Exceptions

Mathieu Bourgais
mathieu.bourgais@insa-rouen.fr

2023

Préambule

Vous pouvez utiliser l'éditeur de texte de votre choix. Vous pouvez aussi utiliser un IDE comme Eclipse, Visual Studio Code ou encore Intellij, qui présente de nombreux avantages (auto-complétion, par exemple). Dans Eclipse, vous devez toujours travailler dans un projet (à créer au début de chaque TP/mini-projet) dans lequel vous définirez vos classes.

Pour rappel, pour compiler et exécuter un code source Java, vous pouvez utiliser votre éditeur favori, ou ouvrir un terminal puis naviguer jusqu'au répertoire désiré, et taper :

— javac monFichier.java
— java monFichier

1 Exercice 1 - Manipuler les exceptions

La gestion des erreurs doit obliger le développeur à prendre en compte dans son code les erreurs importantes, mais aussi lui permettre de créer son propre jeu d'erreurs, en fonction du code produit, et lui permettre de gérer proprement ces erreurs.

Jusqu'alors (en Pascal ou en C par exemple) les erreurs étaient gérées directement via les fonctions. Ces dernières retournaient une valeur (ou un code) indiquant si elles s'étaient bien déroulées, ou si une erreur était survenue. La valeur du code permettait alors d'indiquer le type d'erreur. Par exemple, une fonction multEgyptienne(int a, int b), prenant en paramètre deux entiers positifs, et ayant pour valeur de retour -1, signale clairement un problème lors de l'exécution de la fonction. Cette manière de traiter les erreurs oblige cependant le développeur à mélanger le traitement des erreurs au code du programme, le forçant à vérifier explicitement la valeur de retour de chaque fonction.

En Java, la gestion des erreurs se fait à l'aide d'**exceptions**. Une exception est un évènement exceptionnel, pouvant survenir lors de l'exécution d'un programme ou d'une fonction. L'exception arrête alors le déroulement normal du

programme afin d'entrer dans une branche de traitement spécifique. Les avantages de la gestion des erreurs à l'aide d'exceptions sont multiples. D'une part, cela permet de séparer le code à proprement parler du traitement des erreurs. D'autre part, les exceptions propagent les erreurs à travers la pile des appels, et permettent d'identifier clairement l'instruction provoquant l'exception, et donc d'isoler / corriger les problèmes plus facilement. Enfin, les exceptions permettent également une organisation plus fine des erreurs, notamment en définissant un type d'exception précis pour chaque erreur.

Lever une exception consiste à dérouter le flot d'instructions normal du programme afin de propager ou de traiter cette exception, et l'erreur lui étant liée.

Propager une exception consiste à renvoyer son traitement à la méthode ou instruction appelante. Si une exception est propagée jusqu'à la méthode main, le programme est alors arrêté, et affiche des informations sur l'exception. En Java, pour signaler qu'une méthode peut propager une exception, on utilisera lors de sa déclaration le mot clé **throws**, suivi du (ou des) type(s) d'exception(s) pouvant être propagées (par exemple, **void lireFichier(String f) throws FileNotFoundException)**. Pour lever explicitement l'exception au sein du corps de la méthode, on utilisera alors le mot clé **throw** (par exemple, throw new FileNotFoundException() ;). De plus, en Java, toute exception non contrôlée levée par une méthode est implicitement propagée.

Traiter (ou attraper) une exception consiste à dérouter le flot d'instructions vers une autre partie du programme, spécifiquement dédiée au traitement des erreurs. En Java, on utilisera pour cela un bloc **try ... catch**, où la (ou les) clause(s) catch spécifie(nt) le(s) type(s) d'exception(s) attrapée(s) (par exemple, **catch (FileNotFoundException e))**. Les instructions contenues dans le try correspondent à l'exécution normale du programme, tandis que les instructions contenues dans la (ou les) clause(s) catch correspondent au traitement des erreurs. Attention cependant, une seule clause catch (la première correspondant à l'exception) ne pourra être exécutée.

Compilez et exécutez la classe **TestExceptions** fournie. Que se passe-t-il ? Pourquoi ?

Modifiez la déclaration de la méthode **division** afin d'indiquer explicitement que cette méthode peut propager une exception. Vous ferez attention à spécifier correctement le type de l'exception.

Modifiez le corps de la méthode **division** afin de lever explicitement l'exception avant même de lancer le calcul, si b = 0.

Faites les modifications nécessaires au sein de la méthode main afin de traiter l'exception convenablement. Vous pourrez, par exemple, vous contenter d'afficher un message d'erreur du type "Seul Chuck Norris peut diviser par zero...". Compilez et exécutez de nouveau la classe **TestExceptions**. Que se passe-t-il ? Pourquoi ?

2 Exercice 2 - Créer les exceptions

Java permet également au développeur de créer ses propres exceptions. Les exceptions sont des classes comme les autres, et se manipulent donc comme tel. Pour créer une nouvelle exception, il est nécessaire de dériver une des deux classes **Exception** ou **RuntimeException**. Comme d'habitude, il est alors nécessaire de définir deux constructeurs pour la classe représentant la nouvelle exception : un constructeur sans arguments, et un constructeur prenant en paramètre un String, faisant tous deux simplement appel au constructeur de la classe mère, à l'aide d'un appel à super. L'exception nouvellement créée se manipule alors comme les exceptions de base proposées par Java.

Récupérez l'ensemble des classes et interfaces développées lors du TD3.

Créez une nouvelle exception **DebitImpossible**. Cette exception sera utilisée pour signaler que le débit demandé sur un **CompteCourant** ou sur un **CompteEpargne** est impossible, car le compte n'est pas suffisamment provisionné.

Modifiez vos interfaces et classes **Compte**, **CompteBancaire**, **CompteCourant** et **CompteEpargne**, et plus précisément leurs méthodes **debiter**, pour que ces dernières lèvent une exception de type **DebitImpossible** si le débit n'est pas autorisé, plutôt que d'afficher un simple message d'erreur.

Modifiez la classe **TestBanqueExceptions** pour traiter correctement les exceptions. Ici, le traitement pourra se contenter d'afficher un simple message d'erreur.