PAQUETAGES

Jusqu'à présent nous avons vu des programmes simples. Il s'agissait d'une classe déclarant une méthode `main`. Cette méthode était déclarée `static`. Le langage Java possède un autre type de structure que la classe : le paquetage[1]. Un paquetage permet de regrouper un ensemble de classes entre elles. Nous verrons plus tard que certains éléments des classes ne peuvent être accédés que depuis l'intérieur du paquetage, d'autre de l'extérieur.

Exercice 1. Utilisation des paquetages

Question 1.1. Vous avez déjà utilisé différentes classes au cours des exercices précédents : `Math`, `Integer`, `Long`, `System`, etc. Ces classes appartiennent toutes au même paquetage. Déterminez, en consultant la documentation, quel est le nom de ce paquetage.

Question 1.2. Recherchez la documentation de la classe `InetAddress`. Sa méthode `getLocalHost` permet d'obtenir l'adresse de la machine sur laquelle votre programme s'exécute. Ecrivez un programme affichant cette adresse et **tentez** de le compiler. vous devriez obtenir un message du type :

```
shell] javac Adresse.java
   Adresse.java:3: cannot resolve symbol
   symbol  : variable InetAddress
   location: class Adresse
           System.out.println(InetAddress.getLocalHost());
                              ^
   1 error
```

Question 1.3. En consultant la documentation, déterminez à quel paquetage la classe `InetAdress` appartient-elle.

Question 1.4. Le nom complet d'une classe est constitué du nom du paquetage auquel elle appartient suivi du nom de la classe, le tout séparé par un point. Par exemple le nom complet de la classe `System` est en fait `java.lang.System`. Corrigez le programme précédent en vous inspirant de cela. Tentez de le compiler, si vous ne vous êtes pas trompé, vous devriez obtenir un message du type :

```
shell] javac Adresse.java
      Adresse.java:5: unreported exception java.net.UnknownHostException; must be
   caught or declared to be thrown
             System.out.println(InetAddress.getLocalHost());
                                                        ^
      1 error
```

Nous verrons plus tard comment gérer les exceptions en Java. En attendant vous pouvez résoudre ce problème en modifiant la signature de la méthode `main` :

```
public static void main(String args[]) throws Exception {
```

Exercice 2. Création de paquetages

Pour faire en sorte qu'une classe appartienne à un paquetage donné, on ajoute, au début du fichier `.java`, le mot clé `package` suivi du nom du paquetage dans lequel on souhaite l'inclure. D'autre part, les répertoires doivent refléter la structure de paquetage. Par exemple pour la classe `InetAddress`, le fichier `InetAdress.java` serait placé dans un répertoire nommé `java/net` et commencerait par la ligne :

1. package en anglais.

```
package java.net;
```

Question 2.5. Reprenez la classe `HelloWorld` du début. Créez les répertoires nécessaires et modifiez `HelloWorld.java` – après l'avoir recopié dans le bon répertoire – afin qu'elle appartienne au paquetage `ccsi.<votre nom>.hello`[2].

Question 2.6. Compilez cette classe et **tentez** d'exécuter le programme. Vous devriez obtenir un message de ce type :

```
shell]java HelloWorld
   java.lang.NoClassDefFoundError: HelloWorld (wrong name:
ccsi/grange/hello/HelloWorld)
   ...
```

En effet la classe ne s'appelle plus `HelloWorld` mais `ccsi.<votre nom>.hello.HelloWorld`. Ainsi il est nécessaire de préciser ce nom complet pour lancer l'application. De plus, la machine virtuelle va déduire, du nom de la classe, le répertoire dans lequel rechercher le `.class` qui correspond. Elle ira donc chercher dans le répertoire `ccsi/<votre nom/hello`.

Question 2.7. En prenant en compte les remarques précédentes et en reprenant les exercices précédent sur le *classpath*, faites en sorte de faire fonctionner cette application.

Question 2.8. Qu'en concluez-vous sur les relations *classpath*/paquetage ?

2. Bien entendu vous remplacerez `<votre nom>` par votre nom.