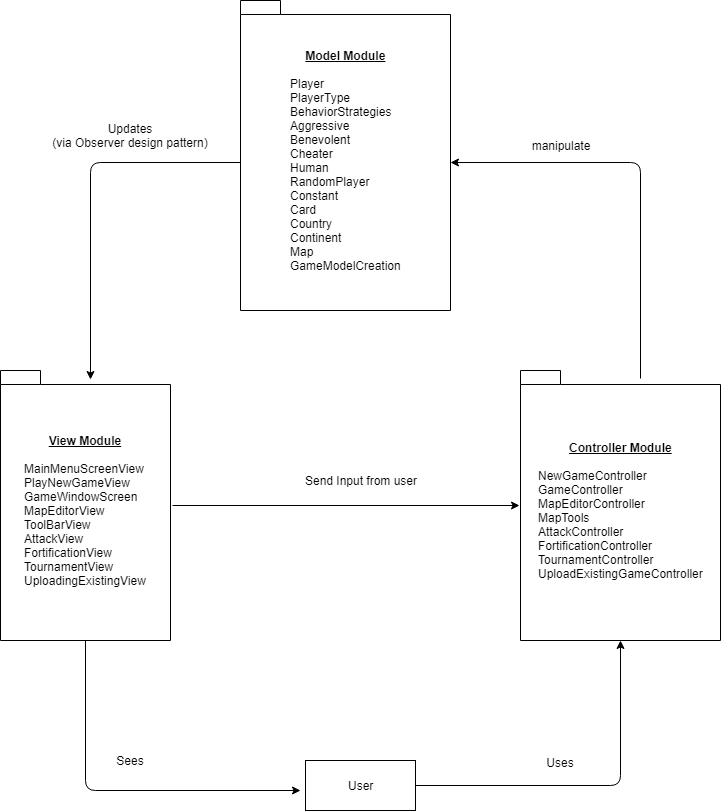
**Project Architecture – The Risk Game**



***Fig No – 1***

The following project architecture of Risk game is based on MVC Pattern (Model View Controller) consists of various entities which are as follows:

**Views:** Views are the main Design which is shown in the below screen. There are mainly 6 design view which are used in making of the risk games. Below are the views explained with their respective working.

From the Menu screen below we have few options which says the following:

File 🡪Play Game 🡪 New Game ***Fig No -3***

File 🡪Play Game 🡪Tournament ***Fig No -4***

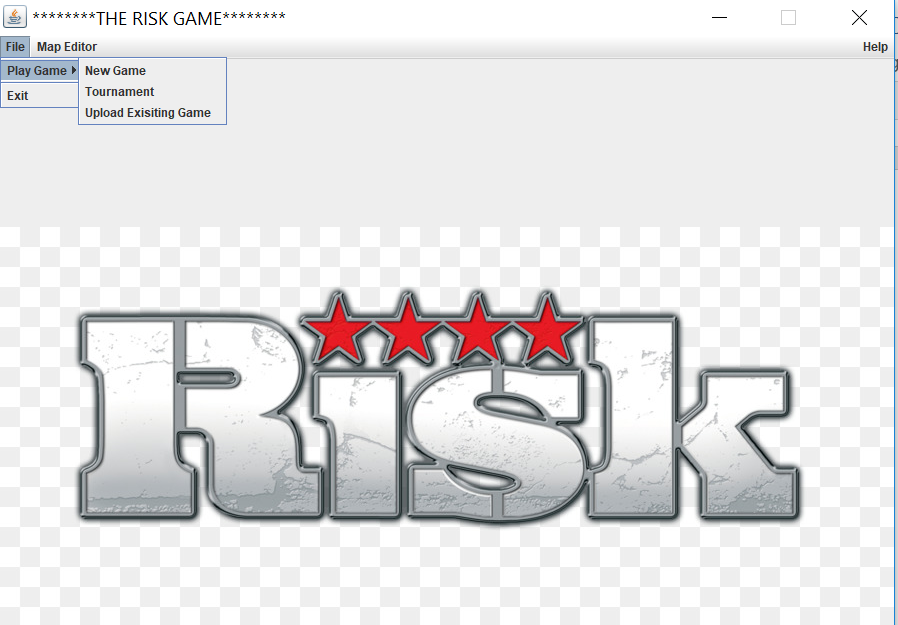
File 🡪Play Game 🡪Uploading Existing Game ***Fig No -9***

File 🡪Exit

Map Editor 🡪 Create New Map ***Fig No -10***

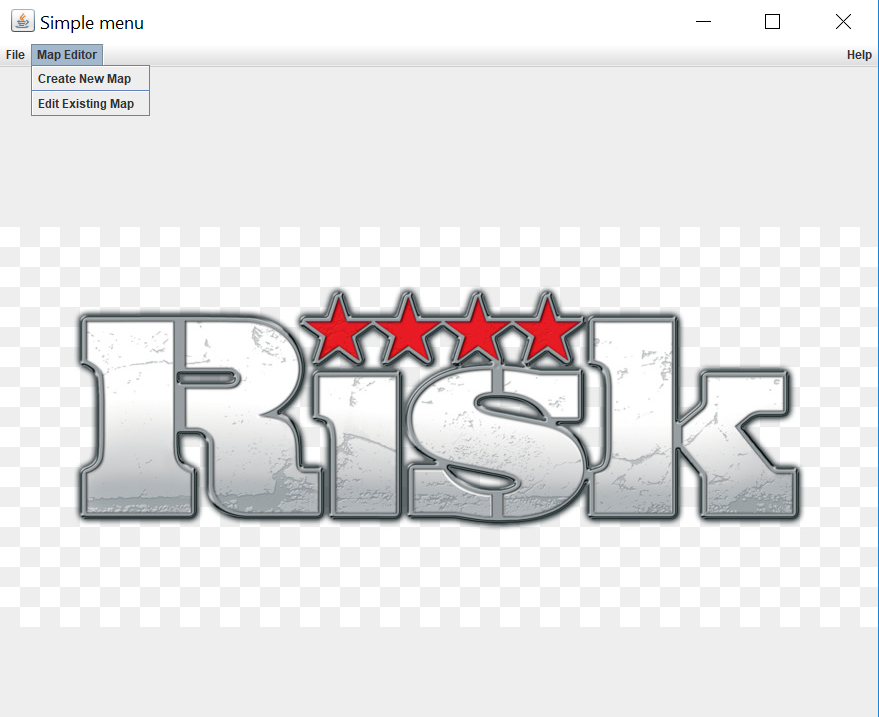
Map Editor 🡪Edit Existing Map ***Fig No -11***

1)**MainMenuScreen**: Main Menu screen show the option to start a new game or to exit the game



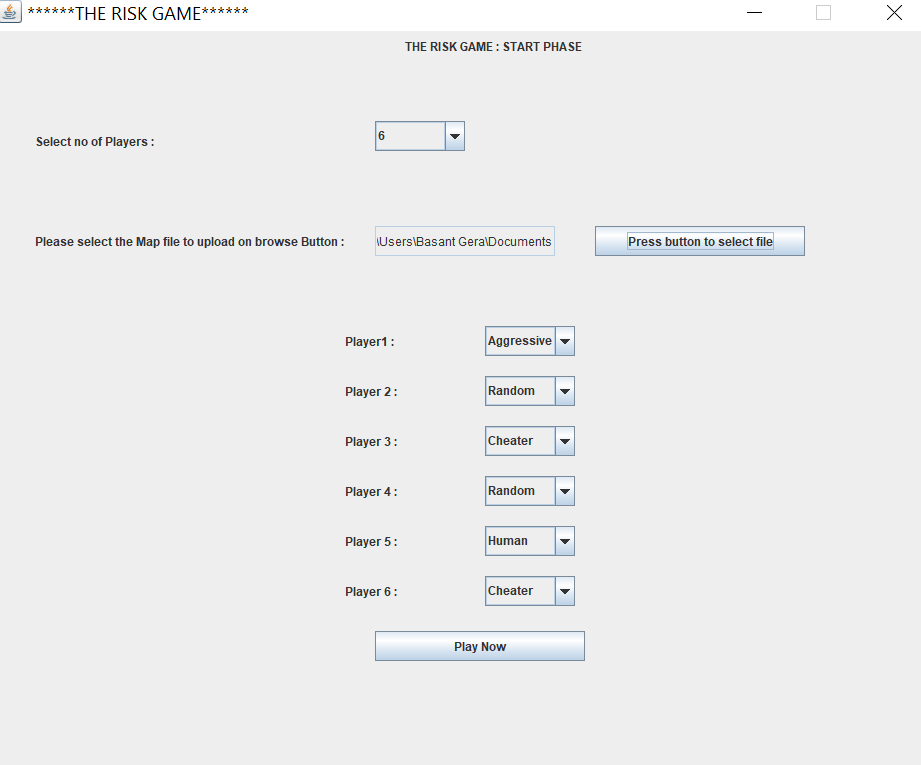
***Fig No – 2***

**MainMenuScreen**: Main Menu screen show the option to create new map or edit an existing map.



2)**PlayNewGameView**: Play new game is the screen which will show details to upload the file and select the no of player who wants to play the game.

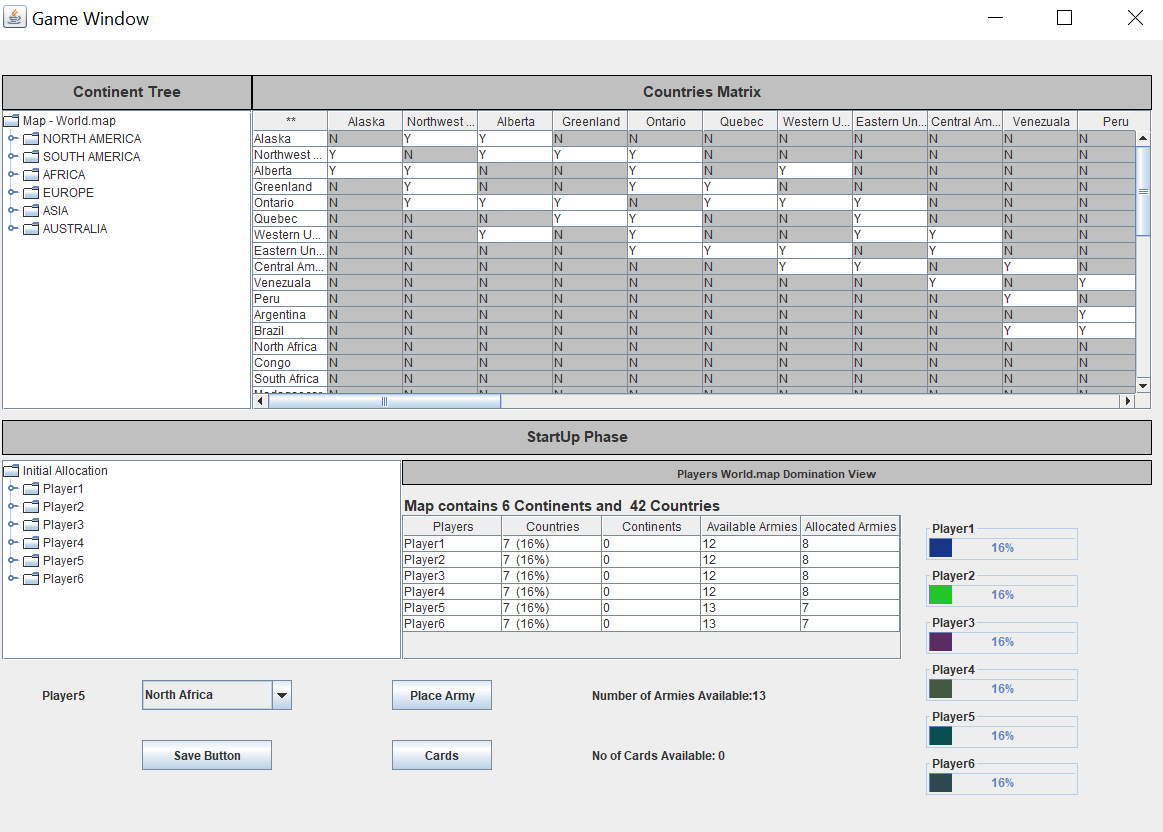
Below figure shows the no of players selected from the combo box and file is selected via browse functionality. Fig ***No -3***



***Fig No – 3***

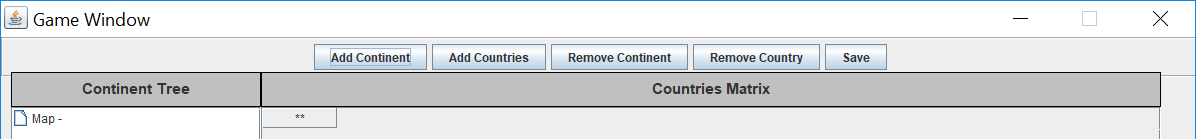
3) **Game Window Screen View**: It has a continent and country tree on the left-hand side and on the right-hand side it shows country by country matrix which represents N & Y in which Y represents that countries are neighbor of each other and N represents Countries are not adjacent to each other and are far away from each other

The startup phase explains the no of players in which counties are allocated randomly based on round robin fashion. And on the right-hand side shows the player w.r.t count of the countries. As shown ***in Fig No 4.***



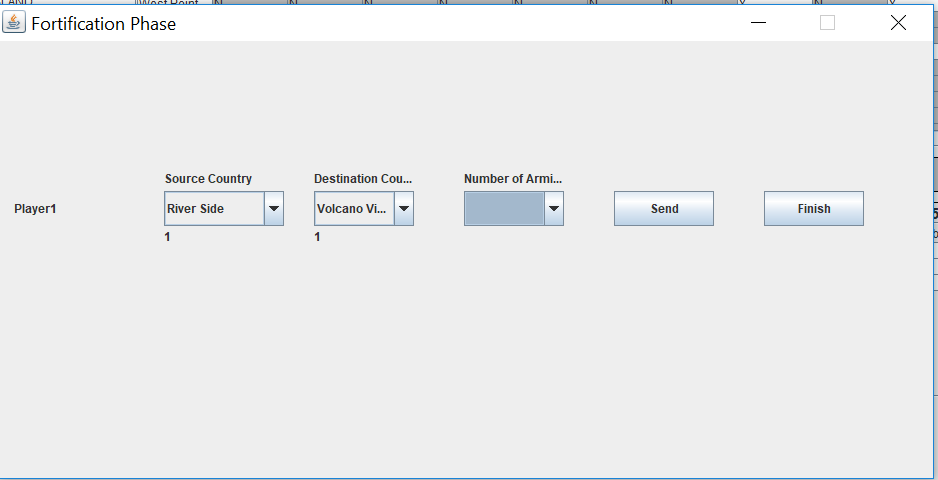
***Fig No – 4***

4)**Toolbar View**: Toolbar view has the following option Add a continent & Add a country & Remove Country and Remove Continent and save the existing map with country and content which you try to save. ***Fig No -5***



***Fig No -5***

5)Fortification View: Fortification phase is a phase in which we need to allocate the army to respective countries if a respective player has. ***Fig No -6***



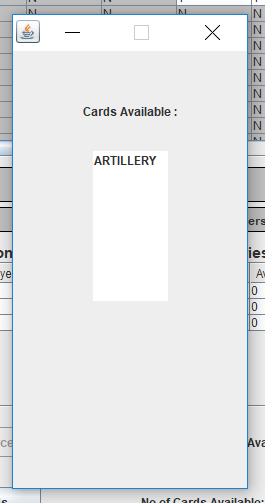
***Fig No -6***

6)Attack View: Attack view is the phase in which we need Attack is done between Source country and Destination country and them based on the same dice are rolled and winning and losing of a no of armies is decided. At the same time on every move winning is decided and a card is allocated. Please see the figure which you can see below.



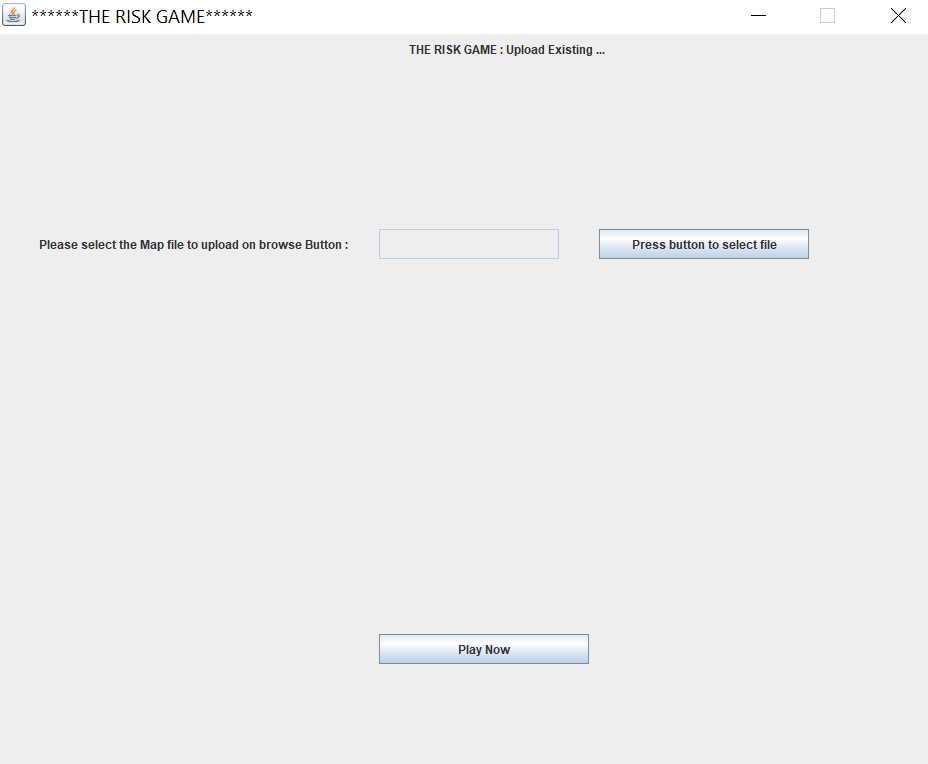
***Fig No -7***

7)**Card Screen**: Card are allocated 2 times in the game. When game starts then a player can allocate him a card. Apart from that when winning take place after attack phase and then card is given to winning country. Please have a look on the screenshot below.



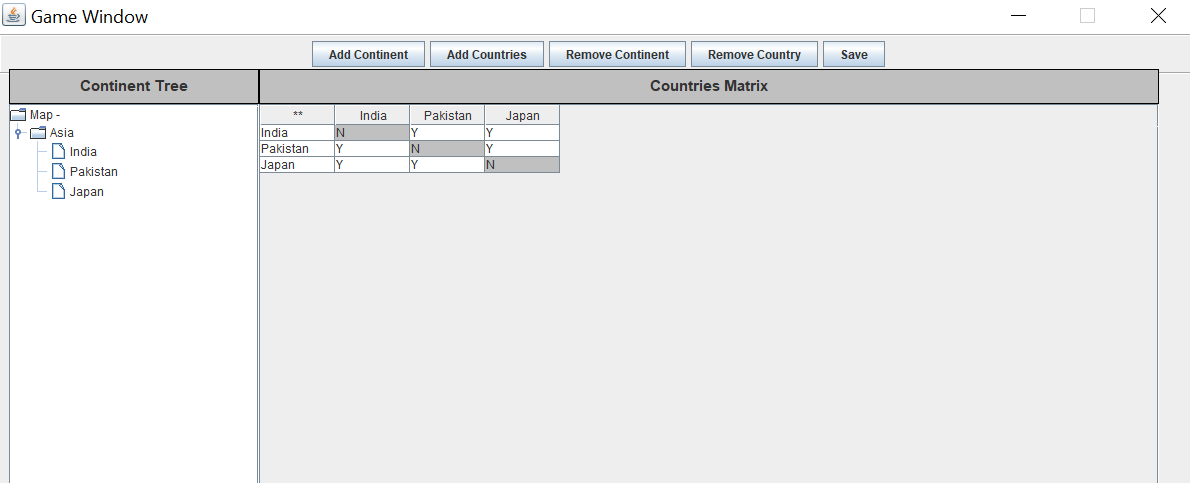
***Fig No -8***

8) **Upload Existing Screen:** This screen tells us to upload the serialize file and to convert it into deserialize so that paused game can be played.



***Fig No -9***

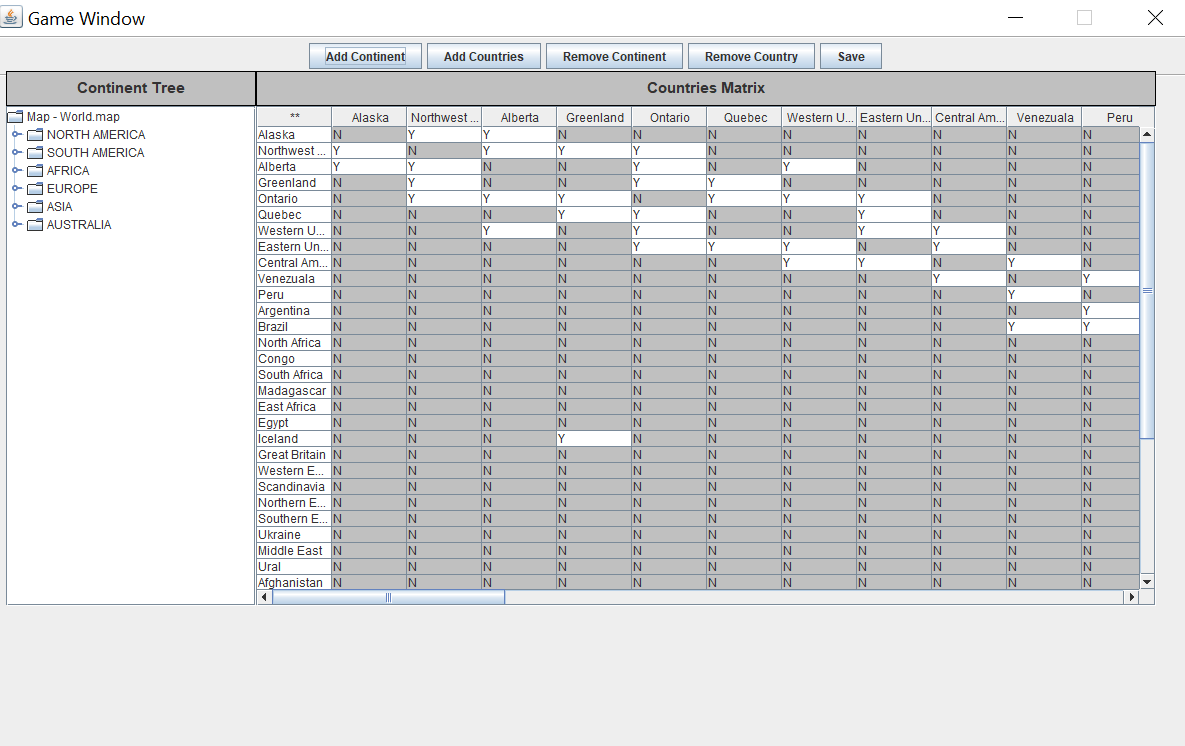
9) **Map Editor Screen (Create Map Editor Screen):** This screen tells us about the map editor screenwhich Continent and Country are added which will afterwards shown in matrix format.



***Fig No -10***

10)Edit Existing Map: This screen uploads the .map file and show it in the form of Counties/ Counties.

In this X represents Connected counties and Y represents Counties are not connected to each other.



***Fig No -11***

**Model: Models are the main entities over which Controller works. For successive builds we are using the following models which are as follows:**

* **ContinentModel:** It has getter and setter method regarding Continent details and their Corresponding country.
* **CountryModel:** It has getter and setter method regarding Country details and their control value, neighbors, armies and their respective latitude and longitude.
* **MapModel:** It has getter and setter method regarding Map Editor details. And their respective author name.
* **PlayerModel:** It has getter and setter method regarding Player details. And their respective player name and armies owned and countries owned.
* **GameWindowModel:** It has getter and setter method regarding Game Window details.
* **CardModel:** It has getter and setter method regarding Card details.
* **PlayerTypeModel**: It consists of type of player which exists in the player model which is used in the game. It also consists of enumeration of strategies such as aggressive, benevolent, cheater, human and random player.
* **BehaviourStrategiesModel**: Behavior strategies are those in which strategies of how player performs such as aggressive, benevolent, cheater, human and random player.
* **AggressivePlayerModel:** Functionality of aggressive computer player strategy that focuses on attack (reinforces its strongest country, then always attack with it until it cannot attack anymore, then

fortifies in order to maximize aggregation of forces in one country).

* **HumanModel:** Functionality of human player that requires user interaction to make decisions.
* **RandomPlayerModel**: Functionality of random computer player strategy that reinforces random a random country, attacks a random number of times a random country, and fortifies a random

country, all following the standard rules for each phase.

* **BenevolentPlayerModel:** Functionality of benevolent computer player strategy that focuses on protecting its weak countries (reinforces its weakest countries, never attacks, then fortifies in order to

move armies to weaker countries).

* **CheaterPlayerModel:** Functionality of cheater computer player strategy whose reinforce () method doubles the number of armies on all its countries, whose attack () method automatically

conquers all the neighbors of all its countries, and whose fortify () method

doubles the number of armies on its countries that have neighbors that belong to other players.

**Controller:** Controllers is the main business logic which rotates the value from view to model. With the help of controller, we can write the business logic. We have made the following controller classes which are as follows:

**MapEdititorController:** Map Editor controller works as it tries to fetch data which shows list of countries in a matrix form in X, Y Coordinates in GAME WINDOW SCREEN VIEW.

**GameWindowController:** The purpose of Game Window Controller is to have the start up phase code and reinforcement phase in which calculations of armies are done and are allocated based on countries.

**NewGameController:** The purpose of New Game Controller is to have some functionality which shows the how a new game has been started with how many players and with respective map files.

**FortificationController:** Fortification phase tells us how many country’s respective armies we have fortified and allocate if we have any extra army which is left so that we can make countries armies stronger by sending the army from source to destination by increment of +1.

**AttackController:** The purpose of Attack controller is consists of 2 main things which are Alloutattack and Attack. In which selection of source and destination countries are done by player and selection of dice and randomly allocation of dice take place which tells us which whether players won or defender.

At the same time allocation of army also take place if Alloutattack is button is pressed than Player and defender plays till one will not loose the game and get all his card and armies.

**GameController**: The purpose of Game controller contains phases such as place army and reinforcement phase and contains creation of Game model which helps to make an object of a game.

**Tournament Controller**: A tournament is collection of game which is to be played by user. The tournament view consists of no of map selected, no of turns, no of games played and no of player.

Uploading Existing game view: This view takes the serialize paused game and deserialize it to play the paused games.

**Utilities:** Utilities are the some of the helper classes which we have made which will be used throughout the various View. We can also say things which are repetitive can be wrote here and can be used back by any view or controller.

**Upload existing game controller**: The controller file consists of business logic of upload existing game controller.

We have made the following helper classes which are as follows:

**Contanta.java:** This file tells us the screen size which will be used by views

**MapTools.java:** We have used this file in which uploading of map file is done which repetitive in mainly many screens.