

**SOEN 6441 (Advance Programming Practices)**

**Project: Risk: Strategy Build 1**

**Submitted to**: Dr. Joey Paquet

Team 7:

|  |  |
| --- | --- |
| Nirav Patel | 40047970 |
| Karthik B P | 40094485 |
| Krishnan Krishnamoorthy | 40089054 |
| Charles Jebalitherson Augustin Moses | 40084105 |
| Rohan Deepak Paspallu | 40093648 |

Table of Content

[1.1 Scope 3](#_Toc527373736)

[1.2 Architecture Style 3](#_Toc527373737)

[1.3 Risk Strategy. Modules Description 4](#_Toc527373738)

[1.3.1 Controllers 4](#_Toc527373739)

[1.3.2 Models 5](#_Toc527373740)

[1.3.3 Utility 5](#_Toc527373741)

[1.3.4 Main 6](#_Toc527373742)

[1.3.5 Exception 6](#_Toc527373743)

[1.4 Technologies and Tools used: 6](#_Toc527373744)

[1.4.1 Tools and technology used for the development of the game. 6](#_Toc527373745)

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:
* Map Editor: 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.

## 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.

Controller

Entity

Risk Game Views [FXML]

Game Play

Map Editor Splash

Map Editor

Splash Screen

Updates the view based on business logic

Map Redactor Controller

Continent

Sets the attributes

Map Redactor Splash Controller

Map

Returns attribute values

Play Game Controller

Player

Splash Controller

Territory

Utilizes set of common functions

Common Map Utils

Map Operations

Game Utils

Exceptions

Utilities

Map Verifier

Map Reader

Map Writer

## 

## Risk Strategy. Modules Description

### Controllers

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

| **File\_Name** | **Description** |
| --- | --- |
| PlayGameController.java | The class acts as a mediator between the GameUtils class and the playgame.fxml file. It captures all the user action like :   * Creation of player * Assigning armies * All the three phases of the risk game (Reinforcement, Attack, Fortify)   It updates the GameView based on the data changed published by the GameUtils class.  It also serves to all the request issued by the GameView. |
| MapRedactorController.java | This class act as a mediator between the MapOperations and the mapeditor.fxml. It captures all the user actions like:   * Add/Update/delete Continent/Territory/Adjacent Territories   It updates the MapView based on the data changed published by the MapOperations.  It also serves to all the request issue by the MapView |
| MapRedactorSplashController.java | This class controls the user action from mapeditorsplash.fxml and calls new controller to open appropriate pane of javaFX. i.e Based on editing existing map or creating new map |
| [SplashController.java](https://bitbucket.org/niravjdn/risksoen6441/src/master/src/main/java/com/risk6441/controller/SplashController.java) | This class handles the user action frorm main screen such as opening map editor splash screen or opening new game screen. |
| DiceController.java | This class is the controller for the dice view. |
| CardExchangeController.java | This class is the controller for the card exchange view. |

### Entity

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

| **File Name** | **Description** |
| --- | --- |
| Map.java | It contains all the information of the Map file like: author, name  It contains a list of the continents that forms a map |
| Continent.java | 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.java | 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.java | 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.java | It contains all the information related to Card. |

### Maputils

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

| **File Name** | **Description** |
| --- | --- |
| [MapReader.java](https://bitbucket.org/niravjdn/risksoen6441/src/master/src/main/java/com/risk6441/maputils/MapReader.java) | 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. |
| [MapWriter.java](https://bitbucket.org/niravjdn/risksoen6441/src/master/src/main/java/com/risk6441/maputils/MapWriter.java) | This class is responsible for writing the Map object to the file in the same format as read from the conquest file |
| [CommonMapUtil.java](https://bitbucket.org/niravjdn/risksoen6441/src/master/src/main/java/com/risk6441/maputils/CommonMapUtil.java) | Contains all the utility method like: pushing data to console, saving map object, opening a dialogue box. |
|  |  |
| [MapOperations.java](https://bitbucket.org/niravjdn/risksoen6441/src/master/src/main/java/com/risk6441/maputils/MapOperations.java) | This class represents the Map model and perform operation like , add/update/delete territory, add/update/delete continent. |

### Gameutilis

| **File Name** | **Description** |
| --- | --- |
| [Config.java](https://bitbucket.org/niravjdn/risksoen6441/src/master/src/main/java/com/risk6441/maputils/MapReader.java) | This class defines global constans and message for the application. i.e no of armies according to number of players. |
| CardKind.java | Enumration for defining different card types. |

### Gameutilis

| **File Name** | **Description** |
| --- | --- |
| [GameUtils.java](https://bitbucket.org/niravjdn/risksoen6441/src/master/src/main/java/com/risk6441/maputils/MapReader.java) | This class handles the common openration for the game play phase such as enabling and disabling controls. It also shows alert box and performs some other common tasks. |

### Main

| **File\_Name** | **Description** |
| --- | --- |
| CardModel.java | This class handles the operation regarding classes. This clsss notifies the PlayGameController in case if it is changed. |
| PlayerModel.java | This class handles operations of the players such as reinforce, attack, fortification and many others. This class also notifies the PlayGameController if it is changed. |
| DiceModel.java | This class handles the operation related to dice such as rolling dice and comparing dice. This class also notifies the PlayerModel if it is changed . |
| WorldDominationModel.java | This class provides the data for the pie chart and bar chart to the PlayGameController. |

### Main

The Main folder includes all the phases in the risk game

| **File\_Name** | **Description** |
| --- | --- |
| Main.java | Main entry point for the application |
| MapEditor.java | Help to load the map editor screen |

### Exception

The Exception folder includes all the phases in the risk game

| **File\_Name** | **Description** |
| --- | --- |
| InvalidMapException | Custom exception class to manage exception of the map validation. |
| InvalidGameActionException | Custom exception class to manage exception of the game play phase. |

## Technologies and Tools used:

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

| **Technology and Tools** | **Description** |
| --- | --- |
| Eclipse Photon | 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 |
| Scene Builder | It is an open source JavaFX ecosystem used to design the UI of the game and gives an overall skeleton of events (e.g. button click, mouse drag...) to be implemented in controller. Separation of design and logic files allows for team members to quickly and easily focus on their specific layer of application development. |