Знакомство с исключениями

Java Syntax (/quests/QUEST_JAVA_SYNTAX)
9 уровень (/quests/lectures/?quest=QUEST_JAVA_SYNTAX&level=9), 3 лекция (/quests/lectures/questsyntax.level09.lecture03)

ОТКРЫТА

— Привет, Амиго! Сегодня будет очень интересный урок. Сегодня я расскажу тебе об исключениях. Исключения – это специальный механизм для контроля над ошибками в программе. Вот примеры ошибок, которые могут возникнуть в программе:

1. Программа пытается записать файл на заполненный диск.

2. Программа пытается вызвать метод у переменной, которая хранит ссылку – null.

3. Программа пытается разделить число на 0.

Все эти действия приводят к возникновению ошибки. Обычно это приводит к закрытию программы — продолжать выполнять дальше код не имеет смысла.

— Почему?

— А есть ли смысл крутить руль, если машина слетела с трассы и падает с обрыва?

— Программа что, должна завершиться?

— Да. Раньше так и было. Любая ошибка приводила к завершению программы.

— Это очень разумный подход.

— А разве не лучше было бы попробовать работать дальше?

— Ага. Ты набрал большущий текст в Word'е, сохранил его, он не сохранился, но программа говорит тебе, что все в порядке. И ты продолжаешь набирать его дальше. Глупо, да?

— Ага.

— Потом разработчики придумали интересный ход: каждая функция возвращала статус своей работы. 0 означал, что она отработала как надо, любое другое значение – что произошла ошибка: это самое значение и было кодом ошибки.

— Но был у такого подхода и минус. После каждого(!) вызова функции нужно было проверять код (число), который она вернула. Во-первых, это было неудобно: код по обработке ошибок исполнялся редко, но писать его нужно было всегда. Во-вторых, функции часто сами возвращают различные значения – что делать с ними?

— Ага. Я тоже об этом подумал.

— Но потом наступило светлое будущее — появились исключения и механизм обработки ошибок. Вот как это работает:

1. Когда возникает ошибка, Java-машина создаёт специальный объект – exception – исключение, в который записывается вся информация об ошибке. Для разных ошибок есть разные исключения.

2 Затем это «исключение» приводит к тому, что программа тут же выходит из текущей функции, затем выходит из следующей функции, и так пока не выйдет из метода main. Затем программа завершается. Еще говорят, что Java-машина «раскручивает назад стек вызовов».

— Но ты же сказала, что теперь программа не обязательно завершается.

— Верно, потому что есть способ перехватить исключение. В нужном месте, для нужных нам исключений мы можем написать специальный код, который будет перехватывать эти исключения и что-то делать. Важное.

— Для этого есть специальная конструкция try-catch. Вот как это работает:

Вот пример программы, которая перехватывает исключение – деление на 0. И продолжает работать.

Вот что будет выведено на экран:

```
1  public class ExceptionExample2
2  {
3      public static void main(String[] args)
4      {
5          System.out.println("Program starts");
6
7          try
8          {
9              System.out.println("Before method1 calling");
10             method1();
11             System.out.println("After method1 calling. Neve
12         }
13         catch (Exception e)
14         {
15            System.out.println("Exception has been caught");
16         }
17
18         System.out.println("Program is still running");
19     }
20
21     public static void method1()
22     {
23         int a = 100;
24         int b = 0;
25         System.out.println(a / b);
26     }
27 }
```

«Program starts»
«Before method1 calling»
«Exception has been caught»
«Program is still running»

— А почему не будет выведено «After method1 calling. Never will be shown»?

— Рада, что ты спросил. В строчке 25 у нас было деление на ноль. Это привело к возникновению ошибки – исключения. Java-машина создала объект *ArithmeticException* с информацией об ошибке. Этот объект является исключением.

— Внутри метода `method1` возникло исключение. И это привело к немедленному завершению этого метода. Оно привело бы и к завершению метода main, если бы не было блока try-catch.

— Если внутри блока try возникает исключение то, оно захватывается в блоке catch. Остаток кода в блоке try, не будет исполнен, а сразу начнётся исполнение блока catch.

— Как-то не очень понятно.

— Другими словами этот код работает так:

1. Если внутри блока try возникло исключение, то код перестаёт исполняться, и начинает исполняться блок catch.

2. Если исключение не возникло, то блок try исполняется до конца, а catch никогда так и не начнёт исполняться.

— Гм?

— Представь, что после вызова каждого метода мы проверяем: завершился ли только что вызванный метод сам по себе или в результате исключения. Если исключение было, тогда мы переходим на исполнение блока catch, если он есть, и захватываем исключение. Если блока catch нет, то завершаем и текущий метод. Тогда такая же проверка начинается в том методе, который вызвал нас.

— Теперь вроде понятно.

— Вот и отлично.

— А что значит Exception внутри catch?

— Все исключения – это классы, унаследованные от класса Exception. Мы можем перехватить любое из них, указав в блоке catch его класс, или все сразу, указав общий родительский класс — Exception. Затем из переменной e (эта переменная хранит ссылку на объект исключения), можно получить всю необходимую информацию о возникшей ошибке.

— Круто! А если в моем методе могут возникнуть разные исключения, можно обрабатывать их по-разному?

— Не можно, а нужно. Сделать это можно вот так:

Пример:

```
1  public class ExceptionExample2
2  {
3      public static void main(String[] args)
4      {
5          System.out.println("Program starts");
6
7          try
8          {
9              System.out.println("Before method1 calling");
10             method1();
11             System.out.println("After method1 calling. Never will be shown ");
12         }
13         catch (NullPointerException e)
14         {
15            System.out.println("Reference is null. Exception has been caught");
16         }
17         catch (ArithmeticException e)
18         {
19             System.out.println("Division by zero. Exception has been caught");
20         }
21         catch (Exception e)
22         {
23             System.out.println("Any other errors. Exception has been caught");
24         }
25
26         System.out.println("Program is still running");
27     }
28
29     public static void method1()
30     {
31         int a = 100;
32         int b = 0;
33         System.out.println(a / b);
34     }
35 }
```

— Блок try может содержать несколько блоков catch, каждый из которых будет захватывать исключения своего типа.

— Гм. Ну, вроде понятно. Сам такого не напишу, конечно, но если в коде встречу – пугаться не буду.

‹ (/quests/lectures/questsyntax.level09.lecture02)

×9 › (/quests/lectures/questsyntax.level09.lecture04)

0 0 0 0 0 +28

Комментарии (29) популярные новые старые

Никита

Введите текст комментария

Ghostik007 10 уровень понедельник, 10:52

Мдя, даже Amigo долго не мог понять что пытаются объяснять. А суть исключений намного проще. Возьмем абстракцию из жизни.
0) Есть дом в нем живут люди.
1) Обыденная, без происшествий жизнь этого дома, считай - **это выполнение куска кода в блоке try**.
2) Если что-то в нем происходит, то инициируется звонок в специальную службу, например 112 - **это будет прерывание выполнения блока try и передача в блока catch**.
3) Диспетчер определяет тип угрозы, например пожар, вызывает пожарную машину на место инцидента - **это будет блок catch с определенным типом исключения.**
4) Тушение пожара - **выполнение блока catch с определенным типом исключения.**

В заключении: если нет происшествия, не будет исключения, код просто выполнится в блоке try

Ответить +2

VM 19 уровень 2 февраля, 19:29

Если блока catch нет, то завершаем и текущий метод. Тогда такая же проверка начинается в том методе, который вызвал нас.
Кого - нас? В каком еще методе, который вызвал нас? Кто-нибудь это понял и может пояснить?

Ответить +4

Дмитрий Оносовский 11 уровень, Одесса 11 февраля, 23:45

Тогда такая же проверка, т.е. поиск блока catch, выполняется в предыдущем из стека методе. Если нету, тогда еще в предыдущем. И так до main метода.

Ответить +1

Радхараман Скороход 14 уровень 15 марта, 20:02

Этот метод

Ответить 0

vlad 11 уровень 7 ноября 2017, 17:24

Фразу:
"Представь, что после вызова каждого метода мы проверяем: завершился ли только что вызванный метод сам по себе или в результате исключения. Если исключение было, тогда мы переходим на исполнение блока catch, если он есть, и захватываем исключение. Если блока catch нет, то завершаем и текущий метод. Тогда такая же проверка начинается в том методе, который вызвал нас."
Я бы перефразировал:
"Представь, что после вызова каждой КОМАНДЫ метода мы проверяем: завершился ли только что вызванная КОМАНДА метода сама по себе или в результате исключения. Если исключение было, тогда мы НЕ ВЫПОЛНЯЕМ ОСТАВШИЕСЯ КОМАНДЫ В МЕТОДЕ И переходим на исполнение блока catch, если он есть, и захватываем исключение. Если блока catch нет, то такая же проверка начинается в том методе, который вызвал нас. ОПЯТЬ ЖЕ ВСЕ КОМАНДЫ ПОСЛЕ ТОЧКИ ВЫЗОВА НАШЕГО МЕТОДА НЕ ВЫПОЛНЯЮТСЯ, А СРАЗУ ИЩЕТСЯ БЛОК CATCH.
Хорошая статья по исключениям с примерами кода, из которого все становится ясно:
https://habrahabr.ru/company/golovachcourses/blog/223821/

Ответить +4

Orion 22 уровень 24 августа 2017, 02:55

Для тех кому не совсем понятно зачем это нужно - в библиотеке Java полно классов, которые не скомпиллируюстя если не поймать их исключения. Можно конечно отделаться выводом Stack Trace, но обработать их всё равно придётся.

Ответить +3

Роман Яковлев 15 уровень, Москва 17 августа 2017, 15:37

ввели бы уже значение: бесконечность. Любые манипуляции с этим числом равно: бесконечность)

Ответить 0

Павел Ляхов 13 уровень, Санкт-Петербург 5 октября 2017, 05:36

При предельном переходе быть может.
А из того, что абсолютно любое число, будь то действительное или комплексное, будь то матрица или тензор - при умножение на 0 даст, офк, 0, можно заключить, что при арифметическом деление на 0 может быть абсолютно любая НЁХ, будь то целое число или же различного рода сложноструктурированные мат. объекты, хоть два гладиолуса

Но на некоторых калькуляторах замечал, что если делить на ноль, то будет выводить ошибку типа "ERR: inf"

Ответить +1

Константин 12 уровень 14 ноября 2017, 18:48

Вы не перепутали случайно деление на ноль, и деление нуля на ноль, которое дает NaN?
Ваши выкладки:
0*a=0; a=0/0; Для любого a.
В случае деления на ноль a/0=x; a = x*0; И под пределом бы тут была бесконечность, а не любое число. Хотя бесконечности тоже разные бывают...

Ответить 0

Павел Ляхов 13 уровень, Санкт-Петербург 10 декабря 2017, 23:25

Не уверен. Для меня что деление на ноль, что деление нуля на ноль - действия, арифметически невозможные и узаконенные лишь в предельных переходах. А дальше лишь философствования: если ты при помощи умножения на ноль можешь "обнулить" любой объект, то противоположная функция, должна воссоздавать любой объект из ничего, а на счет все же деления нуля на ноль, то это даже в пределах считается неопределенностью, а, значит, результат операции 0/0 должен уходить в мистику, наверное, хотя может и так же любой мат. объект.

Ответить +1

Юрій Якимчук 16 уровень 14 августа 2017, 15:48

Я 12 лет занимался коммерческой разработкой, до первого написания try catch :)

Ответить +4

Донат Ощепков 11 уровень 22 июля 2017, 12:51

А если исключение возникнет в блоке catch?

Ответить +8

Movsar Khuchbarov 16 уровень 24 июля 2017, 10:08

этому нас жизнь не учила

Ответить +21

Abomioff 16 уровень 28 июля 2017, 09:46

нужно постараться, чтоб там появилось исключение - обычно там простой вывод на экран ошибок. Но если таки исключение возникнет - программа вылетит с ошибкой, как в принципе и в любой ситуации без try. Особые параноики могут в catch добавить еще один try/catch, и так до бесконечности))

Ответить +13

Макс Куркудюк 25 уровень, Киев 3 февраля, 12:39

Особые параноики)))

Ответить +1

NEMEHANIK 10 уровень 18 июля 2017, 12:01

)))))"...Сам такого не напишу, конечно, но если в коде встречу – пугаться не буду..."

Ответить +3

Dmitry Kaltovich 35 уровень, Минск 6 июля 2017, 15:56

"И падет мир от деления на ноль, и придет сатана на землю грешную..."
(С) Настрадамус Иван Иваныч

Ответить +1

Bockser 27 уровень 18 апреля 2017, 11:40

Про порядок блоков катч не сказали, все родители должны быть ниже потомков - иначе исключительная ситуация для потомка будет перехватываться родителем

Ответить +16

Vladislav 40 уровень, Москва 26 апреля 2017, 13:12

+1

Ответить 0

Сергей Зимовец 31 уровень 17 мая 2017, 18:40

согласен если бы у Шилдта это не вычитал бы то и не знал бы , в принципе паралельно с сайтом нужно либо читать текущие темы или видеолекции смотреть или все вместе так как дофига нюансов упускается !!!!

Ответить +1

Николай Ковчуго 28 уровень, Минск 30 мая 2017, 12:00

Сам шилдта читаю. Правда у него трудно воспринимается часть II полного руководства.

Ответить 0

Альбина Исмаилова 13 уровень, Томск 22 июня 2017, 11:28

Если что, тут целая лекция этому посвящена - далее, а тут пока краткое знакомство)

Ответить 0

Валерий Бойко 23 уровень, Киев 28 июня 2017, 15:55

Привет! Скажи пожалуйста название книги Шилдта которую читал по Java и дай небольшой фидбек, если не сложно (как читается, нравится ли, много ли чего объясняет книга). Заранее спасибо)

Ответить 0

Georgy Naumov 19 уровень, Москва 21 июля 2017, 11:23

"Урезанный" Шилдт (Руководство для начинающих) - лучшая первая книга для изучения Java. IMHO. Кто-то её ругает, кто-то рекомендует выкинуть на свалку и читать "полную" версию. Мои аргументы:

1. Фактически - это азбука Java. Если не воспринимать книгу как полноценный учебник по языку - толку от неё будет в разы больше.
2. Построчное, побуквенное разжевывание примеров. Акцентирование внимания на новых или важных участках кода.
3. Упражнения. Их не очень много, но они есть. И в конце книги для них даются решения.
4. Пожалуй, лучший порядок следования тем по главам. Претензии есть буквально в паре моментов. Но, в основной части - материал подается очень последовательно и методично.
5. Ошибки. Они есть, их немного. Но они очевидны. И поймав самостоятельно такую ошибку, ловишь себя на мысли, что уже что-то понимаешь. Вот такой не очевидный способ мотивации.

После того, как Шилдт прочитан, все примеры и упражнения пройдены - можно идти дальше, база будет уже вполне себе.

Ответить +13

Sprw 29 уровень 24 июля 2017, 16:46

спасибо за развернутый ответ!

Ответить 0

Роман 11 уровень 11 марта, 09:06

А "Head First Java" для первой не пойдет?
Где-то читал интересную статью, где программист сравнивал две книги. "Поругал" Шилдта и порекомендовал прочитать Хорстмана "Библиотека профессионала". Есть какой фидбэк на эту книгу?

Ответить 0

Загрузить еще

ɹsh.ru/) (https://plus.google.com/114772402300089087607) (https://twitter.com/javarush_ru)

JAVARUSH

Программистами не рождаются
© 2018