
Exercice I (Exceptions et UtilMath)

1. Reprenez la classe UtilMath vue dans le TD1 (Exercice I).
 - (a) Si le paramètre de la méthode fact(**int** n) est négatif, levez une IllegalArgumentException avec un message d'erreur informatif.
 - (b) De la même façon, gérez les erreurs possibles dans les méthodes comb(**int** n, **int** p) (si $p > n$) et puissance(**int** n, **int** m) (si $m < 0$).
2. Les modifications apportées aux méthodes de la classe UtilMath doivent être impactées sur le programme qui les utilise (TD1, Exercice VI). Faites les modifications nécessaires pour que l'utilisateur soit informé lors d'une erreur, et que le programme ne s'arrête pas brutalement.
3. Faites les modifications nécessaires pour que l'utilisateur soit informé de l'erreur s'il tape autre chose que des nombres entiers, et que le programme ne s'arrête pas brutalement.

Ce n'est pas précisé dans le reste de l'énoncé, mais pour les exercices suivants, pensez à gérer correctement les exceptions!

- **Fermeture des flux d'entrées-sorties**
- **Affichage d'un message d'erreur compréhensible pour l'utilisateur**

Exercice II (Quelques commandes UNIX)

On souhaite développer des programmes qui correspondent à certaines commandes UNIX :

- grep sert à filtrer un texte en n'affichant que les lignes qui contiennent un certain motif;
 - wc compte le nombre de lignes, de mots et de caractères dans un texte;
 - cp permet de copier un fichier.
1. Écrivez un programme qui prend en entrée le nom d'un fichier texte et une chaîne de caractères,¹ et qui affiche sur la console toutes les lignes du fichier qui contiennent la chaîne de caractères donnée en entrée.
Indice : vous pouvez créer une classe GrepReader qui hérite de BufferedReader, et qui redéfinit la méthode readLine() de manière adéquate.
 2. Écrivez un programme qui prend en entrée le nom d'un fichier texte, et qui affiche sur la console le nombre de lignes, de mots, et de caractères dans ce fichier, suivi du nom du fichier.
On prend en compte les caractères '\n', '\t' et '_' pour les séparateurs de mots.
 3. Écrivez un programme qui prend en entrée deux noms de fichiers² et copie le contenu du premier dans le second. Si le second fichier existe déjà, son contenu initial est écrasé, dans le cas contraire le fichier est créé.

1. Pour qu'un programme lancé dans Eclipse utilise les paramètres de la ligne de commande, il faut les indiquer dans le menu Run → Run Configurations → Arguments → Program arguments.

2. Il s'agit ici de fichiers quelconques, pas forcément de fichiers de texte.

Exercice III (Sauvegarde de répertoire)

On réutilise la classe `RepertoireAmeliore` utilisée précédemment. On souhaite pouvoir enregistrer un répertoire dans un fichier texte, ou au contraire charger un répertoire à partir d'un fichier texte. Le fichier est composé d'autant de lignes que de personnes dans le répertoire :

- `proprietaire(XXX,YYY,ZZZ)` pour la description du propriétaire du répertoire, avec XXX le prénom, YYY le nom et ZZZ le numéro de téléphone;
 - `contact(XXX,YYY,ZZZ)` pour la description des contacts, avec XXX le prénom, YYY le nom et ZZZ le numéro de téléphone.
1. Écrivez une classe `ParserRepertoire` qui permet de lire un fichier texte décrivant un répertoire, et de l'utiliser pour construire une instance de `RepertoireAmeliore`.
 2. Écrivez une classe `SauvegardeRepertoire` qui permet d'enregistrer un `RepertoireAmeliore` dans un fichier texte.
 3. Écrivez un programme qui permet à l'utilisateur de :
 - créer un répertoire lui appartenant;
 - ajouter des contacts dans son répertoire;
 - faire des recherches dans son répertoire (en fonction du prénom et du nom, ou en fonction du numéro de téléphone);
 - d'afficher tous les contacts;
 - de sauvegarder le répertoire dans un fichier texte;
 - de charger un répertoire depuis un fichier texte.

Si l'utilisateur charge un répertoire alors qu'il en possédait déjà un, l'ancien répertoire est supprimé.

Exercice IV (Programme télé)

On reprend les classes définies dans le TD 3 sur la grille des programmes d'une chaîne de télévision. On représente un ensemble de programmes par un fichier texte dont les lignes respectent les contraintes suivantes :

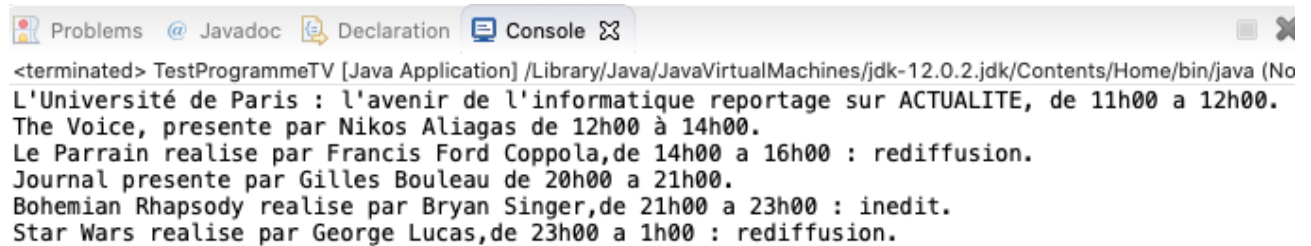
- `divertissement(XXX,YYY,H)` pour les émissions de divertissement, avec XXX le nom du présentateur, YYY le nom de l'émission, et H l'heure de début;
- `journal(XXX,H1,H2)` pour le journal télévisé, avec XXX le nom du présentateur, et H1, H2 pour les heures de début et de fin;
- `reportage(XXX,YYY,H1,H2)` pour les reportages, avec XXX le thème YYY le nom, et H1, H2 pour les heures de début et de fin;
- `fiction(XXX,YYY,ZZZ,H1,H2)` pour les fictions, avec XXX le titre de la fiction, YYY le nom de son réalisateur, ZZZ (qui vaut T ou F) indique si c'est une fiction ou non, et H1, H2 pour les heures de début et de fin.

Créez un programme qui lit un tel fichier texte, et affiche les programmes classés par heure de début.³ Voici un exemple de fichier texte :

3. Pour faciliter l'affichage dans l'ordre chronologique, il peut être nécessaire de modifier légèrement la classe `programme`...

```
divertissement(Nikos Aliagas,The Voice,12)
journal(Gilles Bouleau,20,21)
reportage(ACTUALITE,L'Université de Paris : l'avenir de l'informatique,11,12)
fiction(Le Parrain,Francis Ford Coppola,T,14,16)
fiction(Bohemian Rhapsody,Bryan Singer,F,21,23)
fiction(Star Wars,George Lucas,T,23,01)
```

Et voici l'exécution du programme pour ce fichier :



```
<terminated> TestProgrammeTV [Java Application] /Library/Java/JavaVirtualMachines/jdk-12.0.2.jdk/Contents/Home/bin/java (No
L'Université de Paris : l'avenir de l'informatique reportage sur ACTUALITE, de 11h00 a 12h00.
The Voice, presente par Nikos Aliagas de 12h00 à 14h00.
Le Parrain realise par Francis Ford Coppola,de 14h00 a 16h00 : rediffusion.
Journal presente par Gilles Bouleau de 20h00 a 21h00.
Bohemian Rhapsody realise par Bryan Singer,de 21h00 a 23h00 : inedit.
Star Wars realise par George Lucas,de 23h00 a 1h00 : rediffusion.
```