

## 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.

#### Rendu

Chaque TP doit être rendu sur Moodle à la fin de la séance de TP. Vous pourrez encore faire des modifications jusqu'au début de la séance suivante.

- Consignes : programmez chaque exercice dans un fichier différent. Veillez à **indenter**, **commenter** et **tester** votre code. Suivez les **conventions de codage**, regardez par exemple <https://openclassrooms.com/forum/sujet/conventions-de-codage-en-java-64171>.
- Format de rendu : une unique archive contenant tous vos fichiers `.java` (pas de fichier `.class`). Pour créer une archive `MonArchive.tar` à partir d'un dossier `MonCode` utilisez la commande Linux :  
`tar cvf MonArchive.tar MonCode`
- Nom du fichier : indiquez `NomPrenom.tar`, ou `Nom1Prenom1_Nom2Prenom2.tar` si vous avez travaillé en binôme (dans ce dernier cas, ne soumettre votre travail qu'une seule fois).
- Adresse de dépôt : <https://moodle.u-paris.fr/course/view.php?id=1631>.
- Evaluation : deux TP choisis au hasard durant le semestre seront notés.

#### Documentation

- Documentation Java 14 : <https://docs.oracle.com/en/java/javase/14/docs/api/>
- Antisèche : <https://introcs.cs.princeton.edu/java/11cheatsheetsud>

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

Suivez les instructions que vous trouverez ici : <https://docs.oracle.com/en/java/javase/14/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 présent, vous pouvez ignorer l'étape suivante.
- Exécutez `sudo apt install default-jdk`.

# 1 Révisions

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

## Exercice 1

Utilisez la documentation Java (l'outil "search") pour répondre aux questions suivantes :

1. Trouver la page d'aide de la classe `Main`. Quel est le type d'entrée et le type de sortie de la méthode `main` ?

**Rappel :** la méthode `main` est le point de départ du programme. Elle prend en entrée les arguments donnés via la ligne de commande, ou via Eclipse grâce au menu `Run/Run Configurations/Arguments/Program arguments`.

2. Ecrire un programme qui affiche le nombre d'arguments donnés en entrée (indice : attribut `length` du tableau d'arguments).
3. Trouver une méthode dans la classe `String` pour calculer la taille d'une chaîne de caractères.
4. À quelle classe appartient la méthode `hypot` ? Ecrivez un programme qui l'utilise.

## Exercice 2

### Parcourir un tableau

```
Pour parcourir un tableau int [] t, utiliser soit une boucle for :  
for (int i = 0 ; i < t.length ; i++) { /* ... */ }, soit un foreach :  
for (int i : t) { /* ... */ }.
```

1. Ecrivez une méthode `void affiche(int t[])` qui prend en entrée un tableau d'entiers `t` et l'affiche.
2. Ecrivez 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 remplacera les cases manquantes par 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 `String.valueOf(n)`, qui transforme l'entier `n` en une chaîne de caractères, et `Character.getNumericValue(c)`, qui retransforme 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

#### Arguments et méthode main

Lors de l'exécution d'un programme Java, les valeurs données en arguments à la méthode `main` sont stockées sous la forme d'un tableau `args` de chaînes de caractères. Par exemple,

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

va créer le tableau `{"Il","fait","beau"}`. Ici, `args[2]` correspond à "beau".

Sous Eclipse, pour donner des arguments à la méthode `main`, utilisez le menu `Run/Run Configurations/Arguments/Program arguments`.

Ecrivez un programme `Shift` qui récupère la chaîne de caractères donnée en argument de la ligne de commande et remplace chaque voyelle par la suivante dans l'ordre alphabétique ('y' sera remplacé par 'a').

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

```
>java Shift Il fait beau aujourd'hui, j'y vais !
Ol feot biey eyjuyrd hyo, j a veos !
```

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.

### Exercice 4

#### 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) {
        // le scanner opérera 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();
        // vérification
        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()` ; 2) utiliser un objet de type `Random`.

La méthode `static double Math.random()` retourne un nombre réel aléatoire compris entre 0 (inclu) 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`. Voici un exemple d'utilisation d'un générateur de nombres aléatoires :

```
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);
        // afficher à l'écran
        System.out.println("Nombre créé "+ r);
    }
}
```

1. Ecrivez une méthode `int question()` qui choisit deux nombres aléatoires entre 1 et 9 et demande à l'utilisateur le résultat de leur multiplication. La méthode doit retourner le nombre de tentatives qui ont été nécessaires pour trouver la bonne réponse.
2. Ecrivez une méthode `int evaluation(int n)` qui interroge l'utilisateur `n` fois sur ses tables de multiplication, puis retourne une note entre 0 et 20 (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 pré-défini 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 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 une méthode `void attaque(Personnage def)` dans la classe **Personnage** pour permettre à un personnage (l'attaquant) 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  $\max(1, \text{forceAttaquant} - \text{forceDefenseur})$ . 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.
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 différentes de la lutte (une itérative et une récursive), en ajoutant deux méthodes dans la classe **Personnage** (les méthodes afficheront également les impacts portés).
5. Créez une classe **Combat** pour tester votre code. Cette classe doit créer deux personnages `p1` et `p2` et les mettre en lutte, le `p1` attaquant le `p2` premier.

## Annexe

### Utiliser le terminal

- Utilisez un éditeur de texte, comme **gedit** ou **atom**, 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");  
    }  
  
}
```

- La compilation s’effectue ensuite depuis le terminal avec `javac HelloWorld.java`
- Pour exécuter le fichier `HelloWorld.class` qui vient d’être créé, entrez `java HelloWorld`.

### Utiliser Eclipse

- Ouvrez Eclipse depuis un terminal avec `eclipse &`, ou sélectionnez le menu `Démarrer/Developpement/Eclipse`.
- À la première exécution, vous devez déterminer l’emplacement de votre espace de travail (le dossier où seront stockés vos projets).
- Il faudra éventuellement 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 pop-up “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”).
- Pour exécuter le programme, utilisez le menu `Run/Run As/Java Application` ou cliquer sur le bouton vert avec une flèche blanche.

### Des outils en ligne prêts 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/](https://cscircles.cemc.uwaterloo.ca/java_visualize/).

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