Design Choices :Author: Yasin Jaamac and Yusuf Jaamac

***Dice:*** This class is used for the game to generate random numbers through the roll() method. Players who role the highest go first to start the game.

***Board***: This class is used to create players via an array List , countries via an array List and continents. The appropriate countries were created for the specific continents(Australia, Asia,Africa,Europe,South America and North America). This class contained a method to set the initial amount of armies for the all the cases of players entered by users. Also setting adjacencies for all countries is done here via set Adjacent Countries method in country class. Finally attack method which was done by mapping the country to the index and then as per rule checking if the passed countries are adjacent via checkAdjacentCountries method and then it went on to follow the rules for a player to attack. Also there is a added method ownContinent that uses an arraylist to set which coutnry owns what. Method continentBonus takes in a continent and adds bonus armies to the specified country. we have methods aiaddBonusArmies and setAi to set the ai to the game via setPlayer asAi method via for each loop. ai attackMove and aiFortifyMove were added to attack countries and fortify. For the bonus armies there are 2 methods in board used. ownContinent() returns an arraylist and contains a list of any continents the player owns. The continentBonus method is used to return then number of bonus armies for each continent. In board there are 2 methods used for the troupe movement. dfsSearch() is used to find all the possible paths and runDFS() runs the dfs and stores all the values (if a path was valid or not). Depth first search was used instead of breadth first search because it was easier to implement.

***Country***: This class contains the ruler with type object, with the getRuler(),setRuler() and hasRuler() method. An arraylist is used to set and get adjacent countries as a list would be the most efficient method. Increase army count and decrease army count method were put in this class and the logic for it was having a parameter numArmy for what player wants to increase army by and add that to the current number of armies, decrease Army count was same but subtraction.

***Player:*** We implemented an array list for the ruling Countries and ruling Continents. The rest were getter methods

***Continent:*** This class had methods like add country get county and dealt with the bonus army so we had a getter method for it.

***Controller:*** In the controller class, it acts as an interface between the model and the view. All the logic and data is separated and it intercepts all requests that come in. We had the method actionPerformed use a switch case and if a start, selection or something with mainscreen was selected it would call the specific method to deal with it. For example when startPerformed is called it goes to that method and checks if start or rules was inputed by user, the appropriate input would then be invoked by view and the appropriate method. For example if user enters start it would proceed to have view invoke method createNumOfPlayers() and if rules was inputed view would invoke method showRules(). Same goes for the other methods startperformed(),jmenubarperformed(),selectLevelPerformed() and method mainScreenPerformed().

***View:*** In the view class, it represents the look of the text for our model, what the user sees. It has methods such as startmenu,which creates the startmenu panel and it is where the user is first sees when they start the game.Also method createNumOfPlayers was designed to create number of players panel by using swing components(Jbutton,JPanel etc) so user can select number of players playing. Also have other methods such as main screen, quit, pass etc. Furthermore it contains quit method that interacts with user if user wants to quit, JOptionpane was used to do this. Method showRules() is an option for user to click so it shows all the rules, it prints all the rules once its pressed in the menu,same goes for method help().

***SaveAndLoad:*** We implemented methods save board and load board and we used file input/output stream for implementation. Same goes for saveplayerNum and Load PlayerNum methods, writeobject and close were used to work that method.