TP2 : Premiers pas avec Java

Florian Boudin

Module X0IC020 - 2013

Exercice 1 : rappel de quelques notions d'algorithmique

Cette série d'exercices est un rappel des notions d'algorithmique que vous avez acquises précédemment. Soyez le plus rigoureux possible : **indentez** et **commentez** votre code, nommez les variables de **manière intelligible**, etc. Créez un dossier `TD2` dans lequel vous enregistrerez les différentes classes que vous allez écrire.

1. Créez la classe `MultiplicationTable` dans laquelle la méthode `main` permet de faire afficher la table de multiplication de tous les nombres de 1 à 10. Utilisez le caractère tabulation (`'\t'`) pour aligner les valeurs. La sortie de votre programme doit être comme suit :

```
1   2   3   4   5   6   7   8   9  10
2   4   6   8  10  12  14  16  18  20
...
10 20  30  40  50  60  70  80  90 100
```

2. Créez la classe `SortThreeNumbers` dans laquelle la méthode `main` permet de trier trois entiers par ordre croissant. Les trois nombres seront stockés dans des variables locales `a`, `b` et `c` et seront triés en comparant et échangeant leurs valeurs. Vous pouvez résoudre cet exercice avec seulement trois conditions `if`.

3. Créez la classe `Trees` dans laquelle la méthode `main` permet de faire afficher les figures ci-dessous. Le nombre de ligne `n` étant une variable locale que l'on peut modifier à souhait. Vous utiliserez l'instruction `System.out.print("*");` pour faire afficher une étoile et `System.out.println();` pour faire afficher un retour à la ligne.

```
*              *              *            1 2 3 4 0
**            ***            **            2 3 4 0 1
***          *****          ***            3 4 0 1 2   si n = 5
****        *******        ****            4 0 1 2 3
...          ...            ...            0 1 2 3 4
```

Exercice 2 : utilisation de la méthode `random()`

Une stratégie de jeu à la roulette consiste à toujours miser sur la même couleur : si vous perdez votre mise, alors vous doublez la prochaine mise pour compenser la perte. La probabilité que le rouge (ou le noir) sorte est de 50% (nous ignorons l'existence du zéro). Créez la classe `Roulette` dans laquelle la méthode `main` permet de calculer, sachant une somme de départ de 100 euros et une mise minimum de 1 euro, combien de fois vous pouvez jouer avant de perdre tout votre argent. Utilisez l'expression `Math.random() < 0.5` pour vérifier la probabilité de gagner/perdre.

Exercice 3 : arguments en ligne de commande

Écrire la classe `CalculArithmetique` qui utilise trois arguments en ligne de commande : deux entiers suivis par un opérateur arithmétique (`+`, `-`, `*` ou `/`). Le programme doit réaliser l'opération arithmétique correspondante et afficher le résultat. Par exemple :

```
> java CalculArithmetique 3 2 +
  3+2=5
```

Indices

La méthode `main(String[] args)` possède un paramètre : un tableau de chaînes de caractères nommé `args`. Ce paramètre capture les arguments en ligne de commande au lancement du programme. Par exemple, si un utilisateur exécute la commande `java CalculArithmetique 12345 4567 +`, les trois arguments `"12345"`, `"4567"` et `"+"` seront stockés dans un tableau et passés à la méthode `main()` comme paramètres. Pur cet exemple, le contenu du tableau `args` sera :

```
args is {"12345", "4567", "+"};   // args est un tableau de String
args.length is 3;                 // longueur du tableau
args[0] is "12345";               // 1er element
args[1] is "4567";                // 2nd element
args[2] is "+";                   // 3eme element
```

Pour convertir une chaîne de caractères en entier, vous pouvez utiliser :

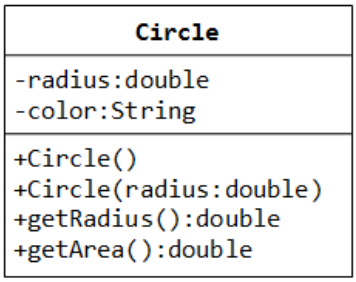
```
int entier = Integer.parseInt("chaine");
```

Exercice 4 : création d'objets et appels de méthodes

Téléchargez le fichier `Bicycle.java` à partir du site web du cours et mettez le dans le répertoire `TD2`. Il s'agit de la classe `Bicycle` que nous avons vu en cours. Créez la classe utilisatrice `DemoVelo` dans laquelle la méthode `main` permet de créer plusieurs instances de la classe `Bicycle` et de tester les méthodes disponibles.

Exercice 5 : définition de classes

Téléchargez le fichier `Circle.java` à partir du site web du cours et mettez le dans le répertoire `TD2`. La classe `Circle` est décrit dans le diagramme de classe ci-dessous. Elle contient : deux variables d'instance privées : `radius` et `color` ; deux constructeurs surchargés ; deux méthodes d'accès publiques : `getRadius()` et `getArea()`.



1. Compilez la classe `Circle.java` et essayez de l'exécuter. A quoi correspond le message d'erreur affiché par la commande `java` ? Compilez et exécutez la classe utilisatrice `TestCircle` décrite ci-dessous.

```
public class TestCircle {
  public static void main(String[] args) {
    Circle c1 = new Circle();
    System.out.println("Le cercle a un rayon de "
        + c1.getRadius() + " et une aire de " + c1.getArea());

    Circle c2 = new Circle(2.0);
    System.out.println("Le cercle a un rayon de "
        + c2.getRadius() + " et une aire de " + c2.getArea());
  }
}
```

2. Ajoutez un troisième constructeur à la classe `Circle` permettant que construire une instance de la classe avec les paramètres de rayon et de couleur spécifiés. Modifiez la classe utilisatrice pour tester votre constructeur.

```
public Circle(double r, String c) {...}
```

3. Ajoutez la méthode d'accès `getColor()` permettant de récupérer la couleur d'une instance de `Circle` (méthode appelée *getter*).

```
public String getColor() {...}
```

4. Dans la classe utilisatrice `TestCircle`, pouvez-vous accéder à la variable d'instance `radius` directement (e.g. `System.out.println(c1.radius)`) ou lui assigner une nouvelle valeur (e.g. `c1.radius = 0.5`). Essayez et expliquez les messages d'erreurs affiché par le compilateur.
5. Pensez-vous qu'il y a besoin de pouvoir modifier les valeurs de rayon de couleur des instances de Circle une fois qu'elles ont été construites ? Ajoutez deux méthodes publiques d'accès aux variables permettant de les modifier (méthodes appelées *setters*). Modifiez la classe utilisatrice pour tester vos méthodes.

```
public void setRadius(double r) {...}
public void setColor(String c) {...}
```

6. A la place d'utiliser des noms de variables tels que `r` (pour le rayon) et `c` (pour la couleur) dans les paramètres des méthodes, il est préférable d'utiliser `radius` et `color` avec le mot-clé `this` permettant de résoudre le conflit entre variable d'instance et paramètre de méthode. Modifiez les constructeurs et les méthodes de la classe `Circle` pour utiliser le mot-clé `this`.

```
public void setRadius(double radius) {
    this.radius = radius;
}
```

7. Toutes le classes java bien pensées doivent contenir une méthode `toString()` qui retourne une courte description de l'instance sous la forme d'une chaîne de caractères. Cette dernière est appelée explicitement lors d'un affichage via la méthode `println()`. Ajoutez la méthode `toString()` à la classe `Circle` permettant d'obtenir les informations sur le rayon et la couleur d'une instance. Modifiez la classe utilisatrice pour tester cette méthode via `println()`.

```
public String toString() {...}
```

Exercice 6 : implémentation de l'exercice 4 de la feuille de TD1

Écrire la classe `NombreCache` dont le seul champ est un entier nommé `unNombre`. Cette classe gère un entier qu'elle cache. Vous devez ensuite écrire :

1. Un constructeur `NombreCache` qui permet de créer une instance de la classe ; le nombre est choisi aléatoirement, entre 0 et 100. Vous utiliserez les méthodes de la classe `Math` suivantes :
   - `float random()` qui génère une valeur de type float appartenant à l'intervalle `[0,1]`
   - `int round(float a)` qui renvoie la valeur entière la plus proche du nombre passé en paramètre
2. Une méthode `compare` qui prend en paramètre un entier et renvoie -1 si l'entier en paramètre est inférieur à `unNombre`, 0 s'il est égal, et 1 si il est plus grand que `unNombre`

Programmez un jeu demandant à un utilisateur de trouver un nombre généré aléatoirement, avec un nombre maximum de tentatives permises. Pour cela, Écrivez la classe `Jeu` qui possède un champ `maxEssais` représentant le nombre de tentatives permises pour deviner le nombre. Vous devez ensuite écrire :

1. un constructeur `Jeu` qui prend en paramètre le nombre de tentatives maximum
2. une méthode `reInit(int nb)` qui prend un entier en paramètre et affecte sa valeur au champ `maxEssais`
3. une méthode `lancer` qui utilise la classe `NombreCache` pour créer un nombre cible entre 0 et 100 puis demande à l'utilisateur une valeur tant qu'il n'a pas trouvé le nombre cible et que le nombre de tentatives n'est pas atteint. Cette méthode renvoie la valeur booléenne `true` si le joueur a trouvé, et renvoie `false` sinon. La méthode affiche trop grand, trop petit ou trouvé selon la valeur proposée par l'utilisateur lors d'une tentative. Dans le cas où la valeur cible est trouvée, le nombre de tentative doit être affiché. Ainsi, avec `maxEssais` égal à 4, et un nombre à trouver égal à 76, un exemple de dialogue avec l'utilisateur après appel de la méthode serait :

```
Donne un entier : 14
trop petit !
Donne un entier : 87
trop grand !
Donne un entier : 76
trouve ! en 3 tentatives
```

Sources

Ce document est une adaptation d'exercices rédigés par Chua Hock-Chuan, Christian Pape et d'un sujet de TD similaire rédigé par Philippe Lamarre. Rédigé par Florian Boudin, 2011–12.