SYSC3110 Design Project

Group Members: Nikolas Paterson, Adi El-Sammak, Erik Iuhas

RISK: Global Domination

Design Process:

When designing RISK: Global Domination, we took multiple variables into account and broke it into a few main class groups. The first group is classes containing game variables and references to other objects; these are Game, Player, Territory. The next group of classes are the ones who allow the user to interact with the objects; these classes are Command, CommandParser, CommandWord and Command. Finally, the last group of classes are responsible for handling the game logic behind a Player's attack and generating dice rolls; these classes include Dice and GameEvent.

Data Structure HashMap:

Risk is a board game that includes 42 Territories, each with a differing number of neighbors. We determined that a HashMap would be the best data structure for storing territory information using the territory name (String) as the key and the Territory object as the value. We believe that this was the most effective way to obtain the Territory object when receiving the user's commands. HashMaps are utilized in multiple classes; Player uses it to store its territories and allows it to find neighbors to attack. The Game class also stores the "WorldMap" which is the entire map, including all territories. The GameSetup class creates the "WorldMap" using a CSV file that contains the Territory neighbours and Continent. Lastly, the Territory class contains a HashMap which stores their neighbouring territories.

GameEvent & Dice Class:

The GameEvent class acts as the controller of the Risk game since it is dependent on user input. Thus meaning, that once a Player in Risk decides to "REINFORCE", "ATTACK" or "FORTIFY" the GameEvent class is responsible for handling the outcomes of an event by updating the Player or Territory class based on the action called by the Player. The GameEvent's attack() method uses the Dice class to set up the corresponding rolls for the attacker and defender. The Dice class's primary responsibility is to generate rolls for the attacker and defender and assist the GameEvent class by handling an attack's outcome. The Dice class generates the attacker's rolls based on the number of dice that the attacking Player wants to roll with. Additionally, the Dice class generates the defender's rolls where the number of the dice the defender rolls with is decided based on how many troops the defending territory has.