CS 1150 Design Notebook Required Sections

Step 1: Problem Statement

This assignment will have me create a basic cargo terminal. It will have one loading dock, an array, with spaces for docks and a tarmac, another array, with spaces for stands. The size of these arrays will be given by the given files for the assignment. Then I will make a Cargo Terminal to hold these arrays. Fill the array with planes and semi-trucks by reading the details for each truck and plane from the files. Print the array after they are filled and will also display a tarmac status report. The report will show the planes and semi-trucks in order with their destination and capacity.

Step 2: Understandings

- What I Know:
  - Objects
  - File Reading
  - Polymorphism
- What I Don't Know:
  - Interfaces and Abstract Classes, still learning them

Step 3: Pseudocode

- Main:
  - Create a cargo terminal object to hold the loading dock array and tarmac array
    - Read Truck and Plane files to get array sizes
  - Fill loading dock with trucks read from truck file
    - Read truck info from file
    - Each truck is an object
    - Call addSemiTruck in Cargo terminal to add truck
  - Fill tarmac with planes from plane file
    - Read plane info from file
    - Each plane is an object
    - Call addCargoPlane in Cargo terminal to add plane
  - Display tarmac and cargo terminal
    - Use dispalyCargoTerminal() method in Cargo Terminal Class
  - Display Cargo Terminal Status report
    - Use printTermianlStatus() method in original class
    -
- printTermianlStatus:
  - create an arraylist and put all trucks from loading dock into it
    - Remove Null Values
  - Sort arraylist using Collections.sort(), sorted by destination
  - Display arraylist, use toString method in SemiTruck Class
  - Repeat for planes
  -

Step 4: Lesson Learned

I spent a while trying to find a way to let the compareTo method allow null values because it added them to the arrayLists, I could not find a way to do it and realized I could just not add the null values to the arrayLists, which was easier and worked. I also had some difficulty figuring out how to compare strings without just equality, I figured charAt was the best option.

Step 5: Code

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Scanner;

/*
Isaiah Hoffer
CS1450 (M/W)
2/26/25
Assignment 4
This assignment will use two files one for semi-trucks and another for cargo planes and create a cargo terminal that will create 2 array one for the truck and one for the plane and initalize them with the given
sizes in the files. It will then add trucks/plnaes to the array with the given object and index number. After it will displaying what docks/stands were loaded and will also display the truck and planes in order
using an overriden compareTo method from the Comparable Interface.
*/



public class HofferIsaiahAssignment4 {

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

		//Creating and Reading Files
		File truckFile = new File("FedExTrucks.txt");
		File planeFile = new File("FedExPlanes.txt");

		Scanner readTruckFile = new Scanner(truckFile);
		Scanner readPlaneFile = new Scanner(planeFile);

		//Size Of Truck And Plane Arrays
		final int TRUCK_ARRAY_SIZE = readTruckFile.nextInt();
		final int PLANE_ARRAY_SIZE = readPlaneFile.nextInt();

		//Creating Cargo Termianl Object
		CargoTerminal cargoTerminalObj = new
CargoTerminal(TRUCK_ARRAY_SIZE,PLANE_ARRAY_SIZE);

		//Adding Trucks To CargoTerminal
		while(readTruckFile.hasNext()) {

			//Getting Semi-Truck Info
			int truckDock = readTruckFile.nextInt();
			int truckNumber = readTruckFile.nextInt();
			String truckDestination = readTruckFile.nextLine();

			//Creating Semi-Truck
			SemiTruck newSemiTruck = new SemiTruck(truckNumber, truckDestination);
```

```
                cargoTerminalObj.addSemiTruck(truckDock,newSemiTruck);

            }//While

            //Adding Planes To CargoTerminal
            while(readPlaneFile.hasNext()){

                //Getting Plane Info
                int planeStand = readPlaneFile.nextInt();
                int planeNumber = readPlaneFile.nextInt();
                double planeCapacity = readPlaneFile.nextDouble();
                String planeCargoType = readPlaneFile.next();
                String planeDestination = readPlaneFile.nextLine();

                //Creating Plane
                CargoPlane newPlane = new CargoPlane(planeNumber, planeCapacity,
planeCargoType,planeDestination);

                cargoTerminalObj.addCargoPlane(planeStand,newPlane);

            }//While

            //Displaying Cargo Terminal
            cargoTerminalObj.displayCargoTerminal();

            //Displaying Terminal Status
            printTerminalStatus(cargoTerminalObj);

            //Closing Files
            readTruckFile.close();
            readPlaneFile.close();

        }//main

        public static void printTerminalStatus(CargoTerminal terminal) {

            //Loading Semi-Trucks From LoadingDock Array To New ArrayList
            ArrayList<SemiTruck> semiTruckArrayList = new ArrayList<>();

            //Loading Planes From Tarmac Array To New ArrayList
            ArrayList<CargoPlane> planeArrayList = new ArrayList<>();

            //Filling ArrayLists
            for(int i = 0; i < terminal.getNumberDocks(); i++) { //Fills semiTruckArrayList

                //Checking If Index Has An Object, Not Null
                if(terminal.getSemiTruck(i) != null) {

                    semiTruckArrayList.add(terminal.getSemiTruck(i));
                }//If
```

```
		}//For

		for(int i = 0; i < terminal.getNumberStands(); i++) { //Fills planeArrayList

			//Checking If Index Has An Object, Not Null
			if(terminal.getCargoPlane(i) != null) {

				planeArrayList.add(terminal.getCargoPlane(i));
			}//If

		}//For

		//Sorting ArrayLists
		Collections.sort(semiTruckArrayList);
		Collections.sort(planeArrayList);

		//Print Semi-Trucks From semiTruckArrayList
		//Pretext

	System.out.printf("\n\n************************************************************
***\n"
				+ "\t\tLoading Dock Status\t\t\n"
				+ "\t\t(By Destination City)\t\t\n"
				+
"************************************************************\n"
				+ "Truck Number\t\tDestination City\n"
				+ "-----------------------------------------------------\n");
		for(int i = 0; i < semiTruckArrayList.size(); i++) {

				System.out.println(semiTruckArrayList.get(i).toString());

		}//For


		//Repeating For Planes
				//Pretext

	System.out.printf("************************************************************\
n"
					+ "\t\t\tTarmac Status\t\t\n"
					+ "\t\t(Lowest To Highest Capacity)\t\t\n"
					+
"************************************************************\n"
					+ "Flight Number\tDeparting
To\tCargo\t\tCapacity(Lbs)\n"
					+ "-----------------------------------------------------\n");
				for(int i = 0; i < planeArrayList.size(); i++) {

					if(planeArrayList.get(i) != null) {
						System.out.println(planeArrayList.get(i).toString());
```

```
                                        }//If
                                    }//For


        }//printTerminalStatus Method

}//Class


class CargoTerminal {

        //Private Data
        private int numberDocks; // Number Of Docks For Trucks
        private int numberStands; // Number Of Stands For Planes

        private SemiTruck[] loadingDock; // Array To Holf Trucks
        private CargoPlane[] tarmac; // Array To Hold Planes

        public CargoTerminal(int numberDocks, int numberStands) {

                //Setting Data
                this.numberDocks = numberDocks;
                this.numberStands = numberStands;

                loadingDock = new SemiTruck[numberDocks];
                tarmac = new CargoPlane[numberStands];

        }//CargoTermianl Cunstructor

        //Getter For numberDocks
        public int getNumberDocks() {

                return numberDocks;
        }//getNumberDocks Method

        //Getter For numberStands
        public int getNumberStands() {

                return numberStands;
        }//getNumbrStands Method

        //Method to add SemiTrucks to loadingDock Array
        public void addSemiTruck (int dock, SemiTruck semiTruck) {

                loadingDock[dock] = semiTruck;
        }//addSemiTruck Method

        //Method To Add CargoPlane to tarmac Array
        public void addCargoPlane(int stand, CargoPlane plane) {
```

```
            tarmac[stand] = plane;
        }//addCargoPlane

        //Method To Get SemiTruck From loadingDock
        public SemiTruck getSemiTruck(int dock) {

            return loadingDock[dock];
        }//getSemiTruck Method

        //Method To Get CargoPlane From tarmac
        public CargoPlane getCargoPlane(int stand) {

            return tarmac[stand];
        }//getCargoPlane

        public void displayCargoTerminal() {

            //Displaying loadingDock Array
            //Pretext
            System.out.printf("Loading Semi-Trucks Into Cargo Terminal...\n\n");

            //Displaying Each Dock
            for(int i = 0; i < loadingDock.length; i++) {

                System.out.printf("Dock %d\t\t",i);

            }//For
            //Displays Semi-Trucks' Truck Number
            for(int i = 0; i < loadingDock.length; i++) {

                //Down A Line
                if(i == 0) {
                    System.out.println("");
                }//If

                //Checking If Array Has Truck
                if(loadingDock[i] != null) {
                    System.out.printf("%d\t\t",loadingDock[i].getTruckNumber());
                }//If

                else {
                    System.out.printf("%s\t\t","------");
                }

            }//For


            //Displaying tarmac Array
                        //Pretext
                        System.out.printf("\n\nLoading Planes into Into Cargo
Terminal...\n\n");
```

```
                                        //Displaying Each Dock
                                        for(int i = 0; i < tarmac.length; i++) {

                                                System.out.printf("Stand %d\t\t",i);

                                        }//For
                                        //Displays Semi-Trucks' Truck Number
                                        for(int i = 0; i < tarmac.length; i++) {

                                                //Down A Line
                                                if(i == 0) {
                                                        System.out.println();
                                                }//If

                                                //Checking If Array Has Plane
                                                if(tarmac[i] != null) {

        System.out.printf("%d\t\t",tarmac[i].getFlightNumber());
                                                }//If

                                                else {
                                                        System.out.printf("%s\t\t","------");
                                                }//Else

                                        }//For




        }//displayCargoTermianl Method

}//CargoTerminal Class


class CargoPlane implements Comparable<CargoPlane>{

        //Private Data
        private int flightNumber; // Planes Flight Number
        private double capacity; // Amount Plane Can Carry

        private String cargoType; // Type of Cargo the Plane Carries
        private String destinationCity; // Where The Plane is Heading


        public CargoPlane(int flightNumber, double capacity,
                        String cargoType, String destinationCity) {

                //Setting Data
                this.flightNumber = flightNumber;
                this.capacity = capacity;
```

```
            this.cargoType = cargoType;
            this.destinationCity = destinationCity;

        }//CargoPlane Constuctor

        //Getter For Flight Number
        public int getFlightNumber() {

            return flightNumber;
        }//getFlightNumber Method


        @Override
        public String toString() {

            return String.format("%4d\t\t%-15s\t%-10s\t%.2f",flightNumber,destinationCity,
                    cargoType, capacity);
        }//toString Method

        @Override
        public int compareTo(CargoPlane otherCargoPlane) {

            if(this.capacity > otherCargoPlane.capacity) {
                return 1;
            }
            else if(this.capacity < otherCargoPlane.capacity) {
                return -1;
            }
            else {
                return 0;
            }



        }//compareTo

}//CargoPlane Class


class SemiTruck implements Comparable<SemiTruck> {

        //Data Fields
        private int truckNumber; // Trucks Number
        private String destinationCity; // Trucks Destination City

        public SemiTruck(int truckNumber, String destinationCity) {

            this.truckNumber = truckNumber;
            this.destinationCity = destinationCity;
```

```
	}//SemiTruck Constructor


	//Getter For Truck Number
	public int getTruckNumber() {

		return truckNumber;
	}//getTruckMethod Method

	//Getter For Destination City
	public String getDestinationCity() {

		return destinationCity;
	}//getDestinationCity Method

	@Override
	public String toString() {

		return String.format("%d\t\t\t%s",truckNumber, destinationCity);
	}//toString Method

	@Override
	public int compareTo(SemiTruck otherSemiTruck) {

		//I'm Assuming City's Will Have At Least 2 Char Placements
		if(this.destinationCity.charAt(1) > otherSemiTruck.destinationCity.charAt(1)) {
			return 1;
		}//If

		else if(this.destinationCity.charAt(1) < otherSemiTruck.destinationCity.charAt(1)) {

			return -1;
		}//Else If

		else {

			if(this.destinationCity.charAt(2) > otherSemiTruck.destinationCity.charAt(2)) {


				return 1;
				}//If

			else if(this.destinationCity.charAt(2) <
otherSemiTruck.destinationCity.charAt(2)) {

				return -1;
			}//Else If

			else {

				return 0;
```

```
                    }//Else

                }//Else

            }//compareTo Method

}//SemiTruck Class
```