**CS 1150 Design Notebook Required Sections**

**Step 1: Problem Statement**

This assignment will take the code from assignment 4 and add another section to it. Assignment 4 created Plane and Trucks objects that hold cargo and have specific destinations. At the end of assignment 4, the planes and trucks were held in their own array, tarmac and loading dock. Assignment 7 will take the planes in the tarmac and split them into two different queues, one priority and the other normal. The priority queue will take only hold specific planes that hold specific cargo, and the basic queue will take the rest. At the end the code will take them out of the tarmac into a new queue called runway and from the runway queue will be removed and displaying to console to simulate takeoff.

**Step 2: Understandings**

* What I Know:
  + Objects
  + Queues
  + Methods
* What I Don’t Know:
  + Not too sure how to compare Strings in compareTo method

**Step 3: Pseudocode**

Main:

* Take code from assignment 4, but remove printTermainlStatus Method

**COPIED PSEUDOCODE FROM ASSINMENT 4 DESING NOTEBOOK**

* + - 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
* After assignment 4 code, create 3 new objects, one for AirTraffficController, Runway, and Taxiways
* Use method in AirTrafficController to move planes from tarmac to the new taxiway object variable
* Use CargoTerminal object variable from assignment 4 to check in tarmac is empty, it should be
* Use AirTrafficController again to move planes from the taxiway to the runway object variable
* Use AirTrafficController to remove planes in the runways and display the planes taking off in correct order
* Check if runway is empty, should be true

MovePlanesToTaxiways:

* Using a for loop to planes from tarmac to taxiways
  + Want to put planes in the runway in order, starting at 0
  + Check for null spots in tarmac
  + Check if object is priority or not, add to corresponding queue in taxiway
  + Display planes added using toString method

**Step 4: Lesson Learned**

It took me a while to figure out how to compare strings with a compareTo method, in the end I decided just checking what object had the highest priority, military and as long as the other object I’m comparing it to wasn’t the same, then then that object would be first and I did the same for the lowest priority, medical and if their our the same then no change happens. I did assuming hard code was allowed because we want the compareTo method to use these cargo types and the basis of priority leveling and it doesn’t make sense to not use them if we do not care about another cargo in the future

**Step 5: Code**

**//package cs1450;**

**import java.io.File;**

**import java.io.FileNotFoundException;**

**import java.util.LinkedList;**

**import java.util.PriorityQueue;**

**import java.util.Queue;**

**import java.util.Scanner;**

**/\***

**Isaiah Hoffer**

**CS1450 (M/W)**

**4/2/25**

**Assignment 7**

**This assignment with use code from assignment 4. This assignment will introduce me to use**

**queues, priority and nested, to be used as taxiways, one is prioirty and one is basic and the**

**priority queue will take in important cargo planes and the basic will take in all the other planes.**

**\*/**

**public class HofferIsaiahAssignment7 {**

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

**//Creating and Reading Files**

**File truckFile = new File("FedExTrucks7.txt");**

**File planeFile = new File("FedExPlanes7.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**

**CargoTerminal7 cargoTerminalObj = new CargoTerminal7(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**

**SemiTruck7 newSemiTruck = new SemiTruck7(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**

**CargoPlane7 newPlane = new CargoPlane7(planeNumber, planeCapacity, planeCargoType,planeDestination);**

**cargoTerminalObj.addCargoPlane(planeStand,newPlane);**

**}//While**

**//Displaying Cargo Terminal**

**cargoTerminalObj.displayCargoTerminal();**

**//Closing Files**

**readTruckFile.close();**

**readPlaneFile.close();**

**//Creating New Objects**

**AirTrafficController airTrafficControllerObj = new AirTrafficController();**

**Taxiways taxiwaysObj = new Taxiways();**

**Runway runwayObj = new Runway();**

**//Moving Planes From Tarmac To Taxiways**

**//Pretext**

**System.out.printf("\n\nControl Tower: Moving Planes From Cargo Terminal Tarmac To Taxiways: \n"**

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

**airTrafficControllerObj.movePlanesToTaxiways(cargoTerminalObj, taxiwaysObj);**

**//Displaying Empty Tarmac**

**//Pretext**

**System.out.printf("\n\nChecking If Tarmac Is Empty... \n\n");**

**cargoTerminalObj.displayCargoTerminal();**

**//Moving Planes From Tarmac To Taxiways**

**//Pretext**

**System.out.printf("\n\nControl Tower: Moving Planes From Taxiways To Runway: \n"**

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

**airTrafficControllerObj.movePlanesToRunway(taxiwaysObj, runwayObj);**

**//M0ving Planes From Runway To Takeoff**

**//Pretext**

**System.out.println("\n\nControl Tower: Ready For Takeoff! \n"**

**+ "------------------------------------------------------------------------");**

**airTrafficControllerObj.clearedForTakeOff(runwayObj);**

**//Checking If Runway Is Empty**

**System.out.printf("\n\nIs Runway Empty? %s",runwayObj.isEmpty());**

**}//main**

**}//Class**

**class CargoTerminal7 {**

**//Private Data**

**private int numberDocks; // Number Of Docks For Trucks**

**private int numberStands; // Number Of Stands For Planes**

**private SemiTruck7[] loadingDock; // Array To Holf Trucks**

**private CargoPlane7[] tarmac; // Array To Hold Planes**

**public CargoTerminal7(int numberDocks, int numberStands) {**

**//Setting Data**

**this.numberDocks = numberDocks;**

**this.numberStands = numberStands;**

**loadingDock = new SemiTruck7[numberDocks];**

**tarmac = new CargoPlane7[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, SemiTruck7 semiTruck) {**

**loadingDock[dock] = semiTruck;**

**}//addSemiTruck Method**

**//Method To Add CargoPlane to tarmac Array**

**public void addCargoPlane(int stand, CargoPlane7 plane) {**

**tarmac[stand] = plane;**

**}//addCargoPlane**

**//Method To Get SemiTruck From loadingDock**

**public SemiTruck7 getSemiTruck(int dock) {**

**return loadingDock[dock];**

**}//getSemiTruck Method**

**//Method To Get CargoPlane From tarmac**

**public CargoPlane7 getCargoPlane(int stand) {**

**return tarmac[stand];**

**}//getCargoPlane**

**//Removes CargoPlane With Given Index**

**public CargoPlane7 removeCargoPlane(int stand) {**

**CargoPlane7 plane = tarmac[stand]; //Store Plane In Temp Variable**

**tarmac[stand] = null; //Deletes Plane**

**return plane; //Returns Plane**

**}//removeCargoPlane Method**

**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 CargoPlane7 implements Comparable<CargoPlane7>{**

**//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 CargoPlane7(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",flightNumber,destinationCity,**

**cargoType);**

**}//toString Method**

**@Override**

**public int compareTo(CargoPlane7 otherCargoPlane) {**

**//Assuming Hard Code Is Allowed Here Too**

**if(this.cargoType.equalsIgnoreCase("military") &&**

**!otherCargoPlane.cargoType.equalsIgnoreCase("military")) {**

**return -1;**

**}//If**

**else if(this.cargoType.equalsIgnoreCase("medical") &&**

**!otherCargoPlane.cargoType.equalsIgnoreCase("medical")) {**

**return 1;**

**}//Else If**

**else {**

**return 0;**

**}**

**}//compareTo**

**//Checks To See If Plane Is A Priority**

**public boolean isPriority() {**

**//Checks What Type Of Cargo Plane Is Holding**

**switch(cargoType) {**

**case "Military","Perishables","Medical": return true;**

**default: return false;**

**}//switch**

**}//isPriority Method**

**public boolean isBasic() {**

**//Checks What Type Of Cargo Plane Is Holding**

**switch(cargoType) {**

**case "Military","Perishables","Medical": return false;**

**default: return true;**

**}//switch**

**}//isBasic**

**}//CargoPlane Class**

**class SemiTruck7 implements Comparable<SemiTruck7> {**

**//Data Fields**

**private int truckNumber; // Trucks Number**

**private String destinationCity; // Trucks Destination City**

**public SemiTruck7(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(SemiTruck7 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**

**class Taxiways {**

**//Private Data Types**

**private PriorityQueue<CargoPlane7> priorityTaxiway;**

**private Queue<CargoPlane7> basicTaxiway;**

**public Taxiways() {**

**//Initalizing Queues**

**this.priorityTaxiway = new PriorityQueue<>();**

**this.basicTaxiway = new LinkedList<>();**

**}//Taxiways Constructor**

**//Checks If PriorityTaxiway is empty, returns true or false**

**public boolean isPriorityTaxiwayEmpty() {**

**return priorityTaxiway.isEmpty();**

**}//isPriorityQueueEmpty Method**

**//Checks If BasicTaxiway is empty, returns true or false**

**public boolean isBasicTaxiwayEmpty() {**

**return basicTaxiway.isEmpty();**

**}//isBasicQueueEmpty Method**

**//Adds CargoPlane To PriorityTaxiway**

**public void addPlaneToPriorityTaxiway(CargoPlane7 plane) {**

**priorityTaxiway.offer(plane);**

**}//addPlaneToPriorityTaxiway Method**

**//Adds CargoPlane To BasicTaxiway**

**public void addPlaneToBasicTaxiway(CargoPlane7 plane) {**

**basicTaxiway.offer(plane);**

**}//addPlaneToBasicTaxiway Method**

**//Removes CargoPlane From PriorityTaxiway**

**public CargoPlane7 removePlaneFromPriorityTaxiway() {**

**return priorityTaxiway.remove();**

**}//removePlaneFromPriorityTaxiway Method**

**//Removes CargoPlane From PriorityTaxiway**

**public CargoPlane7 removePlaneFromBasicTaxiway() {**

**//Removes First Plane In Queue And Returns It**

**return basicTaxiway.remove();**

**}//removePlaneFromBasicTaxiway Method**

**}//Taxiways Class**

**class Runway {**

**//Private Data**

**Queue<CargoPlane7> runway;**

**public Runway() {**

**runway = new LinkedList<>();**

**}//Runway Constructor**

**//Returns True Or False If Queue Is Empty**

**public boolean isEmpty() {**

**return runway.isEmpty();**

**}//IsEmpty Method**

**//Adds Plane To Queue**

**public void add(CargoPlane7 plane) {**

**runway.offer(plane);**

**}//Add Method**

**public CargoPlane7 remove() {**

**return runway.remove();**

**}//Remove Method**

**}//Runway Class**

**//Moves Planes From Tarmac To Taxiways To Runway**

**class AirTrafficController {**

**public void movePlanesToTaxiways(CargoTerminal7 cargoTerminal, Taxiways taxiways) {**

**for(int i = 0; i < cargoTerminal.getNumberStands(); i++) {**

**if(cargoTerminal.getCargoPlane(i) != null) {**

**if(cargoTerminal.getCargoPlane(i).isPriority()) {**

**//Displaying Plane Moved To Taxiway**

**System.out.printf("\nMoved To Taxiway Priority Flight ");**

**System.out.printf(cargoTerminal.getCargoPlane(i).toString());**

**//Removing Plane From CargoTerminal And Added It To Taxiways**

**taxiways.addPlaneToPriorityTaxiway(cargoTerminal.getCargoPlane(i));**

**cargoTerminal.removeCargoPlane(i);**

**}//If**

**else {**

**//Displaying Plane Moved To Taxiway**

**System.out.printf("\nMoved To Taxiway Basic Flight ");**

**System.out.printf(cargoTerminal.getCargoPlane(i).toString());**

**//Removing Plane From CargoTerminal And Added It To Taxiways**

**taxiways.addPlaneToBasicTaxiway(cargoTerminal.getCargoPlane(i));**

**cargoTerminal.removeCargoPlane(i);**

**}//Else**

**}**

**}//For**

**}//movePlanesToTaxiways Method**

**public void movePlanesToRunway(Taxiways taxiways, Runway runway) {**

**while(!taxiways.isPriorityTaxiwayEmpty()) {**

**CargoPlane7 plane = taxiways.removePlaneFromPriorityTaxiway();**

**//Displaying Plane Moved To Runway**

**System.out.printf("\nMoved To Runway ");**

**System.out.printf(plane.toString());**

**runway.add(plane); //Adding Plane**

**}//While**

**while(!taxiways.isBasicTaxiwayEmpty()) {**

**CargoPlane7 plane = taxiways.removePlaneFromBasicTaxiway();**

**//Displaying Plane Moved To Runway**

**System.out.printf("\nMoved To Runway ");**

**System.out.printf(plane.toString());**

**runway.add(plane); //Adding Plane**

**}//While**

**}//movePlanesToRunway**

**public void clearedForTakeOff(Runway runway) {**

**while(!runway.isEmpty()) {**

**System.out.printf("\nCleared For Takeoff ");**

**System.out.printf(runway.remove().toString());**

**}//While**

**}//ClearedForTakeOff Method**

**}//AirTraffocController Class**