Concordia University

Department of Computer Science and Software Engineering

Advanced Programming Practices

SOEN 6441 - Fall 2023

Group Name: W4
Submitted to: Prof. Amin Ranj Bar

Members:
Sneh Satishkumar Patel - 40264053
Romit Manishkumar Patel - 40273551
Yash Subhashbhai Chhelaiya - 40257682
Rushi Hasubhai Donga - 40269583
Abdul Sameer Jani Syed - 40272408
Mayank Nilkanth Parmar - 40269385

UNIVERSITÉ
Concordia
UNIVERSITY

Risk Computer Game

A developed program which is compatible with the rules and map files and the command-line play of the “Warzone” version of Risk, which can be found at: https://www.warzone.com/.

About Risk Game

- A Warzone game setup consists of a connected graph map representing a world map, where each node is a country and each edge represents adjacency between countries.
- Two or more players can play by placing armies on countries they own, from which they can attack adjacent countries to conquer them.
- The objective of the game is to conquer all countries on the map.

Goal of Build 1

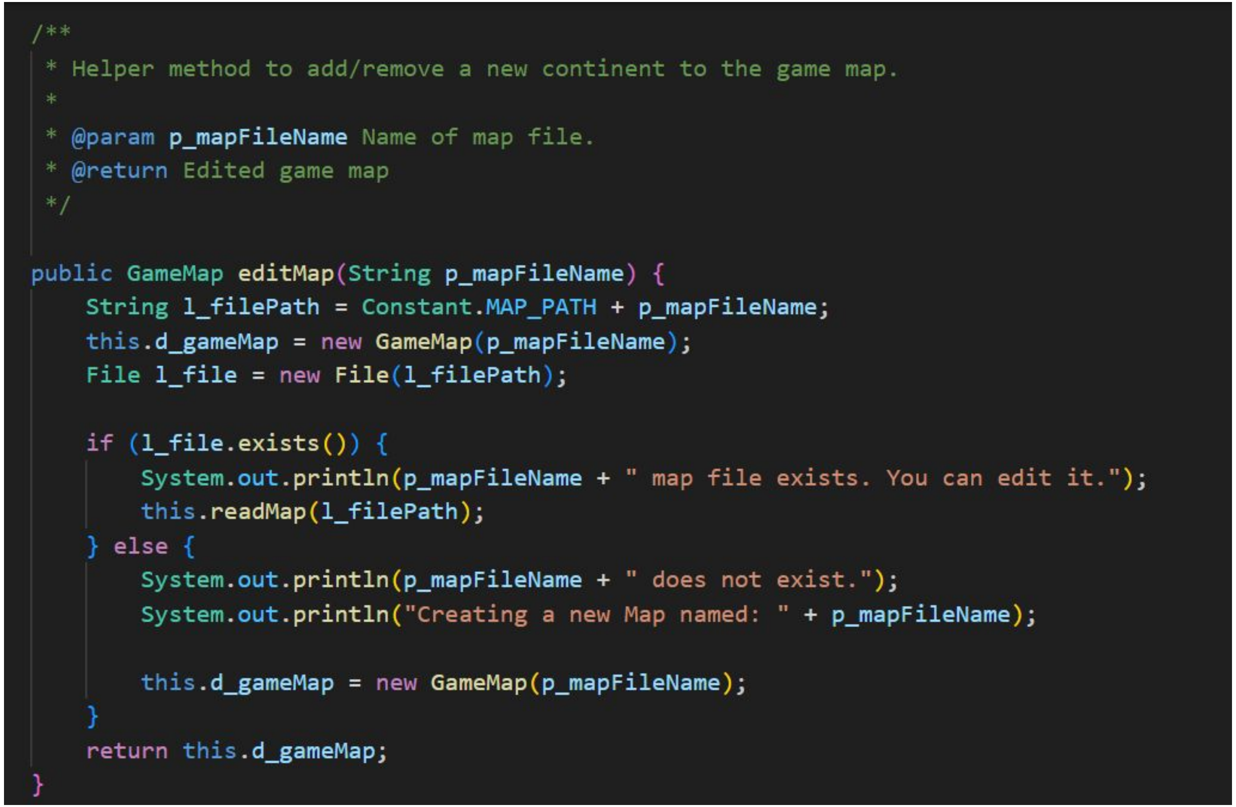
- RiskGame Build 1's goal is to build the basic structure and key functionality of the Java-based Risk game while keeping to the rules and map files of the "Warzone" edition of Risk.
- This initial build specifies the project's folder structure, coding rules, and design principles.
- It also covers game-critical elements like as map development, player management, reinforcement, and deployment.
- Build 1 serves as the foundation for future features, unit tests, and continuous integration/continuous deployment (CI/CD) pipelines.

CODING STANDARDS

Code Layout

- **Indentation:**
  - Used 4 spaces for indentation.
  - The body of loop and conditional statements are indented in relation to their starting lines.
  - Body of a function is appropriately indented beneath its header.
- **Line Spacing:**
  - Blank lines serve the purpose of creating separation within code.
  - They are inserted between class declarations, methods, and significant segments within complex functions to enhance code readability.
- **Positioning of curly braces:**
  - The position of curly braces involves placing an opening curly brace immediately after the statement preceding it, thereby reducing the length of the code.
- **Spacing around operators and operands:**
  - Inserting spaces before and after operators to enhance the legibility of code.

MapHelper.java



```java
/**
 * Helper method to add/remove a new continent to the game map.
 *
 * @param p_mapFileName Name of map file.
 * @return Edited game map
 */

public GameMap editMap(String p_mapFileName) {
    String l_filePath = Constant.MAP_PATH + p_mapFileName;
    this.d_gameMap = new GameMap(p_mapFileName);
    File l_file = new File(l_filePath);

    if (l_file.exists()) {
        System.out.println(p_mapFileName + " map file exists. You can edit it.");
        this.readMap(l_filePath);
    } else {
        System.out.println(p_mapFileName + " does not exist.");
        System.out.println("Creating a new Map named: " + p_mapFileName);

        this.d_gameMap = new GameMap(p_mapFileName);
    }
    return this.d_gameMap;
}
```

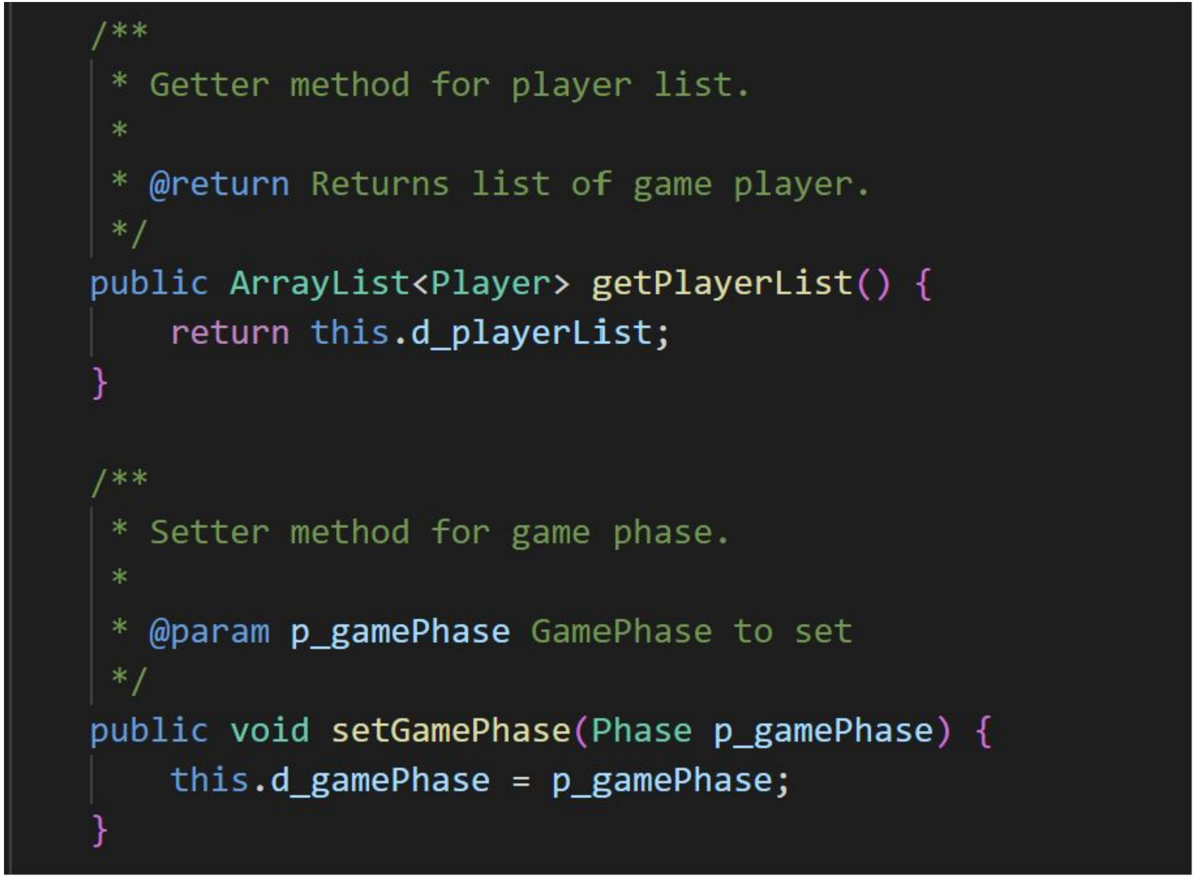
Guidelines for Naming

- **Classes:**
  - Class names follow the convention of being written in UpperCamelCase, also known as Pascal Case.
  - Example : class MyClassName { }
- **Method Parameters, Member functions and Data Members:**
  - These are written in lowerCamelCase.
  - For method parameters, we use **'p_'** as prefix and for data members, we use '**d_'** as prefix.
  - Example : int d_mapIndex = 1, public GameMap loadMap(String p_mapFileName)
- **Local Variables:**
  - Adheres to lower camelCase with the prefix **'l_'** to indicate that it is a local variable.
  - Example : String l_line;
- **Constants:**
  - Uppercase letters with underscores.
  - Example : public static final String MAP_PATH = "src/main/resource/map/";

Guidelines for Writing Comments

- **Code without any commented-out sections:**
  - Excess commented code is removed to enhance clarity and readability.
- **Documentation comments (JavaDoc):**
  - For each class, data member, and member function, there will be JavaDoc documentation provided above them.
- **In Complex Methods :**
  - In lengthy and complex methods, explanatory content is inserted to improve code comprehension.

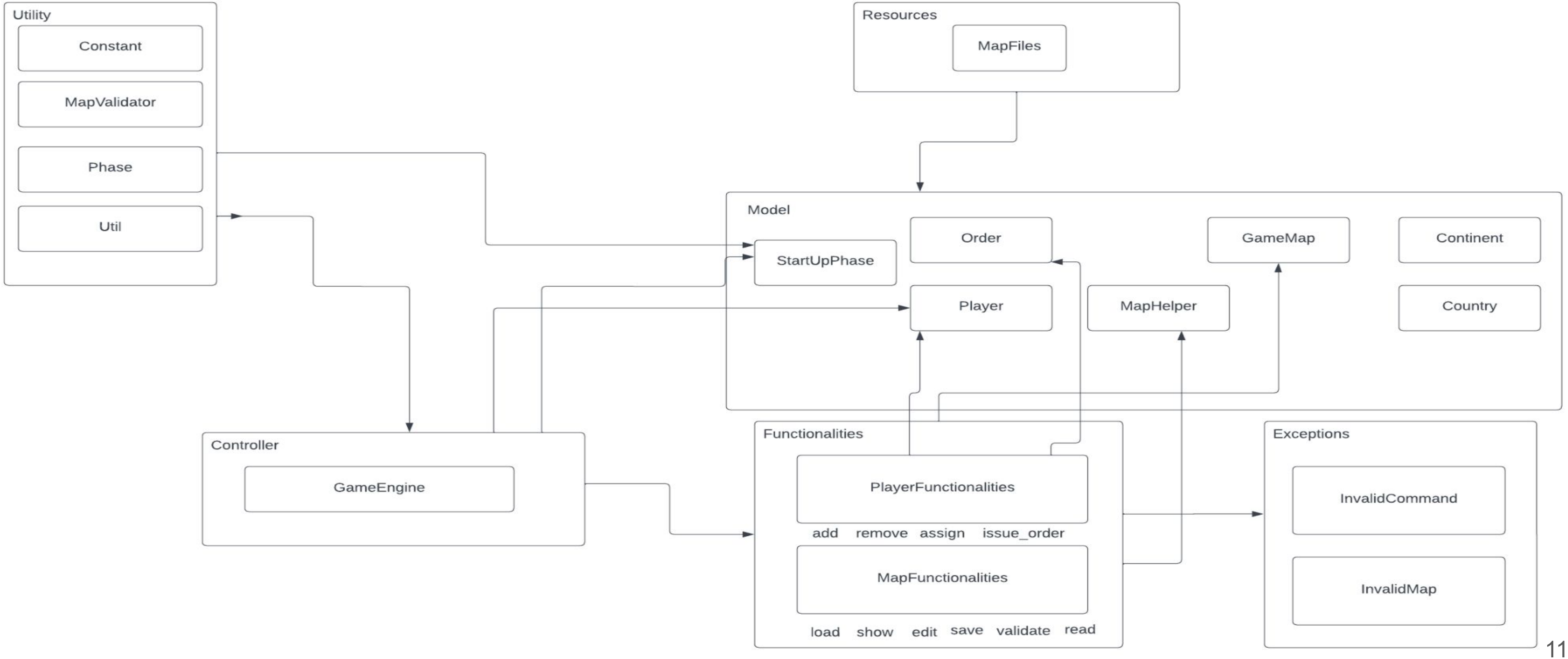
StartUpPhase.java



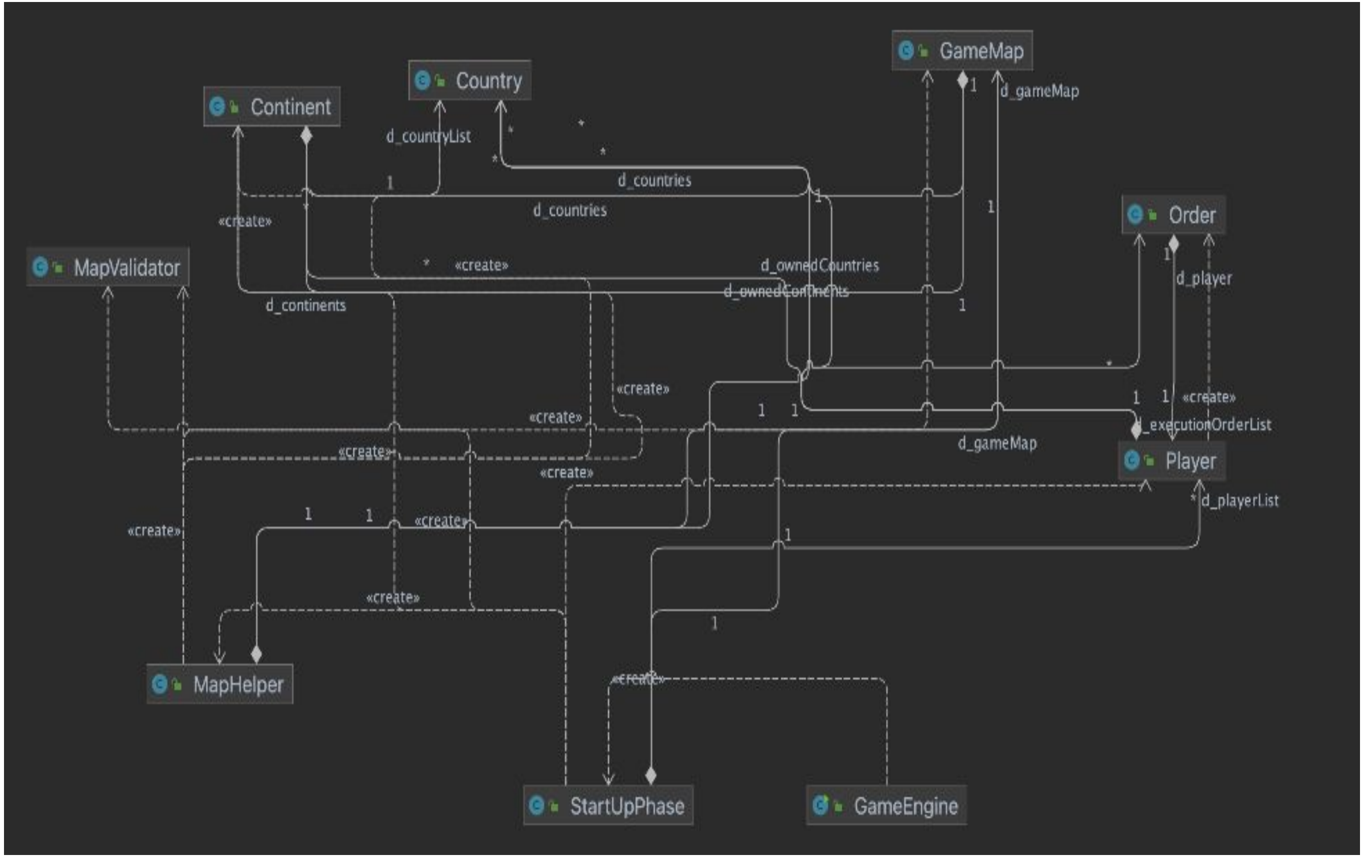
```
/**
 * Getter method for player list.
 *
 * @return Returns list of game player.
 */
public ArrayList<Player> getPlayerList() {
    return this.d_playerList;
}

/**
 * Setter method for game phase.
 *
 * @param p_gamePhase GamePhase to set
 */
public void setGamePhase(Phase p_gamePhase) {
    this.d_gamePhase = p_gamePhase;
}
```

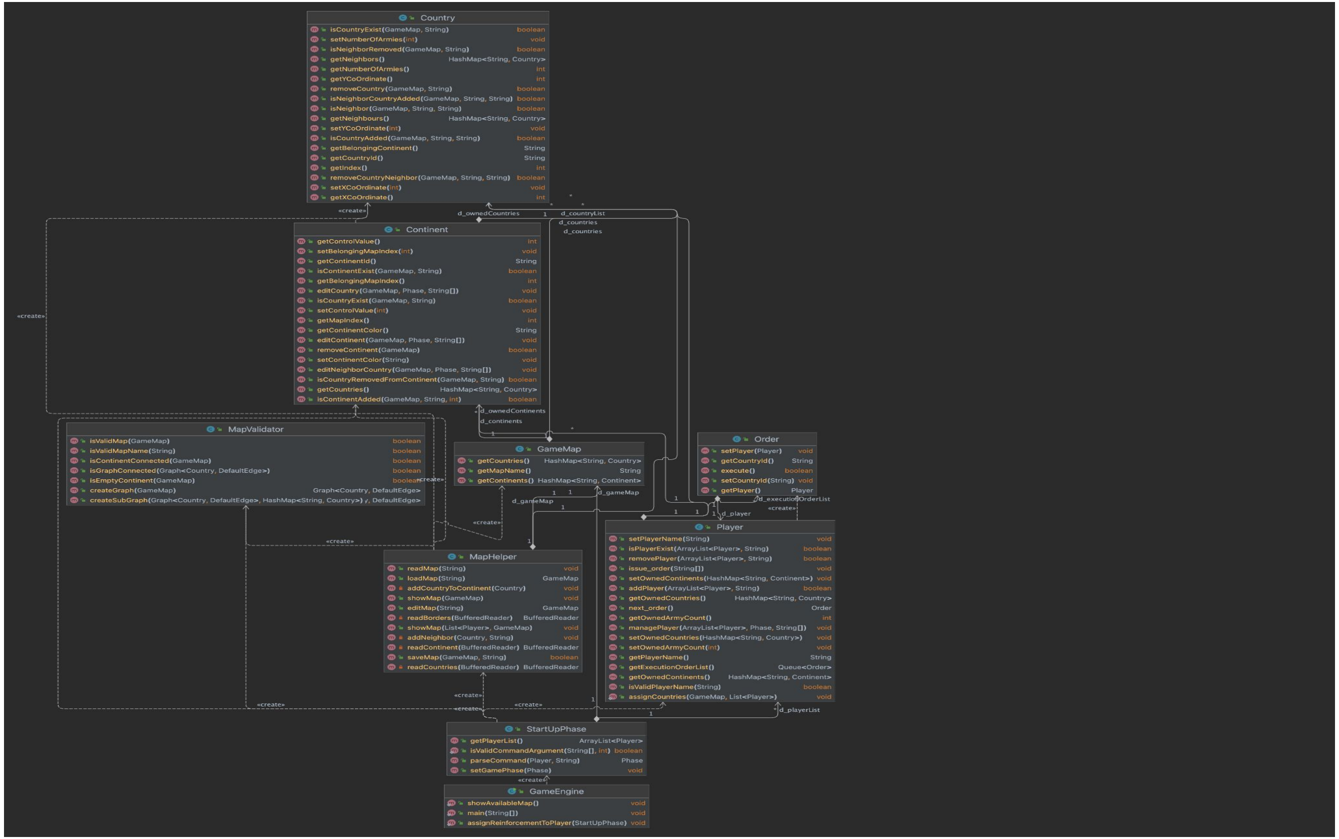
Architecture Diagram



Class Diagram



Functional Diagram



Country	
isCountryExist(GameMap, String)	boolean
setNumberOfArmies(int)	void
isNeighborRemoved(GameMap, String)	boolean
getNeighbors()	HashMap<String, Country>
getNumberOfArmies()	int
getYCoOrdinate()	int
removeCountry(GameMap, String)	boolean
isNeighborCountryAdded(GameMap, String, String)	boolean
isNeighbor(GameMap, String, String)	boolean
getNeighbours()	HashMap<String, Country>
setYCoOrdinate(int)	void
isCountryAdded(GameMap, String, String)	boolean
getBelongingContinent()	String
getCountryId()	String
getIndex()	int
removeCountryNeighbor(GameMap, String, String)	boolean
setXCoOrdinate(int)	void
getXCoOrdinate()	int

Continent	
getControlValue()	int
setBelongingMapIndex(int)	void
getContinentId()	String
isContinentExist(GameMap, String)	boolean
getBelongingMapIndex()	int
editCountry(GameMap, Phase, String[])	void
isCountryExist(GameMap, String)	boolean
setControlValue(int)	void
getMapIndex()	int
getContinentColor()	String
editContinent(GameMap, Phase, String[])	void
removeContinent(GameMap)	boolean
setContinentColor(String)	void
editNeighborCountry(GameMap, Phase, String[])	void
isCountryRemovedFromContinent(GameMap, String)	boolean
getCountries()	HashMap<String, Country>
isContinentAdded(GameMap, String, int)	boolean

MapValidator	
isValidMap(GameMap)	boolean
isValidMapName(String)	boolean
isContinentConnected(GameMap)	boolean
isGraphConnected(Graph<Country, DefaultEdge>)	boolean
isEmptyContinent(GameMap)	boolean
createGraph(GameMap)	Graph<Country, DefaultEdge>
createSubGraph(Graph<Country, DefaultEdge>, HashMap<String, Country>)	y, DefaultEdge>

GameMap	
getCountries()	HashMap<String, Country>
getMapName()	String
getContinents()	HashMap<String, Continent>

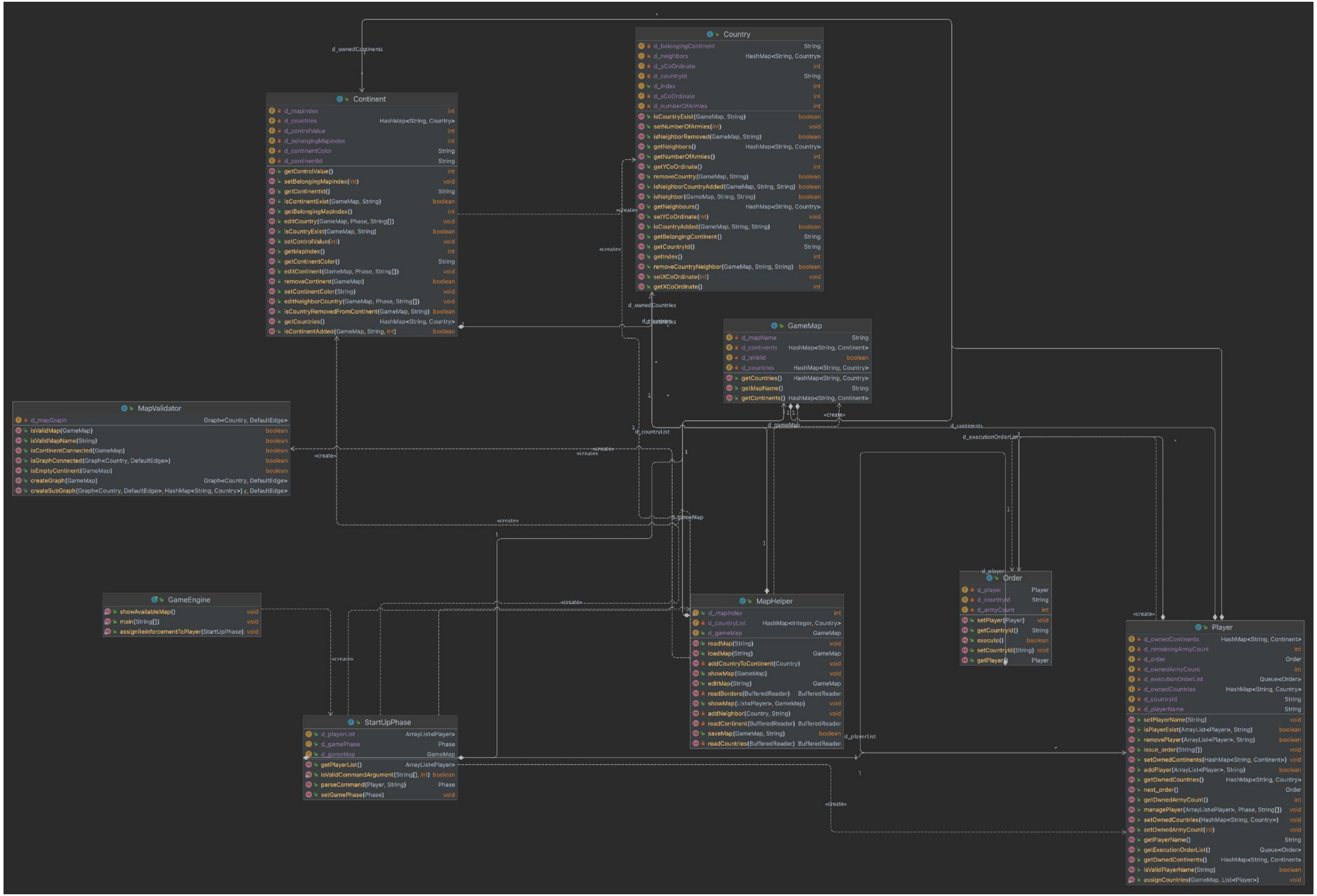
Order	
setPlayer(Player)	void
getCountryId()	String
execute()	boolean
setCountryId(String)	void
getPlayer()	Player

MapHelper	
readMap(String)	void
loadMap(String)	GameMap
addCountryToContinent(Country)	void
showMap(GameMap)	void
editMap(String)	GameMap
readBorders(BufferedReader)	BufferedReader
showMap(List<Player>, GameMap)	void
addNeighbor(Country, String)	void
readContinent(BufferedReader)	BufferedReader
saveMap(GameMap, String)	boolean
readCountries(BufferedReader)	BufferedReader

Player	
setPlayerName(String)	void
isPlayerExist(ArrayList<Player>, String)	boolean
removePlayer(ArrayList<Player>, String)	boolean
issue_order(String[])	void
setOwnedContinents(HashMap<String, Continent>)	void
addPlayer(ArrayList<Player>, String)	boolean
getOwnedCountries()	HashMap<String, Country>
next_order()	Order
getOwnedArmyCount()	int
managePlayer(ArrayList<Player>, Phase, String[])	void
setOwnedCountries(HashMap<String, Country>)	void
setOwnedArmyCount(int)	void
getPlayerName()	String
getExecutionOrderList()	Queue<Order>
getOwnedContinents()	HashMap<String, Continent>
isValidPlayerName(String)	boolean
assignCountries(GameMap, List<Player>)	void

StartUpPhase	
getPlayerList()	ArrayList<Player>
isValidCommandArgument(String[], int)	boolean
parseCommand(Player, String)	Phase
setGamePhase(Phase)	void

GameEngine	
showAvailableMap()	void
main(String[])	void
assignReinforcementToPlayer(StartUpPhase)	void



Continent

d_mapIndex	int
d_countries	HashMap<String, Country>
d_controlValue	int
d_belongingMapIndex	int
d_continentColor	String
d_continentId	String
getControlValue()	int
setBelongingMapIndex(int)	void
getContinentId()	String
isContinentExist(GameMap, String)	boolean
getBelongingMapIndex()	int
editCountry(GameMap, Phase, String[])	void
isCountryExist(GameMap, String)	boolean
setControlValue(int)	void
getMapIndex()	int
getContinentColor()	String
editContinent(GameMap, Phase, String[])	void
removeContinent(GameMap)	boolean
setContinentColor(String)	void
editNeighborCountry(GameMap, Phase, String[])	void
isCountryRemovedFromContinent(GameMap, String)	boolean
getCountries()	HashMap<String, Country>
isContinentAdded(GameMap, String, int)	boolean

Country

d_belongingContinent	String
d_neighbors	HashMap<String, Country>
d_yCoOrdinate	int
d_countryId	String
d_index	int
d_xCoOrdinate	int
d_numberOfArmies	int
isCountryExist(GameMap, String)	boolean
setNumberOfArmies(int)	void
isNeighborRemoved(GameMap, String)	boolean
getNeighbors()	HashMap<String, Country>
getNumberOfArmies()	int
getYCoOrdinate()	int
removeCountry(GameMap, String)	boolean
isNeighborCountryAdded(GameMap, String, String)	boolean
isNeighbor(GameMap, String, String)	boolean
getNeighbours()	HashMap<String, Country>
setYCoOrdinate(int)	void
isCountryAdded(GameMap, String, String)	boolean
getBelongingContinent()	String
getCountryId()	String
getIndex()	int
removeCountryNeighbor(GameMap, String, String)	boolean
setXCoOrdinate(int)	void
getXCoOrdinate()	int

GameMap

d_mapName	String
d_continents	HashMap<String, Continent>
d_isValid	boolean
d_countries	HashMap<String, Country>
getCountries()	HashMap<String, Country>
getMapName()	String
getContinents()	HashMap<String, Continent>

MapValidator

d_mapGraph	Graph<Country, DefaultEdge>
isValidMap(GameMap)	boolean
isValidMapName(String)	boolean
isContinentConnected(GameMap)	boolean
isGraphConnected(Graph<Country, DefaultEdge>)	boolean
isEmptyContinent(GameMap)	boolean
createGraph(GameMap)	Graph<Country, DefaultEdge>
createSubGraph(Graph<Country, DefaultEdge>, HashMap<String, Country>)	γ, DefaultEdge>

GameEngine

showAvailableMap()	void
main(String[])	void
assignReinforcementToPlayer(StartUpPhase)	void

StartUpPhase

d_playerList	ArrayList<Player>
d_gamePhase	Phase
d_gameMap	GameMap
getPlayerList()	ArrayList<Player>
isValidCommandArgument(String[], int)	boolean
parseCommand(Player, String)	Phase
setGamePhase(Phase)	void

MapHelper

d_mapIndex	int
d_countryList	HashMap<Integer, Country>
d_gameMap	GameMap
readMap(String)	void
loadMap(String)	GameMap
addCountryToContinent(Country)	void
showMap(GameMap)	void
editMap(String)	GameMap
readBorders(BufferedReader)	BufferedReader
showMap(List<Player>, GameMap)	void
addNeighbor(Country, String)	void
readContinent(BufferedReader)	BufferedReader
saveMap(GameMap, String)	boolean
readCountries(BufferedReader)	BufferedReader

Order

d_player	Player
d_countryId	String
d_armyCount	int
setPlayer(Player)	void
getCountryId()	String
execute()	boolean
setCountryId(String)	void
getPlayer()	Player

Player

d_ownedContinents	HashMap<String, Continent>
d_remainingArmyCount	int
d_order	Order
d_ownedArmyCount	int
d_executionOrderList	Queue<Order>
d_ownedCountries	HashMap<String, Country>
d_countryId	String
d_playerName	String
setPlayerName(String)	void
isPlayerExist(ArrayList<Player>, String)	boolean
removePlayer(ArrayList<Player>, String)	boolean
issue_order(String[])	void
setOwnedContinents(HashMap<String, Continent>)	void
addPlayer(ArrayList<Player>, String)	boolean
getOwnedCountries()	HashMap<String, Country>
next_order()	Order
getOwnedArmyCount()	int
managePlayer(ArrayList<Player>, Phase, String[])	void
setOwnedCountries(HashMap<String, Country>)	void
setOwnedArmyCount(int)	void
getPlayerName()	String
getExecutionOrderList()	Queue<Order>
getOwnedContinents()	HashMap<String, Continent>
isValidPlayerName(String)	boolean
assignCountries(GameMap, List<Player>)	void

References

Warzone game

Domination game maps

As specified in the Project Build 1 document

Thank You