I. Introduction

- Un grand nombre de classes, fournies par Java, implémentent des données et traitements génériques utilisables par un grand nombre d'applications.
- Ces classes forment la librairie standard ou API[1] standard (Application Programmer Interface) du langage Java
- Le langage Java est indissociable de sa librairie standard. Étudier le langage Java consiste donc aussi à étudier les points les plus importants de cette librairie.
- Ces classes sont grouper dans des packages selon leur caractéristiques voisines qui couvrent un même domaine.

II. Les package

Les packages (paquetage) sont des regroupements logiques de classes et d'interfaces. Ils permettent une protection d'accès de même que la définition d'espace de nom.

II.1 Caractéristiques

- Chaque package porte un nom. Ce nom est soit un simple identificateur ou une suite d'identificateurs séparés par des points.

- Le package en java peut être classé sous deux formes, un package intégré(API standard) et un package défini par l'utilisateur.

[1] Une documentation en ligne pour l'API java est disponible à l'URL : http://docs.oracle.com/javase/7/docs/api/

a) Exemple de packages (API standard)

packages	JDK 1.0	JDK 1.1	JDK 1.2	JDK 1.3	JDK 1.4	JDK 1.5	JDK 6.0	JDK 7.0	JDK 8.0
java.applet Développement des applets	x	x	x	x	x	x	x	x	x
java.awt Toolkit pour interfaces graphiques	x	x	x	x	x	x	x	x	x
java.awt.color Gérer et utiliser les couleurs			x	x	x	x	x	x	x
java.awt.geom dessiner des formes géométriques			x	x	x	x	x	x	x
java.awt.image Afficher des images		x	x	x	x	x	x	x	x
java.awt.image.renderable Modifier le rendu des images			x	x	x	x	x	x	x
java.awt.print Réaliser des impressions			x	x	x	x	x	x	x
java.io Gérer les flux	x	x	x	x	x	x	x	x	x
java.lang Classes de base du langage	x	x	x	x	x	x	x	x	x
java.security Gérer les signatures et les certifications		x	x	x	x	x	x	x	x
java.sql API JDBC pour l'accès aux bases de données		x	x	x	x	x	x	x	x
java.util Utilitaires divers	x	x	x	x	x	x	x	x	x
java.util.zip Gérer les fichiers zip		x	x	x	x	x	x	x	x
javax.net.ssl Utiliser une connexion réseau sécurisée avec SSL					x	x	x	x	x
javax.security.auth API JAAS pour l'authentification et l'autorisation					x	x	x	x	x
javax.swing Swing pour développer des interfaces graphiques			x	x	x	x	x	x	x
javax.swing.colorchooser Composant pour sélectionner une couleur			x	x	x	x	x	x	x
javax.swing.filechooser Composant pour sélectionner un fichier			x	x	x	x	x	x	x
javax.swing.plaf Gérer l'aspect des composants Swing			x	x	x	x	x	x	x

b) Création d'un package

Pour réaliser un package :

1. on écrit un nombre quelconque de classes dans plusieurs fichiers d'un même répertoire
2. au début de chaque fichier on met l'instruction ci-dessous où nom-du-package doit être composé des répertoires séparés par un caractère point :

package nom-du-package;

autrement dit, pour créer un package, il suffit de mettre une classe dedans et la compiler. Pour mettre une classe dedans, il suffit d'ajouter l'instruction package au début du fichier **.java.**

Exemple :

```
1 //save as Simple.java
2 package mypack;
3 public class Simple{
4  public static void main(String args[]){
5     System.out.println("Welcome to package");
6    }
7 }
```

Remarques :

- Le mot clé package doit être la première instruction dans un fichier source et il ne doit être présent qu'une seule fois dans le fichier source (une classe ne peut pas appartenir à plusieurs packages).
- La hiérarchie d'un package se retrouve dans l'arborescence du disque dur puisque chaque package est dans un répertoire nommé du nom du package. Java utilise le répertoire du système de fichiers pour stocker le package.
- Si l'instruction package était absente du fichier, alors c'est le package par défaut qui est pris en considération. Ainsi toutes les classes du dit fichier vont appartenir au package par défaut.
- D'une façon générale, l'instruction package associe toutes les classes qui sont définies dans un fichier source à un même package.

II.3 L'utilisation d'un package

Pour utiliser le package créé, on l'importe dans le fichier selon les instructions suivantes :

Instruction	Rôle
import nomPackage.*;	toutes les classes du package sont importées
import nomPackage.NomClasse;	appel à une seule classe : l'avantage de cette notation est de réduire le temps de compilation
nom complet de la classe	**en indiquant le nom complet de la classe (Fully Qualified Name) selon la nécessité d'utilisation**

Exemples :

1. **Importer un package dans un package**

import nompackage.*

```java
//save by A.java
package pack;
public class A{
  public void msg(){System.out.println("Hello");}
}
```

```java
//save by B.java
package mypack;
import pack.*;

class B{
  public static void main(String args[]){
    A obj = new A();
    obj.msg();
  }
}
```

2. Importer la classe d'un package

import **nompackage.Nomclasse**

```java
//save by A.java

package pack;
public class A{
  public void msg(){System.out.println("Hello");}
}
```

```java
//save by B.java
package mypack;
import pack.A;

class B{
  public static void main(String args[]){
   A obj = new A();
   obj.msg();
  }
}
```

3. Indiquer le nom complet de la classe (Fully Qualified Name)

nom complet de la classe

```java
//save by ClassA.java
package pack;
public class ClassA{
   public void msg(){
           System.out.println("Hello Brother");
          }
}
```

```java
//save by ClassB.java
package mypack;
class ClassB{
   public static void main(String args[]){
    pack.ClassA obj = new pack.ClassA();
    obj.msg();
   }
}
```

Remarques :

- L'utilisation des packages permet d'améliorer la qualité de la programmation. En effet, en regroupant les classes en package, on évite de se donner des accès multiples qui rendent le code illisible (code spaghetti[2]).
- L'utilisation des packages permet de voir quelles classes sont reliées, Celles-ci sont plus faciles à trouver parceque regroupées logiquement,
- les conflits de noms sont presque impossibles
- et finalement, les classes d'un même package peuvent avoir des accès illimités (comme public) alors que d'autres offre un accès limité (comme private).

[2] Un code spaghetti est un code peu clair et qui fait un usage excessif de sauts inconditionnels (voir goto), d'exceptions en tous sens, de gestion des événements complexes et de threads divers.