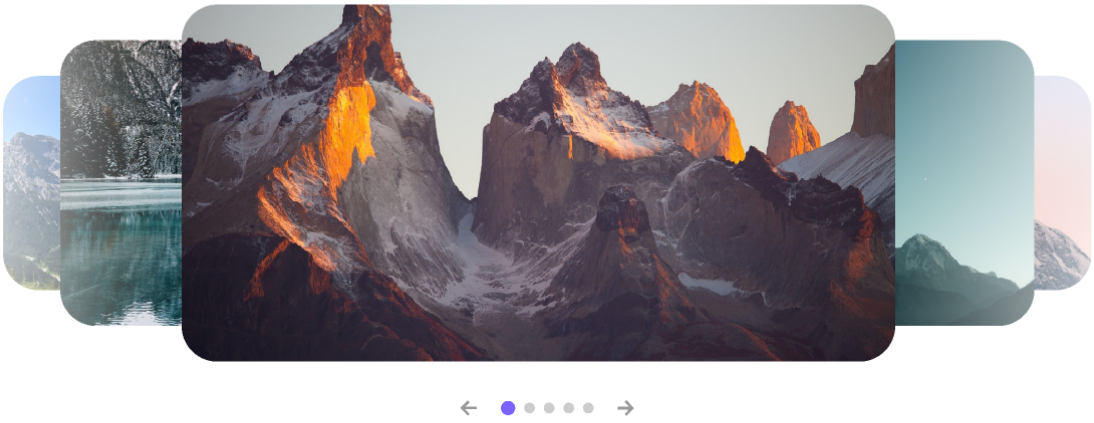
Data Structures Programming Assignment 2: Linked Lists

The year is 2005. You are working for a social media startup called PictureBook. PictureBook makes an app that allows users to upload photo albums and share them with their friends. Users can open their friends' photo albums and view them carousel-style:



Users should be able to move to the next or previous photo in the album. Additionally, once a user reaches the last photo in the album, they should be able to go back to the beginning of the album by going to the "next" photo. Thus, the photo album is circular in nature. The photo album should also "save" where a user left off, so that they can resume viewing at the same place later.

As users view photos, we should track whether they have viewed each photo, and whether they have viewed all the photos in an album. Photos can be added to or removed from an album at any time. If new photos are added to an album, they are considered unviewed.

Finally, we should be able to detect whether two albums are equal. For our purposes, two photo albums are equal if they contain the same photos (independent of order).

Implementation Details

For this assignment we have provided one interface file: IAlbum.java. You may not modify this file. You will use it when implementing the `Album` class. In addition to the interface, you will add the files below.

Photo.java

Objects of the `Photo` class represent individual photos that are part of an `Album`. Add a Photo.java file that contains a `Photo` class with the following members:

```
private String name;
```
Name of the photo.

```
private String photoDigest;
```
A string of characters representing the contents of the file. This uniquely identifies a photo.

```
public void viewPhoto() { /* TODO */ }
```

Should print a message “Now viewing [photo name].” The Photo instance should also have some way of remembering that it has been viewed.

`public boolean equals(Photo other) { /* TODO */ }`

Should return true if the two photos being compared have the same photoDigest.

You may add other data members and methods as you see fit.

Album.java

The `Album` class should implement the provided `IAlbum` interface. Internally, this class should implement a **circular doubly linked list** that models the functionality of a user opening a photo carousel and viewing photos. A circular linked list does not have a head or a tail; instead, it has a current element, which defaults to the first element added to the list. Because the list is circular, every element has both a previous and a next (even if there is only one element in the list—in that case, the previous and next are both the single list element).

Users should be able to add and delete photos from an album. Users should be able to open an album and cycle through the photos in either direction. Users should also be able to close an album. An album’s photos may only be viewed when it is open.

Please see the IAlbum.java interface file for descriptions of each of the methods to implement. You may add additional data members and methods to the `Album` class as you see fit. You must come up with your own circular doubly linked list implementation that does not use any of the linked list code from the lectures or textbook. Note that we have not included any interface for the nodes of the `Album` circular linked list, so that part of the implementation is entirely up to you. You are welcome to implement a separate class for the nodes or add node functionality/properties to your `Photo` class.

Your `Album` class should include a `public static void main(String[] args)` method. This will be the entryoint for testing your program.

Submission Details

Please zip all new source (.java) files and submit on Brightspace. You should not include the provided interface file, since it should be unchanged. Please note in your submission form which version of Java you used to develop and test your code. (If you aren’t sure which version you’re on, run `java --version` in your terminal.)

The naming convention of your zip folder must be **<netid>.zip**.

Sample Inputs/Outputs and Testing

We have provided `sample-input` and `sample-output` directories with corresponding inputs and outputs that you can use to test your program. The input files are Java code snippets that should be pasted into your `main` method. The output files show the expected text output when you run your `Album` class (i.e. `java Album`). The input/output samples are deliberately ordered in a way that may help guide your implementation. Try working through the samples in order as you develop your program. Good luck!