PseudovideoMetadataEditor (2.0)

Kcits970

January 4, 2022

Sections

Section 1. Description

The purpose of the program 'PseudovideoMetadataEditor' is to simulate an online video database. It provides an event-driven interface to add, delete, edit, load, and save pseudovideos.

What's a 'Pseudovideo'?

A pseudovideo is a virtual object that only contains the metadata of a video. To put it simply, pseudovideos are 'fake' videos. Pseudovideos do not contain any information regarding the actual video data[1]. Instead, pseudovideos only contain 5 fields: video title, name of uploader, number of views, number of likes, and number of dislikes.

<Pseudovideo>

```
String title
String uploader
int views
int likes
int dislikes
```

I mentioned above that the purpose of this program is to **simulate**[2] an online video database. Because I'm only interested in the simulation, there's no need for me to use actual video files. Any object that can contain some fields of metadata is enough to satisfy my purpose. I used the terminology 'pseudovideo' to define such objects. This definition is useful, because it explicitly distinguishes its difference from real videos.

Extra Notes

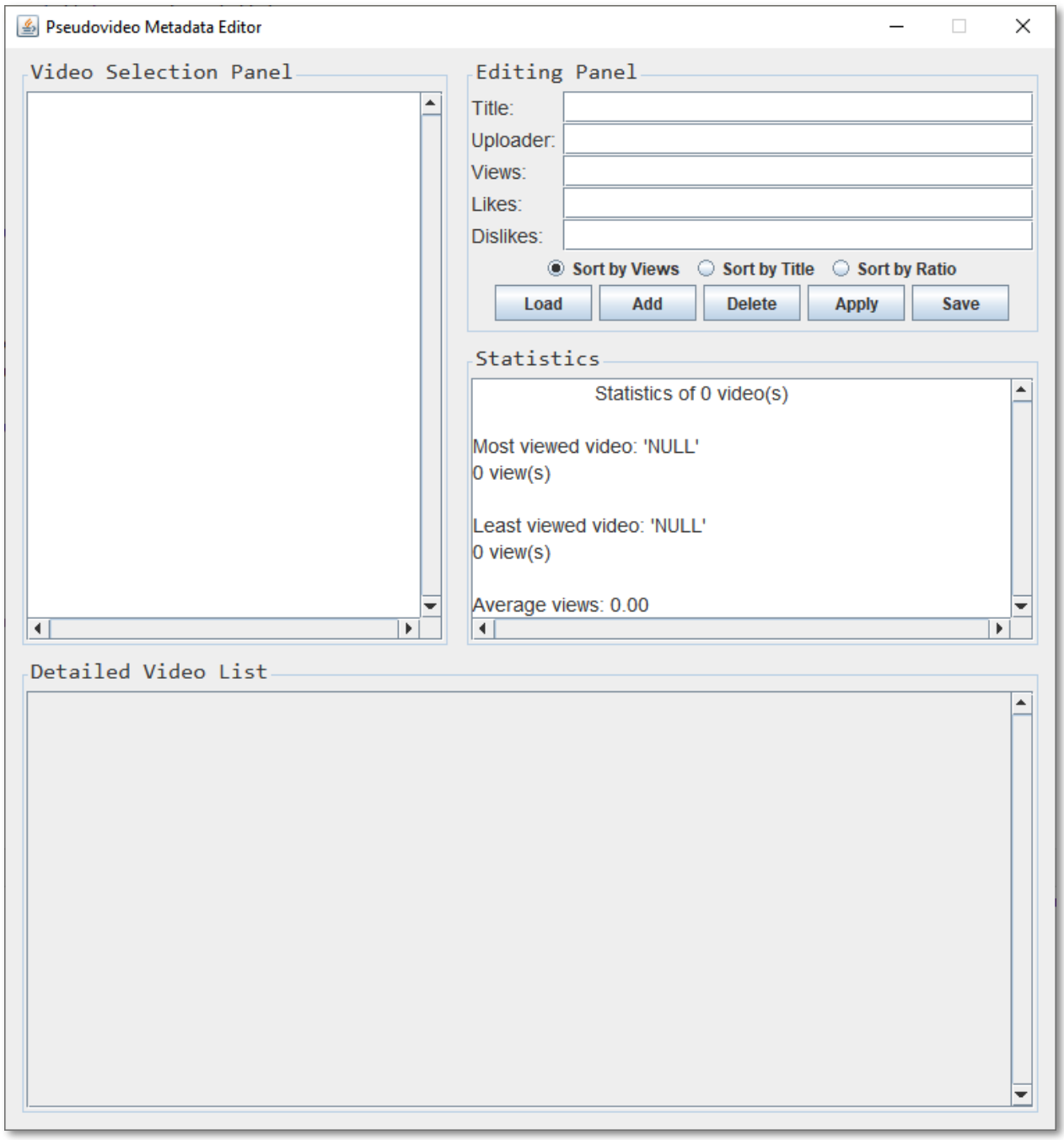
The program is written in Java language, and compiled using Java Compiler 15.0.1. Attempts to compile the given source code with a different compiler may result in unexpected or undefined behavior.

---

[1] compression type, frames per second, color of each pixel, sounds, etc.
[2] simulate (verb): imitate the appearance or character of.

Section 2. Features

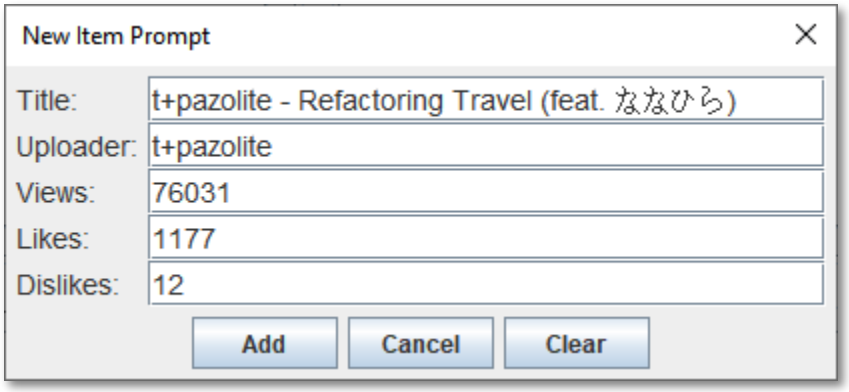
When the program is launched, an empty frame will be shown. From here, the user can perform various actions such as adding new videos, deleting existing videos, writing to an external file, etc.



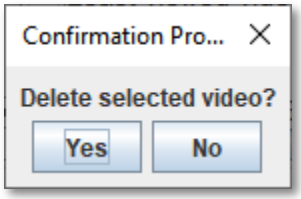
(The initial frame at program launch.)

Adding/Deleting Videos

To add a new video to the list, click the [Add] button in the [Editing Panel]. A prompt containing 5 textfields will appear. Simply enter the video metadata[3] into the fields and click the [Add] button.



As opposed to adding, the video can also be removed from the list. To delete an existing video, select the desired video from the [Video Selection Panel], and click the [Delete] button. A prompt asking for confirmation will appear. Click the [Yes] button to confirm the deletion.

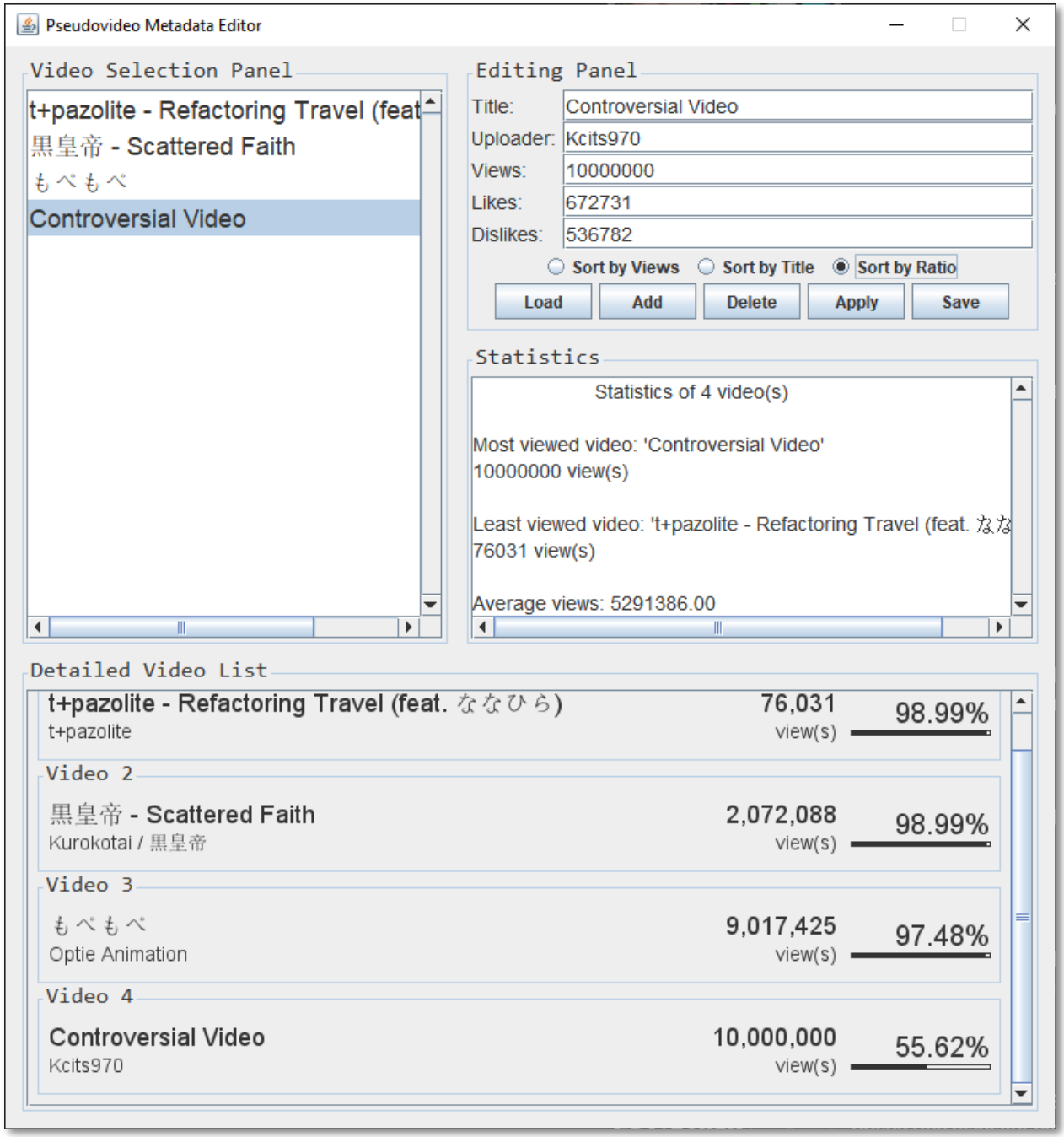


Editing/Sorting Videos

The process of editing or sorting is quite straightforward. To edit a video, select any video from the [Video Selection Panel] and modify the values of the textfields in the [Editing Panel]. The changes will only apply if the user clicks the [Apply] button. If any other action is performed without applying changes, the entered information will be lost.

[3] The metadata doesn't necessarily have to be from a real video. This program is for simulation purposes only, so any inputs are completely fine.

The user may also want to sort the list in a different order. Simply click any one of the radio buttons to sort the list in a specific order.

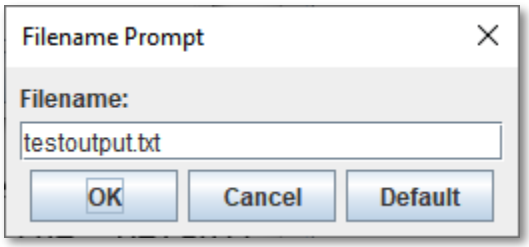


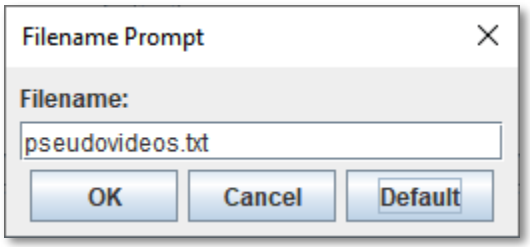
(The frame after adding 4 videos and sorting them by ratio[4].)

[4] 'Ratio' refers to the like/dislike ratio in terms of likes.

Saving/Loading Videos

The program also supports loading and saving videos to an external file. To save the current status, click the [Save] button in the [Editing Panel], which the program responds with a filename prompt. From here, the user can enter a desired filename, or use the default filename by clicking the [Default] button[5].

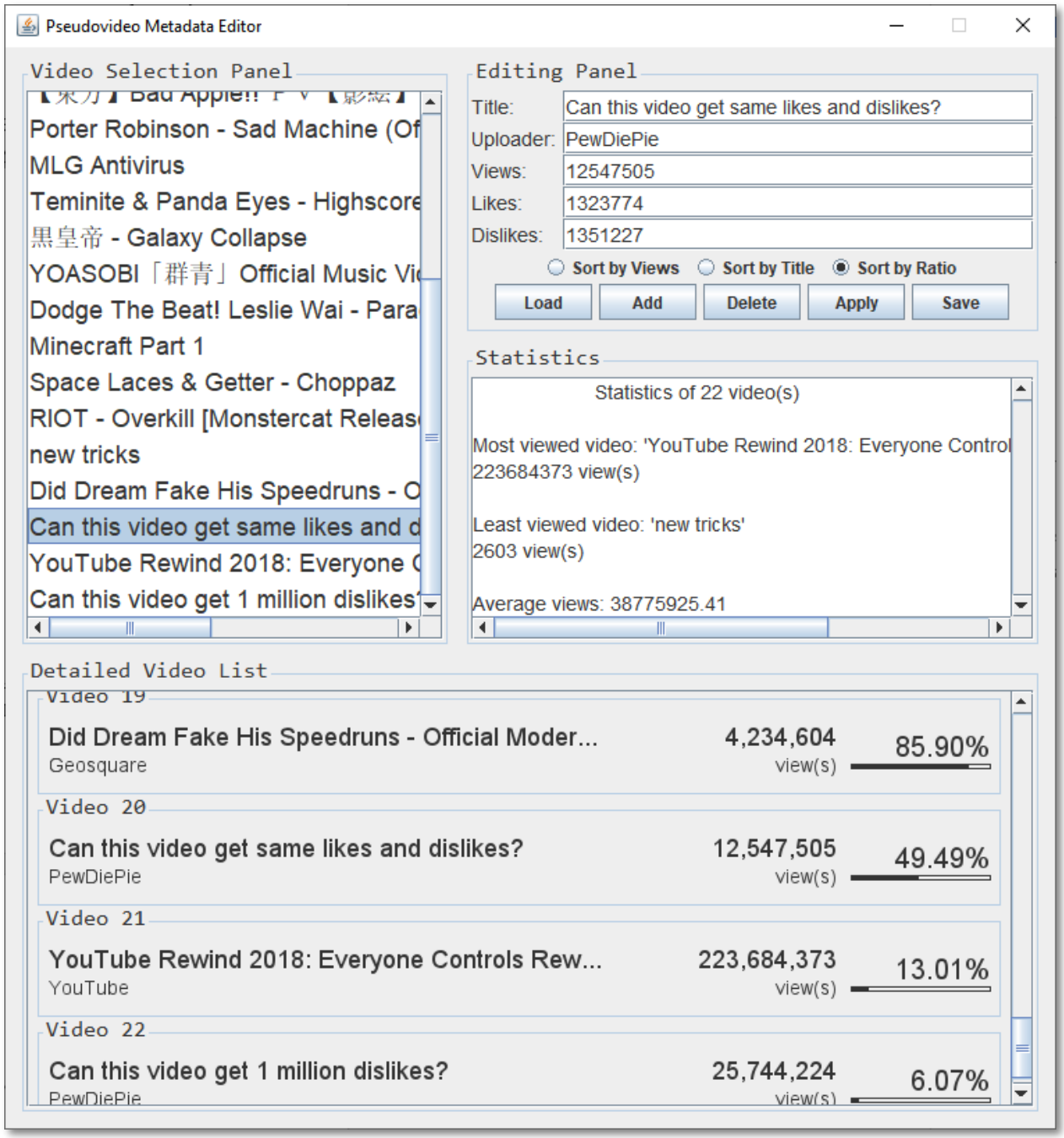




Click the [OK] button to save the file. If successfully saved, the file with the inputted name will appear in the directory of launch.

Loading videos from a file is very similar to saving. Simply click the [Load] button, and follow the same process. If the file-IO succeeds, the list should update with the contents of the specified file.

[5] Clicking the [Default] button will simply put “pseudovideos.txt” into the textfield. It does not automatically close the prompt and save to the default location.



(The frame after loading videos from the default location and sorting them by ratio.)

Section 3. File Format

The files that are used to save/load videos all follow a specific syntax called PMML[6]. Here's an example of one.

```
%pseudovideo%
%title%Space Laces &amp Getter - Choppaz
%uploader%UKF Dubstep
%views%502969
%likes%12228
%dislikes%248
%pseudovideo%
%title%Dodge The Beat! Leslie Wai - Paradigm [a void and ESC ape] 0.00&per 1x100
%uploader%R eg
%views%100470
%likes%2247
%dislikes%38
%pseudovideo%
%title%Camellia - Towards The Horizon
%uploader%Skyward
%views%73338
%likes%1700
%dislikes%3
```

PMML syntax is based on XML syntax, but it's much more simplified. Similar to XML, PMML consists of tags and attribute values. PMML tags are surrounded by '%', and attribute values are placed right after the tags. Because '%' is a special character, it cannot be directly used in attribute values. Therefore, '%' characters are replaced with "&per", and '&' characters are replaced with "&amp" when used to describe attribute values. Furthermore, newline characters cannot be used in attribute values[7].

Only 6 tags are used in PMML syntax. Below is a list of all 6.

pseudovideo	Marks the start of a pseudovideo block
title	Indicates the video title
uploader	Indicates the name of the uploader
views	Indicates the number of views
likes	Indicates the number of likes
dislikes	Indicates the number of dislikes

[6] Pseudovideo Metadata Markup Language

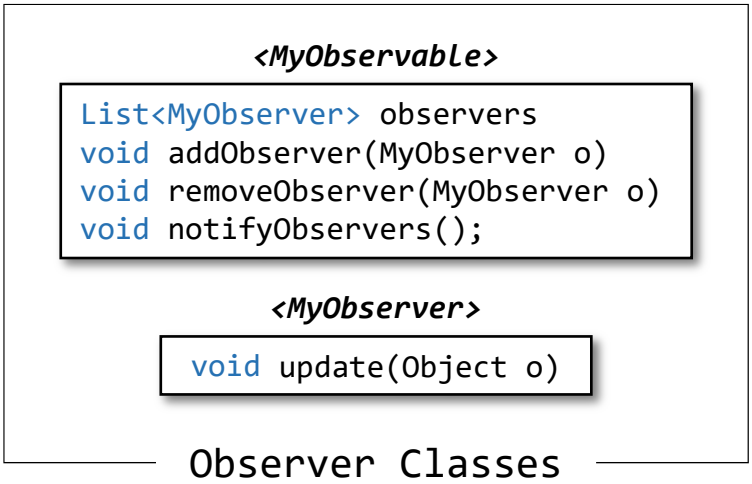
[7] The reason for this is purely to keep the parsing process simple. Allowing newline characters may introduce ambiguous syntax that cannot be easily understood by the parser.

If the user desires to manually type the values in the files, the following set of rules must be followed.

Rule 1	All 6 tags must be used sequentially.
Rule 2	Tags must always begin after a newline
Rule 3	'%' or '&' characters in attribute values must be properly replaced.

Section 4. Design Pattern

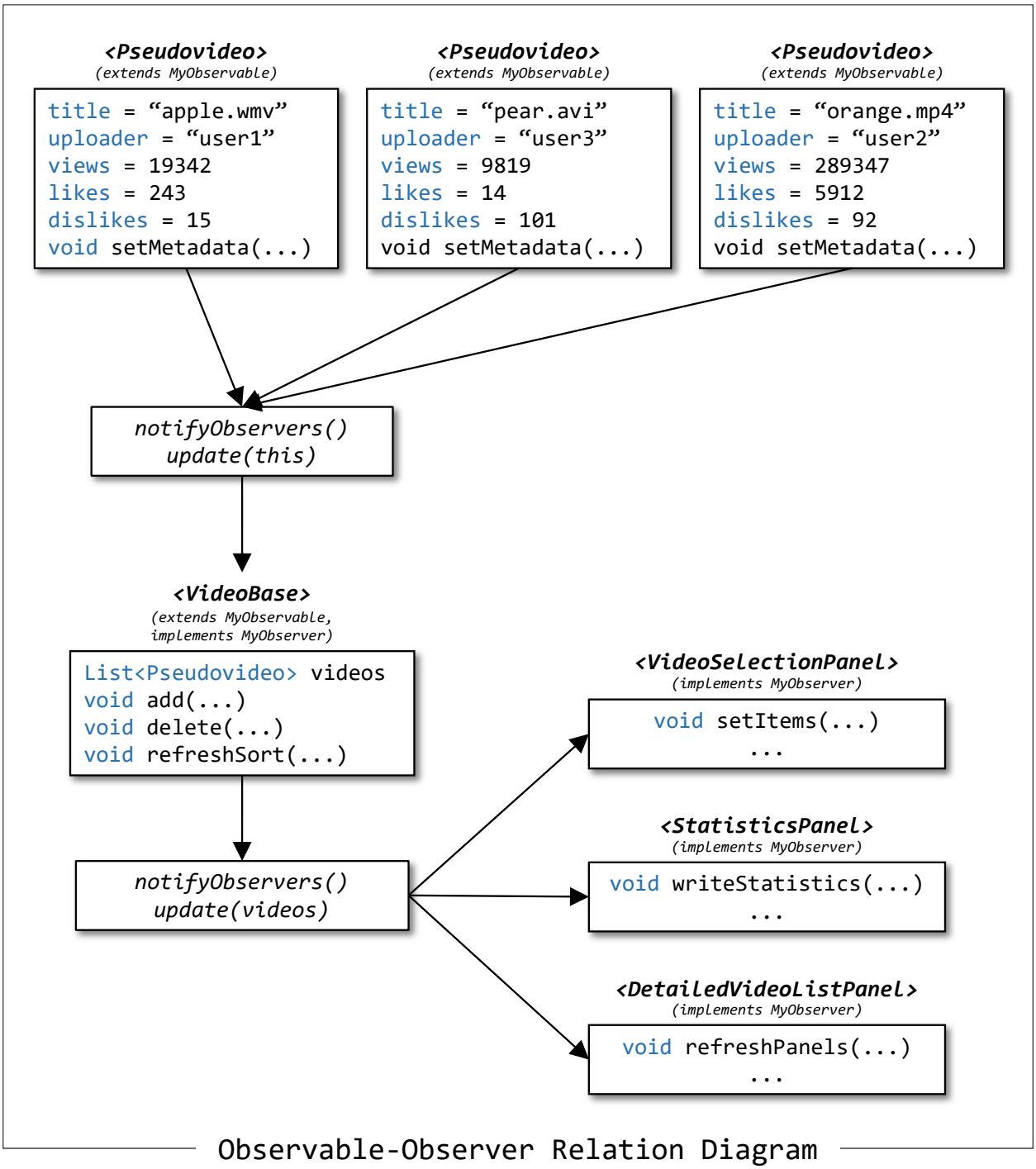
This program follows the observer pattern. I've implemented two classes to follow this design pattern[8].



In the case of PseudovideoMetadataEditor, the frame classes need to refresh themselves every time the video base[9] changes. The changes include: adding/deleting videos, changing the sorting order of videos, and changing the metadata of a video. This means that the registered observers need to be notified not only when changes are made to the base, but also when changes are made to individual videos. The diagram in the next page illustrates how I managed to resolve this issue.

[8] Since the release of Java 9, the observer classes from java.util have been deprecated. To go around this issue, I simply rewrote the classes on my own. MyObservable replaces java.util.Observable, and MyObserver replaces java.util.Observer.

[9] A video base refers to a collection of all videos. It is basically a video database.



Observable-Observer Relation Diagram

The class VideoBase is registered as the observer for each Pseudovideo in its list. The frame panel classes are registered as the observer for VideoBase. The following methods:

```
Pseudovideo.setMetadata(Iterator<String> metadata)
VideoBase.add(Pseudovideo video)
```

```
VideoBase.delete(Pseudovideo video)
VideoBase.refreshSort(Comparator<? super Pseudovideo> comparator)
```

notify the observers, which allows the frame to display the correct updated data from the base.