Code Conventions

(for the Java Programming Language)

Зачем?

- 80% времени жизни ПО находится в поддержке
- Почти никогда ПО не поддерживается автором
- Соглашение об именовании улучшает читаемость кода, ускоряя пониманиее нового кода
- При поставке кода как продукта необходимо убедиться, что код чист и правильно упакован
- http://java.sun.com/docs/codeconv/

Файлы

- Исходные файлы - .java
- Скомпилированный байт-код - .class
- 1 класс – 1 файл

Структура файла

- Комментарий в начале
  - Имя класса, версия, дата, копирайт
- Указания package и import
- Описание класса/интерфейса
  - Описание класса/интерфейса
  - class или interface
  - Поля класса (static)
  - Поля
  - Конструкторы
  - Методы

Отступы

- Длина строки не должна превышать 80 символов
- Отступ 4 пробела
- Переносы строк
  - После запятой
  - Перед операцией
  - Предпочтительно более высокоуровневый перенос
  - Согласовать отступ с началом выражения
  - Если предыдущее правило дает плохой результат, сделать отступ 8 пробелов

Пример отступа

- Пример

```
someMethod(longExpression1, longExpression2, longExpression3,
        longExpression4, longExpression5);

var = someMethod1(longExpression1,
                  someMethod2(longExpression2,
                              longExpression3));
```

- Выберите

```
longName1 = longName2 * (longName3 + longName4 - longName5)
            + 4 * longname6;

longName1 = longName2 * (longName3 + longName4
                         - longName5) + 4 * longname6;
```

Пример отступа

- Пример

```
someMethod(longExpression1, longExpression2, longExpression3,
        longExpression4, longExpression5);

var = someMethod1(longExpression1,
                  someMethod2(longExpression2,
                              longExpression3));
```

- Выберите

```
longName1 = longName2 * (longName3 + longName4 - longName5)
            + 4 * longname6;

longName1 = longName2 * (longName3 + longName4
                         - longName5) + 4 * longname6;
```

Пример отступа (2)

- Пример

```
if ((condition1 && condition2)
    || (condition3 && condition4)
    ||!(condition5 && condition6)) {
    doSomethingAboutIt();
}

if ((condition1 && condition2)
        || (condition3 && condition4)
        ||!(condition5 && condition6)) {
    doSomethingAboutIt();
}

if ((condition1 && condition2) || (condition3 && condition4)
        ||!(condition5 && condition6)) {
    doSomethingAboutIt();
}
```

Пример отступа (2)

- Пример

```
if ((condition1 && condition2)
    || (condition3 && condition4)
    ||!(condition5 && condition6)) {
    doSomethingAboutIt();                  // эту строку просто не заметить
}

if ((condition1 && condition2)
        || (condition3 && condition4)
        ||!(condition5 && condition6)) {
    doSomethingAboutIt();
}

if ((condition1 && condition2) || (condition3 && condition4)
        ||!(condition5 && condition6)) {
    doSomethingAboutIt();
}
```

Пример отступа (3)

- Форматирование троичного оператора

```
alpha = (aLongBooleanExpression) ? beta : gamma;


alpha = (aLongBooleanExpression) ? Beta
                                 : gamma;


alpha = (aLongBooleanExpression)
        ? beta
        : gamma;
```

Комментарии

- Блочные (/* … */)

```
/*
 * Here is a block comment.
 */
```

- Однострочные (// …)

```
if (condition) {

    /* Handle the condition. */
    …
}
```

- Javadoc (/** … */)

Определения

- Рекомендуется одно определение на строку

```
int level; // indentation level
int size;  // size of table
```

- Обязательно один тип в строке!

```
int foo,  fooarray[]; //Плохо!
```

- Объявления переменных только в начале блока
  - Кроме for (int i = 0; i < maxLoops; i++) { ... }
- Вариант форматирования:

```
int    level;         // indentation level
int    size;          // size of table
Object currentEntry; // table entry
```

- Избегайте скрытие переменных, определенных выше!

Описание класса

- Нет пробела между именем метода и “(“
- Открывающая "{" в конце строки определения
- Закрывающая "}" на отдельной строке с отступом соответствующим определению (кроме пустого блока “{ }“)
- Методы разделяются пустой строкой

```
class Sample extends Object {
    int ivar1;
    int ivar2;

    Sample(int i, int j) {
        ivar1 = i;
        ivar2 = j;
    }

    int emptyMethod() {}
    ...
}
```

Операторы

- Не более одного оператора в строке
- Составной оператор ("{ statements }")
  - Вложенные операторы сдвинуты на уровень больше
  - Открывающая скобка – на строке с оператором, закрывающая – на отдельной строке без сдвига
  - Фигурные скобки используются даже для одного вложенного оператора (в if-else, for и т.п.)
- Значение return пишется, по возможности, без скобок

```
return;
return myDisk.size();
return (size ? size : defaultSize);
```

if, if-else, if else-if else

- if-else форматируется так:

```
if (condition) {
    statements;
}

if (condition) {
    statements;
} else {
    statements;
}

if (condition) {
    statements;
} else if (condition) {
    statements;
} else {
    statements;
}
```

Циклы

- for

```
for (initialization; condition; update)
{
    statements;
}
```

- Пустой for

```
for (initialization; condition; update);
```

- while

```
while (condition) {
    statements;
}
```

- или

```
while (condition);
```

- do-while

```
do {
    statements;
} while (condition);
```

switch

- Оператор switch

```
case ABC:
    statements;
    /* falls through */

case DEF:
    statements;
    break;

case XYZ:
    statements;
    break;

default:
    statements;
    break;
}
```

try-catch

- Оператор try-catch

```
try {
    statements;
} catch (ExceptionClass e) {
    statements;
}
```

- С блоком finally

```
try {
    statements;
} catch (ExceptionClass e) {
    statements;
} finally {
    statements;
}
```

WhiteSpace

- Пустые строки
  - Между секциями файла
  - Между описаниями классов и интерфейсов
  - Между методами
  - Между локальными переменными и операторами
  - Перед комментарием
  - Между логическими секциями в методе
- Пробел
  - Между ключевым словом и скобкой
  - После запятой (в списке аргументов)
  - Вокруг бинарных операторов, кроме “.”
  - Выражения в цикле for

Имена

Пакеты	`com.sun.eng`
	`com.apple.quicktime.v2`
	`edu.cmu.cs.bovik.cheese`

Классы	`class Raster;` `class ImageSprite;`
Интерфейсы	`interface RasterDelegate;` `interface Storing;`

Методы	`run();`
	`runFast();`
	`getBackground();`

Переменные	`int i;`
	`char c;`
	`float myWidth;`

Константы	`static final int MIN_WIDTH = 4;`
	`static final int MAX_WIDTH = 999;`
	`static final int GET_THE_CPU = 1;`

Разное

- Не делайте поля класса доступными (public) без веской на то причины
- Не используйте объект для доступа к методу класса (static)
- Литералы (кроме, иногда, 1, 0, -1) не должны встречаться в коде
- Избегайте двойных присваиваний

```
fooBar.fChar = barFoo.lchar = 'c';
```

- В сложных выражениях стоит ставить скобки

```
if ((a == b) && (c == d)) ...
```