CODING CONVENTIONS

1. Introduction

1.1 PURPOSE

The goal of the document is to capture, a set of applied guidelines, based on common practice on Java community. The guidelines are built on information from the information from the following sources,

- Oracle Corporation-Code Conventions for the Java Programming Language
- Google INC. Google Java Style
- Kernighan and Ritchie -The Elements of Programming Style
- Experience of Team Members.

2. Naming Conventions

The following guidelines are based on Sun's Java Coding Conventions with additional idea from Google Java Style. The conventions derive from the *'Camel Case style'* in which a name is formed of multiple words that are joined together as a single word with the first letter of each of the multiple words capitalized so that each word that makes up the name can easily be read.

2.1 Package

Package names are all-lowercase, with consecutive words simply concatenated together (no underscores).

2.2 Class

Class names are written using UpperCamelCase and are generally noun or noun phrases. Interface also follow the same notation of a noun phrase or adjectives.
Test classes are named starting with the name of the class they are tested, and end with Test. For example, HashTest or HashIntegrationTest.

2.3 Methods

Method names follow lowerCamelCase and are typically verbs or verb phrases. JUnit Test Methods also follow the same notation of lowerCamelCase and generally start with the word 'test' succeeded by the tested method name.

2.4 Parameter Names

Parameter Names follow the lowerCamelCase and one-letter parameter names are typically avoided for public methods.

2.5 Constants

Constant names use CONSTANTCASE: all uppercase letters.

3. Source Structure

The Java Application should adopt the conventions, which are based on Sun Java Coding Conventions. This makes it easier for code reading and understandability.

3.1 Package Structure

The RISK application is organized into a set of JAVA packages, with common prefix '*com.soen.risk'* and then succeeds with corresponding layer or infrastructure, *com.soen.risk.*

(Or) *com.soen.risk.layer2.*

3.2 File Structure

The source file should be organized in the following order:
•package statement
•import statements
•public class or interface
•associated non-public classes and interfaces

Order of the Class structure

- JavaDoc Statements
- Class statements
- Variable and constant declarations
- Constructors(including duplicates)
- Methods

Constant and variable declarations in the below order

- Static constants
- Static variables
- Instance variables

3.3 Method Structure

Methods should be grouped by functionality rather than access rights. Each method should be preceded by JavaDoc comments. Modifiers are used, in the following order:

- public
- static
- abstract
- synchronized
- final

4. Indentation and Layout

The Coding Style follows the Kernighan and Ritchie style for braces of non-empty blocks and block like struts.

Each method has its opening brace at the next line on the same indentation level as its header, the statements within the braces are indented, and the closing brace at the end is on the same indentation level as the header of the method at a line of its own. The blocks inside a method, however, have their opening braces at the same line as their respective control statements; closing braces remain in a line of their own, unless followed by a keyword ***else*** or ***while***

Each time a new block or block-like construct is opened, the indent increases by two spaces. When the block ends, the indent returns to the previous indent level. The indent level applies to both code and comments throughout the block.

Multiple assignments are avoided. I.e. x=y=45

Refrain from using embedded assignments. I.e. X=y+(z.getNext())

Omit parentheses around return argument, unless needed for clarity

5. Comments

All Java Source code should include Java Doc comments. JavaDoc should be present for every public class , and every public or protected class members.

Block Comment Style: They are indented as the same level as the code and are enclosed in /*…. */ style.

For Multi line statements, consecutive lines of comment blocks should start with * and aligned with other lines of * on all lines of the block.

Block tags: The tags are used in the order of @param, @return, @throws, @deprecated

Use the JavaDoc @throws tag to document checked and unchecked exceptions. Unchecked Exceptions are used for conditions that violate a precondition.
The @throws tag may therefore be used to document (the complement of) preconditions.

6. Import Statements

Import statements should not use the * wild cards for any package. Explicit imports of each class lead to much less confusion later when looking at code e.g. this makes the subsystems dependencies clearer.

7. Best Practices

- Logger:
  - Logging should be done used with the help of java.util.logging.Logger, by creating a Logger in each class.
  - System.out or System.err statements should not be used. Always log caught exceptions
- Enums should be used liberally instead of integer or String constants whenever possible.
- Use StringBuilder for repeated concatenation of Strings.
- Refer objects with their interfaces and not by their implementations.

8. REFERENCES

Code Conventions for the Java Programming Language: Contents: Technical Report: https://www.oracle.com/technetwork/java/index.html
Google Java Coding Style. https://google.github.io/styleguide/javaguide.html