|  |
| --- |
| ИнтерТраст |
| Стандарты кодирования |
|  |

|  |
| --- |
| Митавский Д. В. |

Содержание

[Стандарты кодирования CM5 2](#_Toc356490928)

[Стандарты 2](#_Toc356490929)

[Форматирование кода 2](#_Toc356490930)

[Именование сущностей 2](#_Toc356490931)

[Кодировка файлов 3](#_Toc356490932)

[Комментарии 3](#_Toc356490933)

[Организационные вопросы 3](#_Toc356490934)

[Настройка IDE 4](#_Toc356490935)

[Eclipse 4](#_Toc356490936)

[IntelliJ Idea 5](#_Toc356490937)

# Стандарты кодирования CM5

Этот документ описывает стандарты кодирования, используемые в проекте CM5. Стандарты должны неукоснительно соблюдаться всеми разработчиками. Данные стандарты во многом соответствуют конвенции кодирования Java, принятой в Oracle (<http://www.oracle.com/technetwork/java/codeconv-138413.html>), однако некоторые моменты, не описанные, остаются на усмотрение разработчиков.

## Стандарты

### Форматирование кода

1. Код не должен содержать символы табуляции.

Вместо символов табуляции используются отступы в 4 пробела (легко настраивается в любой современной среде разработки).

1. Одна строка кода должна содержать не более одного логического выражения или объявления переменной.
2. Максимальная длина строки кода – 120 символов.
3. Тело условных операторов (кроме тернарных) и операторов циклов должно быть заключено в фигурные скобки, даже если тело представляет собой единственную строку кода. Открывающая фигурная скобка находится в конце строки, открывающей условный оператор, закрывающая - на строке, следующей за последней строкой тела условного оператора или цикла. Например, следует использовать следующую конструкцию:

**for** (**int** i = 0; i < 10; ++i) {

**if** (i == 5) {

**break**;

}

}

вместо

**for** (**int** i = 0; i < 10; ++i) **if** (i == 5) **break**;

1. При использовании типизированных коллекций параметры типов в инициализирующем выражении следует опускать в соответствии с возможностями JDK 1.7:

**ArrayList<String> list = new ArrayList<>();**

1. Порядок следования спецификаторов следующий:

**<access> static abstract synchronized <unusual> final native**

1. Пробелы. **TODO**

### Именование сущностей

1. Именование методов и атрибутов классов/интерфейсов соответствует спецификации JavaBeans. В частности, имена переменных и классов не должны содержать (в подавляющем большинстве случаев) символы подчеркивания для разделения слов. Вместо этого используется верхний регистр для начала нового слова в имени.
2. Префиксы и суффиксы (например, символы подчеркивания, или префиксы венгерской нотации) **не допускаются**. Примерами префиксов венгерской нотации являются: "m\_" в названии атрибутов классов или переменных (int m\_ivariable), "I" в названии интерфейсов (ISerializable), "С" в названии классов (CString). Более подробную информацию о венгерской нотации можно найти здесь [[1]](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%BD%D0%B3%D0%B5%D1%80%D1%81%D0%BA%D0%B0%D1%8F_%D0%BD%D0%BE%D1%82%D0%B0%D1%86%D0%B8%D1%8F).
3. Имена классов, интерфейсов и переменных должны представлять собой правильно составленные выражения на английском языке. Например, переменная zapisVBazeDannyh должна быть переименована в databaseRecord. Если возникают сомнения в правильности именования, необходимо проконсультироваться у коллег.
4. Имена констант и перечислений (enums) представляют собой слова (словосочетания), записанные в верхнем регистре и разделенные символом подчеркивания. Например,

**final** **int** MAX\_VALUE = 5;

### Кодировка файлов

1. Все файлы, содержащие исходный код, имеют кодировку UTF-8
2. property-файлы, использующиеся для локализации приложения, имеют кодировку ISO 8859-1. Символы юникода (например, русские буквы) представляются соответствующими escape-последовательностями (\uXXXX). Современные IDE позволяют выполнять редактирование подобных символов прозрачным для пользователя образом (**обсудить**)

### Комментарии

1. Каждый public, protected или package-local класс (метод, атрибут, константа) должен быть снабжен комментарием, достаточным для понимания
2. Комментарии в коде должны быть написаны на русском языке
3. Каждый файл исходного кода должен быть озаглавлен комментарием, определяющем правообладателя системы (**требуется определить**)

### Организационные вопросы

1. Новые папки создаются только после согласования с лицами, ответственными за структуру проекта

## Настройка IDE

### Eclipse

#### Подключение и настройка плагина Checkstyle

Плагин подсвечивает код, который не соответствует стандарту (по умолчанию желтым цветом). При наведении на выделенный код всплывает подсказка с описанием проблемы. Плагин является вспомогательным инструментом и его использование не является обязательным.

#### Установка плагина

На сайте разработчиков пошагово установка плагина.(<http://eclipse-cs.sourceforge.net/downloads.html>)

#### Настройка плагина

**Todo**

#### Настройка форматирования

##### Formatter

Window->Preferences->Java->Code Style->Formatter. New, вводим имя, жмём Ок. Открывается окно с закладками:

Identation: Tab Policy - Spaces Only, identation size - 4, tab size - 4. Устанавливаем все галочки кроме Align fields in columns, Empty Lines.

New Lines: устанавливаем все галочки кроме after labels, insert new line after opening brace of array initializer, insert new line before closing brace of array initializer.

Control Statements: устанавливаем галочку keep 'else if' on one line.

Line Wrapping: Maximum line width - 160, устанавливаем галочку never join already wrapped lines. В списке выбираем Expressions->Assigments и внизу в пункте Line wrapping policy выбираем wrap where neccessary. Аналогично для пунктов Annotations->Element value-pairs, Method Declarations->Declaration, 'enum' deaclararions->Constants. Во всех остальных настройках по умолчанию стоит перенос строк, если где-то не будет срабатывать, то надо посмотреть все остальные пункты в этом списке.

Comments: Устанавливаем галочку Enable Javadoc comment formatting. В блоке javadoc comment settings устанавливаем все галочки кроме blank line before javadoc tags, remove blank lines.

White Space: Заходим в блок Arrays->Arrays initializers и ставим галочки только напротив before opening brace, after comma, before closing brace, between empty braces.

Жмём ОК, в качестве активного профиля должен быть выбран только что созданный.

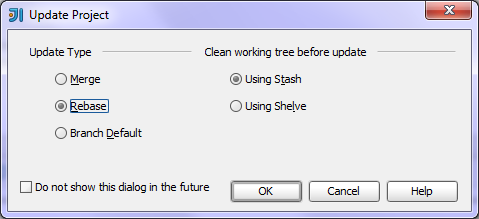
##### Text Editor

Window->Preferences->General->Editors->Text Editors. Устанавливаем все галочки, tab size - 4, print margin column - 160.

### IntelliJ Idea

#### Git Pull

Рекомендованный режим Git Pull – Rebase. Для того, чтобы его использовать, при скачивании изменений (Pull) достаточно выбрать соответствующий способ:



При попытке отправить изменения на сервер и наличии нескачанных обновлений, команда Pull будет выполнена автоматически и при этом будет предложен выбор метода в следующем диалоговом окне.

Если поставить галочки “Do not show this dialog in the future” в этих окнах, Idea будет применять выбранный способ способ и в дальнейшем.

Чтобы снова показывать данные диалоговые окна перед публикацией, необходимо зайти в меню Settings->Version Control->Confirmation и установить галочку “Update” в группе “Display options dialog when these commands are invoked”.