Build #3 – Refactoring Document

Potential Refactoring Targets:

The targets for refactoring were identified based on several key issues and challenges within the codebase that needed improvement. Here's an explanation of how these targets were found: -

- Difficulties introduced while team members were reading the code for different components.
- Redundant code blocks due to divided work structure of the team.
- In summary the goal of the refactoring process was to address these issues and improve the overall quality, maintainability, and understandability of the code.
- A list of potential refactoring targets: -
  - **Logging**
    - Though the log was getting generated. Specific separators were introduced and particular format was decided. Which included file numbers and files. Helped the team mates in logging the errors and exact line the errors were generated.
  - **Test Cases**
    - Some of build #2 test at certain times failed due to their dependency on the function which assign countries to players. Due to which, when the test cases were run for 10-20 times. We were seeing that the test cases are failing at least 2-3 times.
    - This was not something which was happening due to sloppy writing of code, rather it was something which was produced due to randomness of country assignment to the Players. Which was a feature of the game.
    - This behaviour was noted as bug, and unit tests were updated with much better approach in which failing of the unit test cases reduced to 0. Even though they are run multiple number of times.
  - **Commenting**
    - Meanwhile, the operation such as handling the attack () order in the Game were complex. Specially how probabilities are calculated. Extra comments were added so that every one in the group can be on the same page with regard to handling of an Attack order.
  - **Modularity**
    - Few of the code units were long. Due to variety of checks happening in the if-else block it becomes confusing for the members of the team to understand every aspect of that unit.
    - This problem was solved by introducing a Boolean variable with a name for the condition, which is assigned the condition provided in the if-else cases and then variable is used to verify the conditions provided in the unit.
  - **Removing Redundant Code:**
    - Due to coloring being used in lot of parts of our project. There were printing of colored text. This functionality was made available by a function named renderColorString (String color, String text).
    - This function was introduced in the code, on the need basis by team members. Such that every class has one – two copies of the function in different names.
    - This was reduced to single class and function was made static, so that every programmer can access single function.
  - **Handling Exceptions:**
    - Earlier each function had a separate try-catch block in which the exceptions were generated and caught.
    - Which created a lot of problems for handling the state of the game in case an exception occurs.
    - To handle this issue, every function was added with throws-based syntax as to reduce the exceptions handler in the code.
    - Meanwhile in our Phase class, from where all the functions calls are going was put inside a try block.
  - **Output Messages:**
    - Every team member was making their own outputs for when particular exception occurred. Which sometimes had tangling “\n” at the end caused problems while taking input.
    - This problem sometimes resulted in the wrong execution of the Game.
    - As we introduced throw-based syntax, we also come with an idea to throw a exception with message as parameter to the Exception.
    - And “\n” was replaced with System.lineSeparator which is much better approach to show next line symbol.

- **Improving the Mutators:**
  - Due to following of the name convention, each variable had the name d_<name>. By which everyone was generating the Mutators as setD_<variable> name, which created confusion around the team if D signifies anything.
  - We changed our mutators by deleting that extra D_ from every mutator present in the code base.
- **Constants Handling:**
  - Multiple values that are recognized as constants in the code base made available through a separate class called AllTheConstants.java.
- **Naming Convention:**
  - Edits were done to keep the code base within right naming convention as proposed in the build document.
- **Redundant Code Block Removal:**
  - Some code blocks due to introduction of the abstract classes were redundant.
  - These code blocks were removed as they were not used.
- **Javadoc:**
  - Usual spelling mistakes as well as some redundant Javadoc components was removed from the code.

Actual Refactoring Targets:

List of 5 Actual Refactoring targets: -

- **Adapter Pattern**
  - **Tests Added: -**
    - testConquestLoadMap()
    - testConquestLoadMapSecond()
    - testConquestMapValidity()
    - testConquestMapInvalidity()
    - testConquestMapInvalidity()
    - testPerformSaveMapAsConquest()
    - testPerformConquestShowMap()
  - **Tests Modified: -**
    - None
  - **Reason: -**
    - Introduction of the order pattern to introduce reading multiple variety of map during the execution of the game. This pattern implementation would be helpful while reading “conquest” game map.
    - Meanwhile which type of map reader to be used is decided internally.
  - **Before: -**
    - Earlier loadmap read the map details from only one type of Map structure. And all the operations were dependent on the country’s details read.
  - **After: -**
    - Loadmap can automatically read the map based on the inputs provided by the user. Meanwhile further type of map readers can easily be added due to introduction of Adapter Pattern.

- **Single Game Mode and Spectator Mode**
  - **Test Added: -** None
  - **Reason: -**
    - This kind of gameplaying was needed because of the fact the game we developed is a command line-based game.
    - Which makes it difficult to play as every time typing command is difficult task. By the above introduced mode a player can just see how the game works by automating the play.
  - **Before: -**
    - Commands has to be input to play game. For every player-based instruction command has to be typed.
  - **After: -**
    - Commands exists for making the play automated where in players are chosen randomly and user need to provide the strategies, he/she wants to see game play based on.

- **Tournament Mode**
  - **Test Added: -** None
  - **Reason: -**
    - As per the build requirement this function was necessary.
    - Tournament mode basically is a feature by which player can configure the game before and game automatically continues. At end Winner would be decided.
  - **Before: -**
    - Before this there were no Tournament mode.
  - **After: -**
    - A configuration of the tournament has to be provided on basis of a command, tournament -M listofmapfiles -P listofplayerstrategies -G numberofgames -D maxnumberofturns and game continues until the winner is decided and shown.
- **Strategy Pattern**
  - **Test Added: -**
    - None
  - **Reason: -**
    - As per the build requirement this feature was necessary. Basically, strategy design pattern was helpful in providing Player characteristics as how a user wants a defined player to play, in automatic game running scenario.
  - **Before: -**
    - As there was no automatic play before this build, such as tournament mode based automatic play and single player mode. So, no strategy-based play was implemented.
  - **After: -**
    - Now we have strategies option e human, aggressive, benevolent, random, and cheater. Meanwhile this is implemented using Strategy pattern, so we can easily add more strategy patterns in the future.
- **Game Save/Load**
  - **Test Added: -**
    - None
  - **Reason: -**
    - As per the build requirements this feature was is required to save the map and load the map in persistent storage so that a player can easily save the game state. And retrieve the game state, as per their convenience.
  - **Before: -**
    - We had functionality for the saveMap and loadmap but not related to game. It was associated with the Map editing rather than game.
  - **After: -**
    - The feature described is implemented and user can save game using savegame <filename> and will be loading game loadmap <filename>.