**PROGRAMMATION AVANCEE TP N°1**

NOM : BUMA

POST NOM : MASENGO

PRENOM : ADOLPH

MATRICULE :2018020009

**RESUME SUR LA GENERECITE**

En programmation, la **généricité** d'une fonction repose sur son indépendance vis-à-vis du type, et éventuellement du nombre, de ses arguments. C'est un concept important pour un langage de haut niveau car il permet d'augmenter le niveau d'[abstraction](mhtml:file://D:\TRADING\Généricité%20_%20définition%20et%20explications.mhtml!https://www.techno-science.net/definition/168.html) ( En philosophie, l'abstraction désigne à la fois une opération qui consiste a isoler par la...) du langage. Plusieurs mécanismes visant à permettre l'écriture de fonctions génériques ont donc été mis en œuvre par les différents langages de [programmation](mhtml:file://D:\TRADING\Généricité%20_%20définition%20et%20explications.mhtml!https://www.techno-science.net/definition/5286.html) (La programmation dans le domaine informatique est l'ensemble des activités qui permettent...).

Un avantage de la **programmation générique** consiste à abstraire un [ensemble](mhtml:file://D:\TRADING\Généricité%20_%20définition%20et%20explications.mhtml!https://www.techno-science.net/glossaire-definition/Ensemble.html) (En théorie des ensembles, un ensemble désigne intuitivement une collection...) de concepts cohérents pour construire des algorithmes au-dessus indépendamment de leur [implémentation](mhtml:file://D:\TRADING\Généricité%20_%20définition%20et%20explications.mhtml!https://www.techno-science.net/definition/5319.html) (Le mot implantation peut avoir plusieurs significations :). Ainsi la fonction de tri de la STL a [besoin](mhtml:file://D:\TRADING\Généricité%20_%20définition%20et%20explications.mhtml!https://www.techno-science.net/definition/686.html) (Les besoins se situent au niveau de l'interaction entre l'individu et l'environnement. Il est...) en entrée d'itérateurs aléatoires sur une collection d'objets et d'une fonction de comparaison sur ces objets ; cela rend ainsi cette fonction utilisable dans de très nombreux contextes.

Ce mécanisme permet une programmation générique purement statique puisque l'ensemble des résolutions de types est fait au moment de la compilation.

**Par exemple**, si l'on veut créer une liste générique, une déclaration (ici en utilisant les **templates** de C++) serait : List, où T représente un type. À l'instantiation, on pourrait créer une List ou une List. La liste est ensuite traitée comme si le type correspondant avait été spécifié.

* La généricité est un concept très utile pour développer des objets travaillant avec plusieurs types de données.
* Vous passerez donc moins de temps à développer des classes traitant de façon identique des données différentes.
* La généricité permet de réutiliser sans risque le polymorphisme avec les collections.
* Cela confère plus de robustesse à votre code.
* Vous pouvez coupler les collections avec la généricité !
* Le wildcard (?) permet d'indiquer que n'importe quel type peut être traité et donc accepté !
* Dès que le wildcard (?) est utilisé, cela revient à rendre ladite collection en lecture seule !
* Vous pouvez élargir le champ d'acceptation d'une collection générique grâce au mot-clé extends.
* L'instruction ? extends MaClasse autorise toutes les collections de classes ayant pour supertype MaClasse.
* L'instruction ? super MaClasse autorise toutes les collections de classes ayant pour type MaClasse et tous ses supertypes !
* Pour ce genre de cas, les méthodes génériques sont particulièrement adaptées et permettent d'utiliser le polymorphisme dans toute sa splendeur

Conclusion

La généricité apporte souplesse et robustesse dans le code, la généricité apporte une complexité importante, pour utiliser les classes génériques existantes ce n’est pas compliqué mais créer ses propres classes génériques est plus difficile.

**Exemple**

**public** **static** void main(String[] args) {

System.out.println("Liste de voiture");

System.out.println("------------------------------");

List<voiture> listeString= **new** ArrayList<voiture>();

listeString.add("carina");

listeString.add("prado GXR");

listeString.add("urus");

listeString.add("escape");

**for**(voiture str : listeString)

System.out.println(str);

System.out.println("\nListe de moto");

System.out.println("------------------------------");

List<moto> listeFloat = **new** ArrayList<moto>();

listeFloat.add("bugatti");

listeFloat.add("tvs");

listeFloat.add("kaswaki");

listeFloat.add("boxer");

**for**(moto f : listeFloat)

System.out.println(f);

}