-----------------------------------------------------------------------------------------------------------------------------

**Design Notebook**

-----------------------------------------------------------------------------------------------------------------------------

**Max Santomauro**

**Step 1: Problem Statement**

This project involves the use of Java’s Comparable interface, working with a not completely full array, and ArrayList manipulation. This is exemplified by creating **a cargo terminal** program that consists of **a loading dock** for loading and unloading cargo with semi-trucks, a **tarmac** for unloading and loading cargo that accommodates aircrafts, and a **stand** area where a plane is parked for unloading and loading.

**Step 2: Design Sketch**

**A diagram of a method

AI-generated content may be incorrect.**

**Step 4: Pseudocode**

* **Within the Main method**
  + Create a file object that represents a file named “**FedExTrucks.txt”** and a file object that represents a file named “**FedExPlanes.txt**.”
    - Create a scanner for each file.
    - Assign the first int value that represents the size of an array from each file to their own integer variable.
    - Move to the next line for each file after reading the int variable.
  + Create a Cargo Terminal object that contains a loading dock and tarmac based on size values from the files.
    - The object takes in both sizes.
  + Read both files into the object.
    - Use **hasNext()** method within a while loop for both files.
    - Use the **addSemiTruck** and **addCorgoPlane** methods that will be created within the **CargoTerminal** class.
    - Initialize the variables from each file to related variables (e.g. for planes, initialized variables include, **stand**, **flightNumber**, **capacity,** **cargo**, and **destinationCity).**
  + Close both files.
  + Call the **displayCargoTerminal** method from the **CargoTerminal** class in order to display the loading dock and tarmac.
  + Call the printTerminalStatus method, run the cargoTerminal in it.
    - This will display the number and destination of each semi-truck in order of the destination.
    - This will display the number, destinations, cargo, and capacity of each plane in order of their capacity.
* **Outside the Main method within the same class**
  + Create a public static void method called **printTerminalStatus** for displaying semi-truck number and destination by sorted destination city and plane number, destination, cargo type, and capacity by sorted capacity.
    - Create an **ArrayList** for **SemiTruck** and another one for **CargoPlanes** that only take in the semi-truck numbers and plane numbers that are not null.
    - Use getters from **CargoTerminal** class within for loops and add only the semi-trucks and planes to the ArrayLists that don’t have null values for semi-truck number and plane number.
    - Call the **Collections.sort** method and run the variable for the SemiTruck ArrayList into it to sort based on semi-truck destination.
    - Print the semi-truck number and destination
      * Note: use “**size**” and “**get**” when doing this within a for loop.
    - Call the **Collections.sort** method and run the variable for the CargoPlane ArrayList into it to sort based on plane capacity.
    - Print the plane number, destination, cargo type, and capacity
      * Note: use “**size**” and “**get**” when doing this within a for loop.
* **Outside the class where Main method resides** 
  + Create a **CargoTerminal** class that represents a **cargo terminal** and its **loading dock** and **tarmac**.
    - Create private data field of integer variables that are named **numberDocks** and **numberStands** that determine the number of docks and stands from the dock and tarmac arrays.
    - Create two private data field object arrays of **SemiTruck** object called **loadingDock** and **CargoPlane** object called **tarmac** in which the cargo terminal contains a HAS-A relationship with these arrays.
    - Create a constructor called **CargoTerminal** that takes in **numberDocks** and **numberStands**.
      * Initialize instance variables from private data field
      * Allocates memory to arrays from private data field
    - Create getters for the number of docks and stands.
    - Create public void methods for semi-trucks and planes that adds the incoming objects to the corresponding arrays in specific dock or stand. Which are loadingDock and tarmac.
    - Create public object type methods that returns semi-truck and planes stored in the arrays in the dock or stand. Returns “null” if there is not a semi-truck or plane number.
    - Create a public void method for displaying the cargo terminals loading dock and tarmac.
      * Make sure it’s nicely formatted
      * Must print the dock number along with the semi-truck number and the stand number along with the plane number
      * Print semi-truck or plane number that return “null’ with dash lines
  + Create a **CorgoPlane class** that represents one cargo plane and implements **Comparable<CargoPlane>**
    - Create private data field with an integer variable called **flightNumber**, double integer variable called **capacity**, and two String variables called **cargoType**, and **destinationCity**.
    - Create a constructor called **CargoPlane** that takes in **flightNumber**, **capacity**, **cargoType**, and **destinationCity**
      * **I**nitialize instance variables from private data field.
    - Create getter for flightNumber only.
    - Create a toString method that returns a string of flight number, destination city, cargo type, and capacity
      * This method overrides the toString method in the Object class.
      * Use **String.format** method to return a nicely formatted string.
    - Create a **compareTo** method that overrides the CompareTo method in **Comparable**.
      * This compares two semi-trucks based on destination
      * Returns integer values 1, 0, or -1
      * Method never directly called, cut used by **Collections.sort**
  + Create a **SemiTruck class** that represents one cargo plane and implements **Comparable<SemiTruck>**
    - Repeat what is in the CargoPlane class for the SemiTruck class
      * Differences?
        + The private field includes only one integer variable called **truckNumber** and one string variable called **destinationCity**. Which are both ran into the constructor.
        + The getters are returning truckNumber and destinationCity
        + toString returns the string of truck number and destination city.