Refactoring Document

Optimizing Design Patterns:

1. **Strategy Pattern for Player Behaviors:** Refactor the Player's `issueOrder()` method to use the Strategy pattern, allowing different behaviors during the main development phase.
2. **Adapter Pattern for Map File Formats:** Refactor the code to use the Adapter pattern to enable the application to read/write map files in both the "domination" and "conquest" formats. Implement functionality to decide the file reader to use based on the file type and allow the user to choose the file format for saving.
3. **Single Game Mode Implementation:** Create a game mode starting with user-selected map and player settings, proceeding until one player conquers the whole map.
4. **Tournament Mode Implementation:** Add a Tournament Mode that proceeds without user interaction, displaying results at the end.

Code Efficiency Enhancements:

5. **Removal of Redundant Commands:** Eliminating unnecessary command line calls to enhance code efficiency.
6. **Reducing Object Overhead:** Refactoring temp order creation to minimize extra object instantiation.

Enhancing Readability:

7. **Smarter Conditional Checks:** Breaking down concatenated checks for better code readability.
8. **Method Refactoring:** Breaking down large methods into smaller functions for improved readability.
9. **Centralized Validation:** Discarding validation in individual classes, considering centralization.

Cleaner User Interaction:

10. **Streamlined Game Play:** Removing redundant code and implemented choosing options to select game mode.
11. **Reduced Redundancy in ShowMap:** Refactoring showmap functionality to eliminate redundancy.

Complexity Reduction:

12. **Addressing Nested Code:** Considering and addressing deeply nested code for improved clarity.

13. **Naming Consistency:** Renaming methods for a more consistent and understandable codebase.

New Refactoring Targets:

1. **Strategy Pattern for Player Behaviors:** Refactor the Player's `issueOrder()` method to use the Strategy pattern, allowing different behaviors during the main development phase.
2. **Adapter Pattern for Map File Formats:** Refactor the code to use the Adapter pattern to enable the application to read/write map files in both the "domination" and "conquest" formats. Implement functionality to decide the file reader to use based on the file type and allow the user to choose the file format for saving.
3. **Single Game Mode Implementation:** Create a game mode starting with user-selected map and player settings, proceeding until one player conquers the whole map.
4. **Tournament Mode Implementation:** Add a Tournament Mode that proceeds without user interaction, displaying results at the end.
5. **Streamlined Game Play:** Removing redundant code and implemented choosing options to select game mode.