**Université Paris Cité** **Programmation Orientée Objet**
**L2** **Année 2024-2025**

# TP n° 1

# Révisions : Java et objets

**Inscription sur Moodle**

N'oubliez pas de vous inscrire sur le Moodle du cours, `https://moodle.u-paris.fr/course/view.php?id=1631` ! Les groupes sont créés, pensez par conséquent à vous inscrire dans votre groupe. Si vous n'avez pas le même groupe de TD et TP, inscrivez-vous dans les deux.

**Documentation**

**Java 22 :** `https://docs.oracle.com/en/java/javase/22/docs/api/`

**Antisèche :** `https://introcs.cs.princeton.edu/java/11cheatsheet/`

**Installer le java development kit (jdk) sur votre machine**

Suivez les instructions que vous trouverez ici : `https://docs.oracle.com/en/java/javase/22/install/overview-jdk-installation.html`.

**Sur Linux Ubuntu, Debian :** La façon la plus facile est d'installer la version fournie par apt. Si cette version est au moins la 11, c'est bon.

- Ouvrez un terminal et mettez le système à jour avec `sudo apt update` et `sudo apt upgrade`.
- Pour voir si Java est déjà installé, tapez `java -version`. Si un jdk est déjà present, vous pouvez ignorer l'étape suivante.
- Executez `sudo apt install default-jdk`.

## 1 Révisions

Voir l'annexe (p. 6) pour des instructions sur l'utilisation d'Eclipse et du terminal pour écrire un programme java.

## Exercice 1

**Arguments et méthode main**

La méthode `main` est le point de départ du programme. Elle prend en entrée des arguments sous la forme d'un unique tableau `args` de chaînes de caractères. Par exemple, depuis la ligne de commande, l'instruction suivante

```
>java MonProgramme Il fait beau
```

va créer le tableau d'arguments `{"Il","fait","beau"}`.

On peut passer des arguments à la méthode `main` directement avec Eclipse (sans passer par la ligne de commande) en utilisant le menu `Run/Run Configurations/Arguments/Program arguments`.

Utilisez si nécessaire la documentation Java (notamment l'outil « search ») pour répondre aux questions suivantes :

1) Quel est le type de sortie de la méthode `main` ?
2) Écrire un programme qui affiche le nombre d'arguments donnés en entrée, et le tester avec au moins un argument.

   **Rappel :** L'attribut `length` permet d'obtenir la longueur d'un tableau.
3) Trouver une méthode dans la classe `String` pour calculer la taille d'une chaine de caractères.
4) Trouver à quelle classe Java appartient la méthode `hypot`.

## Exercice 2

**Parcourir un tableau**

Pour parcourir un tableau `int [] t`, utiliser un `for` ou un `foreach`.

```
for (int i = 0 ; i < t.length ; i++) { /* ... */ } \\ for
for (int i : t) { /* ... */ } \\ foreach
```

1) Écrivez une méthode `void affiche(int [] t)` qui prend en entrée un tableau d'entiers `t` et l'affiche.
2) Écrivez une méthode `int [] multiplication(int [] t1,int [] t2)` qui prend en entrée deux tableaux d'entiers `t1` et `t2` (de taille $n_1$ et $n_2$) et retourne un tableau `t3` de taille $\max(n_1, n_2)$ obtenu en multipliant `t1` et `t2` case par case. Si un tableau est plus petit que l'autre on remplira les cases manquantes du plus petit tableau avec la valeur 1.

   **Exemple :** Si `t1={1,3,6,7}` et `t2={2,4,6}` alors `t3={2,12,36,7}`.
3) Trouvez dans la documentation Java les méthodes suivantes :
   i) `String.valueOf(n)` qui transforme l'entier `n` en une chaîne de caractères ;

ii) `Character.getNumericValue(c)` qui transforme le caractère `c` en une valeur numérique.

4) Utilisez ces méthodes pour écrire une méthode `int [] split(int [] t)` qui prend en entrée un tableau d'entiers `t` et retourne le tableau obtenu en séparant chacun des chiffres.
**Exemple :** Si `t={2,12,36,7}` il faut retourner `t={2,1,2,3,6,7}`.

## Exercice 3

Écrivez un programme `Shift` qui récupère les chaînes de caractères données en argument de la ligne de commande et remplace chaque voyelle par la précédente dans l'ordre alphabétique (« a » sera remplacée par « y »).

**Exemple d'exécution dans le terminal :**

```
>java Shift Portez ce vieux whisky au juge blond qui fume
Pirtaz ca veaox whesku yo joga blind qoe foma
```

Vous pourrez utiliser la méthode `char charAt(int index)` de la classe `String`. En cas de besoin, cherchez-la dans la documentation Java pour comprendre son utilisation.

**Lecture au clavier**

Voici un exemple de lecture au clavier :

```
import java.util.Scanner; // utiliser la bibliothèque

public class LireEntiers{
   public static void main(String [] args) {
      // opère sur l'entrée système correspondant au clavier
      Scanner sc = new Scanner(System.in);
      // question à l'utilisateur
      System.out.print("Donnez un entier : ");
      // lecture de la reponse en interrogeant le scanner
      int r = sc.nextInt();
      System.out.println("Vous avez donné l'entier "+ r);
   }
}
```

Vous trouverez dans la documentation Java les méthodes qui s'appliquent aux scanners. Vous pourrez avoir besoin en particulier de `nextDouble()`, `nextBoolean()`, `next()`, etc.

**Générer des nombres aléatoires**

Il y a deux façons de générer des nombres aléatoires :

1. Appeler la méthode `static double Math.random()`. Cette méthode retourne un nombre réel aléatoire compris entre 0 (inclus) et 1 (exclu). Vous pouvez obtenir un intervalle plus grand en multipliant la valeur obtenue par un nombre adéquat. Pour obtenir la partie entière inférieure d'un `double x` il suffit d'utiliser la conversion de type `(int) x`.
2. Utiliser un objet de type `Random`, plus polyvalent, dont voici un exemple d'utilisation.

```
import java.util.Random; // utiliser la bibliothèque

public class NombresAlea{
   public static void main(String [] args) {
      // créer un objet de type Random
      Random rd = new Random();
      // produire un nombre entier aléatoire r, 0 <= r < 10
      int r = rd.nextInt(10);
      System.out.println("Nombre créé "+ r);
   }
}
```

### Exercice 4

1) Écrivez une méthode `int question(int maxTentatives)` qui choisit deux nombres aléatoires entre 1 et 9 et demande à l'utilisateur le résultat de leur multiplication. Si l'utilisateur se trompe, la méthode repose la question (au plus `maxTentatives` fois au total) puis retourne le nombre d'erreurs commises.

2) Avec la méthode précédente, écrivez une méthode `int evaluation(int n)` qui interroge l'utilisateur `n` fois sur ses tables de multiplication, puis retourne une note entre 0 et `n` (chaque mauvaise réponse fait perdre un point).

## 2 Un peu d'objet

### Exercice 5

On souhaite modéliser un jeu de combats entre plusieurs personnages. Chaque personnage est caractérisé par un nom, des informations sur son état initial, et des informations sur son état actuel. L'état initial et l'état actuel sont des d'informations de même nature : elles renseignent sur la vitalité, la force, et l'agilité du personnage. Seul l'état actuel va évoluer au cours des combats.

1) Créez une classe `Informations` avec deux constructeurs `Informations(int v,int f,int a)` et `Informations(Informations inf)`, trois accesseurs `int getVitalite()`, `int getForce()`, `int getAgilite()` et les modifieurs correspondant. Ajoutez une méthode `String toString()`.

   **Remarque :** la méthode `toString()` n'est pas une méthode comme les autres, elle a un comportement prédéfini pour tous les objets Java et elle est appelée implicitement dès qu'il faut convertir un objet en chaîne de caractère (par exemple lorsqu'on l'affiche). Sa signature est nécessairement `public String toString()`.

2) Créez une classe `Personnage` avec les attributs appropriés : un nom, des `Informations etatInitial` et `etatActuel`, un constructeur, une méthode `String toString()`, une méthode `boolean estVivant()` qui retourne `true` lorsque la vitalité du personnage est supérieure à 0, et une méthode `void rebirth()` qui restore l'état actuel du personnage à son état initial.

3) Ajoutez dans la classe `Personnage` une méthode `void attaque(Personnage def)` pour permettre à un personnage d'en attaquer un autre (le défenseur `def`). L'attaque se déroulera ainsi : un nombre $n$ est tout d'abord choisi aléatoirement entre 1 et

$$\max(1, \texttt{forceAttaquant} - \texttt{forceDefenseur}).$$

   Si le défenseur est moins agile que l'attaquant, sa vitalité est réduite de $n$. S'il est plus agile, sa vitalité est réduite de $n/2$ mais son agilité est diminuée d'un tiers. Arrondissez toujours à l'entier inférieur.

4) Une lutte est une séquence d'attaques et de ripostes qui s'achève lorsqu'un des combattants a une vitalité inférieure ou égale à 0. Proposez deux implémentations différente de la lutte (une itérative et une récursive), en ajoutant deux méthodes dans la classe `Personnage` (elles afficheront l'évolution de l'état de chaque personnage).

5) Créez une classe `Combat` pour tester votre code. Cette classe doit créer deux personnages `p1` et `p2` et les mettre en lutte, `p1` attaquant `p2` en premier.

# Annexe

**Utiliser le terminal**

— Utilisez un éditeur de texte (« gedit », « atom », etc.) pour créer par exemple une classe `HelloWorld` dans un fichier `HelloWorld.java` :

```
public class HelloWorld {
  public static void main(String[] args) {
      System.out.println("Hello, World");
  }
}
```

— Compiler depuis le terminal avec `javac HelloWorld.java`

— Exécuter le fichier `HelloWorld.class` qui vient d'être créé avec `java HelloWorld`.

**Utiliser Eclipse**

— Ouvrez Eclipse depuis un terminal avec `eclipse &`, ou sélectionnez le menu `Demarrer/Developpement/Eclipse`.

— À la première exécution, vous devez déterminer l'emplacement de votre espace de travail (le dossier où seront stockés vos projets).

— Paramétrer Eclipse pour fonctionner avec Java en sélectionnant la perspective `Java` dans le menu `Window/Perspective/Open perspective`.

— Créez un nouveau projet pour chaque exercice en utilisant le menu `File/New/Java Project`.

— Dans la fenêtre « New module-info.java », choisir « Don't create ».

— Ajoutez des classes au projet avec le menu `File/New/Class` (clic droit sur le package « default »).

— Exécuter le programme en cliquant sur le bouton vert avec une flèche blanche ou en utilisant le menu `Run/Run As/Java Application`.

**Des outils en ligne prets à l'emploi**

— Vous trouverez ici un compilateur java en ligne : `https://repl.it/languages/java10`.

— Un autre outil en ligne pour visualiser l'exécution de votre code pas à pas se trouve ici : `https://cscircles.cemc.uwaterloo.ca/java_visualize/`.

Ces outils n'offrent qu'une solution temporaire et ne remplacent pas votre version java installée !