Université Paris - Diderot
Concepts informatiques (CI2)
CI2 — Partiel du 19 mars 2008 — Durée : 2 heures
Documents, calculettes, ordinateurs et téléphones non autorisés

---

Exercice 1. *Passage de paramètres*

On considère tout d'abord les classes `C1` et `C2` définies de la manière suivante :

```
class C1 {
    int val;
    C1(int n){val = n;}
}
class C2 {
  int val; C1 c1; int[] tab;
  C2(int m, int n, int p){
  // dans l'objet construit, val = m, c1.val = n, tab=[1,2, ..., p]
    val = m; c1 = new C1(n); tab = new int[p];
    for(int i = 0; i < p; i++)
        tab[i] = i + 1;
  }
}
```

On considère les fonctions suivantes :

```
static int f1(int x){ x = x - 1; return x; }
static void f2(int[] T){ T[0]= T[0] + 1; }
static void f3(int[] T,int x, int y){ T[1] = x; T[0] = y; }
static void f4(C1 c, int x){ c.val = x; }
static void f5(C2 c, int[] t2){ int[]t = c.tab; c.tab = t2; t2 = t; }
static void f6(C2 c2, C1 c1, int[] tab, int n){
     c1.val += n; c2.c1 = c1; c2.c1.val += n; c2.tab = tab; }
static void ecrire(int[] tab){ // impression sur une ligne du contenu d'un tableau
     for(int i = 0; i < tab.length; i++)
          System.out.print(tab[i] + " ");
     System.out.println();
     }
```

Dans la suite, on supposera que chaque code exécuté est précédé de la séquence suivante :

```
int m , n; C1 c1; C2 c2; int[] tab;
m = 3; n = 5; c1= new C1(20); c2 = new C2(1, 2, 3);
tab = new int[2]; tab[0] = -1; tab[1] = -2;
```

Que produit dans ces conditions chacune des séquences suivantes (justifier brièvement vos réponses) :

1.1.
```
n = f1(m); tab[1] = f1(tab[0]);
System.out.println(m + " " + n); ecrire(tab);
```

1.2.
```
f1(c1.val); System.out.println(c1.val);
```

1.3.
```
f2(tab); ecrire(tab);
```

1.4.
```
f4(c1, 9); System.out.println(c1.val);
```

1.5.
```
f5(c2, tab); ecrire(c2.tab); ecrire(tab);
f3(c2.tab, 21, 22); ecrire(c2.tab); ecrire(tab);
f2(c2.tab); ecrire(c2.tab); ecrire(tab);
```

1.6.
```
f6(c2, c1, tab, 2); System.out.println(c1.val + " " + c2.c1.val);
ecrire(c2.tab); ecrire(tab); c1.val = f1(c2.c1.val);
System.out.println(c1.val + " " + c2.c1.val);
```

Exercice 2. *Récursion et itération*
On considère la suite u d'entiers définie par :

- $u_n = n + 1$ si $n \leq 2$ (c'est-à-dire que $u_0 = 1$, $u_1 = 2$ et $u_2 = 3$) ;
- $u_n = (n-2) \times u_{n-1} + 3 \times u_{n-2} - 2 \times u_{n-3}$ sinon (c'est-à-dire si $n \geq 3$).

2.1. Donner le code Java d'une fonction f qui, appelée avec un paramètre entier n, calcule u_n par traduction récursive directe de la définition de la suite.
2.2. Dessiner l'arbre des appels récursifs correspondant au calcul de $f(5)$.
2.3. Donner la suite des appels dans l'ordre où ils sont exécutés pour le calcul de $f(5)$.
2.4. Écrire en java une version "programmation dynamique" récursive de la fonction f.
2.5. Dans quel ordre est rempli le tableau utilisé pour le calcul de $f(5)$ et quel est l'arbre des appels récursifs ?
2.6. Donner une version itérative de la fonction f utilisant un tableau d'entiers pour stocker toutes les valeurs effectivement calculées pour calculer u_n.
2.7. Donner une version itérative de calcul de la fonction f pour une valeur n n'utilisant pas de tableau mais uniquement des variables simples pour mémoriser les valeurs utiles au calcul de u_n.
2.8. Donner une version récursive terminale du code de la fonction f.
2.9. Parmi les différentes versions proposées, quelle est la plus lente à l'exécution ? Quelle est la plus rapide ?

Exercice 3. *Entrée en scène à la Samuel Beckett*
Dans sa pièce Quad, Samuel Beckett fait entrer et sortir les n personnages un à un de telle sorte que, la scène étant initialement vide, tous les sous-ensembles possibles (2^n) de personnages apparaissent exactement une fois sur la scène.
Ainsi pour $n = 4$, cela peut donner ($\rightarrow^{+i}$ signifie entrée de i et $\rightarrow^{-i}$ sortie de i) :
$\emptyset \rightarrow^{+1} \{1\} \rightarrow^{+2} \{1,2\} \rightarrow^{-1} \{2\} \rightarrow^{+3} \{2,3\} \rightarrow^{+1} \{1,2,3\} \rightarrow^{-2} \{1,3\} \rightarrow^{-1} \{3\} \rightarrow^{+4} \{3,4\}$
$\rightarrow^{+1} \{1,3,4\} \rightarrow^{+2} \{1,2,3,4\} \rightarrow^{-1} \{2,3,4\} \rightarrow^{-3} \{2,4\} \rightarrow^{+1} \{1,2,4\} \rightarrow^{-2} \{1,4\} \rightarrow^{-1} \{4\}$.

Le but de cet exercice est de donner une solution à ce problème pour un nombre n donné.

On associe à un instant donné à l'ensemble des n personnages (supposés numérotés de 1 à n) une suite de n caractères 0 ou 1, un 1 en position i indiquant que le personnage i est actuellement sur la scène. Ainsi pour $n = 6$, 011010 indique que les personnages sur la scène sont 2, 3 et 5.

Le problème initial se ramène alors à générer une seule fois chacune des séquences de 0 et de 1 de longueur n en commençant par la séquence 0^n constituée de n caractères 0 et de telle sorte qu'en passant d'une séquence à la suivante un seul caractère soit changé.
3.1. Donner la suite des séquences de 0 et de 1 correspondant à la suite des entrées et sorties de personnages donnée en exemple pour $n = 4$.
3.2. Montrer que si on connaît une solution S_n pour la valeur n (c'est-à-dire une suite de 2^n séquences de longueur n constituée de 0 et de 1), une solution S_{n+1} est obtenue de la manière suivante :

1. ajouter un caractère 0 à la fin de chaque suite de la solution S_n ;
2. ajouter après la suite de 2^n séquences de longueur $n+1$ ainsi obtenue, la suite de 2^n séquences de longueur $n+1$ obtenue en inversant la suite S_n et en ajoutant un caractère 1 à la fin de chacune des séquences de la suite ainsi obtenue.

Ainsi si $S_n = g_1, g_2, \ldots g_{2^n}$ alors $S_{n+1} = g_1 0, g_2 0, \ldots, g_{2^n} 0, g_{2^n} 1, \ldots, g_2 1, g_1 1$
3.3. Construire à la main (et complètement) la solution S_4 à laquelle conduit ce procédé.
3.4. Donner le code Java de deux fonctions `gen1` et `gen2` récursives (supposées générer les solutions de taille n dans l'ordre et l'ordre inverse) et s'appelant mutuellement permettant l'affichage de la suite des séquences d'une solution.
On utilisera un tableau global de taille n pour construire les séquences.
Indication : parcourir à l'envers une suite inversée revient à lire la suite d'origine à l'endroit !
3.5. Donner le code java d'une fonction `mvts` n'utilisant pas de tableau et donnant la suite des entrées et sorties successives de personnages dans une solution (typiquement $+1, +2, -1, ..., -2, -1$ pour $n = 4$).
Indication : on définira une fonction à deux paramètres : le premier correspondra à un numéro de personnage et le second sera un indicateur de présence sur la scène de ce personnage.