Name: _____ Andrew Id: _____

15-121 Fall 2024 Assessment 5

Up to 50 minutes. No calculators, no notes, no books, no computers. Show your work!

1. (15 points) **Course Enrollment**

   In this problem you will write a simple class to store information about student course enrollment.

```
public class CourseEnrollment {
    /**
     * Enroll a student to a course. If the student is already enrolled in the
     * course, then do not add them again (and return false).
     *
     * This must be O(1).
     *
     * @param student The student enrolling in the course
     * @param course  The course the student is enrolling it
     * @return true if this method call enrolls the student in the course and false
     *         otherwise.
     */
    public boolean enroll(String student, String course) {
            // You will write this code
    }

    /**
     * Fetch a list of the courses a specific student is enrolled in.
     *
     * This must be O(m), where m is the number of courses the student is enrolled
     * in.
     *
     * @param student The student whose course list is being requested.
     * @return An array of the courses the student is enrolled it. If the student
     *         isn't enrolled in any classes, return an empty array.
     */
    public String[] getCourses(String student) {
             // You will write this code
    }

    /**
     * Fetch a list of the students in a specific course.
     *
     * This must be O(m), where m is the number of students in the course.
     *
     * @param course The course which is having its student list requested.
     * @return An array of the students in the course. If the course doesn't have
     *         any students, or doesn't exist, return an empty array.
     */
    public String[] getRoster(String course) {
             // You will write this code
    }


     public static void main(String[] args) {
        CourseEnrollment c = new CourseEnrollment();
        c.enroll("aamohd", "15-121");
        c.enroll("fnasr", "15-282");
        c.enroll("aamohd", "67-262");
        c.enroll("fnasr", "67-262");
        c.enroll("aamohd", "79-104");
        c.enroll("yusuft", "15-121");
        c.enroll("mozaj", "15-121");
        c.enroll("fnasr", "21-127");

        String[] courseList = c.getCourses("aamohd");
        System.out.println("Course list for aamohd:");
        for (String course : courseList) {
            System.out.println("\t" + course);
        }

        String[] roster = c.getRoster("15-121");
        System.out.println("Course roster for 15-121:");
        for (String student : roster) {
            System.out.println("\t" + student);
        }
    }
}
```

The main method, when executed, outputs:

```
Course list for aamohd:
        67-262
        79-104
        15-121
Course roster for 15-121:
        mozaj
        yusuft
        aamohd
```

Fill in your answer in the following:

```
public class CourseEnrollment {
    // You may add whatever instance variables you need.









    /**
     * Enroll a student to a course. If the student is already enrolled in the
     * course, then do not add them again (and return false).
     *
     * This must be O(1).
     *
     * @param student The student enrolling in the course
     * @param course  The course the student is enrolling it
     * @return true if this method call enrolls the student in the course and false
     *         otherwise.
     */
    public boolean enroll(String student, String course) {









}
```

```
    /**
     * Fetch a list of the courses a specific student is enrolled in.
     *
     * This must be O(m), where m is the number of courses the student is enrolled
     * in.
     *
     * @param student The student whose course list is being requested.
     * @return An array of the courses the student is enrolled it. If the student
     *         isn't enrolled in any classes, return an empty array.
     */
    public String[] getCourses(String student) {









    }
    /**
     * Fetch a list of the students in a specific course.
     *
     * This must be O(m), where m is the number of students in the course.
     *
     * @param course The course which is having its student list requested.
     * @return An array of the students in the course. If the course doesn't have
     *         any students, or doesn't exist, return an empty array.
     */
    public String[] getRoster(String course) {









    }
}
```

2. **Free Response**

(a) (20 points) A `JournalPub` class.

In this problem you are going to write a `JournalPub` class to store information about journal publications. Here are some important requirements for your class:

- A `JournalPub` has a title, journal name, publication year, publication month (as a number), and list of authors. When a new `JournalPub` is created, these items are passed to the constructor. (You must provide the constructor.)
- The only way for code outside of `JournalPub` to read and modify the attributes of a `JournalPub` must be using appropriately named getters and setters. (You must provide the getters and setters.)
- A title must not be `null`. If an attempt is made to set the title to something invalid, an `IllegalArgumentException` must be thrown.
- A journal name must not be `null`. If an attempt is made to set the journal name to something invalid, an `IllegalArgumentException` must be thrown.
- The month must be valid. If an attempt is made to set the month to something invalid, an `IllegalArgumentException` must be thrown.
- The list of authors cannot be null or empty. If an attempt is made to set the list of authors to something invalid, an `IllegalArgumentException` must be thrown.
- When printing a `Song` (such as when `System.out.println(someJournalPubObjectHere)` is used.), it should be formatted as follows:
  `Author List. Title. Journal Name. Year(Month).`
  Here is an example:
  ```
  Christos A. Kapoutsis, Lamana Mulaffer. "A Logical Characterization of Small 2NFAs".
  ↪  International Journal of Foundations of Computer Science. 2017(5).
  ```
- A `JournalPub` needs to be able to be properly used in `HashSet`s and `HashMap`s.
- An array of `JournalPub`s must be sortable using `Arrays.sort(someArrayOfJournalPubs)`. The natural order is based on the date (earlier dates come first), then the title of the journal.

Write the `JournalPub` class to the above specifications. Pay careful attention to which classes you may need to extend or implement as well as which methods you need to write or override.

Restriction: You may not use the the method `Objects.hash`.

Extra space for Question 2(a).

Extra space for Question 2(a).

(b) (5 points) Without modifying the `JournalPub` class, write some code that would allow you sort an array of `JournalPub`s by the article titles. You do not need to write a complete example with a main function and a testcase, instead just focus on what is needed in order to use the `Arrays.sort` method to sort an array of `JournalPub`s named `myJournalPubArray`.