UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Licence 1 INFO
INFO0202

TP n°3

Gestion des exceptions - Membre d'instance / de classe - Enumération

1 Les Toutous [Utilisation des exceptions]

Un toutou est décrit par son nom, la couleur de sa robe et son nombre de puces. Son nom est une chaîne de caractères, non vide. La couleur de sa robe est un énumération (BLANCHE, SABLE, MARRON, CHOCOLAT, NOIRE). Son nombre de puces est nécessairement positif. Cette classe, ainsi qu'une classe de test, vous est fournie dans les documents du TP3 sur le Moodle de INFO0202.

Nous souhaitons enrichir cette classe avec des exceptions. La classe `Toutou` doit lever des exceptions de type `IllegalArgumentException` lorsque le nom du `Toutou` est *null* ou une chaîne vide, ou lorsque le nombre de puces est strictement négatif.

1. Testez de compiler et exécuter les différentes classes.
2. Modifiez la classe `Toutou` en levant des exceptions de type `IllegalArgumentException` lorsque c'est nécessaire.
3. Testez cette nouvelle classe avec la classe de test fournie. Cela fonctionne-t-il ?
4. Changez le type de l'exception pour `IOException`[1]. Testez à nouveau la gestion ou non des exceptions. Quels changements remarquez-vous ? Quelle en est la raison ?
5. Modifiez la classe de test pour qu'elle fonctionne. Aurait-on dû faire la même chose pour le type `IllegalArgumentException` ?

2 Le distributeur de *Kawa*

Dans cet exercice, il ne vous est pas demandé de gérer les erreurs avec les exceptions, vous pourrez le faire par la suite à la maison pour vous entrainer.

Un distributeur de café permet d'obtenir 2 types de café : un café court ou un café long. Les prix sont fixés au départ, communs pour tous les distributeurs de café. Il est possible de les modifier par la suite.

Dans un distributeur de café, il faut mettre des dosettes. Les mêmes dosettes sont utilisées pour les cafés courts et longs. Le support à dosettes d'un distributeur possède une capacité maximale qui est spécifiée lors de la construction du distributeur.

Dans le distributeur, il y a une cagnotte qui contient l'argent des usagers. C'est une cagnotte magique car elle possède toujours suffisamment d'argent pour rendre la monnaie.

Nous supposons que tous les distributeurs possèdent un réservoir d'eau. À chaque fois qu'un café est commandé, le réservoir d'eau est mis à jour : 10cl pour un café court et 25cl pour un café long. Bien

1. N'oubliez pas l'import de `java.io.IOException`

entendu, afin d'éviter d'arnaquer le client, s'il n'y a plus de dosette ni assez d'eau, le café n'est pas servi et l'argent est rendu.

Nous souhaitons écrire une classe `DistributeurCafe` qui implémente l'interface `IDistributeur` (le fichier `IDistributeur.java` est fourni sur le moodle), sachant que :

- Un distributeur possède une capacité de dosettes ainsi qu'un nombre de dosettes.
- La capacité de dosettes est fixée au moment de la construction soit par une valeur par défaut (10) soit par une valeur quelconque.
- Un distributeur possède un réservoir d'eau dont la capacité par défaut est de 2L mais qui peut être spécifiée lors de la construction.
- Un distributeur possède une cagnotte représentée par une somme d'argent (à 0 au moment de la construction).

1. Créez le fichier qui contiendra la classe, spécifiez les attributs et créez les trois constructeurs.
2. Ajoutez un *getter* pour chaque attribut.
3. Écrivez les méthodes `afficher` et `toString`.
4. Écrivez une méthode `boireCafeCourt` prenant en paramètre une somme donnée par l'utilisateur et retournant la monnaie. Le distributeur sert un café court (de 10cl).
5. Écrivez une méthode `boireCafeLong` permettant de servir un café long (de 25cl).
6. Écrivez des méthodes pour consulter et modifier le prix des cafés (communs pour tous les distributeurs).
7. Ajoutez la méthode `augmenter` qui prend en paramètre une augmentation sous forme d'un pourcentage et qui permet d'augmenter tous les prix suivant ce pourcentage.
8. Écrivez une classe `TestDistributeur` permettant de tester la classe précédente :
   - Créez un distributeur.
   - Faites un menu en demandant à l'utilisateur de choisir entre :
     (1) Afficher le distributeur
     (2) Boire un café court (demander la monnaie à insérer)
     (3) Boire un café long (demander la monnaie à insérer)
     (4) Ajouter des dosettes (demander le nombre de dosettes à remettre)
     (5) Remplir le réservoir (demander la quantité d'eau à ajouter)
     (6) Permettre de récupérer, modifier ou augmenter les prix

La suite des exercices correspond à du travail personnel à réaliser à la maison, vous pouvez contacter votre enseignant par mail si vous avez des questions.

3 Les cercles le retour

Reprenons la classe `Cercle` réalisée à l'exercice 4 du TP1 : pour rappel voici le diagramme de classe qui a été réalisé. Nous souhaitons gérer le problème du rayon négatif avec les exceptions.

Cercle
- abscisse : double - ordonnee : double - rayon : double
+ getAbscisse() : double + getOrdonnee() : double + getRayon() : double + setAbscisse(double) : - + setOrdonnee(double) : - + setRayon(double) : - + toString() : String

1. A quelles méthodes devons-nous apporter des modifications ? et quels types de modifications ?
2. Sachant que nous souhaitons utiliser une `ArithmeticException` avec une message personnalisé, réécrivez le code de ces méthodes.
3. Et qu'est-ce que cela change dans une classe de test ? Réécrivez une classe de test qui permettra la gestion correcte des exceptions possiblement engendrées.

4 Senseo

Cette fois il serait bien de gérer les erreurs avec les exceptions pour vous entrainer.

Une cafetière `Senseo` ne prend qu'une seule dosette à la fois. Le réservoir est obligatoirement de 1L. Et pour boire le café, il n'y a pas besoin d'argent. Si de l'argent est donné, il est rendu intégralement.

1. Écrivez la classe `Senseo` qui implémente l'interface `IDistributeur`.
2. Modifiez la classe de test.