Program Description

This program manages aircraft for a hypothetical Delta Airlines Company. It has four classes:

1. **Aircraft Class-** Representing a single Aircraft.
2. **Fleet Class-** Representing the entire fleet of aircrafts
3. **TextMenu Class-** This class will store menu options
4. **FleetDriver Class-** The main class which provides abstraction, and also an execution point for the program

The program will read aircraft data from a file(datafleet.txt), store the data in an array, query this data, allow you to add or remove an aircraft and perform other operations as given in menu options.

Aircraft Class:

Instance Variables: aircraftName, regNumber, manufacturer, maxRange, crewSize, yearPutInService, maxServiceWeight, numPassengers, lastMaintenanceDate, lastMaintenanceMiles.

- Methods:
  - **Two constructors,** the default, and one that takes all instance variables
  - **needsMaintenance()-** this method will determine if an aircraft is in need of maintenance. It returns a Boolean value based on the following criteria. An aircraft requires maintenance on the following schedule:
    - Last maintenance date is 3 months or more
    - Last maintenance mileage is 150,000 air miles or more
  - **shouldRetire()-** this method returns a Boolean value that determines whether an aircraft is ready to be retired from service, based on the following criteria:
    - More than 20 years past date put into service
    - More than 2,000,000 air miles
  - **toString()-** a format to output data

Fleet class:

- Instance Variables:
  - An array that stores Aircraft that represents Delta Airlines entire fleet
  - A variable that represents the count for the number of aircraft in the fleet
- Methods:
  - **Constructor-** One constructor that instantiates the array and sets the count to zero
  - **readFile()-** This method accepts a string that represents the name of the file to be read. It will then read the file. Once the data is read, it creates an aircraft and then passes the vehicle to the addAircraft method without any duplication of records.
  - **writeFile()-** This method accepts a string that represents the name of the file to be written to and then writes the contents of the array to a file. This method also calls a sort method to sort the array before writing to it.
  - **sortArray()-** This method returns a sorted array. The array is sorted by registration number.
  - **addAircraft()-** This method accepts an aircraft and adds it to the fleet(the array) only if it is not already in the list.
  - **displayFleet()-** This method outputs all aircraft that are in the fleet.
  - **displayMaintenanceList()-** This method outputs all aircraft in the array that currently require maintenance, either by miles or date.

- **displayNextMaintenanceList()** -This method outputs all aircraft in the fleet by name, manufacturer, model, next maintenance date, and next maintenance miles.
- **updateMiles()-** This method accepts a regNumber and miles and updates the total miles for the designated aircraft based on regNumber
- **updateMaintenance()-** This method accepts a regNumber and changes the lastMaintenance date to the current date and resets last maintenance mileage.
- **displayAircraft()-** This method accepts a regNumber and searches the array for an aircraft that matches that regNumber and displays the information about that vehicle.
- **removeAircraft()-** This method accepts a regNumber and searches the array for that aircraft. For each item that doesn't match, it will copy the contents to the new array. It will not copy the contents of the aircraft that is found. In other words, all aircraft in the array will be moved up a position to fill in the gap in the array.

TextMenu class:

- Instance Variables:
  - An array that stores all of the menu item options
- Methods:
  - **Constructor**- This accepts an array of menu items, displaying only the text options, not the number of the option.
  - **displayMenu()-** This method prints the contents of the text menu array. The output includes the number choices for each item.
  - **getChoice()-** This method will call the displayMenu and then allow the user to input their response. The response will be sent to the validateChoice() method and then if the choice is valid (not -1), then It will return the integer value of the user's choice. It will re-ask for input if the choice was not valid.
  - **validateChoice()-** This method accepts a string that represents the users choice. This uses try/catch to convert the string choice to an integer and return -1 if the user does not enter a number or if choice is outside the acceptable range.

FleetDriver class:

- Instance variables:
  - Scanner instance
- main method:
  - Contains three items:
    - Instantiate an instance of the Fleet class
    - Welcome message
    - Call the setup method
- **setup()-** This method will load the menu items into the array, instantiate an instance of the TextMenu class, call the getChoice method from the TextMenu class and then process the user's choice. This method processes each option from the menu by calling the appropriate methods. It repeatedly accepts choices until the user chooses to quit.
- getAircraftInfo()- This method accepts a numeric representation of the aircraft type and converts the numeric vehicle type to a character type. It then prompts for and accepts all the other information for an aircraft and then instantiates an Aircraft object and then returns that