* **A description of the application and how it works**

I made an application that utilizes the iterator, singleton, and factory patterns to compare 2016 to other years to see if 2016 was really that bad. The user interface has two areas to display the array of facts related to each year. The user can switch between the years and also add facts that may have been missed. However, the 2016 fact array is static as the application is supposed to show the context of how bad 2016 has been. If the user tries to remove a fact, a message box shows that events cannot be removed from history.

**An explanation of how each of the three design patterns were used within the application.**

The majority of the design is based around the iterator pattern. This is the basis for the two arrays being simultaneously compared to one another. There is also a function to add new facts to the arrays. In an ideal world, there would be validation on the submitted facts to assert that they were true. The factory pattern allows the user to switch between years as each year is created within a class that inherits from a base class year. The singleton pattern is used to display the Remove error message.

* **An explanation of what a possible alternative would have been to using each of the design patterns**

For the iterator pattern, a strategy pattern could have been deployed to assert the behaviors of each of the iterator classes. It could have had a Next, Is Done, First, Current Item behaviors represented by the strategy pattern. The strategy pattern could have worked in place of the Factory pattern in that each of the years could have inherited from an interface that would have made them interchangeable. A proxy pattern could have also worked in place of the singleton pattern on the Remove Message. Could have used a proxy class to display the message and made it so that the message could not be edited.