# Instructions simples et introduction aux outils de développement Java

Dans le TD, nous travaillons sur la traduction en Java des exercices de la partie 3. Dans le TP, nous étudions deux manières différentes d'effectuer du développement en Java : à l'aide de l'IDE Eclipse, qui vous aidera notamment à corriger les erreurs mais vous masque la réalité des répertoires et des outils de base (compilateur et interprète), puis à l'aide d'un éditeur de texte standard et du JDK (commandes en ligne pour le compilateur et l'interprète).

## 1 Eclipse

### 1.1 Eclipse : quelques manipulations de base

Eclipse est un environnement de développement gratuit, écrit en Java, et dédié à Java, ainsi qu'à d'autres langages grâce à la notion de plugin que nous n'aborderons pas ici. Nous indiquons ici des manipulations de base. Si vous connaissez déjà Eclipse, lisez ce qui suit et assurez-vous que vous connaissez toutes les manipulations indiquées. Eclipse existe dans plusieurs versions, donc ce document peut comporter de légères différences avec la version que vous utilisez.

### 1.2 Lancer Eclipse

Lancez Eclipse par le menu Application/Développement de votre bureau (clic droit souris). Vous êtes invité à préciser votre workspace. Le workspace correspond à l'espace de travail d'Eclipse, c'est en fait un répertoire où seront notamment stockés vos fichiers sources. Vous pouvez avoir plusieurs workspaces si vous le souhaitez.

#### 1.2.1 Création d'un projet

2 solutions :

1. Dans le package explorer (partie gauche de la fenêtre appelée explorateur de paquetages) : clic droit → new → project
2. Dans le menu : File → New → Project

Un wizard s'ouvre.

— Choisissez Java Project (Next)
— Nommez le projet (*e.g.* TP1). Eclipse crée dans le workspace un répertoire de ce nom.
— Garder la case Use Default Location cochée
— Choisir Create Separate folders for sources and class files, afin de séparer vos fichiers sources (d'extension .java) des fichiers de bytecode (d'extension .class). Eclipse va créer dans votre projet un répertoire src et un répertoire bin
— Dans la partie `JRE`, vérifiez que la JRE est bien 1.8.
— Si vous faites Next, vous arrivez à des configurations fines qu'il n'est pas nécessaire de modifier pour l'instant. (faites Finish)

Dans l'explorateur de gauche, on voit le nouveau projet avec un répertoire src, et la référence à la librairie système JRE (cliquer sur le triangle pour voir s'ouvrir votre projet et observer ce répertoire).

### 1.2.2 Création d'un package

Ne manquez pas cette étape, qui est très importante pour développer des projets bien organisés Dans l'explorateur de gauche, sur l'icône src dans votre projet, faire un clic droit → new → package. Dans le wizard, donnez un nom à ce package. Le nom commence usuellement par une minuscule.

### 1.2.3 Création d'une classe

Dans l'explorateur de gauche, sur l'icône du paquetage, faire un clic droit → new → class. S'ouvre un wizard "New Java Class".

- ne pas changer le source folder qui doit être correct, ni le nom du package, qui doit l'être aussi (vérifier)
- nommer la classe (MonPremierProgramme par exemple)
- ne pas toucher à la partie permettant de modifier les superclasses ni les interfaces pour le moment
- cocher la case `public static voir main` car nous mettrons un `main` dans la classe.

Cela crée dans l'Explorer la classe à l'intérieur du package. Dans l'outline (à droite), s'ouvre un explorateur sur la classe faisant apparaître ses propriétés. Au centre, se trouve le code source de la classe. Vous pouvez contrôler ce qui a été généré (notez qu'il y a un embryon d'annotation pour la génération de la documentation). Notez un commentaire TODO qui signale qu'il faut compléter le code de la méthode main. Les TODO sont générés automatiquement ou ajoutés par le programmeur. L'ensemble des tâches à faire peut être visualisé (menu Window puis Show view puis Tasks).

**Nota.** Si vous perdez des fenêtres et vous trouvez perdu, allez dans le menu "Window" et choisissez reset perspective, vous retrouverez le package explorer, la fenêtre de code centrale et la fenêtre d'outline à droite.

### 1.2.4 Quelques manipulations du code Java

Vous pouvez maintenant écrire le code de la classe. Vous remarquerez qu'il est directement indenté. Si vous copiez du code non indenté ou si vous voulez rafraîchir l'indentation : sélectionner la partie concernée, clic droit → source → correct indentation (ou crtl +I).

**Eclipse corrige quelques erreurs de compilation** Créer une variable `clavier` de type `Scanner` :

- une petite croix apparaît en début de ligne pour signaler un problème (qui est explicité en passant la souris sur la croix)
- une petite lumière signale une solution
- cliquer sur la lumière (clic gauche) et choisir la solution appropriée (ici : import java.util.Scanner)

**Complétion sémantique** Dans le corps de votre méthode main, tapez `clavier.` (c'est-à-dire "clavier" suivi de '.'). La liste des méthodes définies pour l'objet clavier est proposée ; il suffit de choisir celle qui nous intéresse (par exemple `nextInt`) en double-cliquant dessus.
Eclipse nous permettra par la suite d'ajouter automatiquement beaucoup d'autres éléments dans le code et de le modifier facilement, par exemple pour renommer des éléments.
Pour renommer un élément, sélectionnez cet élément. Toutes ses occurrences dans les fichiers le contenant sont grisées. Dans le menu contextuel, sélectionnez Refactor puis Rename. Donnez un nouveau nom. Cela renomme alors l'élément dans tous les fichiers du même projet.

### 1.2.5 Exécution d'un programme

Pour exécuter le programme (qui contient un `main`), choisir menu run → run as → java application (ou utiliser l'icône avec la flèche blanche dans un disque vert).

# 2 Introduction au « Java Development Kit » Et quelques instructions en Java

Cette partie du TP n'est pas obligatoirement réalisée. Elle vous permet de travailler dans un environnement différent (sans eclipse).
Le *Java Development Kit* offre un ensemble d'outils de développement d'applications Java. Pour utiliser ces outils, JDK ne propose pas d'interface utilisateur, on doit donc écrire des lignes de commandes dans une fenêtre « terminal ». Cette manière plus primitive de travailler vous permettra de bien comprendre les mécanismes mis en jeu lors de l'élaboration et de l'exécution des programmes et bien évidemment aussi de travailler indépendamment d'Eclipse.

## 2.1 Création des répertoires d'accueil des programmes Java

Soyez très soigneux dans toute la suite en ce qui concerne les noms de répertoires et de fichiers, principalement pour les majuscules et minuscules.
Vous devez créer les répertoires suivants pour accueillir le travail que vous ferez pendant les TP de Java (aux feuilles de l'arbre, apparaissent certains des premiers fichiers que nous placerons dans les répertoires).

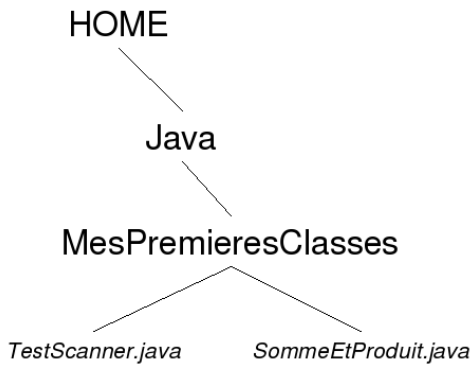


FIGURE 1 – Organisation des répertoires

Vous trouverez à l'adresse :

`http://www.lirmm.fr/users/utilisateurs-lirmm/marianne-huchard/enseignement/hmin111m`

deux fichiers à copier dans le répertoire `MesPremieresClasses` :
- `SommeEtProduit.java`,
- et `TestScanner.java`.

## 2.2 Édition des codes sources

Nous utiliserons xemacs ou kate pour l'édition des sources. Le lancement de xemacs se fait par la commande `xemacs &` ou en utilisant les menus ou icônes de votre environnement.
Éditez le code source du programme `SommeEtProduit.java` que nous vous rappelons au listing 1. Vous noterez que le nom du fichier `SommeEtProduit.java` privé de l'extension `.java` et le nom de la classe `SommeEtProduit` se correspondent. C'est obligatoire.

Listing 1 – SommeEtProduit.java

```
1 package MesPremieresClasses;
2
3 import java.util.Scanner;
4
5 public class SommeEtProduit {
```

```
6     public static void main(String[] a) throws java.io.IOException
7     {
8         Scanner sc = new Scanner(System.in);
9
10        double x, y;
11        System.out.println("entrez deux doubles ");
12        x = sc.nextDouble();
13        y = sc.nextDouble();
14        double somme = x+y;
15        double produit = x*y;
16        System.out.println("somme = "+somme+" produit = "+produit);
17    }
18
19 }
```

### 2.3 CLASSPATH

Vous devez également mettre à jour une variable d'environnement nommée CLASSPATH. Dans votre fichier nommé `.bashrc`, placez la ligne suivante (n'oubliez pas de la faire suivre d'un « saut de ligne » si c'est la dernière ligne de votre fichier, sinon elle ne sera pas prise en compte) :
`export CLASSPATH=${HOME}/Java:.`

Pour prendre en compte cette modification, tapez la commande :
`. ~/.bashrc`
ou encore la commande
`source ~/.bashrc`
La prochaine fois que vous lancerez un `shell` (fenêtre terminal) il ne sera pas nécessaire de retaper cette commande.
Java permet de désigner les classes de manière unique. Pour cela, Java concatène deux chemins : celui que contient `CLASSPATH` et celui qui correspond au nom du paquetage, par exemple, avec le programme qui précède : `${HOME}/Java` est concaténé avec `MesPremieresClasses`, ce qui donne le chemin absolu pour accéder à `SommeEtProduit`.

$$\underbrace{\$HOME/Java/}_{\text{CLASSPATH}}\ \underbrace{MesPremieresClasses/}_{\text{Paquetage}}\ \text{SommeEtProduit}$$

La résolution d'une directive d'import fonctionne sur le même modèle.

### 2.4 Compilation

Le compilateur, qui va analyser le programme source et produire le programme en bytecode s'appelle `javac`.
Placez-vous dans une fenêtre terminal, et dans le répertoire `MesPremieresClasses`, la commande de compilation sera la suivante :
`javac SommeEtProduit.java`

Si la commande `javac` n'est pas trouvée, lancez la commande `. jdk_setup`. Alternativement Vous pouvez constater qu'un nouveau fichier `SommeEtProduit.class` est apparu dans votre répertoire
`MesPremieresClasses`.

### 2.5 Exécution de programmes

Notez que l'on ne peut exécuter (interpréter) que les classes qui contiennent une méthode `main`. La méthode `main` est la méthode principale de votre programme. Votre application commence par le début de cette méthode. Elle se terminera quand l'exécution sortira du bloc `main`.

Pour la classe `SommeEtProduit` par exemple, cela se fait par la commande :
`java MesPremieresClasses.SommeEtProduit`
à exécuter depuis le répertoire Java, qui contient le répertoire `MesPremieresClasses`.

## 2.6 Scanner

Regardez d'un peu plus près le programme `TestScanner.java`. Il utilise une instance de la classe `Scanner`, qui se trouve dans le paquetage `java.util` de l'API [1] Java fournie avec le `JDK`. La bibliothèque de classes fournie par Java est très grande et variée : elle permet la lecture et l'écriture dans différents médias, la communication réseau, la réalisation d'interfaces graphiques, fournit beaucoup de structures de données classiques, etc. Il est à noter que le chemin vers l'emplacement de la bibliothèque n'est pas à ajouter dans le `CLASSPATH`, il est connu grâce à une autre variable d'environnement. Quand on souhaite utiliser une classe de la bibliothèque, il suffit donc de connaître dans quel paquetage elle se trouve, ce qui permet soit d'importer la classe (par exemple : `import java.util.Scanner;` puis : `Scanner sc= ...`), soit d'utiliser le nom absolu de la classe (`java.util.Scanner sc=...`).
La classe `Scanner` permet l'analyse de texte, notamment de texte provenant de l'entrée standard (texte tapé sur la console). Quand on crée une instance de `Scanner`, on passe en paramètre du constructeur le texte à analyser : on peut lui passer un nom de fichier par exemple, ou bien un flux de texte comme `System.in`, qui permet l'analyse de l'entrée standard. L'analyseur permet de lire les éléments d'un texte les uns après les autres, pour cela, il est nécessaire de connaître le caractère séparateur des éléments dans le texte. Par défaut, il s'agit de l'espace.
L'analyseur permet aussi de traduire les éléments (suites de caractères) lus vers le type que l'on lui fournit : on peut demander à lire le prochain entier, et on récupère alors un élément de type entier, et pas de type chaîne. Bien sûr, l'analyse échoue si l'on demande la lecture d'un entier et que l'on fournit des caractères non traduisibles en entiers.
La classe `Scanner` dispose notamment des méthodes :

- `next()` qui retourne la prochaine chaîne lue dans le texte
- `nextInt()` qui retourne le prochain entier lu dans le texte
- `nextFloat()` qui retourne le prochain flottant lu dans le texte. Le caractère séparant la partie entière de la partie décimale est la virgule.

Compilez le fichier `TestScanner.java`, exécutez-le, et modifiez-le si vous le souhaitez.
La classe Scanner utilise la localisation : elle permet notamment la saisie de réels avec la virgule (par exemple 6,3). Cependant, l'affichage au moyen de `println` utilisera le point comme séparateur. Pour obtenir de façon homogène en saisie et en affichage la virgule, il faudra utiliser les classes `Scanner` et `DecimalFormat`. De plus, n'oubliez pas que lorsque vous faites une affectation d'un réel dans le code, il faut utiliser le point, on écrira par exemple `d= 5.6;`.
Le code ci-dessous donne un exemple d'utilisation conjointe des classes `Scanner` et `DecimalFormat`.

```
Scanner sc =new Scanner(System.in);
DecimalFormat df = new DecimalFormat("0.##");
System.out.print("un double ?");
double d = sc.nextDouble();
System.out.print("double lu : "+d+"\n");
System.out.print("double lu : "+df.format(d)+"\n");
```

1. Application Programming Interface

# 3 Exercices

Nous vous indiquons que les chaînes de caractères s'écrivent entre guillemets et sont de type `String`.

A présent, vous pouvez développer quelques programmes notamment ceux vus lors des cours d'algorithmique. Vous pourrez réaliser tous vos exercices dans des fichiers séparés. Dans chaque fichier, vous définirez une classe munie d'une méthode `main`.

- Ecrire un algorithme qui demande une temperature (tempC) exprimée en degrés Celsius et affiche la température équivalente (tempF) en degrés Farenheit sachant que tempF vaut tempC * 1,8 + 32.
- Ecrire un algorithme qui demande une durée en seconde, la convertit en heures, minutes, secondes et l'affiche sous la forme ...HH ...MN ...SEC.
- Ecrire une série d'instruction effectuant l'échange de deux variables de même type.
- Ecrire un algorithme, qui à partir de trois notes rentrées au clavier, calcule la moyenne de l'étudiant Variante : les coefficients des trois notes ; calculer la moyenne pondérée.
- Connaissant l'âge et le sexe ('M', 'F') d'une personne, écrire un algorithme déterminant si la personne a droit ou non à la carte vermeil S.N.C.F. L'âge requis est 60 ans pour les femmes, 65 ans pour les hommes.
- Etant donné l'infinitif d'un verbe du premier groupe entré au clavier, afficher la conjugaison du verbe au futur de l'indicatif.
- La caissière d'un magasin désire rendre la monnaie à ses clients. Pour rendre une somme entière de S euros, elle dispose de pièces et billets de 100 euros, 10 euros, 5 euros, 2 euros et 1 euro. Elle a autant de pièces de chaque catégorie qu'elle le désire. Ecrire un algorithme calculant le nombre de pièces de chaque catégorie.