**Exercice de révision**

Cet exercice est tiré de l’examen de janvier 2021.

Il a été modifié pour tenir compte des acquis des étudiants à ce stade du quadrimestre.

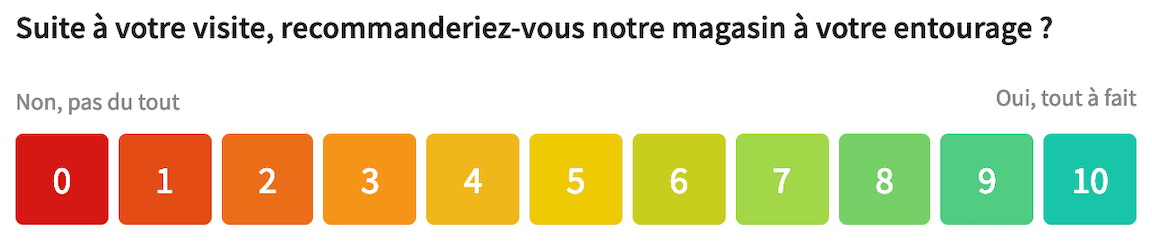
Les 2 dernières méthodes demandent pas mal de maîtrise sur les tables. Cette maîtrise est le résultat de nombreux exercices sur les tables durant plusieurs semaines. Il n’est donc pas étonnant que vous éprouviez des difficultés à ce stade de l’année. Des « coups de pouce » se trouvent en fin de ce document. Il devrait permettre d’aider de nombreux étudiants. N’allez pas le consulter trop vite.

**Mise en situation :**

Il existe différents types de questionnaires de satisfaction client. Parmi ceux-ci, on retrouve le questionnaire avec échelle de 0 à 10, appelé questionnaire NPS.

Les réponses des clients sont des cotes comprises entre 0 et 10.

Voici un exemple de question NPS:



Dans le cadre de cet exercice, on ne va pas s’intéresser à tout un questionnaire, mais à une question de ce questionnaire et à l’analyse des réponses obtenues.

Voici un exemple de réponses obtenues :

5 9 5 6 8 7 6 0 1 5 9 2 8 2 6 3 9 7 8 8 3 6 3 6 4 8 4 8 4 6 4 5 9 9 5 9 10 5 6 7 7 9 7 8 7 9 7 0 7 8 5 8 6 7 8 6 8 7 6 8 7 8 9 6 7 8 10 1 10 1 6 3 5 4 5

Constatations :

Cette table peut être immense ! Sa taille dépend du nombre de clients qui ont participé à l’enquête.

Les réponses ne sont pas vraiment interprétables sous ce format.

Il faut donc les exploiter autrement en tirant quelques statistiques comme la moyenne et la médiane.

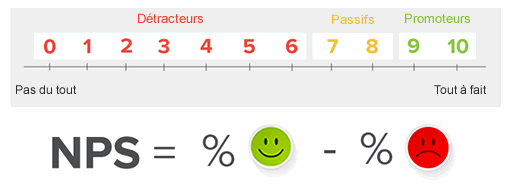
On peut même « résumer » ces réponses dans une table de fréquences. Et cette table de fréquences pourrait aboutir à un graphique :

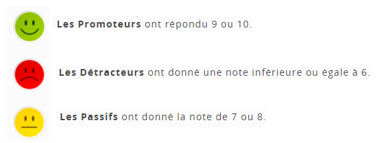




Le score NPS a donné son nom à ce type de question.

Son calcul est très simple : % promoteurs - % détracteurs





**Travail à effectuer**

Vous allez compléter la classe *QuestionNPS*. La classe *TestQuestionNPS* permet de tester les différentes méthodes à compléter.

**Suivez bien la *JavaDoc* et les consignes indiquées dans la classe.**

Il est interdit de passer par des classes *Arrays*, *System* ou autres !

**Calcul de la médiane : mediane()**

L’algorithme pour calculer la médiane est imposé.

La médiane est la valeur centrale d’un ensemble de valeurs

La moitié des valeurs est ≤ médiane, l’autre moitié est ≥.

La médiane est donc la ((n/2) + 1)ème plus petite valeur parmi n valeurs!

Exemple : 2 8 9 1 9 7 2 🡪 la médiane = 7 (1 2 2 **7** 8 9 9)

Algorithme

On définit le nombre d’étapes : (nbReponses/2) + 1

A chaque étape, on recherche la plus petite valeur dans le tableau et on la remplace par 11 (pour ne plus retomber sur elle au tour suivant).

La plus petite valeur trouvée à la dernière étape est la médiane.

Attention, pour cet algorithme, **il est indispensable de travailler sur une table temporaire** dans laquelle on a copié toutes les réponses qui se trouvent dans tableReponses.

Exemple :

2 8 9 1 9 7 2

7 réponses donc nombre d’étapes = (7/2) + 1 = 4

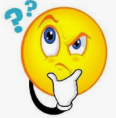
Etape 1 : 2 8 9 **1** 9 7 2 🡪 min = 1 🡪 2 8 9 **11** 9 7 2

Etape 2 : **2** 8 9 11 9 7 2 🡪 min = 2 🡪 **11** 8 9 11 9 7 2

Etape 3 : 11 8 9 11 9 7 **2** 🡪 min = 2 🡪 11 8 9 11 9 7 **11**

Etape 4 : 11 8 9 11 9 **7** 11 🡪 min = 7 🡪 11 8 9 11 9 **11** 11

La médiane est 7.

****

**Coup de pouce :** **méthode tableFrequences()**

5 9 5 6 8 7 6 0 1 5 9 2 8 2 6 3 9 7 8 8 3 6 3 6 4 8 4 8 4 6 4 5 9 9 5 9 10 5 6 7 7 9 7 8 7 9 7 0 7 8 5 8 6 7 8 6 8 7 6 8 7 8 9 6 7 8 10 1 10 1 6 3 5 4 5

|  |  |
| --- | --- |
| 0 | 2 |
| 1 | 3 |
| 2 | 2 |
| 3 | 4 |
| 4 | 5 |
| 5 | 8 |
| 6 | 13 |
| 7 | 12 |
| 8 | 14 |
| 9 | 9 |
| 10 | 3 |

Il faut tout d’abord bien comprendre le contenu de cette table de fréquences :

A l’indice 0, la valeur 2 signifie que parmi toutes les réponses, il y a 2 réponses 0.

A l’indice 1, la valeur 3 signifie que parmi toutes les réponses, il y a 3 réponses 1.

A l’indice 2, la valeur 2 signifie que parmi toutes les réponses, il y a 2 réponses 2.

…

Il y a 11 réponses possible : 0 1 2 3 … 10 🡪 la table de fréquences à créer doit contenir 11 cases.

La table de fréquences est une table de compteurs.

Avant de parcourir la table des réponses, tous ces compteurs = 0

|  |  |
| --- | --- |
| 0 | 0 |
| 1 | 0 |
| 2 | 0 |
| 3 | 0 |
| 4 | 0 |
| 5 | 0 |
| 6 | 0 |
| 7 | 0 |
| 8 | 0 |
| 9 | 0 |
| 10 | 0 |

On parcourt la table des réponses :

Il faut une boucle for :

for (int i = 0; i < tableReponses.length; i++)

**5** 9 5 6 8 7 6 0 1 5 9 2 8 2 6 3 9 7 8 8 3 6 3 6 4 8 4 8 4 6 4 5 9 9 5 9 10 5 6 7 7 9 7 8 7 9 7 0 7 8 5 8 6 7 8 6 8 7 6 8 7 8 9 6 7 8 10 1 10 1 6 3 5 4 5

Réponse 5 🡪 on incrémente le compteur associé à la réponse 5. C’est à l’indice 5 de la table que se trouve ce compteur.

|  |  |
| --- | --- |
| 0 | 0 |
| 1 | 0 |
| 2 | 0 |
| 3 | 0 |
| 4 | 0 |
| 5 | 1 |
| 6 | 0 |
| 7 | 0 |
| 8 | 0 |
| 9 | 0 |
| 10 | 0 |

A ce stade du parcours, on a rencontré 1 x la réponse 5

5 **9** 5 6 8 7 6 0 1 5 9 2 8 2 6 3 9 7 8 8 3 6 3 6 4 8 4 8 4 6 4 5 9 9 5 9 10 5 6 7 7 9 7 8 7 9 7 0 7 8 5 8 6 7 8 6 8 7 6 8 7 8 9 6 7 8 10 1 10 1 6 3 5 4 5

Réponse 9 🡪 on incrémente le compteur de la réponse 9. C’est à l’indice 9 de la table que se trouve ce compteur.

|  |  |
| --- | --- |
| 0 | 0 |
| 1 | 0 |
| 2 | 0 |
| 3 | 0 |
| 4 | 0 |
| 5 | 1 |
| 6 | 0 |
| 7 | 0 |
| 8 | 0 |
| 9 | 1 |
| 10 | 0 |

A ce stade du parcours, on a rencontré 1 x la réponse 5 et 1 x la réponse 9.

5 9 **5** 6 8 7 6 0 1 5 9 2 8 2 6 3 9 7 8 8 3 6 3 6 4 8 4 8 4 6 4 5 9 9 5 9 10 5 6 7 7 9 7 8 7 9 7 0 7 8 5 8 6 7 8 6 8 7 6 8 7 8 9 6 7 8 10 1 10 1 6 3 5 4 5

Réponse 5 🡪 on incrémente le compteur de la réponse 5. C’est à l’indice 5 de la table que se trouve ce compteur.

|  |  |
| --- | --- |
| 0 | 0 |
| 1 | 0 |
| 2 | 0 |
| 3 | 0 |
| 4 | 0 |
| 5 | 2 |
| 6 | 0 |
| 7 | 0 |
| 8 | 0 |
| 9 | 1 |
| 10 | 0 |

A ce stade du parcours, on a rencontré 2 x la réponse 5 et 1 x la réponse 9.

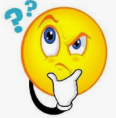
Etc…

Après avoir parcouru toutes les réponses de la table des réponses, la table des fréquences sera :

|  |  |
| --- | --- |
| 0 | 2 |
| 1 | 3 |
| 2 | 2 |
| 3 | 4 |
| 4 | 5 |
| 5 | 8 |
| 6 | 13 |
| 7 | 12 |
| 8 | 14 |
| 9 | 9 |
| 10 | 3 |

On a rencontré 2 x la réponse 0, 3 x la réponse 1, 2 x la réponse 2, …

En un seul parcours de la table des réponses, on a obtenu la table des fréquences.

****

**Coup de pouce :** **méthode mediane()**

Il faut créer une table temporaire qui contient le même contenu que la table des réponses.

Pour transférer les réponses de la table des réponses vers cette table temporaire, il faut un parcours de table.

(Si vous ne voyez pas comment effectuer ces transferts, regardez le code du constructeur. Le constructeur copie une table dans une autre table.)

Il ne faut pas être distrait ! Il faut toujours utiliser la table temporaire. La table des réponses doit rester inchangée à la suite de l’appel de cette méthode.

L’algorithme demande un nombre d’étapes bien déterminé : (nbReponses/2) + 1.

Il faut donc une boucle for :

for (int etape = 1; etape <= nbEtapes; etape++)

A chaque étape, il faut trouver la réponse min dans la table temporaire. Cette recherche demande une boucle for :

for (int i = 1; i < copie.length; i++)

Voici le squelette de cet algorithme (à compléter) :

// instructions avant de lancer l’algorithme

for (int etape = 1; etape <= nbEtapes; etape++){

//instructions nécessaires à chaque étape pour rechercher le min

for (int i = 1; i < copie.length; i++){

//instructions nécessaires à chaque réponse parcourue

}

//instructions nécessaires à la fin d’une étape

}

Votre code sera plus clair en introduisant une variable **min** et une variable **indiceMin.**

N’oubliez pas la notion de visibilité d’une variable. La variable **min** contient la valeur renvoyée.

A chaque recherche de min, il faut réinitialiser la variable **min**.

Pour l’initialisation du min, il y a 2 solutions possibles : une grande valeur (11) ou la 1ère valeur de la table.