Université Paris Diderot **Programmation Orientée Objet**
L2, EIDD et M1 linguistiques **Année 2015-2016**

TD n° 9

I/O et Exceptions

1 Types Génériques

Exercice 1 : Type safety.

1. Considérez le programme suivant.

```
1 List list = new ArrayList();
2         list.add(1000);
3         Integer s=(Integer) list.get(0);
```

   Compile-t-il sans erreur ? S'éxecute-t-il sans erreur ?

2. Considérez maintenant la variante ci-dessous :

```
1   List list = new ArrayList();
2   list.add(1000);
3   String s=(String) list.get(0);
```

   Ce code compile-t-il ? S'éxecute t'il ?

3. Ecrivez une version du programme de la première question utilisant List<Integer>. En quoi peut-on dire que cette manière de programmer est plus sûre que la précédente ?

4. La classe List<int> existe-t-elle ? Pourquoi ?

Exercice 2 : Classe Paramétrée.
On a vu ci-dessus comment utiliser la classe paramétrée `List<T>`. On peut définir ses propres classes paramétrées.

1. Définissez une classe paramétrée `Box<T>`, qui a en attribut un objet de type `T`, et qui a deux méthodes de signature `public T get()` et `public void set(T)`.

2. Ecrivez une classe Test, dans laquelle vous créez un objet appartenant à la classe `Box<Integer>`, et un objet appartenant à la classe `Box<List<Integer>>`

Exercice 3 : Arbres Paramétrés.
On veut maintenant modéliser la structure d'arbre. Un arbre est donné par une racine, un fils droit qui est un arbre, et un fils gauche qui est également un arbre.

1. Ecrire une classe paramétrée `public class Arbre<T>`, qui modélise la classe arbre.

2. Ajoutez à cette classe une méthode `public void insertion(T)`, qui permet d'ajouter un élément, et une méthode `public boolean research(T)` qui décide si un élément de type `T` est dans l'arbre ou non.

3. Construire l'arbre de hauteur 2, de racine `1`, et de feuilles `3` et `4`.

4. Ajouter une méthode `public void affiche()` qui permet d'afficher l'arbre.

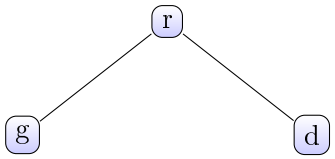
Exercice 4 Bounded type parameter.
Il y a des situations où on veut pouvoir paramétrer une classe (ou une méthode) non pas par n'importe quel type `T`, mais seuleument par les classes `T` qui ont la structure dont on a besoin. Cet exercice présente un exemple d'une telle situation : on veut considérer une sous-classe de la classe des arbres, pour laquelles on peut calculer un entier qui donne une mesure de la complexité (ou simplement du poids total) de l'arbre.

1. Ecrire une interface `Taille`, qui spécifie une méthode `public int taille()`+
2. Ecrire une classe `TreeMeasurable`, qui correspond aux arbres construit sur des types qui implémentent l'interface `Taille`. Cette classe doit elle-même implémenter l'interface `Taille`. La signature est :

   ```
   public TreeMeasurable <T extends Taille> extends Tree<T> implements Taille
   ```

3. Ecrire une classe A qui implémente `Taille`, et une instance de la classe TreeMeasurable<A>.
4. On peut également écrire des méthodes paramétriques. Ecrire une classe `Compteur`, qui a une méthode qui prend en argument un objet de la classe TreeMeasurable<T>, et incrémente son compteur de la taille de l'objet en question.

Exercice 5 Arbres Binaires de Recherche.
On veut maintenant implémenter la structure d'arbre binaire de recherche. On rapelle qu'il s'agit d'un arbre tels que les éléments de l'arbre `g` sont tous plus petits que `r`, et que ceux de l'arbre `d` sont tous plus grand que `r` :



1. On ne peut définir une structure d'ABR que sur des types qui ont une relation d'ordre. Spécifiez une interface (paramétrique) ComparableTo<T>, qui correspnd à comparer deux éléments d'un type `T`.
2. On va maintenant définir une classe de signature :

   ```
   public class ABR< T extends ComparableTo<T> > extends Arbre<T>.
   ```

   Quel est ici le paramètre de type ? Quel est la contrainte qu'on lui impose ? De quelle classe cette classe hérite-t-elle ?
3. Redéfinir les méthodes `insertion` et `recherche` de la classe ABR (on ne demande pas de conserver l'équilibre de la structure...).

Exercice 6 Shéma Observateur-Observé.
On veut modéliser une banque comme une structure pouvant être observé par diversobservateurs : les clients, qui peuvent observer les comptes dont ils sont titulaires, et une autorité de régulation des banques(AMF), qui peut observer l'argent total dont la banque dispose. De quelles classes avez vous besoin pour modéliser cette situation ? Quelle est la classe observée ? Quelle sont les classes observateurs ? (On peut remarquez que l'on a deux types différents d'observateurs : les clients et l'AMF). De quelles méthodes avez vous besoin ? Donnez la modélisation uml.