Examen de Programmation en Java (2 heures)

DEUG MIAS-MP 1ère année

UNSA, Juin 2001

AUCUN DOCUMENT AUTORISE. Soignez l'indentation et la précision du code. Nous lirons exactement ce que vous écrivez, à la faute de ponctuation, de majuscule ou d'orthographe près, pas ce que vous avez voulu dire ! Soyez simples et précis.

1 Une classe `Couple` modélisant $\mathbb{Z} \times \mathbb{R}$

Un programmeur s'intéresse, pour des besoins mathématiques, aux couples (n, x) formés d'un entier n et d'un réel x. Programmant en Java, il a l'idée d'écrire une classe `Couple` modélisant de tels objets, et munie :
- de variables privées.
- d'un constructeur `Couple(int n, double x)`.
- d'accesseurs `proj1` et `proj2` retournant respectivement la première ou la deuxième composante d'un couple instance de la classe.
- d'une fonction `toString(...)` comme toute classe d'objets affichables.

Question 1 — Ecrivez cette classe pour lui.

2 Etude d'une suite définie par récurrence

Question 2 — **Une suite comme fonction récursive**
On considère la suite réelle (u_n) définie ci-dessous, pour $n \geq 0$:

$$\begin{aligned} u_0 &= 2 \\ u_n &= \frac{1}{2}u_{n-1} + 3 \end{aligned}$$

Programmez une classe `Suite` contenant :
- une fonction récursive `u` prenant en entrée un indice n, et retournant le terme $u(n) = u_n$.
- une méthode `main(...)` faisant afficher sur une même ligne les 10 premiers termes u_0, u_1,..., u_9 de cette suite.

Question 3 — **Une version itérative**
Donnez une version itérative (avec une boucle `for` par exemple) de la fonction retournant le terme $u(n) = u_n$.

Question 4 — **La limite**
Le programmeur observe alors le résultat de l'exécution à l'écran et se trouve conduit à conjecturer que :

$$\lim_{n\to\infty} u_n = 6$$

ce qu'il prouve sans problème avec ses connaissances de MP1. Un ami physicien qui passait par là lui demande s'il est facile d'écrire un morceau de programme permettant de faire afficher le plus petit indice n tel que $|u_n - u_{n-1}| < 10^{-4}$. *Facile, lui dit-il, grosso modo :*

- *j'initialise n à 0.*
- *et j'utilise la fonction u(n) écrite plus haut : tant que la distance entre u(n-1) et u(n) est plus grande que* 10^{-4}*, j'augmente n de 1. Puis je fais afficher n...*

Facile. On vous demande de rédiger correctement le morceau de programme en suivant à la lettre sa description, sauf qu'il y a une petite étourderie que vous corrigerez au passage.

Question 5 — **Le temps c'est de l'argent**
Son ami physicien, pour qui le temps est chose importante, lui fait observer que le morceau de programme de la question précédente est correct, mais pas très efficace. Montrez en effet que le nombre d'opérations faites n'est pas proportionnel à n. A quelle fonction de n est-il précisément proportionnel ? Pourquoi ?...

Question 6 — **Optimisation...**
Vexé, il avoue la faiblesse de son programme et fait appel à vous. Pourriez-vous écrire le même programme affichant le plus petit nombre n tel que $|u_n - u_{n-1}| < 10^{-4}$, mais fonctionnant avec un nombre d'opérations proportionnel à n ?

Question 7 — **Conservation des résultats**
a) Souhaitant examiner à la loupe les termes de cette suite, il vous demande de construire un *vecteur* `v`, instance de la classe `Vector`, contenant les couples $(n, u(n))$ pour tous les n tels que $|u_n - 6| > 10^{-6}$. Vous utiliserez la classe `Couple` de la question 1.
b) Une fois ce vecteur construit, quelle est l'instruction qui ferait afficher le couple $(5, u(5))$ de ce vecteur ?

3 Interface graphique

On vous propose d'écrire une classe `Examen` réalisant une interface graphique destinée à visualiser l'expérience de statistique consistant à jeter un grand nombre de fois un dé à 6 faces équiprobables numérotées de 1 à 6. Au lancer numéro n, on obtient une valeur x_n (un entier de $[1, 6]$) que l'on fait afficher, et l'on calcule la moyenne des lancers $x_0, x_1, \ldots, x_n$ que l'on fait afficher également.

N.B. Pour calculer un entier aléatoire de $[1, 6]$*, vous utiliserez la méthode primitive :*

```
public static double Math.random()
```

qui retourne un réel aléatoire de $[0, 1[$*, et aucune des méthodes de la classe* `Numerik` *!*

Question 8 — **Moyenne des lancers**
Vers quelle valeur s'attend-on à voir converger la moyenne ?

Question 9 — **Mise à jour de la moyenne**
Utiliser un vecteur pour mémoriser tous les lancers et recalculer la moyenne chaque fois eût été prohibitif en occupation-mémoire et bien inutile. Décrivez simplement une méthode économique occupant un espace-mémoire constant (on attend une courte explication en français et non du code Java).

Question 10 — **Interface graphique**

Vous allez écrire l'interface graphique répondant aux prescriptions suivantes. La simulation se déroule dans une fenêtre non redimensionnable de fond jaune, de titre *Juin 2001*, munie d'une case de fermeture dont la pression provoque la terminaison du programme. Cette fenêtre comportera quatre composants :

- un bouton (variable `button1`) nommé *Lancer* en haut à gauche.
- un autre bouton (variable `button2`) nommé *Reset* en bas à gauche.
- un texte (variable `label1` initialisée à `"LANCER = "`) en haut à droite.
- un autre texte (variable `label2` initialisée à `"MOYENNE =??????????????"`) en bas à droite. Voir la copie d'écran ci-dessous.

Vous avez bien noté la disposition des quatre composants, n'est-ce pas ? Il y a plusieurs solutions pour y parvenir...

L'action associée au bouton *Lancer* est la suivante : un entier aléatoire de $[1, 6]$ est calculé, il est affiché dans le `label1`, par exemple sous la forme **LANCER = 4**. La moyenne de tous les lancers jusqu'à ce 4 inclus est recalculée et affichée dans le `label2` par exemple sous la forme **MOYENNE = 3.56398621845**.

L'action associée au bouton *Reset* est la suivante : on oublie tout, l'expérience peut recommencer comme si l'on avait démarré le programme.

Cette simulation est très simple. Profitez-en mais rédigez très lisiblement et avec une grande précision. Voir au dos l'Annexe contenant des constructeurs et méthodes utiles pour ce problème (mais vous pouvez en utiliser d'autres si vous les connaissez bien)...

Annexe : aide-mémoire pour la section 3

`package java.awt`

```
public class Button
   public Button(String title)     // le constructeur
   public void addActionListener(java.awt.event.ActionListener al)

public class Frame
   public Frame(String title)      // le constructeur
   public void add(Component comp)
   public void addWindowListener(java.awt.event.WindowListener wl)
   public void pack()
   public void setBackground(Color col)
   public void setLayout(LayoutManager mgr)
   public void setResizable(boolean resizable)
   public void show()

public class Label
   public Label(String str)        // le constructeur
   public void setText(String str)

public class Panel
   public Panel()                  // le constructeur
   public void setLayout(LayoutManager mgr)

public class FlowLayout
   public FlowLayout()             // le constructeur

public class BorderLayout
   public BorderLayout()           // le constructeur

public class GridLayout
   public GridLayout(int rows, int cols)  // le constructeur
```

`package java.awt.event`

```
public abstract class WindowAdapter implements WindowListener
   public void windowClosing(WindowEvent e)

public abstract interface ActionListener
   public abstract void actionPerformed(ActionEvent e)
```

`package java.lang` *(chargé automatiquement)*

```
public double Math.random()
```