

**Code Inspection Document:**

**Glassfish 4.1.1**

Authors:

Bucci Giovanni

De Togni Riccardo

1. **Introduction**
2. **Inspected Classes**

In Java, the idea of a namespace is embodied in Java packages. All code belongs to a package, although that package need not be explicitly named.[[1]](#footnote-1) The only inspected class is Archivist, a class contained in the com.sun.enterprise.deployment.archivist package.

* 1. **Functional Role**

Archivist, as the javadoc says “[…] contains all common behaviour for Archivists. Archivists Classes are responsible for reading and writing correct J2EE Archives”. Archives in Java Enterprise Edition are container for modules or applications, which are composed of one or more module. When a Developer has to deploy a module on a web server for instance, it has to package it as a Web Archive (.war files). Archivist is an Abstract class since it is declared “abstract” and contains many abstract methods. The main known implementations are AppClientArchivist, ApplicationArchivist and WebArchivist.

The function of each analysed method will be presented in the specific section.

* 1. **Found Issues and improvable features**
     1. **Class Declaration**

Archivist already observe all the declaration rules. Packages are the first statements, then there are imports and after that start the class code, which is the only public class of the file.

* + 1. **Attributes**

Some issues in the attributes declaration have been found in this class. In order to bring all the attributes in a state that respects the standard they have been just re-ordered since the naming rules were already respected. The schedule followed is presented below:

* + - * Static Class Variables
      * Instance Variables

Both ordered by visibility:

* + - * Public
      * Protected
      * Package Level
      * Private

Here it is presented a before/after situation, it allows to understand at a glance all the modifications:

Attributes: Before and After



* 1. **Method Analysis**
     1. **Method: processAnnotations**

Annotation processing is an operation for scanning and processing annotations **at compile time**. This method processes the annotations contained in a Bundle Descriptor. The processing depends on the type of descriptor passed as parameter.

The method is found to be well-parenthesized (Kernigan and Ritchie style) and correctly indented. All other standard rules are observed except the line length of 120, which is violated in line 605. This issue has been corrected with the insertion of a line break and a “+” operator placed at the next line begin.

The comparisons are always made with “==” but it do not violate the rule 40 because are all comparison to null pointer and that is the only way to make it.

It is worth to highlight that since Archivist is abstract cannot be instanced and this method cannot have an output.

Personally, we would have created a local variable that contains the to-be-returned object (initialized to null) in order to eliminate all the return statement that breaks the execution before the end. In this way the “one return only” will be respected.

Now we will proceed with the checklist:

Naming Conventions:



Indentation:



Braces:



File Organisation:



Wrapping Lines:



Comments:



Java Source Files:



Package and Import Statement:



Class/Interface Declaration:



Initialization and Declarations:



Method Calls:



Arrays:



Object Comparison:



Output Format:



Computation, Comparisons and Assignments:



Exceptions:



Flow of Control:



Files:

4. * 1. **Method: readStandardDeploymentDescriptor**

This method is responsible to get a Standard Deployment Descriptor and instance it. In other words this method receive an Archive and returns an initialized descriptor instance. In case something goes wrong and it is impossible to get an Input Stream from the archive the method returns the Default Bundle Descriptor. That is to prevent application crash for instance during a server restart when there are no physical descriptors for a while.

Now we will proceed with the checklist:

1. https://en.wikipedia.org/wiki/Namespace#In\_programming\_languages [↑](#footnote-ref-1)