Programmation & Algorithmique 2

TP - Séance 7

26 avril 2023

Matière visée : `ArrayList`

1 Utilisation des `ArrayList`

Cette section a pour but de proposer des exercices afin de vous familiariser avec l'utilisation de la classe `ArrayList`.

Créez une classe `ArrayListTest`.

Dans cette classe, implémentez les quatre méthodes de classe suivantes :

1. `create( )`
   Entrée : un nombre entier n.
   Sortie : une `ArrayList` contenant n chaine de caractères générées aléatoirement, de taille comprise entre 0 et 100 caractères (de code ASCII entre 32 et 126).
2. `add( )`
   Entrées : une `ArrayList` l et une chaine de caractères s.
   Sortie : /.
   Insère la donnée s dans la liste l.
3. `empty( )`
   Entrée : une `ArrayList` l.
   Sortie : /.
   Vide la liste l.
4. `display( )`
   Entrée : une `ArrayList` l.
   Sortie : /.
   Affiche les éléments de l.

Implémentez une méthode `main` pour cette classe. Dans cette méthode, veuillez utiliser les méthodes précédentes pour créer une `ArrayList`, y ajouter des éléments, la vider et ajouter d'autres éléments. Après chaque modification de la liste, affichez son contenu.

2 `MyArrayList`

Lisez entièrement toute la question et assurez-vous d'avoir bien compris avant d'entamer l'implémentation.

Lors du cours théorique, vous avez vu la manière dont les `ArrayList` sont implémentés en Java. Pour rappel, les `ArrayList` sont implémentés à l'aide d'un tableau T_1, ayant une taille initiale fixée. La force des `ArrayList` est que lorsque l'on y ajoute un élément e alors que le tableau T_1 est rempli, un tableau T_2 de taille plus grande est créé pour représenter l'`ArrayList`. Tous les éléments de T_1 y sont alors copiés et le nouvel élément e peut être inséré.

La manière de choisir la taille du nouveau tableau T_2 (en fonction de la taille de T_1) est ce que l'on va appeler la stratégie de réallocation mémoire. En voici deux exemples :

— Augmenter de 1 la taille du tableau. (1)
Donc ici, $T_2.length = T_1.length + 1$ (exemple vu au cours).
— Multiplier par 2 la taille du tableau. (2)
Donc ici, $T_2.length = T_1.length \times 2$.

Évidemment, cette réallocation mémoire se produit également lorsque T_2 est plein et ainsi de suite.
Cet exercice se déroulera en deux étapes. Vous devrez :

1. créer deux types d'implémentations des `ArrayList` qui vont principalement différer sur la stratégie de réallocation mémoire ; et
2. comparer les différentes stratégies à l'aide d'outils statistiques et graphiques.

2.1 Implémentation des différentes stratégies

Vous allez implémenter deux types d'`ArrayList` basés sur une stratégie de réallocation mémoire différente. L'une sera dite additive et l'autre multiplicative.

L'additive doit être implémentée dans la classe `AdditiveArrayList`. Cette stratégie est une généralisation de l'exemple (1). En effet, lors d'une réallocation mémoire, vous ajouterez un nombre x de cases supplémentaires au tableau ($x = 1$ dans l'exemple). La valeur de x ainsi que la taille initiale du tableau doivent être passées en paramètres du constructeur.

La multiplicative doit être implémentée dans la classe `MultiplicativeArrayList`. Cette stratégie est une généralisation de l'exemple (2). En effet, lors d'une réallocation mémoire, vous multiplierez par y la taille du tableau ($y = 2$ dans l'exemple). La valeur de y ainsi que la taille du tableau initial doivent être passées en paramètres du constructeur.

Ces deux classes devront hériter de la classe `MyArrayList` qui comporte ce qui est en commun entre les deux sous-classes.

Remarque : notez que, comme vous instanciez un `ArrayList` contenant des `String` en utilisant la syntaxe `ArrayList<String>` (vous utilisez un *generic type*), il serait intéressant de pouvoir instancier de la même façon un `MyArrayList`. Pour cela, déclarez votre classe avec la syntaxe `class MyArrayList<T>`. Le `T` représente dans la suite le type du contenu de votre classe. Vous devrez cependant utiliser un tableau interne de type `Object[]`. Aussi, afin de retourner un élément du tableau à l'utilisateur, vous allez devoir effectuer un *unchecked cast*. Vous pouvez utiliser la syntaxe suivante :

```
@SuppressWarnings("unchecked")
final T o = (T) array[i];
return o;
```

Si vous vous sentez dépassé par la précision précédente, vous pouvez également ignorer ce paragraphe ainsi que l'utilisation du `<T>`, et utiliser des `Object` partout, en attendant de voir les *generics* au cours.

Vos classes devront supporter les méthodes suivantes :

— `size()` qui retourne la taille logique de l'`ArrayList` (le nombre d'éléments stockés) ;
— `physicalSize()` qui retourne la taille physique de l'`ArrayList` (la taille du tableau « en mémoire » utilisé par votre `ArrayList`) ;
— `get(int i)` qui retourne le `i`^{e} élément de l'`ArrayList`. Cette méthode doit lancer une exception si `i` n'est pas un indice autorisé pour l'`ArrayList` ;
— `add(T o)` ajoute l'objet `o` à la fin de l'ArrayList ;
— `add(int i, T o)` ajoute l'objet `o` en position `i` dans l'`ArrayList`. Cette méthode doit lancer une exception si `i` n'est pas un indice autorisé pour l'`ArrayList` ;
— `getV()` retourne l'évaluation de la fonction V (définie ci-dessous) en la taille logique actuelle de l'`ArrayList` ;
— `getW()` retourne l'évaluation de la fonction W (définie ci-dessous) en la taille logique actuelle de l'`ArrayList` ;
— `getX()` retourne l'évaluation de la fonction X (définie ci-dessous) en la taille logique actuelle de l'`ArrayList` ; et

— `getY()` retourne l'évaluation de la fonction Y (définie ci-dessous) en la taille logique actuelle de l'`ArrayList`.

Lisez bien la deuxième partie avant de vous lancer dans l'implémentation de la première, car des travaux supplémentaires peuvent être réalisés par les méthodes citées précédemment afin de faciliter le calcul des statistiques demandé dans cette deuxième partie.

2.2 Comparaison des différentes stratégies

Pour la réalisation de cette partie, vous aurez besoin de l'outil graphique **Gnuplot** qui va vous permettre, à partir d'un fichier texte "bien formaté", d'afficher des fonctions.

Avant de vous décrire comment utiliser **Gnuplot** et comment formater les fichiers texte, nous allons définir les différentes fonctions qui devront être affichées pour les différentes stratégies de réallocation mémoire utilisées par vos `ArrayList`.

Le domaine de chacune des fonctions représentera le nombre d'ajouts d'éléments déjà effectués dans l'`ArrayList` (ou de manière équivalente, le nombre d'éléments dans l'`ArrayList`). Il est donc de la forme $[0, m] \subset \mathbb{N}$. Vous devez représenter les fonctions suivantes :

1. V telle que $V(n)$ représente la taille physique de l'`ArrayList` après n ajouts d'éléments ;
2. W telle que $W(n)$ représente le nombre de copies cumulées effectuées après n ajouts d'éléments ;
3. X telle que $X(n)$ représente le pourcentage d'utilisation effective du tableau après n ajouts d'éléments ; et
4. Y telle que $Y(n)$ représente le pourcentage moyen de l'utilisation effective du tableau après n ajouts d'éléments.

Par exemple, prenons le cas d'une stratégie multiplicative de facteur 2 ayant un tableau de taille initiale 3, nous avons :

1. $V(0) = V(1) = V(2) = V(3) = 3$; $V(4) = V(5) = V(6) = 6$; $V(7) = 12$;
2. $W(0) = W(1) = W(2) = W(3) = 0$; $W(4) = W(5) = W(6) = 3$; $W(7) = 9$;
3. $X(0) = 0$; $X(1) = 0,33$; $X(2) = 0,66$; $X(3) = 1$; $X(4) = 0,66$; $X(5) = 0,83$;
4. $Y(0) = 0$; $Y(1) = 0,16$; $Y(2) = 0,33$; $Y(3) = 0,5$; $Y(4) = 0,53$; $Y(5) = 0,58$.

Remarque : À chaque ajout d'un élément dans un `ArrayList`, vous devez mettre à jour des compteurs (variables d'instances) qui vont vous permettre de calculer facilement l'évaluation de ces fonctions en la taille courante de l'`ArrayList`. Par contre, il vous est interdit de stocker le résultat de toutes les évaluations.

La section suivante concerne l'utilisation de Gnuplot et de la manière dont il faut formater vos fichiers texte.

2.3 Gnuplot

Gnuplot est un utilitaire qui permet de tracer des graphiques.

Voici quelques commandes et exemples qui vous permettront de tracer des fonctions mathématiques et des courbes en fonction d'informations stockées dans un fichier.

Pour lancer gnuplot, tapez `gnuplot` dans l'invite de commande.

— `gnuplot> plot x`
trace la fonction x

— `gnuplot> plot sin(x)`
trace la fonction $sin(x)$

— `gnuplot> plot sin(x)/x`
trace la fonction $sin(x)/x$

— `gnuplot> plot x**2`
trace la fonction x^2

— `gnuplot> plot "data.txt" index 0 using 1:2 title 'cumulatives copies' with lines`

  — `"data.txt"` : considère le fichier `data.txt`
  — `index 0` : considère le premier bloc d'information (les blocs sont séparés par des lignes vides).
  — `using 12` : trace les points en considérant en `x` la première colonne et en `y` la 2ème colonne.
  — `title 'cumulatives copies'` : utilise *cumulatives copies* comme légende des informations.
  — `with lines` : relie les points par une ligne.

— `gnuplot> plot "data.txt" index 0 using 1:2 title 'cumulatives copies ADD' with lines, "data.txt" index 1 using 12 title 'cumulatives copies MULT' with lines`
trace les deux fonctions sur le même graphique

— `gnuplot> replot x**2`
ajoute le tracé de la fonction x^2 au graphique courant.

— `set xrange [0:20]`
calibre le graphique pour n'afficher que la plage des x allant de 0 à 20. (Faire `replot` pour actualiser le graphique)

— `set yrange [-1:15]`
calibre le graphique pour n'afficher que la plage des y allant de -1 à 15. (Faire `replot` pour actualiser le graphique)

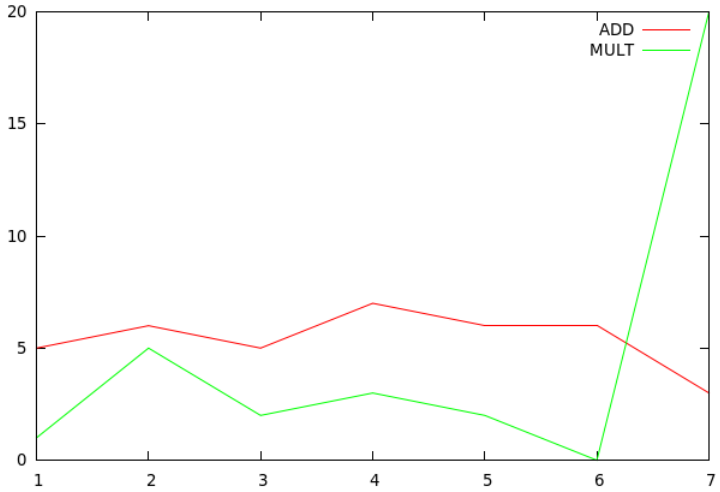
Par exemple, si le fichier `"data.txt"` contient les informations suivantes :

```
1 1 5 14
2 5 6 14
3 2 5 6
4 3 7 8
5 2 6 4
6 0 6 9
7 20 3 6



8 4 2 36
9 4 3 14
10 6 2 15
```

où les différentes valeurs d'une même ligne sont séparées par un espace et les différents blocs sont distingués par au moins une ligne blanche, les commandes suivantes :
`gnuplot> plot "data.txt" index 0 using 1:3 title 'ADD' with lines`
`gnuplot> replot "data.txt" index 0 using 1:2 title 'MULT' with lines`
permettent de dessiner le graphique suivant :



2.4 Testeur

Le testeur va utiliser vos différentes méthodes afin d'observer l'efficacité de vos `ArrayList` en fonctions des paramètres.

Pour chacun des `ArrayList` utilisés pour ce test, vous allez ajouter un certain nombre d'éléments en imprimant après chaque ajout la valeur de retour de chacune des fonctions V, W, X et Y. En les imprimant de telle manière à ce que le résultat de l'impression respecte le formatage utilisé par Gnuplot.

Afin de stocker vos informations dans un fichier texte, il vous suffit d'ajouter une redirection de la sortie standard vers un fichier « `java maCommande > nomDuFichier.txt` » dans l'invite de commande lors de l'exécution de votre programme. Alternativement, vous pouvez également écrire dans un fichier directement en Java, en utilisant `BufferedWriter` et `FileWrite`.

Voici une liste de paramétrisations qu'il vous est demandé de comparer (avec l'ajout de 10000 éléments) :

Type de stratégie	**Facteur**	**Taille initiale du tableau**
Additive	1	2
Additive	10	2
Additive	100	2
Additive	10	100
Additive	100	100
Multiplicative	2	1
Multiplicative	3	1
Multiplicative	5	1
Multiplicative	10	1
Multiplicative	5	100