Вопросы по теме Перехват и обработка исключений:

1. Какие существуют способы обработки исключений?
2. Как принудительно выбросить исключение?
3. Можно/нужно ли обрабатывать ошибки JVM?
4. В чем особенность блока finally? Всегда ли он исполняется?
5. Может ли не быть ни одного блока catch при отлавливании исключений?
6. Могли бы вы придумать ситуацию, когда блок finally не будет выполнен?
7. Может ли один блок catch отлавливать несколько исключений?
8. Что такое try-catch-finally?
9. Что такое try-with-resources?
10. Как лучше всего освобождать ресурсы?
11. Может ли конструктор вызывать исключения?
12. Есть метод, который может выбросить IOException и FileNotFoundException. В какой последовательности должны идти блоки catch? Сколько блоков catch будет выполнено?
13. В блоке try возникло исключение и выполнение переместилось в блок finally. В блоке finally тоже возникло исключение. Какое из исключений будет показано.
14. Если оператор return содержится и в блоке catch и в finally, какой из них “главнее”?

Перехват и обработка исключений.

Существует два способа обработки исключений:

1. обработка с помощью блока try-catch;
2. проброс исключения в сигнатуре метода (для не конечного пользователя).

Эти два способа можно комбинировать, часть исключений обработать с помощью try-catch, а остальные пробросить дальше.

Ключевые слова для обработки исключений:

1. **try** – обозначает блок кода в котором может произойти ошибка. Не может существовать без блока catch и / или finally.
2. **catch** – обозначает блок кода, который перехватывает и обрабатывает исключение. Таких блоков может быть сколько угодно. Также в одном блоке можно перехватывать несколько типов исключений (при условии, что они принадлежат разным иерархиям наследования). Может отсутствовать в блоке try, если присутствует блок finally.
3. **finally** – обозначает блок кода, который выполняется всегда, в не зависимости от того, произошла ошибка или нет, исключение – аварийное завершение работы программы (например, пропало питание ПК, умерло JVM или в блоке try / catch использовали системный выход из программы System.exit()). Обычно используется для закрытия открытых ресурсов (например, закрыть файл или соединение). Может отсутствовать в блоке try, если присутствует блок catch.
4. **throw** – принудительная генерация исключения (throw new Exception).
5. **throws** – используется в сигнатуре метода для обозначения исключений, которые могут возникнуть в данном методе.

Виталий Жуков
unicraft1989@gmail.com
Ссылка на курс Stepik

Перехват нескольких исключений:

```java
public class Main {
    public static void main(String[] args) {
        try {
            System.out.println(division( dividend: 5,  divider: 0));
        } catch (ArithmeticException e) {
            System.out.println("На ноль делить нельзя");
        } catch (Exception e) {
            System.out.println("Неизвестная ошибка");
        }

        // на экран будет выведено: На ноль делить нельзя

    }

    static int division(int dividend, int divider) {
        return dividend / divider;
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        try {
            System.out.println(division( dividend: 5,  divider: 0));
        } catch (Exception e) {
            System.out.println("Неизвестная ошибка");
        } catch (ArithmeticException e) {
            System.out.println("На ноль делить нельзя");
        }

        // не скомпилируется, т.к.
        // ArithmeticException наследник Exception

    }

    static int division(int dividend, int divider) {
        return dividend / divider;
    }
}
```

С блоком try можно использовать несколько блоков catch, каждый из которых может перехватывать свой тип исключений. Однако в этом случае важна последовательность указания типов исключений. Исключение-наследник должно идти перед исключением-родителем. Это связано с тем, что исключение-родитель будет ловить все типы исключений, которые от него наследуются. Например, если мы укажем catch(Exception e), то такой блок будет ловить абсолютно любое исключение (не считая ошибок Error) (т.к. все ошибки наследуются от класса Exception) и в цепочке блоков catch должен стоять последним.

В одном блоке catch можно указывать несколько типов исключений, если эти типы не принадлежат одной иерархии наследования. Если указать исключение-родитель и исключение-наследник, то код не скомпилируется. Придется удалить исключение-наследник.

catch (ArithmeticException | NullPointerException e) { }

Конструкция try-with-resources.

Начиная с Java 7 правильным закрытием ресурсов считается конструкция try-with-resources, а не блок finally. При использовании блока finally может случится потеря исключения, возникшего в основном коде. Если основной код выбросит исключение и код в блоке finally тоже выбросит исключение, то будет показано только исключение из блока finally. Конечно можно в блоке finally написать конструкцию try-catch и попытаться исправить эту ситуация, но такой код будет очень громоздким и нечитаемым.

```java
try(InputStream is = new FileInputStream( name: "path")) {

} catch (IOException e) {
    e.printStackTrace();
}
```

В конструкцию try-with-resources можно передавать только объекты классов, реализующих интерфейс AutoCloseable. Этот интерфейс содержит всего один метод close(), который используемый класс должен реализовать.

Вопрос: Можно/нужно ли обрабатывать ошибки JVM?

Ответ: Ошибку JVM можно отлавливать так же, как и любую другую ошибку, с помощью блока catch. Однако делать это нельзя. Ошибка класса Error (и наследники) означают, что JVM находится в неисправном состоянии, и ее дальнейшая работа непредсказуема.

Вопрос: Может ли конструктор вызывать исключения?

Ответ: Да, может. Например, FileInputStream в своем конструкторе пробрасывает исключение FileNotFoundException.

Вопрос: Если оператор return содержится и в блоке catch и в finally, какой из них “главнее”?

Ответ: Сработает return из блока finally.

Вопрос: В блоке try возникло исключение и выполнение переместилось в блок finally. В блоке finally тоже возникло исключение. Какое из исключений будет показано.

Ответ: Если присутствует блок catch, то будут показаны оба исключения. Если блок catch отсутствует, то будет показано исключение из блока finally. А исключение из блока try потеряется.

Вопрос: Есть метод, который может выбросить БлаБлаException и БлаБлаБлаException. В какой последовательности должны идти блоки catch? Сколько блоков catch будет выполнено?

Ответ: Блоки должны идти от исключения-наследника, к исключению-предку. Будет выполнен только один блок catch, который словит ошибку своего типа.

Виталий Жуков
unicraft1989@gmail.com
Ссылка на курс Stepik