Partiel TO2

19 mars 2011 - Durée 2h

Exercice 1 (Questions de cours)

1. Récursivité
   (i) Rappelez les règles à appliquer lors de la conception d'un programme récursif.
   (ii) Dans le cas de la fonction de Morris, on a le code suivant :

```
static int morris(int m, int n) {
     if (m == 0) return 1;
     else        return morris(m - 1, morris(m, n));
}
```

   Les règles mentionnées à la question (i) sont-elles bien appliquées ? Justifiez clairement votre réponse.

2. Tris de tableaux
   On dispose d'un tableau d'entiers que l'on voudrait trier.
   (i) Décrivez un algorithme de tri de votre choix et présentez son code java.
   (ii) Quelle est la complexité (dans le pire et dans le meilleur des cas) de votre algorithme ?

Exercice 2 On considère la méthode suivante :

```
public static int toto(int x, int y) {
     int a = 0, b = x;
     while(b >= y) {
          b -= y;
          a++;
     }
     return a;
}
```

1. Donnez une précondition **Pre** et une postcondition **Post** de la méthode `toto`.
2. Donnez un invariant **I** pour la boucle.
3. Prouvez que **I** est un invariant de la boucle (faire le lien avec **Pre** et **Post**).
4. Que fait la méthode `toto` ?

Exercice 3 On considère une classe `Recipient` avec :

(i) un attribut privé `volume` de type entier (le volume maximal de liquide que peut contenir le récipient) et un attribut privé `contenu` de type entier (le volume contenu actuellement dans le récipient),

(ii) un constructeur `Recipient(int volume)` qui construit un récipient de capacité `volume`, complètement vide,

(iii) et des méthodes publiques :
- `void vider()` qui vide le contenu du récipient courant,
- `void remplir()` qui remplit complètement le récipient courant,
- `boolean vide()` qui teste si le recipient courant est vide,
- `boolean plein()` qui teste si le recipient courant est plein,
- `void transvaser(Recipient R)` qui transvase le contenu du récipient courant dans le récipient `R` (attention à ne pas transvaser plus de liquide que la place qu'il reste dans `R`).

A. Programmation des objets

1. Écrivez la classe `Recipient`.
2. Écrivez une classe `TestRecipient` avec une méthode `main` qui teste les méthodes de la classe `Recipient`.

B. Comparaison des volumes

Dans cette partie, on veut comparer les volumes de deux récipients `R1` et `R2`, en utilisant uniquement les méthodes de la classe `Recipient` (et pas l'attribut `volume` qui est privé !). Pour cela, on s'autorise à changer les contenus de `R1` et `R2`.

1. Comment peut-on comparer le volume des deux récipients `R1` et `R2` en utilisant uniquement les méthodes de la classe `Recipient` ? Dans la classe `TestRecipient`, écrivez deux méthodes :
   - `public static boolean plusVolumineux(Recipient R1, Recipient R2)` qui teste si le recipient `R1` a un `volume` supérieur ou égal à celui de `R2`,
   - `public static boolean memeVolume(Recipient R1, Recipient R2)` qui teste si les deux récipients ont le même volume.
2. Donnez un exemple d'utilisation des méthodes de la question précédente dans la méthode `main`. Pourquoi a-t-on déclaré ces deux méthodes en `static` ?
3. Écrivez une méthode `public static int quotient(Recipient R1, Recipient R2)` qui renvoie le quotient du `volume` du récipient `R1` par celui du récipient `R2`. Souvenez vous que pour calculer le quotient de a par b, on calcule "combien de fois il y a b dans a".

C. Minimum et maximum simultanés dans un tableau

1. Écrivez une méthode `minEtMax` qui prend en argument un tableau `t` de récipients, trouve les indices des récipients de volume minimum et maximum de ce tableau, et renvoie le quotient du volume maximum par le volume minimum des récipients de ce tableau. Vous pouvez utiliser les méthodes `plusVolumineux` de la question B.1 et `quotient` de la question B.3 même si vous ne les avez pas programmées.
2. Combien de comparaisons utilise votre méthode `minEtMax` ? Combien de liquide gaspillez-vous si votre tableau contient n récipients de volume v, tous vides au départ ?
3. Lorsque l'on appelle `plusVolumineux(R1, R2)`, on change le contenu des récipients `R1` et `R2`. Pour calculer le minimum et le maximum d'un tableau de récipients sans changer leurs contenus, on doit faire une copie en profondeur de ce tableau.
   Expliquez ce qu'est une copie en profondeur d'un tableau d'objets. Écrire un constructeur `Recipient(Recipient R)` dans la classe `Recipient` qui copie un récipient, et une méthode `copieTableauRecipients` qui copie en profondeur un tableau de récipients.

Pour gaspiller moins de liquide, on adopte une stratégie plus économique. On commence par parcourir le tableau en considérant les récipients deux par deux : le récipient `t[0]` avec le récipient `t[1]`, le récipient `t[2]` avec le récipient `t[3]`, etc. Dans chaque paire, on place le récipient le moins volumineux en premier et le récipient le plus volumineux en deuxième. Ensuite, on cherche le récipient de volume minimum parmi les récipients du tableau en position paire, et le récipient de volume maximum parmi les récipients du tableau en position impaire.

4. Écrivez une méthode `minEtMaxEconomique` qui implémente cette stratégie.
5. Combien de comparaisons utilise la méthode `minEtMaxEconomique` ?

En fait, cette méthode est optimale pour le calcul simultané du minimum et du maximum d'un tableau : c'est la méthode qui fait le moins de comparaisons.