Individual reflection:

@ author: Christina He

@ version: October 25, 2017

Through this week’s assignment I know more about the iterator in linked list. For example, I always had an error called “ConcurrentModificationException”, and after I google it I found that you cannot manipulate the iterator while you loop through it. That’s, if you have “ListIterator p= cardpile.listIterator()”, and you manipulate the size of the iterator before it finishes its loop from the first position to the last position, you would have this error. But we can use a while loop with the continuity condition so that when the loop stops, we are at the position we want.

Talking about the efficiency of these three sort methods, I found the result is out of my expectation. I thought at first that insertion sort or selection sort would be more efficient, for the codes of them are simpler. But the result showed up that merge sort is the most efficient one. As showed in the typescript, selection sort and insertion sort worked slowly at 40000 cards, and could hardly work at 80000 cards, while the times needed for merge sort are almost the same at 40000 cards and 80000 cards (0.239u-0.302u). I think that may be because every time we run the insertion sort and selection sort, they need to go through that 80000 cards again and again, but in merge sort, we just need to go through the first two cards at first and we would never need to go through 80000 cards.