CONCORDIA UNIVERSITY
DEPARTMENT OF
COMPUTER SCIENCE AND SOFTWARE ENGINEERING

SOEN 6441: Advanced Programming Practices
Fall 2019

Project
Risk Domination Game
(Build 2)

Architectural Design Document

Team Name: Group_U_I

Name	**ID**
Van Tuan Tran	40124288
Benjamin Osei Asante	40080998
Tejinder Singh	40114377
Bharti Saini	40089008
Roger Madhu	40076461

Architectural design of the game

The game modules have been separated using both folders and separation of task. The figure-1 describes the top level view of the folder structure. We have followed the base Model View Controller (MVC) architecture. All concerned tasks have also been separated by folders and subfolders.

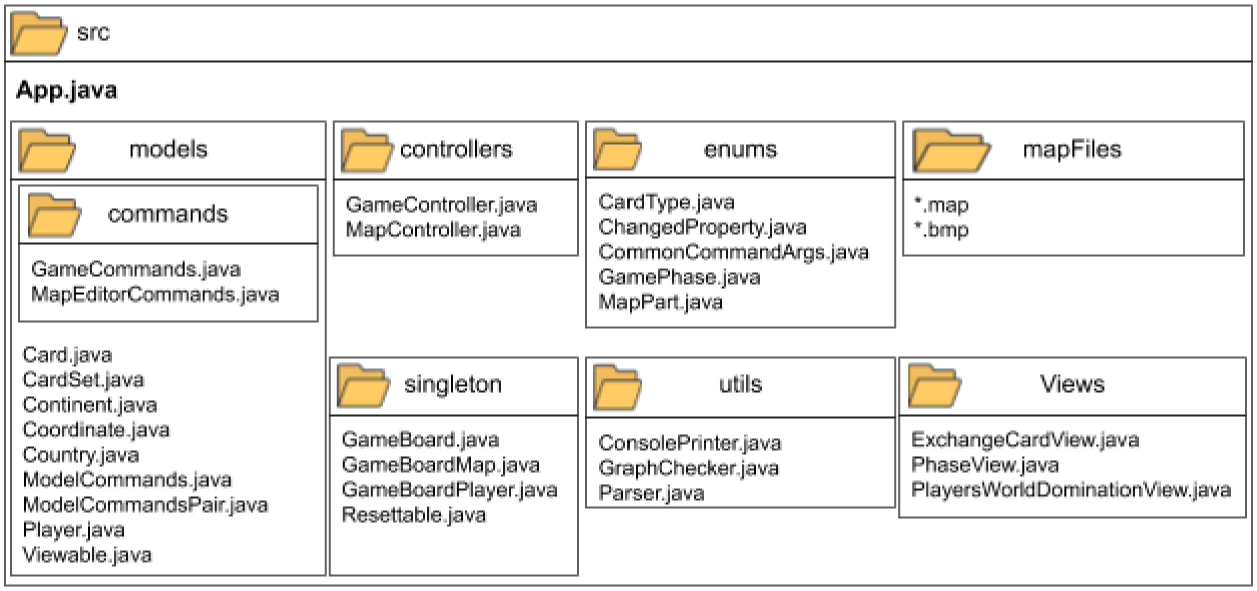

Figure 1

The top folder “src” contains only one file “App.java” which contains the *main* function to run the game. The “commands” folder is concerned with comparable game and map commands which can be edited from the files contained in this folder, the code base does not need to change for modification of commands. The “Controllers” folder is very important which controls and contains all major methods. “Enums” folder contains iterateable objects concerned with the game. “Map” folder has two different types of files “*.map” containing predefined map criterias and “.bmp” hold the images of the predefined map. The “Models” folder contains all the objects related to the game except the Singleton objects which is located in the “Singleton” folder. The “Models folder also holds a subfolder named “commands” which contains all the methods necessary for the game rules. The “utils” folder contain helper method such as the class “Parser” which helps perse command line instructions. All the files list is provided at the end of the document along with variable names and methods.

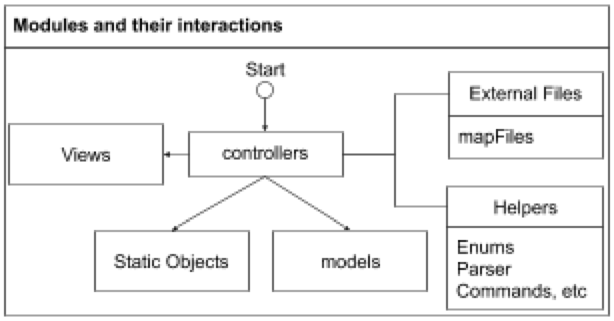

Figure 2

Figure 2 describes the main modules and their interaction. As we have followed MVC pattern as per build 2. The game starts by running the main method located in “app.java”, which in turn runs either the “MapController” or “GameController” object based on the user input from command prompt. The controller classes is concerned with the runtime business logic, it creates the necessary game objects using models. The helper classes are used for repetitive methods and for the ease of modification the command line commands are also located there. The helper classes contains ENUMS, parser and command line parameters. For ease of understanding the helper classes shown in figure-2 has been grouped together even though they have been separated in folders as per concern with the game. The controllers also use external files to load the map file, if no map file is loaded in the game then the controller creates a new map file as per user instruction. The views as been implemented using the observable and observer class as per specification within the scope of build 2.

The Unit Test classes and Test Suites is not shown in this document. As the test cases are more concerned on the programming side, the details has been concealed for this document.

Tools and Technologies Used

Tools and Technologies Description	**Description**
Java	Java is the base language used for development.
JUnit4	Junit4 for writing the test cases
JUnit5	Junit5 for writing the test cases and test suites
Eclipse	IDE for ease of code development
Visual Studio Code	IDE for ease of code development

List of files with each file, list of the methods and variables

```
==========================================================================================
GameController.java
==========================================================================================
----------------------- VARIABLES ------------------------------------------------
public static int MAX_INITIAL_ARMY_AMOUNT = 50;
public static final int MINIMUM_NUMBER_OF_ARMY_ON_COUNTRY = 1;
----------------------- METHODS --------------------------------------------------
public void handlePlayerAddAndRemoveCommand(String[] arg)
public void populateCountries()
public void initPlayersUnplacedArmies()
public void handlePlaceArmyCommand(String countryName)
public void handlePlaceAllCommand()
public void startRoundRobinPlayers()
public void turnToNextPlayer()
public void enterReinforcement()
public void handleReinforceCommand(String[] args)
public void handleFortifyCommand(String[] args)

private void placeArmy(Country country, Player player)
private Player getCurrentPlayer()
private Player getCurrentPlayer(boolean isShowMessage)
private int getArmiesFromAllConqueredCountries(Player currentPlayer)
private int getArmiesFromConqueredContinents(Player currentPlayer)
private void calculateReinforcementArmies(Player currentPlayer)

==========================================================================================
MapController.java
==========================================================================================
----------------------- VARIABLES ------------------------------------------------
private static final int MINIMUM_AMOUNT_OF_COUNTRIES
----------------------- METHODS --------------------------------------------------
public void addContinent(String continentName, String continentValue, int... order)
public void addCountry(String countryName, String continentName)
public void addNeighbor(String countryName, String neighborCountryName)
public void createNewCountry(String countryName, Continent continent)
public void editContinent(String[] args)
public void editCountry(String[] args)
public void editMap(String fileName)
public void editNeighbor(String[] args)
public Continent getContinentFromName(String continentName)
public boolean isContinentExisted(String continentName)
public boolean isCountryExisted(String countryName)
public void loadMap(String fileName)
public void removeContinent(String continentName)
public void removeCountry(String countryName)
public void removeNeighbor(String countryName, String neighborCountryName)
public void resetMap()
public void saveMap(String fileName) throws IOException
public void showMap()
public boolean validateMap()

private void addBorders(int countryOrder, int... borderWithCountries)
private void addCountryFromMapFile(String name, int continentOrder, Coordinate coordinate)
private void increaseBorder(int newBorderSize)
private ArrayList<Continent> getEmptyContinents()
private ArrayList<Country> getIsolatedCountries()
private boolean isNotEnoughCountries(int minimumNumberOfCountries)
private boolean isStillInCurrentDataBlock(int currentLineIndex, List<String> lines)
private int loadBordersFromFile(int currentLineIndex, List<String> lines)
private int loadContinentsFromFile(int currentLineIndex, List<String> lines)
private int loadCountriesFromFile(int currentLineIndex, List<String> lines)
private void removeBorder(int borderLocation)
private void writeBordersToFile(FileWriter writer) throws IOException
private void writeCountriesToFile(FileWriter writer) throws IOException
private void writeContinentsToFile(FileWriter writer) throws IOException
private void updateCountryContinent(Country country, Continent continent)

==========================================================================================
Continent.java
==========================================================================================
------------------------ VARIABLES --------------------------------------------------
private String name;
private int army;
private ArrayList<Country> countries = new ArrayList<Country>();
private int order;
------------------------ METHODS --------------------------------------------------
public Continent(String name, int army, int... order)
public int getOrder()
public void setOrder(int order)
public int getArmy()
public void setArmy(int army)
public ArrayList<Country> getCountries()
public void setCountries(ArrayList<Country> countries)
public String getName()
public void setName(String name)
public Player getConquerer()
public void view(int indent)
public void viewWithoutCountry()

==========================================================================================
Coordinate.java
==========================================================================================
------------------------ METHODS --------------------------------------------------
public Coordinate(int x, int y)
public int getY()
public void setY(int y)
public int getX()
public void setX(int x)
```

==
Country.java
==
------------------------ VARIABLES ---
private Coordinate coordinate;
private int armyAmount;
private String name;
private Continent continent;
private Player conquerer;
------------------------ METHODS ---
public Country(String name, Coordinate coordinate, Continent continent)
public Player getConquerer()
public void setConquerer(Player conquerer)
public Continent getContinent()
public void setContinent(Continent continent)
public String getName()
public void setName(String name)
public int getArmyAmount()
public void setArmyAmount(int armyAmount)
public int getOrder()
public Coordinate getCoordinate()
public void setCoordinate(Coordinate coordinate)
public ArrayList<Country> getNeighbors()
public boolean isConquered()
public void receiveArmiesFromUnPlacedArmies(int amount)
public void increaseArmies(int amount)
public void decreaseArmies(int amount)
public void moveArmies(Country toCountry, int armiesToMove)
public void view(int indent)
public void viewWithoutNeighbors(int indent)

==
Player.java
==
------------------------ VARIABLES ---
private String name;
private int armies;
private int unplacedArmies;
private boolean isPlaying = false;
private boolean isLost = false;
private Player nextPlayer;
private Player previousPlayer;
------------------------ METHODS ---
public Player(String name)
public Player getPreviousPlayer()
public void setPreviousPlayer(Player previousPlayer)
public Player getNextPlayer()
public void setNextPlayer(Player nextPlayer)
public String getName()
public boolean isLost()

```
public void setLost(boolean isLost)
public ArrayList<Country> getConqueredCountries()
public boolean isPlaying()
public void setPlaying(boolean isPlaying)
public int getUnplacedArmies()
public void setUnplacedArmies(int unplacedArmies)
public int getArmies()
public void setArmies(int armies)
public void setName(String name)

==========================================================================================
Viewable.java
==========================================================================================
------------------------ METHODS -----------------------------------------------------
public void view(int indent)
public default void view()
public default void printIndent(int indent)

==========================================================================================
GameMap.java
==========================================================================================
------------------------ VARIABLES --------------------------------------------------
private static final GameMap instance = new GameMap();
private String mapName;
private ArrayList<Continent> continents = new ArrayList<Continent>();
private ArrayList<Country> countries = new ArrayList<Country>();
private ArrayList<Player> players = new ArrayList<Player>();
private int[][] borders;
------------------------ METHODS -----------------------------------------------------
private GameMap()

public ArrayList<Player> getPlayers()
public int[][] getBorders()
public void setBorders(int[][] graph)
public ArrayList<Continent> getContinents()
public void setContinents(ArrayList<Continent> continents)
public String getMapName()
public void setMapName(String mapName)
public ArrayList<Country> getCountries()
public void setCountries(ArrayList<Country> countries)
public static GameMap getInstance()
public void reset()
public void showContinents()
public void showCountries()
public Country getCountryFromName(String countryName)
public Player getPlayerFromName(String name)
public void addPlayer(String name)
public void removePlayer(String name)
public boolean isNeighboringCountries(String countryName, String neighborCountryName)
public boolean isNeighboringCountries(Country country, Country neighborCountry)
```

```
=====================================================================================
Parser.java
=====================================================================================
------------------------ METHODS ----------------------------------------------------
public static int parseWithDefault(String number, int defaultVal)


=====================================================================================
App.java
=====================================================================================
------------------------ METHODS ----------------------------------------------------
private App()
public static void main(String[] args)
public static void jumpToCommand(String[] args)
public static void runFromBegining()
```