CSCI 1130 Introduction to Computing Using Java 2021-2022 First Term
Department of Computer Science and Engineering
The Chinese University of Hong Kong

Due date: 4 November 2021 (Thu)

Assignment 4

Full mark: 100
Expected normal time spent: 8 hours

Music App

Aim: 1. practise file I/O and exception handling;
2. practise String processing.

Task: Create a Java application that transcripts a simple song score sheet.

The program asks user for filename of a text file that contains a simple song score sheet in this format:

Sample song score sheet 1, **song1.txt**:

```
#simple song with assumed time signature = 4
d1m1s2o2s1m1d2o2
d1r1m1f1s3o1s1f1m1r1d4
```

Lines beginning with a hash '**#**' is a comment line. A song contains one or more lines of musical notes, **d, r, m, f, s, l, t** and also **o** meaning "Off" or rest. Each single character note is followed by a single digit integer in [1 – 4], denoting the duration of the note in number of beats. After transcription, the program outputs the melody on the screen as well as to another text file:

Melody output 1, on screen AND in **melody_song1.txt**:

```
#simple song with assumed time signature = 4
Do Me So- | Off- So Me | Do- Off- |
Do Re Me Fa | So-- Off | So Fa Me Re | Do--- |
```

The default time signature is 4 beats per section. Sections are delimited by bars '**|**'. Notes are converted into full-names with 2 characters, Do, Re, Me, Fa, So, La, Ti, and Off. Notes lasting for one beat are displayed directly. A note lasting two or more beats are elongated with a display of dashes '**-**', e.g., **So--** for **s3**. Notes and bars are delimited by a space.

Time signature can be changed with a newline starting with an asterisk '*****', followed by a single digit integer *n* in [1 – 6]. Transcript and output such a line as "**#Time Signature = *n***", e.g.,

Sample song score sheet 2, **song2.txt**:

```
#song with different time signatures
*3
d1r1m1o3m1r1d1
d1r1m1f1s1o1s1f1m1r1d2
*2
d2r1m1f1s1l1t1
*4
d4r4o2m2
```

Melody output 2, on screen AND in **melody_song2.txt**:

```
#song with different time signatures
#Time Signature = 3
Do Re Me | Off-- | Me Re Do |
Do Re Me | Fa So Off | So Fa Me | Re Do- |
#Time Signature = 2
Do- | Re Me | Fa So | La Ti |
#Time Signature = 4
Do--- | Re--- | Off- Me- |
```

Then, the program terminates.

To simplify this task, we **assume**:

- Input song text files always contain valid data.
- The time signature designation and the note durations are always valid.
- Section bars shall NOT cut a multiple beat notes, e.g., `Do-- | - Re-` will NOT happen.
- A section shall not continue between two lines.
- We accept optionally an extra space at the end of the output lines.

Procedure:

1. Work under a NetBeans Java Application project **`MusicApp`**. Name the package as **`musicapp`**.

2. Complete the program according to the above descriptions as well as the samples given.

   Note that file operations may fail. We give the user THREE chances to input a proper input filename. If all fails, the program terminates with **`System.exit(1)`**.

```
Song filename: a.exe
Something wrong!
Song filename: C:/xyz
Something wrong!
Song filename: a
Something wrong!
```

3. **Screen output** of a complete sample run with text in blue indicates user keyboard input:

```
Song filename: a
Something wrong!
Song filename: a.exe
Something wrong!
Song filename: song2.txt
Reading song file song2.txt
#song with different time signatures
#Time Signature = 3
Do Re Me | Off-- | Me Re Do |
Do Re Me | Fa So Off | So Fa Me | Re Do- |
#Time Signature = 2
Do- | Re Me | Fa So | La Ti |
#Time Signature = 4
Do--- | Re--- | Off- Me- |
```

The program also outputs those text in yellow highlight to **another output text file** named **`melody_song2.txt`**, i.e., prefixed ***melody_<inputFilename>***.

4. Points to note:

   a) The program shall handle all expected *Exception* and/ or error condition gracefully.

   b) If the input file cannot be opened, print "**`Something wrong!`**" on the screen. Let user try THREE times before making a success OR ending with termination.

   c) If the output file cannot be opened, printed, closed, etc. OR any Exception happened during processing, print "**`Cannot output melody file!`**" on the screen and terminate.

   d) Close the Scanner and PrintStream objects properly.

Submission:

1. **ZIP** the whole NetBeans project folder **`MusicApp`** in **`MusicApp.zip`**.

2. Submit the SINGLE file **`MusicApp.zip`** via our Blackboard Homework Collection Box **`https://blackboard.cuhk.edu.hk`**

3. Reminder: make sure you have typed your name and SID in your source files!

Notes:

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

2. Comment/remark, indentation, style is 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. **ONLY** the content and time-stamp of the **latest** submission would be taken into account.

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.