TP2 Java : Pour enfoncer le clou

Le travail est individuel. Les programmes informatiques sont à envoyer sous forme d'une archive `.tar` ou `.zip` par email à `petru.valicov@ens-lyon.fr`. L'archive doit contenir le nom et le prénom. La date limite est fixée au dimanche 24/11/2013.

Avant-propos : comme d'habitude, tout programme informatique doit être testé ! Mais
Program testing can be used to show the presence of bugs, but never to show their absence !

Edsger W. Dijkstra

Lecture/écriture dans un fichier

Pour pouvoir traiter les entrées/sorties (console ou fichiers) il faut ajouter au tout début de votre fichier `.java` :

```
import java.io.*;
```

En Java il y a plusieurs façons d'ouvrir un fichier. Nous allons utiliser le mode caractère. La méthode suivant (à intégrer dans une classe) lit et affiche toutes les lignes du fichier `MonFichier.txt` :

```
public static void main(String[] args) throws IOException {
  File fichier = new File("monFichier.txt");
  Scanner sc = new Scanner (fichier);
  while (sc.hasNextLine())
  {
    String line = sc.nextLine();
    System.out.println (line);
  }
  sc.close();
}
```

Le code suivant écrit dans le fichier `output.txt` deux lignes de texte.

```
public static void main(String[] args) throws IOException{
  File fichierSortie = new File ("output.txt");
  FileWriter fWriter = new FileWriter (fichierSortie);
  PrintWriter pWriter = new PrintWriter (fWriter);
  pWriter.println ("cette ligne est écrite dans le fichier");
  pWriter.println ("Une autre ligne écrite dans le même fichier.");
  pWriter.close();
}
```

Remarquez l'utilisation de l'instruction `throws IOException`. Il s'agit de la gestion des exceptions – des cas d'anomalies qui peuvent éventuellement se produire. Par exemple, le fichier auquel vous essayez d'accéder peut ne pas exister ou a été modifié en même temps par un autre programme (le super-utilisateur). Ici le type d'exception est très général (ce qui n'est pas très bien !) car il s'agit de toutes les exceptions d'entrées/sorties. Un bon programmeur Java spécifie les exceptions éventuelles de façon à adapter le traitement. Cependant, dans notre cas nous allons nous contenter avec l'utilisation de `throws IOException` à la déclaration de la méthode qui manipulent des fichiers.

Suite de Conway (c'est dur le réveil)

La suite de Conway est définie comme suit. Le premier terme est 1. Chaque terme de la suite se construit en *annonçant* le terme précédent :
$c_0 = 1$
$c_1 = 11$ (dans la suite précédente c_0 il y a *un 1*)
$c_2 = 21$ (dans la suite précédente c_1 il y a *deux 1*)
$c_3 = 1211$ (dans la suite précédente c_2 il y a *un deux* et *un 1*)
$c_4 = 111221$ (???)
...

Exercice 1 *Programmez une fonction récursive qui lit dans un fichier* `conwayEntree` *un entier n et écrit dans un fichier* `conwaySortie` *les n premiers termes de la suite de Conway (un terme par ligne).*

Exercice 2 *Ajoutez une fonction qui lit le fichier* `conwaySortie` *et recopie :*
- *dans un fichier* `conwaySortieOdd` *les termes se terminant par 11 ;*
- *dans un fichier* `conwaySortieEven` *les termes se terminant par 21 ;*
- *dans un fichier* `conwaySortieOthers` *les autres termes.*

Les algorithmes, les cours, les graphes, les vélos, ...

en informatique ce sont des **objets**. Un objet est l'association d'un état et d'un comportement. L'état est représenté par des **champs** (attributs ou variables) et le comportement est exposé via des **méthodes** (ou fonctions). Les méthodes manipulent l'état interne de l'objet et servent à la communication inter-objets : c'est le principe d'*encapsulation*. En gros la seule façon d'accéder à la description d'un objet c'est de passer par ses méthodes, d'où la nécessité de mettre les attributs de classe (les variables globales) en privé avec le mot clé `private`.
Une application est un ensemble d'objets collaborant entre eux. Le comportement d'une application (orientée objet) repose sur la communication entre les objets qui la composent.

Les classes

Une classe est une description abstraite d'un ensemble d'objets "de même nature". C'est en quelque sorte un modèle pour la création de nouveaux objets. Désormais nous allons créer une classe par fichier (qui va porter le même nom). Le cas des plusieurs classes par fichiers arrive lorsqu'on parle de classes internes (privées) que nous ne traitons pas ici.
Les classes Auteur et Livre donnez sur la page suivante permettent de modéliser une partie d'un système de gestion de livres dans une librairie. Notez qu'aucune des ces deux classes ne contient une méthode `static void main(String args[])`. Et c'est normal car dans une application il n'y a qu'un seul programme principal. Dans notre exemple cette méthode se trouve dans la classe `TestLivre`.

Exercice 1 *Quels seront les résultats d'affichage de la fonction* `main` *de la classe* `TestLivre` *?*

Exercice 2 *Si on change la déclaration de la variable* `prix` *de la classe* `Livre` *en*

```
private static double prix
```

quelle sera la valeur du prix du `livre1` *? Testez ce changement sur différentes variables.*

```java
package bibliotheque;

public class Livre {
    // Des attributs privés
    private String nom;
    private Auteur auteur;
    private double prix;
    private int qtteEnStock;

    // Constructeurs
    // 1er constructeur
    public Livre(String n, Auteur a, double p) {
        nom = n;
        auteur = a;
        prix = p;
        qtteEnStock = 0; //on met la valeur à 0
    }

    /* 2ème constructeur avec
    un paramètre supplémentaire pour la quantité en Stock*/
    public Livre(String n, Auteur a, double p, int qtteEnStock) {
        this.nom = n;
        this.auteur = a;
        this.prix = p;
        this.qtteEnStock = qtteEnStock;
        /* le mot clé "this" signifie
           qu'il s'agit de l'objet membre de la clase */
    }

    // Accesseurs et Modifieurs
    public String getNom() {
        return nom;
    }

    // la fonction retourne un objet de type Auteur
    public Auteur getAuteur() {
        return auteur;
    }

    public double getPrix() {
        return prix;
    }

    public void setPrix(double p) {
        prix = p;
    }

    public int getQtteEnStock() {
        return qtteEnStock;
    }

    public void setQtteEnStock(int q) {
        this.qtteEnStock = q;
    }
}
```

```java
package bibliotheque;

//The Author class definition
public class Auteur {

    // Des attributs privés
    private String nom;
    private String email;
    private char gender;

    // Constructeurs
    public Auteur(String name, String email, char gender) {
        this.nom = name;
        this.email = email;
        this.gender = gender;
    }

    /* Les accesseurs et les modifieurs
       pour les variables privées */
    public String getNom() {
        return nom;
    }

    public String getEmail() {
        return email;
    }

    public char getGender() {
        return gender;
    }

    public void setEmail(String email) {
        this.email = email;
    }

}
```

```java
package bibliotheque;

public class TestLivre {

    public static void main(String[] args) {

        Auteur monNom = new Auteur("Toto", "toto@ens-lyon.fr", 'f');
        System.out.println("Le nom de l'auteur : "+monNom.getNom());
        System.out.println("L'email de l'auteur : "+monNom.getEmail());
        Livre livre1 = new Livre("Un livre de maths ",
                monNom, 15, 88);
        System.out.println("Le nom du livre "+ livre1.getNom());
        System.out.println("Le prix = "+ livre1.getPrix());
        System.out.println("Il y a "+
                livre1.getQtteEnStock()+" livres en stocks");

        Livre livre2 = new Livre("Encore un livre de maths",
                new Auteur("Poincaré", "poincare@ens-lyon.fr", 'm'),
                25, 85);
        System.out.println("Le nom du livre "+ livre2.getNom());
        System.out.println("Il y a "+
                livre2.getQtteEnStock()+" livres en stocks");
        System.out.println("Le prix = "+ livre2.getPrix());

    }
}
```

Les classes `Livre` et `Auteur` et la classe de test qui manipulent des objets correspondants
Pour les compiler et exécuter il faut tenir compte du package :

```
javac bibliotheque/*.java
java -cp ./bibliotheque TestLivre
```

De manière générale, il est fortement déconseillé d'utiliser des variables statiques et leurs utilisation est très rare. Comme le terme l'indique leurs utilisation empêchent les objets d'être modifiés dynamiquement ce qui limite leurs utilisations. Donc règle :

pas de variables statiques !

Si vous en avez besoin dans votre programme vous pouvez être presque sûrs que la modélisation est mal choisie.

Ci-dessous un exemple de programme modélisant des figures géométriques Cercle et Rectangle.

```java
public class Cercle {

    // Variables privees
    private double rayon;
    private String couleur;

    /* Les constructeurs
       Il s'agit de la surcharge de la méthode :
       il y a plusieurs versions */

    // 1er Constructeur
    public Cercle() {
        rayon = 2.0;
        couleur = "bleu";
    }

    // 2nd Constructeur
    public Cercle(double r) {
        rayon = r;
        couleur = "bleu";
    }

    // 3ème Constructeur
    public Cercle(double r, String c) {
        rayon = r;
        couleur = c;
    }

    // Méthodes publiques

    // les accesseurs :
    // renvoient la valeur des champs privés
    // (le mot clé GET)
    public double getRadius() {
        return rayon;
    }
    public String getColor() {
        return couleur;
    }
    public double getArea() {
        return rayon*rayon*Math.PI;
    }
}
```

```java
public class Rectangle {

    // Variables privees
    private double hauteur;
    private double largeur;
    private String couleur;

    // Les constructeurs
    public Rectangle(double lrg, double htr){
        hauteur = htr;
        largeur = lrg;
    }

    public Rectangle(){
        hauteur = 1;
        largeur = 1;
        couleur = "White";
    }

    // les modifieurs :
    // permettent de changer la valeur des champs privés
    // (le mot clé SET)
    public void setHeight(double htr){
        hauteur = htr;
    }

    public void setWidth(double lrg){
        largeur = lrg;
    }

    public void setColor(String col){
        couleur = col;
    }

    // les accesseurs :
    // renvoient la valeur des champs privés
    // (le mot clé GET)
    public double getArea(){
        return hauteur*largeur;
    }

    public double getPerimeter(){
        return 2*(hauteur + largeur);
    }

    public void getColor(){
        System.out.println("La couleur est : " + couleur +"\n");
        return;
    }
}
```

Exercice 3 *Écrivez une classe contenant une méthode* `main` *pour tester ces classes. Enrichissez l'application pour permettre aux cercles d'avoir un centre de gravité.*

Exercice 4 *C'est quoi un carré d'après la spécification qui vous est donnée ? Implémentez le.*

Les collections et d'autres structures de données

Jusque là, la seule structure de données en Java que nous avons vu sont les tableaux. Le problème avec les tableaux c'est qu'ils ne sont pas du tout dynamiques : par exemple, il n'est pas possible de modifier la taille d'un tableau sans effacer entièrement son contenu.
En Java il y a un mécanisme très puissant de gestion des différentes structures de données appelées des **Collections**. Ainsi, il est possible d'utiliser des listes chaînées, que vous avez certainement déjà rencontré, des files, des piles, des ensembles etc. On vous invite à regarder le polycopié qui vous a été distribué pour plus de détails.
Pour s'entraîner faites l'exercice suivant

Exercice 5 (cette exercice est à faire en dernier) *Utilisez une structure de données plus adaptée pour implémenter une nouvelle version de l'algorithme de calcul du factoriel d'un grand nombre entier.*

Exercice 6 *Implémentez la structure d'arbre binaire (vous aurez votre propre collection !). Écrivez également une fonction d'affichage d'un arbre binaire (le style est laissé libre, la seule règle est qu'il ne doit pas y avoir d'ambiguïté pour la lecture)*

Exercice 7 *On souhaite implémenter l'algorithme d'encodage de Huffman qui sert à la compression d'un texte donné. On suppose qu'on utilise l'alphabet latin sans accents (26 lettres donc). L'idée de l'algorithme est très simple :*

***Algorithme 1** : Algorithme Huffman*

Input *: Un texte*
Output *: Un codage binaire de chaque lettre*

1. *Calculer le nombre d'apparitions de chaque lettre dans le texte.*
2. *Pour chaque caractère créer un arbre binaire avec un seul nœud étiqueté par la fréquence d'apparition de la lettre correspondante*
3. *Itérativement, fusionner deux arbres dont la fréquence est minimale en rajoutant le même père aux deux racines. L'étiquette du père est la somme des étiquettes des fils*
4. *L'algorithme s'arrête lorsqu'il reste qu'un seul arbre.*

Implémentez l'algorithme en Java et affichez les caractères et l'encodage correspondants. Est-il unique ? Testez sur des exemples.