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

**Design Notebook**

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

**Max Santomauro**

**Step 1: Problem Statement**

The purpose of this program is the implementation of a **single-linked list** and **double-linked list** with methods to add, remove, and display methods. The program is based on rail cars being assembled to a train in New York and has a final destination at Palm Beach, Florida. All the rail cars that match intermediate destinations are removed and are completely removed at Palm Beach. The intermediate destinations are Washington DC, Charleston, and Orlando. The train carts are carrying crazy parrots.

**Step 2: Design Sketch**

**A diagram of a train

AI-generated content may be incorrect.**

**Step 3: Pseudocode**

* **Within the Main method**
  + Creates a new File object and scanner for the file named **Railcars.txt**
  + Create instances of single linked list class & double linked list class
  + Fill the single and double link lists with the variables from the file using a while loop and **hasNextLine()** method within the condition.
    - Create instance of RailCar object
    - Add rail car object to single linked list using **addByDestination** method
    - Add rail car object to end of double linked list using addToEnd method
  + Make sure to close the Railcars.txt file
  + Display rail car's number, freight, and destination city based on priority using the **displayTrain()** method from the **TrainLinkedList** class
    - Note: This will be displayed based on the destination order due to the **compareTo** method.
  + Display crates after removal at each destination
    - Create a string array variable prioritizing destination.
    - Use this string array variable within for loop and if/else conditions
      * Assign to a created integer variable the **removeByDestination** method in the TrainLinkedList class and have it take in an element at index i of the string array variable.
      * Do if statement that displays rail cars after removal at each destination excluding the rail cars that have parrots on them
      * Do else statement that removes rail cars after removal based on destination and if they have parrots on them
  + Display rail car's number, freight, and destination city backwards using the **displayTrain()** method from the **DoubleLinkedList** class
* **Outside the class where Main method resides** 
  + Create a class called **RailCar** that represents one rail car and implements Comparable
    - Create in private data field int number, String freight, and String destination variables
    - Create constructor that takes in all three private data field variables and initialize them
    - Create getters for freight and destination variables
    - Create a **toString** method that returns a string of the rail car number, freight, and destination
    - Create a **compartTo** method that return the indication of whether the rail car should be in front of or behind the rail car that it's compared to or if they have the same destination
      * Create a variable to store the current rail car in and another variable to store the incoming rail car in.
      * Do if/else conditions that return 1, 0, or -1 based on destination priority of **Washington DC**, **Charleston**, **Orlando**, and **West Palm Beach** respectively.
  + Create a class called **TrianLinkedList** that uses single linked list and nodes to manipulate rail cars
    - Create in private data field a Node head variable that references to the first node in linked list
    - Create method called **addByDestination** that takes in an object that's an instance of RailCar and adds rail cars based on priority order.
      * Make use of the compareTo method and the private Node class for finding correct location to insert new rail car
      * Use if/else condition and a while loop with the else condition to determine if the list is empty or if the new rail car should come before the head node
    - Create method called **removeByDestination** to remove all rail cars from the linked list with the incoming specific destination
      * Create an integer count variable for return the number of rail cars removed.
      * Within while loops, use the head variable along with the getters.
    - Create method called **removeByFreight** to remove all rail cars from the linked list with the incoming specific freight
      * Create an integer count variable for return the number of rail cars removed.
      * Within while loops, use the head variable along with the getters.
    - Create void method called **displayTrain** for displaying rail cars from the head of the linked list to the end
      * Assign head to a variable and use this variable along with the **toString** method for displaying the rail cars
    - Create a private static node class that represents one node in the linked list
      * In the private data field, create a rail car node called **data** for storing a rail car in this node and create a **next** Node variable called **next.**
      * Create constructor that takes in rail cars and initialize data to itself and next to null.
  + Create a class called **DoubleLinkedList** that uses double linked list and nodes to manipulate rail car
    - In the private data field, Create a node called **head** as a reference to the 1st node in linked list and a node called **tail** as a reference to the last node of the linked list
    - Create a method called **addToEnd** that takes in rail cars and adds the rail cars to the end of the double linked list
      * Create an object instance of Node called **newNode** that takes in rail cars
      * Create if/else conditions that sets both head and tail to the new node
    - Create a void method called **displayBackwards** for displaying the rail cars from tail to head (backwards)
      * Assign tail to a variable and use this variable along with the **toString** method for displaying the rail cars
    - Create a private static node class that represents one node in the linked list
      * In the private data field, create a rail car node called **data** for storing a rail car in this node, a nextNode variable called **next**, and a previous Node called **previous.**
      * Create constructor that takes in rail cars and initialize data to itself, next to null, and previous to null.