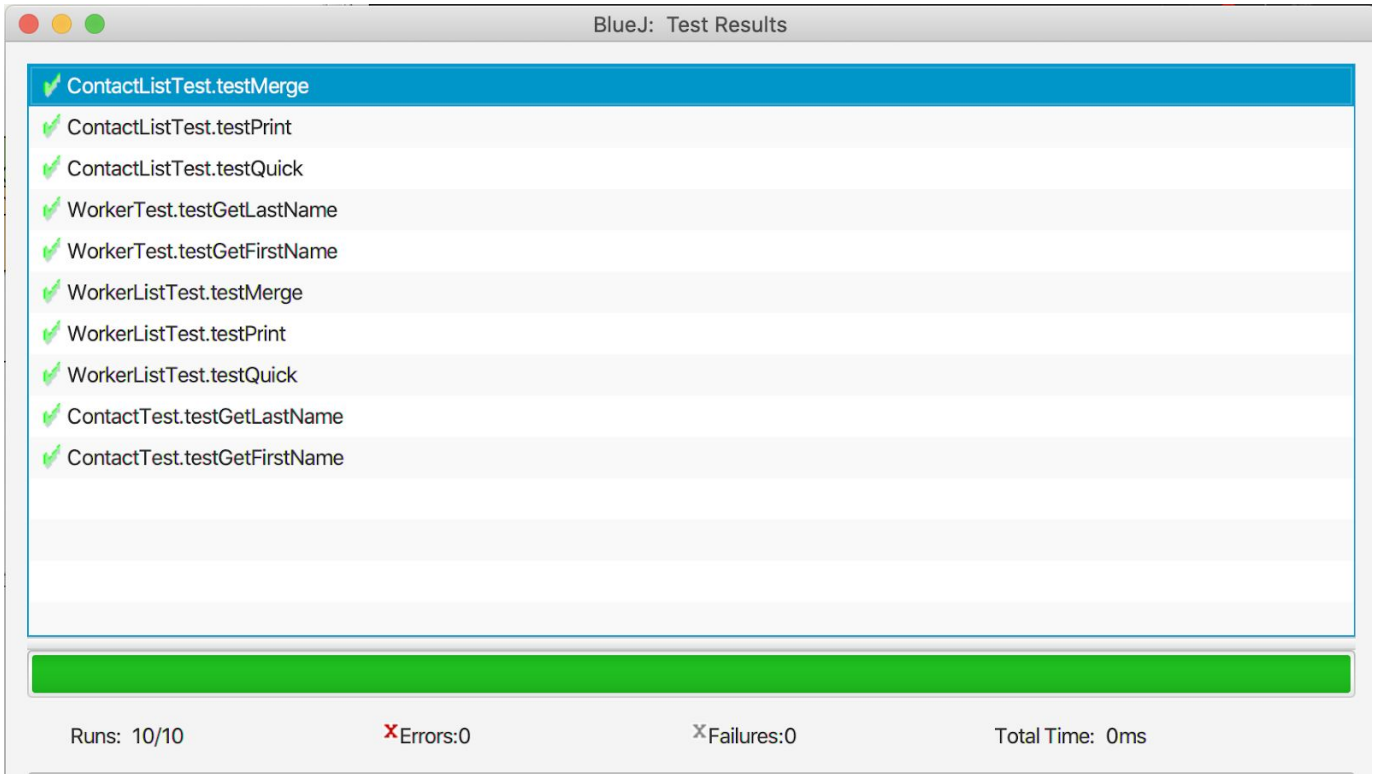
Dylan Maloy
CS150 Lab 6
Lab #6 write-up
10/12/19

Introduction:
The goal of this lab was to become more familiar with abstract classes, as well as creating an abstract implementation of the MergeSort and QuickSort sorting algorithms into these classes for different sets of data. There will be two different types of data, both deriving from the person class - Worker and Contact. Each data type will be sorted by separate parameters.

Unit Tests:



- BlueJ unit testing output

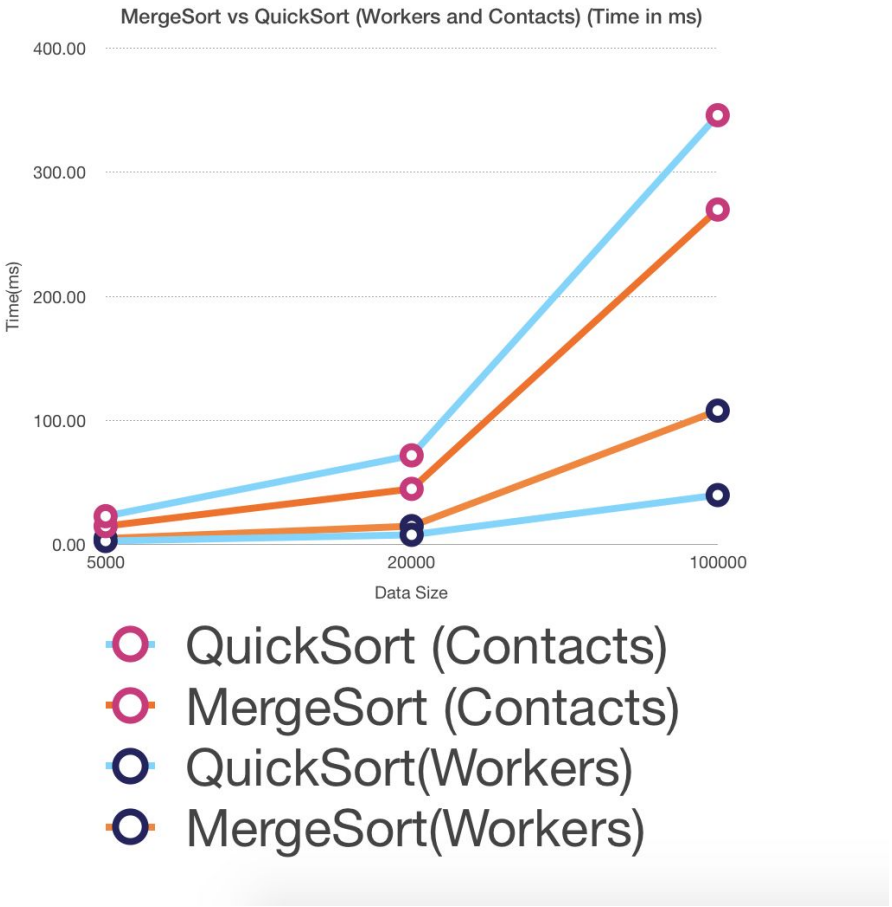
Required Output:

```
BlueJ: Terminal Window - lab6
 Time to quickSort contacts size=5000: 23ms
Time to mergeSort contacts size=5000: 15ms
Time to quickSort workers size=5000: 3ms
Time to mergeSort workers size=5000: 5ms
```

- Output received after sorting 5000 elements (averaged over 20 runs)

```
BlueJ: Terminal Window - lab6
 Time to quickSort contacts size=100000: 322ms
Time to mergeSort contacts size=100000: 247ms
Time to quickSort workers size=100000: 45ms
Time to mergeSort workers size=100000: 94ms
```

- Output received after sorting 100000 elements (averaged over 20 runs)



- Comparison of sort methods per unit time(ms)

Trouble Report:
This section is not applicable because all of my methods work as intended.

References:

Java Generics Example. Retrieved from
https://docs.oracle.com/javase/tutorial/java/generics/index.html

Java Comparable Interface. Retrived from
https://docs.oracle.com/javase/7/docs/api/java/lang/Comparable.html