

**SOEN 6441 (Advance Programming Practices)**

**Project: Risk: Strategy Build 2**

GitHub repository - <https://github.com/credo92/RiskConquer>

Team 19:

|  |  |
| --- | --- |
| Member Name | Student Id |
| Rahul Goyal | 40047970 |
| Garvpreet Singh | 40045076 |
| Gurpreet Singh | 40027930 |
| Vipul Srivastav | 40027931 |

**Submitted to**: Dr. Joey Paquet

1. Contents

[1.1](#_1fob9te) Scope: 2

[1.2](#_3dy6vkm) Architecture Style 3

[1.3](#_1t3h5sf) Risk Strategy. Modules Description 3

[1.3.1](#_4d34og8) 1.1 Controllers 3

[1.3.2](#_2s8eyo1) 1.2 Entity 4

[1.3.3](#_17dp8vu) 1.3 Utility 4

[1.3.4](#_3rdcrjn) 1.4 Model 5

[1.3.5](#_26in1rg) 1.5 Validate 5

[1.3.6](#_lnxbz9) 1.6 Main 5

[1.3.7](#_35nkun2) 1.7 Exception 5

1.3.8 1.8 Strategy 5

[1.4](#_1ksv4uv) Technologies and Tools used: 6

[1.4.1](#_44sinio) Tools and technology used for the development of the game. 6

Introduction

This introduction provides an overview of the entire *Software Architecture Document* for the Risk Strategy game. It includes the purpose, scope, overview of the system.

## Scope:

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

## MapEditor: Covering the following below functionality, which is a form of a connected graph with proper interconnection between the continent and territories and abiding to the conquest map file format.

* Create a new map file
* Edit an existing map file
* Add/update/delete Continent
* Add/update/delete Territory
* Add/Delete Adjacent Territory
* Make sure that the integrity of the connected graph is maintained.
* Game Play: The game play covers:
  + Assigning territory to player
  + Player ability to assigning armies to each territory in round robin manner
  + Reinforcement phase, with proper calculation of armies
  + Fortification phase, with a valid fortification move.
* Furthermore, the scope of the build 2 is as per the instruction guidelines for the build:
  + Conquer View : The conqueror player dominator view shows percentage of territories owned by player graphically
  + Dice Roll: The dice roll covers:
    - Rolling the dice
    - Decide who wins on the basis of values of number of dice rolled by attacker and defender.
    - Validates the number of dices attacker/defender can use in single turn.
  + Player: Observer pattern is applied to this class containing methods mainly attack, reinforcement and fortification.
  + Card: Card functionality covers: -
    - Assigning card to a player after he wins.
    - Trade card player has valid set of cards
    - Open card window before reinforcement.
* Furthermore, the scope of the build 3 is

Behavior Strategies: -

• A human player that requires user interaction to make decisions.

• An 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).

• A 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).

• A 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.

• A 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.

Tournament Mode

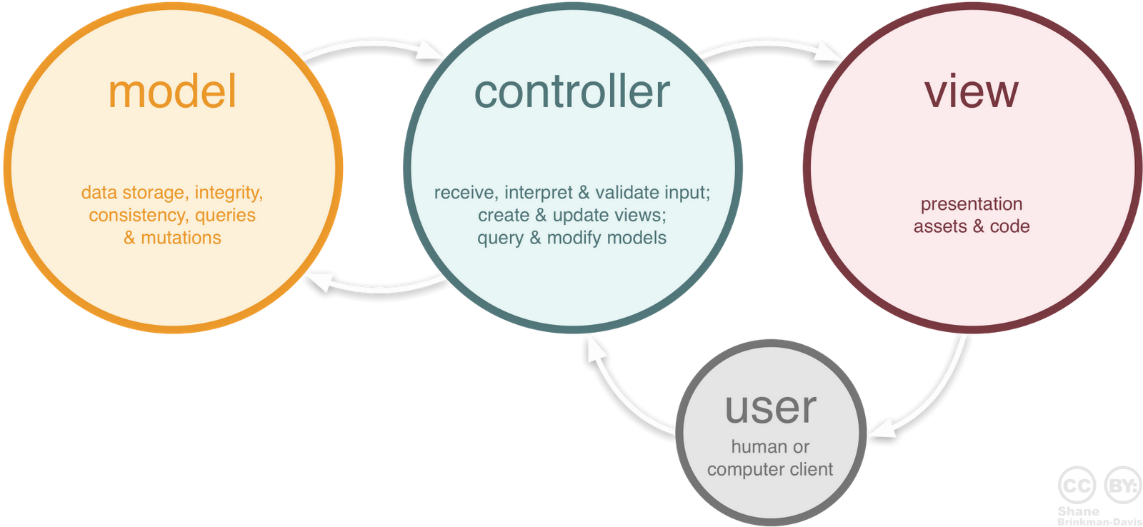
A tournament starts with the user choosing M = 1 to 5 different maps, P = 2 to 4 different computer players strategies, G = 1 to 5 games to be played on each map, D = 10 to 50 maximum number of turns for each game. A tournament is then automatically played by playing G games on each of the M different maps between the chosen computer player strategies. In order to minimize run completion time, each game should be declared a draw after D turns. Once started, the tournament plays all the games automatically without user interaction.

Game State

As a game is being played, allow the user to save the game in progress to a file, and allow the user to load the game in exactly the same state as saved.

## Architecture Style

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 internal representations of information from the ways 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.



## Risk Strategy. Modules Description

### 1.1 Controllers

The Controllers folder includes the Game and Map controller modules of the risk game.

|  |  |
| --- | --- |
| **File\_Name** | **Description** |
| GamePlayController | The class acts as a mediator between the GameModel class and the GameView file. It captures all the user action like :   * Creation of player * Assigning armies * Assigning cards and updates the GameView based on the data changed published by the GameModel class. * It also serves to all the request issued by the GameView. |
| MapEditorController | This class act as a mediator between the GameModel and the MapEditorView. It captures all the user actions like:   * Add/Update/delete Continent/Territory/Adjacent Territories * It update the MapView based on the data changed published by the MapModel. * It also serves to all the request issue by the MapView |
| CardController | This class act as a mediator between the CardModel and the CardView. It captures all the user actions like:   * Creation of card * Creation of valid card sets * Card Trade and assign armies according to the rules of the RISK game. |
| DiceRollController | This class acts as a mediator between the DiceModel and the Diceview file. It captures all the user action like:   * Setting dice values after roll * Creation of dice as per number of armies * Other operations like move armies, cancellation of dice roll, continuation of dice roll. |
| TournamentController | This class acts as a mediator between the TournamentModel and the Tournamentview file. It captures all the user action like:  Choose number of maps, turns and different type of players.  Play tournament according to rules of different strategies.  At the end shows result in the console. |

### 1.2 Entity

The Entity folder includes all the entities used in the game.

|  |  |
| --- | --- |
| **File\_Name** | **Description** |
| Map | It contains all the information of the Map file like: author, name  It contains a list of the continents that forms a map |
| Continent | It contains all the information related to the continent like , name , control value  It contain a list of all the territories that belong to a continent. |
| Territory | It contains all the information related to the territory like , name, x/y coordinates  It also maintains a reference to which continent the territory belongs  It also maintains a list of all the adjacent territory of this territory.  It contains the count of armies currently residing on the territory  It contains a reference to the player, indicating which player holds the territory. |
| Player | It contains all the information related to a player, like player name, list of territory assigned to the player.  It contains the number of armies associated with the player. |
| Card | It contains all the information related to card.  It contains territory name and card type which is a type of army. |
| GameState | It contains all the information about saving and loading game.  It contains all the information about state through GamePlayController |

### 1.3 Utility

The Utility folder includes all the elements used in the map file.

|  |  |
| --- | --- |
| **File\_Name** | **Description** |
| MapFileParser | This class is responsible for reading the Conquest map file format and parsing in to Map object. It also checks for the validity of the data of the map file. |
| MapFileWriter | This class is responsible for writing the Map object to the file in the same format as read from the conquest file |
| MapUtil | Contains all the utility method like: pushing data to console, saving map object, opening a dialogue box. |
| Gameutil | Contains all the utility method like: pushing data to console, opening a dialogue box, disabling and enabling controls. |

### 1.4 Model

The Model folder includes all the phases in the risk game

|  |  |
| --- | --- |
| **File\_Name** | **Description** |
| GameModel | This class represents the Game Model like creating players, assigning territory to the player, calculating the reinforcement armies, get the continents are owned by the players. |
| MapModel | This class represents the Map model and perform operation like , add/update/delete territory, add/update/delete continent. |
| PlayerGamePhase | This class represents the Player Model. This class acts as an Observer as well as Observable. It consists of: -   * All the three phases of the risk game (Reinforcement, Attack, Fortify) * Validation for attacks and moves * Creation of Players |
| Dice Model | This class represents the Dice Model and perform operations: -   * Assigning values to each dice randomly * Validation before dice rolls * Assigning Territory and move armies when player wins |
| Card Model | This class represents the Card Model and contains information about assiging and trading card between players. |
| Player Word Dominator | This class represents the Player Word Dominator and perform operations like populate world domination data. |
| PlayerSelectionModel | This class represents the player selection object which contains all the player objects and types of the players. |
| Tournament Model | This class contains represents tournament and contains information like play tournament and more. |

### 1.5 Validate

The Validate folder includes all the phases in the risk game

|  |  |
| --- | --- |
| **File\_Name** | **Description** |
| MapValidator | This class represents the validation functionality of the map. It validate all the possible combination of Continents and territory and making sure that the remain as a connected graph. |

### 1.6 Main

The Main folder includes all the phases in the risk game

|  |  |
| --- | --- |
| **File\_Name** | **Description** |
| CreateMap | Helps to load the new map screen |
| GamePlay | Helps to load the Game play screen |
| Main | Main entry point for the application |
| MapEditor | Help to load the map editor screen |
| MapOption | Helps to load the map options screen |
| LoadGamePlay | Main entry point for loading and choosing saved state file. |
| TournamentPlay | Help to load the fxml or view of tournament screen. |

### 1.7 Exception

The Exception folder includes all the phases in the risk game

|  |  |
| --- | --- |
| **File\_Name** | **Description** |
| InvalidMapException | Custom exception class to manage exception of the game related to map. |
| InvalidGameMoveException | Custom exception class to manage exception of the game related to the game play. |

### 1.8 Strategy

The Main folder includes all the phases in the risk game

|  |  |
| --- | --- |
| **File\_Name** | **Description** |
| AggressiveStrategy | Overrides methods attackPhase, ReinforcementPhase and Fortification phase as per Rules |
| BenevolentStrategy | Overrides methods attackPhase, ReinforcementPhase and Fortification phase as per Rules |
| RandomStrategy | Overrides methods attackPhase, ReinforcementPhase and Fortification phase as per Rules |
| CheaterStrategy | Overrides methods attackPhase, ReinforcementPhase and Fortification phase as per Rules |
| PlayerBehaviourStrategy | Interface which contains methods attackPhase, ReinforcementPhase and Fortification phase for other strategies to get override. |
| HumanStrategy | Overrides methods attackPhase, ReinforcementPhase and Fortification phase as per Rules |

## Technologies and Tools used:

## Tools and technology used for the development of the game.

|  |  |
| --- | --- |
| **Technology and Tools** | **Description** |
| Eclipse | IDE for the game development |
| Maven | Maven as a build automation tool to manage all project dependencies. |
| JavaFx | Library to control the UI components of the Risk Game |
| FXML Editor | To generate the UI components for the Risk Game |
| Junit 4 | Junit 4 for writing test cases |