CMSC133, Spring 2020, Exam #2, Deadline: Mon, April 13, 6:15 pm (No late deadline)

INSTRUCTIONS:

- Download the code distribution available at http://www.cs.umd.edu/class/spring2020/cmsc133/projects/zipFiles/e3spr/Exam2133.zip
- Import the above distribution as you usually import a project.
- Immediately after importing the code, verify you can submit (check the submit server submission). We will not accept projects submitted via e-mail.
- Grading: 34% release tests, 66% secret tests / style and requirements.
- There are no public tests.
- Student tests that you provide will not be graded.
- There is no late submission period, therefore you need to submit often and before the above deadline; we will grade the highest scoring submission.
- Plan to submit before the deadline. Network problems or submit server overloading are not valid excuses for not submitting your work.
- You have 8 release tokens and you can release-test your project anytime.
- You can only post clarification questions in Piazza. Debugging questions, why code is not compiling, why is code not passing a test, are invalid questions to post in Piazza. Use the Piazza folder **exam2questions** to post your questions; we will use the folder **exam2clarifications** to post clarifications (please, don't use this folder). Before posting a question, check the **exam2clarifications** folder.
- **Posting of any kind of code in Piazza, during this assignment period, represents an academic integrity violation and will be reported as such.**
- You may not address questions other students post in Piazza (only TAs and instructors can address questions).
- You must work by yourself.
- You can use class resources (lecture notes, lecture/lab examples, videos, etc.), but no other resources (e.g., code from the web).
- Sharing of this assignment solutions represents an academic integrity violation and will be reported as such.
- Submissions will be checked with cheating detection software.
- If you are an ADS student: The provided time period takes into consideration your time allocation. If you need any other assistance, contact your instructor. Ignore this entry if you don't know what an ADS is.

Specifications

For this assignment you will implement methods for the classes **Video**, **Playlist** and **ArrayUtilities**. You will find the classes in the **sysImplementation** package. A description of each method is provided below. A driver (that you can ignore if you know what to implement) and associated output is provided at the end. This driver is not part of the code distribution. Regarding the code you need to implement:

- You don't need to add comments to your code, but you must have good variable names, indentation and you should avoid code duplication.
- At this point you may want to take a look at the classes you will find in sysImplementation. We have provided a shell for each method you need to implement. You can also see the instance variables associated with each class.
- During the implementation of the above classes, you can add instance variables, constants (static final) and private methods.
- You can assume parameters are valid, unless we indicate otherwise (e.g., if the parameter is null throw …).
- You can create a StringBuffer out of another (e.g., new StringBuffer(anotherStringBuffer)) or by using a string (e.g., new StringBuffer(aString)).
- You can add toString() methods to the classes, but you are not required to do so. **We have provided a toString() method for the Video class; do not modify it, otherwise you will not pass release/secret tests.**
- You can use the compareTo method of the String class.
- You must provide an implementation for every method, otherwise your code will not work in the submit server. If you don't know what to do for a method, you can leave the following statement in the provided method shell:

```
throw new UnsupportedOperationException("Not Implemented");
```

ArrayUtilities Class Specification

This class defines the method **getRowsEvenCnt.** This method returns a new two-dimensional array of characters with **copies** of rows from the **src** parameter that have an even number of columns. The copies must appear in the same order they appear in the **src** array. Any row entry in **src** that is null (e.g., src[0] == null) or representing an empty array (e.g., array[1].length == 0) will be ignored. Notice the length of the array returned by the method is the same as the length of **src**. You can assume **src** is not null.

Video Class Specification

The class represents a video that has a title, duration and a set of reviews. The following instance variables are associated with this class:

```
private String title;
private int duration;
private StringBuffer allReviews;
```

The class methods are:

1. **Constructor** - Takes a title and duration as parameters, initializing the corresponding instance variables. Associates a StringBuffer object with the **allReviews** instance variable.
2. **Copy Constructor** - Creates a copy where changes to the copy will not affect the original.
3. **getTitle** - get method for title.
4. **getDuration** - get method for duration.
5. **getAllReviews** - get method for allReviews. You must avoid privacy leaks.
6. **addReview** - Appends the review provided in the parameter to the **allReviews** instance variable. No review will be added if the parameter is null or represents the empty string. The method returns a reference to the current object.

Playlist Class Specification

The class represents a list of videos. The following instance variables are associated with this class:

```
private String name;
private int entries, maxSize;
private Video[] videoList;
```

The class methods are:

1. **Constructor** - Takes a name and maximum size as parameters, initializing the corresponding instance variables. The method will create a Video array of size maxSize, and set the number of entries to 0. The method will throw an IllegalArgumentException (with any message) if the name parameter is null and/or the maximum size parameter is less than 1.
2. **addVideo** - Takes a video's title and duration as parameters. It creates a Video object based on the parameters and adds the new Video object to the videoList array. Entries will be added to the array starting at index position 0. Make sure you update any other instance variables, accordingly. The method will throw an IllegalArgumentException (with any message) if the title parameter is null and/or the duration parameter is less than or equal to 0. You can assume we will not add the same video more than once.
3. **getName** - get method for name.
4. **getEntries** - get method for entries.
5. **getMaxSize** - get method for maxSize.
6. **getVideoList** - returns a deep copy of the videoList.
7. **findVideo** - returns true if there is a video with the specified parameter in the videoList and false otherwise.
8. **addReview** - appends the provided review to the video associated with the specified title. The method will throw an IllegalArgumentException (with any message) if the title and/or review parameters are null. The method returns true if the review is added and false if there is no video associated with the specified title.
9. **compareTo** - This method will allow us to compare two Playlist objects. It has a Playlist as parameter and returns an integer. This method will return:
    a. A negative value if the number of entries of the current object is less than the parameter.
    b. A positive value if the number of entries of the current object is greater than the parameter.
    c. If the current object and the parameter have the same number of entries, the method will return:
        i. A negative value if the name of the current object precedes the name of the parameter in alphabetical order.
        ii. A positive value if the name of the current object follows the name of the parameter in alphabetical order.
        iii. 0 otherwise.
10. **removeVideos** - removes from the videoList those videos that have a duration that fall in the range specified by the parameters (inclusive). The method will throw an IllegalArgumentException (with any message) if the upper limit parameter is less than the lower limit parameter. The method returns the number of videos (if any) that were removed. **Notice the maximum size of the list will not change.**

Driver / Expected Output (Feel free to ignore)

Driver

```
public static void main(String[] args) {
   String answer = "";
   Video video = new Video("Aliens", 135);
   video.addReview("Scary").addReview("Excellent!");

   answer += getVideoInfo(video) + "\n";
   answer += "============================\n";

   Playlist playlist = new Playlist("interesting", 10);
   String videoTitle = "Emma";
   playlist.addVideo(videoTitle, 210);
   playlist.addReview(videoTitle, "Awesome");
   playlist.addVideo("Rocky", 95);
   playlist.addReview(videoTitle, "Awesome");
   playlist.addVideo("The Road", 330);

   answer += "Name: " + playlist.getName();
   answer += "\nEntries: " + playlist.getEntries();
   answer += "\nMaxSize: " + playlist.getMaxSize();

   boolean found = playlist.findVideo(videoTitle);
   answer += "\nFound: " + found;

   int removed = playlist.removeVideos(80, 95);
   answer += "\nRemoved: " + removed;
   answer += "\nVideos:\n";

   Video[] result = playlist.getVideoList();
   for (int i = 0; i < result.length; i++) {
      answer += result[i].getTitle() + "\n";
   }

   answer += "============================\n";
   char[][] src = { { 'A', 'B' }, {'F'}, { 'C', 'D' } };
   char[][] data = ArrayUtilities.getRowsEvenCnt(src);

   answer += data[0][0] + ", " + data[0][1] + "\n";
   answer += data[1][0] + ", " + data[1][1];

   System.out.println(answer);
}

private static String getVideoInfo(Video video) {
   String title = video.getTitle();
   int duration = video.getDuration();
   StringBuffer reviews = video.getAllReviews();

   String answer = title + ", ";
   answer += duration + ", ";
   answer += reviews;

   return answer;
}
```

Output

```
Aliens, 135, ScaryExcellent!
============================
Name: interesting
Entries: 3
MaxSize: 10
Found: true
Removed: 1
Videos:
Emma
The Road
============================
A, B
C, D
```