Refactoring operations - Build 2

Potential targets

1. Replace phase string values by Enum.
2. Encapsulate the map building operations.
3. Moved current player and current phase
4. Extract class behavior of phase execution as implementation methods

1. Type safety

Phase string literals - StartupPhase, ReinforcePhase, AttackPhase and FortifyPhase.

Replaced the phase string literals by Enum. This made the code to throw the compilation errors in case of type mistake. Also, the case sensitivity is now controlled by the Phase class.

Impact on other methods - updateCurrentPhase
Removed the redefinition of phase collection by using enum *values()*.

2. Encapsulation

Refactoring technique - Extract as method

Map building and its operation to start the new game involved the following operations:

1. Instantiate map object
2. Load map using filename
3. Validation of map
4. Initialise the players

Earlier, the *GamePlay* knew too much about the behavior of the *Map* class. Also, the method to allocate initial set of armies was defined inside the game. So, it made more sense to push the steps down to the Game class.

We turned the multiple operations involved in GamePlay initialisation into a method whose name explains the purpose.

3. Moved current player and current phase

Refactoring technique - Move a field

Moved the field variables - *currentPlayer* and *currentPhase* to the *Game* class from GamePlay. As per the architecture thought process, GamePlay should only be responsible to manage the instance of the Game and carry out the executions.
Also, it was found that these fields are more used by the inner classes which made more sense to move the field to *Game* class.

4. Extract class behavior of phase execution as implementation methods
Refactoring technique - Inline class

Moved the phase execution class to the player class. As hinted for the build 3, in order to implement the strategy pattern we had to move the class as method implementation of the *Player* class. All the phases will be invoked through the player class.

In further builds, in order to change the execution of the *CurrentPlayer* on the basis of their type, the execution method will be polymorphically taken care.