Build #2 – Refactoring Document

To identify the potential refactoring targets in the previous Build(#1), following parameters were laid emphasis on:

- Methods containing more than one logic.
- Similar logic being called at multiple places.
- Classes with many methods.
- Longer nesting and wraps of conditional logic.

Refactoring Targets:

1. **Previous methods modification to suit game logic**
2. **State Pattern**
3. **GameState**
4. **Logging of game progress**
5. **Endgame logic**
6. **Order class converted to interface to implement card functionality.**

Potential Refactoring Targets:

7. Formatting Functions for Phases and Orders
8. Unhandled exceptions or Error Handling
9. Test case coverage improved
10. Complexity Reduction
11. Code Commenting
12. Large Classes
13. Nested Conditionals
14. Global variables
15. Dependency management

State Pattern:

Before: Game Engine handled the complete game play

After: Split into phases to handle Start Up, Issue Order and Order Execution phases

Reason: Better code understandability and documentation

Added Test Cases: None

Modified Test Cases (if any): None

GameState

Before: GameState keeps track of only map and player

After: GameState keeps track of map, players and logs related to players

Reason: To provide logging, as GameState is used throughout the game.

Added Test Cases: None

Modified Test Cases (if any): None

Logging of Game Progress.

Before: Used print statement to notify user of a wrong command.

After: Used logging to notify user of a wrong command.

Reason: To keep track of game play, debugging and monitoring.

Added Test Cases: None

Modified Test Cases (if any): None

End Game Logic:

Before: Use exit command to quit the game.

After: The game automatically ends when a single player conquers all the countries on the map.

Reason: To have a definitive end of the game at some point

Added Test Cases:

1. testEndOfTheGame() – checks if all countries are conquered by one player.

Modified Test Cases (if any): None

Previous methods modification to suit game logic:

Before: Game Engine handled every operation of the gameplay

After: Modified the code to incorporate phases and to initialize them

Reason: To meet coding standards

Added Test Cases: None

Modified Test Cases (if any): None

Order Class converted to type Interface:

Before: Class executed and deployed orders and updated the game state

After: Interface that handles deploying and advancing.

Reason: To handle better abstraction between deploying and attacking operations

Added Test Cases: None

Modified Test Cases (if any): Removed order test class as it is now an interface