Développement informatique avancé orienté application - TP

Comparaison d'objets, méthodes et variables de classe et API Java

Virginie Van den Schrieck

Ce TP se base sur les classes réalisées pendant les TPs précédents. Si vous n'avez pas fini ce dernier, mettez-vous en ordre au plus vite, et soyez sûrs de rattraper votre retard avant la prochaine séance !

Les notions illustrées par les exercices ci-dessous sont expliquées dans les sections 2.2, 2.3 et 2.4 du site INGInious du cours : Référez-vous à ce texte pour la compréhension théorique de ces notions.

1 Comparaison d'objets

Pour chacune des classes que vous avez écrites jusqu'ici, documentez, testez et implémentez :

— une méthode `equals()`
— une méhtode `compareTo()`, compatible avec la méthode `equals()`.

Les classes à compléter sont :

— La classe Personne. Les personnes devront être comparées sur base de leur nom et de leur prénom, par ordre alphabétique (conseil : la classe String peut vous être utile).
— La classe Calculatrice. Deux calculatrices sont dites équivalentes si elles affichent la même valeur courante.
— La classe Etudiant. Deux étudiants sont égaux s'ils ont le même matricule.
— La classe Date
— La classe Livre, qui sera Comparable sur base de son ISBN.
— La classe IP

Profitez-en pour vérifier que vos classes sont complètes, et disposent bien :

— d'une spécification javadoc complète et précise
— de constructeurs ad-hoc
— d'une méthode toString()
— d'accesseurs et de mutateurs (+ respect de l'encapsulation)

2 Variables et méthodes de classe

Nous allons à présent mettre en pratique le mot-clé `static` au travers de la définition de variables et méthodes de classes.

2.1 Classe Etudiant

Dans la classe Etudiant, ajoutez une variable statique `nbEtudiants` qui est mise à jour à la création d'un nouvel objet. Utilisez cette variable statique pour générer automatiquement un numéro de matricule unique à chaque nouvel étudiant.

Ecrivez une méthode getNbEtudiants(), qui permet d'obtenir le nombre d'étudiants créés depuis le début de l'exécution du programme. N'oubliez pas de tester ces nouvelles fonctionnalités avec des tests JUnit ad-hoc.

3 Utilisation de l'API Java

3.1 Utilisation de la classe String

1. Reprenez la classe Date que vous avez créée plus tôt. Ajouter un constructeur qui prend un String en paramètre, avec la structure `dd/MM/YYYY` (par ex. 10/12/2014).
2. De même, dans la classe IP, rajoutez un constructeur qui permet de créer un objet IP sur base de sa représentation décimale pointée (ex : `10.0.2.3`).

3.2 Utilisation de la classe Math

Créez une classe Cercle dont les objets seront caractérisés par leur rayon. Grâce à la classe Math, ajoutez une méthode à la classe Cercle qui calcule l'aire et une autre qui calcule le périmètre.

N'oubliez pas les éléments habituels de toute classe bien écrite ! Ajoutez également les méthodes permettant de comparer deux cercles sur base de leur rayon.

3.3 Utilisation des objets Date de Java

Dans la classe Etudiant, remplacez l'utilisation de votre classe Date par les dates natives de Java. Documentez, testez et implémentez également une méthode `age()`.

3.4 Entrées/Sorties

1. Dans la méthode `main` d'une classe de test quelconque, utilisez la classe `Scanner` pour demander à l'utilisateur une adresse IP, et créez

l'objet IP correspondant. Même chose pour un objet Date.

2. Créez un fichier texte dont chaque ligne comporte trois éléments par ligne : le nom, le prénom et le matricule d'un étudiant, séparés chaque fois par un point-virgule. Créez ensuite une méthode `main` dans une classe de test, qui lit le fichier et crée les objets `Etudiant` correspondant.