**System Re-Engineering CO4206 Assignment I Part 1(Inspecting Code)**

*Inspecting the “tools”(INPUT) folder in FreeCol turn-based strategy game*

By Indayan Bera (ib137@student.le.ac.uk)

The “tools” folder consists of java files that implements tools beyond the core game logic that either generates or modifies core game data based on certain constraints established during the overall design process. Classes in the folder have been indented property and adheres to standard coding practices. Variables and sub-classes are initialized with proper access specifiers so that data remains encapsulated and are used only in the manner intended by specific functions. However, there are a very few comment lines in the code lines so as to promote better understanding. Even though separate java classes promote better understanding of the code from a programmers point of view, certain tools could have been merged and written in a concise manner so as to reduce overall LOC. Roles of each java files have been discussed briefly in the following points with special emphasis on certain java files more than others:-

1. **ColonizationMapReader**

This class is used to render the Colonization Map in the game. The class either loads an existing map by output in the console when main() is executed with the filename as arg[0] ,or reads from an pre-existing file in case the param --palette is used as arg[0] followed by filename.

1. **ColonizationSaveGameReader**

The class reads an existing file containing information of a saved game. The main class consists of several static sub-classes namely GameData, PlayerData, ColonyData, Colonist and several methods namely main(), run(), getString().

GameData subclass accepts and displays information related to the map such as map size, level difficulty and number of colonies.

PlayerData subclass stores information such as player name, new land name and whether the player is human or AI

Colonist subclass stores attributes such as occupation, specialty and tike of each colonist and displays them in the console.

ColonyData subclass stores map information and an array of Colonies.

1. **DesktopEntry.java**

The class creates a desktop entry file which stores information regarding the desktop icon of the game and information such as icons and directory of the executable.

1. **Flagtest.java**

The class creates and stores flag information of different colonies in the game.

1. **Forestmaker.java**

Generates in-game forest tiles based on logical locations of components of the map.

1. **FSGConverter.java**

This class converts a given input file to an uncompressed indented XML file to be stored and fetched later for load.

1. **GenerateDocumentation.java**

Generates language-specific documentation .

1. **InstallerTranslations.java**

This class handles translation for the installer translating back and forth between specified language to be displayed to the user. This ensures that the core installation process remains independent of any language and ensures that any user is able to interact with the installer.

1. **MapConverter.java**

Converts from one map format to another based on provided specifications.

1. **MergeTranslations.java**

This class allows translation based updates to be merged to the main-game code. For example, after installation of a new languages pack, this class allows for changes to be reflected.

1. **RiverMaker.java**

Similar to forest maker, this class generates in-game forest tiles based on the logical components of the game so as to avoid overlap and rendering based collisions.

1. **SaveGameValidator**

Generates and validates a save game, which can later be used by ColonizationSaveGameReader to be loaded.

1. **TranslationReport.java**

Documents the state of translation.