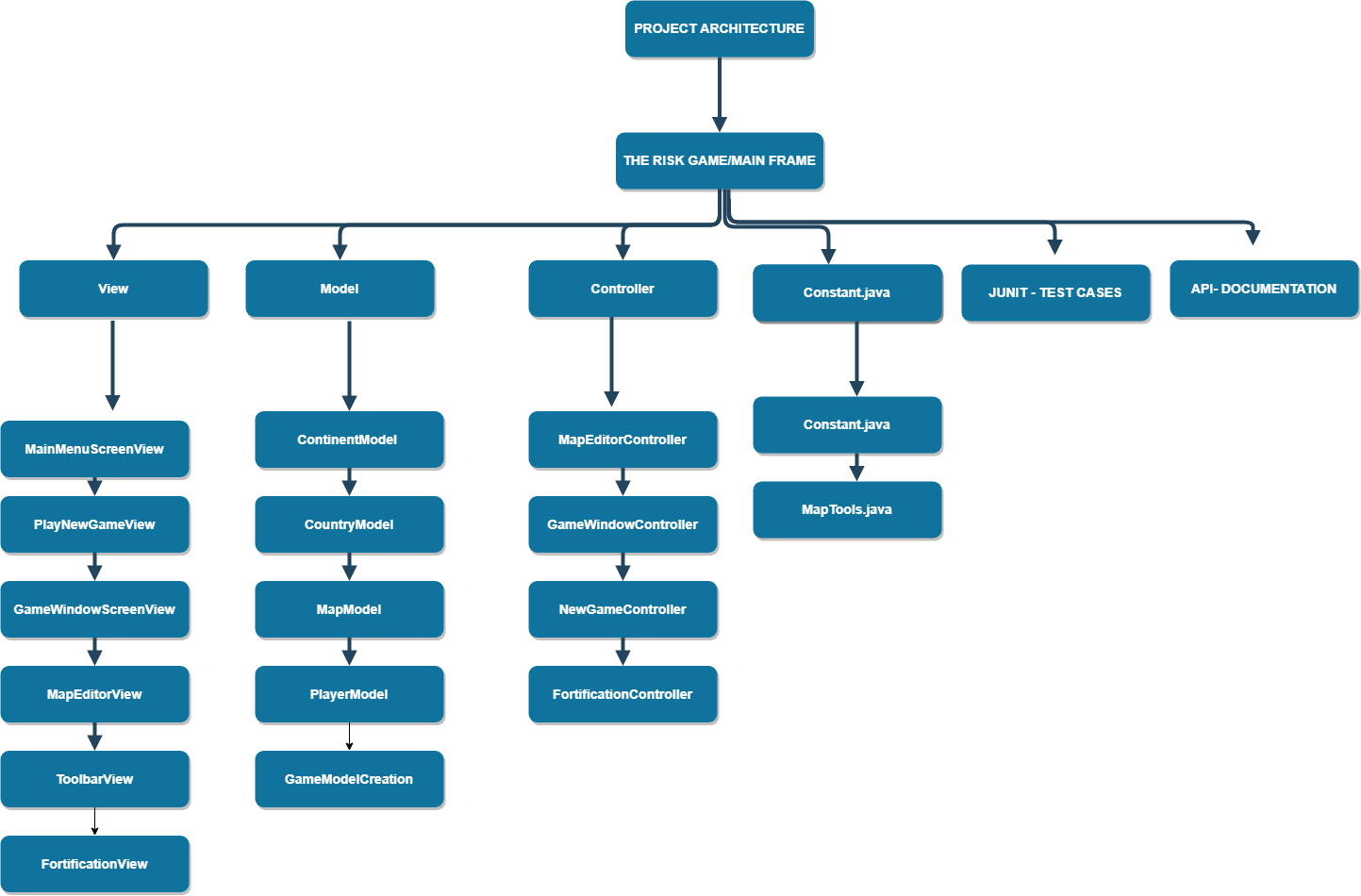
**Project Architecture – The Risk Game**



***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:

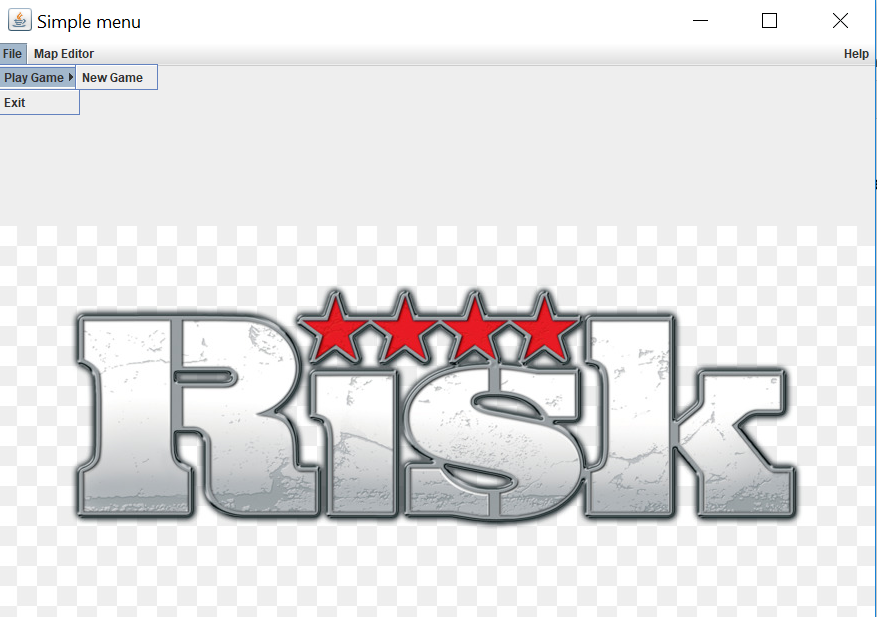
File 🡪Play 🡪Play New Game ***Fig No -2***

File 🡪Exit

Map Editor 🡪 Create New Map ***Fig No -3***

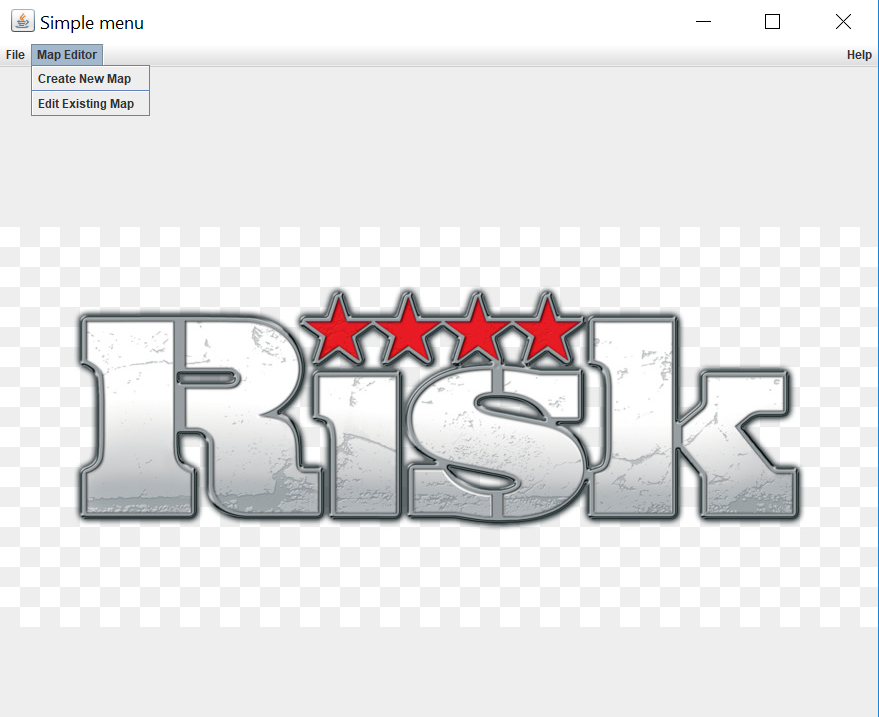
Map Editor 🡪Edit Existing Map

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



***Fig No -2***

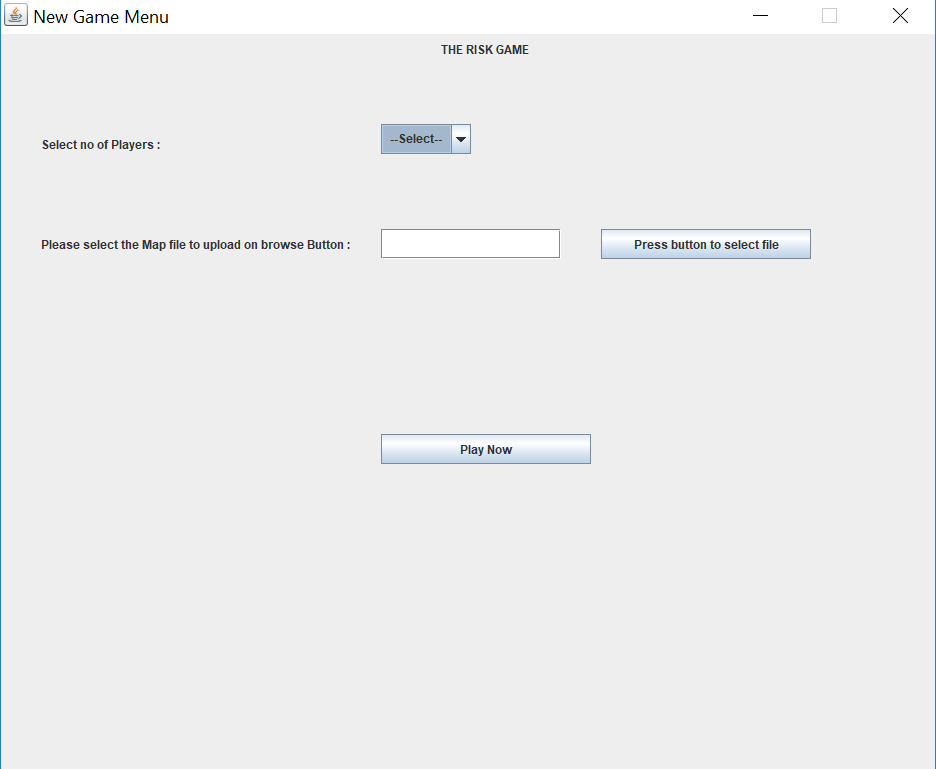
MainMenuScreen: Main Menu screen show the option to create new map or edit an existing map.



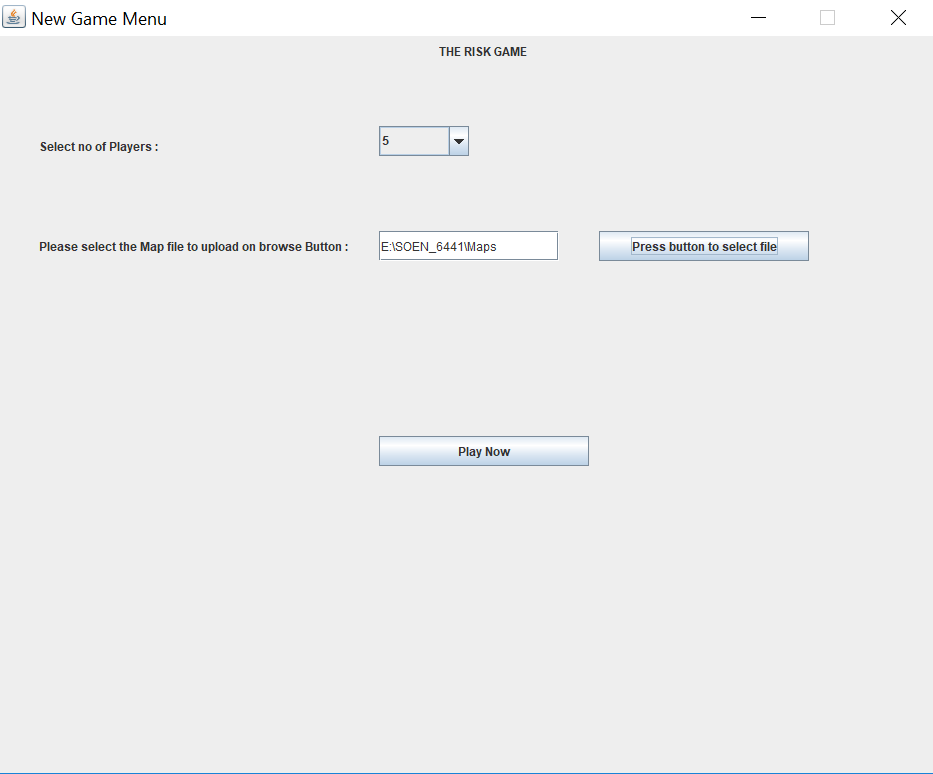
***Fig No -3***

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. As shown in Fig ***No -4 &***  ***Fig No -5***



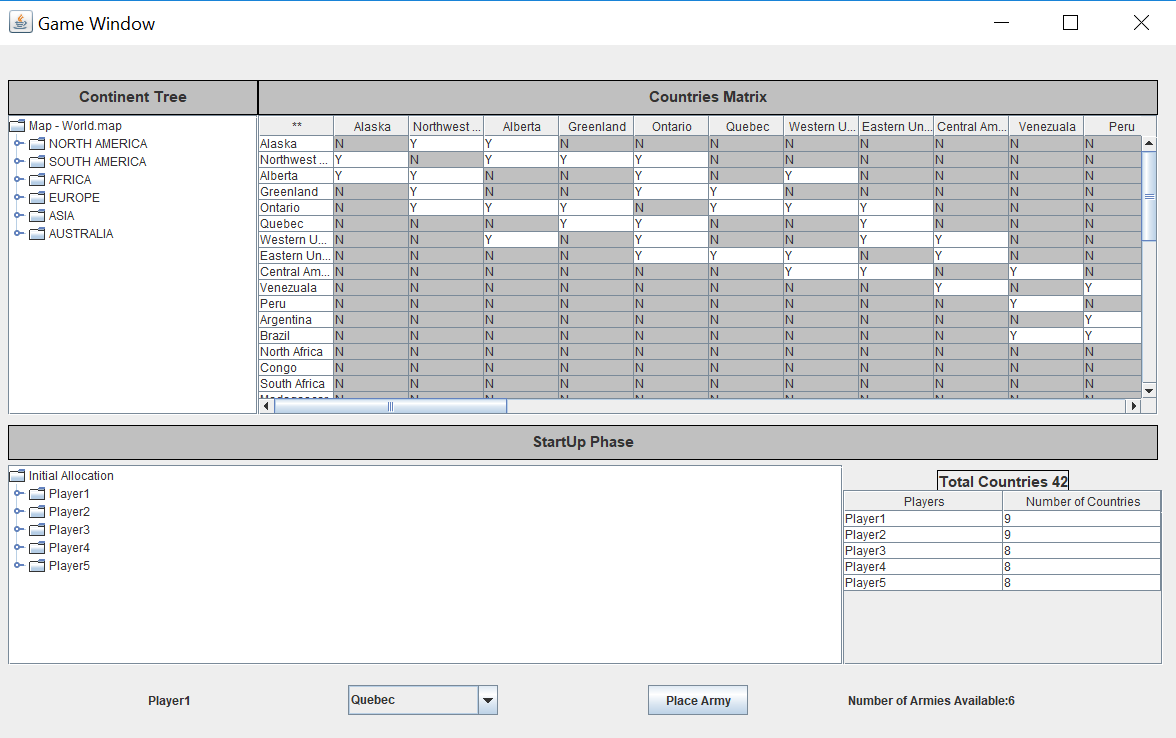
***Fig No -4***



***Fig No -5***

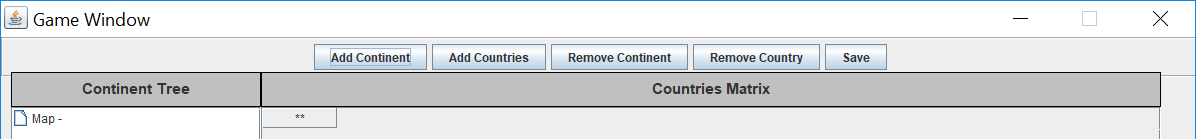
3) Game Window Screen View: It has a continent and country tree on the left-hand side and on the right-hand side it shows country by country matrix which represents N & Y in which Y represents that countries are neighbor of each other and N represents Countries are not adjacent to each other and are 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 -6***



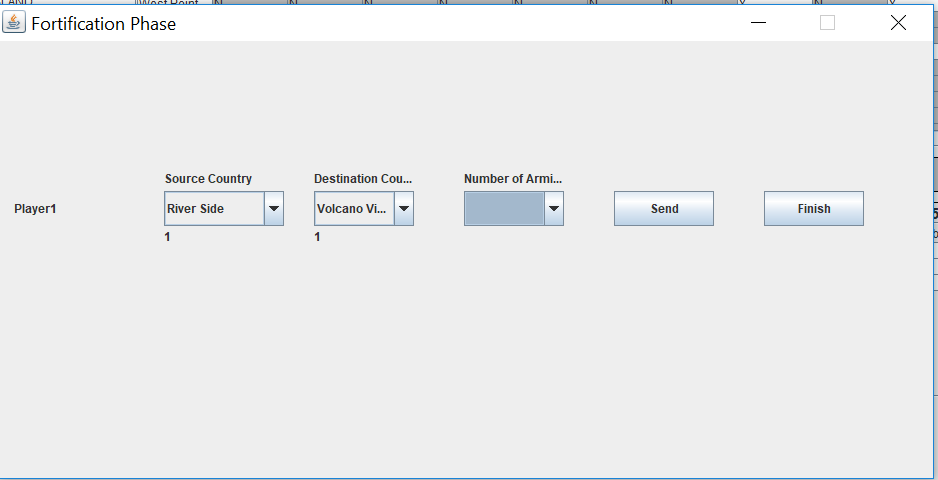
***Fig No -6***

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 -7***



***Fig No -7***

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 -8***



***Fig No -8***

**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.

**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.

**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.

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.

**Junit Test:** We are performing mainly 15 test cases for build 1 and making their respective mirror image and performing assert functions.

**API Documentation:** We are doing API documentation which will be there in folder using tags which are taught in class.