CSCI1530 Computer Principles and Java Programming 2014-2015 Second Term
Department of Computer Science and Engineering
The Chinese University of Hong Kong

Due date: 8 April 2015 (Wed) Assignment 5 Full mark: 100
Expected normal time spent: 5 hours

Music String Player

Aim: 1. build a practical music string player using Java;
2. practise creating and using classes and objects;
3. practise manipulating strings and characters.

Task: Create a Java program for playing songs, playing MP3 recordings and talking to us!!

Let's sing: Do Re Me Fa So La Ti ...
Copy and paste the following "strings" one-by-one into the program Input Dialog to have fun!

```
"Kum Bah Ya"
D M S S S / L L S / D M S S S / F M R / D M S S S / L L S / S M D / R R D

"Ode to Joy" / <<快樂頌>>
M M F S S F M R / D D R M M R R / M M F S S F M R / D D R M R D D /
R R M D / R M F M D / R M F M R D R S / M M F S S F M R / D D R M R D D

say:Hello, CSCI1530 Assignment due on 8 Apr 2015

http://www.cse.cuhk.edu.hk/csci1530/asgdemo/Cantonese0-9.mp3

http://freepd.com/Classical/Allemande.mp3
```

In this assignment, we play with 7 musical notes (**音符**). The seven notes are pre-recorded and given in seven MP3 sound files, **D.mp3**, **R.mp3**, **M.mp3**, **F.mp3**, **S.mp3**, **L.mp3** and **T.mp3**.

The program presents an Input Dialog to the user to ask for a "music string" text input. For each character in the string, the program tries to match and to play the corresponding pre-recorded note sound file if it is one of the characters in **DRMFSLT**. All other unmatched characters should be ignored silently. After playing a song, the program repeats the Input Dialog until the user clicks Cancel or Closes the dialog box for termination, i.e. getting a ***null*** String input.

In addition to playing musical notes, we want to extend its capability. The program shall also play a MP3 recording given by an URL (with an **MP3Player** object) or talk to us (with a **GoogleVoice** object.) If the user input **startsWith** the seven characters **"http://"**, it will try playing the URL. If the user input **startsWith** the four characters **"say:"**, it will speak out!

Procedure (Step-by-Step):

1. Plug a pair of speakers or headset to the computer you use. Make sure it sounds! Try the Sample Program with some example music strings (suggested in the box above.)

2. A NetBeans starter kit has been prepared for you. Unzip and use it to begin with.

3. You should work on the classes Main and MusicStringPlayer. You should NOT modify the given classes GoogleVoice and MP3Player. They are provided as is.

4. The class Main should contain only a simple **`main()`** method which shows Input Dialog and creates different objects to do the magic.

5. The class MusicStringPlayer should contain non-static field(s), method(s) and constructor(s). All fields and methods are instance (i.e. non-static,) apart from the given **`errorReporting()`** method.

6. A MusicStringPlayer object stores the song String and analyzes the characters in the String. It also creates MP3Player objects to play the corresponding sound files.

7. The program needs not do any data validation and exception handling. We also assume the pathnames and sound files are valid, as provisioned in the project.

Demo and Submission:

1. Unzip our sample program package in a folder, with all the sound note MP3 files and the JAR file together. Try our sample program **`MusicStringPlayer.jar`** by double-clicking it.

2. Online demo at: http://www.cse.cuhk.edu.hk/csci1530/asgdemo/MusicStringPlayerDemo/ However, it cannot make use of Google online services due to usual system security settings.

3. ZIP and Submit YOUR whole NetBeans project folder **`MusicStringPlayerKit\`** in **`MusicStringPlayer.zip`** via eLearn.

Marking Scheme and Notes:

1. The submitted program should be free of any typing mistakes, compilation errors and warnings.

2. Comment/remark, indentation, style are under assessment in every programming assignments unless specified otherwise.

3. Remember to do your submission before 18:00 p.m. of the due date. No late submission would be accepted.

4. If you submit multiple times, **ONLY** the content and time-stamp of the **latest** one would be counted. You may delete (i.e. take back) your assignment from CUForum and re-submit. We ONLY take into account the last submission.

5. **Plagiarism is strictly monitored and punished** if proven. Lending your work to others for reference is considered as supplying a source for copying which is subject to the **SAME** penalty. Being a mature participant in this course, you are supposed to be able to differentiate between teaching others, discussion and partial/full copying. Please refer to the policy of our University: http://www.cuhk.edu.hk/policy/academichonesty.

Further Information:

1. To simplify this assignment, beat and timings are not handled.

2. The delay (waiting) time is not exact in Java. That's why your may notice audible differences by playing the same song a few times.

3. See http://en.wikipedia.org/wiki/C_(musical_note) for a description of the musical note *Middle C*. In this assignment, we use C Scale. Do you notice the "powers of two" in the musical scale frequencies? Harmonics are everywhere in music.

4. You may create a musical jukebox. Well, the use and application of your work is unlimited!

5. We make use of the **`jlayer`** library from JavaZoom for playing MP3. (http://www.javazoom.net/javalayer/javalayer.html)