**Coding Conventions – Group 7**

**Introduction:**

The **‘**coding conventions’ document is an important tool for the software development process. It defines a standard set of guidelines that needs to be followed by programmers which consequently helps in improving the readability of the software and the ability to understand the code in an easier, faster and thorough manner. The following coding conventions have been adopted by the team members in building the RISC Game Project.

**File Organization:**

**// left empty intentionally. Things to be done soon.**

**Naming Convention:**

Naming conventions help in providing meaningful names to the identifiers. The name of an identifier can provide information about the functionality of that identifier.

1. **Classes:** Class names are written in mixed case format with the first letter of each internal word in uppercase.

E.g. class Player, class GamePlay

1. **Interfaces:** Follows same naming convention as Classes.

E.g. interface ArmyAssociation

1. **Methods:** Methods are written in mixed case format with the first letter in lowercase and first letter of each internal word in uppercase.

E.g. getName(), printAll()

1. **Variables:** Variable names are generally short and meaningful; and are designed by the programmer according to the need of program. One-character variable like ‘i’, ‘j’, ‘k’, ‘m’ should be used only for temporary purposes.

E.g. int temp, double counter, int numPlayers, etc.

1. **Constants:** Constant variables are written in all Capitalized letter with words separated by underscore (“\_”).

E.g. int CONTINENT\_VALUE

**Indentation:**

We have used four spaces (1 tab) as the unit of indentation.

1. **Wrapping Lines:**

Since it is very difficult to read and handle long characters in a single line on a screen, we have used less than 80 characters per line. Expressions which have greater than 80 characters in a line have been broken down by using following technique:

Split Long Line into Several Shorter Lines and either:

* + 1. Break to new line after a comma.

E.g. getPlayer (longArgument1, LongArgument2, LongArgument3,

longArgument4, longargument5);

* + 1. Break to new line before an operator

E.g.

longName1 = longName2 \* (longName3 + longName4 + longName5)

+ 4 \* longname6;

Alignment of beginning of the new line of the expression is at the same level of the previous line.

**Declarations:**

1. Declare only one statement per line.

E.g.

int count; //some comment

int value; // some comment

1. Local variables are initialized at the point of declaration, with few exceptions.

**Comments:**

There are two types of comments used for improving the code understandability:

1. **Implementation comments:** Used to give comments/descriptions on implementation of code. It is generally inside of a block. The formats of implementation comments are **/\*.......\*/** and **//.**
2. **Document comments:**  Used to describe the specifications of the code without providing actual implementation. These comments are generally delimited by **/\*\*. . .\*/ .** Javadoc will be used for document comments.

Add comments at the beginning of each:

1. **File:** The comment helps in explaining the purpose of the file in the project.
2. **Class:** It is used to describing the purpose of the class and must be written just before its declaration.
3. **Method or function:** A comment before a method helps in explaining what that method does and how it works, as well as what is the purpose of its parameters and what to expect as return value.
4. **Variable declarations:** A comment on the side of a variable describes its purpose. For a class data member, a comment describes the role of the data member in that class.

**References:**

* **Robert L. Glass: Facts and Fallacies of Software Engineering; Addison Wesley, 2003. ISBN-13: 978-0321117427.**
* **Oracle Corporation. Code Conventions for the Java Programming Language. http://www.oracle.com/technetwork/java/codeconvtoc-136057.html**
* **Google Inc. Google Java Style: https://google.github.io/styleguide/javaguide.html**
* **Joey Paquet, Course notes for COMP6441: Advanced Programming Practices. Concordia University, Fall 2019.**