Code Styling – Team 6

1) Introduction –

The main purpose of the document is to avoid a difference in code design between the programmers.

2) Concept –

- All the names must be compatible to the naming conventions (below).
- As a rule, there will be no public variables, instead private variables, getters and setter should be used.
- The priority is not to use anonymous implementation of classes with content.
- Should avoid writing functions that are longer than 40 lines, such functions should be divided into a number of functions**.**
- All the comments must implements JavaDoc rules (where it possible).
- All the names in the project (interfaces, classes , methods, etc ..) must explain what the meaning of the type! A long and understandable name is preferable to a short and incomprehensible name.
- **Try to keep the code clean and understandable**!

3) Naming Convention

#	Type	Name convention	Example
1	Interface	IInterfaceName	public interface IFile { … }
2	Class	ClassName	public class File { … }
3	Methods	methodName	public void saveFile() { … }
4	method parameter	parameterName	public void setFileName(String filename) { … }
5	getter	getFieldName	public void getFile() { … }
6	setter	setFieldName	public void setFile() { … }
7	Field - private	m_fieldName	private IFile m_file;
8	Field - protected	m_FieldName	protected IFile m_File;
9	Field – private static	s_fieldName	private static IFile s_file;
10	Field – protected static	s_FieldName	protected static IFile s_File;
11	Field – public static final	FIELD_NAME	public static final IFile FILE;

4) Documentation

- Interfaces content must have documentation.
- Public, protected and abstract methods must have documentation.
- Private functions do not require documentation, subject to the developer's discretion**.**
- Comments should be added wherever the content is unclear, or the action is complex.
- **A comment must clearly describe the reprehensible, be short and clear!**