BPO
TP 5 - Les listes - Adresse et Message

Exercice 1 - artEoz et les chaînes de caractères

Imaginez un programme pour montrer la différence de représentation entre **String** et **StringBuilder**. Vous pouvez utiliser et modifier les exemples directement proposés par **artEoz** (rubrique “Un exemple?”).

Qu’en déduisez-vous pour leur utilisation?

Exercice 2 - artEoz et les listes

La documentation des classes de la bibliothèque se trouve à l’adresse :

http://docs.oracle.com/javase/8/docs/api/

Retrouver la documentation de la classe **ArrayList**.

1. Dans **artEoz,** exécutez le code ci-dessous (attention au copier-coller ...) :

```
ArrayList<Point> liste = new ArrayList<>(5) ;
liste.add(new Point(0,0));
liste.add(new Point(1,1)) ;
```

2. Quelle est la capacité de la liste créée ? Quelle est sa taille ?

3. Modifiez le code, visualisez les schémas correspondants, et regardez l’évolution de la liste :

   a. ajoutez un **Point** à la position 1 de la liste : `liste.add(1, new Point(10,10)) ;`
   b. ajoutez un **Point** en fin de liste : `liste.add(new Point(3,3)) ;`

   Quelle est la capacité de la liste ? Quelle est sa taille ?

4. Supprimez le **Point** en position 2. Regardez l’évolution de la liste.

5. Que fait l’instruction : `Point p = liste.get(2) ;`

6. Les cases d’indices 3 et 4 sont libres (initialisées avec **null**). Ajoutez un **Point** à l’indice 4. Concluez.

7. Ajoutez successivement 3 points dans la liste. Quelle est la capacité de la liste ? Quelle est sa taille ? Demandez la visualisation des objets morts. Concluez.

Exercice 3 - Les briques de base d'une nouvelle application

Préparation de l'environnement de travail

Sur Arche, vous trouverez un certain nombre de ressources, qu'il convient de télécharger et de ranger correctement dans vos répertoires :

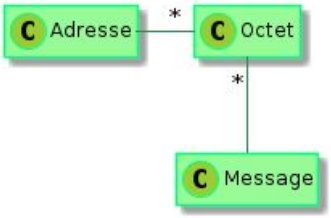
- la classe **reseau.adresses.Octet** (sous forme d'un fichier .class)
- les "squelettes" des classes **reseau.adresses.Adresse** et **reseau.Message**
- la javadoc,
- les deux classes de tests **reseau.tests.TestAdresse** et **reseau.tests.TestMessage**

*À ne pas oublier : la classe **reseau.adresses.Octet** doit être rangée dans le répertoire **reseau/adresses***

Développement des classes

Lire attentivement ce paragraphe en entier, avant de se lancer tête baissée sur le clavier. La majorité des informations ne sont que des rappels de TD.

Le diagramme de classes UML suivant fixe les relations entre ces classes : une adresse et un message sont composés d'un nombre quelconque d'octets.



On décide de représenter une instance de **Adresse** par un tableau d'**Octet** et une instance de **Message** par une **ArrayList** d'**Octet**.

Dans un **Message,** on représente un entier **short** sur un octet et un entier **int** sur deux octets ; on se contente de traiter les entiers positifs.

Les fonctions doivent être toutes implantées en respectant la documentation fournie. La validité des paramètres doit être vérifiée avec l'instruction **assert.** Pour mémoire, l'option **-ea** de la commande **java** indique à la machine virtuelle que les instructions **assert** doivent être exécutées. Par défaut, elles ne le sont pas.

La méthode de développement est guidée par les tests. On a choisi de commencer par développer la classe **Adresse**, donc il est nécessaire de travailler en parallèle avec les deux fichiers **Adresse.java** et **TestAdresse.java**.

Dans la fonction **main** de la classe de test se trouvent déjà quelques petits exemples de tests. Pour réaliser le premier test, il faut écrire uniquement un constructeur d'**Adresse** et la fonction **getNbreOctets** par exemple. Une fois ce premier test réalisé avec succès, vous pouvez enchaîner avec le suivant, il peut s'agir de :

- un autre cas de test pour cette même fonction appelé avec le même constructeur,
- ou bien un autre cas de test pour cette même fonction avec un autre constructeur,
- ou bien un autre cas de test pour une autre fonction.

Il faut faire preuve de rigueur pour ne pas en oublier ; vous pouvez choisir l'ordre dans lequel vous les faites, du moment qu'à la fin, tous les cas de tests sont prévus.

Une mauvaise méthode de travail (donc, à bannir) consiste à écrire d'abord le corps de toutes les fonctions de la classe **Adresse** puis se préoccuper de la compilation et du test en fin de séance. Tout code écrit mais non testé n'a aucune valeur.

Les fichiers **Adresse.java** et **Message.java** et les fichiers de tests **TestAdresse.java** et **TestMessage.java** sont à déposer sur Arche à la fin de la séance et donneront lieu à un dépôt ultérieur pour la version finale.