Risk Game (Build-II)
Architectural Design

Advanced Programming Practices

SOEN 6441
Fall-2019

Team: Group U_J

Mehul Prajapati	(40076930)
Komal Panchal	(40130791)
Maryam Giahi	(40016260)
Mahmoudreza Entezami	(40058782)
Parth Patel	(40081116)
Shubham Ranadive	(40083991)

Index

Introduction

Develop a RISK game using Model View Controller (MVC) software design architecture with iterative development to deliver working modules in small builds. It was an effort to use extreme programming key features such as Pair programming, Collective ownership, Coding Standards and many more.

1. SCOPE

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

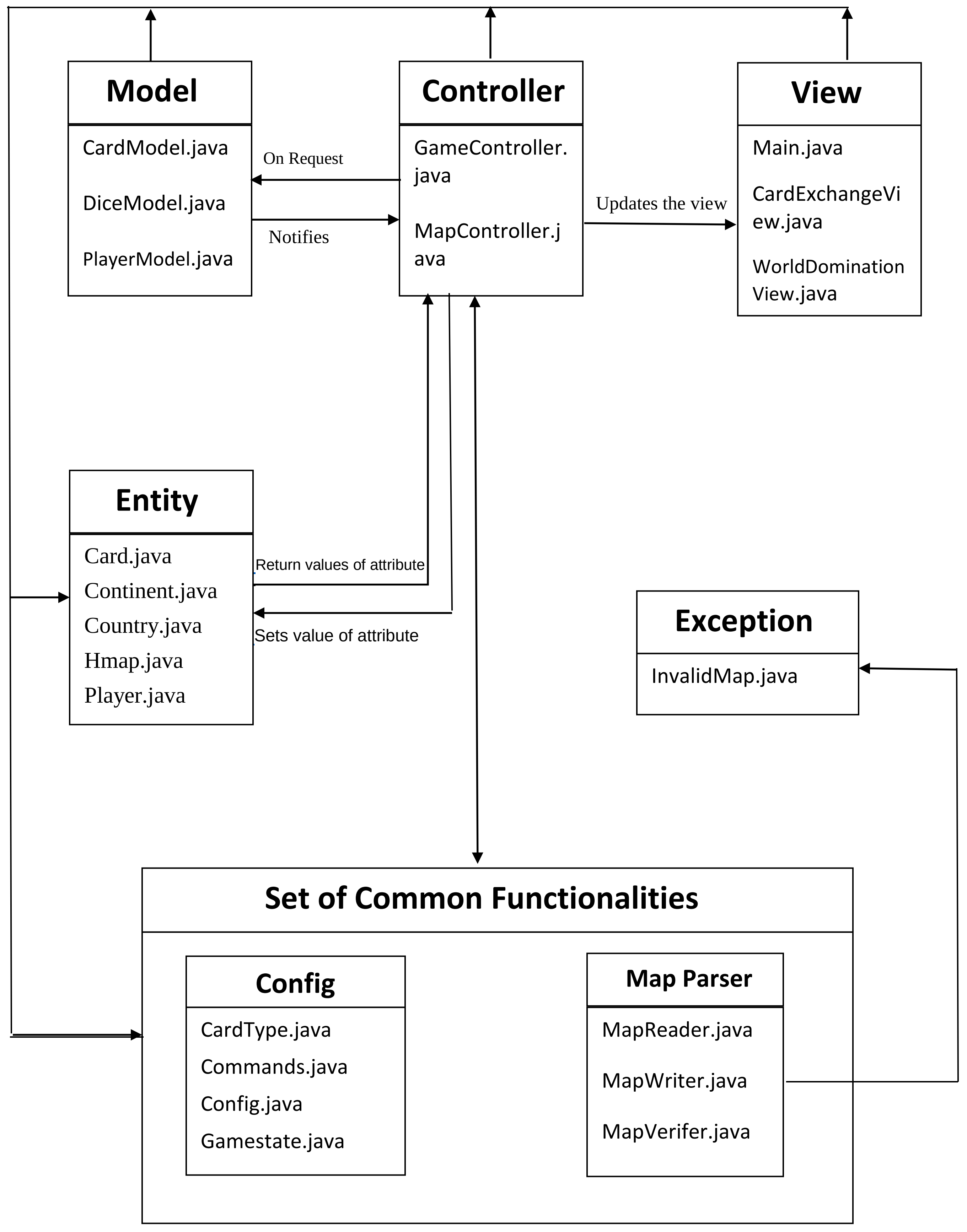
1.1 MAP EDITOR:

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

1.2 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.

2.Architecture Design



3.Modules Description

3.1 Controllers

File Name	Description
GameController.java	It parses Command line string from user input. It captures all user actions such as creation of player, assigning armies, and all three phases of the risk game.
MapController.java	It provides map commands to add and remove continents, countries and their neighboring countries.

3.2 Entity

File Name	Description
Hmap.java	It contains all the information of the Map and a list of the continents.
Continent.java	It contains all the information of the continent and a list of all the countries that belong to a continent.
Country.java	It contains the information of the country like name, a reference to which continent the country belongs, list of all the adjacent country, count of armies currently residing on the country.
Player.java	It contains all information related to a player and the number of armies assigned to the player.
Card.java	It contains all the information regarding to Card.

3.3 Exception

File Name	Description
InvalidMap.java	It manages exception of the map validation.

3.4 MainGame

File Name	Description
Main.java	Entry point for the application

3.5 MapParser

File Name	Description
MapReader.java	It reads the map file format and parsing in to Map object and also checks for the validity of the data of the map file.
MapVerifier.java	This class validates the map.
MapWriter.java	It is responsible for writing the Map object to the file.

3.6 Config

File Name	Description
CardType.java	Enumeration for defining 3 card types for the game.
Commands.java	Contains all the common method of the map.
Config.java	This class defines static properties for army configuration.
GameState.java	It contains game play phase commands.

3.7 Models

File Name	Description
CardModel.java	This class handles the operation regarding card.
DiceModel.java	This class also notifies the PlayerModel if it is changed and handles operation related to dice like rolling and comparing dice.
PlayerModel.java	It handles operation for players such as reinforce, fortification, and many others.

3.8 Views

File Name	Description
CardExchangeView.java	It will display all cards owned by the current player.
WorldDominationView.java	It displays the total number of armies owned by every player, continent controlled by every player and percentage of the map controlled by every player.

4.Test Cases (Junit) Description

4.1 Entity

File Name	**Description**
CardTest.java	This is a test class for testing methods of Card class.
ContinentTest.java	This is a test class for testing methods of Continent class.
CountryTest.java	This is a test class for testing methods of Country class.
PlayerTest.java	This is a test class for testing methods of Player class.
EntityTestSuite.java	This is a test class for running all test suits in Entity.

4.2 MapParser

File Name	**Description**
MapReaderTest.java	This is a test class for testing methods of Map Reader class.
MapVerifierTest.java	This is a test class for testing methods of Map Verifier class.
MapCommandsTest.java	This is a test class for testing methods of Map commands.
MapParserTestSuite.java	This is a test class for running all test suits in MapParser.

4.3 Model

File Name	**Description**
PlayerModelTest.java	This is a test class for testing methods of Player commands.
PlayerModelTestSuite.java	This is a test class for running all test suits in Model.

4.4 MainGame

File Name	**Description**
MainTestSuite.java	This is a test class for running all test suits (EntityTestSuite.java, MapParserTestSuite.java, PlayerModelTestSuite.java).

5.Tools & API

Tools	Description
Eclipse	IDE for the game development.
Git	It is Git code management system which gives one place to plan projects, collaborate on code test and deploy.
Junit4	It is used for writing test class.
JavaDoc	JavaDoc is automation tool to generate document to refer regarding libraries and modules of code.

6. Refactoring

- Changed architecture of the game from Object-Oriented to MVC

File	Description
PlayerModel.java	In our first phase, this class was for player commands. Now all the methods and all the game phases are included in this class. And this class is renamed from PlayerCommand to PlayerModel.
GameController.java	This class renamed as GameController.java from Commandparser.java. Initially all the commands were included in the previous class. Now all the models and observables are included in the new class.
MapReader.java	Initially during populate countries command a local copy of country list was used for assigning country. Now it gets the country list from the main Map object from Controller
Maingame.java	Initially command parsing was also done at some level in maingame.java but now it contains only the view and updates it according to the observables that is GameController.java

7.References:

- Rules Followed: https://www.wikihow.com/Play-Risk
- https://sourcemaking.com/refactoring/refactorings
- https://www.sourcetreeapp.com