**The application of design patterns for the concrete code base:**

In the program, I choose two main design patterns abstract factory and builder, also in order to implement abstract factory, I have to use factory method.

Generally, I follow the requirements that use abstract factory to get the elements of games and use builder to finalize the installation work. For abstract factory, I get abstract factory, concrete factory, abstract product and concrete product. I use these classes to make up my abstract factory structure. The concrete factory derived from abstract factory produces concrete product derived from abstract product. And in the factory, it can produce 2 different kinds of product pool balls and pool table as the same time to form a pool game. For builder, I replace the product class with the abstract factory structure above, concrete builder derived from abstract builder builds the game by the product from product according to the director.

In conclusion, the main body of the structure is builder, but abstract factory provides the necessary elements as the basis of a pool game.

**The advantages and disadvantages of the design patterns used with respect to the code:**

The advantages of the code are from the isolated structure. The client who wants to get product does not need to know the detail of production. It means when the client wants to gain something what she only need to know is the basic detail of the final product. Besides, this isolation also can be beneficial. When changes of product take place, the whole structure does not need to be modified but the product itself. It decreases the job for switching for implementation. After getting the necessary elements for the finalization, using builder can be really helpful for this case. Because the pool game can be really complex with different number and exterior balls. The great flexibility builder provides can save a lot of work on constructing the game. For the whole structure, my code does combine these two design patterns in a concise way. It replaces the product in normal builder structure with the abstract factory.

The disadvantages can be from several perspectives. First, if I want to add more concrete classes in the code, it can be really complicated. That is to say that the concrete class should be derived from some exist abstract class or a new abstract class, and the factory should also be modified for compatibility. Second, once the variability of balls in the game is low, the builder could be too complex to construct the game. For example, all the balls share the same size and weight but color. In that case, builder is definitely not one of the optimal solutions. As an integrity, the combination of two design patterns is still not so optimal. The derivation between subclasses and base classes is not optimal either.

