UNIVERSITÉ DU LUXEMBOURG

Programmation 1

Christian Grévisse (christian.grevisse@uni.lu)
Gilles Neyens (gilles.neyens@uni.lu)

TP 4 – Sous-programmes en Java

2017-10-17

Part I – Exercices pratiques sur la programmation orientée objet

Exercice 1 – Calculatrice

- Créez une classe `Calculatrice` avec un attribut `value` (valeur).
- Ajoutez un *constructeur* qui permettra d'initialiser la valeur de cet attribut avec la valeur du paramètre du constructeur.
- Ajoutez un autre constructeur sans paramètre qui appelle le premier constructeur et initialise la valeur initiale à 0.
- Ajoutez l'accesseur `getValue` qui retourne la valeur actuelle.
- Ajoutez les méthodes `ajouter`, `soustraire`, `multiplier` et `diviser`. Chacune d'entre elles prend un seul paramètre. La valeur actuelle va être modifiée selon la méthode appelée. La méthode `soustraire` doit réutiliser l'implémentation de la méthode `ajouter` et la méthode `diviser` doit réutiliser l'implémentation de la méthode `multiplier` (Notez bien: Il ne faut pas copier le code des méthodes `ajouter` et `multiplier`). Faites attention aux arguments pas valides!
- Ecrivez un programme principal qui contient une boucle REPL (anglais: *read-eval-print loop*, lecture-évaluation-affichage), c.-à-d. un menu d'options est présenté à l'utilisateur (pour les différentes options de la calculatrice) duquel il en pourra choisir une. Quand l'utilisateur a choisi une opération à effectuer il peut entrer la valeur pour effectuer l'opération. A chaque fois la nouvelle valeur de la calculatrice est affichée (au début la valeur est initialisée à 0). L'utilisateur doit aussi avoir la possibilité de quitter le programme grâce au menu d'options.

Exercice 2 – Compte bancaire

1° Créez une classe `Person` avec un seul attribut `name`, un accesseur pour pouvoir accéder à sa valeur et un constructeur pour l'initialiser.

2° Créez une classe `BankAccount` avec les attributs `holder` (titulaire) du type `Person` et `balance` (solde) du type `double`.
- Ajoutez un constructeur pour initialiser les attributs. Le solde doit être plus grand ou égal à 0.
- Ajoutez une méthode `deposit` (déposer) qui ajoute une somme d'argent au solde. Le montant déposé doit être positif.
- Ajoutez une méthode `withdraw` (retirer) qui va déduire un montant du solde actuel dans le cas où il reste assez d'argent. Le montant à déduire doit être positif. La méthode retourne une valeur `booléenne` qui indique le succès ou l'échec de la transaction. Quand il n'y a pas suffisamment d'argent un message d'erreur est affiché.
- Ajoutez une méthode `printBalance` qui affiche le solde actuel dans la console..
- Ajoutez une méthode `transfer` (virement) qui prend un objet du type `BankAccount` et un montant d'argent en paramètre. Le virement vers le compte du bénéficiaire va avoir lieu seulement si le montant précisé peut être déduit du compte du donneur d'ordre.

3° Ecrivez un programme principal où vous créez deux comptes bancaires et testez les méthodes `deposit`, `withdraw` et `transfer`.

Exercice 3 – Date & Heure

1° Créez une classe `Date` avec les attributs `day`, `month` et `year`.
- Ajoutez une méthode `isLeapYear` qui retourne vrai si l'année stockée dans l'attribut `year` est une année bissextile.
- Ajoutez une méthode `daysInMonth` qui retourne le nombre de jours du mois stocké dans `month`.
- Le constructeur de la classe initialise la valeur des attributs avec les valeurs passées en paramètre dont on vérifie la validité. Si les paramètres ne sont pas valides ils seront adaptés pour qu'ils représentent une date correcte.

- Ajoutez une méthode `advance` qui avance la date actuelle d'un jour.
- Ajoutez une méthode `format(boolean US, String delimiter)` qui retourne la date formatée en tant que `String`. Utilisez des zéros en tête des jours et mois plus petit que 10. Le string `delimiter` sépare les trois parties. Si `US` est `vrai` le mois précède le jour dans la date formatée.

2° Créez une classe `Time` avec les attributs `hours`, `minutes` et `seconds`.

- Le constructeur de la classe initialise la valeur des attributs avec les valeurs passées en paramètre dont on vérifie la validité. Si les paramètres ne sont pas valides ils seront adaptés pour qu'ils représentent un temps correct.
- Ajoutez une méthode `tick` qui avance l'heure actuelle d'une seconde (cf. TP 2 - Exercice 7). La méthode retourne une valeur `booléenne` qui indique si un nouveau jour a commencé. Notez que les heures dans cet exercice auront toujours une valeur entre 0 et 23.
- Ajoutez une méthode `format(boolean US)` qui retourne l'heure formatée en tant que `String`. Utilisez des zéros en tête des heures, minutes et secondes qui sont plus petites que 10. Le format est `hh:mm:ss`. Si `US` est `vrai`,les heures seront formatées avec des valeurs entre 1 et 12 et le suffixe AM ou PM sera ajouté à la fin.
- Ajoutez des méthodes `secondsSinceMidnight` et `secondsUntilMidnight` qui retournent le nombre de secondes écoulées depuis minuit et le nombre de secondes jusqu'à minuit.

3° Créez une classe `DateTime` qui a un attribut du type `Date` et un autre du type `Time`.

- Ajoutez une méthode `tick` qui avance l'heure d'une seconde et si nécessaire avance la date d'un jour.
- Ajoutez une méthode `print(boolean US, String delimiter)` qui utilise les méthodes `format` des classes `Date` et `Time` pour afficher la date et l'heure actuelle dans la console.

4° Ecrivez un programme principal pour tester l'implantation de ces classes.

Exercice 4 – Cadeaux promotionnels

Jean Serien possède 3 magasins au Luxembourg. Pour célébrer le 50ème anniversaire de sa chaîne de magasins il décide de donner des cadeaux à ses clients de façon aléatoire quand ils achètent des produits dans un de ses magasins (La chance augmente si le produit est plus cher). Comme ses magasins s'étendent sur tout le Luxembourg, il veut garantir que les clients de chaque région ont une chance de gagner un cadeau. Ecrivez un programme qui aide Jean Serien à gérer les cadeaux dans ses magasins.

1° Créez une classe `Item` avec les attributs `price` et un constructeur qui initialise la valeur du prix. Ecrivez aussi un accesseur pour cet attribut.

2° Creez une classe `Shop` (magasin) avec l'attribut `localNumberOfGiveaways` (nombre de cadeaux par magasin) et l'attribut de classe `maxNumberOfGiveaways` (nombre total de cadeaux).

- Ecrivez un constructeur qui initialise la valeur de `localNumberOfGiveaways` avec une valeur qu'il reçoit en paramètre.
- Ajoutez une méthode `buy(Item item)` qui:
  - affiche le prix du produit acheté dans la console
  - affiche un message approprié dans la console si le magasin n'a plus de cadeaux en stock ou s'il n'y a plus de cadeaux disponibles au niveau national.
  - attribut un cadeau aléatoirement s'il y a encore des cadeaux disponibles (en total et dans le magasin lui-même). Pour les produits de moins de 20 € la chance de gagner un cadeau sera de 2% , pour ceux entre 20 € et 100 € la chance de gagner sera de 5%, et pour les produits plus chères que 100 € la chance de gagner sera de 10%. Un message sera affiché dans la console si le client a gagné un cadeau. N'oubliez pas d'actualiser le nombre de cadeaux restants.

3° Ecrivez un programme principale qui lit le nombre total de cadeaux, qui crée 3 magasins et répartit les cadeaux uniformément parmi eux. Le programme va alors simuler des clients qui achètent des produits avec des prix entre 0 € et 120 € (prix aléatoire) dans les 3 magasins tant qu'il y a encore des cadeaux.

Part II – Questions théoriques sur la programmation orientée objet

Name: ..... **Student Number:** .....

Exercice 5 – It's a kind of magic ...

Dans le TP 1 - Exercice 4, vous avez rencontré pour la première fois la classe `Scanner` qui vous permet de lire des données d'entrée de la console dans un programme Java.

Les lignes de codes suivantes ressemblaient un peu à de la magie:

```
1  Scanner scanner = new Scanner(System.in);
2  int n = scanner.nextInt();
```

Après le cours de vendredi ça ne devrait plus être de la magie:

1° Qu'est-ce qu'est `Scanner`? .....

2° Qu'est-ce qu'est `scanner`? .....

3° Quel est l'utilité du mot clé `new`? .....

4° Qu'est-ce qu'est `System.in` dans ce contexte? .....

5° Qu'est-ce qu'est `nextInt()`? .....

6° Pourquoi est-ce qu'on peut être sûr que `scanner.nextInt()` retourne bien une valeur du type `int`? .....

Exercice 6 – Questions rapides

Complétez les pharses suivantes:

1° Le mot clé ..... peut être utilisé dans une méthode pour passer la valeur d'une expression à la méthode appelante.

2° Le mot clé ..... indique qu'une méthode ne retourne pas de valeur.

3° Le mot clé ..demande de la mémoire du système pour stocker un objet, puis appelle le ..correspondant pour initialiser l'objet.

4° Chaque paramètre est composé d'un ..... et d'un .....

5° Une variable connue seulement dans la méthode où elle est déclarée est une variable .....

6° Une variable ..... contient de l'information qui est partagée par toutes les instances d'une classe.

Exercice 7 – Vrai ou faux?

Indiquez si les affirmations suivantes sont vraies ou fausses. Si elles sont fausses, expliquez pourquoi.

1° Des parenthèses vides à la fin d'un nom de méthode indiquent que la méthode n'a besoin d'aucun paramètre pour accomplir sa tâche. .....

2° Une méthode peut être invoquée sur une variable de type primitif. .....

3° Des variables déclarées dans le corps d'une méthode sont des *varables d'instances* et peuvent être utilisées dans toutes les méthodes de la classe. .....

4° Chaque classe contenant `public static void main(String[] args)` peut être utilisée pour exécuter une application Java. .....

Exercice 8 – Variables dans la nature

```
1  public class Employee {
2
3    static int countEmployees = 0;
4
5    int employeeID;
6    String name;
7    double salary;
8
9    public Employee(String name, double salary) {
10     this.employeeID = ++countEmployees;
11     this.name = name;
12     this.salary = salary;
13   }
14
15   public double calculateSalaryGrowth(int years) {
16     double annualGrowth = 0.005;
17     double futureSalary = salary;
18
19     for(int i = 0; i < years; i++) {
20       futureSalary *= (1 + annualGrowth);
21     }
22
23     return futureSalary;
24   }
25 }
```

Analysez le code ci-dessus et répondez aux questions suivantes:

1° Quel est la portée de :

a) `employeeID`, `name`, `salary` (lignes 5-7) .....

b) `years` (ligne 15) .....

c) `annualGrowth`, `futureSalary` (lignes 16-17) .....

d) `i` (ligne 19) .....

2° Si 3 instances du type `Employee` sont créées, quel sera la valeur de leurs attributs `employeeID` et `countEmployees` (après que toutes les trois aient été créées)? Expliquez. .....