Paquetage, Structure de données, Relation d'implantation

Exercice 1 - Les listes chaînées

Le but de cet exercice est d'écrire une implantation de listes chaînées.

Pour la suite de l'exercice, l'ensemble des classes créées devront être dans le paquetage `fr.umlv.data`. Les fichiers sources (.java) doivent être dans le répertoire `src` et les fichiers destinations (.class) doivent être dans le répertoire `classes`.
Si ce n'est pas le cas, configurer votre Eclipse dans Window > Preferences > Java > Build Path et dans project > Properties > Java > Build Path.

Nous allons dans un premier temps créer une liste chaînée d'entiers.

1. Créer une classe `Link` dans le paquetage `fr.umlv.data` correspondant à un maillon de la liste chaînée stockant des entiers.
   En aucun cas, l'utilisateur de la classe ne devra lui-même manipuler des maillons.
   Quelle doit être la visibilité de la classe `fr.umlv.data.Link` ainsi que la visibilité de ses champs ?
   Écrire un `main` de test dans cette classe créant deux maillons contenant les valeurs 13 et 144.
2. Quelle est la commande pour exécuter le `main` de la classe `fr.umlv.data.Link` à partir d'un terminal (pas dans Eclipse) ?
3. Créer une classe `fr.umlv.data.LinkedLink` qui permettra de manipuler une liste chainée par son premier maillon.
   1. `add(int value)` qui ajoute un élément en tête de la liste.
   2. `toString()` qui affiche le contenu de la liste.

Pour tester la classe `fr.umlv.data.LinkedLink`, créer une classe `Main` dans le package `fr.umlv.data.main`.

Exercice 2 - Liste chaînée (suite)

1. Implanter `int get(int index)` qui renvoie la `index`-ième valeur de la liste chainée.
   Que doit-on faire si l'indice est invalide ?
   Vérifier que votre implantation respecte l'adage `blow early, blow often`, sinon changer l'implantation !
2. Dans le but de pouvoir ré-utiliser la liste dans différents codes, changer les classes `fr.umlv.data.LinkedLink` et `fr.umlv.data.Link` pour une implantation plus générique à base d'`Object`.
3. Dans la classe `Main`, créer une liste contenant des chaînes de caractères (au moins 2) et écrire un test affichant la longueur de la deuxième chaîne de la liste.
4. Expliquer pourquoi on est obligé d'ajouter un `cast` dans la méthode `main` et pourquoi en tant que développeur Java, on n'aime pas les casts.

Exercice 3 - Généréfication de LinkedLink

Le but de cet exercice est de "généréfier" les classes `fr.umlv.data.LinkedLink` et `fr.umlv.data.Link`

1. Rappeler quel est l'intérêt d'utiliser un type paramétré ici ?
2. Paramétrer la classe `fr.umlv.data.LinkedLink` pour que celle-ci soit générique.
3. Modifier la classe `fr.umlv.data.main.Main` en conséquence.
4. Dans la classe `fr.umlv.data.LinkedLink`, implanter la méthode `boolean contains(Object o)` indiquant si un objet est ou non contenu dans la liste chaînée.
   Pourquoi `contains` prend un `Object` en paramètre et pas un `T` ou un `E` ?
