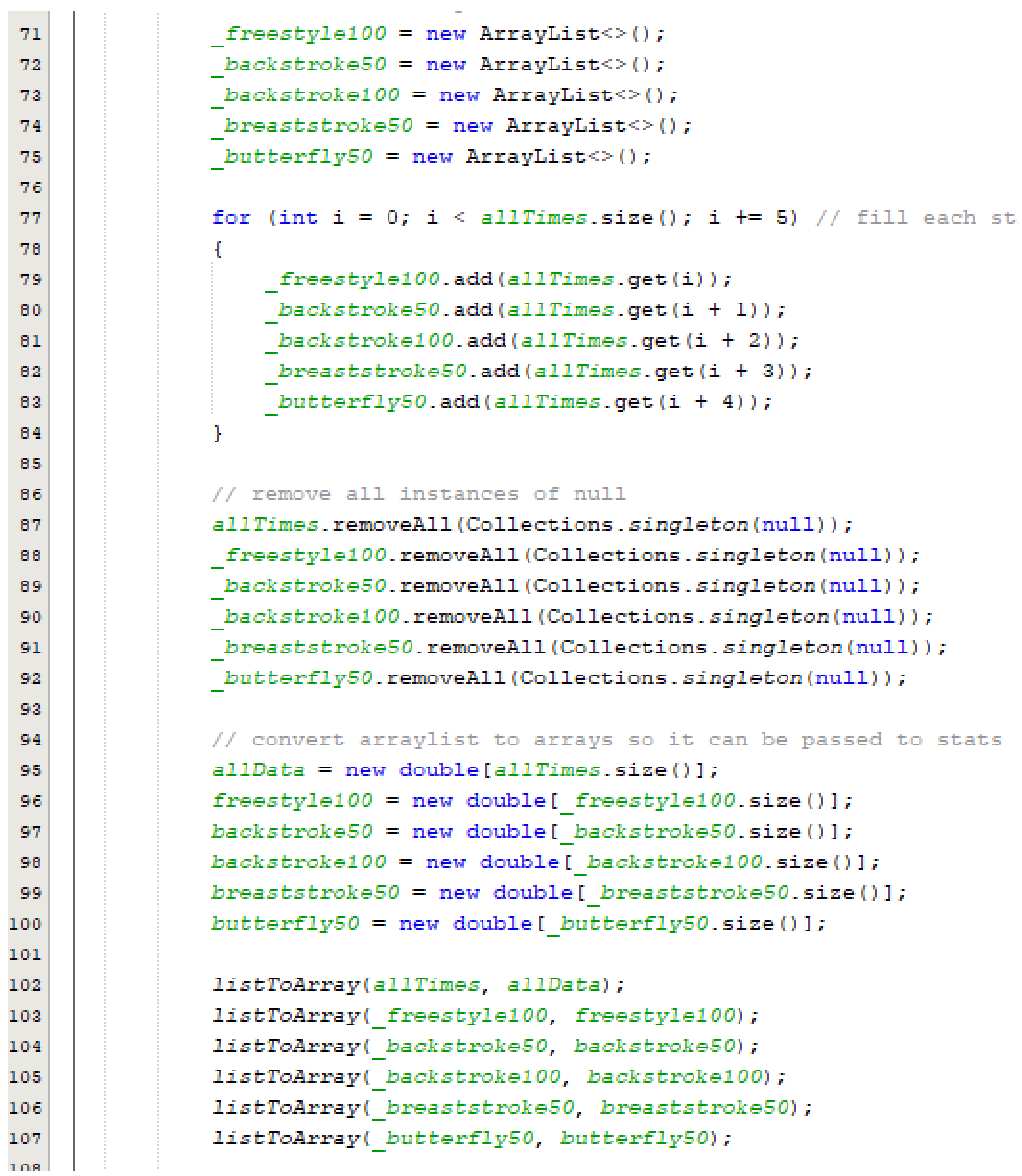
Criterion E: Evaluation

1. *The program should be able to keep track of the swimming times for each stroke.* **MET**
   The user is able to input swim time data and the program can perform statistical calculations on it.
2. *The program should be able to include a video of the swim, either embedded within the program if possible, or a link to Youtube if not.* **Not met**
   Unfortunately, I ran out of time on this project before I could implement this feature.
3. *The program should be able to analyze the times by listing the highest and lowest times.* **MET**
   I used a for-loop to iterate over an array of doubles in order to determine the minimum and maximum values.
4. *The program should be able to compute and display statistical calculations on the swim times (for example, mean, median, mode, and standard deviation).* **MET** I used various methods to compute the statistical formulas.
5. *The program should be able to compare the times against the standard times.* **MET**
   The program compares the user's average swim times for each stroke, as well as one overview against the national standards.
6. *The program should be able to keep track of schedules of future meets, practices, and other events.* **MET**
   The user can choose to store reminders for dates that are in the future.
7. *The user should be able to store profiles with the program. Each profile will have its own swim times associated with it.* **MET**
   The program allows the user to add, deleted, and edit profiles. Each profile contains its own swim times.

Recommendations for furthur development

The code I used to load data for the `StatsViewer` was too repetitive.



```java
_freestyle100 = new ArrayList<>();
_backstroke50 = new ArrayList<>();
_backstroke100 = new ArrayList<>();
_breaststroke50 = new ArrayList<>();
_butterfly50 = new ArrayList<>();

for (int i = 0; i < allTimes.size(); i += 5) // fill each st
{
    _freestyle100.add(allTimes.get(i));
    _backstroke50.add(allTimes.get(i + 1));
    _backstroke100.add(allTimes.get(i + 2));
    _breaststroke50.add(allTimes.get(i + 3));
    _butterfly50.add(allTimes.get(i + 4));
}

// remove all instances of null
allTimes.removeAll(Collections.singleton(null));
_freestyle100.removeAll(Collections.singleton(null));
_backstroke50.removeAll(Collections.singleton(null));
_backstroke100.removeAll(Collections.singleton(null));
_breaststroke50.removeAll(Collections.singleton(null));
_butterfly50.removeAll(Collections.singleton(null));

// convert arraylist to arrays so it can be passed to stats
allData = new double[allTimes.size()];
freestyle100 = new double[_freestyle100.size()];
backstroke50 = new double[_backstroke50.size()];
backstroke100 = new double[_backstroke100.size()];
breaststroke50 = new double[_breaststroke50.size()];
butterfly50 = new double[_butterfly50.size()];

listToArray(allTimes, allData);
listToArray(_freestyle100, freestyle100);
listToArray(_backstroke50, backstroke50);
listToArray(_backstroke100, backstroke100);
listToArray(_breaststroke50, breaststroke50);
listToArray(_butterfly50, butterfly50);
```

I used 12 total variables. One variable for each stroke as an array and array list, and two for the combined swim times; one array and one array list. Next time, I would probably find a more efficient way to convert array lists to arrays. Also, instead of five different variables, I would create an array or array list that would store each of the arrays and array lists in the five variables, then handle it with nested for-loops.
Also, since it was in the original success criteria in Criterion A, I would also add a function that allows the user to enter a link to a video of the swim in the `DateEditor` UI.
Other things I would add would include ways to compare swim times against swim times in other dates. For instance, I could add graphs that track the user's swim times

over time. Also, reminders should be profile-specific; that is a profile adding a reminder should not affect reminders in other profiles.

Advisor evaluation

The student has accomplished a very impressive project on his own. This project delivers the basic swim event management functions that meet the goals required by its original client. This project is intuitive and simple to use, and can be easily extended to more complex features because it is built on top of Java Swing. It is nice that the project uses common libraries such as calendar, etc.

Word Count : 517