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

Professeur: Marcel Turcotte

Février 2003, durée: 2 heures

Identification

Nom de l'étudiant: _____

Numéro d'étudiant: _____

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. Ne retirez pas l'agrafe.

Barème

Question	Maximum	Résultat
1	38	
2	18	
3	18	
4	10	
5	16	
Total	**100**	

Question 1 (38 points)

(a) Concevez une interface nommée **Pair**. Une *paire* représente un groupe de deux *éléments*. L'interface doit contenir les méthodes suivantes:

- **Object getFirst();**
- **Object getSecond();**
- **void swap();**

(b) Vous devez créer une implémentation à l'aide d'une liste chaînée pour le *type de données abstrait* **Pair**. Cette classe, appelée **LinkedPair**, doit réalisée l'interface **Pair** et utiliser la classe **Node** ci-bas afin de contenir les éléments.

```
class Node {
    protected Object value;
    protected Node next;
    protected Node(Object value, Node next) {
        this.value = value;
        this.next = next;
    }
}
```

Variables

La classe **LinkedPair** doit posséder une variable d'instance, nommée **first**, afin de désigner le premier élément de la liste chaînée. Aucune autre variable n'est requise ou permise.

Constructeur

Créez un constructeur ayant deux paramètres, tous deux de type **Object**, dont les valeurs sont utilisées afin d'initialiser les valeurs du premier et second élément de la liste.

Méthodes

Écrivez toutes les méthodes nécessaires afin que la classe **LinkedPair** réalise l'interface **Pair**.

boolean equals(Object obj)

Il faut aussi redéfinir la méthode **equals(Object obj)** afin que la méthode retourne **true** *i)* si **obj** désigne un objet d'une classe réalisant l'interface **Pair** et *ii)* les deux structures contiennent des objets équivalents, dans le même ordre; et **false** (faux) sinon.

Indice: "**o instanceof I**" peut être utilisé afin de déterminer si la variable référence **o** désigne un objet d'une classe réalisant l'interface **I**.

Question 1b (suite)

(c) Concevez une implémentation à l'aide d'un tableau pour le *type de données abstrait* **Pair**. Cette classe, nommée **ArrayPair**, doit réaliser l'interface **Pair** et utiliser un tableau afin de contenir ses éléments.

Variables

La classe **ArrayPair** doit posséder une variable d'instance nommée **elements** afin de désigner un tableau de deux éléments. Aucune autre variable n'est nécessaire ou permise.

Constructeur

Créez un constructeur ayant deux paramètres, tous deux de type **Object**, que vous utiliserez afin d'initialiser les valeurs du premier et du second élément du tableau.

Méthodes

Écrivez toutes les méthodes nécessaires afin que la classe **ArrayPair** réalise l'interface **Pair**.

boolean equals(Object obj)

Il faut aussi redéfinir la méthode **equals(Object obj)** afin que la méthode retourne **true** *i)* si **obj** désigne un objet d'une classe réalisant l'interface **Pair** et *ii)* les deux structures contiennent des objets équivalents, dans le même ordre; et **false** (faux) sinon.

Question 1c (suite)

Question 1 (suite)

Les classes conçues pour les parties b et c devraient être telles que le programme suivant affiche “bravo”.

```
public class Test {
    public static void main(String[] args) {

        Pair a = new LinkedPair(new String("data"), new String("structure"));
        Pair b = new ArrayPair(new String("structure"), new String("data"));

        if (a.equals(b)) {
            System.out.println("wrong :-(");
        } else {
            a.swap();
            if (a.equals(b)) {
                System.out.println("bravo !");
            } else {
                System.out.println("wrong :-(");
            }
        }
    }
}
```

Question 2 (18 points)

(a) Vous devez concevoir un hiérachie de classes pour représenter des véhicules:

- Tous les véhicules ont un poids (de type double);
- Une voiture est spécialisation de la classe véhicule ayant un nombre de passagers (int);
- Un camion est un type spécialisé de véhicule pouvant transporter une certaine charge (double);
- Un camion peut charger et décharcher sont contenu (sa charge). Décharger un poids excédant le poids du chargement n'a aucun effet (et doit retourner la valeur false);
- Le poids d'un camion est le poids du véhicule plus sa charge;
- Tous les poids sont exprimés en tonne.

Question 2 (suite)

(b) Complétez la définition de la méthode **calculerFrais** pour la classe **Traversier** ci-bas. La méthode **calculerFrais** retourne le total des droits de passage pour tous les véhicules sur le traversier.

- le droit de passage pour une voiture est de $ 10 par tonne plus $ 5 par passager;
- le droit de passage pour un camion est de $100 par tonne.

```
class Traversier {
    private Vehicule[] vehicules;

    Traversier(Vehicule[] vehicules) {
        this.vehicules = vehicules;
    }

    double calculerFrais() {
        // completez la methode









    }
}
```

Indice: "**o instanceof I**" peut être utilisé afin de déterminer si la variable référence **o** désigne un objet d'une classe réalisant l'interface **I**.

Question 3 (18 points)

Pour la classe **SimpleList** ci-bas, complétez la méthode **remove(Object obj)** afin que la méthode retire l'occurrence la plus à gauche de **obj** de la liste. La méthode n'a aucun effet si **obj** est absent de la liste.

```
public class SimpleList {

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

    private Node first;

    public void remove(Object obj) {

        // cas special: liste vide
        if (                          )
            return;

        // cas special: obj est le premier element
        if (                          ) {




        } else {
            // cas general:









        }
    }
```

Question 4 (10 points)

Pour la classe **ArrayList** ci-bas, implémentez la méthode **toArray()**. La classe **ArrayList** utilise la technique du tableau de taille variable de sorte qu'un nombre illimité d'éléments puissent être sauvegardés. La méthode **toArray()** retourne un tableau *i)* dont la taille est la même que le nombre d'éléments se trouvant présentement dans la structure de données et *ii)* contient les mêmes éléments, dans le même ordre.

```
public class ArrayList {

    private Object[] elements;
    private int last;

    public ArrayList(int capacity) {
        elements = new Object[capacity];
        last = 0;
    }

    public int size() {
        return last;
    }

    private void increaseSize() {

        Object[] tmp = elements;
        elements = new Object[tmp.length * 3 / 2];

        for (int i=0; i < last; i++)
            elements[i] = tmp[i];
    }

    public void add(Object t) {
        if (last == elements.length)
            increaseSize();

        elements[last] = t;
        last++;
    }
```

```
    // complete the method toArray
    Object[] toArray() {

    }
} // end of ArrayList
```

Question 5 (16 points)

Étant donné les définitions suivantes:

```
public interface I {
    public void i();
}

public abstract class A implements I {
    private int a = 1;
    public int get () { return a; }
}

public class B  extends A {
    private int b = 2;
    public int get () { return b; }
    public int i() { return a + b; }
}
```

(a) Pour chacun des énoncés suivants, indiquez si l'énoncé est **valide** ou **non-valide**.

i)
```
I o = new B();
o.i();
```

ii)
```
I o = new I();
```

iii)
```
I o = new B();
o.get();
```

iv)
```
B o = new B();
```

(b) **Vrai** ou **faux**.

i) La valeur 2 sera affichée. **vrai** ou **faux**

```
B o = null;
System.out.println(o.get());
```

ii) La valeur 2 sera affichée. **vrai** ou **faux**

```
A o = new B();
System.out.println(o.get());
```

iii) La valeur 1 sera affichée. **vrai** ou **faux**

```
A o = new A();
System.out.println(o.get());
```

(c) Étant donné la méthode suivante:

```
public static void mystery(String s) {
    s = "NEW";
}
```

Quelle sera la valeur affichée à la suite des énoncés suivants:

```
String x = "OLD";
mystery(x);
System.out.println(x);
```

(page blanche)