**NavUP**

**Instance Longsword**

**Team Integration**

**Implementation Coding Standards Document**

**Version 1.0**

Table of Contents

[Table of Contents 2](#_Toc479094163)

[Introduction 3](#_Toc479094164)

[Coding Standards 3](#_Toc479094165)

[Sections of the Coding Standards 3](#_Toc479094166)

[Class Purpose Description 3](#_Toc479094167)

[File Header 3](#_Toc479094168)

[Method Description 3](#_Toc479094169)

[In-code Comments 4](#_Toc479094170)

[Style 4](#_Toc479094171)

[Naming Conventions 4](#_Toc479094172)

[Layout Rules 4](#_Toc479094173)

[Clarity 5](#_Toc479094174)

# Introduction

The NavUP Longsword instance is made up of 7 teams representing Access, Data, GIS, Integration, Navigation, Notifications and User modules respectively. The teams make use of each other’s functionality and it is the duty of the Integration team to facilitate smooth operation between various teams. Hence, the purpose of this document is to outline a set of coding standards that should be adhered to by all teams in an aim to reach overall software consistency, integration and reuse. The standards have been specified in a way that will make it easier for integration with the JavaDoc tool.

# Coding Standards

## Sections of the Coding Standards

### Class Purpose Description

Every program file should begin with a brief overview of the class. This is provided in the form of a comment. Comments in JavaDoc follow the following format:

/\*\*

\*

\*

\*/

### File Header

The file header section provides information about the overall file. For integration with JavaDoc, each file header item description is preceded by a block tag of the form @tag. The following tags should be included:

@author *authorName* - Full Name of each author

@version *versionNumber -* Version Number of file

### Method Description

Every method declaration should be preceded by a brief description of the method in the same JavaDoc format as that used for Class Purpose Description above.

The following a required tags that follow the method description:  
[@param](http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html#@param) parameterName Description

The above tag is to be used for every parameter and should be followed by a short description of the parameter in the function.

@return Description

If a function has a return type, a description of the format of the return type should be specified.

### In-code Comments

JavaDoc comments for in-code comments follow the same structure as was described in the above sections. These comments are to be limited to describe lines of code that may otherwise be too complicated for other developers to understand. They should be used sparingly.

## Style

### Naming Conventions

* Constant variables should be in ALL\_CAPS
* camelCase should be used for all other identifiers (variables, functions) except constants
* Interfaces and Classes should start with an initial capital letter
* Names should be as descriptive as possible – self-documentation

### Layout Rules

* There should be a line break separating every program statement.
* Binary Operators should be separated by a blank space e.g. 5 + 5 = 10
* Lines of code should not extend the vertical red line in an IDE.
* Use indentation in blocks of code sparingly
* Block of code must have grouping symbols to indicate their start and end points

## Clarity

* Every class should start with an instance variables list followed by the methods list
* Public members should be listed first followed by protected members and lastly private members
* Nouns and noun phrases should be used for class names
* Verbs represent functions
* Avoid deep nesting of loops and conditionals