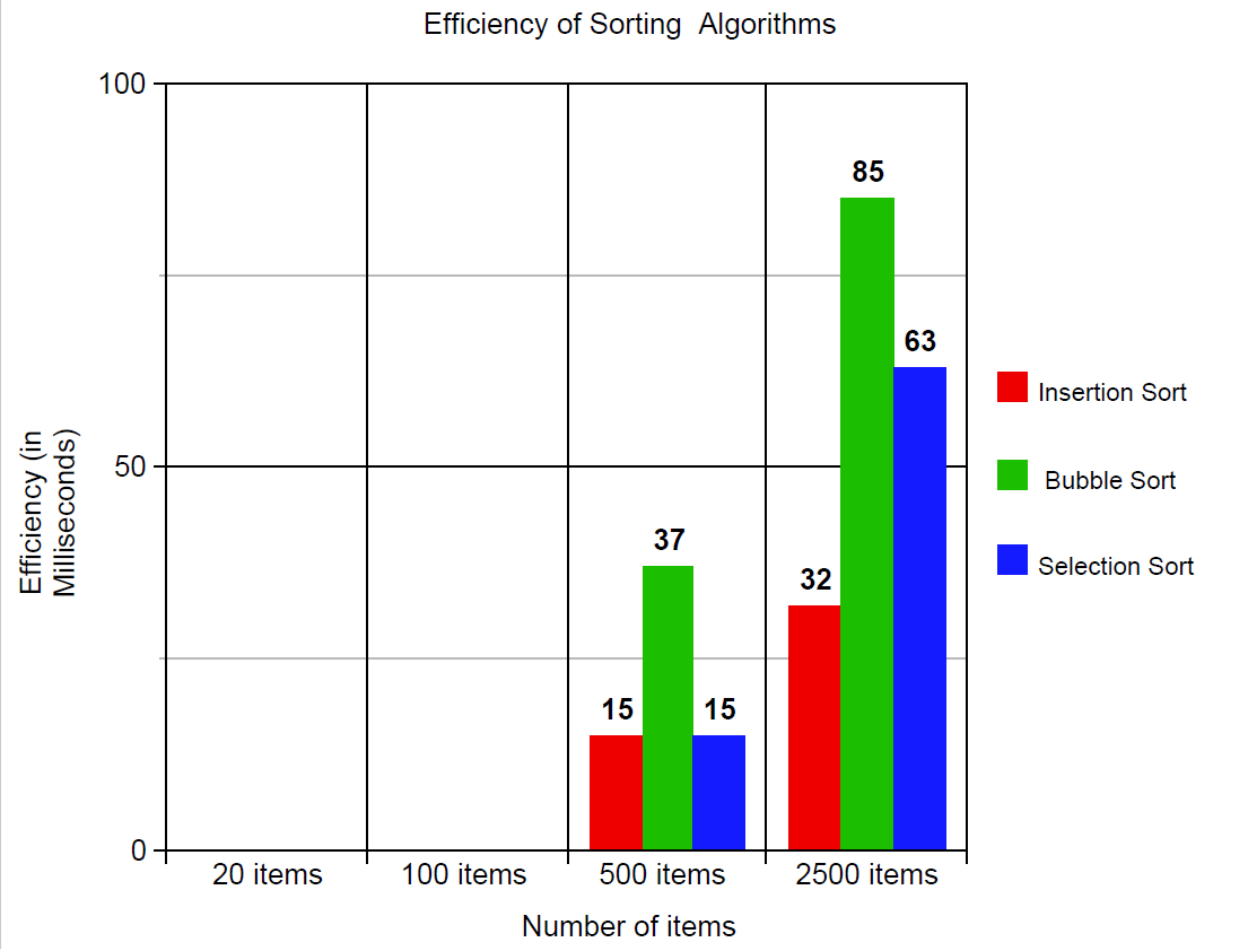
10/2/2016

Project 1: Simple Sorting

CS249

Stephen White

The first portion of this project required me to write out three basic algorithms to sort numeric data: Insertion sort, selection sort, and bubble sort. Each algorithm utilized a different set of steps to achieve the final sort, but ultimately each required a swap of some kind. Bubble sort would go through every item in a list, and "bubble" the biggest item to the top, and iterate through until the list was in order. This required a double-for-loop, which led to $O(N^2)$ efficiency, as it would take N steps to go through the outer loop and N steps to go through the inner loop. Selection sort, would loop through every item in the list, find the minimum, and swap it starting at index 0. This algorithm also ran at $O(N^2)$ efficiency, for the same reasons listed above. The final sorting algorithm to take care of was Insertion sort, whose job was to compare items one at a time, as an outer loop iterates through the list, and swap the items if one is bigger. This algorithm, however runs at O(N) efficiency as the inner loop is only responsible for comparing two items and not iterating through any other items in the list. A bar graph documenting the efficiency of the algorithms are depicted below



All of these sorting algorithms had one thing in common: they were generic, and could take in any object so long as it had a compareTo() method. But what if I wanted to sort data in a very specific way using an object that was not built into java? This led to the development of three separate classes which implemented the comparator interface: An Artist Comparator, a Chronological Comparator, and a Hot and New Comparator. The basis for these three comparator classes, was that they could each be utilized to sort data in the form of a Music Track in three separate ways. The Artist comparator would compare music track items in the following

order: artist name alphabetically, album name alphabetically, and track number ascending. This required the use of several if statements. In the event that two music tracks had the same artist, the next part of the track to be compared would be the album name, followed by the track number in ascending order. These comparators would return an integer, 1 if object one was greater than object two, -1 if object one was less than object two, and 0 if the items were the same. This is how the sorting algorithms would determine which item to swap. The chronological comparator compared items in the following order: year ascending, album name alphabetically, and track number ascending, in the same way as the artist comparator. The final comparator would compare items by song rating descending, year descending, artist name alphabetically, album name alphabetically, and track ascending.

With the completion of these comparators, it was time to go back into my sorting algorithms and re-write them so that they not only took in a generic list, but also a music track comparator. With this simple change, I would not only be able to sort through numeric data or string data, but I could parse through any number of music tracks and sort them to my heart's desire. After ensuring that all unit tests were passing, I wrote a main method to test all comparator types with all of my algorithms to negate any possibility of fault. Shown below are three separate comparators, which utilized three separate algorithms to display their sorted result. The first photo shows the bubble sort algorithm, which utilized the artist comparator.

```
1. Hail to Exile - Hail to Hump(1995) - Greese Monkeys *
2. Never say the Alamo - Hail to Hump(1995) - Greese Monkeys **
3. My Forever Love, Exile - Hail to Hump(1995) - Greese Monkeys ***
4. This is our Sleep - Hail to Hump(1995) - Greese Monkeys **
5. Never say Exile - Hail to Hump(1995) - Greese Monkeys ****
6. Dreaming of Exile - Hail to Hump(1995) - Greese Monkeys **
1. Dreaming of Freedom - Dreaming of Mama(2002) - Ninja Firefighters ****
2. My Forever Love, Freedom - Dreaming of Mama(2002) - Ninja Firefighters ***
3. Hail to Sleep - Dreaming of Mama(2002) - Ninja Firefighters ****
4. Never say the Big Cheese - Dreaming of Mama(2002) - Ninja Firefighters ****
5. Dreaming of the Big Cheese - Dreaming of Mama(2002) - Ninja Firefighters ***
6. My Forever Love, Sleep - Dreaming of Mama(2002) - Ninja Firefighters ***
1. This is our Sleep - Hail to Mama(2010) - Ninja Firefighters ****
2. Never say Freedom - Hail to Mama(2010) - Ninja Firefighters ****
3. You took my Hump - Hail to Mama(2010) - Ninja Firefighters ****
4. This is our Tonight - Hail to Mama(2010) - Ninja Firefighters **
5. My Forever Love, Exile - Hail to Mama(2010) - Ninja Firefighters ***
6. Give up Exile - Hail to Mama(2010) - Ninja Firefighters *
7. Dreaming of Sleep - Hail to Mama(2010) - Ninja Firefighters ***
1. Dreaming of Sleep - Never say Exile(2007) - Ninja Firefighters *
2. Remember me by Sleep - Never say Exile(2007) - Ninja Firefighters **
3. This is our the Big Cheese - Never say Exile(2007) - Ninja Firefighters ***
4. Never say Exile - Never say Exile(2007) - Ninja Firefighters ***
5. Give up Sleep - Never say Exile(2007) - Ninja Firefighters ***
6. Dreaming of Freedom - Never say Exile(2007) - Ninja Firefighters ***
1. Dreaming of Mama - Remember me by Tonight(1998) - Robot Fellows ***
2. Dreaming of the Alamo - Remember me by Tonight(1998) - Robot Fellows *
```

The next photograph demonstrates use of the chronological comparator and selection sorter:

```
1. My Forever Love, Hump - Give up Sleep(1988) - Saturday Morning Pirates *
2. Remember me by Sleep - Give up Sleep(1988) - Saturday Morning Pirates *
3. Hail to Tonight - Give up Sleep(1988) - Saturday Morning Pirates *
4. My Forever Love, Exile - Give up Sleep(1988) - Saturday Morning Pirates ****
5. Dreaming of Exile - Give up Sleep(1988) - Saturday Morning Pirates **
1. Hail to Exile - Hail to Hump(1995) - Greese Monkeys *
2. Never say the Alamo - Hail to Hump(1995) - Greese Monkeys **
3. My Forever Love, Exile - Hail to Hump(1995) - Greese Monkeys ***
4. This is our Sleep - Hail to Hump(1995) - Greese Monkeys **
5. Never say Exile - Hail to Hump(1995) - Greese Monkeys ****
6. Dreaming of Exile - Hail to Hump(1995) - Greese Monkeys **
1. Dreaming of Mama - Remember me by Tonight(1998) - Robot Fellows ***
2. Dreaming of the Alamo - Remember me by Tonight(1998) - Robot Fellows *
3. Hail to Mama - Remember me by Tonight(1998) - Robot Fellows *****
4. This is our Mama - Remember me by Tonight(1998) - Robot Fellows **
5. My Forever Love, Exile - Remember me by Tonight(1998) - Robot Fellows ***
6. You took my Hump - Remember me by Tonight(1998) - Robot Fellows ****
7. Never say Exile - Remember me by Tonight(1998) - Robot Fellows *
1. Dreaming of the Big Cheese - This is our Mama(1999) - Robot Fellows ***
2. You took my Tonight - This is our Mama(1999) - Robot Fellows ****
3. Never say Tonight - This is our Mama(1999) - Robot Fellows **
4. Never say the Alamo - This is our Mama(1999) - Robot Fellows **
5. Dreaming of the Alamo - This is our Mama(1999) - Robot Fellows ****
1. Dreaming of Freedom - Dreaming of Mama(2002) - Ninja Firefighters ****
2. My Forever Love, Freedom - Dreaming of Mama(2002) - Ninja Firefighters ***
3. Hail to Sleep - Dreaming of Mama(2002) - Ninja Firefighters ****
4. Never say the Big Cheese - Dreaming of Mama(2002) - Ninja Firefighters ****
```

The final photograph demonstrates the functionality of the insertion sorter with the hot and new comparator:

```
3. Hail to Mama - Remember me by Tonight(1998) - Robot Fellows *****
1. This is our Sleep - Hail to Mama(2010) - Ninja Firefighters ****
2. Never say Freedom - Hail to Mama(2010) - Ninja Firefighters ****
3. You took my Hump - Hail to Mama(2010) - Ninja Firefighters ****
1. Dreaming of Freedom - Dreaming of Mama(2002) - Ninja Firefighters ****
3. Hail to Sleep - Dreaming of Mama(2002) - Ninja Firefighters ****
4. Never say the Big Cheese - Dreaming of Mama(2002) - Ninja Firefighters ****
2. You took my Tonight - This is our Mama(1999) - Robot Fellows ****
5. Dreaming of the Alamo - This is our Mama(1999) - Robot Fellows ****
6. You took my Hump - Remember me by Tonight(1998) - Robot Fellows ****
5. Never say Exile - Hail to Hump(1995) - Greese Monkeys ****
4. My Forever Love, Exile - Give up Sleep(1988) - Saturday Morning Pirates ****
5. My Forever Love, Exile - Hail to Mama(2010) - Ninja Firefighters ***
7. Dreaming of Sleep - Hail to Mama(2010) - Ninja Firefighters ***
3. This is our the Big Cheese - Never say Exile(2007) - Ninja Firefighters ***
4. Never say Exile - Never say Exile(2007) - Ninja Firefighters ***
5. Give up Sleep - Never say Exile(2007) - Ninja Firefighters ***
6. Dreaming of Freedom - Never say Exile(2007) - Ninja Firefighters ***
2. My Forever Love, Freedom - Dreaming of Mama(2002) - Ninja Firefighters ***
5. Dreaming of the Big Cheese - Dreaming of Mama(2002) - Ninja Firefighters ***
6. My Forever Love, Sleep - Dreaming of Mama(2002) - Ninja Firefighters ***
1. Dreaming of the Big Cheese - This is our Mama(1999) - Robot Fellows ***
1. Dreaming of Mama - Remember me by Tonight(1998) - Robot Fellows ***
5. My Forever Love, Exile - Remember me by Tonight(1998) - Robot Fellows ***
3. My Forever Love, Exile - Hail to Hump(1995) - Greese Monkeys ***
4. This is our Tonight - Hail to Mama(2010) - Ninja Firefighters **
2. Remember me by Sleep - Never say Exile(2007) - Ninja Firefighters **
```

While coding this project, I ran into a numerous amount of problems that tested my patience, and intuition as a programmer. I began the process of planning and writing code a week before the project was due to give me ample time to complete the project, and even then, I had to fight to finish on time. My very first issue was running into problems with my basic sorting algorithms. I knew in my head what needed to be done, and yet, no matter what I wrote down in code form, the solution was just out of reach. After spending painstaking hours in the same rut, I utilized a google search and stack overflow to compare my solution to other who had written the sort I was having problems with, and quickly found out what I needed to fix. After spending two days thinking and writing out these algorithms, my next big hurdle came in the form of comparators. Having never utilized a comparator before, I spent a great deal of time understanding what they were used for, and how to implement them. My confusion was slightly cleared up through utilizing video tutorials on YouTube, but even then, I could not clearly comprehend how they worked in the context of this project. Through asking my instructor (Patrick Kelley) I very quickly understood how the comparators were going to be implemented, and got them to pass two out of the three unit tests. It seemed that I had one final, horrible issue to take care of. This came in the form of a small bit of code that hid below the surface, that was keeping my artist comparator from working properly. After asking for assistance, as well as searching online, I resorted to the one thing left that I could do: learn how to use a debugger. I stepped through my tests one line at a time for nearly an hour, before I saw a potential problem… The way I had

written my sorting algorithms only looked for the integers 1, -1, or 0 from a compareTo() method, but with the artist comparator, integers such as -16, or 5 were appearing. It hit me like a ton of bricks that I should not be looking for a returned integer of 1 or -1 to swap items given a comparator, but I should look for compareTo() integers that were greater than or equal to one, or less than or equal to -1. With this minute change, all of my problems disappeared, and the code worked like magic. This project taught me that determination is key in this discipline, and that you should exhaust every option you have, without giving up in the process. I may have been faced with a great challenge, but I one-hundred percent believe that I walked away from this project with more experience as a programmer than I had beforehand, as I learned how to write comparators, sorting algorithms, and even use a debugger when all else fails. I might have hated the process, but in the end I was glad I endured it.