

Contrôle terminal info1 / Semestre 1

R1.01 : Initiation au développement

Nom du responsable :	Adam Michel
Date du contrôle :	8 novembre 2021
Durée du contrôle :	2 heures
Nombre total de pages :	15 pages
Impression :	Recto – Verso
Documents autorisés :	Aucun
Calculatrice autorisée :	non
Réponses :	Sur le sujet

- Les exercices sont indépendants.
- Le barème est donné à titre indicatif.
 1. Partie "Compréhension de programme" : 4 points, soit 24 minutes
 2. Partie "Conception de programme" : 8 points, soit 48 minutes
 3. Partie "Procédures et fonctions" : 8 points, soit 48 minutes
- Le document est à rendre dans son intégralité avec les réponses.
- L'ensemble des connaissances vues dans la ressource est utilisable dans tous les exercices.
- Les réponses doivent être justifiées. L'absence de justification sera considérée comme une réponse incorrecte.
- Un programme mal ou pas indenté sera sanctionné au niveau de la notation.

Compréhension de programme (4 points)

Nom :

Prénom :

Groupe :

Soit le programme Java :

```
1  /**
2   * Programme inconnu
3   * @author Les enseignants de R1.01
4   */
5
6  class Mystere {
7      void principal() {
8          int[] t = {0, 1, 1, 3, 0, 4, 3, 3, 0, 0};
9          int[] c = new int[10];
10
11          int i = 0;
12          while (i < t.length) {
13              int j = t[i];
14              c[j] = c[j] + 1;
15              i = i + 1;
16          }
17      }
18  }
```

Question 1 (1 point)

Que contient le tableau `int[] c` à l'issue de l'exécution du programme `Mystere.java`? Plus généralement quel est le rôle du programme?

Question 2 (1 point)

Avec le tableau `int[] t = {0, 1, 1, 3, 10, 4, 3, 3, 0, 0}`; l'exécution se termine avec le message :

```
java.lang.reflect.InvocationTargetException
at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.base/java.lang.reflect.Method.invoke(Method.java:566)
at Start.main(Start.java:17)
Caused by: java.lang.ArrayIndexOutOfBoundsException: Index 10 out of bounds for length 10
at Mystere.principal(Mystere.java:14)
... 5 more
```

Expliquer la cause de l'erreur.

Question 3 (2 points)

Modifier le programme pour qu'il réalise le même calcul sans que ce type d'erreur puisse se produire.

Conception de programme (8 points)

Nom :

Prénom :

Groupe :

Soit la déclaration suivante :

```
int[] valeurs = new int[XX];
```

- La valeur **XX** est un nombre entier **pair**.
- Le tableau **valeurs** ne contient que des 0 et des 1.

Attention tous les programmes demandés dans cette partie nécessitent des boucles. Évidemment vous devrez donc rendre pour chaque boucle :

- le principe de la boucle, s'il n'est pas donné dans le sujet,
- le corps de boucle,
- les conditions de sorties,
- la condition de continuation,
- l'initialisation,
- la terminaison.

Il n'est donc pas utile d'écrire le programme complet de la boucle. La conception méthodique suffit. Il vous est possible d'écrire la boucle **while** directement au brouillon, mais il faut compléter ensuite les parties demandées sur la copie.

Dans chaque exercice le tableau **valeurs** est supposé déjà initialisé avec uniquement des 0 et des 1.

Question 1 (1 point)

Écrire un programme qui vérifie que le nombre de 0 est identique au nombre de 1 dans `valeurs`.
Un message final indique si la propriété est vérifiée ou pas.

Le principe suivant doit être appliqué :

- le tableau est parcouru intégralement de la première à la dernière case,
- deux variables `nb0` et `nb1` comptabilisent respectivement le nombre de 0 et de 1 ;
- la propriété est vérifiée si et seulement si les deux valeurs `nb0` et `nb1` sont identiques.

Corps de boucle

Conditions de sortie

Condition de continuation

Initialisation

Terminaison

Conception de programme (suite)

Nom :

Prénom :

Groupe :

Question 2 (1 point)

En fait, il n'est pas utile de parcourir tout le tableau quand le nombre de **nb0** ou **nb1** devient supérieur à la moitié du nombre de valeurs. En plus clair, si le tableau contient 10 valeurs, il est possible de s'arrêter quand le nombre de **nb0** ou de **nb1** est égal à 6.

Modifier le programme de la "question 1" avec **une couleur différente** pour prendre en compte cette nouvelle information.

Question 3 (2 points)

Une autre propriété est demandée : il ne peut y avoir plus de deux 0 ou plus de deux 1 à suivre.

La séquence 0010011011 est correcte, alors que 0001101101 ne l'est pas, car trois 0 se suivent.

Écrire le programme qui vérifie cette propriété et uniquement celle-la. Un message final indique si la propriété est vérifiée ou pas.

Principe

Corps de boucle

Conditions de sortie

Condition de continuation

Initialisation

Terminaison

Conception de programme (suite)

Nom :

Prénom :

Groupe :

Question 4 (3 points)

Pour qu'une ligne soit correcte, il faut et il suffit :

- qu'elle contienne autant de 0 que de 1,
- qu'elle ne contienne pas plus de deux 0 ou plus de deux 1 à suivre.

Écrire un programme qui en **une seule boucle** vérifie les deux propriétés. Un message final indique quelles propriétés sont ou pas respectées.

Principe

Corps de boucle

Conditions de sortie

Condition de continuation

Initialisation

Terminaison

Question 5 (1 point)

Pour vérifier les deux propriétés d'une ligne, deux options sont possibles :

- écrire une seule boucle qui vérifie les deux propriétés,
- écrire deux boucles en séquence, une pour vérifier chaque propriété.

Donner les avantages de chacune des deux solutions.

Méthodes (8 points)

Nom :

Prénom :

Groupe :

Attention

- même si vous ne parvenez pas à répondre à une question, vous pouvez utiliser la fonction ou la méthode dans les questions suivantes,
- les conceptions méthodiques des éventuelles boucles ne sont pas demandées.

Question 1 (2 points)

Écrire la méthode :

```
/**  
 * décale les entiers d'un tableau d'une position vers la gauche  
 * L'élément en 0 se retrouve à la fin du tableau  
 * @param tab tableau d'entiers  
 */  
void decalerGauche (int[] tab)
```

A titre d'exemple, le tableau {3, 10, 6, 20, 7} devient {10, 6, 20, 7, 3}.

Question 2 (1 point)

En utilisant `decalerGauche()`, écrire la méthode :

```
/**
 * décale les entiers d'un tableau de n positions vers la gauche
 * @param tab tableau d'entiers
 * @param n entier nombre de cases à décaler
 */
void decalerGaucheN (int[] tab, int n)
```

A titre d'exemple, le tableau `tab` contenant `{3, 10, 6, 20, 7}`, devient `{20, 7, 3, 10, 6}` après l'appel `decalerGaucheN(tab, 3)` .

méthodes et fonctions (suite)

Nom :

Prénom :

Groupe :

Question 3 (1 point)

La méthode `testDecalerGaucheN()` teste `decalerGaucheN()`. Donner en français la liste des tests à effectuer sans écrire la méthode `testDecalerGaucheN()`.

Question 4 (3 points)

Ecrire la méthode :

```
/**
 * cherche l'indice de la première occurrence d'une valeur dans un tableau
 * @param tab tableau d'entiers
 * @param v    valeur à chercher
 * @return l'indice de la première valeur v dans tab si v est dans tab, -1 sinon
 */
int indiceTab (int[] tab, int v)
```

méthodes et fonctions (suite)

Nom :

Prénom :

Groupe :

Question 5 (1 point)

Ecrire la méthode :

```
/**
 * décale les valeurs d'un tableau de manière à ramener la valeur cherchée à l'indice 0
 * Si la valeur n'est pas présente, le tableau n'est pas modifié
 * @param tab tableau d'entiers
 * @param v    valeur à chercher
 */
void decaleValeur (int[] tab, int v)
```

A titre d'exemple, le tableau `tab` contenant `{3, 10, 6, 20, 7}` devient `{20, 7, 3, 10, 6}` après l'appel `decaleValeur(tab, 20)`.