**Как работает множественный catch**

[Java Syntax](https://javarush.ru/quests/QUEST_JAVA_SYNTAX)

[Уровень 9](https://javarush.ru/quests/lectures?quest=QUEST_JAVA_SYNTAX&level=9), Лекция 7



— И ещё немного интересных уроков. Мне так нравится преподавать.

— Хочу рассказать тебе, как работает множественный **catch**. Все очень просто: при возникновении исключения в блоке **try**, выполнение программы передаётся на первый **catch**.

— Если тип, указанный внутри круглых скобок блока catch, совпадает с типом объекта-исключения, то начинается выполнение кода внутри блока {}. Иначе переходим к следующему **catch**. Там проверка повторяется.

— Если блоки **catch** закончились, а исключение так и не было перехвачено, то оно выбрасывается дальше, а текущий метод аварийно завершается.

— Ясно. Будет выполнен тот catch, тип в котором совпадает с типом исключения.

— Да, верно. Но в реальности все немного сложнее. Дело в том, что классы можно наследовать друг от друга. И если класс «Корова» унаследовать от класса «Животное», то объект типа «Корова» можно хранить не только в переменной типа «Корова», но и в переменной типа «Животное».

— И?

— Т.к. все исключения унаследованы от классов **Exception** или **RuntimeException** (который тоже унаследован от **Exception**), то их все можно перехватить командами catch **(Exception** e) или **catch (RuntimeException e)**.

— И?

— Отсюда два вывода. Во-первых, с помощью команды catch(Exception e) можно перехватить любое исключение вообще. Во-вторых — порядок блоков catch имеет значение.

Примеры:

— Возникший при делении на 0 ArithmeticException будет перехвачен во втором catch.

Код

try

{

System.out.println("Before method1 calling.");

int a = 1 / 0;

System.out.println("After method1 calling. Never will be shown.");

}

catch (NullPointerException e)

{

System.out.println("Reference is null. Exception has been caught.");

}

catch (ArithmeticException e)

{

System.out.println("Division by zero. Exception has been caught.");

}

catch (Exception e)

{

System.out.println("Any other errors. Exception has been caught.");

}

— В примере ниже возникший ArithmeticException будет перехвачен в первом catch, т.к. классы всех исключений унаследованы от Exception. Т.е. Exception захватывает любое исключение.

Код

try

{

System.out.println("Before method1 calling.");

int a = 1/0;

System.out.println("After method1 calling. Never will be shown.");

}

catch (Exception e)

{

System.out.println("Any other errors. Exception has been caught.");

}

catch (NullPointerException e)

{

System.out.println("Reference is null. Exception has been caught.");

}

catch (ArithmeticException e)

{

System.out.println("Divided by zero. Exception has been caught.");

}

— В примере ниже исключение ArithmeticException не будет перехвачено, а будет выброшено дальше в вызывающий метод.

Код

try

{

System.out.println("Before method1 calling.");

int a = 1/0;

System.out.println("After method1 calling. Never will be shown.");

}

catch (NullPointerException e)

{

System.out.println("Reference is null. Exception has been caught.");

}

— Ну, вроде все понемногу проясняется. Непростая штука эти исключения.

— Это только кажется так. На самом деле – это чуть ли не самая простая вещь в Java.

— Не знаю, радоваться по этому поводу или огорчаться…