Liste générique, auto-boxing et génerics

Exercice 1 - Les listes chaînées génériques

Le but de cet exercice est de changer l'implantation de `fr.umlv.datas.Link` et de `fr.umlv.datas.LinkedLink` pour obtenir des listes génériques.
Le plus simple pour obtenir des structures de données génériques en Java consiste à utiliser `java.lang.Object` comme type permettant de stocker les données.

1 Changer les méthodes `add()` et `get()` pour que celle-ci prenne en paramètre ou en type de retour un `Object`.
Changer l'implantation en conséquence.

2 Que doit-on changer dans la méthode `main` de la classe `fr.umlv.datas.Main` ?
Expliquer pourquoi un `cast` est nécessaire lorsque l'on utilise la méthode `get()`.

Exercice 2 - Auto-boxing et auto-unboxing

Nous allons maintenant voir comment marche le mécanisme qui transforme tout type primitf en objet et vice-versa.

1 Que fait le code suivant :

```
Integer value=777;
int i=value;
```

Dans ce code, quelle instruction peut effectuer une allocation ?
PS: utiliser `javap` pour vous aider.

2 Qu'affiche le code ci-desssous ?

```
public static boolean isEquals(Integer i,Integer j) {
  return i==j;
}
public static void main(String[] args) {
  System.out.println(isEquals(3,3));
  System.out.println(isEquals(128,128));
}
```

Expliquer ?

Nous allons maintenant utiliser le mécanisme de boxing avec la liste chaînée générique.

1 Ecrire une classe `fr.umlv.datas.PrimitiveTest` ayant une méthode `main` qui ajoute des entiers dans la liste chaînée.

2 Le code suivant ne marche pas que doit-on faire ?

```
LinkedList list=new LinkedList();
list.add(3);
int i=list.get(0);
```

Exercice 3 - Utilisation de la classe ArrayList

Le but de cet exercice est la manipulation de la classe `java.util.ArrayList` et la notions de generics.

1 Extraire l'interface `fr.umlv.datas.IList` de la classe

`fr.umlv.datas.LinkedList`.
Faire en sorte que `fr.umlv.datas.LinkedList` implante `fr.umlv.datas.IList`.

2 Changer la classe `fr.umlv.datas.Main` pour introduire le type `fr.umlv.datas.IList` là où c'est possible.

3 Regarder la javadoc de `java.util.ArrayList` et écrire la classe `fr.umlv.datas.DynamicList` qui implante l'interface `fr.umlv.datas.IList` en utilisant un `java.util.ArrayList` pour stocker les éléments.

4 Quel est l'intérêt d'avoir une interface, ici, `fr.umlv.datas.IList` et deux implantations `fr.umlv.datas..DynamicList` et `fr.umlv.datas.LinkedList` ?

5 Changer la classe `fr.umlv.datas.Main` pour que celle-ci prenne en paramétre une chaîne de caractères. Si cette chaîne de caractères est "linked" alors le main doit utiliser la classe `fr.umlv.datas.LinkedList` pour les tests sinon si la chaîne de caractères est "dynamique" le main doit utiliser la classe `fr.umlv.datas..DynamicList`.

6 Changer l'interface `fr.umlv.datas.IList` et les classes `java.util.DynamicList` et `fr.umlv.datas.LinkedList` pour que celle-ci soit paramètrées.
Réécrire la classe `fr.umlv.datas.Main`.