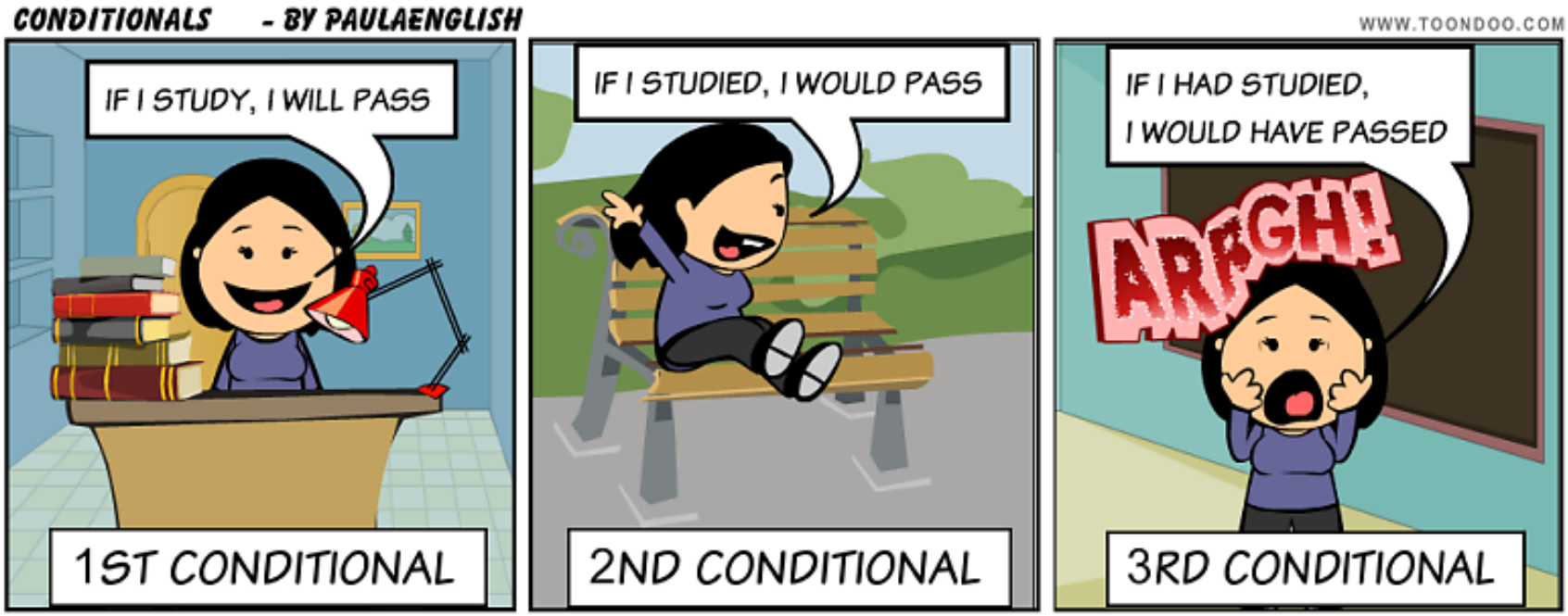
Управляющие конструкции и исключения в Java

Алексей Владыкин

7 октября 2013

1. Условные операторы
2. Циклы
3. Исключения


CONDITIONALS - BY PAULAENGLISH
WWW.TOONDOO.COM
IF I STUDY, I WILL PASS
1ST CONDITIONAL
IF I STUDIED, I WOULD PASS
2ND CONDITIONAL
IF I HAD STUDIED,
I WOULD HAVE PASSED
ARRGH!
3RD CONDITIONAL

Оператор if

```
if (weatherIsGood) {
    walkInThePark();
} else {
    readBooksAtHome();
}
```

- Есть вариант без `else`
- Условие типа `boolean`, нельзя написать `if (i = 3) { ... }`
- Фигурные скобки не обязательны, если внутри один оператор

Оператор ?:

```
if (weatherIsGood) {
    System.out.println("Weather is good");
} else {
    System.out.println("Weather is bad");
}

// same effect, but much shorter
System.out.println("Weather is "
        + (weatherIsGood ? "good" : "bad"));
```

Оператор switch

```
switch (digit) {
    case 0:
        text = "zero";
        break;

    case 1:
        text = "one";
        break;

    // case 2 - case 9

    default:
        text = "???";
        break;
}
```

- Без `break` исполнение продолжается
- Работает для примитивных типов `byte`, `short`, `char`, `int`, а также для `enum`
- В Java 7 добавлен `switch` для `String`

```
if (digit == 0) {
    text = "zero";
} else if (digit == 1) {
    text = "one";
} else if (digit == 2) {
    text = "two";
} /* 3 - 9 */
else {
    text = "???";
}
```


I will not waste chalk
I will not waste chalk
I will not waste chalk
I will not waste chalk
I will not waste chalk
I will not waste chalk
I will not waste chalk
I will not waste chalk
I will not waste chalk
I will not waste chalk

Цикл while

```
while (haveTime() && haveMoney()) {
    goShopping();
}
```

- Цикл с предусловием
- Условие типа boolean
- Фигурные скобки не обязательны, если внутри один оператор

Цикл do while

```
do {
    goShopping();
} while (haveTime() && haveMoney());
```

- Цикл с постусловием
- Нужна точка с запятой в конце
- Фигурные скобки не обязательны, если внутри один оператор

Цикл for

```
for (int i = 0; i < args.length; i++) {
    System.out.println(args[i]);
}
```

- Все части заголовка не обязательны
- `for (;;)` — бесконечный цикл
- Фигурные скобки не обязательны, если внутри один оператор

Цикл foreach

```
for (String arg : args) {
    System.out.println(arg);
}
```

- Добавлен в Java 5
- Применим к массивам и классам, реализующим интерфейс `java.lang.Iterable`

Оператор break

```
for (String s : haystack) {
    if (needle.equals(s)) {
        found = true;
        break;
    }
}
```

- Передает управление на следующий за циклом оператор
- Применим ко всем видам циклов

Оператор continue

```
for (String s : haystack) {
    if (!needle.equals(s)) {
        continue;
    }
    count++;
}
```

- Прерывает текущую итерацию цикла и начинает следующую
- Перед новой итерацией проверяется условие цикла
- Применим ко всем видам циклов

Метки

- Операторы `break` и `continue` действуют на ближайший цикл
- Можно указать другой цикл при помощи метки

```
outer:
for (int[] row : matrix) {
    for (int x : row) {
        if (x > 100) {
            break outer;
        }
    }
}
```


© 17/03/04 Interwizard2001
Windows 95
An error occured whilst trying to load the application, "bluescreen.exe"
Error Error
An error occured whilst trying to load the previous error.
OK
A fatal e
VXB UMM (82)
88018E35
* Press an
* Press CT
. You will
lose any
* Buy a Mac you fucktart.
Press any key to continue _
Hey, it looks like you're having an error!

Что такое «исключение»

- Исключение (exception) — событие, возникающее в процессе работы программы и прерывающее её нормальное исполнение

Примеры:

- `java.lang.NullPointerException`
- `java.lang.ArrayIndexOutOfBoundsException`
- `java.lang.ClassCastException`
- `java.lang.ArithmeticException`
- `java.lang.OutOfMemoryError`
- `java.io.IOException`

java.lang.Throwable

- Исключение в Java — полноценный объект
- Все исключения в Java наследуются от класса Throwable

- `String getMessage()`
- `Throwable getCause()`
- `StackTraceElement[] getStackTrace()`
- `void printStackTrace()`

Stack Trace

```
java.lang.NullPointerException
    at ru.compscicenter.java2013.Test.baz(Test.java:19)
    at ru.compscicenter.java2013.Test.bar(Test.java:14)
    at ru.compscicenter.java2013.Test.foo(Test.java:10)
    at ru.compscicenter.java2013.Test.main(Test.java:6)
```

Классификация исключений

- Исключительные ситуации в JVM
  `java.lang.Error`
- Исключительные ситуации в пользовательском коде
  - Проверяемые (checked)
    `java.lang.Exception`
  - Непроверяемые (unchecked)
    `java.lang.RuntimeException`

Пользовательские исключения

```
public class InvalidUserException
        extends Exception {

    public InvalidUserException() {
        super("Please replace user and continue");
    }
}
```

Выброс исключения

```
// from java.lang.Integer
public static int parseInt(String s, int radix)
        throws NumberFormatException {
    if (s == null) {
        throw new NumberFormatException("null");
    }
    // code skipped
}
```

- Оператор `throw` прерывает нормальное исполнение программы и запускает поиск обработчика исключения
- Если исключение проверяемое, метод должен содержать его в списке `throws`

Перехват исключения: try-catch

```
System.out.print("Please enter number: ");
int n = 0;
while (true) {
    String s = readUserInput();
    try {
        n = Integer.parseInt(s);
        break;
    } catch (NumberFormatException e) {
        System.out.print(
                "Bad number, try again: ");
    }
}
```

Перехват нескольких исключений

```
try {
    // ...
} catch (IOException e) {
    e.printStackTrace();
} catch (NumberFormatException e) {
    e.printStackTrace();
}

// this works since Java 7:
try {
    // ...
} catch (IOException | NumberFormatException e) {
    e.printStackTrace();
}
```

Обработка исключения

- Если в коде вызываются методы, бросающие проверяемые исключения, эти исключения надо либо поймать и обработать (`catch`), либо добавить в список `throws`

Стратегии обработки:

- Игнорирование (пустой `catch`)
- Запись в лог
- Проброс дальше того же или нового исключения
- Содержательная обработка (например, повтор операции)

Исключения и освобождение ресурсов

```
InputStream is = new FileInputStream("a.txt");
try {
    readFromInputStream(is);
} finally {
    is.close();
}
```

- Блок `finally` будет выполнен в любом случае
- В нем обычно освобождают использованные ресурсы

try с ресурсами

```
try (InputStream is =
        new FileInputStream("a.txt")) {
    readFromInputSteam(is);
}
```

- Добавлен в Java 7
- Метод `close()` будет вызван автоматически, как в `finally`
- Можно перечислить сразу несколько ресурсов
- Ресурсы должны реализовать интерфейс `java.lang.AutoCloseable`

Гарантии безопасности

- Гарантии отсутствия исключений
- Сильные гарантии
- Слабые гарантии
- Гарантия отсутствия утечек
- Никаких гарантий

Что сегодня узнали

- Как в Java написать ветвление и цикл,
  какие есть разновидности
- Что такое «исключение», какие бывают типы исключений
- Как пользоваться исключениями