IS2A4 – PPO – T.P. 3b

Résumé

L'objectif de ce TP est de simuler un supermarché tout en utilisant des exceptions. Dans le supermarché, on va avoir un ensemble de rayons. Dans chaque rayon, on va voir un ensemble d'articles et chaque article va avoir un nom, un type d'article et un prix.

N'hésitez pas à consulter la documentation en ligne sur le site d'oracle : `https://docs.oracle.com/javase/8/docs/api/` (regardez la version correspondant la version de java que vous utilisez).

I Préparation du TP (5 min)

I.1. Créer un dossier spécifique pour les TP de PPO qui vous appellerez `TP_PPO` par exemple.

I.2. Dans ce dossier, créer un dossier `TP3b`. C'est dans ce dossier que vous travaillerez aujourd'hui. Durant les prochaines séances de TP, essayez de garder cette pratique.

I.3. Téléchargez sur le Moodle du cours le fichier `TP3b_Base.zip` dans `TP3b` et dézippez le dans votre dossier `TP3b`.

II TypeArticle (25 min)

II.1. Complétez dans le fichier `lib/TypeArticle.java`, le constructeur pour qu'il n'accepte que 4 valeurs possibles : `alimentaire`, `textile`, `electromenager` et `cosmetique` et envoie comme message "Problème de catégorie !".

II.2. Testez votre code en compilant et exécutant `TesteTypeArticle.java`.

II.3. Créez une exception `ProblemeCategorieException` qui hérite de la classe `Exception` dans le dossier `exceptions/`. N'oubliez pas d'ajouter `package exceptions;` au début du fichier.

II.4. Provoquez l'exception `ProblemeCategorieException` dans votre classe `TypeArticle` lorsque vous avez un problème de catégorie. (N'oubliez pas d'importer les classes du package `exceptions` dans votre fichier de classe)

II.5. Modifiez le code du fichier `TesteTypeArticle.java` pour capturer les exceptions.

III Article (20 min)

III.1. Complétez les méthodes de la classe `Article` dans le package `lib` et ajouter une propagation de l'exception `ProblemeCategorieException` au constructeur.

III.2. Testez votre code en compilant et exécutant `TesteArticle.java`.

III.3. Comme "Si on ne sait pas ce que c'est, c'est que ça doit bien se manger !", modifiez le constructeur de la classe `Article` en ne propagant plus l'exception mais en la capturant. (Si le type n'est pas compréhensible, prenez par défaut le type `alimentaire`)

III.4. Modifiez le code de `TesteArticle.java` en enlevant la capture.

IV Rayon (20 min)

IV.1. Construisez une classe `Rayon` qui a comme variable d'instance un tableau d'`Article` et une nombre maximum d'article par rayon et un constructeur qui a comme seul paramêtre le nombre d'article maximum.

IV.2. Ajoutez les méthodes suivantes :

- `getAvailPos()` qui donne la première position du tableau qui est `null`.
- `add(Article a,int i)` qui ajoute l'élement `a` à la position `i`.
- `add(Article a)` qui ajoute l'élement `a` à la première position du tableau qui est `null`. (Conseil : utiliser les deux méthodes précédentes)
- `remove(int i)` qui supprime la position `i` du tableau (`T[i] = null`)

IV.3. Complilez et exécutez le fichier `TesteRayon.java` pour vérifier votre implémentation.

IV.4. Complilez et exécutez le fichier `TesteRayonArrayPointer.java`. Que se passe-t-il ? Proposez une solution pour le fichier `Rayon.java` à l'encadrant et implémentez la après validation.

V Affichage (20 min)

V.1. Créez une nouvelle exception `NomVideException` dans le package `exceptions` et provoquez cette exception si le nom donné au constructeur de la classe `Article` est vide.

V.2. Créez une nouvelle exception `NomTropLongException` dans le package `exceptions` qui hérite de `NomVideException`.

V.3. Provoquez ces deux exceptions dans la méthode `add(Article a)` de `Rayon` et capturez les différentes exceptions dans `TesteRayon.java` que vous adapterez.

V.4. Montrez votre code à l'encadrant.

`Temps total du sujet : 1 heures et 30 minutes`