Refactoring

For the current build, 2 new functionalities were to be added, which are Attack phase and RISK card logic. Also, the design pattern of the project had to be implemented using the Observer Pattern which we had initially implemented using MVC architecture.

After a thorough discussion and analysis of the changes that were expected in Build 2, following targets were identified and changes were put into place :

- ★ **The Observers:**
  - Each view class implements java.util.Observer class and all of them has an Overridden Update(Observable,Object) method which is used to update the view. Views acting as Observer are:
  - AddContinentPanel, CreateContinentPanel, EditContienetValuesPanel, RemoveContinentPanel are observing GameMap model for retrieving the continent list to display.
  - AddCountry, AddNewCountryPanel, EditCountryValuesPanel, RemoveCountryPanel are observing GameMap model for retrieving the country list to display.
  - TradePanel is observing Player class to get all the information regarding Risk cards.
  - WorldDominationView is observing Player model to get information of Player domination over the map, its total armies and continent names.
  - GamePlay view observer the GameState model to get the current state of the phase and make changes into the game view accordingly.
- ★ **The Observables:**
  - ○ The Models that are made observable are:
    Let’s call them : MethodsObs - setChanged() and notifyObservers()
  - GameMap:
    Added the MethodsObs at the end of the methods that change the state of the stored data.(All such methods except “setGetGuiHashMapParameter()”).
  - GameState:
    Added the MethodsObs in methods:
    - ○ “notifyGameStateChange, attack, allOutAttack”

    - “Fortification” calls the notifyGameStateChange method that inturn calls the MethodObs

- Player: Added the MethodsObs in methods:
    - setRemainingArmies, updateRemainingArmies, setPlayerArmies, setCountries, removeCountry, setContinents, setCardsHeld, addRiskCard, addPlayerArmy

- RiskCard: The Observers are notified with change of each property of RiskCard.

Refactoring Activities

- Relocated the Hashmap datasets in a seperate model called GameMap
- GameMap contains methods like:
    - addCountry , addCountinent , removeCountry, removeContinent, getContinentHashmap,getCountryHashmap, getGuiHashmap

- Methods from MapGenerator that dealt with the above hashmaps are transferred to the model GameMap.

- The Class MapEditor was removed and everything related to map is now handled in map generator(a controller). Methods in the mapgenerator refactored to use the accessor methods of GameMap.

- Also all methods in MapValidator that used hashmaps refactored to use GameMap methods.

- GameState model added to hold the whole Game State throughout a single game play