Стандарты оформления кода JepRia-приложений

Оглавление

[Стандарты оформления кода JepRia-приложений 1](#_Toc454975884)

[1. Термины и сокращения 3](#_Toc454975885)

[2. Введение 3](#_Toc454975886)

[3. Соглашения оформления java кода 3](#_Toc454975887)

[3.1 Общие соглашения 3](#_Toc454975888)

[3.2 Фигурные скобки 3](#_Toc454975889)

[3.3 Методы 4](#_Toc454975890)

[3.4 Переменные 4](#_Toc454975891)

[3.5 Исключения 4](#_Toc454975892)

[3.6 Аннотации 4](#_Toc454975893)

[4. Регламент внесения изменений 5](#_Toc454975894)

[5. Источники 6](#_Toc454975895)

# Термины и сокращения

TBD

# Введение

Данный документ описывает соглашения по оформлению java кода JepRia-приложений. Соглашения имеют рекомендательный характер, подходят для использования в большинстве случаев. Стандартные ситуации необходимо оформлять одинаково. Если появляются специфичные случаи, то необходимо оставлять подробный комментарий, почему стандартное соглашение не подошло.

Данные соглашения необходимо использовать при написании прикладных модулей.

Некоторые соглашения поможет контролировать IDE, такие как: максимальная длина строки, символы отступов, максимальное количество строк в одной функции.

В качестве основы берем стандарт от компании sun: <http://www.oracle.com/technetwork/java/codeconventions-150003.pdf>

# Соглашения оформления java кода

## Общие соглашения

1. В качестве одного отступа использовать символ табуляции (не пробелы).
2. Стиль написание составных слов – CamelCase.
3. Максимальная длина одной строки – 120 символов.
4. Использовать соглашения в прикладных модулях – комментировать нестандартный функционал прикладных модулей.

## Фигурные скобки

1. Для открывающих фигурные скобок не выделяется отдельная строка, они находятся в той же строке, что и код перед ними:

                class MyClass {

                               int func() {

                                               if (something) {

                                                               // ...

                                               } else if (somethingElse) {

                                                               // ...

                                               } else {

                                                               // ...

                                               }

                               }

                }

1. Фигурные скобки всегда требуются для оператора условия. Исключением является, когда оператор условия и его тело помещаются в одну строку. То есть можно писать так:

                if (condition) {

                               body(); // ok

                }

                if (condition) body(); // ok

                Но так нельзя:

                if (condition)

                               body(); // bad

## Методы

1. Названия методов должны быть глаголами, первая буква должна быть строчной.
2. Должны быть короткими, и выполнять только одну задачу.
3. Имена методов должны быть самодокументированными.
4. Максимальное количество строк для одной функции – 40 строк. Если метод превышает 40 строк, то вам, возможно, стоит подумать о том, можно ли его разбить на части, не нарушив структуры программы.

## Переменные

1. Minimize scope of variables. Объявлять каждую переменную там, где она используется в первый раз.
2. Инициализация должна производиться, по возможности, сразу.

## Исключения

1. Не игнорируйте никакие исключения.
2. Не перехватывайте обобщенные исключения.

        try {

                       someComplicatedIOFunction();        // may throw IOException

                       someComplicatedParsingFunction();   // may throw ParsingException

                       someComplicatedSecurityFunction();  // may throw SecurityException

                       // phew, made it all the way

        } catch (Exception e) {               // I'll just catch all exceptions

                       handleError();                      // with one generic handler!

        }

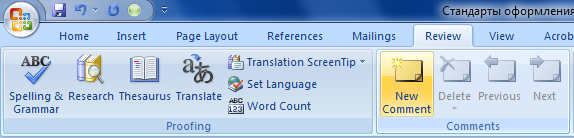
Вам не следует так делать. Суть в том, что возможно появление исключения, которого вы не ожидали и, в итоге, ошибка будет отлавливаться на уровне приложения. То есть, если кто-то добавит новый тип исключения, то компилятор не сможет вам помочь понять, что это другая ошибка.

## Аннотации

1. При переопределении родительского метода обязательно использовать аннотацию @Override.

# Регламент внесения изменений

В случае редактировании, удалении существующих соглашений в документе необходимо оставлять комментарий и инициировать дискуссию по согласованию изменений.



# Источники

1. Стандарт, -  <http://www.oracle.com/technetwork/java/codeconventions-150003.pdf>
2. Совершенный код (Code complete). Автор: Стив Макконнелл.
3. <https://habrahabr.ru/post/112042/>