Concordia University
Department of Computer Science
and Software Engineering

Advanced program design with C++
COMP 345 --- Fall 2018

Team project assignment #3

Deadline:	November 16th, 2018
Evaluation:	8% of final mark
Late submission:	not accepted
Teams:	this is a team assignment

Problem statement

This is a team assignment. It is divided into distinct parts. Each part is about the development of a part of the topic presented as the team project. Even though it is about the development of a part of your team project, each assignment is to be developed/presented/tested separately. The description of each part describes what are the features that the part should implement, and what you should demonstrate. Note that the following descriptions describe the baseline of the assignment, and are related to the project description. See the course web page for a full description of the team project, as well as links to the details of the game rules to be implemented.

Part 1: Player Strategy Pattern

Using the Strategy design pattern, implement different kinds of players that make different decisions during the reinforcement, attack, and fortification phases. The kinds of players are: (1) human player that requires user interaction to make decisions, (2) an aggressive computer player that focuses on attack (reinforces its strongest country, then always attack with it until it cannot attack anymore, then fortifies in order to maximize aggregation of forces in one country), (3) a benevolent computer player that focuses on protecting its weak countries (reinforces its weakest countries, never attacks, then fortifies in order to move armies to weaker countries). You must deliver a driver that demonstrates that (1) different players can be assigned different strategies that lead to different behavior for the reinforcement, attack, and fortification phases using the strategy pattern; (2) the strategy adopted by a player can be changed dynamically during play, (3) the human player makes decisions according to user interaction, and computer players make decisions automatically, which are both implemented using the strategy pattern.

Part 2: Phase Observer

Using the Observer design pattern, implement a view that displays information happening in the current phase. It should first display a header showing what player and what phase is currently being played, e.g. “Player 3: Attack phase” or “Player 1: Fortification phase” Then it should display important information related to what is happening in this phase, which should be different depending on what phase is being played. This should dynamically be updated as the game goes through different players/phases and be visible at all times during game play. You must deliver a driver that demonstrates that (1) the information displayed by the phase view is cleared every time the phase is changing (2) the phase view is displaying the correct player:phase information as soon as the phase changes; (3) the phase view displays relevant information which is different for every phase.

Part 3: Game Statistics Observer

Using the Observer design pattern, implement a view that displays some useful statistics about the game, the minimum being a “players world domination view” that shows using some kind of graph or table depicting what percentage of the world is currently being controlled by each player. This should dynamically be updated as the map state changes and be visible at all times during game play. You must deliver a driver that demonstrates that (1) the game statistics view updates itself every time a country has been conquered by a player; (2) the game statistics updates itself when a player has been eliminated and removes this player from the view; (3) as soon as a player owns all the countries, the game statistics view updates itself and displays a celebratory message.

Assignment submission requirements and procedure

You are expected to submit a group of C++ files implementing a solution to each of the separate problems stated above (Part 1, 2, 3). Your code must include a *driver* (i.e. a `main` function) for each part that allows the marker to observe the execution of each part during the lab demonstration. Each driver should simply create the components described above and demonstrate that they behave as mentioned above.

You have to submit your assignment before midnight on the due date using the ENCS Electronic Assignment Submission system under the category “programming assignment 3”. Late assignments are not accepted. The file submitted must be a .zip file containing all your code. You are allowed to use any C++ programming environment as long as you can demonstrate your assignment in the labs.

Evaluation Criteria

Knowledge/correctness of game rules:	2 pts (indicator 4.1)
Compliance of solution with stated problem (see description above):	12 pts (indicator 4.4)
Modularity/simplicity/clarity of the solution:	2 pts (indicator 4.3)
Proper use of language/tools/libraries:	2 pts (indicator 5.1)
Code readability: naming conventions, clarity of code, use of comments:	2 pts (indicator 7.3)
Total	**20 pts (indicator 6.4)**