**Refactoring for Build 3**

**Potential List for Refactoring Build 3 –**

1. **Magic** **String** replacement with add, remove command.

Example: -add, -remove.

1. Use of **Dependency** **Injection**: The Commands class is directly instantiated in the main method. This could be replaced with dependency injection to make the code more testable and flexible.
2. **User** **Interface** for Validation Strategy: Use of an MapValidationStratergy interface to define the validation logic.
3. Updated **execute** method: The executed method was used to perform many operations within the single method. We have split the execute method into smaller code and in different methods, splitting the code for **Airlift**, **Bomb** and **Blockade.**
4. Improved **LogWriter** class: The instances variables d\_logFile, d\_dtf, and d\_now are currently package private. It can be made private to adhere to the principle of execution. The FileWriter can be closed in finally block or using try-with-resources, which could lead to resource leaks. The IOException is being caught and printed, but not rethrown or wrapped in a runtime exception.
5. Use of **TextTable** library for displaying the output map in the console.
6. **Adapter** Pattern – Adapter pattern is used to enable the application to read or write from or to a file using the “conquest” game map format. The application decides whether to use the “domination” file reader or the “conquest” file reader, when a file is opened, depending on the file type.

When the map file is saved, the user is given the option to decide the file format to use as output.

1. **Strategy** Pattern – The player issueOrder () method’s behavior is altered by using the Strategy pattern. The strategy provides different behaviors that support the Player class to execute varying behavior when executing the issueOrders () method.
2. Resource Management with **try**-**with**-**resources**: The Scanner object is currently closed using a finally block. You can simplify this by using a try-with-resources statement. This ensures proper resource management and eliminates the need for an explicit close call.
3. **Removal** of **Modifiers** from Interface properties – Remove the public modifiers from method declarations and public static final modifiers from field declarations in interfaces.

Removes the clutter and unifies the attributes of Java interfaces.

1. Use of **StringBuilder** for Efficiency - Use of StringBuilder can be done instead of using l\_dataString while reading the lines. String Concatenation in Java is inefficient as it creates new strings every time.
2. Remove **Explicit** **Type** **Argument** in Continent.java, Country.java, GameEngine.java, GameMap.java, MapValidation.java and MapValidationTest.java
3. Replace assignment with **Compound** **Operators** in the Commands.java.
4. Split of **Multiple** **Variable** **declaration** in the ReadMap.java and WriteMap.java.
5. Using **@Override** notations – This rule adds the @Override annotation to methods overriding or implementing another method declared in a parent class. It improves readability and ensures method signature is a sub-signature of the overridden method.

**Main Refactoring Operations -**

1. **Adapter** Pattern –

Adapter pattern is used to enable the application to read or write from or to a file using the “conquest” game map format. The application decides whether to use the “domination” file reader or the “conquest” file reader, when a file is opened, depending on the file type.

When the map file is saved, the user is given the option to decide the file format to use as output.

1. **Strategy** Pattern –

The player issueOrder () method’s behavior is altered by using the Strategy pattern. The strategy provide different behaviors that support the Player class to execute varying behavior when executing the issueOrders () method.

Player Types are –

1. A human player requires user interaction to make decisions.
2. An aggressive player strategy i.e. deployment on its strongest country, attacking with its strongest country, moving its armies in order to maximize aggregation of forces in one country.
3. A benevolent player strategy i.e. deployment on its weakest country, never attacks, then moving its armies in order to reinforce its weaker country.
4. A random player strategy i.e. deployment on a random country, attacking random neighboring countries, moving armies randomly between its countries.
5. A cheater player strategy whose issueOrder() method conquers all the immediately neighboring enemy countries, and then doubles the number of armies on its countries that have enemy neighbors.
6. Updated **execute** method: The executed method was used to perform many operations within the single method. We have split the execute method into smaller code and in different methods, splitting the code for **Airlift**, **Bomb** and **Blockade.**

Screenshot –

Before –

A computer screen shot of a program code

Description automatically generated

After –

A computer screen with white text

Description automatically generated

A computer screen shot of text

Description automatically generated

1. Improved **LogWriter** class: The instances variables d\_logFile, d\_dtf, and d\_now are currently package private. It can be made private to adhere to the principle of execution. The FileWriter can be closed in finally block or using try-with-resources, which could lead to resource leaks. The IOException is being caught and printed, but not rethrown or wrapped in a runtime exception.

Screenshot –

Before -



Screenshot – After



1. Removal of **Modifiers** from Interface properties –

Remove the public modifiers from method declarations and public static final modifiers from field declarations in interfaces.

Removes the clutter and unifies the attributes of Java interfaces.

Screenshot –

Before –

A screenshot of a computer program

Description automatically generated

After –

A screenshot of a computer program

Description automatically generated