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 MI-SESSION

Instructeurs: Opeyemi Adesina, Sherif Aly, Guy-Vincent Jourdan et Marcel Turcotte

Mars 2017, durée: 2 heures

Identification

Nom de famille : _____ Prénom(s) : _____

Étudiant : _____ 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.
6. Ne retirez pas l'agrafe du livret d'examen.
7. Attendez l'annonce de début de l'examen.

Barème

Question	Maximum	Résultat
1	25	
2	15	
3	30	
Total	**70**	

Question 1 : Notions de base en Java [25 points]

Vous devez concevoir 3 classes. Premièrement, vous devez créer une classe **Point** afin de représenter un point dans un plan à deux dimensions. Les objets de cette classe ont deux variables d'**instance**, **x** et **y**, afin de mémoriser les coordonnées du point. Le **constructeur** de la classe reçoit ces éléments d'information en paramètres.

Dans l'espace ci-dessous, donnez l'implémentation de la classe **Point**, incluant un **constructeur**, et toutes les méthodes d'accès nécessaires (**getters** et **setters**), ainsi que l'implémentation d'une méthode **toString**.

Créez une classe nommée **ColoredPoint** qui sera une version spécialisée de la classe **Point**. En plus des coordonnées, une instance de la classe **ColoredPoint** possède une variable d'instance **c** qui représente la couleur de ce point (**c** est une référence vers un objet de la classe **Color**, définie ailleurs dans le programme). Le **constructeur** de la classe reçoit en paramètre les coordonnées et la couleur.

Dans l'espace ci-dessous, donnez l'implémentation de la classe **ColoredPoint**, incluant un **constructeur**, toutes les méthodes d'accès nécessaires (**getters** et **setters**), ainsi que l'implémentation d'une méthode **toString**.

Finalement, créez une classe **BoundingBox** (zone de délimitation). Son constructeur reçoit en paramètre la référence d'un tableau contenant un mélange d'objets **Point** et **ColoredPoint**. Il calcule la zone de délimitation constituée des points en haut à gauche et en bas à droite du plus petit rectangle contenant tous les points du tableau.

Si le tableau ne contient qu'un seul élément alors la zone de délimitation est réduite à ce point. S'il n'y a pas d'élément ou si la référence est **null** alors la zone de délimitation est le point (0,0).

Dans l'espace ci-dessous, donnez l'implémentation de la classe **BoundingBox**, incluant un **constructeur** (qui devra calculer la zone de délimitation), des méthodes d'accès pour les points en haut à gauche et en bas à droite, ainsi que l'implémentation d'une méthode **toString**.

Question 2 : Types paramétrés [15 points]

Pour cette question, un index (un nombre) est associé à un élément d'information. Vous devrez fournir deux solutions pour ce problème, sans et avec les types paramétrés («*generics*»).

Dans l'espace ci-dessous, donnez l'implémentation de la classe **Indexed**. Un objet de la classe **Indexed** a deux variables d'instance, **index** de type **int** et **value** de type **String**. Donnez l'implémentation d'un constructeur ayant deux paramètres qui servent pour l'initialisation des variables d'instance. Donnez les méthodes d'accès en lecture seulement (**getters**) pour les variables d'instance. Donnez l'implémentation d'une méthode d'instance **isEqual** qui reçoit en paramètre la référence d'un autre objet **Indexed**. La méthode retourne **true** si et seulement si cet objet et celui dont la référence est passée en paramètre ont le même contenu. Cette implémentation n'utilise pas les types paramétrés.

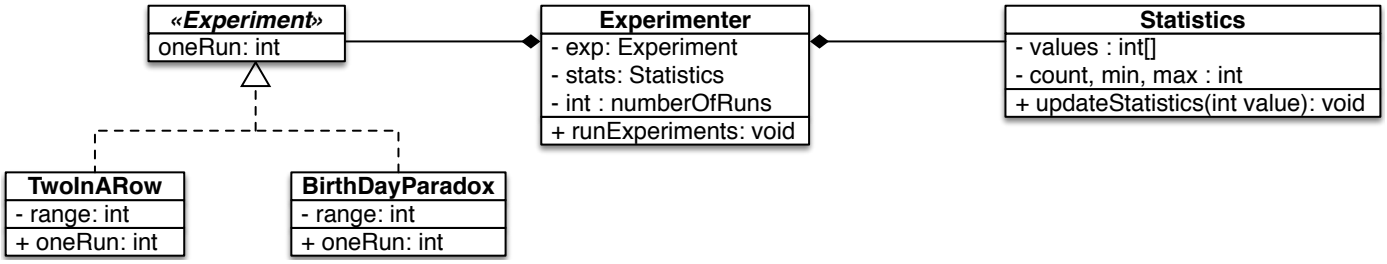
Dans l'espace ci-dessous, déclarez une variable référence, créez un objet de la classe **Indexed** pour sauvegarder les valeurs 0 et «Ottawa». Finalement, sauvegardez la référence de l'objet dans la variable.

Dans l'espace ci-dessous, donnez l'implémentation d'une classe **Indexed** ayant un paramètre de type, **T**. La classe **Indexed** déclare deux variables d'instance, **index** de type **int** et **value** de type **T**. Donnez l'implémentation d'un constructeur ayant deux paramètres qui servent pour l'initialisation des variables d'instance. Donnez les méthodes d'accès en lecture seulement (**getters**) pour les variables d'instance. Donnez l'implémentation d'une méthode d'instance **isEqual** qui reçoit en paramètre la référence d'un autre objet **Indexed**. La méthode retourne **true** si et seulement si cet objet et celui dont la référence est passée en paramètre ont le même contenu.

Dans l'espace ci-dessous, déclarez une variable référence, créez un objet de la classe **Indexed** pour sauvegarder les valeurs 0 et «Ottawa». Finalement, sauvegardez la référence de l'objet dans la variable.

Question 3 : Programmation orientée objet et interface [30 points]

Cette question met en oeuvre les concepts de la programmation orientée objet et d'interface afin de généraliser la solution du devoir 2. Le diagramme UML ci-dessous présente les relations entre les classes et l'interface.



Les énoncés Java ci-dessous fournissent un exemple de l'utilisation de ces objets.

```
TwoInARow t;
Experimenter e;

t = new TwoInARow(10);
e = new Experimenter(t, 100);

e.runExperiments();
```

L'exécution du programme ci-dessus produira le résultat qui suit sur la console.

```
We have run 100 experiments:
 the minimum was 3
 the maximum was 41
 the mean was 10.62
 the standard deviation was 7.86
```

A. Donnez l'implémentation de l'interface **Experiment**. L'interface déclare une méthode **oneRun** sans paramètre dont le type de la valeur de retour est **int**.

B. Implémentez la classe **TwoInARow**.

- La classe **TwoInARow** réalise l'interface **Experiment**.
- Son constructeur a un paramètre pour spécifier l'intervalle (**range**) de valeurs pour une expérience.
- La méthode **oneRun** génère des nombres aléatoires dans l'intervalle spécifié de valeurs. Elle s'arrête lorsque deux valeurs **consécutives** sont égales. La méthode retourne le nombre de tentatives qui ont été nécessaires pour trouver deux valeurs consécutives égales. Dans l'exemple ci-dessous, la méthode **oneRun** a fait 8 tentatives avant de produire deux valeurs consécutives identiques. La valeur retournée est 8.

```
4, 7, 2, 9, 7, 5, 1, 1
```

```
import java.util.Random;
```

Indice : un objet de la classe **Random** possède une méthode **nextInt** qui retourne un nombre aléatoire dans l'intervalle 0 (inclu) à **n** (exclu), où **n** est le paramètre de la méthode.

C. Implémentez la classe **Experimenter**.

- Le constructeur de la classe **Experimenter** a deux paramètres. Le premier paramètre est une référence vers un objet dont la classe réalise l'interface **Experiment**. Le second paramètre spécifie le nombre d'expériences a effectuer.
- La méthode **runExperiments** exécute l'expérience le nombre de fois spécifié, accumule des résultats dans un objet de la classe **Statistics** (voir page suivante), et affiche les statistiques à la fin de l'exécution.

D. Sur la page qui suit, vous devez modifier la classe **Statistics** pour utiliser la technique du tableau dynamique présentée en classe. Pour l'implémentation de droite, ne réécrivez que les parties qui changent. Nous supposons que les autres parties demeurent inchangées.

```java
public class Statistics {

    private int[] values;
    private int count, min, max;

    public Statistics(int numberOfRuns) {
        values = new int[numberOfRuns];
        count = 0;
    }

    public void updateStatistics(int value) {
        if (count == 0) {
            min = max = value;
        }
        min = Math.min(min, value);
        max = Math.max(max, value);
        values[count] = value;
        count = count + 1;
    }

    public int getMin() { return min; }

    public int getMax() { return max; }

    public double average() {
        double result = 0.0;
        for (int i = 0; i < count; i++) {
            result = result + values[i];
        }
        return result / count;
    }

    public double standardDeviation() {
        double mean = average();
        double squareSum = 0;
        for (int i = 0; i < count; i++) {
            squareSum += Math.pow(values[i] - mean, 2);
        }
        return Math.sqrt((squareSum) / count);
    }

}
```

```java
public class Statistics { // Seulement les changements



}
```