Programmation Objet en Java - Feuille de TD 7

Exercice 1. Construction d'un index

On part d'une suite d'entrées formées d'un mot et d'un numéro de page, comme dans l'exemple ci-dessous :

```
22, "Java"
23, "Map"
25, "Java"
25, "Java"
29, "Java"
25, "Iterator"
```

et on veut obtenir un "index" comprenant la liste des mots avec les numéros de pages où ils apparaissent.

```
Iterator [25]
Java [22, 25, 29]
Map [23]
```

Chaque mot apparaît une fois, dans l'ordre alphabétique, et la liste des numéros correspondants et donnée en ordre croissant, sans répétition.

a) Créez une classe `Index` permettant de stocker l'index avec un champ `map` qui est une `Map` et constructeur qui créera au début un index vide.

b) Écrire une méthode `insert` qui prend en argument un numéro de page et un mot, et modifie l'index pour tenir compte de cette nouvelle entrée.

c) Écrire une méthode permettant d'afficher l'index comme ci-dessus.

La classe `Index` sera utilisée par la classe `IndexTest` suivante :

```java
public class IndexTest {
   public static Index makeIndex(){
     Index index = new Index();
     index.insert(22,"Java");
     index.insert(23,"Map");
     index.insert(25,"Java");
     ...
     return index;
   }

   public static void main(String[] args){
     Index index = makeIndex();
     System.out.println(index);
   }
}
```

Exercice 2. Le tirage du Loto avec des `Point`

On réutilisera le record `Point` des TDs précédents :

```java
public record Point(int x, int y){}
```

On désire effectuer un tirage aléatoire de 6 points (`Point`) dont les coordonnées sont comprises entre 0 et 9 et afficher le tirage avec les points en ordre croissant (odre croissant suivant `x` puis, en cas d'égalité, suivant `y`).

```
> java Loto
[(2,3), (2,7), (3,8), (5,0), (5,8), (8,4)]
```

On écrit pour cela une classe `Loto` contenant une méthode `main`.

```
public class Loto {
  public static void main(String[] args) {
    List<Point> points  = new ArrayList<Point>();
    ...
```

a) Compléter la classe `Loto` en remplissant `points` avec tous les points dont les coordonnées sont comprises entre 0 et 9.

b) Mélanger les points en utilisant une méthode bien choisie de la classe `Collections` (voir extrait de la javadoc page suivante).

c) Extraire une vue des 6 premières cases de la liste `points`.

d) Trier cette sous-liste et l'afficher.

e) Êtes-vous sûr que la méthode de tri fonctionne? Que faut-il rajouter pour cela ? Écrire le code correspondant.

f) Est-ce que le liste `points` a changé ?

static <T> boolean	addAll(Collection<? super T> c, T... elements)	Adds all of the specified elements to the specified collection.
static <T> int	binarySearch(List<? extends Comparable<? super T>> list, T key)	Searches the specified list for the specified object using the binary search algorithm.
static int	frequency(Collection<?> c, Object o)	Returns the number of elements in the specified collection equal to the specified object.
static <T> ArrayList<T>	list(Enumeration<T> e)	Returns an array list containing the elements returned by the specified enumeration in the order they are returned by the enumeration.
static <T extends Object & Comparable<? super T>> T	max(Collection<? extends T> coll)	Returns the maximum element of the given collection, according to the natural ordering of its elements.
static <T extends Object & Comparable<? super T>> T	min(Collection<? extends T> coll)	Returns the minimum element of the given collection, according to the natural ordering of its elements.
static <T> boolean	replaceAll(List<T> list, T oldVal, T newVal)	Replaces all occurrences of one specified value in a list with another.
static void	reverse(List<?> list)	Reverses the order of the elements in the specified list.
static void	rotate(List<?> list, int distance)	Rotates the elements in the specified list by the specified distance.
static void	shuffle(List<?> list)	Randomly permutes the specified list using a default source of randomness.
static <T extends Comparable<? super T>> void	sort(List<T> list)	Sorts the specified list into ascending order, according to the natural ordering of its elements.
static void	swap(List<?> list, int i, int j)	Swaps the elements at the specified positions in the specified list.