UQÀM
Université du Québec à Montréal

INF1120 Programmation I

Automne 2004

Examen Final

18 décembre 2004

CONSIGNES

Solutionnaire

IDENTIFICATION

PRÉNOM: .....

NOM: .....

CODE PERMANENT: .....

SIGNATURE: .....

GROUPE: .....

PROFESSEUR: .....

#1	_____	/ 15
#2	_____	/ 15
#3	_____	/ 15
#4	_____	/ 15
#5	_____	/ 20
#6	_____	/ 20
TOTAL	_____	/ 100

COMMENTAIRES

Aide-mémoire pour l'examen final

Quelques méthodes publiques de la classe `String` :

```
char charAt(int indice)
Retourne le caractère à l'indice spécifié.

int indexOf(int ch)
Retourne l'indice à l'intérieur de cette chaîne de la première occurrence du caractère
ch.

int lastIndexOf(int ch)
Retourne l'indice à l'intérieur de cette chaîne de la dernière occurrence du caractère
ch.

int length()
Retourne la longueur de cette chaîne.

String replace(char oldChar,  char newChar)
Retourne une nouvelle chaîne résultant du remplacement de toutes les occurrences de
oldChar par newChar.

String substring(int beginIndex, int endIndex)
Retourne une nouvelle chaîne qui est une sous-chaîne de cette chaîne constituée des
caractères de chaîne de la position beginIndex incluse et finissant à la position
endIndex exclue.  La sous-chaîne contient donc endIndex - beginIndex caractères.

String trim()
Retourne une copie de la chaîne à laquelle on a enlevé les blancs au début et les
blancs à la fin.

static String valueOf(int i)
Retourne la chaîne correspondant l'entier i.
```

Calcul de taxes

Calculer la tps de 7 % sur un montant : montant * 0.07
Calculer un montant augmenté de la tps : montant * 1.07

On utilise le même principe pour calculer la tvq, mais le taux est de 7.5 %.

Le montant utilisé pour calculer la tvq est le montant original auquel on a ajouté la tps.

Pour obtenir le prix avec escompte, on peut appliquer cet escompte avant ou après avoir ajouté les taxes.

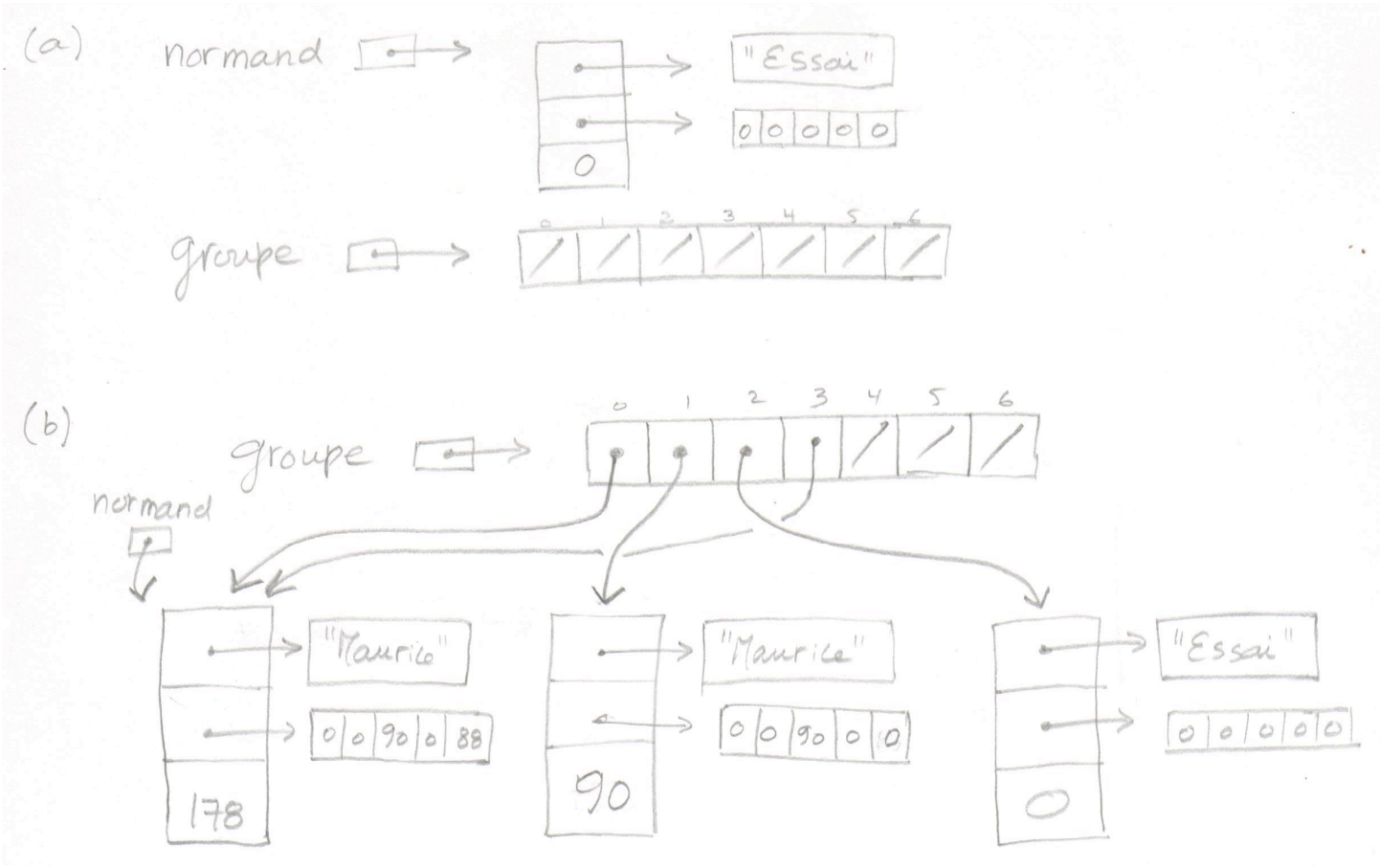
Question 1 (15 points)

Considérez les deux classes suivantes :

```
public class Etudiant {
    private String nomPrenom;
    private int[] notes;
    private int somme;

    public Etudiant(String ident) {
        nomPrenom = ident;
        notes = new int[5];
        somme = 0;
    }

    /* Ajoute la note à l'endroit indiqué dans le tableau notes si la note est entre 1 et
     * 100 inclusivement et si l'endroit est valide puis ajuste la somme en conséquence.
     * Si la note ou l'endroit n'est pas valide, aucune modification n'est faite. */

    public void placerNote(int note, int endroit) {
        if (endroit >= 0 && endroit < 5 && note > 0 && note <= 100) {
            somme = somme - notes[endroit];
            notes[endroit] = note;
            somme = somme + note;
        } // if
    } // placerNote

    /* Construit un nouvel objet de la classe Etudiant identique au présent objet
     * et retourne une référence vers ce nouvel objet. */

    public Etudiant copie() {
        Etudiant reponse = new Etudiant(nomPrenom);
        for (int i = 0; i < notes.length; i++) {
            reponse.notes[i] = notes[i];
        }
        reponse.somme = somme;
        return reponse;
    } // copie

} // Etudiant

public class Schema {

    public static void main (String[] params) {

        Etudiant normand = new Etudiant("Essai");
        Etudiant[] groupe = new Etudiant[7];

        //(a) Dessinez les contenus de normand et groupe rendu ici (utilisez la page suivante)

        groupe[2] = normand ;
        normand = new Etudiant("Maurice");
        groupe[0] = normand ;
        groupe[3] = groupe[0];
        normand.placerNote(90, 2);
        groupe[1] = normand.copie();
        groupe[3].placerNote(88, 4);

        //(b) Dessinez les contenus de normand et groupe rendu ici (utilisez la page suivante)

    } // main

} // Schema
```

suite page suivante

Exécutez à la main la méthode `main` de la classe `Schema` et dessinez `normand` et `groupe` aux deux endroits indiqués dans le code de la méthode `main` de la classe `Schema`. Vos dessins doivent inclure tout espace mémoire accessible à partir de `normand` et à partir de `groupe`.



Question 2 (15 points)

Écrivez une méthode de classe publique qui a comme paramètre une chaîne de caractères (dénotée `chaineEntree`) et retourne une chaîne contenant la première phrase de `chaineEntree` (sans les blancs précédant le début de la phrase). Une phrase commence par une lettre majuscule et se termine par un point (`'.'`), un point d'interrogation (`'?'`) ou un point d'exclamation (`'!'`). Si `chaineEntree` ne contient pas de lettre majuscule (c'est-à-dire de début de phrase) ou si la phrase ne se termine pas par un des symboles `'.'`, `'?'` ou `'!'`, votre méthode doit retourner une chaîne vide. Par exemple, si `chaineEntree` est la chaîne

```
"  au secours!  La terre est-elle plate?  Bien sur... ",
```

la méthode retournera la chaîne

```
"La terre est-elle plate?".
```

Voici l'en-tête de la méthode.

```
public static String extrairePhrase(String chaineEntree)
```

Votre méthode pourra invoquer d'autres méthodes, soit des méthodes de la classe `String`, soit des méthodes que vous écrirez vous-même.

```
public static boolean estMajusc ( char c ) {
    return c >= 'A' && c <= 'Z';
}

public static boolean estFin ( char c ) {
    return c == '.'  || c == '!'  || c == '?';
}

public static int indiceMajuscule ( String chaine ) {
    int i = 0;
    while ( i < chaine.length() && !estMajusc ( chaine.charAt ( i ) ) ) {
        i = i + 1;
    }
    return i;
} //

public static int indiceFin ( String chaine, int debut ) {
    while ( debut < chaine.length() && !estFin ( chaine.charAt ( debut ) ) ) {
        debut = debut + 1;
    }
    return debut;
} //

public static String extrairePhrase (String chaineEntree) {

    String reponse = "";
    int debut;
    int fin;

    debut = indiceMajuscule ( chaineEntree );
    fin = indiceFin ( chaineEntree, debut );

    if ( debut < chaineEntree.length() && fin < chaineEntree.length() ) {
        reponse = chaineEntree.substring ( debut, fin + 1 );
    }

    return reponse;
} //
```

Question 3 (15 points)

Dans le pays de Cocagne les numéros d'assurance sociale sont des numéros à neuf chiffres, c'est-à-dire des nombres compris entre 100 000 000 et 999 999 999. Le programme ci-dessous lit un numéro d'assurance sociale et l'affiche. La lecture elle-même est effectuée par la méthode `lireNodAssSoc`. Modifiez cette méthode de telle sorte qu'elle lève une exception (de classe `ExcNodAssSoc`) si le nombre entier saisi est plus petit que 100 000 000 ou plus grand que 999 999 999. Si l'exception est levée, elle doit se propager à la méthode principale (`main`) que vous devez aussi modifier. Celle-ci doit traiter cette exception en affichant un message approprié.

```
public class Test {

    public static int lireNodAssSoc() {
        int numero;
        numero = Clavier.lireInt();
        return numero;
    }

    public static void main(String[] args) {
        int numero;

        numero = lireNodAssSoc();
        System.out.println("Le numero d'assurance sociale "
                           + "est " + numero + ".");
    } // main
} // Test

class ExcNodAssSoc extends Exception {
}
```

```
public class TestSolution {

    public static int lireNodAssSoc() throws ExcNodAssSoc {
        int numero;
        numero = Clavier.lireInt();
        if (numero < 100000000 || numero > 999999999) {
            throw new ExcNodAssSoc();
        }
        return numero;
    }

    public static void main(String[] args) {
        int numero;

        try {
            numero = lireNodAssSoc();
            System.out.println("Le numero d'assurance sociale "
                               + "est " + numero + ".");
        } catch (ExcNodAssSoc e) {
            System.out.println("Le numero d'assurance sociale n'est pas valide.");
        }
    } // main
} // TestSolution
```

Question 4 (15 points)

Qu'affiche la méthode `main` ?

```
public class Tableau {

    public static void traitementA (int[] tabEntiers)  {
        for (int indice = 0; indice < tabEntiers.length; indice++) {
            if (indice <= 2) {
                tabEntiers[indice] = indice * 2;
            } else {
                tabEntiers[indice] = indice * 3;
            }
        } // for
    }// traitementA

    public static void traitementB (int[] tabEntiers)  {
        for (int indice = 1; indice < tabEntiers.length; indice++) {
            tabEntiers [indice - 1] = tabEntiers[indice];
        }
    } // traitementB

    public static void traitementC (int[] tabEntiers)  {
        for (int indice = 0; indice < tabEntiers.length; indice++) {
            if (tabEntiers[indice] % 2 == 0) {
                tabEntiers[indice] = 0;
            } else {
                tabEntiers[indice] = 1;
            }
        } // for
    } // traitementC

    public static void afficher (int[] tabEntiers)  {
        for (int indice = 0; indice < tabEntiers.length; indice++) {
            System.out.print (tabEntiers[indice] + " ");
        } // for
        System.out.println();
    } // afficher

    public static void main (String[] args) {
        int [] tabEntiers;
        tabEntiers = new int[5];
        traitementA (tabEntiers);
        afficher (tabEntiers);        // Afficher le contenu du tableau
        traitementB (tabEntiers);
        afficher (tabEntiers);        // Afficher le contenu du tableau
        traitementC (tabEntiers);
        afficher (tabEntiers);        // Afficher le contenu du tableau
    } // main

}// classe
```

Réponses

0 2 4 9 12

2 4 9 12 12

0 0 1 0 0

Question 5 (20 points)

On dispose de deux fichiers de texte. Le premier, portant le nom de "NomsPrenoms.txt", contient sur chaque ligne un nom et un prénom. Le deuxième, portant le nom de "Telephones.txt", contient sur chaque ligne un numéro de téléphone. Les données dans les deux fichiers ont déjà été validées et on est assuré que les deux fichiers sont présents. On doit produire un troisième fichier "Fusion.txt" qui fera la fusion des informations des deux fichiers de la façon suivante : la première ligne du troisième fichier sera constituée des caractères de la première ligne du fichier des noms et prénoms suivis d'un espace puis des caractères de la première ligne du fichier des numéros de téléphone. La deuxième ligne est construite de la même façon, On continue de la même manière jusqu'à la fin de l'un des (ou des deux) fichier(s). Normalement, il devrait y avoir le même nombre de lignes dans les deux fichiers "NomsPrenoms.txt" et "Telephones.txt". Cependant, il se peut que ça ne soit pas le cas et il faudra mentionner à la console lequel des deux fichiers a plus de lignes que l'autre. Voici un exemple donnant le contenu des deux fichiers d'entrée et celui du fichier de résultats, suivis du message qui sera affiché à la console dans ce cas.

"NomsPrenoms.txt"	"Telephones.txt"	"Fusion.txt"
Lapin Jeannot	1111111	Lapin Jeannot 1111111
Nguyen Abdel	123415	Nguyen Abdel 123415
Charest Jeanne	5149999999	Charest Jeanne 5149999999
Martin Paul	9873000	Martin Paul 9873000
McDonald Ronald		

```
Il y a plus de noms que de numéros de téléphone.
```

Complétez le code suivant.

```
import java.io.*;
public class QuestionFichiers {
    public static void main ( String[] params ) throws IOException {


        BufferedReader fNomsPrenoms = new BufferedReader (
                                      new FileReader ( "NomsPrenoms.txt" ) );
        BufferedReader fTelephone = new BufferedReader (
                                      new FileReader ( "Telephones.txt" ) );
        PrintWriter fFusion = new PrintWriter (
                                      new FileWriter ( "Fusion.txt" ) );

        sNomsPrenoms = fNomsPrenoms.readLine();
        sTelephone = fTelephone.readLine();

        while ( sNomsPrenoms != null && sTelephone != null ) {
            fFusion.println ( sNomsPrenoms + " " + sTelephone );
            sNomsPrenoms = fNomsPrenoms.readLine();
            sTelephone = fTelephone.readLine();
        }

        // sNomsPrenoms == null ou sTelephone == null
        if ( sNomsPrenoms != null ) {
            System.out.println ( "Il y a plus de noms que de numeros de telephone." );
        } else if ( sTelephone != null ) {
            System.out.println ( "Il y a plus de numeros de telephone que de noms." );
        }

        fNomsPrenoms.close();
        fTelephone.close();
        fFusion.close();

        System.out.println ( "Fin du programme" );
    } // main

} // QuestionFichiers
```

Question 6 (20 points)

Compléter la définition de la classe suivante.

```
/**
 * Classe servant à développer un catalogue d'objets promotionnels. À chaque
 * objet promotionnel du catalogue correspond une instance de cette classe et
 * vice-versa. Un objet promotionnel comporte un nom, un prix et une catégorie
 * ('A', 'B' ou 'C'). Il peut aussi être exempt de la TPS (dont le taux est de
 * 0.07), de la TVQ (dont le taux est de 0.075 et applicable au prix incluant
 * la TPS) ou des deux taxes.
 */
public class ObjetPromotionnel {

  // Variables de classe s'il y a lieu
```

```
    private static int nbObjetsConstruits = 0;

    private static final double tauxTPS = 0.07;

    private static final double tauxTVQ = 0.075;
```

```
  // Variables d'instances s'il y a lieu
```

```
    private String nom;

    private double prix;

    private char categorie;

    private boolean exemptTPS;

    private boolean exemptTVQ;
```

```
/**
 * Construit un nouvel objet promotionnel. Le nom doit comporter au moins
 * deux caractères et le prix doit être une valeur positive.
 * Si le nom comporte moins de deux caractères ou si la valeur du prix passée
 * en paramètre effectif est négative ou nulle, une exception de la classe
 * Exception comportant le message "Erreur sur le nom ou le prix" sera lancée
 * et aucun objet ne sera créé.
 * La catégorie de l'objet promotionnel est soit 'A', 'B' ou 'C'. Pour tout
 * autre caractère passé en paramètre, l'objet sera considéré de catégorie 'A'.
 * Les paramètres sansTPS et sansTVQ indiquent si l'objet est exempt de TPS
 * et de TVQ (true : exempt, false : non exempt)
 */
public ObjetPromotionnel(String leNom, double lePrix, char laCategorie,
                         boolean sansTPS, boolean sansTVQ) throws Exception {

    if ( (leNom.length() < 2) || (lePrix <= 0) ) {

        throw new Exception("Erreur sur le nom ou le prix");

    }

    nom = leNom;

    prix = lePrix;

    categorie = laCategorie;

    if ( (categorie != 'B') && (categorie != 'C') ) {

        categorie = 'A';

    }

    exemptTPS = sansTPS;

    exemptTVQ = sansTVQ;

    nbObjetsConstruits++;
}

/**
 * Construit un nouvel objet promotionnel. Le nom doit comporter au moins
 * deux caractères et le prix doit être une valeur positive.
 * Si le nom comporte moins de deux caractères ou si la valeur du prix passée
 * en paramètre effectif est négative ou nulle, une exception de la classe
 * Exception comportant le message "Erreur sur le nom ou le prix" sera lancée
 * et aucun objet ne sera créé.
 * L'objet construit est considéré comme étant de catégorie 'A' et n'est
 * exempt ni de TPS ni de TVQ.
 */
public ObjetPromotionnel(String leNom, double lePrix) throws Exception {

      this(leNom, lePrix, 'A', false, false);

}
```

```
/**
 * Retourne le nombre d'objets promotionnels dans le catalogue, ce nombre
 * correspond au nombre d'instances de la classe.
 */
public static int getNbObjetsConstruits()  {

      return nbObjetsConstruits;
}
/**
 * Retourne le prix au détail de l'objet promotionnel, incluant les taxes.
 * Ce prix tient compte du fait que certains objets sont exempts de TPS
 * et/ou de TVQ.
 */
public double getPrixAuDetailTTC() {

      double tps = prix * tauxTPS;

      double tvq = (prix + tps) * tauxTVQ;

      if (exemptTPS) {

          tps = 0.0;

      }

      if (exemptTVQ) {

          tvq = 0.0;

      }

      return prix + tps + tvq;
}
```

```
  /**
   * Retourne le prix de gros de l'objet promotionnel, incluant les taxes et
   * l'escompte accordé en fonction de la catégorie de l'objet promotionnel.
   * Le taux d'escompte est de 10% pour les objets appartenant à la catégorie
   * 'A', de 20% pour les objets appartenant à la catégorie 'B' et de 30% pour les
   * objets appartenant à la catégorie 'C'.
   * Ce prix tient compte du fait que certains objets sont exempts de TPS
   * et/ou de TVQ.
   */
  public double getPrixDeGrosTTC() {

        double tauxDEscompte = 0.0;

        switch (categorie) {

          case 'A' :

            tauxDEscompte = 0.10;

            break;

          case 'B' :

            tauxDEscompte = 0.20;

            break;

          case 'C' :

            tauxDEscompte = 0.30;

            break;

        }

        return getPrixAuDetailTTC() * (1.0 - tauxDEscompte);
  }
}
```