иерархия исключений, errors

Java Collections
8 уровень, 3 лекция

ОТКРЫТА

— Привет, Амиго!

— Привет, Элли! Как жизнь?

— Отлично, спасибо. А ты как?

— Супер, сегодня утром столько всего нового рассказали.

— Так это же отлично. Не устал?

— Да, есть такое дело. Подустал немного.

— Тогда тебе просто повезло. Потому что хотела тебе сегодня дать большую и сложную тему, но в последний момент передумала и решала рассказать маленькую и легкую.

— Маленькую и легкую? Я готов.

— Сегодня мы с тобой детально разберем тему исключений — **Exception**.

— Ты про обработку ошибок?

— Считать исключение ошибками – это неверный подход. Исключения больше похожи на отчет о том, «что пошло не так». На основе которого можно предложить альтернативные сценарии действия.

Все дело в методах. Когда ты вызываешь какой-то метод, он гарантированно должен сделать то, для чего его вызвали.

Когда метод по каким-либо причинам не может сделать то, для чего его вызвали, он обязан уведомить об этом вызывающего.

Т.е. самое худшее, что может случиться – это когда метод не выполнил свою работу, и никого об этом не уведомил. Хуже этого не может быть ничего. Это потеря контроля над ситуацией.

Когда ты начинающий программист, тебе кажется, что ты вызываешь методы и они обязательно сделают то, о чем ты их просишь.

Когда ты опытный программист, ты знаешь, что могут быть десятки факторов, которые влияют на выполнение методом своей работы, и возможны очень много различных случаев, когда метод не смог выполнить свою работу.

С точки зрения программиста, если в программе произошел сбой, и она закрылась – это в тысячу раз лучшая ситуация, чем, если в программе произошел сбой, она работает неправильно и пользователь об этом не подозревает.

— Как то, что программа что-то показывает неправильно, может быть хуже, чем, если программа закрылась, и все данные потерялись?

— А с чего ты взял, что программа просто что-то неправильно показывает? Может у нее внутри куча ошибок и все данные твоей работы с ней будут безвозвратно потеряны? Ты 3 часа набирал текст, который не сохранится, потому что на второй минуте работы произошла ошибка.

Когда начинающий программист сталкивается с исключениями, он расстраивается.

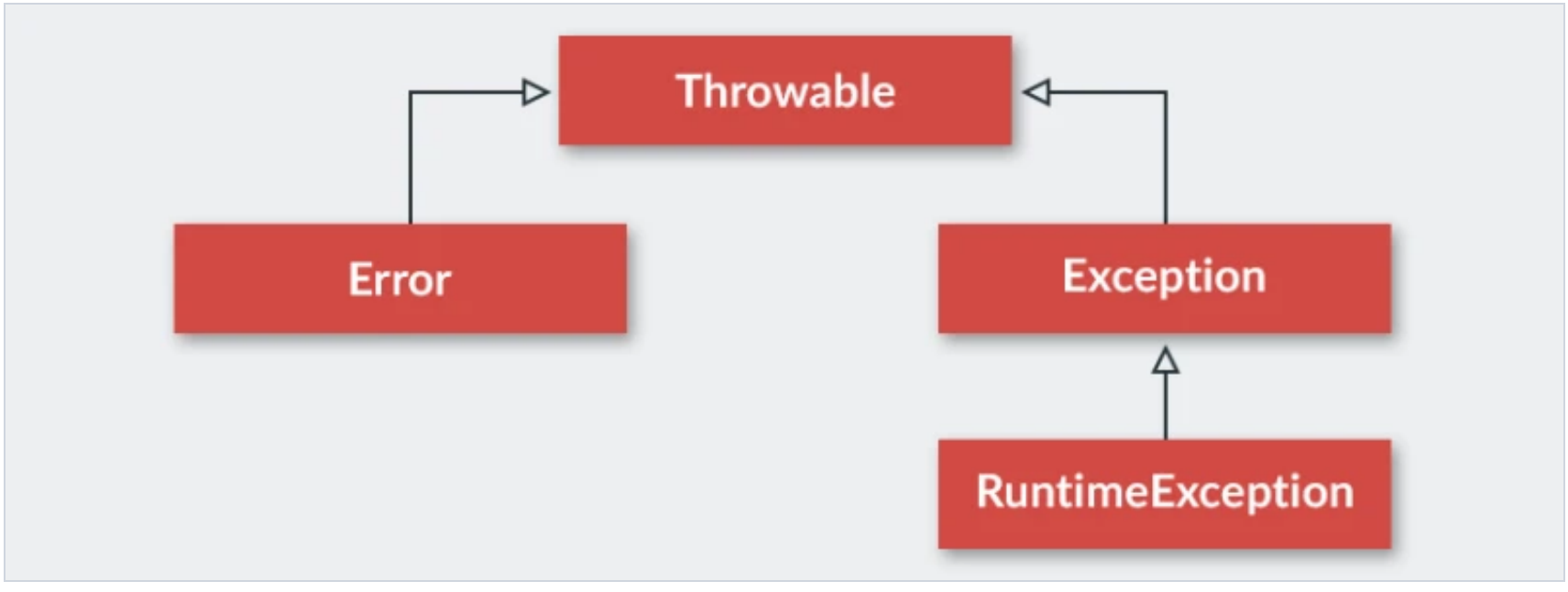
Но на самом деле, исключения показывают ему все те сценарии поведения, которые он должен был предусмотреть и не предусмотрел.

НАЧАТЬ ОБУЧЕНИЕ

Метод либо делает то, для чего он написан, либо кидает исключение. Третьего не дано!

— Ладно, верю. Обещаю пользоваться исключениями.

— Отлично. Тогда давай я тебе расскажу про иерархию исключений:



Иерархия исключений базируется на четырех классах.

Самый базовый класс – это **Throwable**.

От него унаследованы классы **Error** и **Exception**.

От **Exception** унаследован **RuntimeException**.

Класс **Error** является базовым классом для ошибок Java-машины, таких как **StackOverFlow**, **OutOfMemory**, …

Исправить последствия таких ошибок программа обычно не может, и это приводит к ее завершению.

Действительно, что можно сделать, если для дальнейшей нормальной работы программы не хватает памяти или переполнился стек.

Класс **Exception** является базовым для всех обычных исключений, которые выбрасываются программой. RuntimeException – это подкатегория исключений Exception, к которым применяются немного другие правила.

— Это какие?

— Как раз сейчас и расскажу.

Как ты, наверное, помнишь, исключения делятся на две категории **checked** (проверяемые) и **unchecked** (непроверяемые).

Если метод выкидывает **checked** исключения, то вызывающий его метод обязан обернуть вызов такого метода в блок **try-catch**. Ну, или пробросить исключение выше (своему вызывающему), явно указав его в списке **throws** в сигнатуре метода.

На unchecked такие правила/ограничения не распространяются.

Так вот, все исключения, унаследованные от Exception, считаются checked. Кроме исключений, которые унаследованы от RuntimeException – они unchecked.

— Ага. Вспоминаю, что-то такое мне уже раньше рассказывали.

— Амиго! Иерархию исключений спрашивают на каждом собеседовании. Еще раз – на каждом. Ты должен великолепно знать эту тему.

— Ок. Я еще раз все перечитаю и разберусь. Спасибо, что помогаешь мне, Элли.

НАЧАТЬ ОБУЧЕНИЕ

+42

Комментарии (23) популярные новые старые

JavaCoder

Введите текст комментария

Igor Petrashevsky Уровень 47 — 28 августа, 00:39

опять мана на ветер

Ответить +1

Олег Вербецкий Уровень 41 — 15 августа 2021, 18:48

Меня спрашивали, какие выди есть и привести 2-3 примера)

Ответить +1

LuneFox инженер по сопровождению в **BIFIT** EXPERT — 8 марта, 07:02

Куда выди?

Ответить +2

Oleg Khilko Уровень 51, Москва, Russian Federation — 14 августа, 18:51

Так это не должно же составлять большой сложности если к этому процессу подойти на стадии подготовки к собесам и разложить по полочкам как Throwable, так и Iterable (Коллекцию).

Тут больше тебя пытаются зацепить на то понимаешь ты когда какая будет или нет. Один раз себе расписать на листочке и выучить👍

Ответить 0

LokiLaufeyson Уровень 41, Киев — 28 апреля 2021, 17:57

Вообще, третье дано - можно вернуть Optional, и тогда клиентский код сам пусть решает, что делать, если метод не смог что-то сделать

Ответить +3

Justinian Judge в **Mega City One** MASTER — 11 февраля, 12:55

Это включено в первое.

```
1  Метод либо делает то, для чего он написан, либо кидает исключение. Третьего
```

Если задача метода к примеру findById искать энтити, как процесс.
Если находит - ок, не находит возвращает энтити.
Тогда возвращается Optional, и это абсолютно соотносится с первым подходом "метод делает то, для чего написан - осуществляет ПРОЦЕСС поиска, и ЕСЛИ находит результат, возвращает , ЕСЛИ нет, возвращает пустой опшенал"

Возможно переформулировать иначе: метод должен НАЙТИ ! энтити обязательно.
Тогда Optional никак не подходит под первый кейс, пустой опшенал это нарушение контракта метода. ТОгда идет второй сценарий - исключение:
ЕСЛИ энтити не найдена - ТОГДА бросаю исключение, поскольку метод, не сделал свою работу, ничего не нашел.

Поэтому, в этой плоскости, третьего и не дано, Optional это лишь дополнительный инструмент работы с отсутствием значения. Но работа с Optional в принципе относится к первому пункту - это флоу, который программист определяет как корректный.

То есть метод может вернуть, может не вернуть и это ок - в этом плант работа с Optional мало чем отличается с методом к примеру, который возвращает либо коллекцию с данными либо пустую коллекцию. Это тоже первый пункт - корректная работа программы.

А исключение это другая ситуация, когда мы говорим - метод ОБЯЗАН что-то сделать, если он не сделал - бросаем исключение здесь и сейчас.

Ответить 0

Алексей Мирный Backend Developer — 18 января 2021, 13:31

alex_us Уровень 41, Симферополь — 30 января 2021, 15:29

а что на собесах спрашивают? и устроился ты ?) если не секрет конечно

Ответить 0

Алексей Мирный Backend Developer — 5 февраля 2021, 21:14

После всей этой теории, перешли к практике logg, дебаг, XML итд, сказал как есть...) пока на стажbровке от JR

Ответить 0

alex_us Уровень 41, Симферополь — 5 февраля 2021, 21:37

на многих собесах был?)

Ответить 0

Алексей Мирный Backend Developer — 8 февраля 2021, 08:13

На одном))

Ответить 0

alex_us Уровень 41, Симферополь — 8 февраля 2021, 09:40

а почему так?)

Ответить 0

Алексей Мирный Backend Developer — 8 февраля 2021, 12:27

Решил пока паузу взять в поисках и сосредоточится на стажировке, да и резюме надо переделывать

Ответить 0

Андрей Уровень 41, Минск, Беларусь — 10 августа 2019, 17:24

Если не забуду, то отпишусь спросили ли

Ответить +4

Даниил Salesforce Developer в **Viseven** MASTER — 8 октября 2019, 18:10

Забыл или ещё не ходил?

Ответить +1

Андрей Уровень 41, Минск, Беларусь — 9 октября 2019, 01:00

Я рассчитывал что напомнят в комментариях, спрашивали

Ответить +2

Даниил Salesforce Developer в **Viseven** MASTER — 9 октября 2019, 10:23

А в каком объёме ориентировочно?

Ответить 0

Андрей Уровень 41, Минск, Беларусь — 9 октября 2019, 23:21

Какие бывают, как создать свое исключение (от чего наследовать) и как выбросить (throw new)

Ответить 0

Даниил Salesforce Developer в **Viseven** MASTER — 9 октября 2019, 23:23

И всё?

Ответить 0

Андрей Уровень 41, Минск, Беларусь — 10 октября 2019, 22:46

Что еще по этой теме можно спросить?

Ответить 0

Даниил Salesforce Developer в **Viseven** MASTER — 10 октября 2019, 23:38

Ну что могут спросить...

Ответить 0

mila Уровень 40, Самара, Россия — 27 июня 2018, 00:17

На каждом, Карл!

Ответить +15

wan-derer.ru Уровень 40, Москва, Россия — 17 декабря 2020, 16:50

нэтЪ :)

Ответить 0