Concordia University
Department of Computer Science and Software Engineering

Advanced program design with C++
COMP 345 --- Fall 2015

Individual assignment #3

Deadline:	Friday, November 27th, 2015
Evaluation:	5% of final mark
Late submission:	not accepted
Teams:	this is an individual assignment

Problem statement

This is an individual assignment. It is divided into three distinct parts. Each individual student is expected to select one of these parts as his/her assignment. 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 has to be developed independently of the others and is not to be presented as an integrated part of the team project. Each member of your team is free to choose to do any part, and is expected to follow a different design approach than the other team members that have selected the same assignment topic. 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).

Part 1: Map Save/Load Adapter Pattern

Use an Adapter pattern to enable your game to save/load a map into two different file formats, where the save/load functions done in assignment 1 are the Adaptee, and you design a new Adapter class that can be called in the same way, but reading/writing in a different file format. One of your two file formats must be the file format for the conquest game (http://www.windowsgames.co.uk/conquest_maps.html).

Part 2: Game Save/Load Builder Pattern

Use a Builder pattern to enable the saving/loading of a game in progress, where the different parts to be constructed are for example: 1) the current state of the map e.g. the map itself, along with how many armies are one each country and to which player they belong, 2) the current state of the players e.g. the number of players, the cards held by each player, the strategy adopted by each computer player, 3) the current state of the game engine e.g. what player's turn/phase it was as the game was saved.

Part 3: Game Statistics Observer/Decorator Pattern

Use a combination of Observer and Decorator pattern to provide a game statistics observer that is updated in real time as the game is being played. The basic statistics displayed by the game statistics observer are: 1) number of countries controlled by each player, 2) total number of armies owned by each player and 3) number of cards owned by each player. Use a decorator pattern to allow the addition of other statistics to be displayed by the game statistics observer, for example: percentage of world controlled by each player, percentage of battles won/lost by each player, etc. Each new statistic must be implemented as a different decorator that is decorating the game statistics observer.

Part 4: Game Log Observer/Decorator Pattern

Use an Observer pattern to provide a game log observer that shows dynamically selected actions taken in selected players/phases of the game. Use a Decorator pattern to enable the dynamic toggling of logging for each individual player's actions or logging of individual phases of the game. For example, the Log Observer can show all phases for all players, but can also show the logging for only one player, or show the log of only the battle phases.

Assignment submission requirements and procedure

You are expected to submit a group of C++ files implementing a solution to one of the problems stated above (Part 1, 2, 3 or 4). Your code must include a *driver* that allows the marker to compile and execute your code on a standard lab computer. The driver should use the pattern to manage game objects and somehow demonstrate that the code conforms to the above-mentioned specifications, as well as following the applicable tower defense game rules, and that the patterns are correctly implemented. The use of unit testing such as cppUnit is not mandatory but encouraged. Along with your submitted code, you have to explain your analysis and design. Briefly explain the game rules involved in the creation of you assignment, citing external sources for specific game rules. Briefly describe the design you adopted as a solution. The design description can be backed-up, for example, by doxygen-generated documentation, regular code comments, or a simple diagram. The focus of this course being the coding aspect of software development, you are discouraged to submit extensive documentation.

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 they are available in the labs. No matter what programming environment you are using, you are responsible to give proper compilation and usage instructions to the marker in a README file to be included in the zip file.

Evaluation Criteria

Analysis:	
Clarity and correctness of statement of game rules involved:	5 pts (indicator 4.1)
Design:	
Compliance of solution with stated problem:	20 pts (indicator 4.4)
Simplicity and appropriateness of the solution:	3 pts (indicator 4.3)
Clarity of design description:	3 pts (indicator 7.3)
Use of tools:	
Rationale for selection of tools/libraries:	2 pts (indicator 5.2)
Proper use of language/libraries:	4 pts (indicator 5.1)
Implementation:	
Code readability: naming conventions, clarity, use of comments:	3 pts (indicator 7.3)
Coding style: `.h` and `.cpp` files, use of comments:	3 pts (indicator 5.1)
Relevance of driver and presented results:	7 pts (indicator 4.4)
Total	**50 pts (indicator 6.4)**