# Assignment-2

**Assignment 2**

**Due date**

- 11.59 PM EST, June 24th

**Git url**

- https://classroom.github.com/a/jTL4tq8Y

Submit your code as per the provided instructions.

**Assignment Goal**

Implement the State Pattern to capture the project requirements.

**Team Work**

- No team work is allowed. Work individually. You cannot discuss the assignment with ANYONE other than the instructor and TA.

**Programming Language**

You are required to program using Java.

**Compiling and Running Commands**

- Compilation: Your code should compile on remote.cs.binghamton.edu with the following command: ant -buildfile src/build.xml all
- Running the code: Your code should run on remote.cs.binghamton.edu with the following command: ant -buildfile src/build.xml run -Dinput="<path/to/inputfile>" -Doutput="<path/to/outputfile>"

**Policy on sharing of code**

E[illegible]RY line of code that you submit in this assignment should be written by you. Do NOT show your code to an[illegible]other student. Do not copy any code from any online source. Code for File I/O or String operations, if found online, should be clearly cited, and you cannot use more than 5 lines of such online code.

Code snippets, for File I/O, if used from an online source should be cited by mentioning it in the README.md and also in the documentation of every source file in which that code appears.

Post to the piazza if you have any questions about the requirements. Sending an email will only get a response requesting you to post on piazza. DO NOT post your code to piazza asking for help with debugging.

**Project Description**

Design and Implement a program for Youtube to categorize a single channel based on popularity.

- Prior to working on the assignment, please read through the grading instructions to understand the minimum requirement of the assignment.
- A Youtube channel has a popularity score, defined as *The average popularity score of all videos contained in the channel*. Based on its popularity score, a channel can be in one of the following states.
  - UNPOPULAR - This is the starting state of a channel. For a channel to be in this state, its popularity score should in the range [0, 1000].
  - MILDLY_POPULAR - For a channel to be in this state, its popularity score should be in the range (1000, 10000].
  - HIGHLY_POPULAR - For a channel to be in this state, its popularity score should be in the range (10000, 100000].
  - ULTRA_POPULAR - For a channel to be in this state, its popularity score should be in the range (100000, INT_MAX].

  *Note: () represents an open interval (not including end points) and [] represents a closed interval (including endpoints). For example, $X \in (10, 20]$ iff $X \in R$ AND $10 < X \leq 20$.*
  *Note: The popularity score of a video should not be negative. Therefore, popularity score of a video should be set to 0 if it is negative.*
  *Question to ask yourself at this point - Is Channel a state or a context?*
  *Enumerate the names of the states.*
- A channel can hold multiple videos. The popularity score of a video is defined by the following three metrics.
  - Total number of views (≥ 0).
  - Total number of likes (≥ 0).
  - Total number of dislikes (≥ 0).

  Using the above information, the popularity score of a video at any point is given by the formula ***#Views + 2 * (#Likes - #Dislikes)*** where # signifies total count so far. The data for number of views, likes and dislikes are provided in the input file.
  *Question to ask yourself at this point - How and where are videos stored?*
- The current state of the channel determines whether advertisement requests are approved or rejected based on how long they are. The following are the rules for the same.
  - When state=UNPOPULAR, advertisements of length in range (1,10] are approved and the rest are rejected.
  - When state=MILDLY_POPULAR, advertisements of length in range (1, 20] are approved and the rest are rejected.
  - When state=HIGHLY_POPULAR, advertisements of length in range (1, 30] are approved and the rest are rejected.
  - When state=ULTRA_POPULAR, advertisements of length in range (1, 40] are approved and the rest are rejected.

  The requests to add advertisements of a certain length are also provided via the input file.
  *Question to ask yourself at this point - Where are the advertisement requests processed?*

Input Processing and Control flow

- Adding a video to the channel. Input format: **ADD_VIDEO::<video name>**.
- Removing a video from the channel. Input format: **REMOVE_VIDEO::<video name>**.
- Views, Likes and Dislikes. Input format: **METRICS__<video name>::[VIEWS=<delta in #views>,LIKES=<delta in #likes>,DISLIKES=<delta in #dislikes>]**.
  *Note: There are no spaces before or after the comma character.*
  *Note: Views, Likes and Dislikes MUST be integers.*
- Advertisement requests. Input format: **AD_REQUEST__<video name>::LEN=<length>**.
  *Note: Advertisement length MUST be an integer.*

3. ADD_VIDEO - If the video already exists or the format of the input is invalid, throw the appropriate exception reporting a meaningful error message and terminate. If the input is valid, then add the video to the collection of videos in the channel. This new video is assigned an initial popularity score of 0. This operation is performed by the current state.
   The processing of this line should result in the string **<current state name>__VIDEO_ADDED::<video name>** being written to Results.

4. REMOVE_VIDEO - If the video does not exist or the format of the input is invalid, throw an appropriate exception reporting a meaningful error message and terminate. If the input is valid, then remove this video from the collection of videos in the channel. The channel's popularity score will need to be updated after the removal. This operation is performed by the current state.
   The processing of this line should result in the string **<current state name>__VIDEO_REMOVED::<video name>** being written to Results.

5. METRICS - If the video does not exist or the format of the input is invalid, throw an appropriate exception reporting a meaningful error message and terminate. If the input is valid, then update the metrics of the corresponding video and recalculate the popularity score of the channel. This operation is performed by the current state. The metric data is provided as delta changes.

   Consider the input line METRICS__testvideo::[VIEWS=10,LIKES=10,DISLIKES=-20]. This means that the total views increased by 10, total likes increased by 10 and the total dislikes decreased by 20, for the video named 'testvideo'.

   The processing of this line should result in the string **<current state name>__POPULARITY_SCORE_UPDATE::<new popularity score of channel>** being written to Results.
   *Note: unlike likes and dislikes, the delta in the number of views cannot be negative.*
   *Note: If there is a decrease in the number of likes or dislikes, it cannot be more than the total number of likes or dislikes received thus far for the video.*

6. AD_REQUEST - If the video does not exist or the length is negative, throw an appropriate exception reporting a meaningful error message and terminate. If the input is valid, report if the request is approved or rejected based on whether the length of the advertisement falls within the current state's acceptable ad length range (mentioned above).
   The processing of this should result in the string **<current state name>__AD_REQUEST::<APPROVED / REJECTED>** being written to Results.

7. If any of the lines in the input file are invalid, then the whole input is to be considered invalid.
8. Upon processing of a ADD_VIDEO, REMOVE_VIDEO or METRICS input if the new popularity score of the channel falls in a range acceptable by a state other than the current state then a state change occurs.

ⓘ Once all the lines of the input file have been processed cast the results instance to the appropriate interface and call the necessary method to persist the results to the output file.

will be provided using the following command-line options.

- ***-Dinput***: Path to the Input file.
- ***-Doutput***: Path to the output file.

**EXAMPLES**

*input*

```
ADD_VIDEO::video1
ADD_VIDEO::video2
METRICS__video1::[VIEWS=1000,LIKES=20,DISLIKES=20]
AD_REQUEST__video1::LEN=8
METRICS__video2::[VIEWS=2000,LIKES=400,DISLIKES=20]
METRICS__video1::[VIEWS=20000,LIKES=1000,DISLIKES=-10]
AD_REQUEST__video2::LEN=39
METRICS__video2::[VIEWS=50,LIKES=-50,DISLIKES=0]
REMOVE_VIDEO::video2
ADD_VIDEO::video3
METRICS__video3::[VIEWS=2000,LIKES=100,DISLIKES=20]
METRICS__video1::[VIEWS=0,LIKES=-1000,DISLIKES=500]
ADD_VIDEO::video4
METRICS__video4::[VIEWS=100,LIKES=5,DISLIKES=0]
REMOVE_VIDEO::video1
AD_REQUEST__video3::LEN=15
REMOVE_VIDEO::video3
REMOVE_VIDEO::video4
```

*output*

```
UNPOPULAR__VIDEO_ADDED::video1
UNPOPULAR__VIDEO_ADDED::video2
UNPOPULAR__POPULARITY_SCORE_UPDATE::500
UNPOPULAR__AD_REQUEST::APPROVED
UNPOPULAR__POPULARITY_SCORE_UPDATE::1880
MILDLY_POPULAR__POPULARITY_SCORE_UPDATE::12890
HIGHLY_POPULAR__AD_REQUEST::REJECTED
HIGHLY_POPULAR__POPULARITY_SCORE_UPDATE::12865
HIGHLY_POPULAR__VIDEO_REMOVED::video2
HIGHLY_POPULAR__VIDEO_ADDED::video3
HIGHLY_POPULAR__POPULARITY_SCORE_UPDATE::12590
HIGHLY_POPULAR__POPULARITY_SCORE_UPDATE::11090
HIGHLY_POPULAR__VIDEO_ADDED::video4
MILDLY_POPULAR__POPULARITY_SCORE_UPDATE::7430
MILDLY_POPULAR__VIDEO_REMOVED::video1
MILDLY_POPULAR__AD_REQUEST::APPROVED
MILDLY_POPULAR__VIDEO_REMOVED::video3
UNPOPULAR__VIDEO_REMOVED::video4
```

clarification on the assignment or finding ambiguities, and also to those who respond and participate to help make the discussion interesting and informative.
- Class participation points will be given to students who also post sample interesting input examples with corresponding output and metrics.

**Sample Input Files sent by students in this course**

Please check piazza.

**Compiling and Running Java code**

- Your submission must include a readme in markdown format with the name **README.md**.
- Your README.md file should have the following information:
  - instructions on how to compile the code
  - instructions on how to run the code
  - justification for the choice of data structures (in terms of time and/or space complexity).
  - citations for external material utilized.
- You should have the following directory structure (replace username with your github username).
- ./csx42-summer-2020-assign2-username
- ./csx42-summer-2020-assign2-username/README.md
- ./csx42-summer-2020-assign2-username/.gitignore
- ./csx42-summer-2020-assign2-username/channelpopularity
- ./csx42-summer-2020-assign2-username/channelpopularity/src
- ./csx42-summer-2020-assign2-username/channelpopularity/src/build.xml
- ./csx42-summer-2020-assign2-username/channelpopularity/src/channelpopularity
- ./csx42-summer-2020-assign2-username/channelpopularity/src/channelpopularity/operation
- ./csx42-summer-2020-assign2-username/channelpopularity/src/channelpopularity/operation/Operation.java
- ./csx42-summer-2020-assign2-username/channelpopularity/src/channelpopularity/context
- ./csx42-summer-2020-assign2-username/channelpopularity/src/channelpopularity/context/ContextI.java
- ./csx42-summer-2020-assign2-username/channelpopularity/src/channelpopularity/state
- ./csx42-summer-2020-assign2-username/channelpopularity/src/channelpopularity/state/StateName.java
- ./csx42-summer-2020-assign2-username/channelpopularity/src/channelpopularity/state/StateI.java
- ./csx42-summer-2020-assign2-username/channelpopularity/src/channelpopularity/state/AbstractState.java
- ./csx42-summer-2020-assign2-username/channelpopularity/src/channelpopularity/state/factory
- ./csx42-summer-2020-assign2-username/channelpopularity/src/channelpopularity/state/factory/SimpleStateFactoryI.java
- ./csx42-summer-2020-assign2-username/channelpopularity/src/channelpopularity/state/factory/SimpleStateFactory.java
- ./csx42-summer-2020-assign2-username/channelpopularity/src/channelpopularity/util
- ./csx42-summer-2020-assign2-username/channelpopularity/src/channelpopularity/util/FileProcessor.java
- ./csx42-summer-2020-assign2-username/channelpopularity/src/channelpopularity/util/Results.java