**Coding Conventions**

**Introduction:**

Coding conventions are important tool for the software development process. It helps in improving the readability of the software and the ability to understand the code in an easier, faster and thorough manner to a developer. 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 helps in providing meaningful name to the identifiers. The name of identifiers 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.

Ex: class Player, class GamePlay, etc

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

Ex: 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 lowercase.

Ex: getName(), printAll(), etc

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

Ex: int temp, Double counter, etc.

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

Ex: int CONTINENT\_VALUE;

**Indentation:**

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

1. **Wrapping Lines:**

Since it’s very difficult to handle long character in a single line in a screen, we have used less than 80 characters per line. The expressions which has greater than 80 character in a line, have been broken down by using following techniches.

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

Ex: getPlayer(longArgument1, LongArgument2, LongArgument3,

longArgument4,longargument5);

* + 1. Break to new line before an operator

Ex:

longName1 = longName2 \* (longName3 + longName4 longName5)

+ 4 \* longname6;

* + 1. Alignment of beginning of the new line of the expression is at the same level of the previous line.

**Comments:**

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

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

Some of the places where we have mentioned comment.

1. **Beginning of each file:** The comment helps in explaining the purpose of the file in the project.
2. **Class declaration:** It is used to describing the purpose of the class and written just befor its declaration.
3. **Method or function:** The comments before methods helps in explaining what it does and how it works, as well as what is the purpose of its parameters.
4. **Variable declarations:** The comment on the side of variable describe its purpose and its on the class data member then it describes the role of data member in the class.

**Declarations:**

1. We have declared only one statement per line.

Ex,

int count; //some comment

int value; // some comment

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

**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.**