**CS 1150 Design Notebook Required Sections**

**Step 1: Problem Statement**

This assignment will create 2 classes that simulate a linked list. One linked list will be single, and the other is double. Linkedlist will carry trains that hold information from the given file. The code will simulate a train going to specific cities and trains with that city will be removed from the single linked list. Throught the simulated stop it will display the updated linked list and number of trains removed. After all stops are done, it will display the double linked list backwards

**Step 2: Understandings**

* What I Know:
  + Objects
  + Methods
  + Interfaces
* What I Don’t Know:
  + Creating My Own Linked List, mainly use of pointers

**Step 3: Pseudocode**

Main:

* Create 2 LinkedLists, one single and one double
* Create and read given file
* Store file info into 3 different variables
  + Need nextInt(), next(), nextLine()
* Create new RailCar object with the variables
* And new Rail Car to both linkedlists
  + Single linkedlist add to beginning
  + Double linkedlist adds to end
* Display Single LinkedList
* Simulate Stop 1, call removeByDestination() method In single LinkedList class
  + Removes Washington DC Trains
  + Display updated linkedlist
* Repeat For stop 2
  + Removes Charleston Trains
  + Display updated linkedlist
* Stop 3 call removeByDestination() and removeByFreight()
  + Removes Orlando trains and trains that carry parrots
  + Display updated linkedlist
* Stop 4, call removeByDestination()
  + Removes West Palm Beach Trains
  + Display updated linkedlist
* Display Double linkedlist using displayBackwards() Method in double linkedlist class

removeByFreight:

* Takes in wanted freight to remove
* Goes through linkedlist and removes equal freights
* Use placeholders to track position
* Loop until the end of linkedlist
  + Update position placeholders
* Return amount removed

**Step 4: Lesson Learned**

It took me a while to figure out the pointers to correctly add nodes to the linked list and how to remove them in the middle in the removeByDestination/freight method. The compareTo method also took a while to figure out because I thought my code for one of the last assignments would work by I did not release I was not checking all needed situations, making it so some objects would not get sorted correctly, the fix was to just check what it it’s the other way around.

**Step 5: Code**

**//package cs1450;**

**import java.io.File;**

**import java.io.FileNotFoundException;**

**import java.util.Scanner;**

**public class HofferIsaiahAssignment9 {**

**public static void main(String[] args) throws FileNotFoundException{**

**TrainLinkedList trainLinkedList = new TrainLinkedList();**

**DoubleLinkedList doubleLinkedList = new DoubleLinkedList();**

**//Getting And Reading File**

**File railCarFile = new File("RailCars.txt");**

**Scanner readRailCarFile = new Scanner(railCarFile);**

**while(readRailCarFile.hasNext()) {**

**//Getting File Info**

**int trainNumber = readRailCarFile.nextInt();**

**String trainCargo = readRailCarFile.next().trim();**

**String trainDestination = readRailCarFile.nextLine().trim(); //Do Not Want Space In Front**

**//Adding RailCar Objects To LinkedLists**

**trainLinkedList.addByDestination(new RailCar(trainNumber, trainCargo, trainDestination));**

**doubleLinkedList.addToEnd(new RailCar(trainNumber, trainCargo, trainDestination));**

**}//While**

**//Displaying Trains Leaving New York**

**//Pretext**

**System.out.printf("RailCar Freight\t Destination City\n"**

**+ "-------------------------------------------\n");**

**trainLinkedList.displayTrain();**

**//Display LinkedList After STOP 1**

**//Pretext**

**System.out.printf("\nStop 1: Train Arrives In Washington DC ");**

**int amountRemoved = trainLinkedList.removeByDestination("Washington DC"); //Removing And Getting Amount Removed**

**System.out.printf("\nRemoved %d Washington DC Rail Cars\n"**

**+ "-------------------------------------------\n", amountRemoved);**

**//Printing Updated LinkedList**

**trainLinkedList.displayTrain();**

**//Display LinkedList After STOP 2**

**//Pretext**

**System.out.printf("\nStop 2: Train Arrives In Charleston ");**

**amountRemoved = trainLinkedList.removeByDestination("Charleston"); //Removing And Getting Amount Removed**

**System.out.printf("\nRemoved %d Charleston Rail Cars\n"**

**+ "-------------------------------------------\n", amountRemoved);**

**//Printing Updated LinkedList**

**trainLinkedList.displayTrain();**

**//Display LinkedList After STOP 3**

**//Pretext**

**System.out.printf("\nStop 3: Train Arrives In Orlando ");**

**amountRemoved = trainLinkedList.removeByDestination("Orlando"); //Removing And Getting Amount Removed**

**System.out.printf("\nRemoved %d Orlando Rail Cars", amountRemoved);**

**//Remove Trains With Parrots**

**int amountRemovedCargo = trainLinkedList.removeByFreight("Parrots");**

**System.out.printf("\nRemoved %d Parrot Rail Cars With Parrots\n"**

**+ "-------------------------------------------\n", amountRemovedCargo);**

**//Printing Updated LinkedList**

**trainLinkedList.displayTrain();**

**//Display LinkedList After STOP 4**

**//Pretext**

**System.out.printf("\nStop 4: Train Arrives In ");**

**amountRemoved = trainLinkedList.removeByDestination("West Palm Beach"); //Removing And Getting Amount Removed**

**System.out.printf("\nRemoved %d West Palm Beach Rail Cars\n"**

**+ "-------------------------------------------\n", amountRemoved);**

**//Printing Updated LinkedList**

**trainLinkedList.displayTrain();**

**//Displaying Double LinkedList**

**//Pretext**

**System.out.printf("\nRail Cars In Double Linked List --- Printed Backwards\n"**

**+ "-------------------------------------------\n");**

**System.out.printf("RailCar Freight\t Destination City\n"**

**+ "-------------------------------------------\n");**

**doubleLinkedList.displayBackwards();**

**//Closing Scanner**

**readRailCarFile.close();**

**}//Main**

**}//Class**

**class RailCar implements Comparable<RailCar> {**

**//Private Data**

**private int number; //Rail Car's number**

**private String freight; //Type Of Freight In Rail Car**

**private String destination; //City Where Rail Car Is Heading**

**public RailCar(int number, String freight, String destination) {**

**//Initalizing Private Data**

**this.number = number;**

**this.freight = freight;**

**this.destination = destination;**

**}//RailCar Constructor**

**//Getter For Freight**

**public String getFreight() {**

**return freight;**

**}//GetFreight Method**

**//Getter For Destination**

**public String getDestination() {**

**return destination;**

**}//GetDestination Method**

**@Override**

**public String toString() {**

**return String.format("%3d\t%10s\t%s\n",number, freight, destination);**

**}//toString Method**

**@Override**

**public int compareTo(RailCar otherRailCar) {**

**//Setting Destination Strings To Variables So It Took Nicer**

**String destination = this.destination.toLowerCase();**

**String otherDestination = otherRailCar.getDestination().toLowerCase();**

**//Checking For West Palm Beach First, Highest Priority**

**if(destination.equals("west palm beach") && !otherDestination.equals("west palm beach")) { //Check If destination Is Front Of List**

**return -1;**

**}//If**

**else if(!destination.equals("west palm beach") && otherDestination.equals("west palm beach")) { //Checks If otherDestination is In Front**

**return 1;**

**}//Else If**

**//Orlando Is Second On List**

**if(destination.equals("orlando") && !otherDestination.equals("orlando")) {**

**return -1;**

**}//if**

**else if(!destination.equals("orlando") && otherDestination.equals("orlando")) {**

**return 1;**

**}//Else if**

**//Charleston Is Third**

**if(destination.equals("charleston") && !otherDestination.equals("charleston")) {**

**return -1;**

**}//If**

**else if(!destination.equals("charleston") && otherDestination.equals("charleston")) {**

**return 1;**

**}//Else If**

**//Washington is Last**

**if(destination.equals("washington dc") && !otherDestination.equals("washington dc")) {**

**return -1;**

**}//If**

**else if(!destination.equals("washington dc") && otherDestination.equals("washington dc")) {**

**return 1;**

**}//Else If**

**//If Object Destinations Are Equal**

**return 0;**

**}//compareTo Method**

**}//RailCar Class**

**class TrainLinkedList {**

**Node head;**

**public TrainLinkedList() {**

**this.head = null;**

**}//TrainLinkedList Constructor**

**public void addByDestination(RailCar railCarToAdd) {**

**Node newNode = new Node(railCarToAdd);**

**//No Need To Check If First Node**

**if(head == null) {**

**head = newNode;**

**}//If**

**else {**

**//Putting Node In First Spot**

**if(railCarToAdd.compareTo(head.railCar) < 0) {**

**newNode.next = head;**

**head = newNode;**

**}//If**

**//Putting Node In Middle Or Last Spot**

**else {**

**Node current = head;**

**while(current.next != null && railCarToAdd.compareTo(current.next.railCar) >= 0) {**

**current = current.next;**

**}//While**

**newNode.next = current.next;**

**current.next = newNode;**

**}**

**}//Else**

**}//addByDestination Method**

**public int removeByDestination(String destination) {**

**int amountRemoved = 0;**

**//PlaceHolders**

**Node current = head;**

**Node previous = null;**

**while(current != null) {**

**//Checking If Current Node Has Wanted Destination**

**if(current.railCar.getDestination().equalsIgnoreCase(destination)) {**

**//Move Head If Previous Is Not Initalized**

**if(previous == null) {**

**head = current.next;;**

**}**

**else { //Remove Middle Node**

**previous.next = current.next;**

**}**

**current = current.next;**

**amountRemoved++;**

**}//If**

**else { //If Not Equal Move Node Up By One**

**previous = current;**

**current = current.next;**

**}//Else**

**}//While**

**return amountRemoved;**

**}//removeByDestination Method**

**public int removeByFreight(String freight) {**

**int amountRemoved = 0;**

**//PlaceHolders**

**Node current = head;**

**Node previous = null;**

**while(current != null) {**

**//Checking If Current Node Has Wanted Destination**

**if(current.railCar.getFreight().equalsIgnoreCase(freight)) {**

**//Move Head If Previous Is Not Initalized**

**if(previous == null) {**

**head = current.next;;**

**}**

**else { //Remove Middle Node**

**previous.next = current.next;**

**}//Else**

**current = current.next; //Always Move Current Up By One**

**amountRemoved++; //Counter**

**}//If**

**else { //If Not Equal Move Node Up By One**

**previous = current;**

**current = current.next;**

**}//Else**

**}//While**

**return amountRemoved;**

**}//removeByFreight Method**

**//Displays Train Starting From The Head**

**public void displayTrain() {**

**Node current = head;**

**//Go Until No More Values**

**while(current != null) {**

**System.out.printf(current.railCar.toString());**

**current = current.next; //Update Postion**

**}//While**

**}//DisplayTrain Class**

**//Creating Nodes For Single Linked List**

**private class Node {**

**//Private Data**

**RailCar railCar;**

**Node next;**

**public Node(RailCar railCar) {**

**this.railCar = railCar;**

**this.next = null;**

**}//Node**

**}//Private Node Class --- Inner**

**}//TrainedLinkedList Class**

**class DoubleLinkedList {**

**//Private Data**

**Node head;**

**Node tail;**

**//Method To Add Values To End Of Linked List**

**public void addToEnd(RailCar railCarToAdd) {**

**Node newNode = new Node(railCarToAdd);**

**if(tail == null) {**

**head = tail = newNode;**

**}//If**

**else { //Changing Tail And Updating Pointers**

**tail.next = newNode;**

**newNode.previous = tail;**

**tail = newNode;**

**}//Else**

**}//addToEnd Methodd**

**//Display Double Linked List Satrting At The Tail**

**public void displayBackwards() {**

**Node current = tail;**

**//Go Until No More Values**

**while(current != null) {**

**System.out.printf(current.railCar.toString());**

**current = current.previous; //Update Position**

**}//While**

**}//DisplayBackWards Method**

**//Creating Nodes For Double Linked List**

**private class Node {**

**//Private Data**

**RailCar railCar;**

**Node previous;**

**Node next;**

**public Node(RailCar railCar) {**

**this.railCar = railCar;**

**previous = null;**

**next = null;**

**}//Node Constructor**

**}//Node**

**}//DoubleLinkedList**