# Annexe 8 – Traitement des exceptions

Pour gérer les situations inhabituelles …

En procédural ( C ) :

* Alourdit le code ( plusieurs ifs )
* Le rend alors difficile à comprendre
* Découragement vis-à-vis cette tâche

En orienté-objet (Java ) :

* On traite les situations inhabituelles / problématiques selon le contexte à l’aide d’objets Exception

## Avantages de cette méthode

1. Séparation entre le traitement des exceptions et le déroulement normal du programme ( évite le code spaghetti, perd le fil conducteur )
2. Richesse des objets : peut contenir plusieurs infos complémentaires sur l’erreur en question plutôt qu’un simple code d’erreur ( valeur qui a causé l’erreur , etc )
3. Traitement de la situation problématique à l’endroit voulu seulement :

Activité ( interface graphique lecteur musique )

Classe ListeLecture

Méthode trierOrdreDuree

Classe Fichier

Variable nbSec =0 ???

\*\*\*On peut traiter l’exception à l’endroit désiré ( l’exception se propage dans la pile d’appel tant qu’elle n’est pas captée pour être traitée )

1. Hiérarchie des objets Exception : une erreur de lecture ( transmettre une string quand on demande un int ) peut être considérée comme telle ou comme une erreur générale d’entrée – sortie.

Hiérarchie des classes :

Object

Exception controller

Throwable ( interface )

Exception Error

Exception non-controller

SQLException IOException Nos propres classes D’exception RuntimeException StackOverflowError OutOfMemoryError

NullPointerException IndexOurOfBoundsException ArithmeticException NumberFormatException  **…**

**Erreurs : situation dont le programme ne peut pas récupérer, on ne peut pas traitrer des erreurs**

**Exceptions non-contrôlées ( unchecked ) : Toutes les exception dont runTime est parent. dont la class d’origine est une sous class, diect ou non.elle sont les plus commune**

**Les exceptions controller et non controller ce gere different**

* NullPointerException : appelle une méthode sur une référence nulle
* ArithmeticException : division par 0
* NumberFormatException : problème de transtypage ( essayer de transtyper "bonjour" en double )

**Traitement des exceptions non-contrôlées**

Exercice Placement : Récupérer le projet Placement présent sur LÉA. Dans un nouveau projet, intégrez la classe Placement, l’activité ainsi que le fichier de positionnement xml fournis sur LEA

1. Regarder la classe Placement
2. Regarder le fichier de positionnement XML
3. Gérer l’événement clic sur le bouton de manière à afficher le résultat de l’investissement
4. Qu’arrive-t-il si on écrit du texte dans le champ texte plutôt qu’un montant ?

Solution try / catch

Voir projet placement

Autres exercices …

## En résumé blocs try / catch

Try :

On isole les lignes susceptibles de lancer une exception dans le bock try

Examiner la structur ou le coulage des lignes de manière a former un bloc plustot que plusieurs try different

1

2 – null pointer exception lance resultat 2 pas effectuer ni la lign e3-4

3

Catch()

4

5

Si l’exception correspond au catch, les lignes de code du bloc catch sont executees

Si le compilateur ne revient pas dans le try une fois l’exception capteee, il continue a la suite du bloc catch

\*\*\*les blocs catch doivent être organises du plus specifique vers le plus general sinon des blocs deviendraient inutiles car jamais atteint -> une exception ne peut être captee qu’une fois

Si le ou les blocs catch ne correcpondent pas a l’exception lancee, elle remonte a pile d,appel a la rechercher d’un catch qui saura l’acceuillir tout comme si un exception est lancee a l’exterieur de bloc try

Structure try / catch : commune dans plusieurs langages

## Autre bloc possible : finally

* Facultatif, qu’on place a la suite du catch, utile pour placer des methode qui doivent absolument être executees (ferme connexion sql, lien vers un fichier txt)

**Un bloc finally est toujours traversé au complet, qu’une exception ait été lancée précédemment dans le bloc try ou non**

**Un bloc finaly est traverse quand on quitte le try peu importe la raison**

try

{

}

catch ( NullPointerException npe )

{

}

finally

{

}