package com.google.api.services.samples.youtube.cmdline.youtube\_cmdline\_channelbulletin\_sample;  
  
import java.io.File;  
import java.util.Calendar;  
import java.util.List;  
  
import com.google.api.client.auth.oauth2.Credential;  
import com.google.api.client.extensions.java6.auth.oauth2.AuthorizationCodeInstalledApp;  
import com.google.api.client.extensions.java6.auth.oauth2.FileCredentialStore;  
import com.google.api.client.extensions.jetty.auth.oauth2.LocalServerReceiver;  
import com.google.api.client.googleapis.auth.oauth2.GoogleAuthorizationCodeFlow;  
import com.google.api.client.googleapis.auth.oauth2.GoogleClientSecrets;  
import com.google.api.client.googleapis.json.GoogleJsonResponseException;  
import com.google.api.client.http.HttpTransport;  
import com.google.api.client.http.javanet.NetHttpTransport;  
import com.google.api.client.json.JsonFactory;  
import com.google.api.client.json.jackson2.JacksonFactory;  
import com.google.api.services.youtube.YouTube;  
import com.google.api.services.youtube.model.Activity;  
import com.google.api.services.youtube.model.ActivityContentDetails;  
import com.google.api.services.youtube.model.ActivityContentDetails.Bulletin;  
import com.google.api.services.youtube.model.ActivitySnippet;  
import com.google.api.services.youtube.model.Channel;  
import com.google.api.services.youtube.model.ChannelListResponse;  
import com.google.api.services.youtube.model.ResourceId;  
import com.google.common.collect.Lists;  
  
/\*\*  
 \* Creates a video bulletin that is posted to the user's channel feed.  
 \*  
 \* @author Jeremy Walker  
 \*/  
public class ChannelBulletin {  
  
  /\*\* Global instance of the HTTP transport. \*/  
  private static final HttpTransport HTTP\_TRANSPORT = new NetHttpTransport();  
  
  /\*\* Global instance of the JSON factory. \*/  
  private static final JsonFactory JSON\_FACTORY = new JacksonFactory();  
  
  /\*\* Global instance of YouTube object to make all API requests. \*/  
  private static YouTube youtube;  
  
  /\*  
   \* Global instance of the video id we want to post as a bulletin into our channel feed. You will  
   \* probably pull this from a search or your app.  
   \*/  
  private static String VIDEO\_ID = "L-oNKK1CrnU";  
  
  /\*\*  
   \* Authorizes the installed application to access user's protected data.  
   \*  
   \* @param scopes list of scopes needed to run upload.  
   \*/  
  private static Credential authorize(List<String> scopes) throws Exception {  
  
    // Load client secrets.  
    GoogleClientSecrets clientSecrets = GoogleClientSecrets.load(  
        JSON\_FACTORY, ChannelBulletin.class.getResourceAsStream("/client\_secrets.json"));  
  
    // Checks that the defaults have been replaced (Default = "Enter X here").  
    if (clientSecrets.getDetails().getClientId().startsWith("Enter")  
        || clientSecrets.getDetails().getClientSecret().startsWith("Enter ")) {  
      System.out.println(  
          "Enter Client ID and Secret from https://console.developers.google.com/project/\_/apiui/credential"  
          + "into youtube-cmdline-channelbulletin-sample/src/main/resources/client\_secrets.json");  
      System.exit(1);  
    }  
  
    // Set up file credential store.  
    FileCredentialStore credentialStore = new FileCredentialStore(  
        new File(System.getProperty("user.home"), ".credentials/youtube-api-channelbulletin.json"),  
        JSON\_FACTORY);  
  
    // Set up authorization code flow.  
    GoogleAuthorizationCodeFlow flow = new GoogleAuthorizationCodeFlow.Builder(  
        HTTP\_TRANSPORT, JSON\_FACTORY, clientSecrets, scopes).setCredentialStore(credentialStore)  
        .build();  
  
    // Build the local server and bind it to port 8080  
    LocalServerReceiver localReceiver = new LocalServerReceiver.Builder().setPort(8080).build();  
  
    // Authorize.  
    return new AuthorizationCodeInstalledApp(flow, localReceiver).authorize("user");  
  }  
  
  /\*\*  
   \* Authorizes user, runs Youtube.Channnels.List to get the default channel, and posts a bulletin  
   \* with a video id to the user's default channel.  
   \*  
   \* @param args command line args (not used).  
   \*/  
  public static void main(String[] args) {  
  
    // Scope required to upload to YouTube.  
    List<String> scopes = Lists.newArrayList("https://www.googleapis.com/auth/youtube");  
  
    try {  
      // Authorization.  
      Credential credential = authorize(scopes);  
  
      // YouTube object used to make all API requests.  
      youtube = new YouTube.Builder(HTTP\_TRANSPORT, JSON\_FACTORY, credential).setApplicationName(  
          "youtube-cmdline-channelbulletin-sample").build();  
  
      /\*  
       \* Now that the user is authenticated, the app makes a channel list request to get the  
       \* authenticated user's channel. https://developers.google.com/youtube/v3/docs/channels/list  
       \*/  
      YouTube.Channels.List channelRequest = youtube.channels().list("contentDetails");  
      channelRequest.setMine("true");  
      /\*  
       \* Limits the results to only the data we need making your app more efficient.  
       \*/  
      channelRequest.setFields("items/contentDetails");  
      ChannelListResponse channelResult = channelRequest.execute();  
  
      /\*  
       \* Gets the list of channels associated with the user.  
       \*/  
      List<Channel> channelsList = channelResult.getItems();  
  
      if (channelsList != null) {  
        // Gets user's default channel id (first channel in list).  
        String channelId = channelsList.get(0).getId();  
  
        /\*  
         \* We create the snippet to set the channel we will post to and the description that goes  
         \* along with our bulletin.  
         \*/  
        ActivitySnippet snippet = new ActivitySnippet();  
        snippet.setChannelId(channelId);  
        Calendar cal = Calendar.getInstance();  
        snippet.setDescription("Bulletin test video via YouTube API on " + cal.getTime());  
  
        /\*  
         \* We set the kind of the ResourceId to video (youtube#video). Please note, you could set  
         \* the type to a playlist (youtube#playlist) and use a playlist id instead of a video id.  
         \*/  
        ResourceId resource = new ResourceId();  
        resource.setKind("youtube#video");  
        resource.setVideoId(VIDEO\_ID);  
  
        Bulletin bulletin = new Bulletin();  
        bulletin.setResourceId(resource);  
  
        // We construct the ActivityContentDetails now that we have the Bulletin.  
        ActivityContentDetails contentDetails = new ActivityContentDetails();  
        contentDetails.setBulletin(bulletin);  
  
        /\*  
         \* Finally, we construct the activity we will write to YouTube via the API. We set the  
         \* snippet (covers description and channel we are posting to) and the content details  
         \* (covers video id and type).  
         \*/  
        Activity activity = new Activity();  
        activity.setSnippet(snippet);  
        activity.setContentDetails(contentDetails);  
  
        /\*  
         \* We specify the parts (contentDetails and snippet) we will write to YouTube. Those also  
         \* cover the parts that are returned.  
         \*/  
        YouTube.Activities.Insert insertActivities =  
            youtube.activities().insert("contentDetails,snippet", activity);  
        // This returns the Activity that was added to the user's YouTube channel.  
        Activity newActivityInserted = insertActivities.execute();  
  
        if (newActivityInserted != null) {  
          System.out.println(  
              "New Activity inserted of type " + newActivityInserted.getSnippet().getType());  
          System.out.println(" - Video id "  
              + newActivityInserted.getContentDetails().getBulletin().getResourceId().getVideoId());  
          System.out.println(  
              " - Description: " + newActivityInserted.getSnippet().getDescription());  
          System.out.println(" - Posted on " + newActivityInserted.getSnippet().getPublishedAt());  
        } else {  
          System.out.println("Activity failed.");  
        }  
  
      } else {  
        System.out.println("No channels are assigned to this user.");  
      }  
    } catch (GoogleJsonResponseException e) {  
      e.printStackTrace();  
      System.err.println("There was a service error: " + e.getDetails().getCode() + " : "  
          + e.getDetails().getMessage());  
  
    } catch (Throwable t) {  
      t.printStackTrace();  
    }  
  }  
}