NOM :	PRÉNOM :	GROUPE :
Lefort	**Yanis**	**1i1**

Méthode : slideRocks

1. DESCRIPTION DU « QUOI » = COMPRENDRE LE PROBLÈME

Nom de la méthode?

slideRocks

Sa tâche ? *Expliquer brièvement le traitement que doit réaliser la fonction/méthode*

Dans un tableau de caractères, déplacer toutes les pierres à l'extrêmité gauche/droite

Préconditions (= ses entrées et les valeurs admises) ? *Lister et nommer les données nécessaires à la réalisation du traitement – ces données peuvent varier d'une exécution à l'autre de la méthode. Dans un contexte orienté objets, listez aussi les champs de l'objet dont la méthode a besoin mais qu'elle ne modifie pas.*

A côté de chaque donnée ou champ de la liste, indiquez les éventuelles conditions qu'il/elle doit remplir pour que la méthode puisse réaliser le traitement demandé.

Une référence non null vers un tableau de caractères

Un booléen déterminant dans quel sens aller (vrai : gauche , faux : droite)

Postconditions (= sa sortie et les garanties concernant celle-ci) ? *Indiquer la donnée qui doit être retournée au terme du traitement réalisé par la fonction. Dans un contexte orienté objets, listez et nommez aussi les champs de l'objet que la méthode initialise ou modifie.*

A côté du résultat et des champs initialisés ou modifiés, indiquez les conditions qu'il doit remplir pour être considéré comme correct.

On a déplacé les pierres vers la gauche ou la droite en fonction du choix

Son plan de test = résolution d'au moins six cas concrets : 2 cas « normaux », 1 cas par « branchement » réalisé par la méthode, 1 cas d'erreur géré par la méthode, 1 cas limite (ensemble vide, première ou dernière valeur, zéro ou nul, etc.), éventuellement un cas de dépassement de capacité, 1 à 2 combinaisons de cas

Pour chacun, indiquer 1. le type de cas 2. les entrées spécifiées et les champs utilisés en lecture 3. les effets attendus = le résultat et les valeurs des champs d'objets initialisés ou modifiés

- [O] , true ➔ [O]
- [O # O] , false ➔ [O# O]
- [O] , true ➔ [O] *(Ce cas ne doit pas générer d'erreur ni modifier le tableau)*
- [] , true ➔ [] *(Ce cas ne doit pas générer d'erreur ni modifier le tableau)*
- [O# O O A] , false ➔ **Erreur : caractères autres que (O, #, ␣) trouvés**

2. FORMALISATION DES TESTS = PRÉPARER LEUR ÉCRITURE EN JAVA

La traduction de chaque test du plan de test

Ecrivez chaque test en respectant le schéma Given-When-Then. Exemple : étant donné un objet dictionnaire contenant les 5 mots arbre, fleur, chien, chat, ordinateur et leurs définitions, quand j'appelle la méthode de recherche avec le mot « arbre » en entrée, alors j'obtiens bien la définition d'un arbre

- Etant donné le tableau contenant les éléments suivants : [␣ ␣ O ␣ ␣] et **false** pour la direction, alors le résultat renverra : [␣ ␣ ␣ ␣ O]
- Etant donné le tableau contenant les éléments suivants : [␣ O ␣ # ␣ ␣ O] et **true** pour la direction, alors le résultat renverra : [O ␣ ␣ # O ␣ ␣]

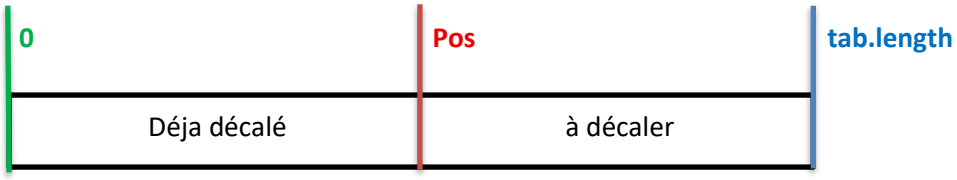
3. DESCRIPTION DU « COMMENT » = IMAGINER UN ALGORITHME

Nom de l'algorithme ? *Si vous appliquez un algorithme célèbre, indiquez son nom.*

Boucle avec branchements

Dessin. Réalisez un dessin de la situation finale de votre problème et déduisez-en une situation plus générale

- Si l'argument de la direction est false(droite) alors on parcourt le tableau en commençant par la fin. Sinon, on commence par le début du tableau. Ici je vais expliquer le cas ou l'on décale vers la droite



Ses étapes ? *Répertorier textuellement ou en « pseudocode » ou encore sous forme d'organigramme, dans l'ordre, les étapes principales que la méthode doit appliquer pour résoudre le problème pour toutes les entrées possibles. Vérifiez que votre algorithme « fonctionne » au moins pour tous les cas de tests prévus ! Assurez-vous de la* ***terminaison*** *de votre algorithme*

slideRocks(tableau, direction)

- D'abord on s'assure que le tableau contient les caractères autorisés. (O # ␣)
- On initialise une boucle for. Si l'argument de la direction est false(droite) alors on parcourt le tableau en commençant par la fin. Sinon, on commence par le début du tableau. Ici je vais expliquer le cas ou l'on décale vers la gauche
- Si on tombe sur un mur (#) on stocke l'index où se trouve le mur dans une variable.
- Si on tombe sur une pierre (O), on la décale à l'endroit de l'espace le plus à gauche avant le mur trouvé
- Si on tombe sur un un espace (␣) on ne fait rien.
- La boucle s'arrete quand on a parcouru et modifié tout le tableau.

4. CODAGE = PROGRAMMER ET VALIDER LA SOLUTION EN JAVA