Université d'Ottawa
Faculté de génie

École de science informatique
et de génie électrique

uOttawa

University of Ottawa
Faculty of Engineering

School of Electrical Engineering
and Computer Science

Introduction à l'informatique II (ITI1521)
Examen de mi-session

Guy-Vincent Jourdan, Mehrdad Sabetzadeh et Marcel Turcotte

Mars 2020, durée: 2 heures

Identification

Nom de famille : _____ Prénom(s) : _____

Étudiant : _____ # Lab : _____ Signature : _____

Instructions

1. Examen à livres fermés.
2. L'utilisation de calculatrices, d'appareils électroniques ou tout autre dispositif de communication est interdit.
   (a) Tout appareil doit être éteint et rangé.
   (b) Toute personne qui refuse de se conformer à ces règles pourrait être accusée de fraude scolaire.
3. Répondez sur ce questionnaire.
   (a) Utilisez le verso des pages si nécessaire.
   (b) Aucune page supplémentaire n'est permise.
4. Écrivez vos commentaires et hypothèses afin d'obtenir des points partiels.
5. Écrivez lisiblement, puisque votre note en dépend. N'utilisez pas de stylo rouge.
6. Ne retirez aucune page ou l'agrafe de cet examen.
7. Attendez l'annonce de début de l'examen.

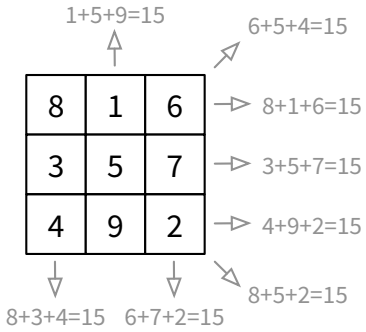
Barème

Question	Maximum	Résultat
1	20	
2	10	
3	15	
Total	**45**	

Question 1 (20 points)

Pour cette question, nous développons une représentation pour un carré magique (**Magic Square**). Dans sa forme la plus élémentaire, un **carré magique** est une grille $n \times n$ constituée des nombres $1, 2, \ldots, n^2$, tel que chaque ligne et chaque colonne ainsi que les deux diagonales principales ont toutes la même somme, appelée la **constante magique** (**Magic Constant**) M, où :

$$M = \frac{n(n^2+1)}{2}$$



Dans l'exemple de droite, $n = 3$ et $M = 15$.

Pour cette question, vous devez compléter l'implémentation de la classe **MagicSquare** à la page 4. Tout comme pour le devoir 1, **MagicSquare** utilise un tableau à une dimension pour sauvegarder les valeurs de la grille.

A. (4 points) Dans la déclaration de classe à la page 4, déclarez trois variables d'instance :

(a) Une variable référence utilisée pour désigner un tableau à une dimension pour des valeurs de type **int**. Le tableau désigné sera utilisé pour sauvegarder les valeurs de la grille.

(b) Une variable pour mémoriser la taille de la grille, où la taille représente à la fois le nombre de lignes et le nombre de colonnes du carré magique (3 dans l'exemple ci-dessus).

(c) Une variable pour mémoriser la constante magique, telle que définie ci-dessus (15 dans cet exemple).

B. (8 points) Dans la déclaration de classe de la page 4, implémentez le constructeur. Assurez-vous que votre implémentation est conforme aux directives suivantes :

(a) Le constructeur sauvegarde les valeurs de ce carré magique spécifiées par le paramètre. Comme l'illustre le programme de test de la page suivante, un objet de la classe **MagicSquare** n'est pas affecté par les modifications ultérieures du tableau de valeurs passé en paramètre au constructeur. Ici, vous pouvez supposer que la valeur du paramètre ne sera pas **null** et que le tableau donné forme un carré. En d'autres termes, la longueur du tableau est une puissance de 2 pour un entier $n \geq 3$.

(b) Le constructeur calcule et sauvegarde la taille du carré magique, où la taille représente à la fois le nombre de lignes et le nombre de colonnes.

(c) Enfin, vous devez calculer et sauvegarder la valeur de la constante magique.

C. (4 points) Implémenter les trois (3) méthodes d'accès suivantes.

(a) Implémenter la méthode d'instance **getSize()** qui retourne la taille de **ce carré magique** (**MagicSquare**), où la taille est le nombre de lignes et de colonnes.

(b) Implémenter la méthode d'instance **getConstant()** qui retourne la valeur de la constante magique, telle que définie ci-dessus.

(c) Enfin, implémentez la méthode **getValue(int row, int column)**. La méthode retourne la valeur sauvegardée à l'emplacement **row** et **column** de la grille. La plus petite valeur valide pour **row** et **column** est 0 (c'est le premier élément pour cette rangée ou cette colonne). Vous pouvez supposer que les valeurs de **row** et **column** seront valides. Attention, il s'agit d'une coordonnée en deux dimensions alors que l'information est sauvegardée dans un tableau à une dimension.

D. (4 points) Implémenter la méthode d'aide **isMagicDiagonals()**. Cette méthode retourne **true** si et seulement si les deux sommes, la somme des éléments sur la diagonale principale et la somme des éléments sur l'anti-diagonale principale, sont égales à la constante magique. La méthode est appelée par la méthode **isMagic()**. Vous n'êtes pas tenu d'implémenter les deux autres méthodes appelées par **isMagic()**. Plus précisément, vous n'avez pas à implémenter **isMagicRows()** et **isMagicColumns()**.

Le programme ci-dessous vous montre l'utilisation et le comportement prévus pour la classe **MagicSquare**.

```
MagicSquare s1, s2;

int[] values = new int[] {1,8,6,3,5,7,4,9,2};

s1 = new MagicSquare(values);

values[0] = 8;
values[1] = 1;

s2 = new MagicSquare(values);

System.out.println(s1);
System.out.println(s1.getSize());
System.out.println(s1.getConstant());
System.out.println(s1.isMagic());

System.out.println();

System.out.println(s2);
System.out.println(s2.getSize());
System.out.println(s2.getConstant());
System.out.println(s2.isMagic());
```

L'exécution du programme ci-dessus affiche ce qui suit sur la console :

```
 1  8  6
 3  5  7
 4  9  2
3
15
false

 8  1  6
 3  5  7
 4  9  2
3
15
true
```

Rappels : En Java, **Math.sqrt(double a)** peut être utilisé pour calculer la racine carrée d'un nombre, tandis que **Math.pow(double a, double)** retourne la valeur du premier argument élevé à la puissance du second. Vous pouvez forcer les types de **double** à **int**, si nécessaire.

```
public class MagicSquare {

    // Partie A. Variables d'instance









    // Partie B. Constructeur

    public MagicSquare(int[] values) {









    }
```

// *Partie C. Méthodes d'accès*

```
    // Partie D. isDiagonals()

    public boolean isMagic() {
        return isMagicRows() && isMagicColumns() && isMagicDiagonals();
    }

    // Le code source des autres méthodes, y compris isMagicRows() et isMagicColumns(), est caché.

    private boolean isMagicDiagonals() {









    }
}
```

Question 2 (10 points)

Vous devez compléter l'implémentation de la méthode de classe **reverse(String[] a)**. À la suite d'un appel à cette méthode, les valeurs du tableau désigné par le paramètre **a** sont dans l'ordre inverse, comme le montre l'exemple ci-dessous. Votre implémentation **doit** utiliser une pile pour faire le travail de renverser l'ordre des éléments. Ici,

- **Stack** est une interface, avec les méthodes habituelles : **push**, **pop** et **isEmpty**;
- **StackImplementation** est une classe qui réalise l'implémentation de l'interface **Stack**. Cette implémentation peut sauvegarder un nombre arbitrairement grand d'éléments. Vous n'avez pas à implémenter cette classe.

Le petit programme Java ci-dessous illustre le comportement attendu.

```
String[] alphabet;

alphabet = new String[] {"alpha", "bravo", "charlie", "delta", "echo"};

System.out.println(Arrays.toString(alphabet));

StringUtils.reverse(alphabet);

System.out.println(Arrays.toString(alphabet));
```

Son exécution produit le résultat suivant :

```
[alpha, bravo, charlie, delta, echo]
[echo, delta, charlie, bravo, alpha]
```

Donnez votre solution dans la boîte prévue à cet effet à la page suivante.

```
public class StringUtils {

    public _____ _____ reverse(String[] a) {

        Stack<_____> s;

    }
}
```

Question 3 (15 points)

Pour cette question, vous devez implémenter une classe appelée **Token**. Un objet de la classe **Token** peut soit sauvegarder la référence d'un objet de la classe **String** ou une valeur du type primitif **int**. Par conséquent, la classe possède deux constructeurs, un pour chaque type. Voici une spécification détaillée de son implémentation.

A. (1 point) Écrivez la déclaration de classe pour **Token**.

B. (3 points) Déclarer les variables d'instance nécessaires pour qu'un objet de la classe **Token** puisse sauvegarder la référence d'un objet de la classe **String**. Il peut sauvegarder une valeur du type primitif **int** (la valeur doit être sauvegardée sous forme de **int**). Enfin, l'objet doit savoir s'il sauvegarde la référence d'un objet de la classe **String** ou une valeur de type **int**.

C. (2 points) Donnez l'implémentation du constructeur **Token(String value)**. Le constructeur sauvegarde la référence de l'objet désigné par le paramètre **valeur**. L'objet doit se souvenir qu'il sauvegarde une valeur de type **String**.

D. (2 points) Donnez l'implémentation du constructeur **Token(int value)**. Le constructeur sauvegarde la valeur du paramètre **valeur**. L'objet doit se souvenir qu'il sauvegarde une valeur de type **int**.

E. (1 point) Implémenter la méthode d'instance **isString()** qui retourne **true** si cet objet sauvegarde une valeur de type **String**, et **false** sinon.

F. (2 points) Implémenter la méthode d'instance **toString()** qui retourne une représentation sous forme d'une chaîne de caractères (**String**) de cet objet. Voir ci-dessous pour des exemples.

G. (4 points) Implémenter la méthode de classe **equals(Token a, Token b)** qui retourne **true** si le contenu des objets désignés par les paramètres **a** et **b** sont logiquement équivalents (ont le même contenu) et **false** sinon.

```
Token a, b, c;

a = new Token("alpha");
b = new Token("alpha");
c = new Token(42);

System.out.println(a);
System.out.println(b);
System.out.println(c);

System.out.println();

System.out.println("a.isString() is " + a.isString());
System.out.println("c.isString() is " + c.isString());

System.out.println();

System.out.println("Token.equals(a,b) is " + Token.equals(a,b));
System.out.println("Token.equals(b,c) is " + Token.equals(b,c));
System.out.println("Token.equals(c,null) is " + Token.equals(c,null));
```

```
Token: alpha
Token: alpha
```

```
Token: 42

a.isString() is true
c.isString() is false

Token.equals(a,b) is true
Token.equals(b,c) is false
Token.equals(c,null) is false
```

Donnez l'implémentation de la classe **Token** dans l'espace ci-dessous.