My implementation of the EventList skiplist creates a skiplist with unique “EventNode” objects in a multi forward-linked list (i.e., an EventNode is only linked to EventNodes with a higher key). Each Node contains an ArrayList of “Event” Objects, which contain the actual information about what event occurred. With this implementation, I have implemented the typical insert and remove methods, along with a “findRange” method, which returns all the events occurring between two given dates, and the “findMostRecent” which returns all events occurring on the latest date (i.e. largest key) less than or equal to the provided date. Further, I have also implemented three helper methods: findRecentNodeHelper, which takes a year, and returns the node with the lowest level link to the provided year (which due to the implementation of the skiplist, is always the largest year in the list less than the parameter year), findNode, which returns either the node of the key provided or null, and doubleHead, which doubles the height of the Head node created when the list is initialized in order to ensure that it is always at least as tall as the tallest non-Head element in the list. Many of these methods look similar, as they all follow patterns very similar to the typical SkipList “find” method. However, as they all do slightly different things, I felt it was easier to implement them separately.

In order to handle duplicate keys for different events, I decided to create an EventNode class, which would have unique keys, and would also contain all Events with that duplicate keys. The collection of events was stored using Java’s ArrayList implementation, which provides an efficient toArray() method that we can use to return an array of all events in that year.