Université Paris Cité **Programmation Orientée Objet**
L2 **Année 2022-2023**

TP - Séance n°12bis

Packages etc ...

Le risque de conflit par l'utilisation d'un même nom pour désigner deux concepts différents est important. Afin de résoudre ce problème Java permet d'isoler et de regrouper un ensemble de classes dans un package.

Un package représente à la fois un espace de nommage pour les classes, et un répertoire de stockage pour les fichiers. Ils sont structurés à la manière d'un arbre en répertoires/sous répertoire, cette structure se retrouve dans leurs noms, qui sont séparés par des points pour indiquer qu'un package est directement interne à un autre.

L'usage est d'y regrouper des classes qui se rapportent à un même concept ou à un même rôle dans l'application.

L'appartenance d'une classe à un package est identifié par le mot-clé `package` sur la première ligne du fichier, et pour pourvoir l'utiliser vous devez l'importer dans votre programme, ou spécifier de manière complète le domaine de nom dans la déclaration des objets.

Il n'est pas obligatoire de remettre votre travail sur ce TP12bis, mais pensez à ce que vous aurez vu ici pour votre projet. En effet il est important que les explications contenues dans son README permettent une prise en main rapide du code que vous fournirez (quelle que soit la plateforme).

Exercice 1 La classe `List` est déclarée dans le package `java.util` mais aussi dans `java.awt`. Observez leur spécifications dans l'API. Écrivez une classe **sans importer aucun package** avec un `main` qui crée une liste du premier type par entrées successives de mots au clavier, jusqu'à ce que l'utilisateur rentre `"fin"`; puis qui crée une liste du second type en reprenant les entrées de la première. Affichez le résultat avec le `toString` de chaque classe.

Exercice 2 Créez un répertoire `Exo2` dans lequel vous travaillerez pour cet exercice. Dans un premier temps n'utilisez que *emacs* et faites les compilations en ligne de commande.

1. Créez un package `MonPack` qui contient une classe `A` comportant une méthode qui affiche *"un élément de type A"*.
2. Créez un sous-package `MonSousPack` contenant une classe `B` avec également une méthode d'affichage.
3. Dans `Exo2` définissez une classe de test comportant un `main` qui construit des objets de type `A` et `B` et qui appelle les méthodes d'affichage correspondantes.
4. Reprenez cet exercice avec un IDE (netbeans ...), puis observez où sont stockés vos fichiers, et quels sont les entêtes qui ont été choisis quasi automatiquement.

Exercice 3 Effacez tous les `.class` produit jusqu'à présent, puis, dans le répertoire `Exo2`, créez un nouveau répertoire `MesClass` puis lancez la compilation de votre test avec la commande `javac -d MesClass`. Observez le résultat, et que l'exécution fonctionne.

Exercice 4 Utilisation de `jar`

1. Dans le répertoire qui contient `MonPack` exécutez `jar cf MonPack.jar MonPack`
   Puis ouvrez le fichier produit avec un utilitaire de décompression (celui par défaut de l'explorateur de fichier). Observez son contenu. Vous pouvez également obtenir ces informations avec `jar tf MonPack.jar`
2. Créez un répertoire `Exo4` à la même hauteur que `Exo2` puis un nouveau fichier de test manipulant des objets `A` et `B`. Vous le compilerez et exécuterez avec :
   — `javac -classpath ../Exo2/MonPack.jar TestExo4.java`
   — `java -cp ../Exo2/MonPack.jar:. TestExo4`
3. Vérifiez à nouveau le contenu de `Monpack.jar`.
4. Faites de même, non plus en ligne de commande, mais avec votre IDE en important le jar en changeant les paramètres de votre projet.

Exercice 5 Dans cet exercice, on se propose de transformer des données pour les intégrer à nos objets. Ces données sont stockées dans des fichiers de type `.csv` :

```
1,Pankaj Kumar,20,India
2,David Dan,40,USA
3,Lisa Ray,28,Germany
```

Dans cet exemple, une ligne représente les données d'une personne, et sur chaque ligne les informations sont séparées par des virgules.

1. Définir une classe `Personne` destinée à recevoir les données traitées.
2. Téléchargez le jar d'`opencsv` disponible par exemple ici
   `http://www.java2s.com/Code/Jar/o/Downloadopencsv23jar.htm` Cette bibliothèque vous permet de manipuler des fichiers de type `csv`. Écrivez un programme qui convertit un fichier `csv` en une liste de `Personne`. Pour cela utilisez la classe `CSVReader` définie dans `opencsv`. Utilisez `jar tf opencsv-2.3.jar` pour savoir dans quel package elle se trouve.
   — Pour lire un fichier, on utilise la classe `FileReader` de Java. Elle possède un constructeur `FileReader(String chemin)` qui prend l'adresse d'un fichier en paramètre.
   — Pour lire un fichier `.csv`, on construit un `CSVReader` avec un `FileReader f` et un `char c` qui détermine le caractère de séparation des colonnes.
   — Un `CSVReader` possède une méthode String[] `readNext()` qui renvoie un tableau contenant les informations de la dernière ligne lue. Une fois arrivé à la fin du fichier, `readNext()` renvoie `null`. Notez que cette méthode peut lever une `IOException`.

   L'intérêt de cet exercice consiste à utiliser correctement une bibliothèque importée. Essayer d'écrire le programme demandé sans utiliser votre IDE ni chercher d'exemple en ligne.
3. Refaire la même chose en utilisant votre IDE.

Exercice 6 Le tableau ci dessous résume la visibilité des attributs selon qu'ils soient privés, publics ou protected. L'absence de ces mots clés a également une signification qui est plus stricte que `protected`. Définissez un ensemble de classe et de packages qui illustre le `non` qui distingue `protected` de *sans modificateurs*.

	Private	No Modifier	Protected	Public
Same class	Yes	Yes	Yes	Yes
Same package subclass	No	Yes	Yes	Yes
Same package non-subclass	No	Yes	Yes	Yes
Different package subclass	No	No NO !	Yes	Yes
Different package non-subclass	No	No	No	Yes