SOEN 6441

Advanced Programming Practices

Instructor: Professor Amin Ranjbar

Build 3

Refactoring Document

Team 20

Ali Molla Mohammadi …….40042597
Konstantin Chemodanov…. 40049285
Farzaneh Jamshidi….…...…40075234
Negar Ashouri………………40071530
Saman Safaei………………. 40119399

March 2019

Department of Computer Science and Software Engineering

Refactoring:

After receiving the feedbacks of the second build and new requirements for the final build, we found out that some specific parts of our code needs refactoring in order to meet new requirements. Beside all the new parts and functionalities that we had to add to the code, we came up with a list of potential refactoring targets, as we had to apply and add Strategy pattern design and we needed to add new game modes.

We faced challenges during the refactoring operations such as implementing four different player behavior strategies with "strategy pattern" and different versions of the reinforcement, attack and fortification as methods of the Player class. So we made player the abstract class and added five new classes (aggressive, benevolent, random, cheater, and regular player) as children of the player class. Game phases are all interfaces, using the strategy pattern to change the way each phase call methods. Based on the type of the player behavior, each phase will do some specific calculations for all parameters and arguments and based on these calculations it will call proper methods. We have an additional implementation in every phase for the "cheater" as it does not follow the game rules.
Next challenge was implementing the "*Save and Load*" for the user, as we had to implement functions in order to enable user to save the game in a progress into a file and to allow them to load it back in exactly the same state as saved including the map and owned countries and armies and etc. for doing that To serialize an object means to convert its state to a byte stream so that the byte stream can be reverted back into a copy of the object. A Java object is serializable if its class or any of its superclasses implements either
the java.io.Serializable interface or its subinterface, java.io.Externalizable. Deserialization is the process of converting the serialized form of an object back into a copy of the object.
For example, the java.awt.Button class implements the Serializable interface, so you can serialize a java.awt.Button object and store that serialized state in a file. Later, you can read back the serialized state and deserialize into a java.awt.Button object.
The Java platform specifies a default way by which serializable objects are serialized. A (Java) class can override this default serialization and define its own way of serializing objects of that class. When an object is serialized, information that identifies its class is recorded in the serialized stream. However, the class's definition ("class file") itself is not recorded. It is the responsibility of the system that is deserializing the object to determine how to locate and load the necessary class files. For example, a Java application might include in its classpath a JAR file that contains the class files of the serialized object(s) or load the class definitions by using information stored in the directory.

We have also added the single game mode and if user choose to play in single game mode, they have to select a saved map file of their own and will see the map as a connected graph.

User chooses the number and behavior of players (aggressive, benevolent, random, cheater). The game proceeds until one of the players has conquered the whole map. If no human player is selected, the game proceeds fully automatically without any user interaction.
Also there is this new class "tournament" which provides an option for a Tournament Mode for user once game has started. User should choose maps and number of players and behavior of players and number of the games that should be played on each map and also the maximum number of turns in each game. A tournament then will be automatically played by playing games on each of the different maps between the chosen computer player strategies. In order to minimize run completion time, each game should be declared a draw after the chosen turns. Once started, the tournament plays all the games automatically without user interaction. At the end of the tournament, a report of the results will be displayed as you can see in the picture below.

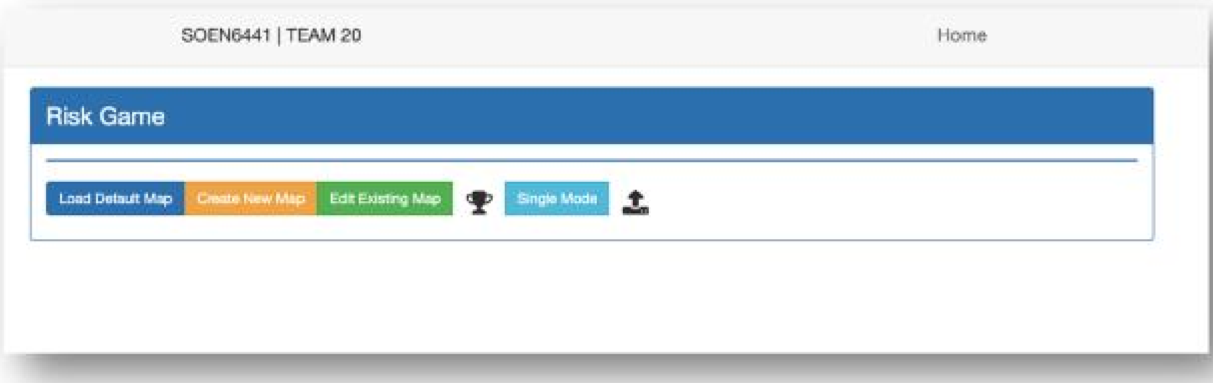
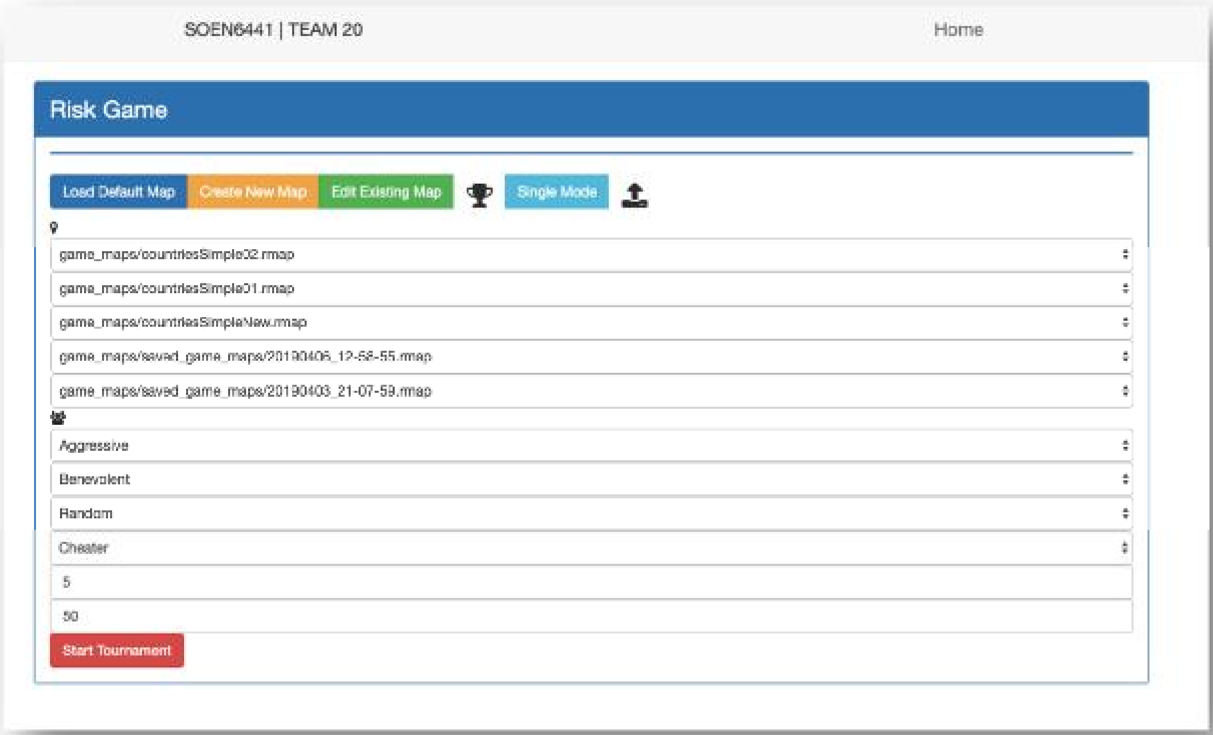
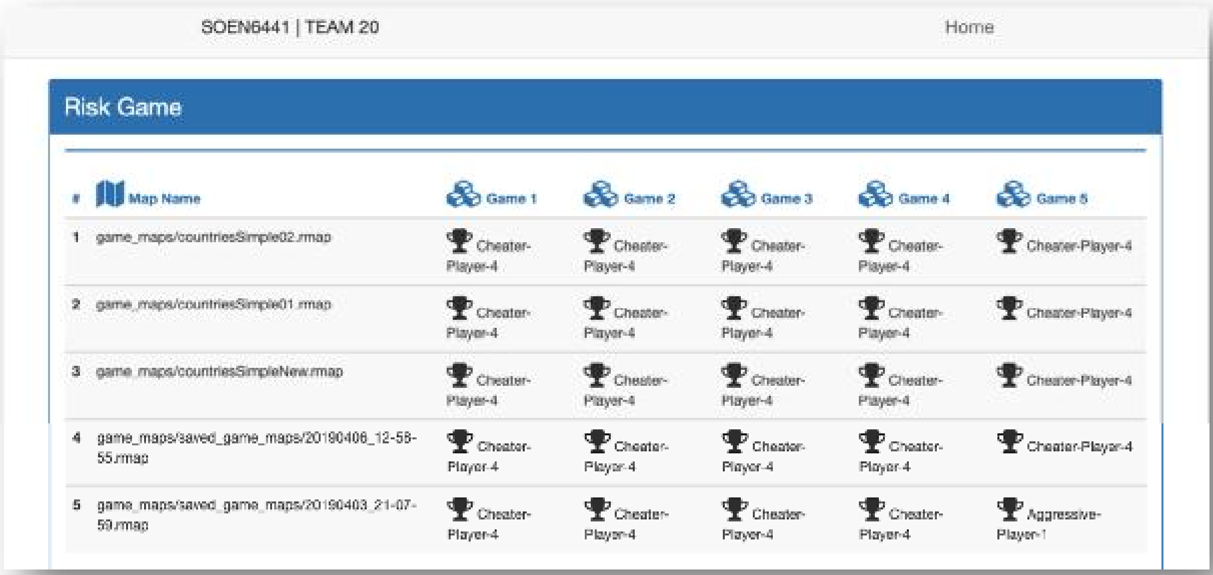


Image-1: Initial page in risk game



game_maps/countriesSimple02.rmap
game_maps/countriesSimple01.rmap
game_maps/countriesSimpleNew.rmap
game_maps/saved_game_maps/20190406_12-58-55.rmap
game_maps/saved_game_maps/20190403_21-07-59.rmap
Aggressive
Benevolent
Random
Cheater
5
50

Image-2: Choosing data for stating Tournament Phase



#	Map Name	Game 1	Game 2	Game 3	Game 4	Game 5
1	game_maps/countriesSimple02.rmap	Cheater-Player-4	Cheater-Player-4	Cheater-Player-4	Cheater-Player-4	Cheater-Player-4
2	game_maps/countriesSimple01.rmap	Cheater-Player-4	Cheater-Player-4	Cheater-Player-4	Cheater-Player-4	Cheater-Player-4
3	game_maps/countriesSimpleNew.rmap	Cheater-Player-4	Cheater-Player-4	Cheater-Player-4	Cheater-Player-4	Cheater-Player-4
4	game_maps/saved_game_maps/20190406_12-58-55.rmap	Cheater-Player-4	Cheater-Player-4	Cheater-Player-4	Cheater-Player-4	Cheater-Player-4
5	game_maps/saved_game_maps/20190403_21-07-59.rmap	Cheater-Player-4	Cheater-Player-4	Cheater-Player-4	Cheater-Player-4	Aggressive-Player-1

Image-3 : Report from ending Tournament phase game