**Part 1: Command processor and command adapter**

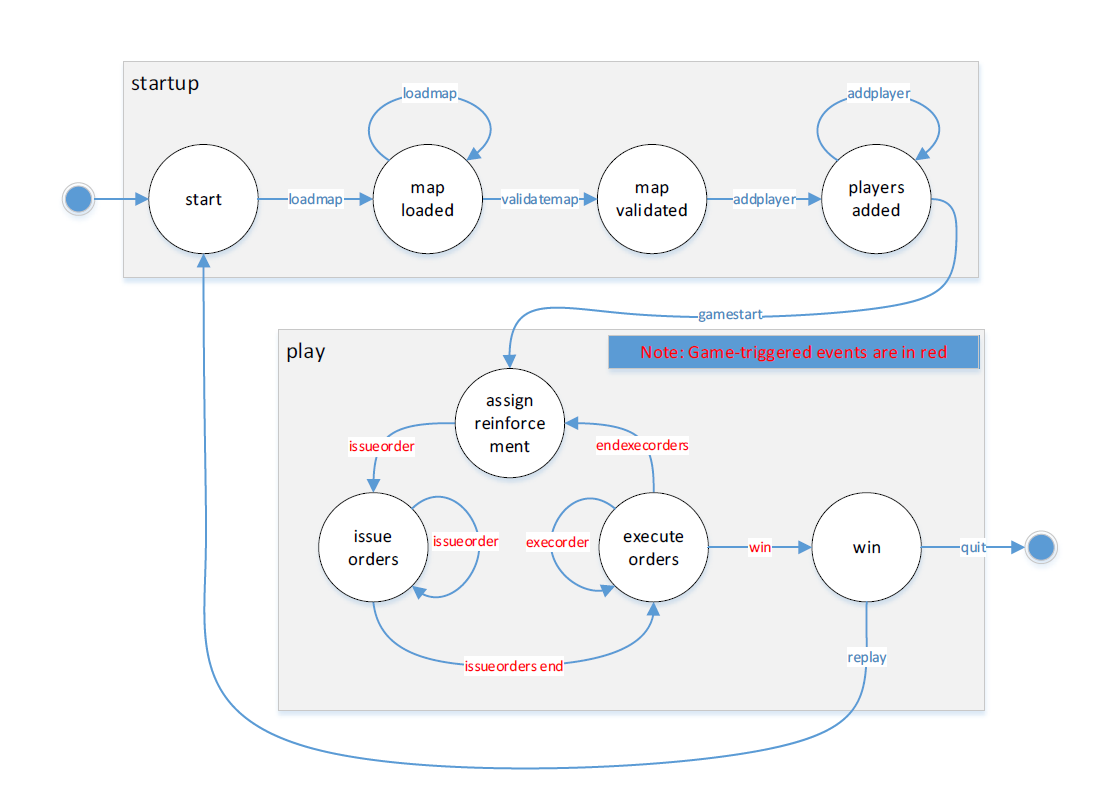
Implement a **CommandProcessor** class that gets commands from the console as a string using its **readCommand()** private method which stores the command internally in a collection of **Command** objects using the **saveCommand()** method, and provides a public **getCommand()** method to other objects such as the **GameEngine** or the **Player**. Once a command gets executed, the effect of the command can be stored as a string in the **Command** object using the **saveEffect()** method. Any game component that operates using commands must get its commands from a **CommandProcessor** object. The command processor should have a **validate()** method that can be called to check if the command is valid in the current game state. If the command is not valid, a corresponding error message should be saved in the effect of the command.

Implement a **FileCommandProcessorAdapter** class that enables the same functionality as the above, except that it reads the commands sequentially from a previously saved text file. This class needs to abide with all the design characteristics and restrictions of the Adapter design pattern. The application should accept a command line argument that enables the user to choose between accepting the commands from the console (-**console**) or from a file (**-file <filename>**).

The commands are:

|  |  |  |
| --- | --- | --- |
| **command** | **valid in state** | **transitions to** |
| loadmap <mapfile> | start, maploaded | maploaded |
| validatemap | maploaded | mapvalidated |
| addplayer <playername> | mapvalidated, playersadded | playersadded |
| gamestart | playersadded | assignreinforcement |
| replay | win | start |
| quit | win | exit program |





**Part 2: Tournament mode**

During the start game state, a new tournament command can be entered by the user, which triggers the Tournament Mode. While in the tournament mode, the game should proceed without any user interaction and show the results of the tournament at the end. The tournament command is the following:

***tournament -M <listofmapfiles> -P <listofplayerstrategies> -G <numberofgames> -D <maxnumberofturns>***

This command lets the user choose the parameters of the tournament, i.e.: *M = 1 to 5* different maps, *P = 2 to 4* different computer players strategies, *G = 1 to 5* games to be played on each map, and *D = 10 to 50* maximum number of turns for each game. Once the command is entered and validated, the tournament is automatically played by playing G games on each of the M different maps between the chosen computer player strategies. In order to minimize run completion time, each game should be declared a draw after D turns. Once started, the tournament plays all the games automatically without user interaction. At the end of the tournament, a report of the results should be output to the log file, e.g.

Tournament mode: M: Map1, Map2, Map3

P: Aggressive, Benevolent, Neutral, Cheater.

G: 4

D: 30

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Game 1 | Game 2 | Game 3 | Game 4 |
| Map 1 | Aggressive | Neutral | Cheater | Cheater |
| Map 2 | Cheater | Draw | Cheater | Aggressive |
| Map 3 | Cheater | Aggressive | Cheater | Draw |