



IMT Atlantique

Bretagne-Pays de la Loire
École Mines-Télécom

Les objets en Java

F. Dagnat & J. Mallet
Diaporama – UE-IR-S6 –
Session 1
1^{er} semestre 2025

- 1 Java
- 2 Compiler et exécuter en Java
- 3 Instanciation d'objets
- 4 Manipulations des références
- 5 Un exemple complet

- 1 Java
- 2 Compiler et exécuter en Java
- 3 Instanciation d'objets
- 4 Manipulations des références
- 5 Un exemple complet

- ▶ Langage objet
- ▶ Typage fort
 - ▶ Les objets, variables sont déclarés avec un type
 - ▶ et leur utilisation est vérifiée par le compilateur
- ▶ Pas de mélange entre programmation objet et procédurale
Un calcul est réalisé par un objet (ou une classe - *static*)
- ▶ Support pour la documentation : Javadoc
- ▶ De très nombreuses bibliothèques
 - ▶ Apprendre un langage objet, c'est apprendre sa syntaxe. . .
 - ▶ et connaître ses bibliothèques de code

```
1 class BankAccount {
2     // Des attributs
3     String first;
4     String last;
5     float balance;
6     // Un constructeur
7     BankAccount(String first,String last,float balance) {
8         ...
9     }
10    // Des méthodes
11    float getBalance() {
12        ...
13    }
14    void deposit(float amount) {
15        ...
16    }
17 }
```

- ▶ déclaration de variable et assignation : `type variable = value;`
- ▶ déclarations de fonctions

```
returnType functionName([comma-separated Type name pairs]) {  
    ...[corps]...  
}
```

- ▶ appels de fonctions : `functionName([comma-separated parameters])`
... souvent `???.functionName([comma-separated parameters])`



Document des bases de la syntaxe impérative java :

<https://hub.imt-atlantique.fr/ueinfo-fise1a/fr/s6/prog/session1/java-syntax/>

- ▶ **Rappel S5** : L'encapsulation assure que l'état d'un objet (attributs) n'est accessible que depuis l'objet lui-même (ses méthodes)
- ▶ Encapsulation pas tout à fait respectée en Java
- ▶ Chaque attribut ou méthode peut contenir un modificateur de visibilité :

Java	accessible par
<code>private</code>	les objets de même classe
<code>public</code>	tous (pas d'encapsulation)

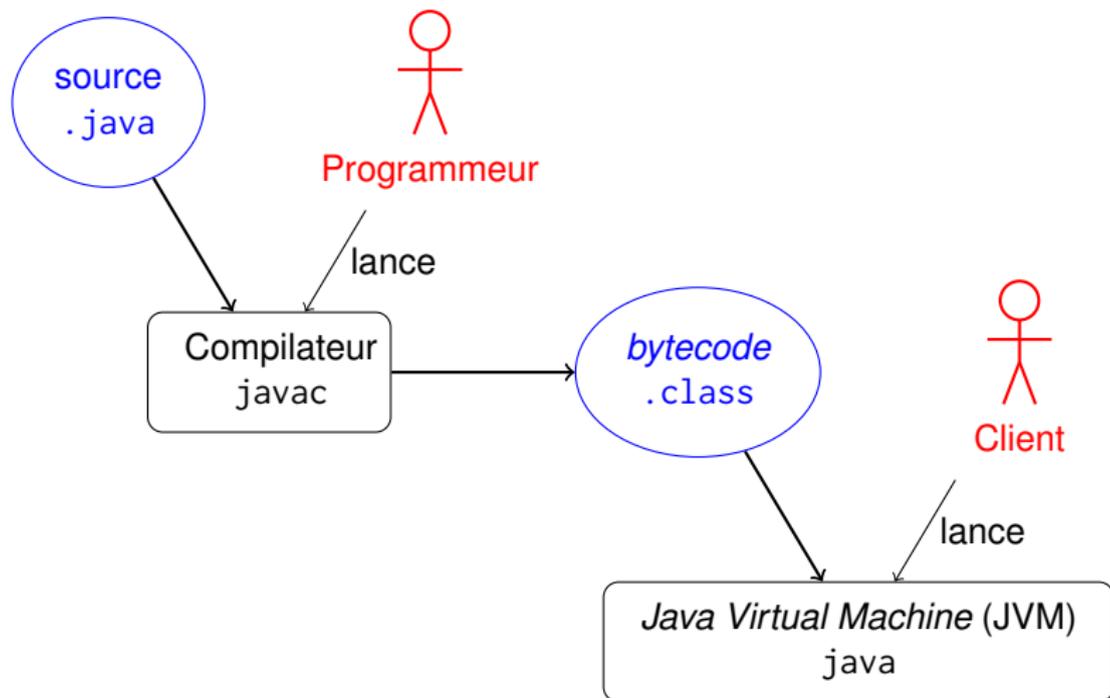
- ▶ Il faut des règles de bonne pratique :
 - ▶ Utiliser au maximum des attributs privés
 - ▶ Ne mettre publique que les méthodes qui doivent l'être

Quand un calcul ne dépend pas d'un objet (fonction mathématique ou constante, par exemple.)

- ▶ Modificateur **static**
- ▶ Par ex : `Math.PI`
- ▶ Par ex : `System.exit(0);, Math.random(); ...`
- ▶ Par ex :

```
1 class BankAccount {
2     // Des attributs
3     ...
4     static float minBalance;
5     // Des méthodes
6     static void setMinBalance(float min) {
7         BankAccount.minBalance = min;
8     }
9 }
```

- 1 Java
- 2 Compiler et exécuter en Java**
- 3 Instanciation d'objets
- 4 Manipulations des références
- 5 Un exemple complet



- ▶ Une classe publique peut être exécutée
- ▶ La JVM lance alors l'exécution de sa méthode `main`
- ▶ Cette méthode doit suivre la définition suivante

```
1 public static void main(String[] args) {  
2     ...  
3 }
```

Seul le nom de l'argument peut être modifié

- 1 Java
- 2 Compiler et exécuter en Java
- 3 Instanciation d'objets**
- 4 Manipulations des références
- 5 Un exemple complet

- ▶ Les objets ne sont jamais manipulés directement :
 - ▶ ils sont créés et stockés dans une zone mémoire spécifique (le tas)
 - ▶ ils sont accédés par des **références** (des adresses)
- ▶ Les types primitifs (**int**, **boolean**, ...) sont manipulés directement.
- ▶ Les tableaux sont manipulés également par référence

- ▶ Des méthodes spécifiques : les **constructeurs**
- ▶ Même nom que la classe et pas de type de retour :

```
1     NomClasse(Type1 nomVar1, Type2 nomVar2) { ... }
```
- ▶ Appelés uniquement à la création d'une nouvelle instance
- ▶ But : initialiser les attributs du nouvel objet

1. création de la structure en mémoire (tas), allocation

1. création de la structure en mémoire (tas), allocation
2. **initialisation des attributs**

1. création de la structure en mémoire (tas), allocation
2. initialisation des attributs
3. exécution du constructeur spécifié

1. création de la structure en mémoire (tas), allocation
2. initialisation des attributs
3. exécution du constructeur spécifié

1. création de la structure en mémoire (tas), allocation
2. initialisation des attributs
3. exécution du constructeur spécifié
4. renvoi d'une référence sur l'objet

- 1 Java
- 2 Compiler et exécuter en Java
- 3 Instanciation d'objets
- 4 Manipulations des références**
- 5 Un exemple complet

- ▶ La référence sur rien `null` qui est de tous les types références
- ▶ Utiliser une référence nulle (valant `null`) lève l'exception `NullPointerException` :
- ▶ ⇒ Pensez à tester une référence reçue en paramètre avant de s'en servir. Précondition `arg != null` :

```
1 public void transfer(float amount, BankAccount target) {  
2     if (target != null && amount < balance) {  
3         balance -= amount;  
4         target.deposit(amount);  
5     }  
6 }
```

- ▶ La comparaison `==` compare les valeurs
 - ▶ sur les types primitifs compare bien la valeur
 - ▶ sur les types références compare les adresses pas les contenus
- ▶ Méthode `equals` (voir plus tard)

- 1 Java
- 2 Compiler et exécuter en Java
- 3 Instanciation d'objets
- 4 Manipulations des références
- 5 Un exemple complet**

```
1 public class BankAccount{
2     private String first;
3     private String last;
4     private float balance;
5     private static float minBalance = 0;
6     public BankAccount(String first,String last,float balance){
7         this.first = first;
8         this.last = last;
9         this.balance = balance;
10    }
11    public float getBalance(){
12        return balance;
13    }
14    public String getOwner(){
15        return first + " " + last;
16    }
17    public void setOwner(String first,String last){
18        this.first = first;
19        this.last = last;
20    }
21    static void setMinBalance(float min) {
22        minBalance = min;
23    }
24    public void deposit(float amount){
25        this.balance += amount;
26    }
27    public void withdraw(float amount){
28        if (this.balance - amount > minBalance) this.balance -= amount;
29    }
30    public static void main(String[] a){
31        BankAccount b = new BankAccount("julien", "mallet",100);
32        System.out.println(b.getOwner()+" with "+b.getBalance());
33        b.deposit(100);
34        b.withdraw(150);
35    }
36 }
```