TP1

Un peu d'algorithmie

Exercice 1 – *Division euclidienne.* Ecrire une méthode `public static void division(int a, int b)` qui effectue la division euclidienne de deux entiers a et b par l'algorithme suivant :

- $q = 0$, $r = a$.
- tant que $r > b$, $q = q + 1$, $r = r - b$.

Le programme devra avoir une fonction main qui utilise votre méthode. L'affichage doit comporter les calculs intermédiaires comme dans l'exemple :

```
11 = 4 * 0 + 11
11 = 4 * 1 + 7
11 = 4 * 2 + 3
```

Exercice 2 – *Triangle de Pascal.* Le triangle de Pascal est la figure suivante :

$$
\begin{array}{ccccc}
1 & & & & \\
1 & 1 & & & \\
1 & 2 & 1 & & \\
1 & 3 & 3 & 1 & \\
1 & 4 & 6 & 4 & 1
\end{array}
$$

1. Écrire une méthode `public static long factoriel(int a)` qui calcule $a!$. Pour rappel factoriel de cinq : $5! = 1 \times 2 \times 3 \times 4 \times 5$
2. Écrire une méthode `public static long aParmib(int a, intb)` qui calcule $\binom{b}{a}$. Pour rappel $\binom{b}{a} = \frac{n!}{k!(n-k)!}$.
3. Écrire une méthode `public static void trianglePascal(int n)` qui affiche le triangle de pascal jusqu'à la ligne n.
   Le nombre situé à l'intersection de la ligne n et de la colonne p est $\binom{n}{p}$. On a donc :

$$
\begin{array}{ccccc}
\binom{0}{0} & & & & \\
\binom{0}{1} & \binom{1}{1} & & & \\
\binom{0}{2} & \binom{1}{2} & \binom{2}{2} & & \\
\binom{0}{3} & \binom{1}{2} & \binom{2}{3} & \binom{3}{3} & \\
\binom{0}{4} & \binom{1}{3} & \binom{2}{4} & \binom{3}{4} & \binom{4}{4}
\end{array}
$$

4. Tracer 20 lignes du triangle de Pascle. Que ce passe t'il ? Pourquoi utiliser des long plutôt que des int ?

Marre de coder ?

Exercice 3 – *Code à trou 1.* Votre tâche consiste à prendre des fragments de code dans ceux proposés ci-dessous et à les remettre dans les programmes à la place des blancs. Vous ne pouvez pas utiliser le même fragment deux fois, et vous n'avez pas besoin de les utiliser tous. Votre but est de créer une classe qui se compile, s'exécute et produit le résultat indiqué. Ne vous y trompez pas : c'est plus difficile qu'il n'y paraît.

Les fragments :

```
x>0   x = x + 1;   System.out.print(‘‘ ‘‘);          System.out.print(‘‘x cyprès si près‘‘);
x<1   x = x + 2;   System.out.print(‘‘si‘‘);         System.out.print(‘‘x scies scient‘‘);
x>1   x = x - 2;   System.out.print(‘‘six‘‘);        System.out.print(‘scies‘‘);
x>3   x = x - 1;   System.out.print(‘‘Sissi‘‘);      System.out.print(‘six scies‘‘);
x<4                System.out.print(‘‘ x cyprès‘‘);
```

Le résultat attendu :

```
% java ExoSadique
si Sissi
six scies scient
six cyprès si près
```

Le code à trous :

```
class ExoSadique {
  public static void main(String [] args) {
    int x = 0;
      while( /* un trou */ ) {
        // un trou
        if( x < 1 ) {
          // un trou
        }
        // un trou
        if( /* un trou */) {
          // un trou
          // un trou
        }
        if( x == 1) {
          // un trou
        }
        if( /* un trou */) {
          // un trou
        }
        System.out.println(" ");
        // un trou
      }
  }
}
```

Classes

Exercice 4 – *une première "vraie" classe.* Voici le programme source de la classe Livre :

```
public class Livre {
  // Données membres
  private String titre, auteur;
  private int nbPages;

  // Constructeur
  public Livre(String unAuteur, String unTitre) {
    auteur = unAuteur;
    titre = unTitre;
  }

  // Accesseur
  public String getAuteur() {
    return auteur;
  }

  // Modificateur
  public void setNbPages(int nb) {
    nbPages = nb;
}
```

Ajoutez une méthode `public static void main(String [] args)` pour
- Créer 2 livres,
- Afficher les auteurs de ces 2 livres.

Exercice 5 – *Accesseurs et modificateurs.* Modifiez la classe Livre :
- Ajoutez un accesseur pour la variable titre et la variable nbPages.
- Ajouter un modificateur pour les variables auteur et titre.
- Changez le modificateur de nbPages : il ne doit changer le nombre de pages que si on lui passe en paramètre un nombre positif ; sinon il affiche un message d'erreur.

Dans la méthode main(), définissez un nombre de pages pour chacun des 2 livres, affichez ces nombres de pages, calculez le nombre de pages total de ces 2 livres et affichez-le.

Exercice 6 – *Utilisation de 2 classes.* Ecrivez une classe TestLivre avec une seule méthode main() qui fait la même chose que la méthode main() de la classe Livre.
Dans quel(s) fichier(s) pouvez-vous définir les deux classes Livre et TestLivre ?
Comment compiler et exécuter la méthode main() de la classe TestLivre ?

En général il vaut mieux avoir un fichier .java par classe.

Exercice 7 – *Méthode toString().*

1. Dans la classe Livre, ajoutez une méthode afficheToi() qui affiche les informations concernant un livre (auteur, titre et nombre de pages). Utilisez afficheToi() dans la méthode main() de TestLivre.
2. Ajoutez l'instruction System.out.println(livre) où livre désigne un des livres que vous avez créés. Que se passe t-il ?
3. Ajoutez une méthode publique toString() qui renvoie une chaîne de caractères décrivant le livre. Que fait alors l'instruction System.out.println(livre) ?

Exercice 8 – *Les constructeurs.* Ajoutez 2 nouveaux constructeurs dans la classe Livre pour permettre les constructions suivantes :

- sans aucun paramètre donné,
- pour un auteur, un titre et un nombre de pages donnés.

Utilisez ces 3 constructeurs (et éventuellement d'autres méthodes) pour créer 3 livres de 300 pages dans la méthode main() de TestLivre.

Exercice 9 – *Contrôle des variables private par les modificateurs*.

1. Ajoutez un prix aux livres (nombre qui peut avoir 2 décimales de type Java double) avec 2 méthodes getPrix et setPrix pour obtenir le prix et le modifier. Ajoutez au moins un constructeur qui prend le prix en paramètre. Si le prix d'un livre n'a pas été fixé, la description du livre (toString()) devra indiquer "Prix pas encore fixé". Attention, un livre peut être gratuit.
2. On bloque complètement les prix : un prix ne peut être saisi qu'une seule fois et ne peut être modifié ensuite (une tentative pour changer le prix ne fait qu'afficher un message d'erreur). Réécrivez la méthode setPrix (et autre chose si besoin est). Ajoutez une méthode "isPrixFixe" qui renvoie vrai si le prix a déjà été fixé. Faut-il écrire une méthode "setPrixFixe" pour modifier la variable booléenne ?

Exercice 10 – *Comparaison de 2 livres*.

1. En utilisant une méthode d'instance dans la classe Livre écrivez une méthode d'instance compare pour comparer 2 livres sur leur nombre de pages. compare prend un livre en paramètre et elle renvoie 0 si ce livre a le même nombre de pages que l'instance courante, 1 si ce livre a moins de pages que l'instance courante et -1 sinon.
   Pour tester, vous utiliserez un code semblable au suivant :

```
System.out.print("L'auteur du plus gros livre est ");
String auteurPlusGrosLivre;
// Le code pour trouver le livre qui a le plus de pages (entre 2 livres)
. . .
System.out.println(auteurPlusGrosLivre);
```

2. En utilisant une méthode de classe dans la classe Livre écrivez une méthode de classe compare pour comparer 2 livres relativement à leurs nombres de pages.

Exercice 11 – *Délégation : une classe Comptable*. Cet exercice montre comment des instances d'une classe peuvent déléguer un travail à une autre classe :
Créez, dans un fichier séparé, une classe Comptable qui possède une méthode de signature "void comptabiliser(Livre l)". Une instance de cette classe permettra de calculer le prix total de tous les livres qu'on lui aura passé par la méthode "comptabiliser". Utilisez cette nouvelle classe dans la méthode main de TestLivre, en créant deux comptables qui comptabilisent chacun quelques livres et affichent les totaux de leurs prix. Pour simplifier l'utilisation de la classe Livre, on veut cacher la classe Comptable aux "clients" de la classe Livre. Sans changer la classe Comptable, modifiez la classe Livre pour que chaque livre enregistre automatiquement son prix auprès d'un unique comptable. Comment (à qui ?) demander alors le total des prix comptabilisés ?

Sources

- Livre : Java la tête la première,
- Site : wikipedia.