NOM :	PRÉNOM :	GROUPE : **1I6**

Méthode : fusionEtDecalage()

1. DESCRIPTION DU « QUOI » = COMPRENDRE LE PROBLÈME

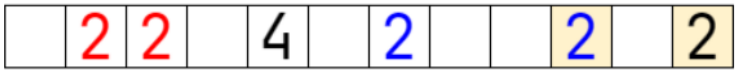
Nom de la méthode?

fusionEtDecalage()

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

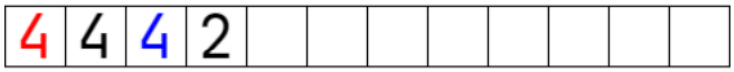
La méthode doit capable de déplacé des puissances de 2 dans un tableau, contenant soit des puissances de 2 soit ou des vides (représentées par un 0), vers la droite ou vers la gauche et lorsque deux puissances de 2 sont identiques, les fusionner sur un seul tour de boucle, un nombre fusionner ne peut pas être

Ce tableau en entrée :

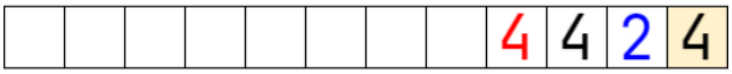


Retourne le tableau trié :

Si vers la gauche



Si vers la 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 attributs de l'objet dont la méthode a besoin mais qu'elle ne modifie pas.*

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

Une référence non-nulle vers un tableau d'entiers composés seulement de 0 et de puissances de 2.

- int[] tableau

Un booléen comme paramètre (true : déplacement vers la gauche, false : vers la droite)

- boolean direction

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 attributs de l'objet que la méthode initialise ou modifie.*

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

Conservé

La valeur totale du tableau ne change pas

Changement

Nombres poussés vers la gauche, tous les zéros vont à droite

Nombres poussés vers la droite, tous les zéros vont à gauche

Si deux valeurs consécutives sont identiques, première valeur est remplacée par 0 et seconde valeur est doublée. 22 -> 4 -> déplacement du 4 à tableau[pos] et remplacement à la position des 2 valeurs par 0

Condition de sortie

vers la gauche : taille du tableau atteint

vers la droite : quand on sort du tableau par la gauche : -1

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 attributs utilisés en lecture 3. les effets attendus = le résultat et les valeurs des attributs d'objets initialisés ou modifiés

Cas normaux

- {0, 2, 2, 0, 4, 0} -> gauche : {4, 4, 0, 0, 0, 0} ou droite : {0, 0, 0, 0, 4, 4}
- {4, 4, 4, 4, 4, 4} -> gauche : {8, 8, 8, 0, 0, 0} ou droite : {0, 0, 0, 8, 8, 8}
- {2, 2, 4, 4, 8, 8} -> gauche : {4, 8,16, 0, 0, 0} ou droite : {0, 0, 0, 4, 8, 16}

Cas de branchement

(gauche – droite : même chose qu'au-dessus)

Cas d'erreur

- Référence null :

assertThrows(NullPointerException.class, () -> {Jeu128. fusionEtDecalage(null);});

Cas limites

- Un tableau vide ne doit pas être modifié (et ne doit pas générer d'erreurs) : {} -> {}
- Un tableau d'un élément ne doit pas être modifié : {2} -> {2}
- Un tableau ne contenant que des 0 ne doit pas être modifié : {0, 0, 0, 0} -> {0, 0, 0, 0}
- {0, 2} -> gauche : {2, 0} ou droite : {0, 2} (cas normal proche de la limite)
- {2, 2} -> gauche : {4, 0} ou droite : {0, 4} (cas normal proche de la limite)
- {2, 4} -> gauche : {2, 4} ou droite : {2, 4} (cas déjà compressé)

Combinaisons de cas

Pas de cas ici

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

Les tests d'au-dessus sous forme de phrases

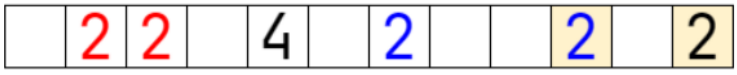
3. DESCRIPTION DU « COMMENT » = IMAGINER UN ALGORITHME

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

Algo perso

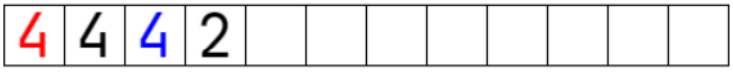
Dessin. Réalisez un dessin de la situation finale de votre problème et déduisez-en une situation plus générale

Ce tableau en entrée :

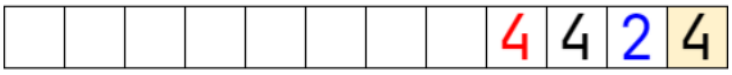


Retourne le tableau compressé :

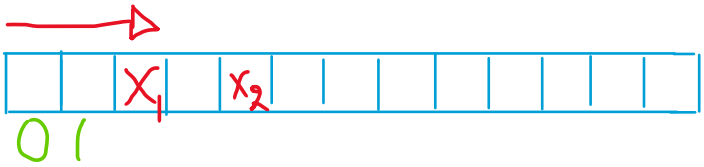
Si nombres vers la gauche



Si nombres vers la droite



Si true :



Si déplacement vers la droite :

tableau.length - position = (

tableau.length - position = 0

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*

INITIALISATION

while(GARDIEN) {

CORPS DE LA BOUCLE

} FIN

INITIALISATION

Un compteur représentant le nombre de puissance de 2 avant déplacement, qui décrémente à chaque fois que 2 valeurs fusionnent et qui servira d'index final à check pour placer dernière valeur ou non avant de sortir de la boucle :

- int compteur = 0

Une variable contenant la valeur à déplacer (check si la valeur suivante est identique ou non afin de la fusionner avant de la déplacer) :

- int valeur

Une variable débutant à 0 représentant la position à laquelle chaque valeur sera placée, s'incrémentant de 1 quand celle-ci a été placée :

- int position = 0

Incrémentation dans boucle :

- Si booléen = true -> i = 0
- Si booléen = false -> i = tableau.length-1

GARDIEN

Si nombres vers la gauche :

i < tableau.length

Si nombres vers la droite :

i >= 0

CORPS

Si nombres vers la gauche :

En partant de i = 0

- Si valeur == 0
  - on boucle jusqu'au 1er non 0
  - quand trouvé : valeur = tableau[i]
  - tableau[i] = 0
  - compteur++

on continue boucle jusqu'au non 0 suivant :

- Si tableau[i] == valeur
  - valeur += valeur
  - tableau[position] = valeur
  - valeur = 0
  - position++
  - tableau[i] = 0

- Si tableau[i] != valeur
  - tableau[position] = valeur
  - position++
  - valeur = tableau[i]
  - tableau[i] = 0
  - compteur ++

i++

Si nombres vers la droite :

En partant de i = tableau.length-1

- Si valeur == 0
  - on boucle jusqu'au 1er non 0
  - quand trouvé : valeur = X1
  - tableau[i] = 0
  - compteur++

on continue boucle jusqu'au non 0 suivant :

- Si tableau[i] == valeur
  - valeur += valeur
  - tableau[tableau.length - 1 - position] = valeur
  - tableau[i] = 0
  - valeur = 0

- positon++

- Si tableau[i] != valeur
  - tableau[tableau.length - 1 - position] = valeur
  - valeur = tableau[i]
  - tableau[i] = 0
  - position++
  - compteur ++

i—

FIN

Après boucle

Si nombres vers la gauche :

- Si valeur == 0 fini
- Si valeur != 0 && compteur != position alors tableau[position] = valeur

Si nombres vers la droite :

- Si valeur == 0 fini
- Si valeur != 0 && compteur != position alors tableau[tableau.length – 1 - position] = valeur

TERMINAISON

Return le tableau avec les valeurs déplacées

4. CODAGE = PROGRAMMER ET VALIDER LA SOLUTION EN JAVA