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

**Design Notebook**

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

**Max Santomauro**

**Step 1: Problem Statement**

Program purpose: Creating regular queues and priority queues and working with queues nested within classes.

This program involves the use of cargo terminal, cargo plane, semi-truck, taxiways, runway, and air traffic controller classes for the movement of cargo. THis movement of cargo will make use of regular queues, priority queues, and the nesting of queues within classes.

**Step 2: Design Sketch**

**A diagram of a method

AI-generated content may be incorrect.**

**Step 3: Pseudocode**

* **Within the Main method**
  + Create a file object that represents a file named “**FedExTrucks7.txt”** and a file object that represents a file named “**FedExPlanes7.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 **CargoTerminal7** 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.
  + Display cargo terminal's loading dock and tarmac by calling the **displayCargoTerminal** method in the **CargoTerminal7** class
  + Display moving plane cargo from tarmac to taxiways
  + Display tarmac being empty after moving to taxiways
  + Display moving plane cargo from taxiways to runway
  + Display the planes taking off
* **Outside the class where Main method resides** 
  + Create a **CargoTerminal7** 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 **CargoTerminal7** 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 called **displayCargoTerminal**.
      * 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 method called **removeCargoPlane** that takes in the number of stands and removes cargo planes stored in tarmac at specific stand.
      * Returns null if there’s no cargo plane in stand location.
      * Returns cargo plane if there is one with the location set to null
  + 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, and cargo type.
      * 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 based on cargo type in this order: military, perishables, and medical respectively
      * Returns integer values 1, 0, or -1 based in indices
      * Method never directly called, but used by **Collections.sort**
    - Create a **isPriority** method that returns true if the plane’s cargo type is **military**, **parishable**, or **medical**.
    - Create a **isBasic** method that returns true if the plane’s cargo type is not **military**, **parishable**, or **medical**.
  + 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.
  + Create a **Taxiways class** that represents taxiways at the cargo terminal and which queues will be used to represent the taxiways
    - For the private data field, create two queue for storing queues. One queue called **priorityTaxiway** with cargo type military, perishables, or medical and another queue called **basicTaxiway** with cargo type that is not military, perishables, or medical.
    - Create Taxiways constructor and allocating memory to both queues
    - Create a method called **isPriorityTaxiwayEmpty** that returns true if priority taxiway is empty and false if not empty
    - Create method called **addPlaneToPriorityTaxiway** that offers plane to taxiway
    - Create a method called **removePlaneFromPriororityTaxiway** that removes plane from priority taxiway and returns the cargo plane
    - Create the same ***isEmpty***, ***add***, ***remove*** methods for the **basicTaxiway** queue called **isBasicTaxiwayEmpty**, **addPlaneToBasicTaxiway**, and **removePlaneFromBasicTaxiway**
  + **C**reate a **Runway class** that represents the runway at the cargo terminal and which queues will be used to represent the runway
    - **F**or the private data field, create a queue called runway for storing queues
    - Create Runway constructor and allocating memory to both queues
    - Create the same type of add, remove, and isEmpty methods as was done in the Taxiways class for priority and basic queue.
      * These methods are called **isRunway, addPlaneToRunway**, and **removePlaneFromRunway**
  + Create an **AirTrafficController** class for moving cargo planes from the tarmac to taxiways to runway.
    - This class has no private field variables or a constructor method
    - Create a method called **movePlanesToTaxiways** that takes in CargoTerminal7 and Taxiways objects
      * Use a for loop and if/else conditions to get planes at the stand location
      * Create cargo plane object that removes cargo planes with the **removeCargoPlane** method from the **CargoTerminal7** class
      * Use the **add** method created for the priority and basic queues to add to taxiways
      * Print out result using the toString method in CargoPlane7
    - Create a method called **movePlanesToRunway** that takes in Taxiways and Runway objects.
      * Use a while loop for priority queue and a while loop for basic queue.
      * Use the **remove** and **add** methods for both queues when moving cargo planes from taxiways to runway
      * Print out result using the toString method in CargoPlane7
    - Create a method called **clearedForTakoff** that takes in Runway
      * Use while loop whose condition uses the **isEmtpy()** method from the Runway class
      * Remove cargo planes from the runway and print out a message for each cargo plane along with the flight number, destination city, and cargo type