SIF3 Framework Coding Style

This document summarises some coding styles that should be used when committing/contributing code to the SIF3 Framework on github. The guidelines are nothing new to any Java developer and standard Java conventions are used.

**Naming Convention**

<http://www.oracle.com/technetwork/java/codeconventions-135099.html>

**Formatting**

The list below summaries some code formatting that is being used within the SIF3 Framework. Provided with this document is a code formatting XML (SIF3FrameworkCodeStyle.xml) file that can be imported in the Eclipse IDE that will apply the format settings to be used otherwise you may want to apply the formatting listed below within you IDE of choice:

* Line indentation is 4 spaces (no tabs)
* Opening “{“ braces are always on a new line with the exception of an array where it is on the same line.
* No statement or statement fragment after a closing “}” brace, so an “else” of an if-statement is on a new line rather than after the closing “}” of the “if”.
* Opening and closing braces must be used even if a block is made up of a single statement only. For example and if-statement with only one statement still must have the {} braces. See example in Appendix A.
* Line wrapping is suggested to be 100 characters but these days of modern editors that is just a suggestion but not required to be enforced.

See “Appendix A” for an example of above formatting in relation to braces.

**Comments & Javadoc**

Javadoc style comments of the type **/\*\*** … \*/ should be used for the following:

* Class & Interface level
* All “public” methods.
* Optional on protected methods.
* All “public static final” variables (which are really constants).

Javadoc style comments of the type **/\*** … \*/ should be used for the following:

* Ideally on “private” methods.
* On protected methods if not commented with the /\*\* … \*/ style comment.

Standard Java line comments (//….) shall be used to comment code fragments.

Appendix A: Code Format Example

class Example

{

int[] myArray = { 1, 2, 3, 4, 5, 6 };

int[] emptyArray = new int[] {};

void bar(int p)

{

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

{

}

if (a > b)

{

setA(a); // single statement still must have {}

}

else // on new line

{

setA(b);

a = 0;

}

try

{

...

}

catch // on new line

{

...

}

switch (p)

{

case 0:

{

fField.set(0);

break;

}

case 1:

{

break;

}

default:

{

fField.reset();

} // end default

} // end switch

}

}