Cours Java 13

Les exceptions

**Diapo 2**

La pile des exceptions se parcours du bas vers le haut pour remonter jusqu’à l’erreur.

**Diapo 3**

Avec la javadoc on peut savoir le type d’exceptions que peut lancer une fonction.

**Diapo 5**

L’exception remonte de blocs en blocs et de méthodes en méthodes, si pas attrapée elle remonte jusqu’au main et cela arrêtera le programme.

**Diapo 6**

L’exception créée, c’est un objet. Quand exception rencontrée, on sort de son bloc pour chercher un catch. On remonte de bloc en bloc pour trouver un catch.

**Diapo 7**

Tout ce qui est dans la partie error de l’arbre, ce sont des erreurs que l’on ne peut pas récupérer. Partie droite, on peut faire des try / catch.

RunTimeException, c’est du à une erreur de programmation. On n’essaye pas de les attraper. Les IOException on peut les gérer.

On peut définir ses propres types d’exception (étant une sous classe d’Exception, ou d’une sous-classe d’Exception plus spécialisée).

**Diapo 8**

Methode printStackTrace : renvoie la pile des méthodes jusqu’à l’erreur.

**Diapo 11**

Les sous-classes possèdent juste constructeur appelant la super classe. Son type (= son nom) permet de savoir à quelle type d’exception on a à faire.

**Diapo 12**

Try englobe des instructions susceptibles de créer une exception. Un bloc try est toujours suivi de catch (une fonction qui prend un paramètre particulier). Le bloc catch gère un seul type d’exception.

FileReader travaille sur des fichiers déjà existant. Le petit « e » est l’objet exception qui a été créé lorsque l’erreur apparait.

De façon standard, l’exécution continue après le catch.

**Diapo 13**

Les blocs catch sont exécutés de façon séquentielle. Si une exception est attrapée par un bloc catch, les autres ne sont pas exécutés.

**Diapo 14-17**

Il faut que les blocs catch soient dans l’ordre inverse de la hiérarchie d’héritage. (on peut avoir les logs qui sont créer au niveau du bloc catch, mais pas forcément. Les logs peuvent prendre des erreurs ou des informations, ex : adresse IP xxxxxx connectée à l’heure : xxxxx).

**Diapo 18**

On décide des blocs catch que l’on met. Pas obligé d’attraper toutes les exceptions, celles non attrapées remontent dans les blocs supérieurs. Dans ce cas-là il faut alors throws avec l’exception à transmettre dans la signature de la méthode.

**Diapo 19**

Bloc finally permet d’exécuter du code quel que soit la façon dont on sort du bloc try.

**Diapo 20**

Permet de factoriser le code. Si on ferme le fichier quel que soit la sortie du bloc try alors il est intéressant de créer le bloc finally pour fermer le fichier.

**Diapo 22**

Avec throw c’est une exception contrôlée, que l’on doit signaler dans la signature de la méthode. Toutes les fonctions qui appellent cette méthode, il faut alors aussi mettre la clause throws dans la signature. Bien penser à la doc de l’exception.

**Diapo 25**

Soit throws dans la signature soit try / catch, mais pas les deux en même temps.

**Diapo 28**

Interdit d’ajouter de nouveaux types d’exception (d’une branche différente). Par contre on peut utiliser des exceptions plus spécialisées (sous-classe d’exception déjà dans le throws de la méthode de la classe mère). On peut aussi ne pas avoir d’exception.

Exception standard sous-classe de Error et de RunTimeException, on peut les catch, mais on n’est pas obligé de mettre le throws.

**Diapo 31**

En jaune erreur contrôlée donc throws ou try catch obligatoire. En bleu, pas d’obligation et même pas possible pour la classe Error ou ses sous-classe.

**Diapo 32**

Bloc catch vide c’est à proscrire. Il faut au moins mettre un minimum de code pour se rendre compte qu’il y a un problème.

**Diapo 33-36**

Exemple code similaire à ce que l’on pourrait faire en C. Le code est pollué par les codes d’erreurs. En Java code dans le bloc try et les exceptions dans les catch.

**Diapo 37**

Bloc catch peut attraper plusieurs types d’exceptions ayant aucun lien dans la hiérarchie d’héritage.

Pour gérer plusieurs exceptions dans un catch, elles sont en paramètres du catch et sont séparées par un pipe « | » (se lit comme un ou).

**Diapo 39**

Dans le bloc try on déclare les objets en paramètre du try, séparés par un point-virgule. Quand on arrive à la fin du bloc try ces objets sont fermés. Les objets doivent implémenter l’interface AutoCloseable.

**Diapo 41**

En java 7 on peut lancer une exception plus générale (dans le catch), vis-à-vis de celles des throws.