Concordia University
Department of Computer Science and Software Engineering

Advanced program design with C++
COMP 345 --- Fall 2016

Individual assignment #2

Deadline:	Sunday, November 6th, 2016
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 four 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, or include the implementation of one another's aspects. 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: Character observer

Implement an Observer pattern for the Character class as implemented in assignment 1. Make the Character class an Observable class, then implement a Character Observer class that displays the character's view. You may use whatever library or solution you see fit to display the character's view (GUI or not). Provide a driver class that creates a character, plugs a Character Observer to it, then changes some values in the character, triggering the re-display of the character view each time a value is changed in the character, using the observer pattern mechanism.

Part 2: Map observer

Implement an Observer pattern for the Map class as implemented in assignment 1. Make the Map class an Observable class, then implement a Map Observer class that displays the map's view. You may use whatever library or solution you see fit to display the map's view (GUI or not). Provide a driver class that creates a map, plugs a Map Observer to it, then changes some values in the map (e.g. moving the character on the map), triggering the re-display of the map view each time a value is changed in the map, using the observer pattern mechanism.

Part 3: Interactive map/campaign editor

Develop a user-interactive C++ component that enables the user to create/edit a maps and connect them in a campaign. The maps and campaigns should have the specifications and constraints as stated in the project description and the first assignment. Using the map editor, the user should be able to 1) create new maps 2) define the adjacency relationships between maps to create campaigns. The editor should allow the user to either create a new map/campaign from scratch or load and edit an existing map/campaign previously saved as a file. The editor should allow the user to save the edited map/campaign to a file (every map and every campaign should be saved as a separate file). Verification of map/campaign validity (as expressed in project description and in the first assignment) should be applied before the file is loaded or saved.

Part 4: Map builder

Use the builder pattern to create a map object by reading it from a file. Create two concrete builders: 1) A concrete builder that reads the map as saved. This is going to be used by the map editor. 2) A concrete builder that reads a map from a file and adapts its content to a specific level. This is going to be used during game play every time a new map is entered by the character. Upon creating the map, this concrete builder will adapt all characters on the map to the level given in input (e.g. the level of the character entering the map), as well as adapting all items contained in the map.

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 simply create a character, map, item container, or dice object and somehow demonstrate that the character, map, item container or dice was created following the above-mentioned specifications, as well as following the applicable d20 game rules.

For each part, you must design a CppUnit test suite composed of at least two relevant test cases. The test cases you provide must be documented. The header file(s) of your implementation must include in Doxygen documentation a statement of: 1) the game rules involved in their respective implementation, 3) a brief description of your design, 3) the libraries used in the code, including a rationale for the selection of these libraries. The submitted code must include a driver (i.e. a main function) that demonstrates possible uses of the implemented code. The focus of this course being the coding aspect of software development, you are discouraged to submit external 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 2”. 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 usable 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:		
	Completeness/correctness of statement of game rules involved (Doxygen) :	5 pts (indicator 4.1)
Design:		
	Compliance of solution with stated problem and game rules:	15 pts (indicator 4.4)
	Simplicity and appropriateness of the solution:	7 pts (indicator 4.3)
	Clarity/completeness of Doxygen documentation:	3 pts (indicator 7.3)
Use of tools:		
	Rationale for selection of tools/libraries used:	2 pts (indicator 5.2)
	Proper use of language/tools/libraries:	3 pts (indicator 5.1)
Implementation:		
	Code readability: naming conventions, clarity, use of comments:	2 pts (indicator 7.3)
	Coding style: `.h` and .cpp files:	2 pts (indicator 5.1)
	Relevance of test cases provided:	4 pts (indicator 4.4)
	Relevance of driver and completeness of presented results:	7 pts (indicator 4.4)
Total		**50 pts (indicator 6.4)**