Real-time bridge monitoring
Team policy - Coding conventions

Version 1.2

Revision History

Date	Version	Description	Author
2013-10-24	1.0	Initial Draft	Marko Brcic
2013-10-28	1.1	Added conventions for OO, database and general	Marko Brcic
2013-11-11	1.2	Removed the scripting language in conventions, since we are using Java, updated naming conventions for database design	Marko Brcic

1. Introduction

1.1 Purpose of this document

The purpose of this document is to define the team policy for coding conventions.

1.2 Document organization

The document is organized as follows:

- Section 1, *Introduction,* describes contents of this guide, used documentation, intended audience, scope of the document and definitions and acronyms.
- Section 2, *Coding conventions,* describes the coding conventions that should be obeyed for each programming language or database environment used during the development on the Distributed Software Development course project

1.3 Intended Audience

The intended audience are team members.

- Andrea Bottoli
- Dzana Kujan
- Lorenzo Pagliari
- Nikola Radisavljevic
- Jörn Tillmanns
- Fifo Miraldi
- Marko Brcic
- Ghazal Shojaee
- Elisabetta di Nitto

1.4 Scope

This documents addresses the rules that need to be obeyed during the development phase of the project strictly regarding writing the code and naming conventions in the database and the source code.

1.5 Definitions and acronyms

1.5.1 Definitions

Keyword	Definitions

1.5.2 Acronyms and abbreviations

Acronym or abbreviation	Definitions

1.6 References

[0] Java code conventions, September 12, 1997. http://www.oracle.com/technetwork/java/codeconventions-150003.pdf

2. Coding conventions

Rule	Definition
Language	English!!!
Names of variables, properties, methods, columns, etc.	Meaningful names that precisely describe what the property/entity is about
Comments	Short comments mandatory! But don't comment something that is obvious from the code or name of the property/entity.
Layered applications	Use layered applications. Make layers for data, business logic, and presentation
Modular applications	Divide code into modules and functions, don't repeat the code
Tools for the project	Each member needs to use the agreed set of tools for development, so that setting up environment and projects goes easily and smoothly. Team members should use also the same versions of those tools

Table 1. General coding conventions

Type	Rule	Example
Variables	Except for variables, all instance, class, and class constants are in mixed case with a lowercase first letter. Variable names should be short yet meaningful. The choice of a variable name should be mnemonic— that is, designed to indicate to the casual observer the intent of its use. One-character variable names should be avoided except for temporary “throwaway” variables. Common names for temporary variables are i, j, k, m, and n for integers; c, d, and e for characters.	allResults
Methods	Methods should be verbs, in mixed case with the first letter lowercase, with the first letter of each internal word capitalized.	getResults()
Classes	Class names should be nouns, in mixed case with the first letter of each internal word capitalized. Try to keep your class names simple and descriptive. Use whole words—avoid acronyms and abbreviations (unless the abbreviation is much more widely used than the long form, such as URL or HTML)	School
Interfaces	Interface names should start with letter i	iSchool
Constants	The names of variables declared class constants	REFRESHING_PERIOD

	and of ANSI constants should be all uppercase with words separated by underscores ("_"). (ANSI constants should be avoided, for ease of debugging.)	
Enumerators	Enumerator names should start with letter e	ePeriodTypes

Referring to class variables and methods	Avoid using an object to access a class static variable or method	School.getAllSchools() //OK
		myS.getAllSchools() //AVOID

Constant values	All constant values should not be coded directly. Use constants or enumerators.	enum ePeriodTypes … REFRESHING_PERIOD
Parentheses	It is generally a good idea to use parentheses liberally in expressions involving mixed operators to avoid operator precedence problems. Even if the operator precedence seems clear to you, it might not be to others—you shouldn't assume that other programmers know precedence as well as you do.	`if((a == b) && (c == d))`

Returning values	Try to make the structure of your program match the intent.	if (booleanExpression) {
		return TRUE;
		} else {
		return FALSE;
		} //AVOID
		
		return booleanExpresion;

Table 2. OO language (Java) coding conventions

Type	Rule	Example
Tables	Nouns in plural	settings, measurements, user_roles...
Columns	Nouns in Singular, ID for PK	start_date, timestamp, ID, ...
Joint tables	Names from both tables	settings_measurements
Triggers	Verb describing the action, preferably the triggering action	after_measurements_insert
Functions	Verb describing the action, preferably including the return parameter	max_wind_speed
Procedures	Verb describing the action	calculate_readings
Constraints	Starting with PK, FK, etc.	FK_measurements_settings

Table 3. Database (MySQL) coding conventions

3. List of Tables