DEPARTMENT OF COMPUTER SCIENCE AND SOFTWARE

ENGINEERING

SOEN 6441, Fall 2019

RISK Game (Build-1)

Architecture Design

Submitted To: JABABO KHALED

Submitted By: **Team E**

Git URL: https://github.com/Surya64/APP_SOEN-6441_TeamE

Sr. No.	Name	Student ID
1	Surya Prakash Govindaraju	40085527
2	Shruthi Kondapura Venkataiah	40091427
3	Sahana Anantha	40085533
4	Sai Charan Teja Doddi	40076338
5	Dolly Modha	40084358

1. Introduction:

We have developed the project on RISK game by following the Model View Controller (MVC) software design architecture style.

2. Scope:

The scope of the build 1 is as per the instruction guidelines for the build:

Map:

- Create a new map file and validate it
- Edit an existing map file and validate it
- Add/Update/Delete Continent, Country
- Make sure that the integrity of the connected graph is maintained

Game Play:

- Assigning country to player
- Player can assign armies to each country in round robin manner
- With proper calculation of armies, Reinforcement phase is implemented
- With a valid fortification move, Fortification phase is implemented

3. Architecture Design

Model View Controller Architecture (MVC):

Following the MVC design pattern for implementing user interfaces on computers. It divides a given application into three interconnected parts. This is done to separate the internal representation of information from the ways the information is presented to and accepted from the user. The MVC design pattern decouples these major components allowing for efficient code reuse and parallel development.

Architecture Diagram of RISK Game development: MVC architectural style

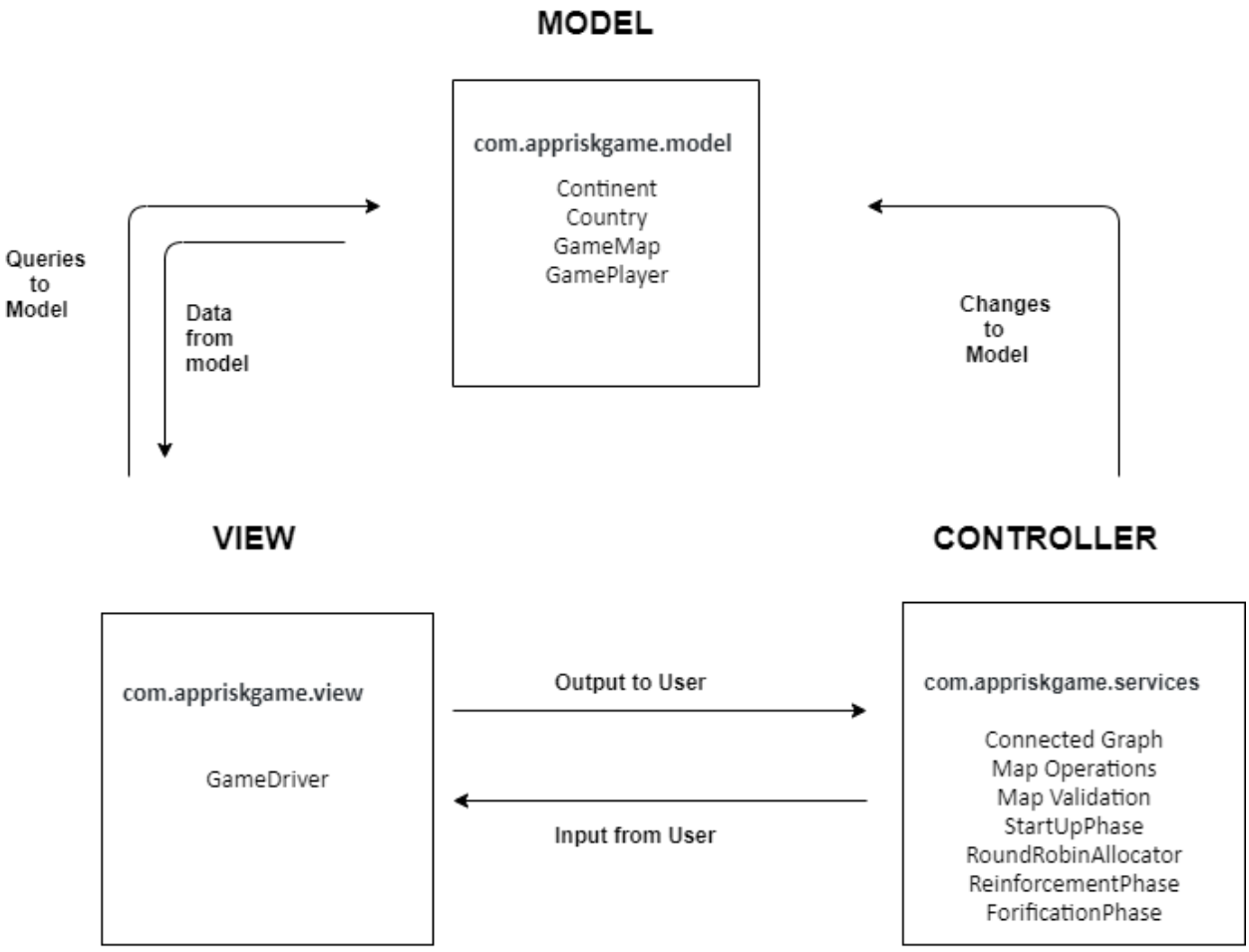

Fig : MVC Architecture Diagram of RISK development Project

Each Layer is described as below:

1. **Model** : Contains the classes which holds the logic of the Risk game
2. **View**: Contains the class which decided where the controls and display go.
3. **Controller** : Contains the classes which bridges between Model and View

4. Modules in each Layer

Modules in View: View has a single driver.

1. **GameDriver**: Class used to provide an interface for the players to interact with the game. It launches the game and in console the user can create or load map to begin the game.

Modules in Controller: We have the below classes.

1. **ConnectedGraph**: This class is used to check the connectivity if the countries form connected graph.
2. **MapOperations**: This class contains the methods for Add/Update/Delete Continent, Country, create new map, load and edit map operations.
3. **MapValidation**: This class is used to check the validation of map which is newly created or existing map is loaded.
4. **StartupPhase**: This class takes the data from GameMap module and initializes data for country, players and armies. It has a method which allocates the countries and armies to players.
5. **RoundRobinAllocator**: This class provides functionality for round robin traversal among the players of the game.
6. **ReinforcementPhase**: This class has methods which gives some extra armies to the player based on assigned countries.
7. **FortificationPhase**: This class has a method in which player can move armies from one country to another.

Modules in Model: We have the below classes.

1. **Continent** It contains all the information of the continent and a list of all the countries that belong to a continent
2. **Country**: This class is used to model the country. Each player will be assigned with few countries and player aims at occupying all the countries.
3. **GameMap**: Class used to model the map. It contains all the details of the map and its content.
4. **GamePlayer**: It contains all information related to a player and the number of armies assigned to the player