CS 2336 – PROJECT 3 – Tie Fighter Patrols Strike Back (Maintenance Project 2)

Pseudocode Due: 2/24 by 11:59 PM
Project Due: 3/19 by 11:59 PM

This is the second of three maintenance projects. Changes from project 1 are colored blue.

KEY ITEMS: Key items are marked in red. Failure to include or complete key items will incur additional deductions as noted beside the item.

Submission and Grading:

- **The file containing `main` must be named Main.java. (-5 points)**
- **The project files must be in a package named `TieFighter`. (-5 points)**
- All project deliverables are to be submitted in eLearning.
- Zip the contents of the `src` directory into a single zipped file
  - Make sure the zipped file has a .zip extension (not .tar, .rar, .7z, etc.) (-5 points)
  - Please review the submission testing information in eLearning on the Course Homepage
- The pseudocode should be submitted as a Word or PDF document and is not accepted late.
- Projects submitted after the due date are subject to the late penalties described in the syllabus.
- Programs must compile and run with JDK 8.
- Each submitted program will be graded with the rubric provided in eLearning as well as a set of test cases. These test cases will be posted in eLearning after the due date. Each student is responsible for developing sample test cases to ensure the program works as expected.
- **Type your name and netID in the comments at the top of all files submitted.** (-5 points)

Objectives:

- Implement a linked list using classes in Java
- Implement input validation
- Perform sorting and search algorithms on a linked list

Problem: Darth Vader wants to check that his TIE fighter pilots are patrolling adequately-sized regions of the galaxy. He has files of data that contain the patrol coordinates for each of his pilots. With the fear of being force choked, you have agreed to write a program that analyzes the data and determines the size of the area patrolled by the pilots.

Pseudocode Details:

- The sorting method of the linked list class
  - What parameters are expected
  - Describe **logically** how you will make the sort universal to sort on name or area
  - Describe the logic for the sorting algorithm you plan to implement
    - Remember that you can treat the linked list like an array logically
- Main
  - Describe the logic to use the linked list to solve the problem
  - Describe the logic you will use to validate data from both of the input files

Class Details:

- Payload class
  - Must be comparable
  - Attributes
    - Pilot name
    - Patrol area
    - Boolean flag
      - Tracks type of comparison
  - Methods
    - Constructor
      - Pass in pilot's name
    - Accessors
    - Mutators
- Node class
  - Must be generic
  - Attributes
    - Generic variable to hold object
      - Will be used to hold payload object in the program
    - Next pointer
    - Prev pointer
  - Methods
    - Constructor
      - Pass in generic object for payload
    - Accessors for pointers
    - Mutators for pointers
- Linked List
  - Attributes
    - Head pointer (points to beginning of list)
    - Tail pointer (points to end of list)
  - Overloaded constructor
    - Takes node and assigns head and tail to point at the node passed in
  - Accessor
  - Mutator
  - toString (overridden)
    - create a string that lists each pilot and their respective area separated on individual lines
      - <name> tab tab <area> newline (please use \r\n for newline)
    - **EXTRA CREDIT: Implement this recursively (5 points)**
  - Sort
    - Sort the linked list in ascending or descending order by name(alphabetical) or area(numerical)
    - Only 1 sort function
      - You must create a single function that can sort in ascending or descending order and can sort either names or areas

        - You may implement any sorting algorithm you prefer
            - **EXTRA CREDIT: Implement a merge sort on the linked list (10 points)**
                - In the comments at the top of main, list the following:
                    - Line numbers in linked list class for implementation
                    - Line number and filename of merge sort function call
        - The sort implementation must rearrange the nodes (-10 points)
            - Do not swap node contents

Details:

- The area of the shape can be calculated with the following formula:

$$\frac{1}{2}\left|\sum_{i=0}^{n-2}(x_{i+1}+x_i)(y_{i+1}-y_i)\right|$$

- Store each pilot's information (name and area) in a node in the linked list
- Add each new node to the end of the linked list
- The number of pilots in each file is unknown
- The number of coordinates for each pilot is unknown
- Comment your code generously. Refer to the grading rubric for more details
- Use as few variables as possible. Don't waste memory holding a calculation just so you can print it out one time.

User Interface: There will be no user interface for this program. All I/O will be performed with files

Input:

- **There will be 2 input files**
- The pilot information will come from a file named `pilot_routes.txt`.
    - **Read this file first**
    - Each line in the file will contain the pilot's first name followed by a list of coordinates
    - There will be a new line at the end of each line except for the last line which may or may not have a new line
    - The pilot's name may be multiple words.
    - The coordinates will be separated by spaces.
    - Each line in the file will represent a different pilot.
    - There will be a space between each pair of coordinates
    - There will be a comma between the x and y coordinates.
    - The first and last set of coordinates will always be the same.
- A second input file will search and sort the data
    - Read this file second
    - The name of the file will be `commands.txt`
    - Each line of the file will contain a command
    - Valid commands:
        - `sort <area/pilot> <asc or dec>`
            - sort the linked list on the given criteria

        - `<pilot name>`
            - Search the linked list for the given pilot
        - `<number>`
            - Search the linked list for the given area
            - A match should be made if the number matches up to two decimal places

Input Validation

- Each file may contain invalid input
- If a line in the file contains invalid input, ignore that line
- Invalid input can contain the following:
    - In `pilot_routes.txt`
        - No space between pilot name and coordinates
        - Invalid characters (in name or coordinates)
            - Valid characters for names are alphanumeric characters, hyphens and apostrophes
            - Coordinates can only be numeric
        - Missing coordinates in a pair
        - Missing space between coordinates
    - In commands.txt
        - Invalid characters in commands
        - Unknown sort criteria
        - Invalid names
            - See criteria above for pilot names in `pilot_routes.txt`
        - Invalid number

Output:

- Output will be written to two files
- As you process `commands.txt` record the results of the commands in a file named `results.txt`
- Arrange the data in two columns
    - Column 1 – display the command processed
    - Column 2 – display results of command
- If a search is performed, write if the value was found or not found
    - Output format: `<value><found/not found>`
- If a sort is performed output the contents of the head and tail nodes
    - Output format: `Head: <value sorted>, Tail: <value sorted>`
- After processing commands.txt, write the linked list to a file named `pilot_areas.txt`.
    - Use the LinkedList class' toString method
    - The area will be rounded to 2 decimal places.
    - Each pilot's data will be written on a separate line.