**package** com.lshs.mytest;  
  
**import** org.junit.Test;  
  
**import** javax.sound.sampled.\*;  
**import** java.io.File;  
**import** java.io.IOException;  
  
*/\*\*  
 \** ***@Description:*** *音频文件格式转换  
 \** ***@author:*** *LuShao  
 \** ***@create:*** *2018-08-21 17:19  
 \*\*/***public class** SoundUtils {  
  
 */\*\*  
 \* 得到音频文件的格式  
 \** ***@param path*** *\** ***@return*** *\** ***@throws*** *IOException  
 \** ***@throws*** *UnsupportedAudioFileException  
 \*/* **public** String getVoiceFormat(String path) **throws** IOException, UnsupportedAudioFileException {  
 File file=**new** File(path);  
 AudioInputStream ais= AudioSystem.*getAudioInputStream*(file);  
 AudioFormat format = ais.getFormat();  
 String s = format.toString();  
 **return** s;  
 }  
  
 @Test  
 **public void** demo1() **throws** IOException, UnsupportedAudioFileException {  
 String path=**"D:\\wav1.wav"**;  
 String voiceFormat = getVoiceFormat(path);  
 System.***out***.println(voiceFormat);  
 }  
  
 */\*\*  
 \* wav文件转换成pcm格式文件  
 \** ***@param path*** *\*/* **public void** wavToPcm(String path) **throws** IOException, UnsupportedAudioFileException {  
 File file=**new** File(path);  
 AudioInputStream format = AudioSystem.*getAudioInputStream*(file);  
*// 此处的转换必须是16bit的音频文件* AudioInputStream audioInputStream = AudioSystem.*getAudioInputStream*(AudioFormat.Encoding.***ULAW***, format);  
 String s = path.substring(0, path.lastIndexOf(**"."**) + 1) + **"pcm"**;  
*// 开始写入文件* AudioSystem.*write*(audioInputStream,AudioFileFormat.Type.***WAVE***,**new** File(s));  
 }  
  
 @Test  
 **public void** demo2() **throws** IOException, UnsupportedAudioFileException {  
 String path=**"D:\\wav1.wav"**;  
 wavToPcm(path);  
 }  
}

**package** com.lshs.mytest;  
  
**import** com.jacob.activeX.ActiveXComponent;  
**import** com.jacob.com.ComThread;  
**import** com.jacob.com.Dispatch;  
**import** com.jacob.com.Variant;  
**import** org.junit.Test;  
  
**import** java.io.File;  
  
*/\*\*  
 \** ***@Description:*** *文字转换成语音MTTS jacob  
 \** ***@author:*** *LuShao  
 \** ***@create:*** *2018-08-21 14:41  
 \*\*/***public class** TxtConvertVoice {  
 **private int volume**=100;*//声音1~100* **private int rate**=-2;*//频率 -10~10* **private int voice**=0;*//语音库序号* **private int audio**=0;*//输出设备序号* **private** ActiveXComponent **ax**=**null**;  
 **private** Dispatch **spVoice**=**null**;*//声音对象* **private** Dispatch **spFileStream**=**null**;*//音频文件输出流对象，读取或保存时使用* **private** Dispatch **spAudioFormat**=**null**;*//音频格式对象* **private** Dispatch **spMMAudioOut**=**null**;*//音频输出对象* **private int formatType**=22;*//音频的输出格式，默认为：SAFT22kHz16BitMono* **public int** getVolume() {  
 **return volume**;  
 }  
  
 **public void** setVolume(**int** volume) {  
 **this**.**volume** = volume;  
 }  
  
 **public int** getRate() {  
 **return rate**;  
 }  
  
 **public void** setRate(**int** rate) {  
 **this**.**rate** = rate;  
 }  
  
 **public int** getVoice() {  
 **return voice**;  
 }  
  
 **public void** setVoice(**int** voice) {  
 **this**.**voice** = voice;  
 }  
  
 **public int** getAudio() {  
 **return audio**;  
 }  
  
 **public void** setAudio(**int** audio) {  
 **this**.**audio** = audio;  
 }  
  
 **public** ActiveXComponent getAx() {  
 **return ax**;  
 }  
  
 **public void** setAx(ActiveXComponent ax) {  
 **this**.**ax** = ax;  
 }  
  
 **public int** getFormatType() {  
 **return formatType**;  
 }  
  
 */\*\*  
 \* \* 设置音频输出格式类型<br>  
 \* SAFTDefault = -1<br>  
 \* SAFTNoAssignedFormat = 0<br>  
 \* SAFTText = 1<br>  
 \* SAFTNonStandardFormat = 2<br>  
 \* SAFTExtendedAudioFormat = 3<br>  
 \* // Standard PCM wave formats<br>  
 \* SAFT8kHz8BitMono = 4<br>  
 \* SAFT8kHz8BitStereo = 5<br>  
 \* SAFT8kHz16BitMono = 6<br>  
 \* SAFT8kHz16BitStereo = 7<br>  
 \* SAFT11kHz8BitMono = 8<br>  
 \* SAFT11kHz8BitStereo = 9<br>  
 \* SAFT11kHz16BitMono = 10<br>  
 \* SAFT11kHz16BitStereo = 11<br>  
 \* SAFT12kHz8BitMono = 12<br>  
 \* SAFT12kHz8BitStereo = 13<br>  
 \* SAFT12kHz16BitMono = 14<br>  
 \* SAFT12kHz16BitStereo = 15<br>  
 \* SAFT16kHz8BitMono = 16<br>  
 \* SAFT16kHz8BitStereo = 17<br>  
 \* SAFT16kHz16BitMono = 18<br>  
 \* SAFT16kHz16BitStereo = 19<br>  
 \* SAFT22kHz8BitMono = 20<br>  
 \* SAFT22kHz8BitStereo = 21<br>  
 \* SAFT22kHz16BitMono = 22<br>  
 \* SAFT22kHz16BitStereo = 23<br>  
 \* SAFT24kHz8BitMono = 24<br>  
 \* SAFT24kHz8BitStereo = 25<br>  
 \* SAFT24kHz16BitMono = 26<br>  
 \* SAFT24kHz16BitStereo = 27<br>  
 \* SAFT32kHz8BitMono = 28<br>  
 \* SAFT32kHz8BitStereo = 29<br>  
 \* SAFT32kHz16BitMono = 30<br>  
 \* SAFT32kHz16BitStereo = 31<br>  
 \* SAFT44kHz8BitMono = 32<br>  
 \* SAFT44kHz8BitStereo = 33<br>  
 \* SAFT44kHz16BitMono = 34<br>  
 \* SAFT44kHz16BitStereo = 35<br>  
 \* SAFT48kHz8BitMono = 36<br>  
 \* SAFT48kHz8BitStereo = 37<br>  
 \* SAFT48kHz16BitMono = 38<br>  
 \* SAFT48kHz16BitStereo = 39<br>  
 \* <br>  
 \* // TrueSpeech format<br>  
 \* SAFTTrueSpeech\_8kHz1BitMono = 40<br>  
 \* // A-Law formats<br>  
 \* SAFTCCITT\_ALaw\_8kHzMono = 41<br>  
 \* SAFTCCITT\_ALaw\_8kHzStereo = 42<br>  
 \* SAFTCCITT\_ALaw\_11kHzMono = 43<br>  
 \* SAFTCCITT\_ALaw\_11kHzStereo = 4<br>  
 \* SAFTCCITT\_ALaw\_22kHzMono = 44<br>  
 \* SAFTCCITT\_ALaw\_22kHzStereo = 45<br>  
 \* SAFTCCITT\_ALaw\_44kHzMono = 46<br>  
 \* SAFTCCITT\_ALaw\_44kHzStereo = 47<br>  
 \* <br>  
 \* // u-Law formats<br>  
 \* SAFTCCITT\_uLaw\_8kHzMono = 48<br>  
 \* SAFTCCITT\_uLaw\_8kHzStereo = 49<br>  
 \* SAFTCCITT\_uLaw\_11kHzMono = 50<br>  
 \* SAFTCCITT\_uLaw\_11kHzStereo = 51<br>  
 \* SAFTCCITT\_uLaw\_22kHzMono = 52<br>  
 \* SAFTCCITT\_uLaw\_22kHzStereo = 53<br>  
 \* SAFTCCITT\_uLaw\_44kHzMono = 54<br>  
 \* SAFTCCITT\_uLaw\_44kHzStereo = 55<br>  
 \* SAFTADPCM\_8kHzMono = 56<br>  
 \* SAFTADPCM\_8kHzStereo = 57<br>  
 \* SAFTADPCM\_11kHzMono = 58<br>  
 \* SAFTADPCM\_11kHzStereo = 59<br>  
 \* SAFTADPCM\_22kHzMono = 60<br>  
 \* SAFTADPCM\_22kHzStereo = 61<br>  
 \* SAFTADPCM\_44kHzMono = 62<br>  
 \* SAFTADPCM\_44kHzStereo = 63<br>  
 \* <br>  
 \* // GSM 6.10 formats<br>  
 \* SAFTGSM610\_8kHzMono = 64<br>  
 \* SAFTGSM610\_11kHzMono = 65<br>  
 \* SAFTGSM610\_22kHzMono = 66<br>  
 \* SAFTGSM610\_44kHzMono = 67<br>  
 \* // Other formats<br>  
 \* SAFTNUM\_FORMATS = 68<br>  
 \*  
 \** ***@param formatType*** *\*/* **public void** setFormatType(**int** formatType) {  
 **this**.**formatType** = formatType;  
 }  
  
 **public** TxtConvertVoice(){  
 ComThread.*InitSTA*();  
 **if** (**ax**==**null**){  
 **ax**=**new** ActiveXComponent(**"Sapi.SpVoice"**);  
 **spVoice**=**ax**.getObject();  
 }  
 }  
  
 */\*\*  
 \* 改变语音库  
 \** ***@param voice*** *语音库序号  
 \*/* **public void** changeVoice(**int** voice){  
 **if** (**this**.**voice**!=voice) **this**.**voice**=voice;  
 Dispatch voices = Dispatch.*call*(**spVoice**, **"GetVoices"**).toDispatch();  
 **int** count = Integer.*parseInt*(Dispatch.*call*(voices, **"Count"**).toString());  
 **if** (count>0){  
 Dispatch item = Dispatch.*call*(voices, **"Item"**, **new** Variant(**this**.**voice**)).toDispatch();  
 Dispatch.*put*(**spVoice**,**"Voice"**,item);  
 }  
 }  
  
 */\*\*  
 \* 改变音频输出设备  
 \** ***@param audio*** *\*/* **public void** changeAudioOut(**int** audio){  
 **if**(**this**.**audio**!=audio) **this**.**audio**=audio;  
 Dispatch audioOutputs = Dispatch.*call*(**spMMAudioOut**, **"GetAudioOutputs"**).toDispatch();  
 **int** count = Integer.*parseInt*(Dispatch.*call*(audioOutputs, **"Count"**).toString());  
 **if** (count>0){  
 Dispatch item = Dispatch.*call*(audioOutputs, **"Item"**, **new** Variant(**this**.**audio**)).toDispatch();  
 Dispatch.*put*(**spMMAudioOut**,**"AudioOutput"**,audioOutputs);  
 }  
 }  
  
 */\*\*  
 \* 播放语音  
 \** ***@param text*** *\*/* **public void** speak(String text){  
 **this**.speak(text,0);  
 }  
  
 */\*\*  
 \* 播放语音  
 \** ***@param text*** *\** ***@param type*** *\*/* **public void** speak(String text,**int** type){  
 **switch** (type){  
 **case** 0:  
 *//设置音量，读的速度* Dispatch.*put*(**spVoice**,**"Volume"**,**new** Variant(**this**.**volume**));*//设置音量* Dispatch.*put*(**spVoice**,**"Rate"**,**new** Variant(**this**.**rate**));*//设置频率* **if** (**spAudioFormat**==**null**){  
 **ax**=**new** ActiveXComponent(**"Sapi.SpAudioFormat"**);  
 **spAudioFormat**=**ax**.getObject();  
 **ax**=**new** ActiveXComponent(**"Sapi.SpMMAudioOut"**);  
 **spMMAudioOut**=**ax**.getObject();  
 }  
 Dispatch.*put*(**spAudioFormat**,**"Type"**,**new** Variant(**this**.**formatType**));  
 Dispatch.*putRef*(**spMMAudioOut**,**"Format"**,**spAudioFormat**);  
 Dispatch.*put*(**spVoice**,**"AllowAudioOutputFormatChangesOnNextSet"**,**new** Variant(**false**));  
 Dispatch.*putRef*(**spVoice**,**"AudioOutputStream"**,**spMMAudioOut**);  
*// 开始朗读* Dispatch.*call*(**spVoice**,**"Speak"**,**new** Variant(text));  
 **break**;  
 **case** 1:  
 Dispatch.*call*(**spVoice**,**"Speak"**,**new** Variant(text),**new** Variant(2));  
 **break**;  
 **default**:  
 **break**;  
 }  
 }  
  
 */\*\*  
 \* 停止播放语音  
 \*/* **public void** stop(){  
 Dispatch.*call*(**spVoice**,**"Pause"**);  
 }  
  
 */\*\*  
 \* 获取系统中所有的语音库名称  
 \** ***@return*** *\*/* **public** String[] getVoices(){  
 String[] voices=**null**;  
 Dispatch voices1 = Dispatch.*call*(**spVoice**, **"GetVoices"**).toDispatch();  
 **int** count = Integer.*parseInt*(Dispatch.*call*(voices1, **"Count"**).toString());  
 **if** (count>0){  
 voices=**new** String[count];  
 **for** (**int** i=0;i<count;i++){  
 Dispatch item = Dispatch.*call*(voices1, **"Item"**, **new** Variant(i)).toDispatch();  
 String voice = Dispatch.*call*(item, **"GetDescription"**).toString();  
 voices[i]=voice;  
 }  
 }  
 **return** voices;  
 }  
  
 */\*\*  
 \* 获取所有音频设备输出名称数组  
 \** ***@return*** *\*/* **public** String[] getAudioOutputs(){  
 String[] audios=**null**;  
 Dispatch audioOutputs = Dispatch.*call*(**spMMAudioOut**, **"GetAudioOutputs"**).toDispatch();  
 **int** count = Integer.*parseInt*(Dispatch.*call*(audioOutputs, **"Count"**).toString());  
 **if** (count>0){  
 audios=**new** String[count];  
 **for** (**int** i=0;i<count;i++){  
 Dispatch item = Dispatch.*call*(audioOutputs, **"Item"**, **new** Variant(i)).toDispatch();  
 String audio = Dispatch.*call*(item, **"GetDescription"**).toString();  
 audios[i]=audio;  
 }  
 }  
 **return** audios;  
 }  
  
 */\*\*  
 \* 将文字转换成音频信号，然后输出到wav文件  
 \** ***@param text*** *\** ***@param filePath*** *\*/* **public void** saveToWav(String text,String filePath){  
 System.***out***.println(filePath);  
 *//创建输出文件流对象* **ax**=**new** ActiveXComponent(**"Sapi.SpFileStream"**);  
 **spFileStream**=**ax**.getObject();  
*// 创建音频流格式对象* **if** (**spAudioFormat**==**null**){  
 **ax**=**new** ActiveXComponent(**"Sapi.SpAudioFormat"**);  
 **spAudioFormat**=**ax**.getObject();  
 }  
*// 设置音频流格式类型* Dispatch.*put*(**spAudioFormat**,**"Type"**,**new** Variant(**this**.**formatType**));  
*// 设置问价输出流格式* Dispatch.*putRef*(**spFileStream**,**"Format"**,**spAudioFormat**);  
*// 调用文件输出流的打开方式，创建一个wav文件* Dispatch.*call*(**spFileStream**,**"Open"**,**new** Variant(filePath),**new** Variant(3),**new** Variant(**true**));  
*// 设置声音对象的音频输出流为输出流文件对象* Dispatch.*putRef*(**spVoice**,**"AudioOutputStream"**,**spFileStream**);  
 *//设置音量和读的速度* Dispatch.*put*(**spVoice**,**"Volume"**,**new** Variant(**this**.**volume**));  
 Dispatch.*put*(**spVoice**,**"Rate"**,**new** Variant(**this**.**rate**));  
*// 开始朗读* Dispatch.*call*(**spVoice**,**"Speak"**,**new** Variant(text));  
*// 关闭输出文件流对象，释放资源* Dispatch.*call*(**spFileStream**,**"Close"**);  
 Dispatch.*putRef*(**spVoice**,**"AudioOutputStream"**,**null**);  
 }  
  
 @Test  
 **public void** demo1(){  
 TxtConvertVoice convertVoice=**new** TxtConvertVoice();  
 String text=**"这是我的测试,物理内存至少需要512MB，建议2GB以上,虚拟内存是主机物理内存的两倍，不要设到系统盘,硬盘空闲空间大于4.77GB."**;  
 convertVoice.setFormatType(6);  
 String bir=**"D:\\"**;  
 String name=**"wav3.wav"**;  
 convertVoice.saveToWav(text,bir+ File.***separator***+name);  
 }  
  
*// public static void main(String[] args){  
// TxtConvertVoice convertVoice=new TxtConvertVoice();  
// String text="这是我的测试,物理内存至少需要512MB，建议2GB以上,虚拟内存是主机物理内存的两倍，不要设到系统盘,硬盘空闲空间大于4.77GB.";  
// convertVoice.setFormatType(6);  
//// String bir="C:\\Users\\Admteinistrator\\Desktop";  
// String bir="D:\\";  
// String name="wav1.wav";  
// convertVoice.saveToWav(text,bir+ File.separator+name);  
// }*}

**package** com.lshs.mytest;  
  
**import** com.baidu.aip.speech.AipSpeech;  
**import** org.json.JSONObject;  
**import** org.jsoup.Connection;  
**import** org.junit.Test;  
**import** org.springframework.core.io.Resource;  
  
**import** java.io.\*;  
**import** java.net.HttpURLConnection;  
**import** java.net.MalformedURLException;  
**import** java.net.URL;  
**import** java.util.List;  
**import** java.util.Map;  
  
*/\*\*  
 \** ***@Description:*** *语音转文字应用  
 \** ***@author:*** *LuShao  
 \** ***@create:*** *2018-08-21 11:45  
 \*\*/***public class** VioceToTxt {  
  
 *//语音上传文件服务器地址* **private static final** String ***serverUrl***=**"http://vop.baidu.com/server\_api"**;  
*// // 语种选择，中文=zh、粤语=ct、英文=en，不区分大小写，默认中文* **private static** String *language* = **"en"**;  
  
 **private static** String *user*=**"lsh"**;  
  
 **private static** String *APP\_ID*=**"11704453"**;  
 **private static** String *API\_KEY*=**"vcsnqV2CaOkQrsVO8TgbGTR5"**;  
 **private static** String *SECRET\_KEY*=**"ZuxjFR7TZtWQlkwUBCiZnjYCjBUReWom"**;  
  
 @Test  
 **public void** demo1() **throws** IOException {  
 String auth = getAuth();  
 System.***out***.println(auth);  
 }  
  
 */\*\*  
 \* 获取token  
 \** ***@return*** *\** ***@throws*** *IOException  
 \*/* **public** String getAuth() **throws** IOException {  
 String apiKey=**"vcsnqV2CaOkQrsVO8TgbGTR5"**;  
 String secretKey=**"ZuxjFR7TZtWQlkwUBCiZnjYCjBUReWom"**;  
 *//获取token地址* String authHost=**"https://aip.baidubce.com/oauth/2.0/token?"**;  
*// 拼接地址* String accessUrl=authHost+**"grant\_type=client\_credentials"**+**"&client\_id="** + apiKey+**"&client\_secret="** + secretKey;  
*// 打开链接* URL url = **new** URL(accessUrl);  
 HttpURLConnection connection = (HttpURLConnection) url.openConnection();  
*// GET必须大写* connection.setRequestMethod(**"GET"**);  
 connection.connect();  
*// 获得所有的响应头字段* Map<String, List<String>> headerFields = connection.getHeaderFields();  
 BufferedReader br = **new** BufferedReader(**new** InputStreamReader(connection.getInputStream()));  
 StringBuilder result = **new** StringBuilder();  
 String line;  
 **while** ((line=br.readLine())!=**null**){  
 result.append(line);  
 }  
 JSONObject js=**new** JSONObject(result.toString());  
 String access\_token = js.getString(**"access\_token"**);  
  
 **return** access\_token;  
 }  
  
 */\*\*  
 \* raw方式上传语音数据  
 \** ***@param rawData*** *\** ***@param token*** *\** ***@return*** *\** ***@throws*** *IOException  
 \*/* **public** String getResult(**byte**[] rawData,String token) **throws** IOException {  
 String url=***serverUrl***+**"?lan="**+*language*+**"&cuid="**+*user*+**"&token="**+token;  
 URL url1=**new** URL(url);  
 HttpURLConnection connection= (HttpURLConnection) url1.openConnection();  
 connection.setRequestMethod(**"POST"**);  
 *// 设置请求头* connection.setRequestProperty(**"Content-Type"**, **"audio/pcm; rate=16000"**);  
 *// 设置是否从conn读入，默认情况下是true;* connection.setDoInput(**true**);  
 *// 设置是否向conn输出，因为这个是post请求，参数要放在  
 // http正文内，因此需要设为true, 默认情况下是false;* connection.setDoOutput(**true**);  
 *// 提交语音数据* DataOutputStream wr = **new** DataOutputStream(connection.getOutputStream());  
 wr.write(rawData);  
 wr.flush();  
 wr.close();  
  
*// 获取服务器访问数据* **if** (connection.getResponseCode()==200){  
 InputStream in = connection.getInputStream();  
 BufferedReader br=**new** BufferedReader(**new** InputStreamReader(in));  
 String line;  
 StringBuffer sb=**new** StringBuffer();  
 **while** ((line=br.readLine())!=**null**){  
 sb.append(line).append(**"\r"**);  
 }  
 br.close();  
  
 */\*\*  
 \* 解析服务器返回的数据  
 \* 返回的数据格式：json格式  
 \* 字段"err\_no" - 错误码  
 \* 字段"err\_msg" - 错误码描述  
 \* 字段"sn" - 语音数据唯一标识，系统内部产生。如果反馈及debug请提供sn。  
 \* 字段"result" - 识别结果数组，提供1-5 个候选结果， 优先使用第一个结果。utf-8 编码。  
 \*/* JSONObject jo=**new** JSONObject(sb.toString());  
 System.***out***.println(jo);  
 String result = jo.getString(**"result"**);  
 **return** result;  
 }  
 **return null**;  
 }  
  
 */\*\*  
 \* 获取音频数据  
 \** ***@param source*** *\** ***@return*** *\*/* **public byte**[] getRawData(File file) **throws** IOException {  
 FileInputStream fis = **new** FileInputStream(file);  
 **long** length = file.length();  
 **byte**[] b=**new byte**[(**int**) length];  
  
 **int** offset=0;  
 **int** numread=0;  
 **while** (offset<b.**length**&&(numread=fis.read(b,offset,b.**length**-offset))>=0){  
 offset+=numread;  
 }  
  
 fis.close();  
 **return** b;  
 }  
  
 @Test  
 **public void** demo2() **throws** IOException {  
 File file=**new** File(**"F:\\HTML5TEST\\myWeb\\m2.mid"**);  
 **byte**[] rawData = getRawData(file);  
 System.***out***.println(rawData);  
 }  
  
 @Test  
 **public void** demo3() **throws** IOException {  
 String auth = getAuth();  
 String url4=**"D:\\wav1.pcm"**;  
 File file=**new** File(url4);  
 **byte**[] rawData = getRawData(file);  
 String result = getResult(rawData, auth);  
 System.***out***.println(result);  
 }  
  
 @Test  
 **public void** demo4(){  
 *// 初始化一个AipSpeech* AipSpeech client = **new** AipSpeech(*APP\_ID*, *API\_KEY*, *SECRET\_KEY*);  
 *// 可选：设置网络连接参数  
// client.setConnectionTimeoutInMillis(2000);  
// client.setSocketTimeoutInMillis(60000);  
  
 // 可选：设置代理服务器地址, http和socket二选一，或者均不设置  
// client.setHttpProxy("proxy\_host", proxy\_port); // 设置http代理  
// client.setSocketProxy("proxy\_host", proxy\_port); // 设置socket代理  
  
 // 调用接口File file=new File("F:\\HTML5TEST\\myWeb\\m2.mid");* String url=**"F:\\HTML5TEST\\myWeb\\m1.mp3"**;  
 String url2=**"F:\\HTML5TEST\\myWeb\\m2.mid"**;  
 String url3=**"D:\\wav1.wav"**;  
 String url4=**"D:\\wav1.pcm"**;  
 JSONObject res = client.asr(url3, **"pcm"**, 16000, **null**);  
 System.***out***.println(res.toString(2));  
 }  
  
}