Introduction à l'informatique II (CSI 1501)
Examen Intra

Professeur: Marcel Turcotte

Février 2004, durée: 2 heures

Identification

Nom, prénom: _____

Numéro d'étudiant: _____ Signature: _____

Consignes

1. Livres fermés;
2. Sans calculatrice ou toute autre forme d'aide;
3. Répondez sur ce questionnaire, utilisez le verso des pages si nécessaire, mais vous ne pouvez remettre aucune page additionnelle;
4. Écrivez lisiblement, votre note en dépend;
5. Commentez vos raisonnements;
6. Ne retirez pas l'agrafe;
7. L'appendice A contient une liste de méthodes des classes `String` et `Character`.

Barème

Question	Maximum	Résultat
1	18	
2	36	
3	12	
4	12	
5	16	
6	6	
Total	**100**	

Question 1 (18 points)

Cette question porte sur une classe permettant de représenter un nombre rationnel (une fraction). Pour chacune des sous-questions suivantes indiquez clairement tous les changements nécessaires à l'implémentation se trouvant à la page 4.

1. (8 points) Effectuez tous les changements nécessaires à la classe `Rational` (voir page 4) afin qu'elle réalise l'interface `Comparable`.

```
public interface Comparable {
    // Compare cet objet et celui passé en paramètre afin d'en
    // déterminer l'ordre.  Retourne une valeur négative, zéro
    // ou positive selon que cet object est plus petit, égal ou plus
    // grand que l'objet specifié.

    public int compareTo(Object o);
}
```

2. (6 points) Implémentez la méthode d'instance `public boolean equals(Object o)` retournant `true` si cette instance et celle passée en paramètre représentent la même fraction (équivalence de contenu).

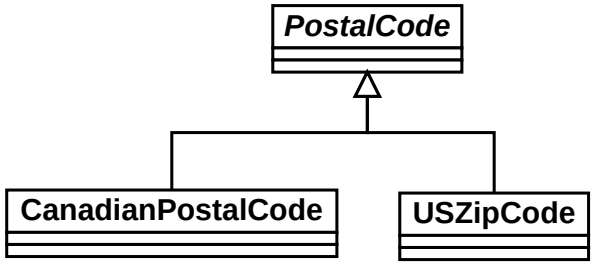
3. (4 points) Ajoutez une méthode `public String toString()` retournant une chaîne de caractères composée du numérateur, suivi du symbole "/", suivi du dénominateur. Si le dénominateur est 1 alors la méthode retourne une chaîne composée du numérateur seulement.

```
public class Rational {
    // variables d'instance
    private int numerator;
    private int denominator;

    // constructeur
    public Rational(int numerator) {
        this(numerator, 1);
    }
    public Rational(int numerator, int denominator) {
        if (denominator < 0) { // signe assigné au numérateur
            denominator = -1 * denominator;
            numerator = -1 * numerator;
        }
        this.numerator = numerator;
        this.denominator = denominator;
        reduce();
    }
    // méthode d'instance
    public Rational plus(Rational other) {
        int newDenominator = denominator * other.denominator;
        int newNumerator = numerator * other.denominator;
        int newOtherNumerator = other.numerator * denominator;
        int sum = newNumerator + newOtherNumerator;
        return new Rational(sum, newDenominator);
    }
    // calcul de la forme réduite
    private void reduce() {
        if (numerator == 0) {
            denominator = 1;
        } else {
            int common = gcd(Math.abs(numerator), denominator);
            numerator = numerator/common;
            denominator = denominator/common;
        }
    }
    // Algorithme d'Euclid calculant le plus grand common diviseur
    private int gcd(int a, int b) {
        while (a != b)
            if (a > b) {
                a = a - b;
            } else {
                b = b - a;
            }
        return a;
    }
}
```

Question 2 (36 points)

Le diagramme UML ci-bas présente une hiérarchie de classes afin de représenter des codes postaux de divers pays.



Sachant que,

- Tous les codes postaux ont une méthode `getCode` retournant le code (de type `String`) représenté par cette instance;
- Tous les codes postaux ont une méthode `isValid` retournant `true` si le code de cette instance est valide, et `false` sinon;
- Un code postal canadien est valide si les positions 0, 2 et 5 sont occupées par des lettres, les positions 1, 4 et 6 sont des chiffres, et la position 3 est un caractère blanc;
- Un code postal américain (Zip code) valide est consistitué de deux lettres, suivies d'un espace blanc, suivi de 5 chiffres.

1. (30 points) Concevoir une implémentation pour les classes `PostalCode`, `CanadianPostalCode` et `USZipCode`. Assurez-vous d'y include les variables d'instance, ainsi que les constructeurs. L'appendice A présente un résumé des méthodes des classes `String` et `Character`.

Question 2 (suite)

Question 2 (suite)

2. Déclarez un tableau, nommé `codes`, pouvant contenir 100 codes postaux (3 points).

3. Sachant qu'exactement `n` codes postaux se trouvent dans la partie gauche du tableau, écrivez une boucle `for` afin de compter le nombre de codes valides (3 points).

Question 3 (12 points)

Pour la classe `ArrayStack` ci-bas (semblable à celle du devoir #4), écrivez une méthode d'instance, `decreaseSize`, afin de décroître la taille du tableau d'`Objects`, `elems`, par un facteur `GROWTH_FACTOR`. La taille du tableau est diminuée aussitôt que possible, sans perdre un élément. La taille minimum du tableau est `DEFAULT_CAPACITY`.

```
public class ArrayStack {

    // constantes
    public static final int DEFAULT_CAPACITY = 10;
    public static final int GROWTH_FACTOR = 2;

    // variables d'instances
    private Object[] elems; // contient les éléments de cette pile
    private int top = -1;   // désigne l'élément du dessus
                            // ou -1 si cette pile est vide
    // constructeur
    public ArrayStack() {
        elems = new Object[DEFAULT_CAPACITY];
    }

    // retourne true si cette pile est vide
    public boolean isEmpty() {
        return top == -1;
    }

    // retire et retourne l'élément du dessus
    public Object pop() {
        // pre-conditions: ! isEmpty()

        Object saved = elems[top];
        elems[top--] = null;

        decreaseSize();

        return saved;
    }
```

Question 3 (suite)

```
 private void decreaseSize() {
```

```
    } // Fin de la méthode decreaseSize
} // Fin de la classe ArrayStack
```

Question 4 (12 points)

Complétez l'implémentation des méthodes d'instance `size()` et `swap()` pour la classe `LinkedStack` ci-bas. La méthode `size()` retourne le nombre d'éléments se trouvant dans la pile. La méthode `swap` inverse l'ordre des deux premiers éléments, c'est-à-dire que le premier élément devient le deuxième et le deuxième devient le premier; la méthode doit échanger l'ordre des éléments et non leurs valeurs. Vous ne pouvez utiliser les méthodes `push` et `pop`, vous devez manipuler les liens (références) directement. La méthode `swap` retourne `false` si la pile contient moins de deux éléments.

```
public class LinkedStack {

    // des objets de la classe Elem servent à sauvegarder les éléments
    // de cette pile

    private static class Elem {
        private Object value;
        private Elem next;
        Elem(Object value, Elem next) {
            this.value = value;
            this.next = next;
        }
    }

    // variables d'instance

    private Elem topElem;

    public int size() {

        Elem _______ = _______;

        int count = 0;

        while (_________________) {

            count++;

            _________________;
        }

        return count;
    }
```

Question 4 (suite)

```
 public boolean swap() {

     // pre-condition(s)

     if (_______________________________________) {
         return false;
     }

     Elem first = _________________;

     Elem second = _________________;

     first.next = _________________;

     second.next = _________________;

     topElem = _________________;

     return true;
 }

 // les autres méthodes d'instance, telles que push et pop,
 // seraient insérées ici; mais elle ne peuvent être
 // utilisées afin de répondre à cette question.

} // Fin de la classe LinkedStack
```

Question 5 (16 points)

Vous devez écrire une méthode de classe (static) qui retourne une copie de la pile passée en paramètre. Pour cette question, il y a une interface nommée `IntStack` et son implémentation, une classe nommée `MysteryIntStack`. Les éléments de la pile sont des entiers (`int`).

```
public interface IntStack {
    // retourne true si cette pile est vide
    public abstract boolean isEmpty();
    // ajoute un élement sur le dessus de cette pile
    public abstract void push(int value);
    // retire et retourne l'élément du dessus
    public abstract int pop();
}
```

La classe `MysteryIntStack` réalise l'interface `IntStack`. Le constructeur (`public MysteryIntStack()`) crée une pile vide. L'implémentation peut contenir un nombre arbitrairement grand d'éléments. Aucun autre détail d'implémentation n'est connu; en particulier, vous ne savez pas s'il s'agit d'une implémentation à l'aide d'une liste chaînée ou d'un tableau.

Pour la classe `Utils` ci-bas, concevez une méthode de classe (static) qui retourne une **nouvelle** pile contenant les mêmes éléments, dans le même ordre, que celle désignée par le paramètre `in`. De plus, la pile désignée par `in` doit demeurer inchangée (i.e. après un appel de méthode, `in` doit contenir les mêmes éléments, dans le même ordre, qu'avant l'appel).

```
public class Utils {
    public static IntStack copy(IntStack in) {
```

Question 5 (suite)

```
    } // Fin de la méthode copy
} // Fin de la classe Utils
```

Question 6 (6 points)

1. (2 points) Donnez l'expression postfixe (RPN) correspondant à l'expression infixe suivante.

$$(6 - 8 / 2) * (4 * 3 - (5 + 2))$$

2. (4 points) Évaluez l'expression postfixe (RPN) suivante en utilisant l'algorithme vu en classe, et dans vos notes de cours. Donnez le contenu de la pile **avant** et **après** l'évaluation de chaque sous-expression.

$$12 \;\; 6 \;\; - \;\; 33 \;\; 3 \;\; / \;\; +$$

A Appendix

La classe `String` contient les méthodes suivantes.

char charAt(int pos): retourne le caractère se trouvant à la position spécifiée par l'index `pos`;

int length(): retourne la longueur de cette chaîne.

La classe `Character` contient les méthodes suivantes.

static boolean isDigit(char ch): détermine si le caractère passé en paramètre est un nombre;

static boolean isLetter(char ch): détermine si le caractère passé en paramètre est une lettre;

static boolean isWhitespace(char ch): détermine si le caractère passé en paramètre est un espace blanc.

(page blanche)