CSCI 1302: Software Development

Fall Semester, 2016

Project 4

Doubly Sorted Linked List

Instructor: *Eman Saleh*

Due Date: 11/29/2016 @ 11:30 PM

In this project you will implement your own Doubly Sorted Linked List. You will practice Generics, Serialization, Comparable interface.

Project Requirements:

1. **Define and implement the following classes:-**
   - **1.1 [10 Points]** Class `Person` class should implement the generic `Comparable<T>` interface and contain the following:

     - `Person`'s *first name* (String)
     - `Person`'s *last name* (String)
     - `Person`'s *id number* (4-digit integer)
     - `Person`'s *date of birth* (Date)
     - `public String toString()`: This method method should return the following string (without \n) for a student with id=1111, firstName="Joe", lastName="Smith", dob="01-25-1990":
       `1111 Joe Smith 01-25-1990`

   The `Person` class should implement all getter and setter methods for these data members, and the `compareTo` method of the above `Comparable` interface. The comparison must be based on the *id* of the `Person` instances.

   - **1.2 [10 Points]** The `Student` class which is a subclass of class `Person`. It should contain a String attribute that holds the student's *college name*. It should also contain the proper `toString` method to add college name (in brackets) to the result of `Person.toString()`. So, for the above example if "Joe Smith" is a student at "Art&Sci" college. The output of the toString

     method should be
     `1111 Joe Smith 01-25-1990 [Art&Sci]`

1.3 [10 Points] The `DoublyNode` class should be generic (let us say the type variable is `T`), and contain:

- a reference of type `T` called *data*
- reference to the previous `Node` called *prev*
- a reference to the next `Node` called *next*

`prev` and `next` should reference the respective nodes in the `SortedDblList` (making this a doubly linked list).

1.4 [10 Points] The `Demo` program class: A class with a `main` method that demonstrates that every method of your `SortedDblList` works. For the methods `union` and `intersection` of `SortedDblList` show that they work even if the current `SortedDblList` is a SortedDblList of `Person`s and the `otherList` is a SortedDblList of `Student`s (see below). Do not forget to demonstrate how to serialize (to a file) and deserialize a SortedDblList.

1.5 [50 Points] The `SortedDblList` class is a serializable sorted doubly linked list that does not allow *duplicates* or *null* elements. The class should contain:

- an integer `size` representing the number of items in the list
- a reference of type `DoublyNode` for the head (first node) of the list which is initially set to null.
- a `PrintList (...)` method: should write the value of `size` and then each element (but not nodes!) of the list to the given stream.

[10 Points] The class should always maintain sortedness of the list based on the result of the `compareTo` method implemented in the underlying object (in this case it will be based on the person's ID in the `Person` class). Since we are using the `compareTo` method, this class should be generic in the sense that it works for ANY `T` object that implements `Comparable<T>`. Do not assume `Person` or `Student` will always be used!! Note: I might test against this when grading. In other words, the class `SortedDblList` should be defined (as a generic class) such that it is a doubly linked list of elements of type `T`, where `T` or a superclass of `T` implements the `Comparable` interface. As you may recall, `T` is called the "type variable" of the `SortedDblList` generic class.

NOTE: PLEASE DESIGN TACH METHOD AND WRITE/DRAW TXAMPLES BEFORE IMPLEMENTING THE FOLLOWING METHODS! IT WILL SAVE YOU MUCH TIME, AND PREVENT FRUSTRATION!

A. [40 Points] You should implement the following functions in the `SortedDblList` class:

```
/*
Returns the element at the specified position in this list.
Parameters: index - index of element to return.
Returns: the element at the specified position in this list.
Throws: IndexOutOfBoundsException - if the index is out of range (index < 0 || index >= size()).
*/

public T get(int index);

/*
The add method will take an instance of the type variable T as the parameter element, and it
will insert it at the correct place within the list using element's compareTo method. This
method should not allow null elements or duplicates be added to the list. The method returns
true if successful and false otherwise. Therefore, the method does NOT throw exceptions in
case the argument is null or the element already exists in the list.
*/

public boolean add(T element);

/*
The remove method removes the first item with its id matching the ID value of the parameter
o, and returns true if this list contained the specified element. This method should return false
if the item is not found! REMEMBER: Use the equals method to find the item to remove! Your
implementation of equals should use the compareTo method. If this list does not contain the
element, it is unchanged. More formally, the method removes the element with the lowest
index i such that

                        get(i).equals(o) == ture.

*/

public boolean remove(Object o);

/*
The isEmpty method returns whether or not the list is empty (true or false)
*/

public boolean isEmpty();

/*
The merge method returns a new SortedDblList that is the union of the current list and
otherList while maintaining sortedness.
*/
```

```
public SortedDblList<T> merge(SortedDblList<? extends T> otherList);
```

```
/*
The isPrefix method returns true if the current Linked list is a prefix of the otherList. The
empty list is Prefix of any other list. A list of integers [2,4,8] isPrefix of the list [2,4,8,10,11]. A list
of integers [12,13,14] is not prefix of the list [3, 12, 13, 14].
*/

public boolean isPrefix (SortedDblList<? extends T> otherList);

/*
The printList method will print all elements of the list to the screen; one element per line. That is,
for each element of the list it should call the toString method of the element, and then append a
newline character to the result, and print the result to the screen. As an example, a list of two
Integers 3 and 4 should be printed to the screen as follows:
3
4
*/

public void printList();

/*
Returns the index in this list of the first occurrence of the specified element, or -1 if this list
does not contain this element. More formally, returns the lowest index i such that
get(i).equals(o) == true, or -1 if there is no such element.
*/

public int indexOf(Object o);
```

2. **[5 Points]** practice good programming style, use proper meaningful identifier names, indentation, and Javadoc comments.
3. **[5 Points]** Submit a README.txt file telling us how to compile and execute your program and how to generate Javadoc.

What to Submit:

Submit all .java files in your project4 directory to user cs1302a on *nike* at a directory called Project4.
(use the following command: submit Project4 cs1302a).

Basic grading criteria:

1. If the project did not compile on *nike* 0%
2. If the project compiled but did not run 0% - 30%
   30% is given if all required files were submitted and program code completely satisfies all functional requirements.
3. If the project run with wrong output 0% - 60%
   Depending on the solution, 50-60% is given ONLY if the cause of the error was minor after checking all functional aspects.

See course syllabus for late submission evaluation