

Support de cours

Cours:

## Introduction à la programmation orientée objet (en Java)

Vidéo:

### W16-03-exceptionscomplements-JAVA-pt2

Concepts (extraits des sous-titres générés automatiquement) :

**Niveau intermédiaire du programme. Meilleur programmation. Indices possibles. Type d'exception. Exception. Niveau supérieur. Liste des exceptions. Entête de la méthode. Syntaxe particulière. Méthode inverse. Cas de checked exceptions. Niveau supérieur du programme. Reste du programme. Bloc try. Intérieur de la méthode inverse.**



[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 2)

Introduction à la programmation orientée objet (en Java)

Jamila Sam, Jean-Cédric Chappelier et Vincent Lepetit

...

notes

résumé

0m 0s





Vous savez maintenant qu'une exception peut être interceptée

notes

---

---

---

---

---

---

---

---

---

---

résumé

0m 1s



---

---

---

---

---

Une méthode lançant une exception sans la traiter localement doit généralement informer qu'elle le fait

Ceci se fait en ajoutant une clause `throws` à l'entête de la méthode  
Syntaxe :

Type method(...) throws *Exception1*, *Exception2*, ...

Exemple :

```
double inverse(double x) throws Exception
{
    if (x == 0.0) {
        throw new Exception("Une division par zéro");
    }
    return 1/x;
}
```

à un niveau intermédiaire du programme pour être traitée partiellement, puis relancée à un niveau supérieur, mieux informé du contexte de l'erreur. Nous allons maintenant aborder un autre sujet, la règle déclarée ou traitée, qui est spécifique à Java. Lorsqu'une méthode lance une exception et qu'elle en délègue la gestion à un niveau supérieur du programme, c'est-à-dire qu'elle ne la traite pas localement, elle doit en général informer qu'elle lance une exception potentiellement. Ainsi, celui qui utilise cette méthode à un niveau supérieur, sait qu'elle peut lancer l'exception, et donc s'apprête lui-même à pouvoir gérer cette exception, ou la relancer à un niveau supérieur. Pour qu'une méthode déclare qu'elle peut lancer potentiellement une ou plusieurs exceptions, on doit utiliser une syntaxe particulière, après l'entête de la méthode, indiqué par le mot réservé `throws`, la liste des exceptions qui peuvent être lancées par la méthode, séparées par des virgules. Par exemple ici, la méthode `inverse` lance une exception, lorsqu'il y a une division par 0, elle ne traite pas cette exception localement, il n'y a pas de bloc `try` et `catch` à l'intérieur de la méthode `inverse` qui capterait cette exception là,

notes

résumé

0m 6s



unchecked !

Toutes les exceptions en dehors des RuntimeException et des Error doivent :

- ▶ soit **être interceptées** dans la méthode où elles sont lancées ;
- ▶ soit **être déclarées** par la méthode.

Si une exception de ce type est lancée sans être interceptée

le compilateur émettra un message d'erreur

« **Checked exceptions** »

elle doit donc signaler au reste du programme, à qui elle délègue la gestion de cette exception qu'elle lance ce type d'exception. Ceci se fait au moyen du mot réservé throws suivi de la liste des exceptions lancées par la méthode. En Java, il faut appliquer une règle particulière, toute exception qui n'est pas RuntimeException ni une Error, doit impérativement respecter la règle "déclarer ou traiter". Ce qui veut dire concrètement que ces expressions doivent soit être directement interceptées dans la méthode où elles sont lancées, ce qui est possible dans certaines situations, soit être déclarées par la méthode, pour indiquer au reste du monde que la méthode délègue la gestion de cette exception au reste du programme. En java donc, si une exception n'est pas de type RuntimeException ou Error, alors si elle n'est pas interceptée dans la méthode où elle est lancée, et si elle n'est pas déclarée par la méthode, le compilateur va émettre un message d'erreur. On parle dans ce cas de Checked exceptions, les exceptions contrôlées par le compilateur, par opposition aux Unchecked, pour lequel le compilateur ne va pas imposer qu'il y ait forcément un traitement ou une indication qu'une telle exception est lancée. Ces contrôles faits par le compilateur ont pour objectif de forcer la gestion des exceptions. Le compilateur ne veut cependant pas vous forcer à gérer les exceptions de type Runtime ou Error, Error correspond en effet à des situations que le programmeur n'est pas en mesure de gérer lui-même, par exemple, plus suffisamment, de mémoire sur la machine sur laquelle s'exécute le programme, il n'y aurait donc pas de sens à ce que le compilateur force le programmeur à traiter ce genre d'exceptions. De leur côté, les RuntimeExceptions correspondent très souvent à des situations qui pourraient être résolues plus proprement par une

notes

résumé

1m 13s



unchecked !

Toutes les exceptions en dehors des `RuntimeException` et des `Error` doivent :

- ▶ soit **être interceptées** dans la méthode où elles sont lancées ;
- ▶ soit **être déclarées** par la méthode.

Si une exception de ce type est lancée sans être interceptée

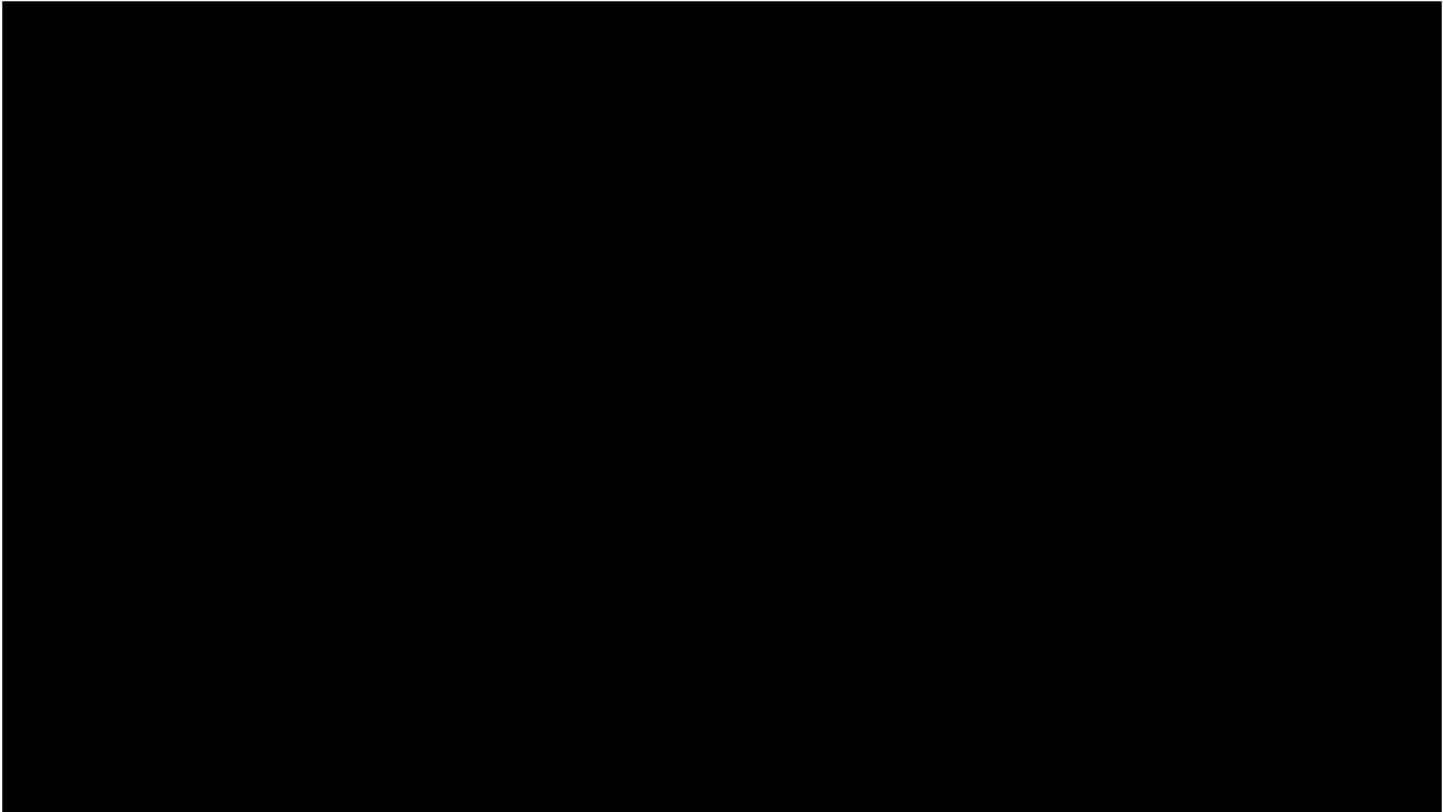
🔊 le compilateur émettra un message d'erreur

🔊 « *Checked exceptions* »

meilleur programmation.

notes

résumé



Par exemple, lorsque l'on sort des indices possibles pour un tableau, ou lorsqu'on fait une division par 0, alors ça peut être des situations qui peuvent être résolues par une meilleure programmation, mais ça peut aussi être des situations que l'on veut pouvoir gérer par le biais des exceptions. Les problèmes qui pourraient être résolus par une meilleure programmation devraient l'être par ce biais là, et non par le biais de gestion d'exception, et c'est pourquoi le compilateur ne vous force pas à déclarer ou traiter des RunTimeExceptions non plus. des RunTimeExceptions non plus.

notes

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

résumé

3m 1s



.....

.....

.....

.....

.....