OES Java Coding Standards

1. **Java Naming Conventions**

Conventions for naming symbols in Java, including class names and method names.

* General Guidelines

1. The more public the name is, the more strictly the rules should be applied.
2. Spell out everything. Avoid abbreviations.
3. Don’t use unnecessary capital letters except for *final static variables*.
4. Avoid underscores.
5. Try to make the code as user friendly as possible. Eg. x=10; should be written as x = 10;
6. The code should be easily readable by other team members.

* Packages

1. Use all lowercase letters, no underscores.

* Classes and Interfaces

1. Use camel casing format. For example, the class named **myclass** should be written as **MyClass.**
2. Exception classes are to have an Exception suffix. For example, ArrayLengthNegativeException.

* Methods (eg. myMethod)

1. Always use lower case for first word, then capitalize each interval, no underscores.
2. Avoid giving methods with similar names but with different meanings.
3. Avoid giving methods different names for similar meanings.
4. Use the method names which are easily understandable.
5. Try to prefix boolean methods with *is or has* (eg. isAvailable)

* Instance Variables (eg. myObject)

1. Use all the access modifiers such as public, private, protected and default.
2. Avoid data type indications in names.
3. Declare static constant variables in capital letters.
4. All the variables should begin with lower case letters.
5. **Java Style**
6. Code indentation rules.

Configure your editor to tabs (by default) do not use spaces.

Set editor tab size to 4.

Open the bracket on the same line.

Body between brackets should be indented with one tab.

Closing bracket should be indented to match with the opening bracket line.

Eg. if(this) {

doThing();

doThing1();

}

1. Declare all the static variables at the beginning of the class.
2. Comment out the purpose of each variable.
3. Try to keep your code as compact as possible. Do not write unnecessary extra variables.
4. Delete the dead code (do not keep unnecessary code which is of no use.)
5. Avoid huge or hugely nested if/else blocks. Consider using private methods or exceptions.
6. **Java Documentation**
7. JavaDoc means converting a java code into the documentation using the comment block /\*\*

\*

\*/

b. **Class header :** These appear immediately prior to class declaration.

/\*\*

\* This is the first paragraph of the class.

\* <p>

\* This is the second paragraph of the class.

\*/

public class MyClass {

}

Describe the purpose of the class. The more public the class is the better the better the description should be.

Provide examples if possible.

Use of { @link class#method } while referring the methods of other class and methods is appreciated.

**Method headers :** This is the format for method headers for javaDoc

/\*\*

\* This describes my method.

\* <p>

\* @param myParam This describes my parameter.

\* @return myParam This describes the return type.

\* @param exception This describes an exception I may throw

\*/

public int myMethod(Object myParameter) {

}

**Variables :** This is the format to describe instance variables.

/\*\*

\* This is a constant

\*/

static public final int CONST = 5;

/\*\*

\* This is a protected variable.

\*/

protected int myVar = 0;

private int var = 0; // no Javadoc for this.