Premiers pas en java

Arnaud Labourel arnaud.labourel@univ-amu.fr

18 janvier 2022

Section 1

Bienvenue dans le cours de programmation 1

Organisation du cours

Volume horaire

- 12 séances de cours d'1h30 (18h au total)
- 12 séances de travaux dirigés d'1h30 (18h au total)
- 12 séances de travaux pratiques de 2h (24h au total)

Évaluation

- un examen sur papier en mai (60% de la note)
- des rendus de travaux pratiques (40% de la note)

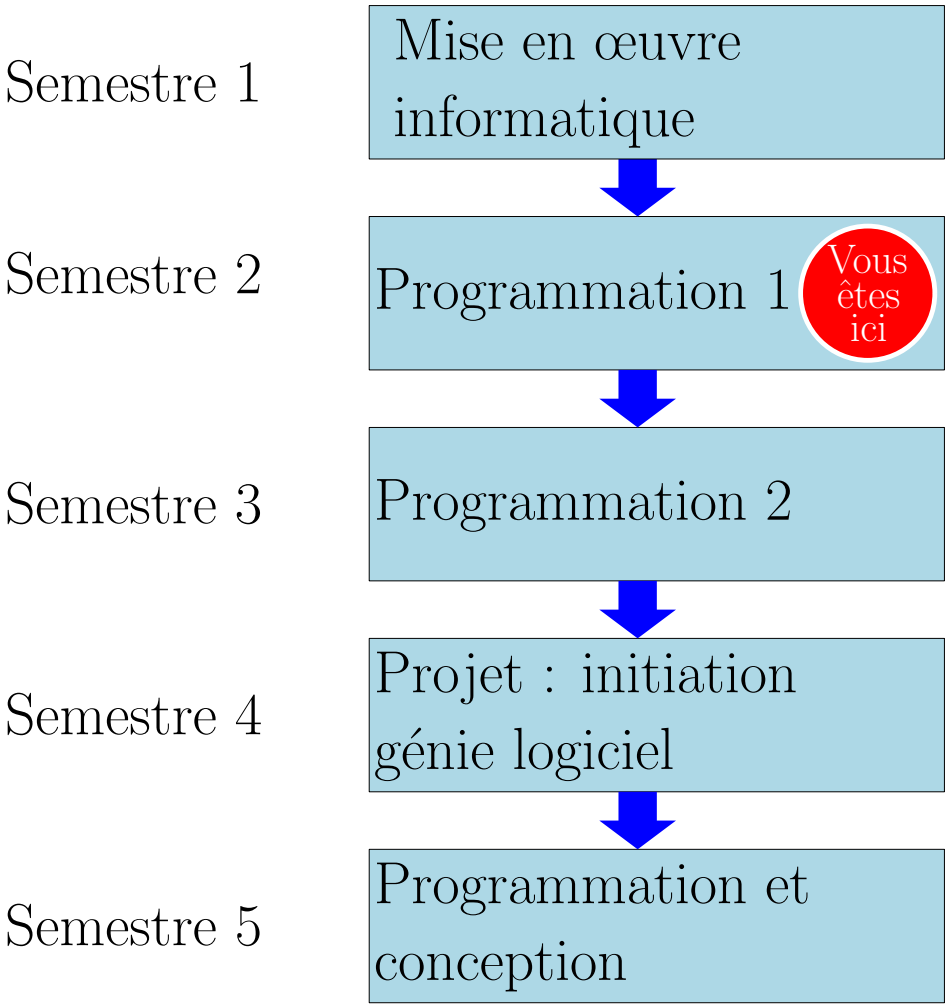
L'équipe pédagogique

- Arnaud Labourel (CM, TD 1, TP 1)
- Karim Nouioua (TD 2, TP 2 et 4)
- Julien Roubertie (TD 3, TP 3)

Objectifs du cours

- Apprendre à coder proprement
- Apprendre ce que permet le langage Java et ces différences avec Python
- Apprendre à programmer avec des objets
  - adopter le “penser objet”
  - connaître et savoir mettre en œuvre les concepts fondamentaux de la programmation objet
- Préparer aux cours de **Programmation 2** du semestre 3 (de licence informatique et mathématiques)

Positionnement du cours dans la licence informatique



Section 2

Contenu du cours

Objectif de la formation : s'initier à toutes les étapes du développement de logiciels

1. **Analyser** les besoins
2. **Spécifier** les comportements du programme
3. **Choisir** et éventuellement **concevoir** les solutions techniques
4. **Implémenter** le programme (coder)
5. **Vérifier** que le programme a le comportement spécifié (tester)
6. **Déployer** le programme dans son environnement, **fournir** une documentation dans le cas de bibliothèque
7. **Maintenir** le programme (corriger les bugs, ajouter des fonctionnalités)

Apprendre à programmer proprement

Un programme propre :

- respecte les attentes des utilisateurs
- est fiable
- peut évoluer facilement/rapidement
- est compréhensible par tous

Méthode pour programmer proprement

- nommer correctement les éléments du code
- écrire du code lisible (par un autre humain)
- relire et améliorer le code

S'initier au penser objet

Connaître les éléments de base de la programmation objet :

- maîtriser le vocabulaire de la programmation objet : *classe*, *instance*, *méthode*, *attribut*, *constructeur*, *interface*, ...
- être capable d'utiliser les objets des classes du JDK (Java Development Kit)
- être capable de coder de nouvelles classes pour définir des nouveaux types d'objets
- savoir décomposer un problème simple en classes et objets
- savoir tester le comportement d'objets

Section 3

Premiers pas en Java et différences avec Python

Python

Lors du premier semestre, vous avez codé en Python dans le cours de mise en œuvre informatique :

Exemple de code Python :

```
def is_even(n):
    return n % 2 == 0
```

```
>>> is_even(10)
True
```

```
>>> is_even("toto")
...
TypeError: not all arguments converted during
  string formatting
```

Erreur à l'exécution car `n % 2` n'est pas possible avec `n="toto"`

Java

```java
static boolean isEven(int n){
  return n % 2 == 0;
}
```

En java, il faut définir le type des paramètres

```
>>> isEven(10)
True

>>> isEven("toto")
Error:
|  incompatible types: java.lang.String cannot be
|  converted to int isEven("toto")
```

En cas de type incompatible donné en argument, une erreur se produit avant l'exécution de la fonction.

Type

Définition d'un type

Un type de données définit :

- l'ensemble des valeurs possibles pour les données du type
- les opérations applicables sur ces données

Java vs Python

- Python :
  - toute donnée/valeur a un type
  - les variables/paramètres **n**'ont **pas** de types
- Java :
  - toute donnée a un type, toute variable a un type
  - les variables/paramètres ont un type

Typage dynamique

En Python, le type d'une valeur d'une variable est défini lors de son affectation, et le type de la valeur de la variable peut changer au cours d'une exécution.

```
>>> x = 8
>>> x
8
>>> type(x)
<class 'int'>
>>> x = "toto"
>>> type(x)
<class 'str'>
```

$\Rightarrow$ Le typage est dit **dynamique**

Typage statique

En Java, les variables ont chacune un type défini à sa déclaration.

Il détermine les valeurs que peut prendre la variable et ce type ne peut pas changer.

Le compilateur vérifie le typage.

```
int x = 8;
x; /* 8 */
x = "timoleon"; // interdit ! ne compile pas
```

$\Rightarrow$ Le typage est dit **statique**

Résumé typage

Java est un langage **typé statiquement** :

- les variables, paramètres et retours de méthode ont des types
- une vérification de compatibilité des types est effectué à la compilation

Python est un langage **typé dynamiquement** :

- les variables, paramètre et retour de méthode **n'**ont **pas** de types
- les valeurs ont des types
- des erreurs liés au types se produisent à l'exécution

Analyse du typage statique

Avantages du typage statique

- séparation entre erreurs de types à la compilation et autres erreurs à l'exécution
- aide à la programmation (surtout avec la complétion automatique)
- aide à l'utilisation de bibliothèque (les types aident à comprendre l'usage des fonctions)
- exécutable généralement plus rapide (code plus facile à optimiser)

Désavantages du typage statique

- verbeux (code plus long à écrire)
- généricité plus difficile (difficile de définir des fonctions acceptant plusieurs types)
- code plus difficile à modifier

Autre exemple Python

```
def even_integers(bound):
    list_even_integers = []
    for i in range(0, bound):
        if is_even(i):
            list_even_integers += [i]
    return list_even_integers
```

```
>>> print(even_integers(10))
[0, 2, 4, 6, 8]
```

Même exemple en Java

```
static List<Integer> evenIntegers(int bound) {
  List<Integer> evenIntegers = new ArrayList<>();
  for(int i = 0; i < bound; i++){
    if(isEven(i)){
      evenIntegers.add(i);
    }
  }
  return evenIntegers;
}
```

```
>>> System.out.println(evenIntegers(10));
[0, 2, 4, 6, 8]
```

Points communs Java/Python

- concept de fonctions similaire :
  - une fonction est définie par un bloc d'instruction
  - une fonction a des paramètres
  - une fonction peut retourner une valeur (instruction `return` similaire dans les deux langages)
- concept de variables/arguments similaire (sauf au niveau du typage)
- concept de bloc d'instructions similaire
- concept de structures de contrôle similaire : `for`, `if`, `while` présents en Java et Python

Différences Java/Python

- Même si les concepts sont similaires, la syntaxe (manière d'écrire est souvent différente) :
  - les blocs en pythons sont définis par l'indentation alors qu'en java il le sont par les accolades `{ }`.
  - instructions séparées par des `;` en java.
- Typage statique vs typage dynamique $\Rightarrow$ il faut préciser le type des éléments d'une liste en Java.
- Très peu d'opérateurs disponibles sur les objets en Java
  - pas de `+` sur des listes
  - pas de `*` sur les `String` (mais quand même le `+`)
- Les `for` sur les entiers *old school* en Java.
- Normes de nommage différentes pour les variables/arguments/fonctions :
  - `snake_case` en Python
  - `camelCase` en Java