

Support de cours

Cours:

Introduction à la programmation orientée objet (en Java)

Vidéo:

W16-03-exceptionscomplements-JAVA-pt1

Concepts (extraits des sous-titres générés automatiquement) :

Gestion des exceptions. Programme principal. Bas niveau inverse. Niveau intermédiaire. Niveau du programme. Partie du programme. Propres classes d'exception. Reste du programme. Bloc catch. Règle particulière. Partie intermédiaire du programme. Position du tableau. Reste du programme de l'indice. Niveau intermédiaire du programme. Inverse des températures.



[vers la recherche de séquences vidéo](#)

(dans Introduction à la programmation orientée objet (en Java).)



[vers la vidéo](#)

Center for Digital Education. Plus de matériel de soutien pédagogique ici :

<https://www.epfl.ch/education/educational-initiatives/cede/educational-technologies-gallery/boocs-en/>

Gestion des exceptions : compléments

(Partie 1)

Introduction à la programmation orientée objet (en Java)

Jamila Sam, Jean-Cédric Chappelier et Vincent Lepetit

...

notes

résumé

0m 0s





Nous abordons dans cette séquence plusieurs compléments

notes

résumé

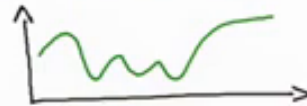
0m 1s



Exceptions

```
-- main (-) {  
try {  
    double[] temperatures = new ...;  
    acquiescer(temperatures);  
    graphiqueInverse(temperatures);  
} catch (...) {  
    rattraper et traiter  
    l'objet  
}
```

{ 35.0, 35.7, 37.2, 0.0, 38.2 ... }



```
void graphiqueInverse(double[] temps) {  
    for (double t : temps) {  
        afficher(inverse(t));  
    } catch (...) { throw new  
                    Exception("pb  
                    en "+t);  
    }
```

```
double inverse(double x) {  
    throw .....  
    return 1/x;  
}
```

lancement d'une exception
(objet)

notes

résumé

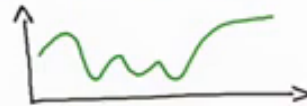
0m 6s



Exceptions

```
-- main (-) {
try {
double[] temperatures = new ...;
acquerir(temperatures);
graphiqueInverse(temperatures);
} catch (...) {
rattraper et traiter
l'objet
}
```

{ 35.0, 35.7, 37.2, 0.0, 38.2 ... }



```
void graphiqueInverse(double[] temps) {
int i;
try {
for (double t : temps) {
afficher(inverse(t));
t++;
} catch (...) { throw new
Exception("pb
en "+i);
}
```

```
double inverse(double x) {
throw .....
return 1/x;
}
```

→ lancement d'une exception
(objet)

où on pourrait décider d'acquérir à nouveau toutes les températures, parce que les premières étaient erronées par exemple. La seule chose que peut faire cette méthode est en fait d'informer le reste du programme de l'indice auquel s'est produit le problème, et de relancer l'exception à un niveau plus élevé, plus informé de comment résoudre exactement cette situation anormale. La méthode graphique inverse pourrait donc à son tour lancer une exception, pour indiquer au reste du programme à quel indice s'est produit le problème, par exemple comme ceci, je déborde un peu de mon cadre mais ce n'est pas trop grave, et il faut pour ceci bien sûr que l'indice soit géré, ce qu'on peut imaginer de faire comme ceci par exemple. Désormais si une exception est lancée lors de l'exécution de cette boucle, le programme principal va l'intercepter en recevant un message qui peut potentiellement l'informer de l'indice

notes

résumé

Une exception peut être **partiellement traitée** par un bloc **catch** et *attendre* un *traitement* plus complet *ultérieur* (c'est-à-dire à un niveau supérieur).

Il suffit pour cela de « **relancer** » **l'exception** au niveau du bloc n'effectuant que le traitement partiel.

(Il faudra bien sûr pour cela que l'appel à ce bloc **catch** soit lui-même dans un autre bloc **try** à un niveau supérieur).

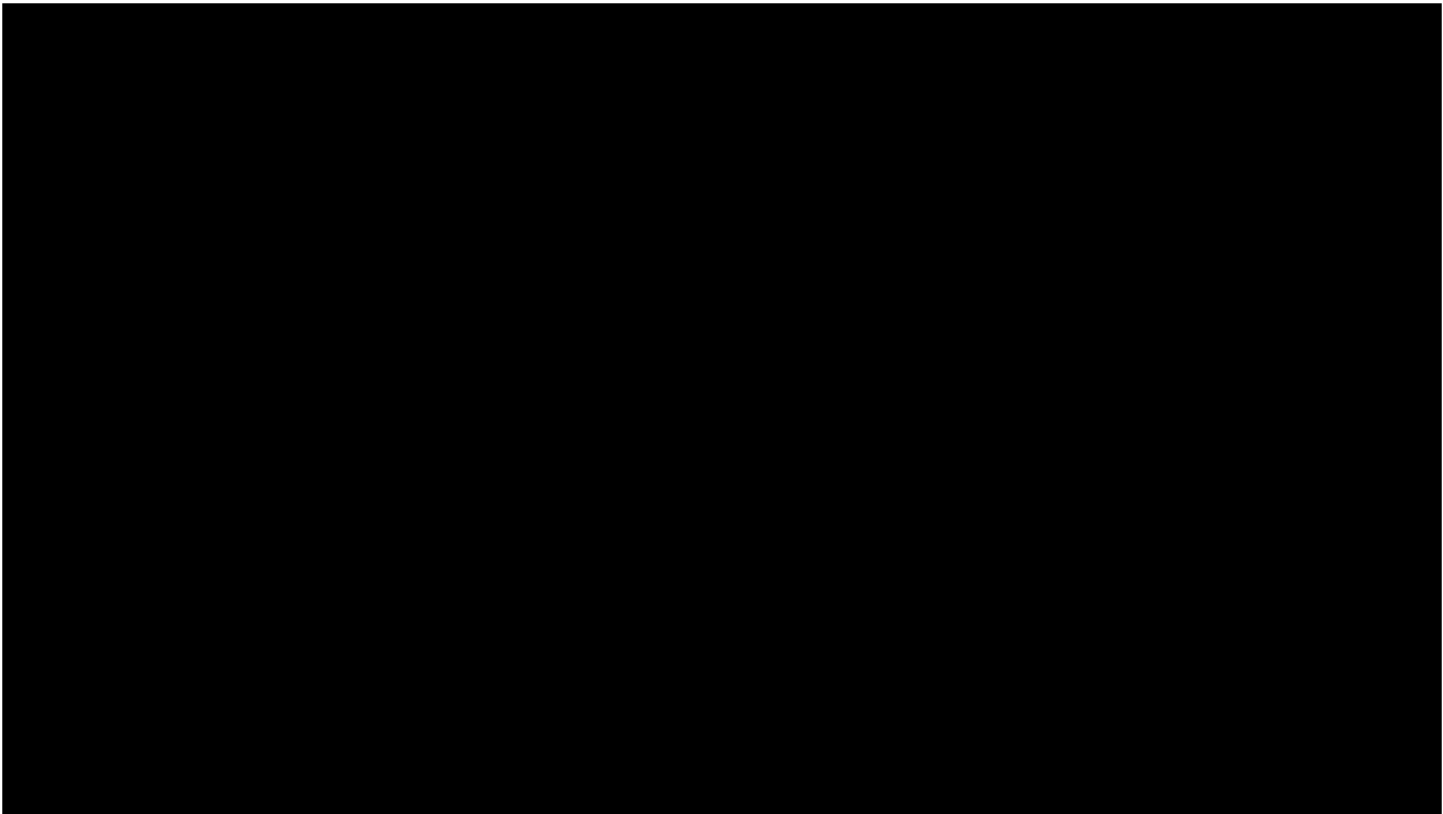
auquel s'est produit le problème. Une partie intermédiaire du programme a donc ici choisi de relancer une exception, un peu plus informée que la précédente, pour que une partie plus informée du programme puisse en faire bon usage On a donc traité ici l'exception d'origine de façon partielle. Comme nous venons donc de le voir sur cet exemple, une exception peut être partiellement traitée à un niveau intermédiaire du programme, par un bloc catch dédié et attendre un traitement plus complet ultérieur, pour cela il faut au niveau intermédiaire relancer l'exception, la même ou une nouvelle plus informée,

notes

résumé

3m 1s





il faut bien sûr pour cela que le niveau supérieur soit équipé de bloc try et catch correspondant, capable d'intercepter et de traiter proprement l'exception relancée. L'idée est donc qu'à un niveau intermédiaire du programme, on intercepte l'exception lancée d'un niveau plus bas, on peut la traiter partiellement, et puis se dire qu'on n'est pas suffisamment informé pour la traiter complètement, donc relancer l'exception, soit la même qu'on a reçue, soit une nouvelle avec un autre message ou un autre contenu. ou un autre contenu.

notes

résumé

3m 37s