CSE214

HOMEWORK - SPRING 2018

HOMEWORK 2 - due Tuesday, February 20th no later than 7:00pm

REMINDERS:

- **Be sure your code follows the coding style for CSE214.**
- **Make sure you read the warnings about academic dishonesty. *Remember, all work you submit for homework assignments MUST be entirely your own work. Also, group efforts are not allowed.***
- **Login to your grading account and click "Submit Assignment" to upload and submit your assignment.**
- **You are not allowed to use ArrayList, Vector or any other Java API Data Structure classes to implement this assignment except where noted.**
- **You may use Scanner, InputStreamReader, or any other class that you wish for keyboard input.**

Following a very productive semester of offloading actual code onto his teammates in CSE 308, Billy Business has decided to open up a local delivery service, bringing students on campus food from exciting places like Chick-Fil-A, China Station, Chipotle and Crazy Crepe Cafe (you know, because C's get degrees). However, since Billy has spent his entire upper division CS career freeloading off of his teammates, he has pretty much forgotten how to code. He therefore needs your help designing a computer-based stop scheduler (bonus points for Gui, App and Web!). Being a clever 214 student, you realize that the best way to schedule stops on a delivery route is with a linked list, so your goal is to implement a stop scheduler that will let Mr. Business arrange his deliveries in the most optimal order. He must be able to add delivery jobs, remove them (in case the client doesn't offer him enough bitcoin), reorder them (by -ie: crl+x- an order, moving the cursor and pasting it) and mark an order as completed. If Billy is successful enough, he might expand his delivery business by hiring his friend Money Mike, so he must be able to switch between his delivery list and Mike's, and transfer delivery jobs back and forth.

NOTE: All exceptions explicitly thrown in Required Classes except for IllegalArgumentException are custom exceptions that need to be made by you.

Required Classes

Delivery

Write a fully-documented class named Delivery which contains the source, destination, and instruction for a delivery . You should provide getter and setter methods for all member variables. In addition, you should provide a constructor as detailed below, though you may create a custom constructor which takes any arguments you see fit. Lastly, you should provide a toString() method that returns a printable representation of the Delivery and its data members.

- private String source
- private String dest
- private String instruction

DeliveryListNode

Write a fully-documented class named DeliveryListNode that wraps a Delivery object to allow it to be inserted into a doubly linked-list data structure. The Delivery object reference should be contained in a field called data and there should be two DeliveryListNode references serving as ‘links’ to the previous and next DeliveryListNodes in the list. You should provide getter and setter methods for all member variables in this class as well. The class will be based on the following ADT specification:

- private Delivery data
- private DeliveryListNode next
- private DeliveryListNode prev
- public DeliveryListNode(Delivery initData)
    - Default constructor.
    - Preconditions:
        - initData is not null.
    - Postconditions:
        - This DeliveryListNode has been initialized to wrap the indicated Delivery, and prev and next have been set to null.
    - Throws:
        - IllegalArgumentException if initData is null.

DeliveryList

Write a fully-documented class named DeliveryList which implements a double linked-list data structure. The DeliveryList should maintain a list of Deliveries by chaining a series of DeliveryListNodes between a head and a tail reference. In addition, a cursor should be provided to allow a user to traverse the list, selecting individual DeliveryListNodes to allow for insertion, deletion, and manipulation of the Deliveries they contain.

- private DeliveryListNode head
- private DeliveryListNode tail
- private DeliveryListNode cursor
- public DeliveryList()
    - Default constructor which initializes this object to an empty list of Deliveries.
    - Postconditions:
        - This DeliveryList has been initialized with head, tail, and cursor all set to null.
- public int numDeliveries()
    - Returns the total number of Deliveries in the list.
    - This method should run in O(1) time.
- public Delivery getCursor()
    - Gets the reference to the Delivery wrapped by the DeliveryListNode currently referenced by cursor.

    - Returns the reference by the Delivery wrapped by the DeliveryListNode currently referenced by cursor. If the cursor is null, then this method should return null as well (i.e. the cursor does not reference an Delivery).
- public void resetCursorToHead()
    - Returns the cursor to the start of the list.
    - Postconditions:
        - If head is not null, the cursor now references the first DeliveryListNode in this list.
        - If head is null, the cursor is set to null as well (there are no Deliveries in this list).
- public void cursorForward()
    - Moves the cursor to select the next DeliveryListNode in the list. If the cursor is at the tail of the list, this method throws an exception (this includes the case where cursor and tail are both null).
    - Throws:
        - EndOfListException if cursor is at the tail of the list.
- public void cursorBackward()
    - Moves the cursor to select the previous DeliveryListNode in the list. If the cursor is at the head of the list, this method throws an exception (this includes the case where cursor and head are both null).
    - Throws:
        - EndOfListException if cursor is at the head of the list.
- public void insertAfterCursor(Delivery newDelivery)
    - Inserts the indicated Delivery after the cursor.
    - Preconditions:
        - newDelivery is not null.
    - Postconditions:
        - newDelivery has been wrapped in a new DeliveryListNode object.
        - If cursor was previously not null, the newly created DeliveryListNode has been inserted into the list after the cursor.
        - If cursor was previously null, the newly created DeliveryListNode has been set as the new head of the list (as well as the tail).
        - The cursor remains unchanged.
    - Throws:
        - IllegalArgumentException if newDelivery is null.
    - Warning:
        - You should NOT move data references around between DeliveryListNodes to accomplish this method. Instead, you should wrap the newDelivery object in a new DeliveryListNode object, and insert this object into the list before the cursor.
- public void appendToTail(Delivery newDelivery)
    - Inserts the indicated Delivery after the tail of the list.
    - Preconditions:
        - newDelivery is not null.
    - Postconditions:
        - newDelivery has been wrapped in a new DeliveryListNode object.
        - If tail was previously not null, the newly created DeliveryListNode has been inserted into the list after the tail.
        - If tail was previously null, the newly created DeliveryListNode has been set as the new head of the list (as well as the tail and the cursor).
        - The tail now references the newly created DeliveryListNode.

- Throws:
  - IllegalArgumentException if newDelivery is null.
- Note:
  - This insertion method does not affect the cursor, unless the list was previously empty. In that case, head, tail, and cursor should all reference the new DeliveryListNode.

- public Delivery removeCursor()
  - Removes the DeliveryListNode referenced by cursor and returns the Delivery inside.
  - Preconditions:
    - cursor is not null.
  - Postconditions:
    - The DeliveryListNode referenced by cursor has been removed from the list.
    - All other DeliveryListNodes in the list exist in the same Delivery as before.
    - The cursor now references the next DeliveryListNode (or the tail, if the cursor previously referenced the tail of the list).
  - Throws:
    - EndOfListException if cursor is null.

DeliveryDriver
Write a fully-documented class named DeliveryDriver which creates two instances of the DeliveryList class and provides an interface for a user to manipulate the list by adding, and removing Deliveries. The following functionality should be provided:

- public static void main(String[] args)
  - The main method runs a menu driven application which first creates an empty DeliveryList and then prompts the user for a menu command selecting the operation. The required information is then requested from the user based on the selected operation. You can find the list of menu options in the UI required functions section.

Note on Exceptions: all exceptions should be handled gracefully - they should be caught in the main, and the user should be notified by a nice printout. Your messages will not be graded for creativity, but they should clearly indicate what the problem is (bad index, full list, negative number, etc.). The program should continue to run normally after an exception is encountered. We will not be checking Input Mismatch cases.

Pretty Printing: Here is a tutorial on how to print tables neatly using printf in java. It is highly encouraged that you print the output neatly, as it makes grading much easier. https://docs.oracle.com/javase/tutorial/java/data/numberformat.html

---

General Recommendations

You might want to implement a toString() method for Book and Bookshelf to make debugging and printing easier. You do not have to do this, but it will help you.

You can feel free to add extra methods and variables if you need.

UI Required Functions

- A - Add Delivery After Cursor
  - Source
  - Destination
  - Special Instruction

- H - Cursor to Head
- T - Cursor to Tail
- F - Cursor Forward
- B - Cursor Backward
- R - Remove Cursor
- X - Cut Cursor - This remove the cursor normally, but save the object in a temp variable.
- V - Paste After Cursor - Insert the last object to be cut after the cursor - there can be a nothing to paste message incase the user tries to paste a delivery without first doing a cut operation.
- S - Switch Delivery Route
- P - Print Current Delivery Route
- Q - Quit

Note: please make sure that the menu is NOT case sensitive (so selecting A should be the same as selecting a).

Program Sample

```
Welcome to the Delinquent Dollar Delivery Scheduler.

Menu:
     A) Add a Delivery After Cursor
     R) Remove Delivery At Cursor
     X) Cut Cursor
     V) Paste After Cursor
     H) Cursor to Head
     T) Cursor to Tail
     F) Cursor Forward
     B) Cursor Backward
     S) Switch Delivery Lists
     P) Print current list
     Q) Quit

Please select an option: A
Please enter a source: China Station
Please enter a destination: Tubman Hall
Please enter any special instructions: Slide noodles under door one by
one.

Order inserted.

Please select an option: A
Please enter a source: Dominos
Please enter a destination: Toscanini College
Please enter any special instructions: Deliver right before sunrise.

Order inserted.

Please select an option: H
Cursor is at head.
Please select an option: A
Please enter a source: Starbucks
Please enter a destination: Chem Department
Please enter any special instructions: Ensure all beverages have pH well
above 7.

Order inserted.

Please select an option: T
Cursor is at Tail.

Please select an option: A
Please enter a source: East Side Dining
Please enter a destination: Nearest Dumpster
Please enter any special instructions: Wear a hazard suit.

Please select an option: P

Biz Billy's Deliveries:
----------------------------------------------------------
To: Tubman Hall | From: China Station
Instruction:Slide noodles under door one by one.
~
To: Chem Department | From: Starbucks
Instruction: Ensure all beverages have pH well above 7.
->
To: Toscanini College | From: Dominos
Instruction: Deliver right before sunrise.              //cursor doesn't
move.
~
To: Nearest Dumpster | From: East Side Dining
Instruction: Wear a hazard suit.
---------------------------------------------------------

Please select an option: X
Cursor is cut.
Please select an option: P

Biz Billy's Deliveries:
----------------------------------------------------------
To: Tubman Hall | From: China Station
Instruction:Slide noodles under door one by one.
~
To: Chem Department | From: Starbucks
Instruction: Ensure all beverages have pH well above 7.
->
To: Nearest Dumpster | From: East Side Dining
Instruction: Wear a hazard suit.               //cursor moves forward
unless it's at tail.
---------------------------------------------------------

Please select an option: S

Money Mike's list is selected.

Please select an option: V

Money Mike's Deliveries:
----------------------------------------------------------
->
To: Toscanini College | From: Dominos
Instruction: Deliver right before sunrise.
---------------------------------------------------------

Please select an option: S

Biz Billy's list is selected.

Please select an option: R

Delivery to Nearest Dumpster removed.

Please select an option: B

Cursor moved backward.

Please select an option: P
```

```
Biz Billy's Deliveries:
----------------------------------------------------------
->
To: Tubman Hall | From: China Station
Instruction:Slide noodles under door one by one.
~
To: Chem Department | From: Starbucks
Instruction: Ensure all beverages have pH well above 7.
---------------------------------------------------------

Please select an option: Q

Next time, try UPS!
```

Extra Credit

You will get up to 6 points extra credit for creating a JavaFX GUI, and up to 12 points for creating an Android App. You may also get up to 15 points for making a webapp with JSP/JSF. Please discuss with your Grading TA if you choose to make an app or use Scene Builder, so you can coordinate submitting supplementary files with them.

Course Info | **Schedule** | **Sections** | **Announcements** | **Homework** | **Exams** | **Help/FAQ** | **Grades** | **HOME**