Université Grenoble Alpes
Licence 2 MIASHS
Inff3 – Algorithmique et programmation par objets
Jérôme David
2023-2024

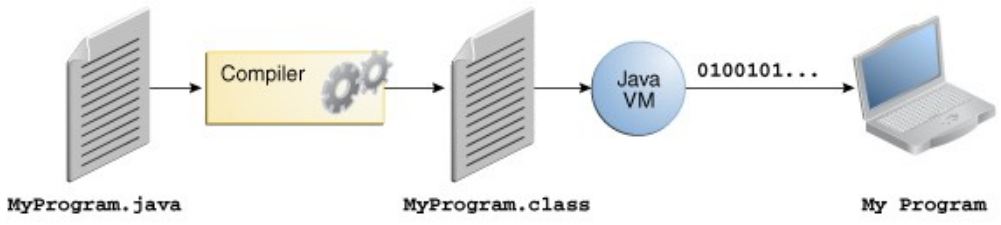
TP1 :Environnement de programmation en Java

Note : Ce TP ne doit prendre qu'une séance. Si il n'est pas terminé, vous le finirez en autonomie.

Il existe des environnements très complets de programmation en Java tels que Eclipse et NetBeans. Ces programmes (appelés IDE, integrated development environments) sont des assistants très pratiques pour le développement logiciels. Cependant, ils « cachent » plusieurs étapes et mécanismes sous-jacents au développement.
Ce TP a pour but de retourner aux racines du développement Java en réalisant manuellement les étapes de :

- création de la structure d'un projet Java
- édition de source
- compilation
- exécution
- génération de la documentation
- Packaging d'une application java

En java, le code source doit être écrit dans des fichiers texte portant l'extension `.java`. Ces codes source sont ensuite compilés dans des fichiers portant l'extension `.class` en utilisant le compilateur `javac`. Les fichiers `.class` (contrairement aux `.exe` ou `.dll` sous windows) ne contiennent pas du code machine directement interprétable par le microprocesseur mais ce que l'on appelle du bytecode (le code de la machine virtuelle java). Une application java s'exécute donc dans une machine virtuelle java appelée JVM (Java Virtual Machine). L'intérêt du bytecode java est qu'il peut s'exécuter sur n'importe quelle architecture du moment qu'il existe une machine virtuelle java de disponible. Ainsi vos `.class` pourront être exécutés indifféremment sur Windows, Linux ou MacOS.



1. Structure de base d'un projet Java

Lorsque l'on veut développer un programme, on crée dans un premier temps un dossier portant le nom du programme et qui contient les sous-dossier suivants :
- `src` : dossier où l'on mettra les fichiers source du programme
- `bin` : dossier où seront placé les fichiers compilés
- `doc` : dossier contenant la documentation du programme
- `lib` : dossier contenant les librairies utilisées dans le programme que l'on va créer

Dans votre lecteur personnel (lecteur H:) , créez un dossier `Java-L2-TP1` contenant les sous-dossiers donnés ci-dessus

2. Edition des sources

Ouvrez un éditeur de texte (Notepad++, Vi, Emac, GEdit ou autre) et copiez le code suivant :

```
import java.util.Scanner;

public class PremierProgramme {
	public static void main(String[] args) {
		double rayon, prmt, surf;
		Scanner s = new Scanner(System.in);
		System.out.println("Quel est le rayon de votre cercle ?");
		rayon = s.nextDouble();
		prmt = 2 * rayon * Math.PI;
		surf = rayon * rayon * Math.PI;
		System.out.println("Le périmètre est de " + prmt
				+ " et la surface est de " + surf);
	}
}
```

Enregistrez le fichier sous le nom `PremierProgramme.java` dans le dossier `src` créé auparavant. Vous remarquerez que le nom du fichier est le nom de la classe suffixé par l'extension `.java`. Ceci doit être toujours le cas. La règle générale est 1 classe = 1 fichier source nommé *`NomDeLaClasse`*`.java`.

3. Compilation et exécution

Avant de pouvoir exécuter un programme, il faut le compiler afin de le transformer en code machine interprétable par la machine virtuelle java. Pour cela, il faut installer une suite de logiciels appelée Java Development Kit (JDK). Il existe plusieurs « distributions » de cette suite logicielle :

- l'officielle d'Oracle : https://www.oracle.com/java/technologies/javase-downloads.html
- La version libre : https://adoptopenjdk.net/

Ces environnements de développement java contiennent entre autre :

- le compilateur `javac`
- la JVM
- la librairie standard java (java API) : ce sont une ensemble de classes prédéfinies que le programmeur peut utiliser (exemple : classes Object, String, Integer, Date, etc.)

Pour compiler le programme, ouvrez un terminal : Démarrer → Rechercher → cmd → cliquer sur OK. Ensuite, dans le terminal, exécutez successivement les commandes suivantes :

```
h:
cd Java-L2-TP1
SET PATH=%PATH%;"P:\Pedagogie\SHS\MIASHS\Java\jdk\bin"
```

Pour compiler votre programme, il faut exécuter la commande `javac`

```
javac -s src -d bin src\PremierProgramme.java
```

Si tout se passe bien (i.e. la commande précédente n'affiche rien à l'écran), allez voir dans le dossier `bin` de votre projet. Que constatez vous ?
Ouvrez le fichier `.class` généré avec le Bloc Note (ou autre éditeur de texte). Que constatez vous ?

L'exécution d'un programme Java se fait au travers de la JVM. Pour cela, il faut taper la commande suivante :

```
java -cp bin PremierProgramme
```

Normalement vous devriez avoir votre programme demandant le rayon du cercle et vous retournant sa surface.

Essayez maintenant la commande :

```
java PremierProgramme
```

Que remarquez-vous ?

Cette erreur (exception) signifie que la machine virtuelle n'a pas pu trouver la classe `PremierProgramme.java`. Il faut spécifier à la commande java où doit chercher les classes dont elle aura besoin durant l'exécution du programme. On appelle cela le classpath. Le classpath peut désigner des dossiers contenant des fichiers `.class` ou des librairies de classes portant l'extension `.jar`.
La déclaration du classpath peut être fait de deux manières :

- en utilisant la commande java avec l'option « -cp » suivie une liste de dossiers (ou fichiers .jar) séparés par « : ».
- en déclarant une variable d'environnement nommée CLASSPATH. Sous windows, la syntaxe est la suivante :

```
SET CLASSPATH=C:\undossieravecmesclasses;C:\unautredossier\unelibrairie.jar
```

4. Une première vrai classe

A partir du programme suivant, créez deux fichiers java. Le premier contiendra la définition d'une classe Cercle et le second contiendra le programme principal suivant :

```
import java.util.Scanner;

public class DeuxiemeProgramme {
	public static void main(String[] args) {
		Scanner s = new Scanner(System.in);
		System.out.println("Quel est le rayon de votre cercle ?");
		Cercle c = new Cercle();
		c.rayon=s.nextDouble();
		System.out.println("Le perimetre est de " + c.perimetre()
				+ " et la surface est de " + c.surface());
	}
}
```

Pour compiler ce second programme, exécutez la commande suivante qui permet de compiler vos deux nouveaux fichiers :

```
javac -s src -d bin src\Cercle.java src\DeuxiemeProgramme.java
```

Si tout se passe bien, exécutez votre programme :

```
java -cp bin DeuxiemeProgramme
```

Que se passe-t-il ?

Essayez la commande suivante :

```
java -cp bin Cercle
```

Que se passe-t-il ? Comment Expliquez vous ce résultat ?

En Java, toute application doit contenir une classe avec une méthode principale ayant la signature

suivante : `public static void main(String[] args)`

Une telle classe constitue le point d'entrée de l'application. Elle peut être exécutée avec la commande java. Les autres classes ne sont pas exécutables. Si on essaye de les exécuter alors le message précédent apparaît.

5. Les commentaires

Documenter correctement son code est une bonne habitude. Cela permet de se rappeler plus rapidement les codes que l'on a écrit quelques temps auparavant et cela facilite grandement la réutilisation du code par d'autres personnes.
En java, on peut commenter son code en utilisant les syntaxes suivantes :

`/* un commentaire */`

Le compilateur ignore tout le texte contenu entre `/*` et `*/`.

`// un commentaire sur une ligne`

Le compilateur ignore tout ce qu'il y a de `//` jusqu'à la fin de la ligne

Java offre un moyen encore plus puissant pour documenter son code : la JavaDoc. JavaDoc est un outil permettant de générer une documentation au format HTML à partir de commentaires présents dans le code. Cela repose sur une syntaxe bien précise. **Un commentaire javadoc commence par `/**` et fini par `*/`**. Il doit être placé juste avant une déclaration de classe, une déclaration d'attribut ou une déclaration de méthode. A l'intérieur des commentaires javadoc, on peut utiliser des tags spéciaux comme `@param` et `@return` qui permettent de décrire les paramètres et le retour des méthodes.

Voici un exemple de formatage de la documentation (à suivre dans les prochains TP) :

```
/**
 * Une classe bidon pour illustrer JavaDoc
 * @author Jérôme David
 * @version 1.0
 */
public class Bidon {

	/**
	 * un attribut qui ne sert pas :-(
	 */
	public int unAttributBidon;

	/**
	 * Description succincte.
	 * <p>
	 * Et on peut même ajouter des détails dans un autre paragraphe.
	 * </p>
	 * @param avecUnParametre ce paramètre est super important
	 * @return la somme de unAttributBidon et avecUnParametre
	 *
	 * @see Bidon#unAttributBidon
	 */
	public int uneMethodeBidon(int avecUnParametre) {
		return unAttributBidon + avecUnParametre;
	}
}
```

Copiez ce code dans un fichier .java.
Exécutez la commande : `javadoc  src/* -d doc`

Avec un navigateur web, ouvrez le fichier index.html du dossier doc de votre projet. Observez et parcourez la documentation.

Documentez le code des classes `PremierProgramme`, `Cercle` et `DeuxiemeProgramme` que vous avez créé précédemment.

6. Utilisation de librairies externes

Lors du développement d'applications Java, on peut avoir recours à des librairies de classes externes. Ces librairies sont distribuées sous forme d'archives portant l'extension .jar. Ce sont des archives ZIP contenant des classes compilées (i.e. des fichiers .class) ainsi que des métadonnées.
Dans un projet classique, on place généralement les librairies externes dans le dossier lib du projet.

Téléchargez la librairie ampoule.jar sur le site du cours. Placez-la dans le dossier lib de votre projet.
Créez la classe suivante :

```
public class TroisiemeProgramme {
	public static void main(String[] args) throws InterruptedException {
		Ampoule amp = new Ampoule();
		amp.allumer();
		Thread.sleep(2000);
		amp.eteindre();
	}
}
```

Compilez cette classe en ajoutant au compilateur la librairie externe au classpath (avec l'option -cp lib/ampoule.jar, comme pour la commande java, c.f. section 3). Exécutez le programme (n'oubliez pas d'ajouter la librairie au classpath).

7. Packaging d'une application (Optionnel)

Un programme ou librairie java est généralement distribué sous forme d'un fichier JAR. Dans le cas d'une application, le fichier jar va contenir les classes ainsi que des informations sur les librairies externe à utiliser ainsi que le nom de la classe principale à exécuter.

Avant de créer l'archive jar, il convient d'éditer le fichier avec les informations sur l'application.
Créez un fichier texte « manifest.txt » dans le dossier TP1 et copiez les ligne suivantes :

```
Main-Class: TroisiemeProgramme
Class-Path: lib/ampoule.jar
```

Dans le terminal, exécutez les commandes suivantes :

```
cd bin
jar cfm ..\PremierJar.jar ..\manifest.txt *
cd ..
```

Maintenant, vous pouvez exécuter votre application soit :
- avec la commande `java –jar PremierJar.jar`,
- en double-cliquant sur le fichier jar dans l'explorateur de fichiers.

8. Eclipse

Essayez de retrouver les fonctionnalités testées dans ce TP sous l'environnement Eclipse.
Lors de l'ouverture d'eclipse, choisissez comme workspace « h:\workspace ». Ceci est TRES IMPORTANT car vous risquez de perdre votre travail si vous ne faites pas cela.