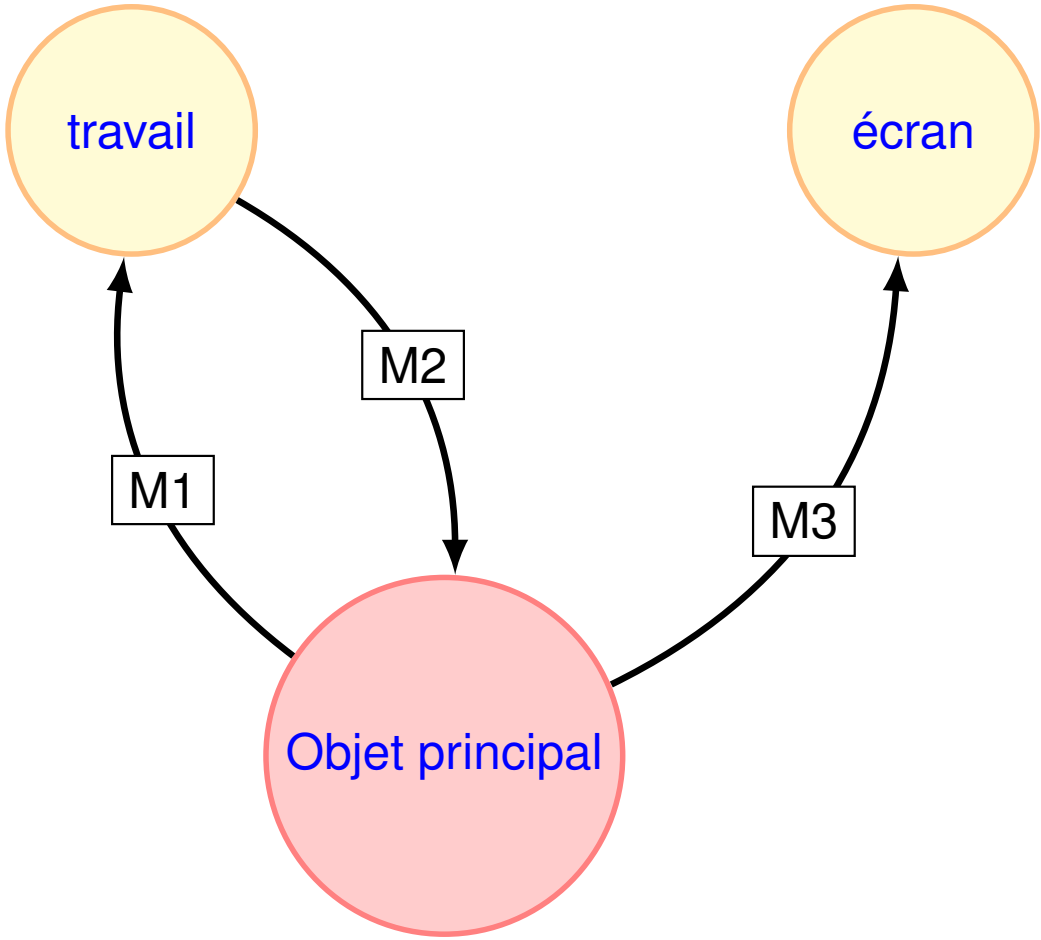
- ☞ Les objets du programme
  - Objet mot **travail**
  - Objet **écran**

Exemple : un programme orienté objets

Écrire un programme permettant d'afficher le nombre de caractères contenus dans le mot **travail**.

- Les objets du programme
  - Objet mot **travail**
  - Objet **écran**
- Algorithme
  1. Demander à l'objet mot **travail** de donner son nombre de caractères,
  2. Affecter le nombre donné par l'objet **travail** à la variable `n`
  3. Demander à l'objet **écran** d'afficher la variable `n`

Exemple : un programme orienté objets



- M1 : donne le nombre de tes caractères
- M2 : nombre de caractères `n`
- M3 : affiche `n`

Notion d'objet

Un autre programme :

Écrire un programme permettant d'afficher le nombre de caractères contenus dans le mot **vouloir**.

Notion d'objet

Un autre programme :

Écrire un programme permettant d'afficher le nombre de caractères contenus dans le mot **vouloir**.

- ☞ Solution : Utiliser le programme précédent en remplaçant l'objet **travail** par l'objet **vouloir**.

Notion d'objet

Un autre programme :

Écrire un programme permettant d'afficher le nombre de caractères contenus dans le mot **vouloir**.

- ☞ Solution : Utiliser le programme précédent en remplaçant l'objet **travail** par l'objet **vouloir**.
- ☞ Question : Pourquoi peut-on remplacer l'objet **travail** par l'objet **vouloir** ?

Les classes

- Un objet a un type. Par exemple **travail** et **vouloir** sont des mots, leur type est **mot**. **mouton** et **chat** sont des animaux, leur type est **animal**.
- Java appelle le type d'un objet, la classe de l'objet. Donc les objets **travail** et **vouloir** sont de la classe **Mot** et les objets **mouton** et **chat** sont de la classe **Animal**.
- Un objet est aussi appelé instance d'une classe. Par exemple, **travail** est une instance de la classe **Mot**.

Les classes

- Une classe définit un type (d'objet). Elle définit toutes les propriètés et actions des objets dont elle est le type.
- Les propriétés sont appelées **données membres** et peuvent être des données élémentaires (primitifs) ou des objets.
- Les actions sont appelées **méthodes**, ce sont des fonctions.

Les données de type primitif

- Un primitif est une donnée élémentaire de base telles que les nombres entiers, les nombres réels, les caractères et les booléens.
- Java définit 8 types de primitifs :
  - 4 types de nombres entiers
  - 2 types de nombres réels
  - 1 type pour les caractères
  - 1 type pour les booléens

Les données de type primitif

Primitifs	Taille en mémoire	Valeurs	Val. par défaut
`boolean`	1 bit	`true`, `false`	`false`
`char`	16 bits	0 à $2^{16}-1$	0
`byte`	8 bits	-128 à $+127$	0
`short`	16 bits	-2^{15} à $2^{15}-1$	0
`int`	32 bits	-2^{31} à $2^{31}-1$	0
`long`	64 bits	-2^{63} à $2^{63}-1$	0L
`float`	32 bits	$\pm 1.4e^{-45}$ à $\pm 3.4e^{+38}$	0.0f
`double`	64 bits	$\pm 4.9e^{-324}$ à $\pm 1.7e^{+308}$	0.0d

Déclaration et initialisation des primitifs

- Le nom d'une données :
  - commence par une lettre minuscule,
  - les mots à l'intérieur du nom commence par une majuscule,
  - n'utilise que les lettres a-z, A-Z, 0-9, _

Exemple :

```
boolean b = true;
char c = 'a';
byte tresPetitEntier = 032;
short petitEntier = 0xAE;
int i = 1024;
long grandEntier = 1234567890L;
float pi = 3.14f;
double grandReel = 3.5e+15;
```

Les données de type chaine de caractères

- Java représente les chaines de caractères à l'aide de la classe **`String`**.
- Une chaine de caractères est donc un objet de la classe **`String`**.

Exemple :

Déclaration et initialisation de chaines de caractères

```
String nom = "Modou";
String adresse = "St Louis";
String info1 = nom + " habite à " + adresse + ".";
float note = 15.5f;
String info2 = nom + " a " + note + " au devoir";
```

Les méthodes

- Une méthode est une fonction, elle représente une action que peut faire un objet de la classe (ou la classe elle même).
- Une méthode à une valeur de retour, un nom et une liste d'arguments.
- Une méthode utilise le mot clé `return` pour retourner une valeur.
- Le nom et la liste des arguments constituent la signature de la méthode.
- Le nom d'une méthode :
  - commence par une lettre minuscule,
  - les mots à l'intérieur du nom commence par une majuscule,
  - n'utilise que les lettres a-z, A-Z, 0-9, _

Les méthodes

Remarque :

Lorsque la méthode ne retourne rien, son type de retour est **`void`**.

Exemple :

Une méthode qui multiplie 2 entiers :

```
int multiplier (int a, int b){
    int c;
    c = a * b;
    return c;
}
```

Définition d'une classe

- La définition d'une classe se fait avec le mot clé **`class`** suivi du nom de la classe puis des données membres et des méthodes entre accolades.
- Le nom d'une classe :
  - commence par une lettre majuscule,
  - les mots à l'intérieur du nom commence par une majuscule,
  - n'utilise que les lettres a-z, A-Z, 0-9, _

Exemple : Définition de la classe **Mot**

```
class Mot{
    // données membres
    // méthodes
}
```

Constructeur d'une classe

- Le constructeur d'une classe est la méthode qui permet de construire les objets de cette classe en initialisant leurs propriétés.
- Le constructeur a la même structure qu'une méthode ordinaire mais ne dispose pas de type de retour et son nom doit être identique à celui de la classe.
- Lorsque le constructeur ne fait rien, sa définition peut être omise de la classe.
- Une classe peut avoir une ou plusieurs constructeurs.

Exemple : une classe avec un constructeur

```
class Mot{
    String valeur;
    String type;

    Mot(String v, String t){
        valeur = v;
        type = t;
    }

    int donnerNbCaracteres(){
        ...
    }

    void seMettreEnMajuscule(){
        ...
    }

    void seMettreEnMinuscule(){
        ...
    }
}
```

Manipulation d'objets

- Un objet est toujours créé à partir d'une classe.
- La création d'un objet se fait à l'aide du mot clé `new` suivi du constructeur de sa classe.
- Lorsque la définition de la classe ne contient pas d'argument, le constructeur par défaut est utlisé pour créer les objets.
- Le constructeur par défaut d'une classe ne prend pas d'argument et ne contient pas d'instruction.
- Le constructeur par défaut n'est disponible que si la classe ne définit pas de contructeur.
- Lorsqu'un objet est déclaré sans être créé sa valeur est `null`.

Manipulation d'objets

Exemple :

- Création d'un objet de la classe **Mot**

```
Mot mot1;
mot1 = new Mot("travail", "nom masculin");

Mot mot2 = new Mot("vouloir", "verbe transitif");
```

- La variable `mot1` est la référence de l'objet crée avec `new`. Elle fait référence à l'emplacement mémoire où réside l'objet.

Manipulation d'objets

- La référence est le nom l'objet.
- La référence permet d'accéder aux propriétés et aux actions de l'objet.
- L'accès à une propriété ou à une action de l'objet se fait avec l'opérateur point (`.`)

Exemple :

Accéder à la propriété **`valeur`** de l'objet **`mot1`** et à la propriété **`type`** de l'objet **`mot2`**

```
String valeurMot1 = mot1.valeur;
String typeMot2 = mot2.type;
```

Demander à l'objet **`mot1`** de se mettre en majuscule :

```
mot1.seMettreEnMajuscule();
```

Les chaines de caractères

Rappel : Java représente les chaines de caractères à l'aide de la classe **`String`**.

- Une chaine de caractères est alors un objet, elle peut alors être créée avec l'opérateur **`new`**

```
String nom = new String("Modou");
```

- La concaténation de 2 chaines de caractères peut se faire avec l'opérateur **`+`**.
- La classe **`String`** dispose entre autre des méthodes suivantes :
  - **`int length()`** pour avoir le nombre de caractères.
  - **`String toLowerCase()`** pour mettre la chaine de caractères en minuscule.
  - **`String toUpperCase()`** pour mettre la chaine de caractères en majuscule.

Accéder aux méthodes de la classe `String`

Utilisation dans la classe **Mot**

```
class Mot{
    String valeur;
    String type;

    Mot(String v, String t){
        valeur = v;
        type = t;
    }

    int donnerNbCaracteres(){
        return valeur.length();
    }

    void seMettreEnMajuscule(){
        valeur = valeur.toUpperCase();
    }

    void seMettreEnMinuscule(){
        valeur = valeur.toLowerCase();
    }
}
```

Le mot clé `static`

- Le mot clé **`static`** est utilisé pour désigner une propriété dont la valeur est identique pour tous les objets de la classe.
- Le mot clé **`static`** est aussi utilisé pour désigner une méthode qui ne peut accéder qu'aux propriétés **`static`**.
- Une méthode qui accède à une propriété **`static`** doit être déclarée **`static`**.
- Les propriétés et méthodes **`static`** sont accessibles à partir du nom de la classe.
- Les propriétés et méthodes **`static`** sont respectivement appelées propriétés de classe et méthodes de classe.
- Les propriétés et méthodes non **`static`** sont respectivement appelées propriétés d'instance et méthodes d'instance.

Exemple :

Soit la classe suivante décrivant les voitures à 4 roues.

```
class Voiture{
    int nombredeRoues;
    int anneeDeMiseEnCirculation;
    String matricule;
    ...
}
```

Toutes les instances de la classe **Voiture** ont le même nombre de roues, donc la donnée membre **nombredeRoues** peut être déclarée **static**.

```
class Voiture{
    static int nombredeRoues;
    int anneeDeMiseEnCirculation;
    String matricule;
    ...
}
```

Le mot clé `static`

Remarque 1 :

Les données et méthodes **`static`** (de classe) sont accessibles depuis la classe, alors que le données et méthodes non **`static`** (d'instance) ne sont accessibles qu'à travers les objets.

Exemple :

```
Voiture.nombredeRoues = 4; // OK, donnée statique
Voiture.matricule = "DK-2013-AA"; //Erreur, donnée non statique
```

Remarque 2 :

Les méthodes **`static`** ne peuvent pas accéder aux données et méthodes d'instance.

Le mot clé `final`

- Le mot clé **`final`** est utilisé pour désigner une propriété qui ne change pas de valeur.
- Le mot clé **`final`** est aussi utilisé pour désigner une méthode qui ne peut être redéfinie.

Exemple : La donnée membre **`nombredeRoues`** vaut toujours **`4`** quelque soit l'instance de **`Voiture`**.

```
class Voiture{
    static final int nombredeRoues = 4;
    String anneeDeMiseEnCirculation;
    String matricule;
    ...
}
```

Le mot clé `this`

- Le mot clé `this` permet de désigner l'objet courant.
- Il est utilisé pour accéder aux données et méthodes de l'objet courant.

Exemple :

```
class Mot{
    String valeur;
    String type;

    Mot(String valeur, String type){
        this.valeur = valeur;
        this.type = type;
    }

    int donnerNbCaracteres(){
        return valeur.length();
    }
    ...
}
```

Les enveloppeurs de primitifs

- Pour chaque type primitif, Java définit une classe enveloppe permettant de le manipuler comme un objet.
- Les enveloppeurs de primitifs disposent de méthodes permettant la conversion du primitif en chaine de caractères et vise-versa.
- Toutes ces méthodes de conversion sont **`static`**.

Primitifs	Enveloppe	Conversion primitif → String	Conversion String → primitif
boolean	**Boolean**	**toString(boolean b)**	**parseBoolean(String s)**
char	**Character**	**toString(char c)**	**-**
byte	**Byte**	**toString(byte b)**	**parseByte(String s)** **parseByte(String s, int base)**
short	**Short**	**toString(short s)**	**parseShort(String s)** **parseShort(String s, int base)**

Les enveloppeurs de primitifs

Primitifs	Enveloppeur	Conversion primitif → String	Conversion String → primitif
int	**Integer**	**toString(int i)**	**parseInt(String s)**
		toString(int i, int base)	**parseInt(String s, int base)**
long	**Long**	**toString(long i)**	**parseLong(String s)**
		toString(long i, int base)	**parseLong(String s, int base)**
float	**Float**	**toString(float f)**	**parseFloat(String s)**
double	**Double**	**toString(double d)**	**parseDouble(String s)**

Exemple :

```
String nombre = "314";
int n = Integer.parseInt(nombre); // n = 314
```

Créer un programme Java

- Un programme Java peut être constitué d'une ou de plusieurs classes.
- Le code source de chaque classe est enregistré dans un fichier texte ayant le même nom que la classe avec l'extension ".java"
- Un programme Java doit obligatoirement avoir une classe principale dont le nom sera celui du programme.
- La classe principale doit contenir la méthode principale **`main()`** avec la signature suivante :

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

- La méthode principale est le point d'entré du programme.

Créer un programme Java

Exemple : Créer le programme permettant d'afficher le nombre de caractères contenus dans le mot **travail**

1. La classe **Mot** (voir page 23) sera enregistrée dans le fichier "Mot.java"
2. La classe principale nommée `Programme` sera enregistrée dans le fichier "Programme.java"

```
class Programme {
   public static void main(String[] args){
       Mot mot = new Mot("travail", "nom masculin");
       int n = mot.donnerNbCaracteres();
       System.out.println(n);
   }
}
```

Remarque : L'objet écran est représenté par `System.out`

Compilation

- Pour compiler un programe Java, l'environnement de programmation Java (jdk) doit être installé.
- Le chemin d'accès (path) aux binaires du jdk doit être configuré.
- La compilation se fait en ligne de commande avec :

```
javac Programme.java
```

- Si la compilation se fait sans erreur, des fichiers bytecode sont produits : **`Mot.class`** et **`Programme.class`**

Exécution

- L'environnement de programmation Java (jdk) doit être installé
- Le chemin d'accès (path) aux binaires du jdk doit être configuré.
- L'exécution se fait en ligne de commande avec :

```
java Programme
```

- Le nombre de caractères du mot **travail** est affiché.