Computer Science 241

Lab 2 (10 points)
Due Sunday, April 22nd, 2018 at 10:00 PM

Read all of the instructions. Late work will not be accepted.

Lab Description

Scenario

You have been asked by a wealthy record collector to write a program to organize information about her record collection. For each record in the collection, she wants to store (at the least):

- The artist/band name
- The title of the record
- The year the record was first released
- The number of copies she has of the record in her collection

Record Collection

You should design a `RecordCollection` class to organize the records in the collection. You may also design, as the need arises, one or more other classes that would be useful in your implementation. The highest priority operations the record collector wishes to perform are:

- Adding a new record to the collection
- Incrementing the number of copies of a particular record already in the collection
- Decrementing the number of copies of a particular record in the collection (deleting it if the number of copies would now be zero)
- Finding the subset of records matching a particular artist
- Finding the subset of records matching a particular release year
- Printing out the collection, sorted by release year

You need only implement **stub** methods that would eventually support these various operations. In particular, each method body should do only two things:

1. Print `Called method XYZ` or `Called method XYZ with arguments A B C`

   Where `XYZ` is the method name and

   `A`, `B` and `C` are the values of the arguments passed

2. Return a trivial value (e.g. 0 if the return type is `int`, `""` if the return type is a String, `null` if the return type is a non-String object, etc.).

For example, a stub for a method to get the number of records in the collection might be:

```
public int collectionSize() {
    System.out.println("Called method collectionSize");
    return 0;
}
```

The focus of this lab is on *design*. You should carefully consider design decisions, including:

- What other classes, if any, would be helpful to introduce?
- What fields should go into which classes?
- What are suitable types and clear names for these fields?
- Which methods should go into which class?
- What are clear, appropriate method names for the various operations?
- What are suitable return types for each method?
- What are suitable parameters for each method?

Driver Program

Neither your `RecordCollection` class nor any of the other classes you might make above should contain a `public static void main` method. Instead, you should implement a separate `Lab2` class with a `public static void main` method. This `main` method should create an instance of your `RecordCollection` and call each of the public methods (possibly with dummy values). If you introduced additional classes, `Lab2`'s `main` method should also create instances of those and call each of their public methods. The goal here is to ensure that the methods in your classes can be called as you expect. Proper testing is the topic of a later lab.

Writeup

You should include a plain text (e.g. created with notepad, kate, gedit, vim, etc.) write up named `writeup.txt` (spelling, spacing and capitalization matter). It should be between 300 and 500 words, and should describe and justify your design decisions in plain English: **why** did you make the decision you did? This writeup should be well-written and free of grammatical and spelling errors. (Hint: you can use `aspell -c writeup.txt` on the commandline to spell check a plaintext document.)

Submission

The `master` branch of the `origin` repository (i.e. the one I made for you in the `hutchteaching` organization) should contain the following files:

- `lab2/RecordCollection.java`
  Your `RecordCollection` class with stub methods
- `lab2/Lab2.java`
  Your `Lab2` driver class with `main`
- `lab2/*.java`
  Source code for all other classes you implemented
- `lab2/writeup.txt`
  Your writeup

You can confirm that your code is properly submitted by checking your github repo URL:

`https://github.com/hutchteaching/201820_csci241_username`

Grading

At the deadline a script will automatically clone your repository. Points will be deducted for any problems in your submission, including:

- Missing or incorrectly named files or directories
- Code that does not compile
- Code that generates run-time exceptions
- Failing to call all of your public methods from `Lab2`'s `main` method
- Poorly designed classes and methods (e.g. bad names, return values, parameters, etc.)
- An overly brief or overly verbose writeup
- Poor writing in your writeup

Acknowledgments

Thanks are owed to Tanzima Islam, Qiang Hao, current TA Jonny Mooneyham and several past TAs for producing and refining the lab on which lab is modeled.