How well designed was the code for extensions, what particular elements aided or hindered extensibility?

The code designed is good overall, the signals and slot part aided the extensibility very much, also stage1ball/table/factory and stage2ball/table/factory aided the extensibility as well. However the game factory and builder all use JSON value to create new object, it hindered the extensibility very much. It is very difficult to create a new ball or a new table through factory and builder without JSON file.

How well documented was the code with respect to both external documentation and comments?

Commence is good enough for most of the header file, the cpp file don't have any comments at all. The comments type is very standard dioxygen type for every class and function.

Was the coding well done? What would you have done differently? What was good/bad about the implementation?

The code is well done overall, however the duplicatedBallDecorator and CueBallDecorator is very duplicated indeed. In my Assinment2 code, three mouseEvent can do the same functionality. Also some of the function was done by the header file, and some was done by the source file, it will confused the following programmers. Finally, the stage2ball contain too many class, it contain stage2ball class, compositeball class and simpleStage2ball class, which is bad on code implementation.

Comment on the style of the code. Were names, layout, and code clichés consistent?

The names, consistent of the code is very well, names is meaningful and easy to understand. However the layout is comparably bad on stage2ball class, duplicatedBallDecorator class and CueBallDecorator class.

Explain the application of the design patterns for your code.

The project use Memento and prototype design patterns.

The prototype design patterns add a clone function to all Ball class. The clone function can deep copy every single Ball object.

The Memento design patterns can revert back the whole game state by deep copy the PoolGame class as a state. Every Memento class has a different deep copy new pointer of the PoolGame. The Originator can save each memento by create a new memento pointer, and revert back to a special memento by revert () function.

Explain advantage and disadvantages of the design patterns used with respect to your code.

Prototype:

Advantage: new prototypes can be created at runtime just by creating a new instance and registering it as a prototype and reduce the class subtyping.
Prototype is a simple way to deep copy a complex object hierarchy by calling the internal function. Prototype also can reduced load of initialization, aided extensibility, and optimized coding efforts.

Disadvantage: All of the subclass must have a clone operation, sometime tricky with deep copy.

Memento:

Advantage: Preserves pre-existing encapsulation as only the Memento can retrieve its own data, removes the requirement of originating objects keeping track of previous states.
Memento is simple enough for revert back operation. Memento also good for auditing, it can check the output and error in each state, allocate the bug.

Disadvantage: Using memento might be very expensive, Mementos might cause overhead if Originator must copy large amounts of state to store in the memento or if dialog create and return mementos to the originator very often.

Memento might be very hard to satisfy for some programming language.