Les exceptions en Java

R2.03 Qualité de développement

IUT 45 — Informatique

2023-2024

Résumé

Les exceptions sont l'outil proposé par Java —et d'autres langages de programmation— pour permettre de réagir à des situations peu courantes dans la vie d'un programme. Il peut s'agir d'un problème lié à des données externes, d'une ressource momentanément inaccessible ou autres. La gestion des execptions se fait en deux parties : une méthode confrontée à une situation rare peut la signaler en *levant une exception*, ce qui interrompt son exécution ; une méthode qui s'attend à ce que cette situation puisse se présenter au cours de l'exécution d'une partie de son code et qui sait comment y réagir peut *rattraper l'exception*, que celle-ci soit levée dans une fonctions qu'elle appelle directement ou indirectement.

Dans cette feuille, vous apprendrez à manier les exceptions en Java en ajoutant une interface basique au week-end entre amis.

1 Une interface en console pour le week-end entre amis

Attention

Dans ce TP, nous allons travailler sur une interace utilisateurs en mode console. Tester une interface demande des techniques de test particulières que nous n'aborderons pas aujourd'hui. Aujourd'hui, pas de tests, pas de Junit. Tant pis, il y aura des bugs ! C'est pourquoi on cherche généralement à avoir le MINIMUM de code dans les interfaces.

Nous commençons l'implémentation d'une interface en mode console pour *le week-end entre amis* avec la classe `UIWeekEnd`. L'abréviation **UI** signifie *User Interface*, elle désigne les interfaces utilisateurs.

1.1 Prise en main

Question 1 Récupérer le fichier `UIWeekEnd.java` sur Célène.

Ce fichier contient une classe `AppWeekEnd` qui implémente un menu permettant d'accéder aux diverses fonctionnalités du Week-End entre amis via la méthode `menu()`. Cette classe est instanciée par la méthode `main()`.

Question 2 Compléter la méthode `aurevoir()` pour afficher un message d'au-revoir quand on quitte l'application.

Question 3 Dans la méthode `menu()`, quelle est l'appel de méthode qui permet de récupérer l'entrée saisie l'utilisateur?

Question 4 Pourquoi conserve-t-on à la fois `commande` et `commande_brute` dans la méthode `menu`?

1.2 Affichages basiques

Pour l'instant, notre application propose uniquement la commande QUITTER. Ajoutons d'abord au menu des commandes d'affichage du contenu du week-end. Jusqu'à nouvel ordre, faute de pouvoir faire remplir un week-end à l'utilisateur, nous allons réutiliser le week-end qui a été créer dans les tests. Nous remédierons à cette situation à la question 19

Question 5 *À l'aide d'un copier-coller* honteux mais nécessaire, initialiser `leWeekEnd` dans le `main()` avec quelques personnes et dépenses.

Question 6 Ajouter les commandes suivantes :

- «p» pour afficher les personnes du week-end.
- «d» pour afficher les dépenses du week-end.
- «t» pour afficher le total des dépenses du week-end.
- «m» pour afficher la dépense moyenne par personne pour le week-end.

2 Sélection et modification des personnes et dépenses

2.1 Sélection des personnes

On veut maintenant pouvoir sélectionner une personne pour lui ajouter des dépenses, afficher ses dépenses et son avoir. Pour cela, il faut transformer la commande `p` en un sous-menu.

Voici un premier exemple d'utilisation du sous-menu que l'on veut obtenir : on entre dans le sous-menu, on obtient l'affichage de la liste des personnes et on ressort :

```
Bienvenue! En week-end comme dans la semaine, les bons comptes font les bons amis.
```

```
  Menu principal
-----------------------------
Q: quitter
D: Menu dépenses
P: Menu personnes
T: Total dépenses
M: Moyenne des dépenses
```

p

```
  Menu amis
-----------------------------
L: liste
S: sélection
Q: quitter
```

l

[Pierre Durand , Paul Dupond , Marie Dumond , Anne Dunon]

```
  Menu amis
-----------------------------
L: liste
S: sélection
Q: quitter
```

q

```
  Menu principal
-----------------------------
Q: quitter
D: Menu dépenses
P: Menu personnes
T: Total dépenses
M: Moyenne des dépenses
```

q

Question 7 Ajouter une méthode `void menu_personne()` dans `AppWeekEnd`. Cette méthode sera appelée par la touche `p`. Elle affichera la liste des personnes et proposera les commandes suivantes :

— «l» pour réafficher la liste des personnes

— «q» pour revenir au menu principal.

Question 8 Ajouter un attribut `private Personne personne_selectionnee` à la classe `AppWeekEnd`. Cet attribut vaut `null` au départ, sa valeur sera modifiée quand l'utilisateur sélectionne une personne.

Pour sélectionner une personne, on entre dans le menu des personnes, on tape la commande «s», puis le numéro de cette personne dans la liste. Voici un exemple d'utilisation de ce mécanisme.

```
Bienvenue! En week-end comme dans la semaine, les bons comptes font les bons amis.
```

```
  Menu principal
-----------------
Q: quitter
D: Menu dépenses
P: Menu personnes
T: Total dépenses
M: Moyenne des dépenses
```

```
p
```

```
  Menu amis
-----------
L: liste
S: sélection
Q: quitter
```

```
l
[Pierre Durand , Paul Dupond , Marie Dumond , Anne Dunon ]
```

```
  Menu amis
-----------
L: liste
S: sélection
Q: quitter
```

```
s
Sélectionner la personne numéro combien?
2
Vous avez sélectionné Marie Dumond
```

```
  Menu amis
```

```
L: liste
S: sélection
Q: quitter
```

```
q
```

```
  Menu principal
```

```
Q: quitter
D: Menu dépenses
P: Menu personnes
T: Total dépenses
M: Moyenne des dépenses
```

```
q
```

Pour implémenter ce mécanisme, il faut pouvoir transformer la chaîne de caractères saisie par l'utilisateur en un entier. Java propose à cet effet une méthode de la classe `Integer` : la méthode *statique* `Integer.parseInt`.

Voici un exemple d'utilisation de `Integer.parseInt`, donné sous forme de test JUnit.

```
String s = "32";
int x = Integer.parseInt(s);
int y = x * 2;
assertEquals("64", y);
```

Question 9 À l'aide de la méthode `Integer.parseInt` mplémenter une première version de la sélection d'une personne dans le sous-menu des personnes.

Si la chaîne de caractères passée à `Integer.parseInt` ne représente pas un entier, elle ne peut pas renvoyer de résultat sensé. Au lieu de renvoyer un entier qui n'a pas de sens, elle *lève une exception* pour signaler qu'une situation particulière s'est produite. En consultant la documentation de la méthode `Integer.parseInt` sur `https://docs.oracle.com/javase/7/docs/api/java/lang/Integer.html#parseInt(java.lang.String)`, on observe que sa déclaration est la suivante :

```
public static int parseInt(String s)
                    throws NumberFormatException
```

Le mot-clé `throws` nous indique que dans certaines circonstances particulière cette méthode est susceptible de refuser de produire un résultat, et de lever à la place l'exception `NumberFormatException`. On constate par exemple que l'exécution du code suivant est arrêtée à la ligne 4, avec un message d'erreur qui indique que l'exception `NumberFormatException` a été lancée.

```
String s = "trente-deux";
System.out.println("Cette ligne et la suivante sont bien exécutées");
System.out.println("Valeur de s: " + s);
int x = Integer.parseInt(s);
System.out.println("Cette ligne et la suivante ne seront jamais exécutées");
System.out.println("Valeur de x plus un:" + (x+1));
```

Question 10 Que se passe-t-il dans votre programme week-end si l'utilisateur donne une réponse qui n'est pas un nombre pour la sélection d'une personne? Comment retrouver la provenance de l'exception à partir du message d'erreur de Java?

À retenir

Une méthode marquée `throws ExceptionTruc` est susceptible dans certaines circonstances de *lever l'exception* `ExceptionTruc` plutôt que de renvoyer un résultat. Si une exception est levée, l'exécution du programme est interrompue. Par défaut, un message d'erreur est affiché qui indique quel appel de fonction a provoqué l'exécution.

Il n'est pas souhaitable d'interrompre le fonctionnement du programme dès qu'il y a une erreur de saisie! Pour éviter cette avanie, il faut que la méthode qui gère le sous-menu de sélection de la personne repère qu'il y a un problème de `NumberFormatException` et agisse en conséquence. Pour cela, il faut utiliser un *bloc try / catch*. On place le code susceptible de provoquer l'exception dans un bloc entre accolades précédé du mot-clé `try`. Dans un second bloc précédé de `catch`, on place le code à exécuter dans le cas où le bloc `try` provoque l'exception `NumberFormatException`. Voici l'adaptation de l'exemple de code précédent avec un `try...catch`.

```
String s = "trente-deux";
System.out.println("Cette ligne et la suivante sont bien exécutées");
System.out.println("Valeur de s: " + s);
try {
  int x = Integer.parseInt(s);
  System.out.println("Cette ligne et la suivante ne seront jamais exécutées");
  System.out.println("Valeur de x plus un:" + (x+1));
}
catch (java.lang.NumberFormatException e) {
  System.out.println("Cette ligne est exécutée car " + s +
                                    " ne représente pas un entier");
}
System.out.println("Cette ligne sera exécutée que l'exception soit levée ou non");
```

Question 11 À l'aide d'un bloc `try...catch`, rendre le menu de sélection des personnes robustes aux erreurs de saisie.

À retenir

On peut utiliser la construction `try...catch` pour exprimer le comportement à adopter dans le cas où une exception donnée est levée par un bloc de code.

Question 12 Ajouter les commandes suivantes au sous-menu des personnes :

— "T" pour le total des dépenses *de la personne sélectionnée*

— "A" pour afficher l'avoir *de la personne sélectionnée*

Question 13 Que se passe-t-il si les commandes «T» ou «A» de la question 12 sont utilisées alors qu'aucune personne n'est sélectionnée? Faites en sorte que ce scénario ne provoque pas de crash.

Question 14 Ajouter une fonction «+» au sous-menu des personnes pour créer une personne et l'ajouter au week-end.

2.2 Sélection des dépenses

Question 15 Créer de même un sous-menu pour les dépenses, avec les commandes suivantes :

— «l» pour afficher la liste des dépenses

— «q» pour quitter sous le sous-menu et revenir au menu principal

On veut maintenant ajouter une option «+» à ce sous-menu pour permettre d'ajouter une nouvelle dépense au week-end. Pour éviter les erreurs de saisie, on ne veut pas accepter de dépenses qui ne soient pas associées à une personne déjà ajoutée au week-end. Pour cela, **dans les questions suivantes**, nous allons créer une méthode `saisie_dépense` qui va :

— demander à saisir un produit

— demander à saisir un prix (qui doit être un entier)

— afficher les participants et demander à en sélectionner un.

On remarque que le code qui permet la sélection d'un participant existe déjà dans le sous-menu des participants (c'est la commande «s» de ce sous-menu). Pour éviter les redondances, nous allons commencer par extraire ce code dans une méthode `selectionner_participant()`. Cette méthode doit disposer d'un moyen de signaler que la valeur qui a été saisie n'est pas valide. Il faut pour cela *déclarer une nouvelle classe d'exception* `PersonneInconnue` puis ajouter `throws PersonneInconnue` dans la déclaration de `selectionner_participant()`.

Question 16 Ajouter à l'extérieur de la classe `AppWeekEnd` la déclaration suivante :

```
class PersonneInconnue extends Exception {}
```

Il est désormais possible de créer la fonction `selectionner_participant`. Elle est *susceptible* de lever l'exception `PersonneInconnue`, c'est pourquoi sa déclaration est :

```
public void selectionner_participant() throws PersonneInconnue {
  ...
}
```

À l'intérieur de cette fonction, quand on repère que le numéro d'utilisateur sélectionné n'existe pas, on peut interrompre l'exécution avec l'instruction `throw new PersonneInconnue();`. Cette instruction interrompt l'exécution de la fonction `selectionner_participant` et lève l'exception `PersonneInconnue`. Cette exception interrompt toute les fonctions appelantes, jusqu'à celle qui aura un bloc `try` avec un `catch(PersonneInconnue e)`.

Question 17 Écrire la méthode `selectionner_participant`

Question 18 Ajouter la méthode de saisie d'une dépense.

À retenir

Il est possible de déclarer ses propres exceptions avec :

```
class MonException extends Exception {}
```

Le nom de la classe d'exception devra décrire le type d'erreur que représente cette exception.

À retenir

Il est possible de *lever* une exception avec l'instruction suivante :

```
throw new MonException();
```

Cette instruction interrompt immédiatement l'exécution de la méthode en cours et de toutes les méthodes appelantes, jusqu'à trouver, le cas échéant, une méthode avec un bloc `catch` correspondant à `MonException`.

Question 19 Supprimer l'initialisation avec le week-end arbitraire.