CSC 212 Programming Assignment
Developing a Photo Management Application

College of Computer and Information Sciences
King Saud University

Fall 2020

1 Introduction

The role of a photo management application is to organize photographs so that they can be easily accessed. In order to help organize photos, the user can provide tags to describe the content of the photos. A tag is a keyword associated with a photo. For example, we can associate the tag "vacation" to any photo taken during a vacation. In Table 1, you may find some examples of photos and the tags that are used to describe them.

Table 1: Example of photos with associated tags.

 **hedgehog.jpg** animal, hedgehog, apple, grass, green	 **bear.jpg** animal, bear, cub, grass, wind	 **butterfly1.jpg** insect, butterfly, flower, color	 **butterfly2.jpg** insect, butterfly, black, flower
 **fox.jpg** animal, fox, tree, forest, grass	 **panda.jpg** animal, bear, panda, grass	 **wolf.jpg** animal, wolf, mountain, sky, snow, cloud	 **raccoon.jpg** animal, raccoon, log, snow

The photo manager organizes the photos into albums created by the user. An album is identified by a **unique** name and regroups photos that satisfy certain conditions. For the purpose of this assignment, the conditions used to create albums consist in a sequence of tags separated by "AND":

`Tag1 AND Tag2 AND Tag3`

Photos that contain all specified tags will appear in the album. An empty tag list matches all photos.

Example 1. *Using the photos of Table 1, the album with the condition* `bear`*, will contain two photos (that of the panda and the grizzly bear). The album with the condition* `animal AND grass` *will contain four photos (hedgehog, grizzly bear, fox and panda). The albbum with no tags will contain all eight photos.*

2 Inverted index

In order to accelerate the search for photos, it is possible to store the tags in an *inverted index*. The idea is that instead of having the photos point to the tags, the inverted index will store all the tags, and each tag will point to all the photos that contain it.

The following is an example showing a partial inverted index for the photos shown above:

animal →	hedgehog.jpg, bear.jpg, fox.jpg, panda.jpg, wolf.jpg, racoon.jpg
apple →	hedgehog.jpg
bear →	bear.jpg, panda.jpg
black →	butterfly2.jpg
butterfly →	butterfly1.jpg, butterfly2.jpg
...	

You are required to:

1. Represent the photos-tags association using an inverted index stored in the class *PhotoManager*.
2. Use a data structure that allows $O(\log n)$ in average to search for a tag.

3 Requirements

You are required to implement the following specification:

```
public class Photo {
   // Constructor
   public Photo(String path, LinkedList<String> tags);
   // Return the path (full file name) of the photo. A photo is uniquely identified
        by its path.
   public String getPath();
   // Return all tags associated with the photo
   public LinkedList<String> getTags();
}
```

```
public class Album {
   // Constructor
   public Album(String name, String condition, PhotoManager manager);
   // Return the name of the album
   public String getName();
   // Return the condition associated with the album
   public String getCondition();
   // Return the manager
   public PhotoManager getManager();
   // Return all photos that satisfy the album condition
   public LinkedList<Photo> getPhotos();
   // Return the number of tag comparisons used to find all photos of the album
   public int getNbComps();
}

public class PhotoManager {
   // Constructor
   public Photomanager();
   // Add a photo
   public void addPhoto(Photo p);
   // Delete a photo
   public void deletePhoto(String path);
   // Return the inverted index of all managed photos
   public BST<LinkedList<Photo>> getPhotos();
}
```

Remark 1. *The list of photos that belong to the album is determined at the time when the method* getPhotos *is called, not when the album is created.*

4 Deliverables and rules

You must deliver:

1. Source code submission to Web-CAT.

The submission **deadline** is: **06/12/2020**.

You have to read and follow the following rules:

1. The specification given in the assignment (**class and interface names, and method signatures**) must not be modified. Any change to the specification results in compilation errors and consequently the mark zero.

2. All data structures used in this assignment **must be implemented** by the student. The use of Java collections or any other data structures library is strictly forbidden.

3. This assignment is an individual assignment. Sharing code with other students will result in harsh penalties.

4. Posting the code of the assignment or a link to it on public servers, social platforms or any communication media including but not limited to Facebook, Twitter or WhatsApp will result in disciplinary measures against any involved parties.

5. The submitted software will be evaluated automatically using Web-Cat.
6. All submitted code will be automatically checked for similarity, and if plagiarism is confirmed penalties will apply.
7. You may be selected for discussing your code with an examiner at the discretion of the teaching team. If the examiner concludes plagiarism has taken place, penalties will apply.