TP11

Exercice – paires génériques

1. Créez une classe générique `Paire<X, Y>` qui permet de manipuler des paires associant n'importe quels types de données et écrivez les méthodes qui définissent la forme canonique de la classe (*i.e.*, constructeurs, accesseurs, `toString`, `equals`, `clone`).
2. Écrivez une méthode générique statique qui prend en paramètre une paire d'objets de même type et retourne le plus grand des deux.
3. Modifiez votre classe pour permettre la comparaison de deux paires : $(x_1, y_1) > (x_2, y_2)$ ssi $(x_1 > x_2) \vee (x_1 = x_2 \wedge y_1 > y_2)$
4. Écrivez une méthode générique statique qui prend en paramètre une paire d'objets dont les types étendent la classe `Number` et retourne la valeur `double` correspondant à la somme des deux nombres.
5. Écrivez trois comparateurs de paires selon différents critères :
   (a) La première valeur : $(x_1, y_1) > (x_2, y_2)$ ssi $x_1 > x_2$
   (b) La seconde valeur : $(x_1, y_1) > (x_2, y_2)$ ssi $y_1 > y_2$
   (c) La somme des deux valeurs : $(x_1, y_1) > (x_2, y_2)$ ssi $x_1 + y_1 > x_2 + y_2$
6. Écrivez un programme de test.

Exercice – piles génériques

Dans un TP précédent, nous avons écrit des implémentations d'une pile de formes géométriques.

1. Modifiez votre code afin de rendre génériques les piles, c'est-à-dire capables de contenir des éléments d'un type particulier (pensez à utiliser le menu *Refactor* d'Eclipse).
2. Ajoutez une implémentation d'une pile, utilisant une liste pour stocker les éléments. Quelle implémentation de l'interface `List` utiliser ? Pourquoi ?
3. Modifiez et complétez le programme de test.