SOEN 6441

ADVANCED PROGRAMMING PRACTICES

PROJECT: RISK DOMINATION GAME

ARCHITECTURAL DESIGN

TEAM 11

TEAM MEMBER	STUDENT ID
Zankhanaben Patel	40067635
Koshaben Patel	40094385
Piyush Thummar	40125029
Raj Mistry	40119206
Jaswanth Banavathu	40080737

SUBMITTED TO: JOEY PAQUET

Table of Contents

INTRODUCTION

1. Overview

RISK game is a strategy game where each player tries to conquer all countries on a map based upon specific turns.

Game is categorized into three different phases

- Reinforcement phase - The phase where player can add new armies to his countries.
- Attack phase - Player can decide and choose to attack countries which are owned by another player.
- Fortification phase - Player can move armies between their countries to fortify them.

Game can be played in two different modes:

- Single game mode - The original game play mode where all the commands are issued by the user. In this mode, the user can select the map and the number of players who will play the game. The game will finish when anyone of the player conquers all the countries.
- Tournament mode - This is the game play mode where all the operations are preceded without any user interaction. In this mode, the user can specify the behavior of the player among four different player behavior strategies. They are:
  Aggressive, Benevolent, Random and Cheater.
  Each behavior has its own set of characteristics.
  The game will only complete when all the turns and the number of games specified by the user are finished.

2. Technologies Adapted

- Spring Framework - Development framework used for Java.
- Java FX - Handling and rendering the User Interface elements.
- JUnit 4 - Unit testing framework used for testing unit test cases.

3. Architectural Design

Goal and architectural design of risk domination game is described in detail. The risk domination game is developed with iterative development and is released by keeping specific goals in mind providing small number of features in each iteration. Architectural design for risk game is MVC (Model View Controller).Each iteration is developed by using Extreme programming features such pair programming, collective ownership, coding standards and continuous integration.

Reasons for MVC (Model View Controller) design pattern suitable for risk domination game is as described below:

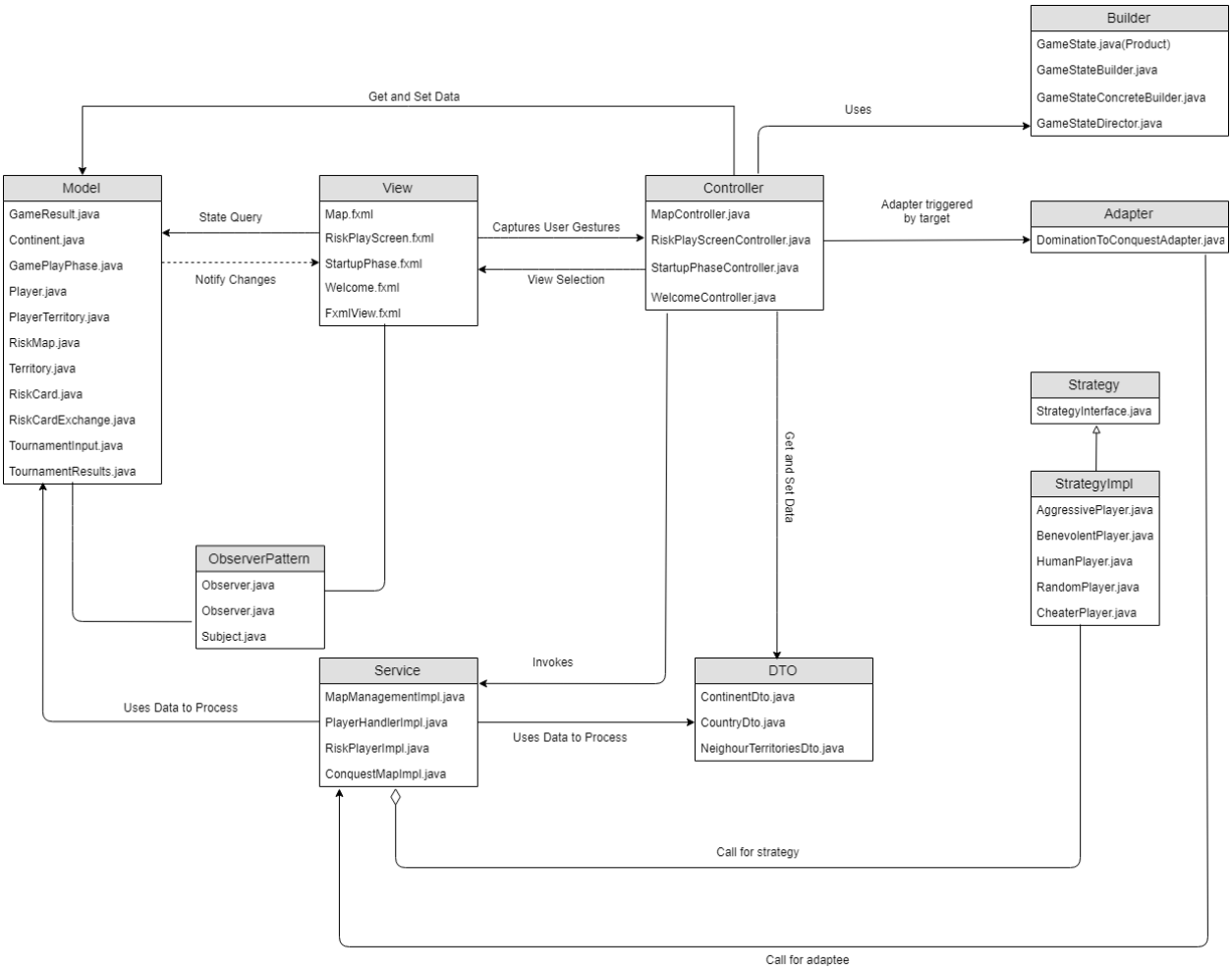
- User interaction is high.
- Provides separation of concerns.
- Provides support for rapid and parallel development.
- Provides multiple view of model.
- Modification in model does not affect entire architecture and views.

SCOPE

Scope of build three is as per requirements described in build statements.

1. Map editor
   - Creation of map
   - Edit existing map
   - Validate map
   - User-driven creation of map elements
2. Game play
   - Place army - armies are assigned to players in round-robin fashion and by claiming it by players.
   - Assign country-countries of a map is assigned randomly.
   - Information regarding game phase, player information and map information
3. Start up
   - Place army - armies are assigned to players in round-robin fashion and by claiming it by players.
   - Assign country - countries of a map is assigned randomly.
4. Reinforcement phase
   - Place reinforcement army –place reinforcement army on map.
   - Card exchange - creation of card making and exchanging it for armies based on cards.
5. Attack phase
   - Attack - Attack on opponent's territory adjacent to country chosen by player.
   - Move army – After country is conquered, number of armies are moved according to risk rules.
6. Fortification phase
   - Implementation of valid fortification move.

ARCHITECTURAL DESIGN



Module Description

Controller

File name	Description
MapController.java	Responsible for map management related operation such as add, remove of map attributes. It does save, validate map attributes.
RiskPlayScreenController.java	Responsible for operations to be performed when the game play begins. It does handle commands such as attack, reinforce, fortify, risk exchange.
StartupPhaseController.java	Responsible for invoking methods to perform operations such as add players and its strategy, place armies, populate countries, start tournament game.
WelcomeController.java	Main controller for invoking views.

Model

File name	Description
Continent.java	Model of continent data involves certain attributes such as continentIndex, continentName, continentValue and the territoryList.
GamePlayPhase.java	Model of game phase contains the getter and setter methods for playerList, gamephase and filename, riskCardList, winner and so on.
GameResult.java	Model of mapName, gameNo, winner.
Player.java	Model of player contains the player information such as playerId, playerName, number of armies owned my player, list of countries possessed by the player, strategy name of player, card list of player.
PlayerTerritory.java	Model of PlayerTerritory contains the attributes for continentName, territoryName and the total number of armies in a territory.
RiskMap.java	Model of map will have the map name, its respective continents and its status whether it is valid or not.
Territory.java	Model of territory data contains the demographic data of the territory such as the territoryIndex, territory's xAxis and yAxis, territoryName and continentIndex.
RiskCard.java	Model of risk card contains the cardIndex and the number of armies for the card.
RiskCardExchange.java	Model of risk card exchange contains the card exchange method where all the cards are

	exchanged in the fortification phase.
TournamentInput.java	Model for tournament attributes such as mapList, strategiesList, noOfGames, maxTurns.
TournamentResults.java	Model for result of the tournament.

View

File name	Description
Map.fxml	View responsible for performing operations of map and displaying it.
RiskPlayScreen.fxml	View responsible for playing the game and displaying it.
StartupPhase.fxml	View responsible for providing value to start the game.
Welcome.fxml	First view of risk domination game.
FxmlView.fxml	Configuration of other fxml pages are handled by this file.

DTO

File name	Description
ContinentDto.java	Responsible for manipulation of continent attributes.
CountryDto.java	Responsible for manipulation of countryattributes.
NeighbourTerritoriesDto.java	Responsible for manipulation of neighbor territories attributes.

SERVICE

File name	Description
MapManagementImpl.java	Responsible for implementing business logic of map management operation, reading operation of conquest map file.
PlayerHandlerImpl.java	Responsible for implementing business logic of handling a player.
RiskPlayImpl.java	Responsible for implementing business logic of three risk game phases, turns of players, exchanging of cards.
ConquestMapImpl.java	Responsible for implementing business logic of reading, saving, validating conquest map files.

LOGGING

File name	Description
ExceptionWriter.java	Errors occurred during execution of a program is handled in this file.

OBSERVERPATTERN

File name	Description
Observer.java	Abstract class that is responsible for defining operations to be used to notify registered observer objects.
Subject.java	Responsible for attaching, detaching observers to the client.

ADAPTER

File name	Description
DominationToConquestAdapter.java	Responsible for implementing return values for saving, reading, validating conquest map files.

BUILDER

File name	Description
GameState.java	The components of game state to be constructed
GameStateBuilder.java	Abstarct class for creating parts of game state object.
GameStateConcreteBuilder.java	Construct and assemble parts of game state object by extending GameStateBuilder class
GameStateDirector.java	Constructs an object using GameStateBuilder class.

STRATEGY

File name	Description
AggresivePlayer.java	Responsible for implementing business logic of three game phases of aggressive player.
BenevolentPlayer.java	Responsible for implementing business logic of three game phases of benevolent player.
CheaterPlayer.java	Responsible for implementing business logic of three game phases of cheater player.
HumanPlayer.java	Responsible for implementing business logic of three game phases of human player.
RandomPlayer.java	Responsible for implementing business logic of three game phases of random player.

CONSTANT

File name	Description
ComputerStrategy.java	Responsible for initializing strategy type as computer.
HumanStrategy.java	Responsible for initializing strategy type as human.
StrategyType.java	Responsible for initializing strategy type of players.
TournamentStrategy.java	Responsible for initializing valid tournament strategy according to risk rules.