**Factory:**

We used this design pattern number of reasons as follows, with single interface type created by factory it is much easier to test. Also it is easier to extend since the code has low coupling. When extending the features of the system existing features do not require additional modification thus ease of extendibility is achieved with factory. We decided that the complication added by this design pattern is worth the positive outcomes.

**Singleton:**

In this game several subsystems are working on different aspects of the game but the game as a whole is affected by each and every one of them. At this point the singleton design pattern is the most suitable one for this purpose. Because there must be only one game at a time and the different components must communicate with this only instance of the game. By having only one instance of the game there cannot be a confusion about for the subsystems that which game they are responding to. Also by implementing singleton the game holds the information throughout its lifespan.

**Abstract occurrence:**

Abstract occurrence pattern is the best way to represent similar objects (i.e. objects that are sharing multiple properties but differ in a slight way) without duplicating common information for every object. This pattern is used because it provides memory benefits by making less duplications for common points.