Mise en place d'un environnement de développement

Chapitre 0

Table des matières

1 Installation du compilateur Java

1. Télécharger de la dernière version du JDK (Java Development Kit) sur le site d'Oracle : http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html.

   Note. Au moment de la rédaction de cette documentation, la version proposée est : ***Java SE Development Kit 7u40***.

   → Accepter la licence.

   → Choisir la version du logiciel en fonction du système d'exploitation (OS X, OS X 64 bits, Windows 32 bits, Windows 64 bits, Linux 32 bits, Linux 64 bits, ...).

   Remarque. Sous Linux il peut être plus pratique d'utiliser le gestionnaire de paquets (apt-get, yum, zypper, ... selon la distribution).

2. Installer le logiciel en suivant les différentes étapes proposées.

   Remarque. Lors de cette étape, il peut être utile de noter quel est le dossier d'installation.
   Sous Windows, c'est généralement :

   → `C:\Program Files\Java\jdk1.7.0_40\`

   pour les systèmes 32bits (jusqu'à Seven, donc)

   → `C:\Programmes\Java\jdk1.7.0_40\`

   pour les systèmes 64 bits (et une version du JDK 64 bits).

   Le compilateur s'appelle `javac`, il se trouve dans le sous-dossier `bin` du dossier d'installation.

3. *(Étape facultative, inutile si on utilise exclusivement l'IDE Eclipse)*
   Lorsqu'on lance un programme en tapant son nom en ***mode console*** (`Démarrer→Tous les programmes→Accessoires→Invite de commande`, sous Windows, `Application→Utilitaires→Terminal`, sous Mac OS X), un système d'exploitation le cherche tout d'abord dans le répertoire dans lequel l'utilisateur se trouve lors de cet appel. Si le programme ne s'y trouve pas, le système d'exploitation examine alors les répertoires listés par la variable d'environnement nommée `PATH`.
   Pour lancer les programmes `java` *(interpréteur Java) et* `javac` *(compilateur) il faut donc que le dossier d'installation du JDK soit présent dans « le* `PATH` *».*

   → Cet ajout est effectué automatiquement à l'installation sur les systèmes d'exploitation OS X et Linux et Windows Seven.

   → Cet ajout doit être effectué manuellement sous Windows XP.

   Comment vérifier s'il est nécessaire de modifier le `PATH` ?
   Dans la ***console de commande***, entrer : `javac -version`

Le numéro de version du compilateur devrait apparaître.

Comment modifier le `PATH` ?

- Pour Windows 2000, XP et NT :
  - → Sélectionner `Démarrer→Panneau de configuration`.
  - → Double-cliquer sur `Système` et sélectionner l'onglet `Avancé`.
  - → Cliquer sur le bouton `Variables d'environnement`.
  - → Sélectionner dans les `Variables Système` la variable `PATH`. Après avoir cliqué sur le bouton `Modifier`, ajouter la ligne `C:\Program Files\Java\jdk.1.7.0_40\bin;` dans le panneau `Modifier la variable système`.
  - → Valider cette nouvelle configuration en fermant la session de travail (après avoir cliqué sur `OK` suffisamment de fois).

  Note. Le symbole `;` séparant chaque entrée est indispensable dans la variable `PATH`.

- Sous Vista et Windows 7 (si nécessaire) :
  - → Sélectionner `Démarrer→Panneau de configuration`.
  - → Double-cliquer sur `Système` et cliquer sur le lien `Paramètres systèmes avancés`, situé à gauche de la fenêtre `Panneau de configuration→Système`.
  - → Cliquer sur le bouton `Variables d'environnement`.
  - → Sélectionner dans les `Variables Système` la variable `PATH`.
  - → Cliquer sur le bouton `Modifier` puis ajouter la ligne `C:\Program Files\Java\jdk.1.7.0_40\bin;` dans le panneau `Modifier la variable système`.
  - → Valider cette nouvelle configuration en fermant la session de travail (après avoir cliqué sur `OK` suffisamment de fois).

4. *(Étape facultative, inutile si on utilise exclusivement l'IDE Eclipse)* Modification de la variable d'environnement `CLASSPATH`. Cette étape n'est nécessaire que lorsqu'on compile en ligne de commande un programme utilisant plusieurs classes. *Nous n'aurons pas à l'utiliser cette année.*
   La procédure est identique à celle décrite pour la variable `PATH` si ce n'est qu'il faut généralement créer la variable dans le panneau `Modifier la variable système`.

2 Installation d'Eclipse

2.1 Téléchargement et installation

1. Télécharger l'IDE Eclipse sur le site de la « Fondation Eclipse » : http://www.eclipse.org/downloads/.

Note. Choisir, en fonction du système d'exploitation utilisé, la version 32 ou 64 bits d'***Eclipse Standard 4.3***.

2. Ce logiciel ne s'installe pas, il suffit de décompresser l'archive téléchargée.
3. Déplacer le dossier `eclipse` à l'endroit souhaité dans l'arborescence du système de fichiers. Les droits d'administrateur sont nécessaires dans le cas d'une installation hors du dossier personnel de l'utilisateur.
4. Pour lancer le logiciel, cliquer sur l'icône nommée `Eclipse` dans le dossier `eclipse` (il est possible de faire un raccourci sur le bureau).

2.2 Éclipse refuse de se lancer car il ne trouve pas la machine virtuelle Java

1. Repérer le chemin d'installation du `JDK` dans l'arborescence du système de fichier.
2. Dans le dossier `eclipse`, se rendre dans le sous-dossier nommé `configuration`.
3. Dans ce sous-dossier ouvrir le fichier `config.ini` à l'aide d'un éditeur de texte.
4. La ligne importante dans ce fichier est `-vmargs "C:\Program Files (x86)\Java\jre6\bin"`. Elle indique au logiciel où se trouve la machine virtuelle Java.
   Remplacer le chemin indiqué par celui trouvé à l'étape 1.

2.3 Premier lancement

Au premier lancement (au moins), le logiciel demande de choisir l'emplacement du dossier de travail (`workspace`). Tous les fichiers (sources et compilés) de travail seront enregistrés dans des sous-dossiers de ce dossier de travail.

Remarque. Il est donc nécessaire d'avoir *le droit d'écrire dans ce dossier*, sous peine de ne pouvoir enregistrer son travail.
Il est donc préférable de choisir de placer le dossier `workspace` dans le dossier personnel de l'utilisateur (qu'il soit situé sur la machine ou sur le réseau).

Note. Il est possible d'examiner *la structure hiérarchique* du dossier `workspace` en utilisant ***l'explorateur de fichiers*** (***finder*** sous OS X) : Eclipse crée *un sous-dossier pour chaque projet* (cf. section 2.4), enregistre *les fichiers source* (le code) dans le sous-sous dossier `src` et place *les fichiers classes* (résultats de la compilation) dans le sous-sous-dossier `bin`.

Note. Il est, à tout moment, possible de modifier l'endroit où est stocké le dossier `workspace` :

– Sous OS X, depuis la barre des programmes, en haut de l'écran :

  → `Eclipse→Preferences→General→Startup and Shutdown→Workspaces`.

- Sous windows :
  → `Window→Preferences→General→Startup and Shutdown→Workspaces`.

2.4 Configuration

- Affichage des numéros des lignes dans l'éditeur :
  → `Preferences→General→Editors→Text Editors` et sélectionner `Show line numbers`.
- Utilisation de l'encodage UTF-8 dans l'éditeur :
  → `Preferences→General→Workspace` et sélectionner `Text File Encoding→Other→UTF-8`.
- Vérification de la machine virtuelle Java utilisée par Eclipse :
  → `Preferences→Java→Installed JREs`.

2.5 Création d'un projet

Sélectionner : `File→New→Java Project`. Un assistant permet alors de choisir le ***nom du projet***.
Il n'est pas nécessaire de modifier les autres entrées de cet assistant.

Note. Au niveau du système de fichiers, un projet est un sous-dossier du dossier `workspace`. Ce dernier possède lui-même des sous-dossiers, en particulier `src` (qui contient les fichiers sources) et `bin` (qui contient les fichiers classes).

Remarque. Il est possible d'***importer*** ou d'***exporter*** un projet : `File→Import` ou `File→Export`.

2.6 Création d'une classe

- Depuis la vue `Package Explorer` d'Eclipse, cliquer, afin de le sélectionner, sur le projet dans lequel la classe doit être créée .
- Sélectionner : `File→New→Class`. Un assistant permet alors de choisir le ***nom de la classe***.

Remarque. Pour les cours d'ISN, il faut systématiquement cocher `public static void main(String[] args)`.

2.7 Création d'une classe à partir d'un fichier

Un fichier correction sera fourni pour chaque exercice proposé. Afin d'intégrer ce fichier à un projet, le plus simple est de le copier-coller son contenu dans un fichier vide créé depuis Eclipse.

- Copier le contenu du fichier réponse.

- Depuis la vue `Package Explorer` d'Eclipse, cliquer, afin de le sélectionner, sur le projet dans lequel le fichier vide doit être créé.
- Sélectionner : `File`→`New`→`File`. Un assistant permet alors de choisir le ***nom du fichier***.

Remarque. *Le nom de ce fichier doit être exactement égal au nom de la classe qu'il contient !*

Remarque. *Le nom du fichier doit se terminer par l'extension* `.java`.

Remarque. Le fichier peut se retrouver mal placé dans l'arborescence. Le déplacer en dessous de `(default package)` s'il est localisé ailleurs dans le projet.

2.8 Compilation et exécution

Eclipse n'est pas pensé pour fonctionner avec plusieurs classes comportant toutes la méthode `main()`.
Pour lancer la compilation et l'exécution d'un programme, il faut :

- Repérer le fichier source dans la vue `Package Explorer` d'Eclipse.
- Cliquer sur ce fichier à l'aide du bouton droit de la souris afin de faire apparaître le menu contextuel.
- Sélectionner `Run As` dans ce menu puis `Java Application`.

3 Exercices du chapitre

Exercice 1. *(Observer et comprendre la structure d'un programme écrit en Java)*
Lire le programme ci-dessus et répondre aux questions.

```
/*
* Premier programme
*/
import java.util.Scanner;

public class PremierProgramme{
    static Scanner lectureClavier = new Scanner(System.in);

    public static void main(String[] args)
    {
        double a;
        System.out.println("Veuillez entrer une valeur : ");
        a = lectureClavier.nextDouble();
        System.out.println("Vous avez entré : " + a);
    }
}
```

1. Repérer les instructions définissant la ***fonction*** `main()` et celles délimitant la ***classe*** `PremierProgramme`.
2. Rechercher les instructions d'affichage.

3. Quel est le délimiteur de fin d'instruction ?
4. Quel est le rôle de l'instruction `double a;` ?
5. Décrire l'exécution du programme, en supposant que l'utilisateur a entré au clavier la valeur 10.
6. Écrire, compiler et exécuter ce programme.

Exercice 2. *(Observer et modifier un programme écrit en Java)*
À partir du code source `PerimetreCercle.java` donné ci-dessous, écrire, compiler et exécuter un programme nommé `SurfaceDisque.java` qui :

1. Demande à l'utilisateur la valeur du rayon du cercle ;
2. Calcule et affiche la valeur de la surface de ce disque.

```
/*
* Exemple de programme en Java
* Calcul du périmètre d'un cercle
* La valeur du rayon est demandée à l'utilisateur
*/

import java.util.Scanner;

public class PerimetreCercle
{
    static Scanner lectureClavier = new Scanner(System.in);

        public static void main(String[] args)
        {
             double rayon, perimetre; // Variables du problème

             System.out.println("Calcul du périmètre d'un cercle.");
             System.out.print("Entrer la valeur du rayon du cercle : ");
             rayon = lectureClavier.nextDouble();

             perimetre = 2 * Math.PI * rayon;
             System.out.println("Le perimètre du cercle de rayon " + rayon
                                +  " est égal à " + perimetre);
        }
}
```

Exercice 3. *(Écrire un programme en Java)*
À partir des exemples précédents, écrire, compiler et exécuter un programme nommé `PerimetreCarre.java` qui calcule le périmètre d'un carré dont l'utilisateur a renseigné la valeur d'un côté.

Exercice 4. *(Écrire un programme en Java)*
À partir des exemples précédents, écrire, compiler et exécuter un programme nommé`SurfaceRectangle.java` qui calcule la surface d'un rectangle dont l'utilisateur a renseigné la valeur de la longueur et de la largeur.