Université Pierre et Marie Curie

Master de Sciences et Technologies, mention Biologie Moléculaire et Cellulaire

Méthodes Informatiques en Biologie - BMC427

Année M1 - Session de Juin 2006

Durée 2h ; tous documents autorisés

A. Première partie

On s'intéresse à la production quotidienne d'oxygène par une forêt, production résultant de la photosynthèse diurne.

A.1. On considère une forêt de feuillus en climat tempéré. On admet que toutes les feuilles d'un arbre sont également exposées à la lumière, et que leur production d'oxygène est proportionnelle à leur surface.

A.1.a. Quels sont les objets qui apparaissent dans l'analyse du problème de la production globale d'oxygène par la forêt ? Quelles sont leurs relations ?

A.1.b. Donner le code des classes correspondantes. On définira les attributs propres à chaque objet, ainsi que les méthodes associées, dont on ne donnera pas le code, mais simplement leur rôle, sous forme d'un simple commentaire.

A.2. On veut maintenant évaluer la production quotidienne moyenne au cours des 12 mois de l'année. On considère pour cela que :
- les arbres n'ont pas de feuille de décembre à mars,
- en mars et avril, les feuilles ont une surface réduite de moitié,
- en octobre et novembre, la production d'oxygène par une feuille de surface donnée est réduite de moitié.

Sans entrer dans le détail du code des méthodes, préciser les modifications à apporter aux attributs et/ou méthodes du A.1. pour traiter ce nouveau problème.

A.3. On étudie maintenant une forêt mixte feuillus/conifères. On suppose que les conifères ont des aiguilles de taille constante, qu'elles ne tombent pas en hiver, et que leur production d'oxygène est constante dans l'année.

En comparant les propriétés des deux catégories d'arbres (feuillus et conifères), proposer une hiérarchie de classes pour représenter tous les arbres.

Définir pour chaque classe les attributs et les méthodes nécessaires, et indiquer les modifications à apporter à la représentation globale de la forêt pour généraliser la notion d'arbre.

B. Deuxième partie

B.1. Les deux classes suivantes composent un programme permettant de comparer deux taxons en calculant un indice de différence à partir des valeurs de plusieurs caractères.

```
class Taxon {
  int nk;
  int caract[];
  Taxon(int n) {
    nk = n;
    caract = new int[n];
  }
}

import java.io.*;
class Compar {
  public static void main(String arg[]) {
    InputStreamReader isr = new InputStreamReader(System.in);
    BufferedReader flux = new BufferedReader(isr);
    String str = "";
    int nb=0;
    System.out.print("Nombre de caractères ? ");
    try {
      str = flux.readLine();
      nb = Integer.parseInt(str);
    }
    catch(IOException e) {
      System.out.println("Entrée incorrecte !");
    }
    System.out.println("Taxon 1");
    Taxon tax1 = new Taxon(nb);
    for(int i=0;i<tax1.nk;i++) {
      System.out.print("Caractère "+i+" = ");
      try {
        str = flux.readLine();
        tax1.caract[i] = Integer.parseInt(str);
      }
      catch(IOException e) {
        System.out.println("Entrée incorrecte !");
        tax1.caract[i] = 0;
      }
    }
    System.out.println("Taxon 2");
    Taxon tax2 = new Taxon(nb);
    for(int i=0;i<tax2.nk;i++) {
      System.out.print("Caractère "+i+" = ");
      try {
        str = flux.readLine();
```

```
        tax2.caract[i] = Integer.parseInt(str);
      }
      catch(IOException e) {
        System.out.println("Entrée incorrecte !");
        tax2.caract[i] = 0;
      }
    }
    int somdif = 0;
    for(int i=0;i<tax1.nk;i++) {
      int d = tax1.caract[i]-tax2.caract[i];
      if (d<0)
        d = -d;
      somdif = somdif + d;
    }
    System.out.println("Différence = "+somdif);
  }
}
```

B.1.a. Corriger la classe `Taxon` afin de respecter le principe d'encapsulation.

B.1.b. Modifier la classe `Compar` pour tenir compte de cette nouvelle version de la classe `Taxon`.

B.2. Les caractères ne sont plus représentés simplement par des valeurs entières, mais par la classe `Caractere` suivante :

```
class Caractere {
  private int info;
  public Caractere (int inf) {
    info = inf;
  }
  public int difference (Caractere autreCaractere) {
    int ecart = info - autreCaractere.info;
    if (ecart < 0)
      return -ecart;
    else
      return ecart;
  }
}
```

Modifier les classes `Taxon` et `Compar` en tenant compte de cette nouvelle représentation.

B.3. On veut intégrer d'autres types de caractères, représentés éventuellement par une information différente d'un simple entier. Sans entrer dans le détail du code, préciser les grandes lignes des modifications à apporter dans la représentation des caractères.

Quelles seraient les conséquences sur le code de la classe `Compar` proposé au B.2. ?