

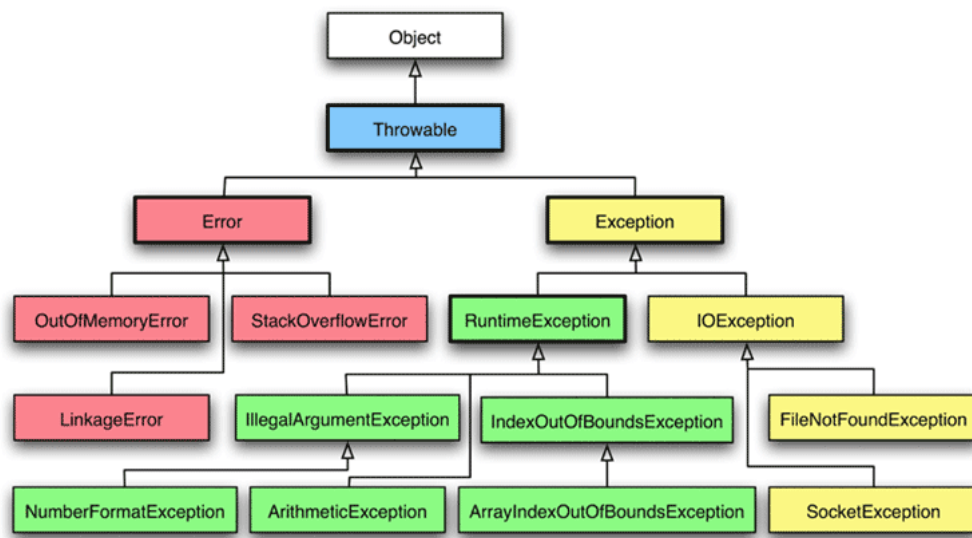


Типы исключений

Java Syntax (/quests/QUEST_JAVA_SYNTAX)

9 уровень (/quests/lectures/?quest=QUEST_JAVA_SYNTAX&level=9), 4 лекция (/quests/lectures/questsyntax.level09.lecture04)

ОТКРЫТА



— Решила поднять сегодня ещё одну тему. В Java все исключения делятся на два типа – **контролируемые/проверяемые (checked)** и **неконтролируемые/непроверяемые (unchecked)**: те, которые перехватывать обязательно, и те, которые перехватывать не обязательно. По умолчанию – все исключения обязательно нужно перехватывать.

— А можно в коде специально выбрасывать исключения?

— В своем коде ты сам можешь выкидывать исключения. Ты даже можешь написать свои собственные исключения. Но это мы разберем позже. Сейчас же давай научимся работать с исключениями, которые выбрасывает Java-машина.

— Ок.

— Если в методе выбрасываются (возникают) исключения `ClassNotFoundException` и `FileNotFoundException`, программист обязан указать их в сигнатуре метода (в заголовке метода). Это **checked** исключения. Вот как это обычно выглядит:

Примеры проверяемых (checked) исключений

```

1 public static void method1() throws ClassNotFoundException, FileNotFoundException
2
3 public static void main() throws IOException
4
5 public static void main() //не выбрасывает никаких исключений
  
```

— Т.е. мы просто пишем `throws` и перечисляем исключения через запятую. Так?

— Да. Но тут интересно другое. Чтобы программа скомпилировалась, метод, который вызывает `method1` в примере ниже, должен сделать две вещи: **или перехватить эти исключения или пробросить их дальше (тому, кто его вызвал) указав их в своём заголовке.**

— Еще раз. Если ты в методе `main` хочешь вызвать метод какого-то объекта, в заголовке которого прописано `throws FileNotFoundException`, ... то тебе надо сделать одно из двух:

1) **Перехватывать** исключения `FileNotFoundException`, ...

Тебе придется обернуть код вызова опасного метода в блок `try-catch`

2) Не перехватывать исключения `FileNotFoundException`, ...

Тебе придется добавить эти исключения в список throws своего метода main.

— А можно пример?

— Вот, смотри:

Примеры проверяемых (checked) исключений

```
1 public static void main(String[] args)
2 {
3     method1();
4 }
5
6 public static void method1() throws FileNotFoundException, ClassNotFoundException
7 {
8     //тут кинется исключение FileNotFoundException, такого файла нет
9     FileInputStream fis = new FileInputStream("C2:\badFileName.txt");
10 }
```

— Этот пример не скомпилируется, т.к. метод main вызывает метод method1(), который выкидывает исключения, обязательные к перехвату.

— Чтобы пример скомпилировался, надо добавить обработку исключений в метод main. Сделать это можно двумя способами:

Способ 1: просто пробрасываем исключение выше (вызывающему):

```
1 public static void main(String[] args) throws FileNotFoundException, ClassNotFoundException
2 {
3     method1();
4 }
5
6 public static void method1() throws FileNotFoundException, ClassNotFoundException
7 {
8     //тут кинется исключение FileNotFoundException, такого файла нет
9     FileInputStream fis = new FileInputStream("C2:\badFileName.txt");
10 }
```

— А тут перехватываем его с помощью try-catch:

Способ 2: перехватываем исключение:

```
1 public static void main(String[] args)
2 {
3     try
4     {
5         method1();
6     }
7     catch(Exception e)
8     {
9     }
10 }
11
12 public static void method1() throws FileNotFoundException, ClassNotFoundException
13 {
14     //тут кинется исключение FileNotFoundException, такого файла нет
15     FileInputStream fis = new FileInputStream("C2:\badFileName.txt");
16 }
```

— Что-то понемногу проясняется.

— Посмотри на пример ниже, чтобы разобраться:

Не обрабатываем исключения — нужно пробросить их дальше, тому, кто знает как


```
1 public static void method2() throws FileNotFoundException, ClassNotFoundException
2 {
3     method1();
4 }
```

Обрабатываем одно исключение, второе – пробрасываем:

```
1 public static void method3() throws ClassNotFoundException
2 {
3     try
4     {
5         method1();
6     }
7     catch (FileNotFoundException e)
8     {
9         System.out.println("FileNotFoundException has been caught.");
10    }
11 }
```

Перехватываем оба – ничего не пробрасываем:

```
1 public static void method4()
2 {
3     try
4     {
5         method1();
6     }
7     catch (FileNotFoundException e)
8     {
9         System.out.println("FileNotFoundException has been caught.");
10    }
11    catch (ClassNotFoundException e)
12    {
13        System.out.println("ClassNotFoundException has been caught.");
14    }
15 }
```

ЗАДАЧА  Java Syntax, 9 уровень, 4 лекция

РЕШЕНА

★☆☆☆☆

Набираем код

×10

Внимание! Объявляется набор кода на JavaRush. Для этого включите режим повышенной внимательности, расслабьте пальцы, читайте код и... набирайте его в соответствующем окошке. Набор кода — вовсе не бесполезное занятие, как может показаться на первый взгляд: благодаря ему новичок привыкает к синтаксису и запоминает его (современные IDE редко дают ему это сделать).

Открыть

— Но есть вид исключений – это *RuntimeException* и классы, унаследованные от него. Их перехватывать не обязательно. Это *unchecked* исключения. Считается, что это трудно прогнозируемые исключения и предсказать их появление практически невозможно. С ними можно делать все то же самое, но указывать в *throws* их не нужно.

[< \(/quests/lectures/questsyntax.level09.lecture03\)](/quests/lectures/questsyntax.level09.lecture03)

[> \(/quests/lectures/questsyntax.level09.lecture05\)](/quests/lectures/questsyntax.level09.lecture05)



0



0



0



0



0

+26

Комментарии (29)

популярные

новые

старые

Никита

Витёк (Vitek) 9 уровень

вчера, 17:16

throws - это "пробрасываем"?
 try-catch - "перехватываем" и "Обрабатываем"?
 правильно ли я понял?

Ответить

0

Дмитрий 16 уровень, Москва

26 декабря 2017, 20:16

Значит, желтые исключения на блок-схеме нужно предусматривать и перебрасывать или перехватывать, зелёные (runtime) не обязательно перебрасывать или перехватывать, а красные - это технические ошибки?

Ответить

+5

Дмитрий Оносовский 11 уровень, Одесса

12 февраля, 00:35

Ответить

0

Дмитрий 31 уровень

8 октября 2017, 19:11

Ответить

0

Alexey Arkhipkin 20 уровень, Москва

20 октября 2017, 19:30

Ответить

+3

Daniel 15 уровень

26 декабря 2017, 10:37

Ответить

0

Виктор Статко 22 уровень, Минск

3 сентября 2017, 23:08

Ответить

+23

Роман 11 уровень

11 марта, 12:03

Ответить

0

Дмитрий Ляпунов 9 уровень, Рига

7 августа 2017, 00:19

Некорректный цвет в карте классов на картинке и в описании. Так желтые - checked, а зеленые и красные - unchecked, хотя из-за цвета шрифта может сложиться впечатление, что зеленые на картинке - это checked-exception.

Ответить

+23

Василий 12 уровень

23 августа 2017, 09:16

Ответить

+2

Дмитрий Оносовский 11 уровень, Одесса

12 февраля, 00:39

Ответить

0

Ренат 16 уровень

16 июня 2017, 05:57

Допустим мы пробросили исключения от метода "метод2" к методу "метод1" и не обрабатываем их конструкцией try-catch. А откуда "метод1" тогда знает, как их обрабатывать?

Ответить

0

Anton Doroshenko 37 уровень

16 июня 2017, 18:16

...

Метод1 не знает, как их обрабатывать. Все, что знает метод1 - это то, что метод2 пробрасывает исключения. И дальше есть 2 пути :

- 1 - обработать исключения в методе1
- 2 - пробросить еще дальше

Ответить

0

Ренат 16 уровень

18 июня 2017, 09:49

...

ну а если он пробросит до метода main и пробрасывать дальше некуда, откуда main знает как обрабатывать исключение? или у main-метода есть какие-то встроенные try-catch или что-то вроде этого?

Ответить

0

Anton Doroshenko 37 уровень

20 июня 2017, 12:10

...

если не обрабатываем в методе main, то метод main должен сам пробросить исключение еще дальше. В таком случае, если возникнет ошибка, то программа прекратит работу, и JVM возвратит стек трейс с указанием, в каком месте произошла ошибка.

Ответить

+11

Володя 20 уровень

15 июня 2017, 08:08

...

Копипаст? зачем нервничать?

Ответить

+4

Osip Akopyants 22 уровень, Харьков

11 июня 2017, 08:32

...

На схеме RuntimeException-зелененький-т.е.-Checked(из первого абзаца).Но последний абзац(это правильно) противоречит этому.Прошу удалить схему и первый абзац чтобы не создавать неправильные представления, неразбериху.

Ответить

+2

Игорь 19 уровень, Новосибирск

10 мая 2017, 06:46

...

Советую посмотреть лекции Головача на тему исключений.

Ответить

+6

Антон Евтух 11 уровень, Киев

5 апреля 2017, 06:49

...

Желтые (на картинке) - проверяемые, зеленые непроверяемые. В красные вообще лучше не лезть. Т.е. для желтых надо писать throws, для зеленых нет.

Ответить

+2

[Загрузить еще](#)

[sh.ru/](#), [G+ \(https://plus.google.com/114772402300089087607\)](#), [Twitter \(https://twitter.com/javarush_ru\)](#), [Facebook](#)



Программистами не рождаются
© 2018