

**RISK GAME (BUILD-1)**

**ARCHITECTURAL DESIGN**

**Advance Programming Language**

**SOEN 6441**

**Fall-2019**

**Group NO. 13**

SHUBHAM RANADIVE (40083991)

PARTH PATEL (40081116)

MEHUL PRAJAPATI (40076930)

KOMAL PANCHAL (40130791)

MARYAM GIAHI (40016260)

MAHMOUDREZA ENTEZAMI (40058782)

Contents

[*Introduction* 3](#_Toc22081777)

[**1. Scope** 3](#_Toc22081778)

[**1.1Map Editor:** 3](#_Toc22081779)

[**1.2Game Play:** 3](#_Toc22081780)

[*2. Architecture Design* 4](#_Toc22081781)

[*3.Modules Description* 5](#_Toc22081782)

[*3.1. Config* 6](#_Toc22081783)

[*3.2. Entity* 6](#_Toc22081784)

[*3.3. Mapparser* 6](#_Toc22081785)

[*3.4 Main* 7](#_Toc22081786)

[*4.Test Cases (Junit) Description* 7](#_Toc22081787)

[*4.1 main Test* 7](#_Toc22081788)

[*4.3 Mapparser Test* 7](#_Toc22081789)

[*5. Tools and API* 8](#_Toc22081790)

[*6. some part of Code* 9](#_Toc22081791)

[*7. References:* 16](#_Toc22081792)

## 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:

## 1.1Map 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.

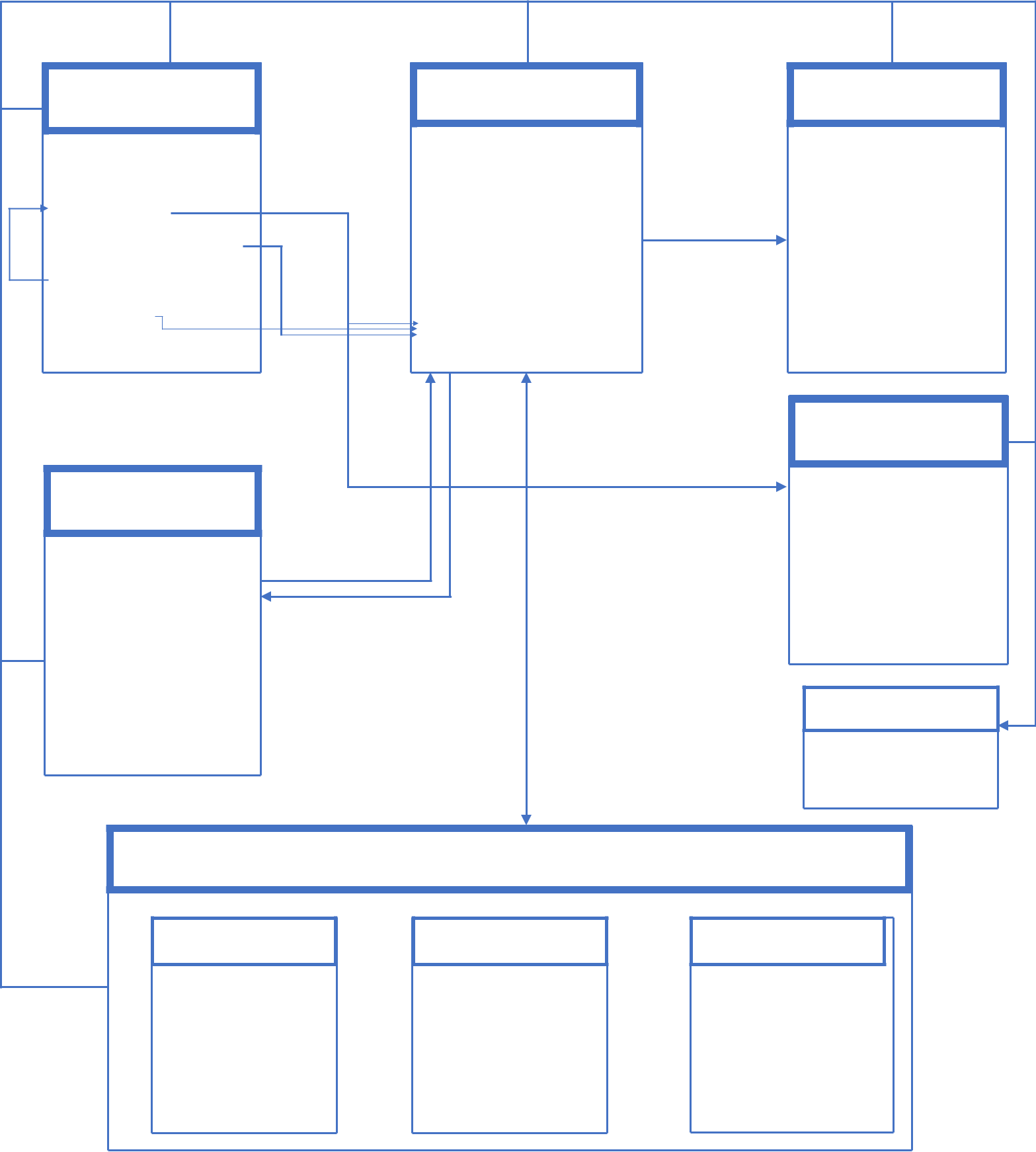
## 1.2Game Play:

* + 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

****

Handles an Exceptions if occurs in any file

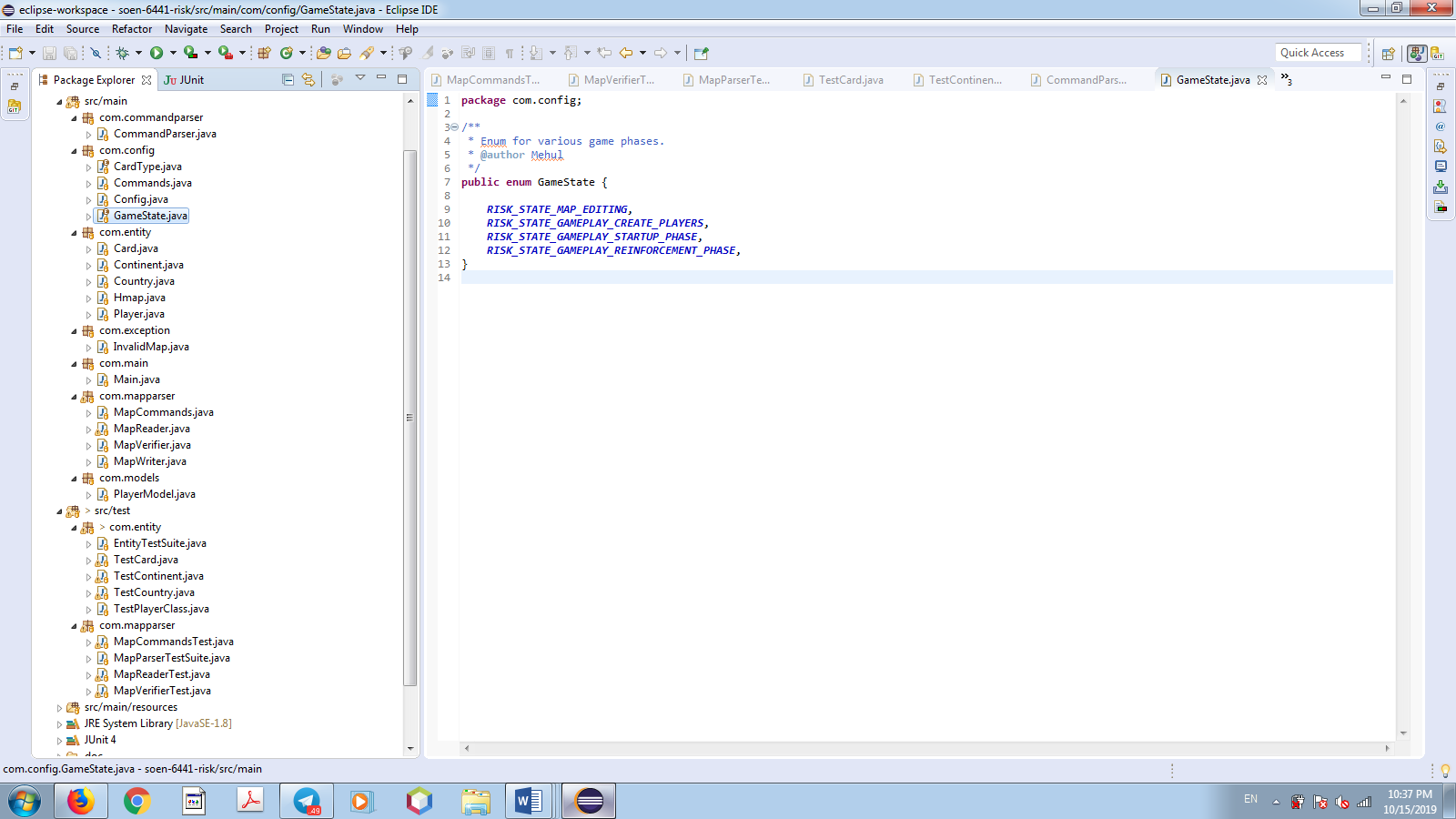


|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Model |  |  | Controller |  | View |
|  |  |  |  | CardExchangeController.java |  | Game Play |
|  | GameUIStateModel.java |  |  | DiceController.java |  |
|  |  |  |  | Map Editor |
|  | PlayerModel.java |  |  | MainController.java |  |
|  |  |  | MapRedactorController.java | Update the | Splash Screen |
| Notify | WorldDominationModel.java |  |  | views |
|  |  | MapRedactorSplashController |  |
|  |  |  | Map Editor Splash |
|  |  |  |  |  |
|  | DiceModel.java |  |  | .java |  |
|  | CardModel.java |  | Notify | PlayerStrategyChooser.java |  | Dice View |
|  |  | PlayGameController.java |  |
|  | TournamentModel.java | On |  |  | Card View |
|  |  | TournamentController.java |  |
|  | Request |  |  |
|  |  |  |  |  |  | Strategy |
|  | Entity |  |  |  | Calls | Human.java |
|  |  |  |  |  |
|  |  |  |  |  |  | Cheater.java |
|  | Continent.java |  |  |  |  | Random.java |
|  | Returns values of attribute | | |  | IStrategy.java |
|  |  |  |  | functions |  |
|  | Country.java | Sets values of attribute | |  | Aggressive.java |
|  |  |  |  |  | Benevolent.java |
|  | Map.java |  |  | common |  |
|  |  |  |  |  |
|  | Player.java |  |  |  |  |
|  |  |  | set of |  | Exceptions |
|  |  |  |  |  |
|  | Card.java |  |  | Utilizes |  | InvalidGameAction.java |
|  |  |  |  |  |
|  |  |  |  |  |  | Invalidmap.java |
|  |  |  | Set of Common functions | |  |  |
|  | Configuration | |  | Map utilities | Game utilities | |
|  | Configuration.java | |  | MapVerifier.java |  |  |
|  |  | Mapwriter.java |  |  |
|  |  |  |  |  |  |
|  | PlayerStrategy.java  CardKind.java | |  | MapCommand.java | GameUtilities.java | |

## 3.Modules Description

**Detailed Structure and Explanation**

The overview of project packages is as shown in Screenshot 1.



****

## 3.1. Config

|  |  |
| --- | --- |
| File\_name | Description |
| CardType.java | It is a class for cardtype configuration. |
| Commands.java | It is a class for Command configuration. |
| Config.java | It is a class for army configuration . |
| GameState.java | It is a class for GameState configuration. . |

## 3.2. Entity

|  |  |
| --- | --- |
| File\_name | Description |
| Hmap.java | It contains all the information of the Map and a list of the continents. |
| Card.java | It contains all the information regarding to Card. |
| 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. |

## 3.3. Mapparser

|  |  |
| --- | --- |
| *File\_name* | *Description* |
| MapCommand.java | Contains all the common method of the map like: saving map |
| [MapReader.java](https://bitbucket.org/niravjdn/risksoen6441/src/master/src/main/java/com/risk6441/maputils/MapReader.java) | It reads the map file format and parsing in to Map object and checks for the validity of data of the map file. |
| [MapWriter.java](https://bitbucket.org/niravjdn/risksoen6441/src/master/src/main/java/com/risk6441/maputils/MapWriter.java) | It is responsible for writing the Map object to the file. |
| MapVerifier.java | object, opening a dialogue box etc. |

## 3.4 Main

|  |  |
| --- | --- |
| File\_name | Description |
| Main.java | Entry point for the application |

## 4.Test Cases (Junit) Description

## 4.1 main Test

|  |  |
| --- | --- |
| File\_name | Description |
| MainTestSuite.java | This is a test class for running all test suits.  ( MapUtilitiesTestSuite.class, GameUtilitiesTestSuite.class,  ModelsTestSuite.class, StrategyTestSuite.class) |

4.2 Entity Test

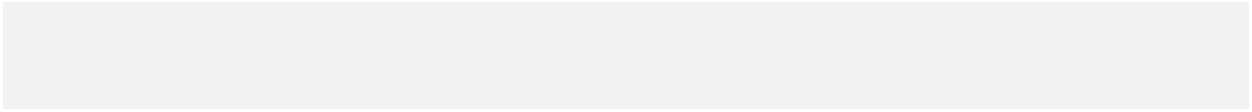
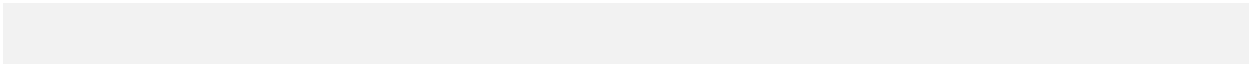
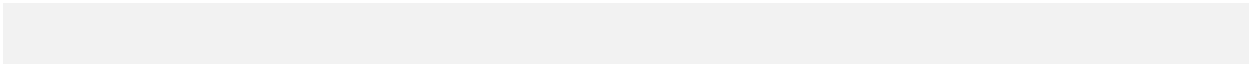
|  |  |
| --- | --- |
| *File\_name* | *Description* |
| TestCard.java | This is a test for card class. |
| TestContinet.java | This is a test for continent class. |
| TestCountry.java | This is a test for country class. |
| TestPlayer.java | This is a test for player class. |
| EntityTestSuite.java | This is a test class for running all test suits in Entity  i.e. (TestCard.class, TestContinent.class,  TestCountry.class, TestPlayerClass.class) |

## 4.3 Mapparser Test

|  |  |
| --- | --- |
| *File\_name* | *Description* |
| MapcommadTest.java | This is a test class for Map operations |
| Mapreader.java | This is a test class for Map-reader. |
| Mapwriter.java | This is a test class for Map writer. |
| Mapverifier.java | This is a test class for Map verifier. |
| EntityTestSuite.java | This is a test class for running all test suits in Mapparser i.e. (MapcommadTest.class, MapreaderTest.class,  MapwriterTest.class, MapverifierTest.class) |

## 5. Tools and API

|  |  |
| --- | --- |
| *Tools* | *Description* |
| Eclipse | IDE for the game development |
| Scene Builder | It is an open source JavaFX system used for UI design and gives a skeleton of the events to be implemented in controller. |
| Source Tree | Library to control the UI component |
| JavaFx | This is a test class for Map verifier. |
| Source Tree | It is Git code management System which gives one place to plan projects, collaborate on code test and deploy. |
| Junit4 | For writing test class |
| Maven | Maven as a build automation tool to manage all project dependencies. |



## 6. some part of Code

**package** com.mapparser;

**import** java.util.ArrayList;

**import** java.util.List;

**import** com.entity.Continent;

**import** com.entity.Country;

**import** com.entity.Hmap;

**import** com.exception.InvalidMap;

**public** **class** MapCommands {

**private** **static** **int** *countryIdx* = 1;

/\*\*

\* Removes continent from the map.

\* **@param** map Current map object.

\* **@param** continentName Name of the continent.

\* **@return** true if continent got removed successfully, otherwise false

\*/

**public** **static** **boolean** removeContinent(Hmap map, String continentName) {

Continent continent = **new** Continent();

continent.setName(continentName);

**for** (Continent c: map.getContinents()) {

**if** (c.getName().equalsIgnoreCase(continentName)) {

map.getContinents().remove(c);

map.getContinentMap().remove(continentName);

System.***out***.println("Successfully removed continent: " + continentName + " from map");

**return** **true**;

}

}

System.***out***.println("Continent: " + continentName + " does not exist in map");

**return** **false**;

}

/\*\*

\* Adds continent to the map with details like control value etc.

\* **@param** map Current map object.

\* **@param** name Name of the continent.

\* **@param** ctrlValue Control value of the continent.

\* **@param** color Color of the continent.

\* **@return** true if continent gets added to map, false otherwise.

\*/

**public** **static** **boolean** addContinent(Hmap map, String name, String ctrlValue, String color) {

Continent continent = **new** Continent();

continent.setName(name);

continent.setValue(Integer.*parseInt*(ctrlValue));

continent.setColor(color);

**if** (map.getContinents().contains(continent)) {

System.***out***.println("The Continent with name " + name + " already exist.");

**return** **false**;

}

System.***out***.println("The continent: " + name + " added successfully");

map.getContinents().add(continent);

map.getContinentMap().put(name, continent);

**return** **true**;

}

/\*\*

\* This method checks whether the continent name is present or not.

\* **@param** listContinents list of all continents

\* **@param** name name of the continents to be updated

\* **@return** true if list does not contain other continents with same name

\*/

**public** **static** **boolean** containsContinentName(**final** List<Continent> listContinents, **final** String name){

**return** listContinents.stream().filter(x -> x.getName().equals(name)).findFirst().isPresent();

}

/\*\*

\* This method updates the continent details if the user selects the continent.

\* **@param** continent The continent whose details must be updated.

\* **@param** map map object {@link Hmap}

\* **@param** name name of the continent to be updated

\* **@param** ctrlValue The control value of the continent.

\* **@return** The current continent.

\* **@throws** InvalidMap InvalidMapException if any error occurs

\*/

**public** **static** Continent updateContinent(Continent continent, Hmap map, String name, String ctrlValue) **throws** InvalidMap {

**if** (!continent.getName().equals(name)) {

**if** (*containsContinentName*(map.getContinents(), name)) {

**throw** **new** InvalidMap("The Continent with name " + name + " already exist.");

}

continent.setName(name);

}

continent.setValue(Integer.*parseInt*(ctrlValue));

**return** continent;

}

/\*\*

\* This checks whether the Country name is there or not.

\* **@param** list list of all Countries

\* **@param** name name of the Country to be checked

\* **@return** true if list does not contain other Country with same name

\*/

**public** **static** **boolean** containsCountryName(**final** ArrayList<Country> list, **final** String name){

**return** list.stream().filter(z -> z.getName().equals(name)).findFirst().isPresent();

}

/\*\*

\* Removes country from the map.

\* **@param** map Current map object.

\* **@param** name Name of the Country.

\* **@return** true if Country gets removed from map, false otherwise.

\*/

**public** **static** **boolean** removeCountry(Hmap map, String name) {

**for** (Continent continent: map.getContinents()) {

**for** (Country country: continent.getCountries()) {

**if** (country.getName().equalsIgnoreCase(name)) {

continent.getCountries().remove(country);

continent.getCountryMap().remove(name);

map.getCountriesIdxMap().remove(name);

**for** (Country adjCountry: country.getAdjacentCountries())

adjCountry.getAdjacentCountries().remove(country);

System.***out***.println("Successfully removed country: " + name + " from map");

**return** **true**;

}

}

}

System.***out***.println("Country: " + name + " does not belong to any continent");

**return** **false**;

}

/\*\*

\* Adds country to the map and the continent with its respective details.

\* **@param** map Current map object.

\* **@param** name Name of the Country.

\* **@param** continentName Continent to which the country belongs to.

\* **@return** true if Country gets added to map, false otherwise.

\*/

**public** **static** **boolean** addCountry(Hmap map, String name, String continentName) {

Country country = **new** Country();

Continent belongToContinent = **null**;

// check if country with same name exist or not

**for** (Continent c : map.getContinents()) {

**if** (c.getCountries().contains(country)) {

System.***out***.println("Country with same name: " + name +

" already exist in continent: " + c.getName());

**return** **false**;

}

**if** (c.getName().equalsIgnoreCase(continentName))

belongToContinent = c;

}

**if** (**null** == belongToContinent) {

System.***out***.println("Belong to continent: " + continentName + " does not exist in map");

**return** **false**;

}

country.setBelongToContinent(belongToContinent);

country.setName(name);

map.getCountriesIdxMap().put(name, *countryIdx*++);

belongToContinent.getCountries().add(country);

belongToContinent.getCountryMap().put(name, country);

System.***out***.println("Country: " + name + " added to the continent: "

+ continentName + " successfully");

**return** **true**;

}

/\*\*

\* This method updates the continent details when the user selects the country.

\* **@param** country The country whose values must be updated.

\* **@param** map Map Object {@link Hmap}

\* **@param** name name for the Country to be updated - new name for the Country

\* **@param** xCo X-Co-ordinate of the Country.

\* **@param** yCo Y-Co-ordinate of the Country.

\* **@param** adjCoun The adjacent Countries list.

\* **@return** The object to the newly updated Country.

\* **@throws** InvalidMap InvalidHmap if any error occurs

\*/

**public** **static** Country updateCountry(Country country, Hmap map, String name,String xCo, String yCo,

Country adjCoun) **throws** InvalidMap {

country.setxCoordinate(Integer.*parseInt*(xCo));

country.setyCoordinate(Integer.*parseInt*(yCo));

**if** (!country.getName().equals(name)) {

ArrayList<Country> listAllCoun = **new** ArrayList<Country>();

**for** (Continent cont : map.getContinents()) {

listAllCoun.addAll(cont.getCountries());

}

**if** (*containsCountryName*(listAllCoun, name)) {

**throw** **new** InvalidMap("The Country with name "+name+" already exist.");

}

country.setName(name);

}

**if** (adjCoun != **null**) {

**if** (!adjCoun.getAdjacentCountries().contains(country)) {

adjCoun.getAdjacentCountries().add(country);

}

**if** (!country.getAdjacentCountries().contains(adjCoun)) {

country.getAdjacentCountries().add(adjCoun);

}

}

**return** country;

}

/\*\*

\* This method removes the country to the corresponding continent.

\* **@param** map Current map object.

\* **@param** countryName Name of the Country.

\* **@param** nbrCountryName Name of the neighbor Country.

\* **@return** true if neighbor Country gets removed from map, false otherwise.

\*/

**public** **static** **boolean** removeNeighborCountry(Hmap map, String countryName, String nbrCountryName) {

**if** (!map.getCountriesIdxMap().containsKey(countryName)) {

System.***out***.println("The country: " + countryName + " does not exist in map");

**return** **false**;

}

**if** (!map.getCountriesIdxMap().containsKey(nbrCountryName)) {

System.***out***.println("The neighbor country: " + nbrCountryName + " does not exist in map");

**return** **false**;

}

**for** (Continent c: map.getContinents()) {

**for** (Country country: c.getCountries()) {

**if** (country.getName().equalsIgnoreCase(countryName)) {

country.getAdjacentCountries().remove(c.getCountryMap().get(nbrCountryName));

country.getNeighborCountries().remove(nbrCountryName);

c.getCountryMap().get(nbrCountryName).getAdjacentCountries().remove(country);

c.getCountryMap().get(nbrCountryName).getNeighborCountries().remove(countryName);

System.***out***.println("The neighbor country: " + nbrCountryName +

" removed from adjacent country: " + countryName);

**return** **true**;

}

}

}

**return** **false**;

}

/\*\*

\* This method adds the country to the corresponding continent.

\* **@param** map Current map object.

\* **@param** countryName Name of the Country.

\* **@param** nbrCountryName Name of the neighbor Country.

\* **@return** true if neighbor Country gets added to map, false otherwise.

\*/

**public** **static** **boolean** addNeighborCountry(Hmap map, String countryName, String nbrCountryName) {

**if** (!map.getCountriesIdxMap().containsKey(countryName)) {

System.***out***.println("The country: " + countryName + " does not exist in map");

**return** **false**;

}

**if** (!map.getCountriesIdxMap().containsKey(nbrCountryName)) {

System.***out***.println("The neighbor country: " + nbrCountryName + " does not exist in map");

**return** **false**;

}

**for** (Continent c: map.getContinents()) {

**for** (Country country: c.getCountries()) {

**if** (country.getName().equalsIgnoreCase(countryName)) {

country.getAdjacentCountries().add(c.getCountryMap().get(nbrCountryName));

country.getNeighborCountries().add(nbrCountryName);

c.getCountryMap().get(nbrCountryName).getAdjacentCountries().add(country);

c.getCountryMap().get(nbrCountryName).getNeighborCountries().add(countryName);

System.***out***.println("The neighbor country: " + nbrCountryName +

" added as an adjacent country to: " + countryName);

**return** **true**;

}

}

}

System.***out***.println("Failed to add the neighbor country: " + nbrCountryName);

**return** **false**;

}

/\*\*

\* This method adds the country to the corresponding continent.

\* **@param** continent continent object which will be assigned Countries

\* **@param** country The country which is added to the continent.

\* **@return** the Object to the newly updated continent.

\*/

**public** **static** Continent mapCountryToContinent(Continent continent, Country country) {

**try** {

continent.getCountries().add(country);

} **catch**(Exception e) {

ArrayList<Country> list = **new** ArrayList<>();

list.add(country);

continent.setCountries(list);

}

**return** continent;

}

/\*\*

\* This method shows the map.

\* **@param** map current map object

\*/

**public** **static** **void** mapEditorShowmap(Hmap map) {

**for** (Continent c : map.getContinents()) {

System.***out***.println("--------------------------------");

System.***out***.println("Continent: " + c.getName() + " having following countries");

**for** (Country con : c.getCountries()) {

System.***out***.print(con.getName() + ": ");

List<String> adjCountries = con.getNeighborCountries();

**for** (**int** i = 0; i < adjCountries.size(); i++) {

System.***out***.print(adjCountries.get(i));

**if** (i != adjCountries.size() - 1)

System.***out***.print(", ");

}

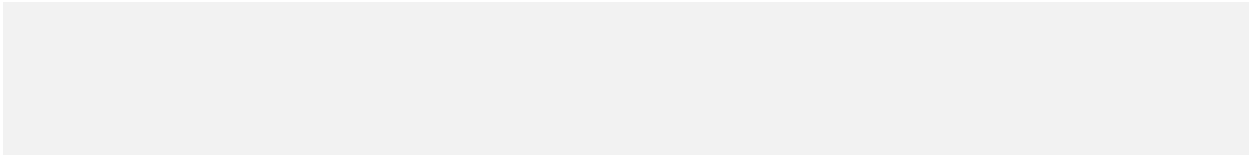
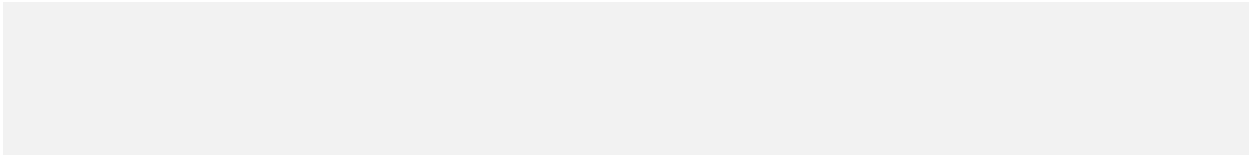
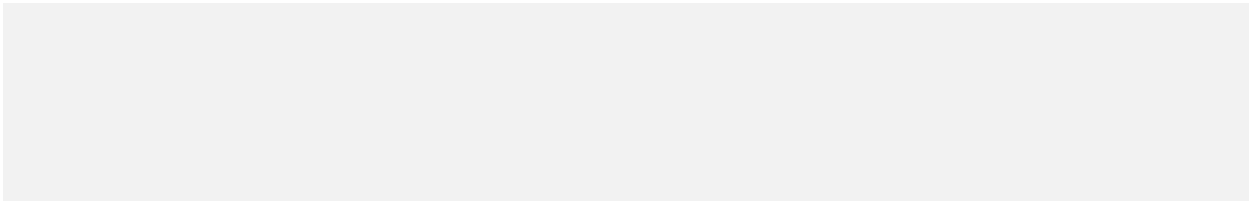
System.***out***.println();

}

}

System.***out***.println("--------------------------------");

}

************

## 7. References:

1. Rules Followed : <https://www.wikihow.com/Play-Risk>
2. <https://sourcemaking.com/refactoring/refactorings>
3. <https://www.sourcetreeapp.com>