1) Responsible for working with linkedList and doublelinked List. They had private node classes in them. I had to sort each car by destination using the compareTo method. I then shifted nodes to certain places to sort them in order. I displayed the list after all railcars were added. Then, I removed based off dest.

2) I had issues understanding how the previous variable worked in the linkclist

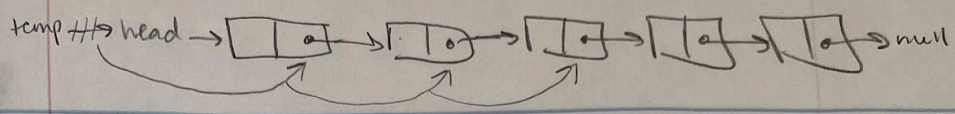
- Lessons I learned, were how to use linkedlist, doublelinkedlist, manipulate nodes, place nodes in front/behind, and how to better use a compareTo methods.

pseudocode: open and read file. Create one
main: railCar object for every railcar read. Add them
to a LinkedList. Add to end of doubleLinkedList.
exit while loop once end of file has no more values
to read.
display the linkedList
remove "Washington Dc" cars.
remove "Charleston" cars
remove "Orlando" cars as well as any cars with parrots
remove "West Palm Beach" cars
display double linkedList
display that list backwards

splay list
ter each
ll to
moveByDest()
ethod

removeByFreight method → if current railCar's freight being read matches the incoming destination, break bonds to nodes and reassign. The node before the node being removed will have it's next field point to the node after deleted node. The current will shift over.
→ if previous is null however, the head will transfer to next node and current will shift over to next node
→ if previous != null, shift over the current and previous to repeat the while loop and see if the next node equals the destination to be removed. and the location deletion has completed for that specific node.

Every time the current railCar and destination numOfRemoved increments and is returned by the method after no more railCars match the destination.



displayTrain

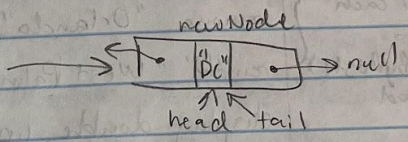
```
Node Temp = head;
while temp! = null
    print : temp.railCar.toString()
    temp = temp.next
```

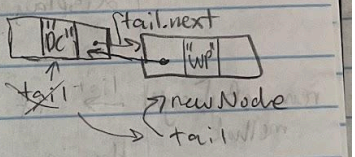
---

addToEnd(railcarToAdd)

incoming railcar

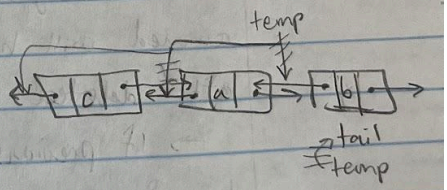
```
Node newNode = new Node(railCarToAdd)
    if (tail == null) {
        head = tail = newNode
    }
    else {
        tail.next = newNode;
        newNode.previous = tail
        tail = newNode
    }
```





displayBackwards

```
Node temp = tail;
while (temp! = null) {
    print : temp.railCar.toString()
    temp = temp.previous
```



loop ends

if = previous = = null

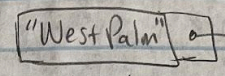
head = newNode;

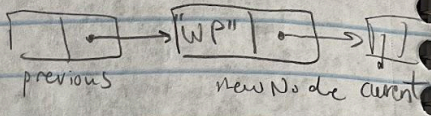
newNod.next = current;

else

previous.next = newNode;

newNode.next = current;





removeByDest method

int count=0;

Node current = head;

Node current = null

while (current! =null) {

incoming dest

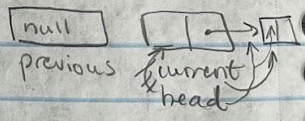
current dest.equals (destination)

count++;

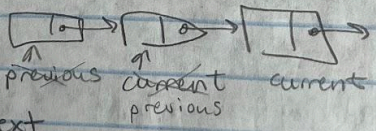
if prev. = null {

head = current.next;

current = head;



} else move prev over to where current is and current where current.next is

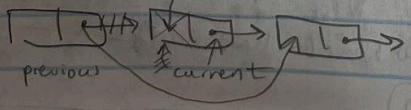


deleting the node:

previous.next = current.next

current = current.next;

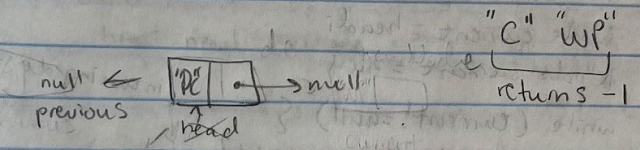
this data field is lost, nothing connected to it



Swannan Swan
CS 1450
Assignment 9

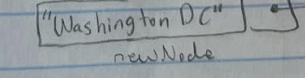
Train file:
"Washington DC"
"WestPalm"
"Charleston"
"Orlando"

- read through file
- make rail car objects
- myLinkedList.addByDestination(car)



if first if doesn't work: else
previous.next = newNode;
newNode.next = current;

addByDest method
Node current = head
Node previous = null;
boolean foundLocation = false
Node newNode = new Node(railCarToAdd);

if (head == null)
head = newNode


else enter while loop
toadd
if this railCar goes in front of other railcar ("Washington DC")
found = true