

throwable, multiple exceptions, эксепшн внутри метода finalize

Java Collections
8 уровень, 5 лекция

ОТКРЫТА

— Ага. Вот ты где. Я уже тебя обыскалась.

— Что-то случилось?

— Нет, но мы же занимаемся.

— Ок. Я тебя слушаю.

— Я хочу тебе рассказать об исключениях еще пару фактов:

В Java 7 конструкция **try-catch** была немного расширена за счет добавления «множественных catch». Смотри пример:

Java 5	Java 7
<pre>1 try 2 { 3 ... 4 } 5 catch (IOException ex) 6 { 7 logger.log(ex); 8 throw ex; 9 } 10 catch (SQLException ex) 11 { 12 logger.log(ex); 13 throw ex; 14 }</pre>	<pre>1 try 2 { 3 ... 4 } 5 catch (IOException SQLException ex) 6 { 7 logger.log(ex); 8 throw ex; 9 }</pre>

— Т.е. теперь можно писать несколько исключений через ИЛИ («|» — это бинарный ИЛИ)?

— Да, правда, удобно?

— Гм. А какой тип будет у этого объекта-исключения внутри блока catch?

Ведь у IOException могут быть свои методы, а у SQLException – свои.

— Тип исключения будет такой же, как и у их общего класса-предка.

— Ага. Т.е. скорее всего **Exception** или **RuntimeException**. Почему тогда просто не написать catch(Exception e)?

— Иногда, когда программисты обрабатывают все ошибки отдельно, бывает удобно разбить их на группы и одни ошибки записать в лог, другие пробросить выше, третьи обработать иным способом.

Т.е. такая схема призвана решить проблему дублирования catch-блоков кода для обработки разных ошибок.

— Ага. Понято. Спасибо, Элли.

— Это еще не все. я хочу еще рассказать немного про блок **finally**.

Когда я говорю всегда, я имеют ввиду **абсолютно всегда**.

Пример:

Пример с finally	
1	try
2	{
3	return 1;
4	}
5	finally
6	{
7	return 0;
8	}

Тут есть **return** в блоке **try** и **return** в блоке **finally**. Так вот, результатом вызова метода будет число 0.

Блок **finally** выполнится, что бы ни произошло. А его метод **return** перезабер старое возвращаемое значение своим значением.

— Ясно.

Более того, **метод может либо вернуть значение, либо выбросить исключение**.

Поэтому, **если в блоке try возвращается значение, а в finally выбрасывается исключение, то в результате будет исключение**.

— А если в блоке **try** выбрасывается исключение, а в блоке **finally** выполняется return?

Тогда считается, что метод корректно отработал и возвращается значение, которое было передано в return.

Пример	Результа
<div><div>1 try</div><div>2 {</div><div>3 return 1;</div><div>4 }</div><div>5 finally</div><div>6 {</div><div>7 return 0;</div><div>8 }</div></div>	0
<div><div>1 try</div><div>2 {</div><div>3 return 1;</div><div>4 }</div><div>5 finally</div><div>6 {</div><div>7 throw new RuntimeException();</div><div>8 }</div></div>	RuntimeException
<div><div>1 try</div><div>2 {</div><div>3 throw new RuntimeException();</div><div>4 }</div><div>5 finally</div><div>6 {</div></div>	0

```
1  try
2  {
3      throw new RuntimeException();
4  }
5  finally
6  {
7      throw new IOException();
8  }
```

IOException

Единственной причиной, по которой может не выполняться метод finally, может быть немедленное завершение программы посредством вызова метода **System.exit();**

Пример

```
1    try
2    {
3        System.exit(0);
4        return 1;
5    }
6    finally
7    {
8        return 0;
9    }
```

— Ясно.

— Учти, обычно на собеседованиях спрашивают все из этой темы, так что лучше тебе это и понять и запомнить.

[← Предыдущая лекция](#)

 ×28

 **+52** 

Комментарии (16)

популярные

НОВЫЕ

старые

JavaCoder

Введите текст комментария

НАЧАТЬ ОБУЧЕНИЕ

— Гм. А какой тип будет у этого объекта-исключения внутри блока catch? Ведь у IOException могут быть свои методы, а у SQLException – свои.

— Тип исключения будет такой же, как и у их общего класса-предка.

Стоит добавить, что класс предка будет использоваться только в пределах блока catch в котором перехвачено исключение. Если пробросить его дальше, то оно будет иметь исходный тип. Даже если перехватывать Exception и пробрасывать его далее, то в вышестоящем методе исключение будет иметь исходный тип. Т.е. нижеприведенный код будет выводить :

```
1 void f() {
2     if (((int) (Math.random() * 2)) ==1 ) throw new NullPointerException();
3     else throw new ArrayIndexOutOfBoundsException();
4 }
5
6 void g() {
7     try {
8         f();
9     }
10    catch (NullPointerException | ArrayIndexOutOfBoundsException e) {
11        throw e;
12    }
13 }
14
15 void j() {
16     try {
17         g();
18     }
19    catch (NullPointerException e) {
20        System.out.println("NullPointerException");
21    }
22    catch (ArrayIndexOutOfBoundsException e) {
23        System.out.println("ArrayIndexOutOfBoundsException");
24    }
25 }
```

Ответить

−

+1

+

Pig Man

Главная свинья в Свинарнике

23 февраля 2021, 13:04

...

Предыдущая лекция: "Считать исключение ошибками – это неверный подход"

Эта лекция: "Когда программисты обрабатывают все **ошибки** отдельно"

Как людям правильно соблюдать терминологию, если даже учителя ее не соблюдают?

Ответить

−

+1

+

Андрей

Уровень 41, Самара

6 февраля 2021, 00:08

...

Javarush, посмотрите эту страницу. На ней куча опечаток. Пожалуйста исправьте.

Ответить

−

0

+

Ars

Уровень 41

29 ноября 2021, 15:30

...

Всё ещё не везде исправили. Например в предпоследнем блоке кода столбец называется "Результата"

Ответить

−

0

+

skrskr

Уровень 39, Санкт-Петербург, Россия

5 февраля 2020, 22:50

...

Есть объяснение почему нельзя использовать сокращённые операторы, а не только побитовые?

Ответить

−

+1

+

Илья

Backend Developer в СберТех

18 февраля 2021, 15:50

...


ты написал одно и тоже))))

Ответить

−

+1

+

 LuneFox

Уровень 41, Москва, Россия

EXPERT

8 марта, 09:10

...

Он разное написал. Сокращённый это ||, а под побитовым имеется в виду |.

Ответить

−

0

+

Игорь

Уровень 41, Екатеринбург, Россия

14 июня 2019, 11:02

...

А где про эксепшн внутри метода finalize?

Ответить

−

+3

+

Валентин Кудинов

Уровень 41, Самара, Россия

7 апреля 2020, 07:34

...

Что произойдет со сборщиком мусора (GC), если во время выполнения метода finalize() некоторого объекта произойдет исключение?

Ответ
Во время старта JVM запускается поток finalizer, который работает в фоне. Этот поток имеет метод runFinalizer, который игнорирует все исключения методов finalize объектов перед сборкой мусора.

То есть если во время выполнения метода finalize возникнет исключительная ситуация, его выполнение будет остановлено и это никак не скажется на работоспособности самого сборщика мусора (garbage collector).

http://www.quizful.net/interview/java/exception-while-finalize

Ответить

+7

Пётр Уровень 41, Москва, Россия EXPERT 26 апреля 2019, 21:35

finally, насколько помню, не выполняется при System.exit(0) или (-1) в catche или основном коде

Ответить

0

Павел X. Team Lead в Иннотех EXPERT 20 декабря 2018, 13:25

Цитата [отсюда](#):
Note: If the JVM exits while the try or catch code is being executed, then the finally block may not execute. Likewise, if the thread executing the try or catch code is interrupted or killed, the finally block may not execute even though the application as a whole continues.

[Здесь](#) перечислены случаи, когда до finally не дойдёт.
If you invoke System.exit();
If the JVM crashes first;
If the JVM reaches an infinite loop (or some other non-interruptable, non-terminating statement) in the try or catch block;
If the OS forcibly terminates the JVM process; e.g. "kill -9 " on UNIX.
If the host system dies; e.g. power failure, hardware error, OS panic, etcetera.
If finally block is going to be executed by daemon thread and all other non daemon threads exit before finally is called.

Мне на собеседовании тоже задавали такой вопрос, я смог придумать пару-тройку причин.

Ответить

+31

AlexandeRogov Уровень 40, Москва, Россия 29 ноября 2017, 20:33

Ходил на собеседование в банк и мне задали такой вопрос, но в джавараш я ещё не дошел до этого уровня, по этому не смог ответить. Полазив в инете нашел ещё один способ, по которому может не выполниться метод finally это бесконечный цикл, по сути до finally дело и не дойдет.

Ответить

+14

Артем Прудников Уровень 35, Москва, Россия 30 января 2018, 20:03

А вызвать Error типа StackOverFlow или OutOfMemory?

Ответить

+3

Naera Meir Уровень 40, Москва, Россия 1 апреля 2018, 19:22

хоть и давно спрашивал, напишу, мб кому будет полезно
проведя пару "опытов" и почитав стак, понял, что блок finaly выполняется, но в реальности, к примеру при OutOfMemory память может быть исчерпана на столько, что finaly не отработает корректно, но запущен будет

Ответить

+7

NodeOne Уровень 41 EXPERT 8 марта 2019, 17:59

OutOfMemory это unchecked Error - уровень за пределами java машины... все равно что рассуждать выполнится ли finali{} при жестком ресете системы...

Ответить

+5

Anton Stezhkin Уровень 41 20 августа 2021, 17:17

OutOfMemory - создай список строк или чего-нибудь неизменяемого и пихай туда что-нибудь пока память не кончится.
StackOverFlow - когда-то какой-то рекурсивный метод у меня бросил это исключение. Но я не помню почему.

Ответить

0

ОБУЧЕНИЕ

Курсы программирования

Курс Java

Помощь по заданиям

СООБЩЕСТВО

Пользователи

Статьи

Форум

КОМПАНИЯ

О нас

Контакты

Отзывы

НАЧАТЬ ОБУЧЕНИЕ



JavaRush — это интерактивный онлайн-курс по изучению Java-программирования с нуля. Он содержит 1200 практических задач с проверкой решения в один клик, необходимый минимум теории по основам Java и мотивирующие фишки, которые помогут пройти курс до конца: игры, опросы, интересные проекты и статьи об эффективном обучении и карьере Java-девелопера.

ПОДПИСЫВАЙТЕСЬ

ЯЗЫК ИНТЕРФЕЙСА



СКАЧИВАЙТЕ НАШИ ПРИЛОЖЕНИЯ

