Homework 5 – YouTube: The Sequel

Authors: Tomas, David, Seo Hyun, Mridul, Nicolas

Problem Description

The Dean of the College of Computing, Chuck Isbell, and Brown University Professor Michael Littman, have collaborated to produce a music parody video. It's available to watch on YouTube now, but not for long. When YouTube discovers this atrocity, the platform will self-destruct. It had a good run, had lots of fun, but now it's done, it went too far. Regardless, the pair are intent that the public at large hear their noise pollution. They are seeking your help to program a new video-hosting platform to achieve this end.

The platform needs a representation of a video on the site. The video has comments which can be liked or disliked. The video itself can be liked or disliked. It is also important to grab certain metrics from the comments of the video like the top and worst comments.

Your program will use ArrayLists, generics, interfaces, and Comparable to accomplish the desired functionality.

Solution Description

Likeable.java

`Likeable` denotes whether an object can receive likes or dislikes. For instance, comments should be able to be liked and disliked. `Likeable`, an interface, should regiment certain requirements for classes that implement it. It will require the following methods:

- Methods:
  - `getLikes();`
    - returns number of likes
  - `getDislikes();`
    - returns number of dislikes

Video.java

`Video` is a class that represents a video page. It has comments and can be liked and disliked.

- Instance variables:
  - `ArrayList comments`
    - Represents the comments on a video
    - Comments should never be null.
    - Remember to specify a type for the ArrayList in the relevant angle brackets depending on what you would like to store in it. Note that not specifying a type will compile and run. However, this is a bad practice so ensure that you are always providing a type in the angle brackets.
  - `double length`
    - Represents the length of the video in minutes
    - The length of the video should never be negative or exceed 60 minutes
  - `int likes`
    - Represents the number of likes for a video
    - The number of likes should never be negative
  - `int dislikes`
    - Represents the number of dislikes for a video

        - The number of dislikes should never be negative
    - Getter and setter methods for each instance variable. For setters, note that if an invalid value is passed in, you should not modify the value of the instance variable.
- Constructor:
    - Should have a `Video` constructor that only takes in a length, likes, and dislikes
        - If an invalid value is passed in for `length`, `likes`, or `dislikes`, simply default to 0.
        - A Video should begin with an empty ArrayList to which comments can be added
- Methods:
    - `createComment()`
        - Adds a comment to the back of `comments`.
        - Takes in a comment.
    - `removeComment()`
        - Remove a specific comment from the video.
        - Takes in the comment to be removed.
        - Returns a boolean whether the comment was removed.
    - `spamComments()`
        - Returns an ArrayList where all comments are spam.
    - `toArray()`
        - Converts `comments` to an array.
        - Returns the array.
    - `topAndWorstComment()`
        - Returns an array with two comments. The first element in the array should be the top comment and the second element in the array should be the worst comment. Both are decided based on the like factor. Look at the explanation for like factor in the Comment.java section.
        - If there is a tie between the top comment and the worst comment, return the same comment for the top comment and worst comment.
        - If there is a tie for top or worst comment use the first one encountered in `comments`.
        - You can assume there will be at least one comment in `comments`
        - [<top comment>, <worst comment>]
    - `toString()`
        - Returns "This video has <number of comments> comments and is <length> long."
    - `equals()`
        - Should properly override Object's .equals().
        - A `video` is equal to another `video` if they have the same number of comments, length, likes and dislikes.

Comment.java

Videos hold comments and comments can be liked, disliked, and compared to each other. Comment should implement Comparable and Likeable.

- Instance variables:
    - `boolean isSpam`
        - Represents whether a comment was spam, true if it was.
    - `int likes`
        - A count of how many likes a comment has.
        - The number of likes should never be negative.
    - `int dislikes`

        - A count of how many dislikes a comment has.
        - The number of dislikes should never be negative.
    - `String value`
        - Represents the text in a comment.
        - Value should never be null.
    - Getter and setter methods for each instance variable. For setters, note that if an invalid value is passed in, you should not modify the value of the instance variable.
- Constructor:
    - Should have a comment constructor that accepts a number of likes, dislikes, and String value, and whether the value is spam
        - If an invalid value is passed in for `likes` or `dislikes`, simply default to 0.
        - If an invalid value is passed in for `value`, simply default to an empty String.
- Methods:
    - `compareTo();`
        - Comments should be compared based on their like factor
        - Like factor is the number of likes minus the number of dislikes
    - `toString();`
        - Returns "<value>, <likes> likes, Spam: <isSpam>."
    - `equals()`
        - Should properly override Object's .equals()
        - A `comment` is equal to another `comment` if they have the same number of likes and dislikes, have the same value, and `isSpam` is the same.

YouTube.java

YouTube, your driver class, should have the following:

- Methods:
    - `main();`
        - Create two video objects.
        - Create two comment objects.
        - Call all the methods of the video class at least once.

Submitting

To submit, upload the files listed below to the corresponding assignment on Gradescope:

- Likeable.java
- Video.java
- Comment.java
- Youtube.java

Make sure you see the message stating, "Homework 5 submitted successfully." We will only grade your last submission be sure to **submit *every file* each time you resubmit.**

Checkstyle

You must run Checkstyle on your submission. (To learn more about Checkstyle, examine our course's external website Java Resources page under "CS 1331 Style Guide".) The Checkstyle cap for this assignment is **20 points.** If you don't have Checkstyle yet, download it from https://faculty.cc.gatech.edu/~stasko/1331/java.html. Place it in the same folder as the files you want to run Checkstyle on. Run checkstyle on your code like so:

```
$ java -jar checkstyle-8.28.jar yourFileName.java Starting
audit...
Audit done.
```

The message above means there were no Checkstyle errors. If you had any errors, they would show up above this message, and the number at the end would be the points we would take off (limited by the Checkstyle cap mentioned above). The Java source files we provide contain no Checkstyle errors. In future homeworks we will be increasing this cap, so get into the habit of fixing these style errors early!

Import Restrictions

You may only import the ArrayList class.

Feature Restrictions

There are a few features and methods in Java that overly simplify the concepts we are trying to teach or break our autograder. For that reason, do not use any of the following in your final submission:

- var (the reserved keyword)
- System.exit

Collaboration

No collaboration is allowed on this assignment. See syllabus for more details.
In addition, note that it is not allowed to upload your code to any sort of public repository. This could be considered an Honor Code violation, even if it is after the homework is due. Only post code on Ed Discussion in a **private** post.

Important Notes (Don't Skip)

- Non-compiling files will receive a 0 for all associated rubric items
- Do not submit .class files.
- Test your code in addition to the basic checks on Gradescope
- Submit every file each time you resubmit
- Read the "Import Restrictions" and "Feature Restrictions" to avoid losing points
- Check on Ed Discussion for all official clarifications
- Make sure to test your program manually based on the assignment instructions and expected outputs.

The Gradescope autograder visible to you is **NOT** comprehensive. A few test cases will be hidden when you submit. This is done intentionally so that you learn to write your own test cases for your assignments.