Department of Engineering and Computer Science

Architectural Design

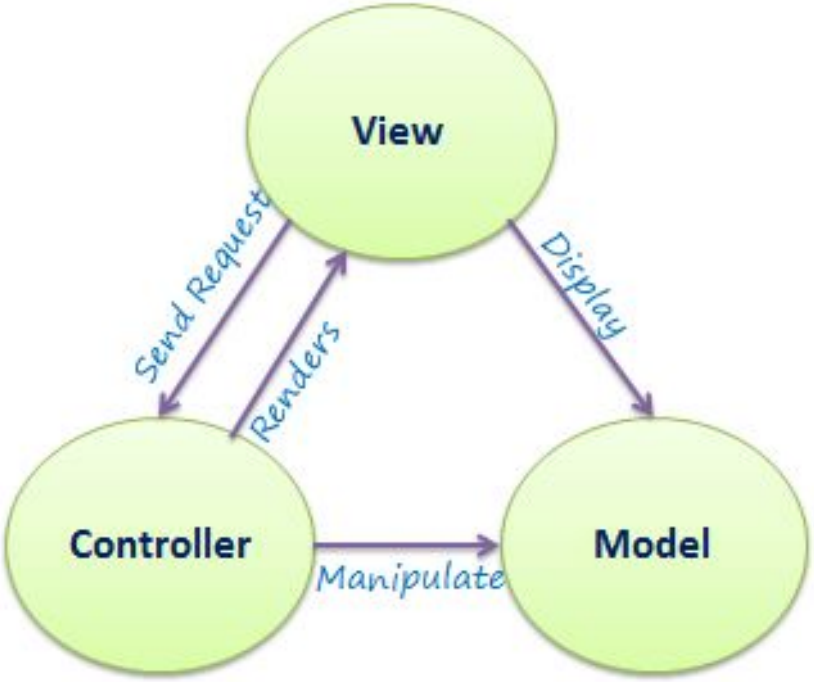
Advanced Programming Practices
Fall 2018

Team 3
Dmitry Kryukov
Ksenia Popova
Rodolfo Mateus Mota Miranda
Dinesh Pattapu
Nikitha Papani

Concordia University
Montreal, Quebec, Canada

October 2018

In our project we are using Model-View-Controller architecture. MVC - three-tier architectural model and it is widely used for many object oriented designs with user interaction.



There are three layers of architecture in this architectural model:

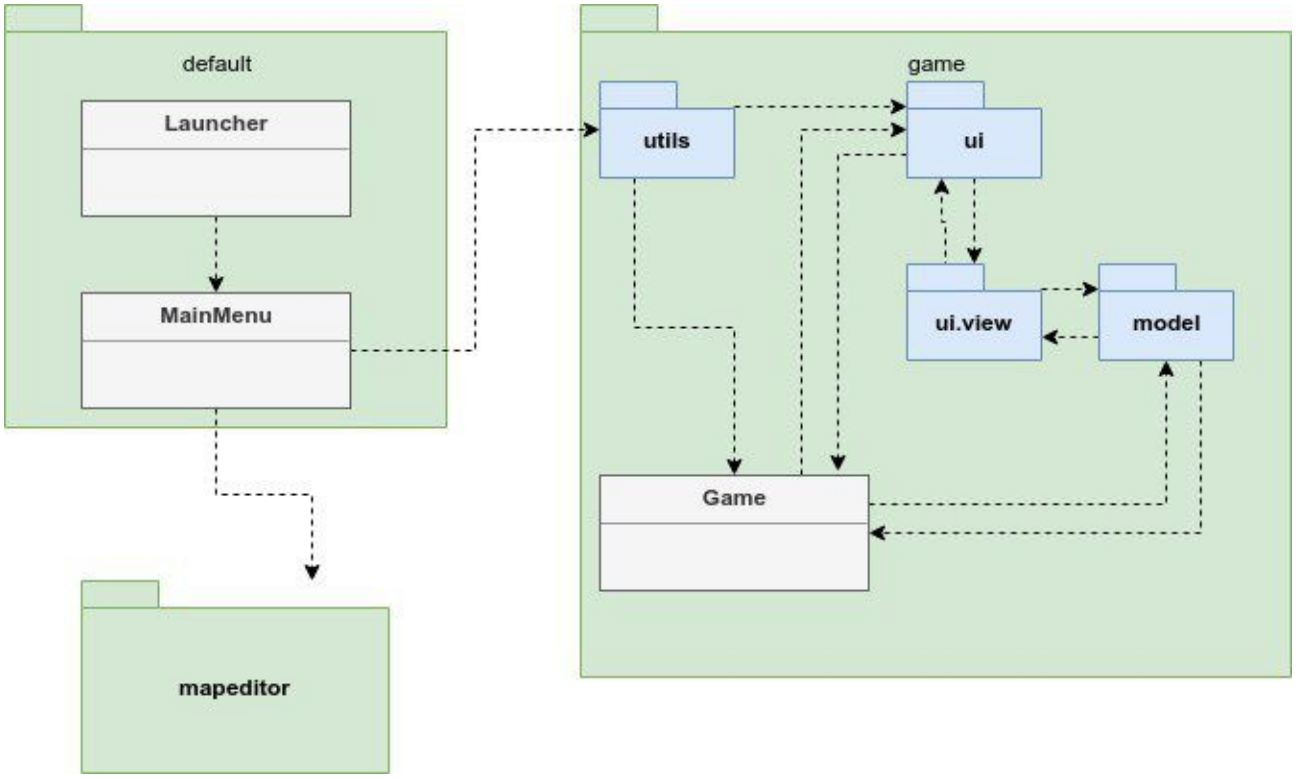
Model - manages the behavior of data (state) of the game. Model responds to the requests from user about the state and to the requests to change the state. In our project Model is responsible for the game data, such as map information (e.g. connections between countries, country's current owner), current game state (e.g. which player is the turn now, what phase of the turn is currently going), current amount of armies for each country, number of cards for each player. It provides the needed information to the View so it can be displayed to the user, and changes it if there is a request from the Controller to do so (e.g. change amount of armies). Model is represented by the classes defining the game entities.

View (game window) - renders the data from model to the user interface element for visualization and interaction. View requests the state information from the model and update the information displayed on the interface if needed. User's actions with the game are sent to the Controller that makes calls to the Model. For example, in the reinforcement phase user clicks on the country to add the army, then View sends the information about this interaction to the Controller that requests an update of the state of this country from the Model. View requests the update from the Model and changes the information on the

screen accordingly to the response.

Controller - renders the View when game starts (creates a game window). Then it accepts input from the View (user's action) and instracts Model how to act based on that input. Controller contains the main logic of the game flow. Controllers translates user's interactions into actions to the Model that it can perform. Also Controller can provide additional data from user's interactions on the View to the Model which can use this data to make changes in the correct way.

Package diagram for the game is represented below.



Default package contains the game launcher that can start a new game or open the map editor. User can choose what he wants to do in the main menu of the launcher.

Map Editor allows to edit the existing map and to create a new map. When creating a new map, user can specify all the necessary data for the map to be valid. When user tried to save the map after the map editing or creation, the map validation process starts. If the map doesn't meet some of the rules, user gets notified and this map cannot be saved until he solve the problem.

The game package contains all the main files of the game, such as Game.java (Controller), model folder with model representations (Player, Country, etc.) and views.

When a new game is started, the main controlling functionality is contained in the Game class (Controller). It generates the game window (View) and launches the game flow. View requests data from the Model and displays it. Controller (game) catches user's interactions with the View through the game process and request data changes accordingly to these interactions. View updates respectfully to these changes showing updated data to the user.