**Exercice Expression Lambda**

**Exercice 1**

Soit Interface fonctionnelle Consumer<T> contenant la méthode public void accept(T t). Implémenter la méthode accpet :

1. Qui affiche un entier donné en paramètre.
2. Qui affiche les éléments d’une liste donnée en paramètre.
3. Qui multiplie (\*2) et affiche les nombres d’une liste donnée en paramètre.

**Exercice 2**

Soit une référence **apples** de type List<Apple>, et à l’aide de l’interface Consumer

1. Coder une fonction permettant d’afficher, pour chaque Apple, le message suivant : "An apple of " + apple.getWeight() + "g" ;
2. Coder une fonction permettant d’afficher un message personnalisé, pour chaque Apple, indiquant s’il s’agit d’un heavy Apple (Apple>= 150) ou bien d’un light Apple ;

**Exercice 3**

Soit l’interface Function définit comme suite :

**public interface Function<T,R> {**

**public <R> apply(T parameter);**

**}**

En utilisant l’interface Function,

1. Coder une fonction ajouter4 permettant d’additionner 4 à chaque nombre stocké dans une liste donnée en paramètre ;
2. Coder la fonction map permettant de renvoyer une liste contant les tailles des String stockés dans listeString ;

**List<Integer> map(List<String> listeString, Function <String,Integer> function)**

1. Soit énumération Color { RED, GREEN}, coder une fonction permettant d’associer à chaque valeur déclarée dans l’énumération Color (RED, GREEN) un String (« red », « green ») ;

**Exercice 4**

Coder une fonction **afficher(int [] t, IntPredicate f)** permettant d’afficher les nombres positifs d’un tableau;

Utiliser la même fonction pour afficher les nombres négatifs, puis les nombres pairs.

**Synthèse :**

Soit la classe Point :

**public** **class** Point {

**private** **int** x, y;

**public** Point(**int** x, **int** y) {

**this**.x = x;

**this**.y = y;

}

**public** **int** getX() {

**return** x;

}

**public** **int** getY() {

**return** y;

}

@Override

**public** String toString() {

**return** "Point [x=" + x + ", y=" + y + "]";

}

**public** **void** affiche() {

System.***out***.print(" [ " + x + ", " + y + "] ");

}

}

Soit la classe Point. Coder une fonction **traiteListe (List<Point> liste, Predicate<Point> predicate, Comparator<Point> comp, Consumer<Point> aff) :**

* Liste : liste des points.
* Predicate : pour sélectionner des éléments réalisant une condition ;
* Comparator : pour trier les points (suivant un critère variable)
* Comsumer : pour afficher les éléments ainsi triés.

On utilisera cette fonction pour

* pour sélectionner les points d’abscisse positive, les trier sur les valeurs de leurs abscisses et les afficher suivant cette forme : [ 2, 5] [ 2, 3] [ 6, -3]
* pour sélectionner tous les éléments, les trier suivant la somme de leurs coordonnées et les afficher ainsi : (abs = -3, ord = 4) (abs = 6, ord = -3) (abs = 2, ord = 3)