Metropolitan State University
ICS 240 - 50: Introduction to Data Structures
Spring 2022

Homework 03: Linked List of `Things`
Total points: 40
Out: Tuesday June 21, 2022
Due: 11:59 PM on Monday, June 27, 2022
Last day to submit with penalty: 11:59 PM on Thursday, June 30, 2022

Objective

In this assignment, you are asked to implement a **bag** collection using a **linked list** and the collection is to store only one type of `Thing` of your choice.

Important: you can **NOT** use the `LinkedList` class from the Java library, instead, you must implement your own linked list as explained below.

Requirements

In order to complete this project, you need to implement the following classes. The details of each class is explained below.

- `Thing`: you **must** use the same `Thing` that you used for the Assignment 2. But make sure to look for the requirements below and change your previous implementation as necessary.
- `ThingNode`: a linked list `node` where the `data` part is of type `Thing`.
- `ThingLinkedBag`: a collection of `Things` in which the elements are stored in a linked list.
- `ThingLinkedBagDriver`: a Java class that includes only a `main` method to test the functionality of your collection.

`class Thing`

Use the same `Thing` class that you used in Homework 2 but make sure all the following requirements are met. Basically, your `Thing` class must have the following:

- has exactly 2 instance variables such that:
  - one instance variable of type `String` and this variable will be used as a search key.
  - one instance variable of type integer (i.e, `int`) and this variable will be used to find aggregate information about `Things` that are stored in a collection class as will be explained later.

- An implementation of `toString` method that returns a one-line representation of the `Thing`.
- Implement `equals` method with the following header to test the equality of two `Things`.

```
public Boolean equals(Object other)
```

- Implement `Comparable<Thing>` interface and implement the corresponding `compareTo` method as explained in Homework 2.
- **Note 1:** do not name your class `Thing.java`, however, give your class a name that is indicative to the type of elements that you are maintaining. Also, the name of your class should be singular not plural (e.g, `Car` not `Cars`) because this class is used to represent only one element.
- **Note 2:** the name of all the other classes should be changed to indicate the type of things you are maintaining. For example, if you chose to implement a collection of cars, then your classes should be named `Car`, `CarNode`, `CarLinkeBag`, and `CarLinkedBagDriver`.

`class ThingNode`

This class represents a node in the linked list. The node class should include the following:

- two private instance variables:
  - `data` of type `Thing`. It is required that the data part to be of a specific `Thing` type.
  - `link` of type `ThingNode`.
- a constructor that takes two input parameters (`Thing` and `ThingNode` respectively) and use them to initialize the two instance variables.
- The following instance methods which are the same as in the `IntNode` class that we discussed in class and also is discussed in the text book (Chapter 4, Section 4.2). You must change the methods as appropriate to reflect your new type of data.
  - `getData, getLink, setData, setLink, addNodeAfter, removeNodeAfter`.
- The following static methods that are the same as `IntNode` class. You must change the methods as appropriate.
  - `display, listPosition, listLength,` and `listSearch.`

`class ThingLinkedBag`

This class is used to manage a collection of things where the things are stored in a linked list (not an array). We discussed how to implement `IntLinkedBag` in class and this is also discussed in Section 4.4 of the text book. The requirements of the `ThingLinkedBag` class are as follows:

- two instance variables:
  - `head` of type `ThingNode` to represents the head of the linked list.
  - `manyItems` of type `int` to represent the number of nodes in the linked list.
- Implement the following **instance** methods in the collection class. Note that all the method headers below are written in terms of `Thing`, however, your implementation should replace each `Thing` with your specific data type.
  1. a constructor to create an empty linked list.
  2. `int size()`: returns the number of nodes in the list.

3. `void display()`: displays the contents of the collection such that each element is displayed on one line. Note that this method displays the list on the screen and does NOT return a String representation of the list.
4. `void add(Thing element)` : a method to add a `Thing` to the collection.
5. `void add(int position, Thing element)`: a method to add an element at a specific position in the collection assuming the head node is at position 1. If the position is greater than the collection length, then the element is added as the last element in the collection. The method does not do anything if `position` is negative.
6. `void addLast(Thing element)`: adding an element to be the last element in the linked list.
7. `boolean remove(Thing target)` : this method removes one occurrence from the `target` from the list if any. The method returns `true` if an item is removed and `false` otherwise.
8. `boolean remove(int position)` : this method removes the element that is located at position `position` in the linked list where the head node is at position 1. The method returns `true` if an item is removed and `false` if no element is removed because position is negative or beyond the list length.
9. `void removeLast()`: this method removes the last node in the linked list.
10. `int countRange(Thing start,Thing end)`: this method counts and returns how many element in the collection falls in the range between `start` and `end` inclusive. Note that things are compared using the `compareTo` method.
11. `void set(int position, Thing element)` replaces the element at position `position` with the input `element`. If `position` is negative or beyond the length of the list, then the method does not do anything.
12. `int total()`: this method returns the sum of all the integer values of all things in the list (remember that it was required that each thing has an integer attribute).
13. `ThingNode lessThan(Thing element)`: this method takes one `Thing` as input and returns an output a linked list that includes all elements that are less than or equal to the input `element`. Note that things are ordered based on the `compareTo` method.
14. `Thing max()`: this method returns the maximum `Thing` in the linked list where things are ordered based on the `compareTo` method.

`class ThingLinkedBagIterator`

Implement `ThingLinkedBagIterator` as an inner class in `ThingLinkedBag`. An implementation of the linked list iterator is discussed in class and the is posted on D2L. Basically, the header of your class should be as follows:

```
class ThingLinkedBagIterator implements Iterator<Thing>
```

Implement therest of iterator class as necessary to reflect your `Thing` data type.

Change your `ThingLinkedBag` class to implement `Iterable<Thing>` and add the `iterator` method that returns an iterator of type `ThingLinkedBagIterator`.

`class ThingLinkedBagDriver`

Write a driver class to test ALL the methods that you implemented in the `ThingNode` class and the `ThingLinkedBag` class.

Important

- Start early
- Include Javadoc comments in your code.
- Use the exact same name for method (case-sensitive) as explained above. Also make sure to use the same input parameter order as explained above. For example, the `set` method takes `int` as first input and `Thing` as a second parameter.
- Solve the problem in stages. Save working versions periodically so you can return to a version that works as needed.
- It is always better to receive partial credit for a program that runs but lacks one or more methods than to receive no credits for a program that does not compile nor run.
- If you are to submit a partial implementation, your driver should include test cases for what you have completed.
- Upload to D2L only **one** zip file that includes all the four java files as explained above.

Grading

Your grade in this assignment is based on the following:

- Your submission meets specifications as described above.
- You must use the exact same name (case matched) and input/output data types for all methods as specified in the above description. Moreover, the method input parameters must be in the same order that is specified above.
- The program is robust with no runtime errors or problems.
- You follow the good programming style as discussed in class (see the file CodingStandards.pdf in the assignments folder of D2L).
- Follow the submission instructions given below.
- Here is a broad outline of the rubric:
  - The `Thing`class is coded exactly as specified with all the required methods. (copied from homeowork 2).
  - The `ThingNode` class is coded exactly as specified with all the required methods. **(7 points).**
  - The `ThingLinkedBag`is coded exactly as specified with respect to constructors and all methods and their signatures. **(23 points)**
  - The Iterator class is implemented and works as expected. **(4 points)**
  - The `ThingLinkedBagDriver` class works correctly and includes test cases for all methods **(3 points**).
  - The program is coded as per coding standards and JavaDoc comments are included. (3 points)

Submission Instructions

Follow the following steps to upload your code to D2L:

- For the final project submission, create a java project and call it `<yourLastName>Assignment3`(e.g., mine will be called *GhanemAssignment3*)
- Create four .java files to implement the classes as described above.
- Archive the project files into **one zip** file using Eclipse using the following steps:
  - In Eclipse Project Explorer, right click on the project folder of the project and click on Export.
  - Choose ***General*** then ***Archive File*** and click ***Next***.
  - Use the Browse key to choose a folder to store the archive file on your hard drive and give the file the same name as your project (e.g., `GhanemAssignment3.zip`), then click **Save**, then click **Finish**.
- Save the project plan time sheet files in the directory as the project's zip file. Then zip everything in another zip file called with the name `<your-zip-file-name>-allfiles` (e.g., `GhanemAssignment3-allfiles.zip`)
- Upload **only one** **.zip** file to the D2L folder called Assignment 3.
- **It is important that you upload your code in only one zip file. Your assignment will not be graded if you upload individual files to the drop box.**
- You may submit more than once, but then I will grade the most recent submission.