



# TP : Introduction aux génériques

CREATION : 2010/04/22

MISE A JOUR : 2011/03/18

Ce TP est une séance légère de manipulation des génériques. Pour une utilisation un peu plus avancée, je vous conseille le tutoriel :

<http://www.oracle.com/technetwork/java/javase/generics-tutorial-159168.pdf>

Depuis la plateforme 1.5, le java s'est doté des *generics*. L'intuition des développeurs C++ sera fortement perturbée car les choix d'implémentation des classes paramétrées formelles sont fondamentalement différents. En Java, une classe paramétrée (souvent un seul caractère) est compilée et partagée par toutes ses invocations (les paramètres formels ne sont que des paramètres). Ce choix a tout de même un avantage, les vieux codes (ou *Legacy*, ie sans *generics*) sont directement utilisables avec des compilateurs récents.

## Legacy code

- Créer une classe Tableau qui peut contenir des objets dans un tableau statique. La taille initiale sera de 1 ou de 2. On mémoriserà le nombre d'éléments réellement dans le tableau par un attribut (on suppose qu'il n'y a pas de trous).
- Écrire le constructeur sans argument qui initialise les différents attributs.
- Écrire le getter pour l'attribut qui donne le nombre d'éléments effectivement dans le tableau.
- Écrire une méthode privée agrandir() qui double la taille du tableau statique. La méthode lancera l'exception MyArrayExtensionException (Exception à créer) si cela est impossible. Grâce à System.arraycopy() les éléments déjà présents dans le tableau seront préservés.
- Écrire une méthode addElement() qui permet d'ajouter l'Object en paramètre au tableau si cela est possible.
- Écrire une méthode getElementAt() qui renvoie l'Object dont l'index est précisé en paramètre. Si l'index ne correspond pas, on lancera l'exception MyArrayOutOfBoundsException (Exception à créer également).
- Redéfinir la méthode toString() pour qu'elle affiche le nom de la classe, l'adresse mémoire de l'objet, la taille réelle et la taille maximale du tableau d'objet.
- Écrire une application qui peuple ce tableau de quelques entiers puis qui en fait la somme, élément par élément. Il est obligatoire de faire un downcasting d'Object en Integer pour pouvoir connaître la valeur de l'entier contenu dans le tableau.

## Conteneur générique

- Adapter la classe Tableau pour en faire une classe paramétrée sur le type O.



Bon d'accord, ce n'est pas si facile ! Java nous empêche d'écrire une instruction comme `O[] o = new O[10]` lorsque O est une classe paramétrée. Une solution possible est de convertir un tableau d'Objects : `O[] o = (O[])new Object[10]` ;

On ajoutera une annotation `@SuppressWarnings("unchecked")` pour enlever l'avertissement à la compilation.

Les génériques et les tableaux ne vont pas bien ensemble : c'est une histoire de *reification* et d'*erasure* pour ceux que cela intéresse. Cela pourrait peut-être changer pour la version 7 de java.

Il n'est pas possible non plus de créer un objet d'une classe paramétrée `O o = new O()` ; Une des solutions les plus élégantes pour faire cela est de se servir d'un élément de référence de la classe O et de le cloner ! Cela n'enlèvera toutefois pas l'annotation.

Écrire le même genre de tests que dans la partie précédente pour vérifier le tableau paramétré. On notera bien que le *downcasting* des éléments renvoyés par `getElementAt()` n'est **plus nécessaire**.

## Conteneur générique trié

Il n'y a aucune condition sur le type de paramètre des classes paramétrées. Imaginons que l'on veuille faire un tableau trié. Pour cela, il faut s'assurer que la classe donnée en paramètre respecte ce que l'on appelle en mathématiques un ordre. On peut trier des objets qui implémentent l'interface Comparable.

- Écrire un tableau générique où les éléments sont triés. On s'assurera que la classe est paramétrée par un élément qui implémente l'interface Comparable
- Chaque élément ajouté par la méthode addElement() sera positionné à sa bonne place.
- On redéfinira la méthode toString() du tableau pour afficher les éléments du tableau.
- On testera avec une classe Point avec deux attributs de type double : x et y. L'ordre sera simplement la distance à l'origine. Le point le plus grand sera celui qui est le plus loin de l'origine.
- Tester avec quelques éléments.

Remarque : pour instancier un tableau de O, il faut utiliser Object ou un surtype (dans sa version non générique) si un surtype (même générique) est précisé (si T2 dérive de T1 alors T1 est un surtype de T2).

## Conclusion

Vous l'aurez compris : l'utilisation des génériques en java n'est pas si simple : il est bien plus facile de lire un code que de l'écrire. Implémenter un tableau n'est pas évident, les concepteurs conseillent plutôt d'utiliser les listes ;-)

N'hésitez pas à explorer les différentes collections offertes par la plateforme pour utiliser celle qui correspond le plus à vos besoins.