List of potential refactoring targets:

1. Implementing state pattern for phases
2. Implementing command pattern for orders
3. Merging all command validation classes for different phases into one class
4. Dividing MapController class into small chunks
5. Handling of console based on phase
6. Dividing MapController further into two classes; MapController and MapEditor.
7. Adding new class to store a players’ information
8. Adding new class to store information related to command information
9. Using one console for map editing and game play phase.
10. Removing ExecuteCommandView and calling respective functions from ValidateCommandView class
11. Creating a class in controller for display console for adding command during issue order phase
12. Using ValidateCommandView for executing orders
13. Adding instance of view in model classes
14. Adding instance of model in view classes
15. Adding instance of controller in both model classes

List of 5 refactoring targets:

1. Implementing state pattern for phases

* It was selected because it was a requirement of build 2.
* Implemented 5 test cases for checking if the phases were initialized properly after each command. (1) Test case to check when Game engine object is created default phase is initialized. (2) 2 Test cases to check when editmap is called map editing phase is initialized and when savemap is called default phase initialized. (3) 2 Test cases to check when loadmap is called start-up phase is initialized and when assigncountries is called issuer order phase initialized.
* Earlier, all commands were validated and checked if they were applicable for current phase (which was stored in an integer variable) and then they were executed. Now, every command is validated and then resp. method is called, then based on the phase the method gets executed or it displays error message.

1. Implementing command pattern for phases

* It was selected because it was a requirement of build 2.
* Implemented 10 test cases for checking if the order command pattern were working properly. (1) 2 test cases to check deploy order issuing and execution process. (2) 3 test cases to check advance order issuing and execution process. (3) 3 test cases to check airlift order issuing and execution process (4) 2 test cases to check bomb order issuing and execution.
* Earlier, the order was created by the player after command validation. Now, player calls dedicated class for command validation and based on validity of the command a respective order object is created and populated in players list of orders. At execution, functionality related to a specific order is handled by sub-classes of order.

1. Merging all command validation classes for different phases into one class
   * It was selected because for every command entered we had to validate the commands and calling resp. functions/methods in the Phase class.
   * Implemented 3 test cases for checking if commands are validated properly. (1) 1 test case for checking if any random incorrect command throws an error. (2) 2 test for validating correct commands and setting correct phases after execution.
   * Earlier, we had three set of classes for validating MapEditing, GamePlay start-up and GamePlay commands. Now, we have one common validation class for all command types and after validation respective methods are called from Phase class to carry out the command. Instances of Phase handles the validity of the command for the phase or not and displays appropriate message.
2. Dividing MapController class into small chunks
   * It was selected because the MapController class had become too lengthy and cumbersome to understand.
   * Implemented 3 test cases for checking the validation of map. 1 test case each for checking if map is connected graph, each continent is a connected sub-graph and check if continent is empty.
   * Earlier, we were handling all functions related to map in MapController class. Now, we have a separate class MapValidator to handle all functionality related to handling validation of the map.
3. Handling of console based on phase
   * It was selected because we had separate console for MapEditing and GamePlay phase.
   * Implemented 2 test cases for checking if the consoles are properly switching. The test cases check if the load map command initialized start-up phase, which is only initialized when console is switched.
   * Earlier, we were handling switching of console based on an integer value. Now, we have instances of phases for different phase. We are checking the type of phase instance to handle the switching.