**DESIGN DOCUMENT**

**RISK GAME**

ARCHITECTURE:

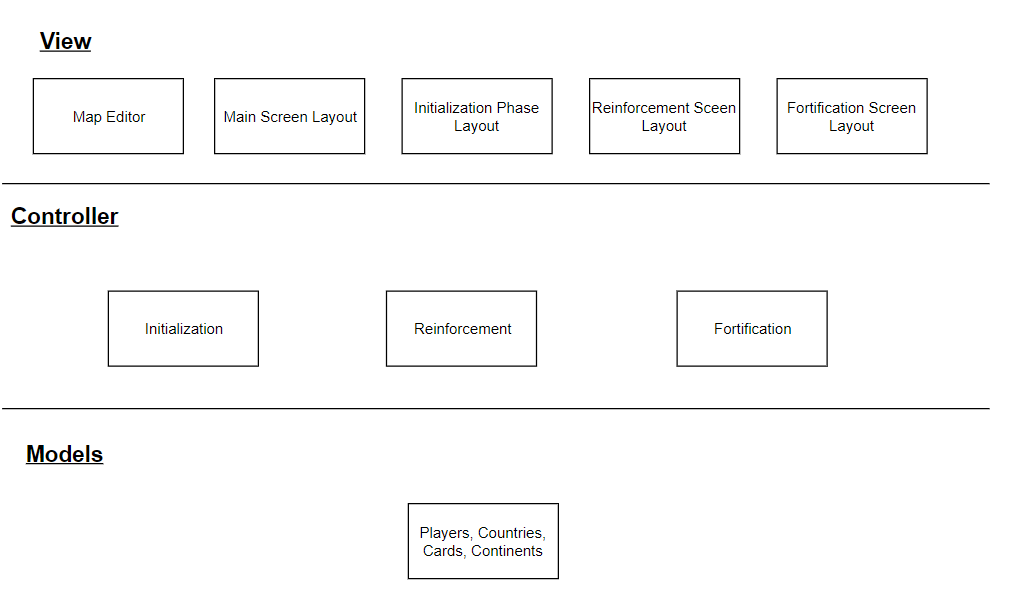
The project follows Model, View, Controller architecture and is divided into the following packages:

* Model - This package contains the structure of player, countries, continents and cards, and helps in organizing the game and manipulating the data in an orderly fashion.
* View -
* Controller - The the package of the project that contains all the classes pertaining to the phases of the game. The package contains the following packages:

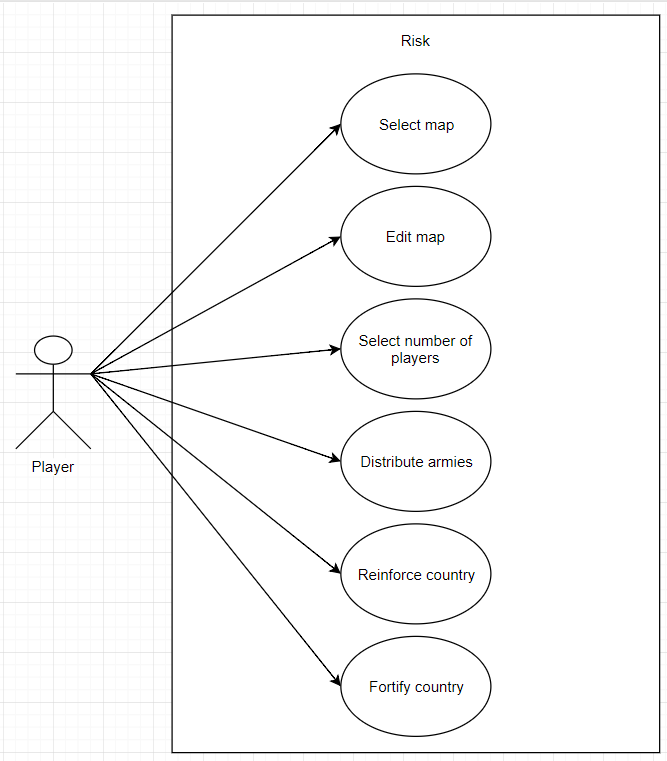
1. Initialization: This package contains the class with all the methods that are required to initialize a player before the game starts. Actions like random allocation of countries, initial allocation of armies are done in this class.
2. Reinforcement: This package contains the class with all the methods for calculating the number of armies that the player will get for the reinforcement phase and also the method that performs the reinforcement.
3. Fortification: This package contains the class that is used for the fortification phase of the game. It contains the method that allows the player to move armies from one country to another.

* Application - This package contains the main method which is used to start the application by calling the main menu layout method.
* Utilities - This package contains all the methods that will be used by the other packages frequently to perform certain actions.
* Constants - This package contains all the constant values for game phases and log levels that will be used throughout the project.
* Test - This package contains all the Junit test cases for individual modules except for the GUI of the project which has been tested manually for correctness.

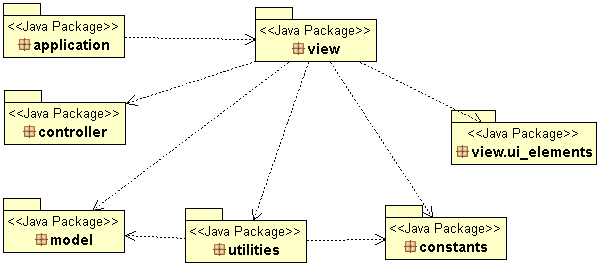
Below is a High Level Diagram for the architecture:



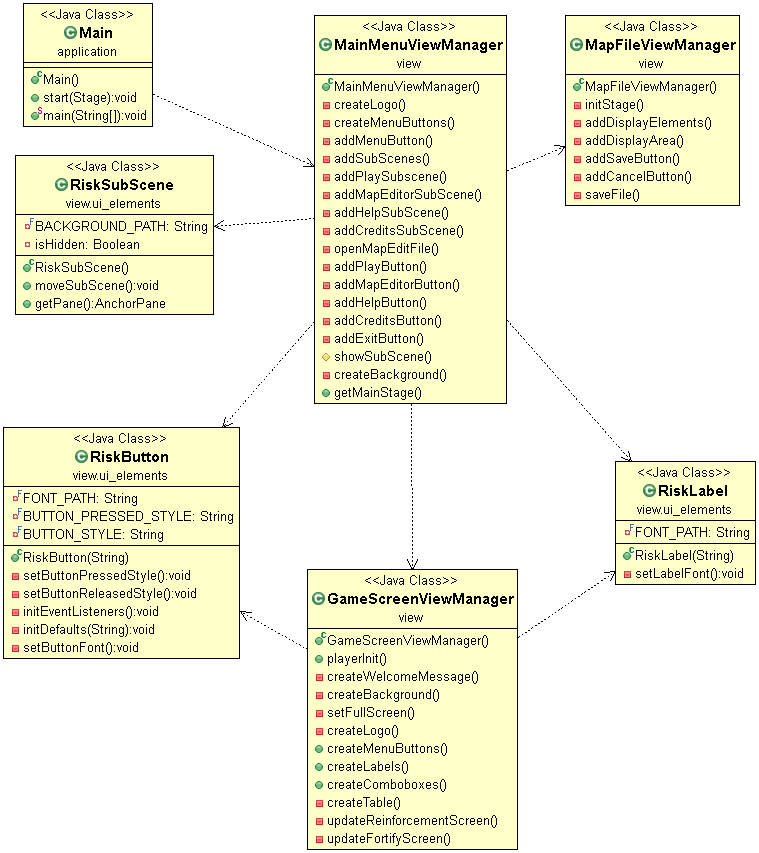
The following use case diagram illustrates the actions a player can perform:



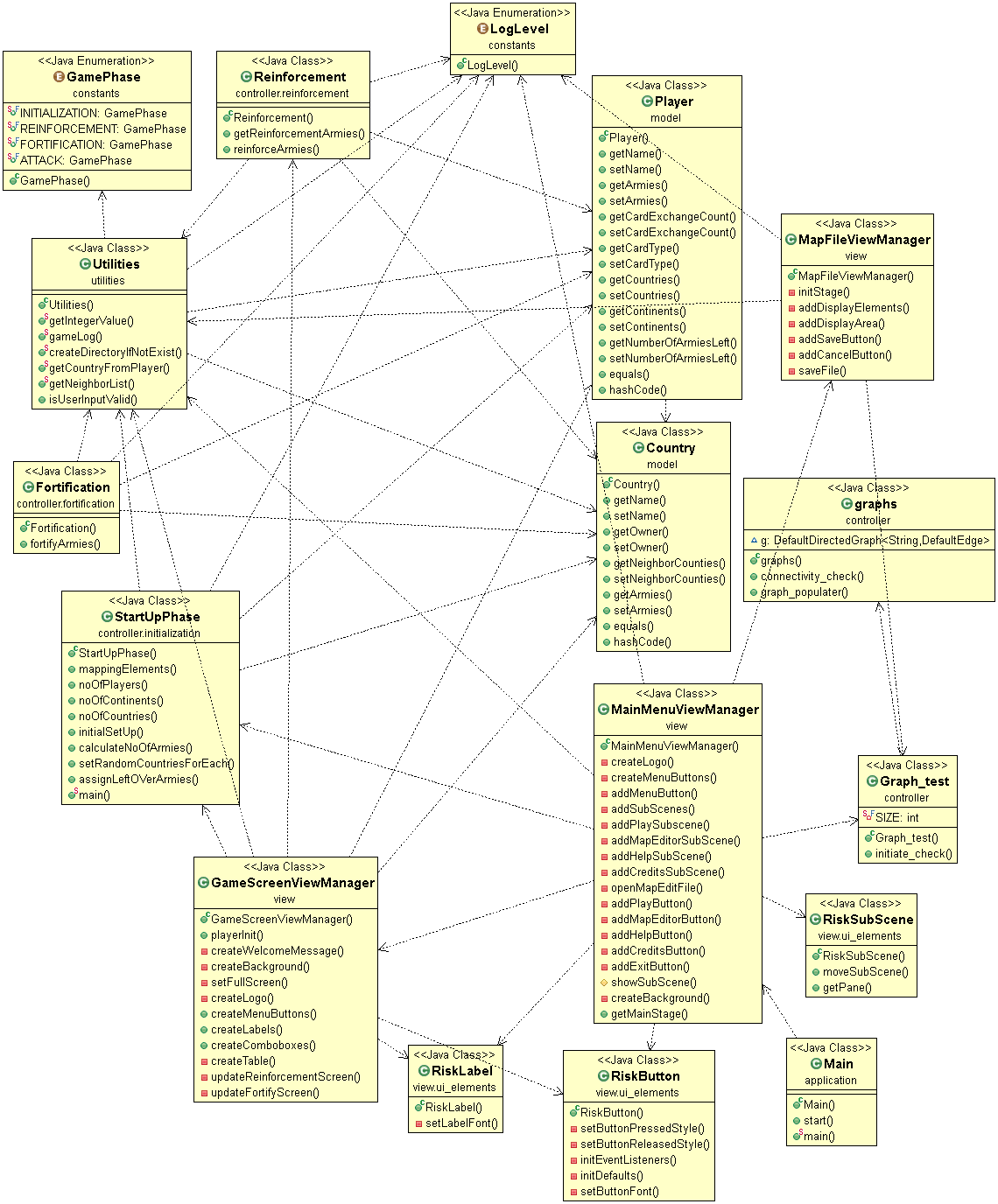
The following UML diagram shows the interaction between all the packages in the system:



The following UML shows the interaction between the classes in the view package with the main class:



The following UML diagram shows the interaction between all the classes:



CODING CONVENTIONS:

The project abides by the coding standards used in java. Following are the standards being used:

**Naming Conventions:**

1. All the methods have logical and self explanatory names and camel case is used to denote the methods.
2. All the local variables use camel case.
3. Constants used across the project are in a separate package and are denoted by capital letters.

**Comments:**

Javadoc comments are used before all the classes and the methods to describe the business logic behind it. Blocks of code that are not self explanatory have comments describing their functioning.

**Indentation and Layout:**

The code is indented using the standard tab i.e., 8 spaces. Different sections in the same class have been separated by blank lines. The braces for loops, methods and classes are kept in the same line.

**Exception handling:**

Logs are being maintained in a text file to capture both successful and unsuccessful moves. All exceptions are being handled and they can be tracked using the log files.

ADDITIONAL UTILITIES USED:

1. JavaFX (e(fx)clipse, version : 3.0.0 ) is used for the GUI of the project.
2. UMl Lab Class Diagram Editor(Version 1.13.0)is used to generate all the UML diagrams involving the packages and classes.
3. Javadoc is used to generate all the API documents for all the methods and classes.
4. JgraphT is the API used to create the vertices and edges between the vertices in the graph.