CS 220 Final Project

In this final project, you will use various audio/music utilities that we have written during the semester to create a program that produces a version of the Wartburg loyalty song.

Format of Song Files

I have provided a file that contains information about the loyalty song in a format that is a simplified version of "abc notation." This is is a text-based notation that is used by some song archives that you can find on the Internet.

Looking at the file `loyalty.sng` you will see that it consists of some general information at the top followed by 4 "parts" separated by blank lines. The parts provide sequences of notes that make up the simultaneous parts of a song. You can think of these as the parts played by different instruments or sung by different voices. In musical termns, the loyalty song is written in 4-part harmony.

Each part is basically just a sequence of notes (pitch and duration). Those who are musically inclined can probably figure out the notation. However, you really don't need to decipher the notation unless you want to construct your own song files, because I have supplied a class to take care of reading the files for you.

Using the SongReader class

The provided SongReader class will take care of reading the song file for you. Here's a little interactive demo:

```
>>> from songreader import SongReader
>>> data = SongReader("loyalty.sng")
>>> data.info
{'TITLE': ' Wartburg Loyalty', 'TEMPO': ' 120'}
>>> len(data.parts)
4
>>> data.parts[0][:5]
[('Ab4', 1.5), ('Ab4', 0.5), ('G4', 1.0), ('Ab4', 1.0), ('F4', 1.0)]
```

As you can see, when you create a `SongReader` you give it the name of the file to read. It processes the file and provides the information from the file in 2 public instance variables, `info` and `parts`. The former is a dictionary with the info from the top of the file, and the latter is a list of the parts that were found in the file.

Notice that the reader found 4 parts in this file. The individual parts can be accessed by subscripting `parts`. The last interaction shows the first 5 notes in the first part. The notes are provided as a sequence of (pitch, beats) pairs. The duration of a note is in terms of musical beats, not seconds. The actual time duration of the note is determined by the tempo of the song (given in beats per minute). To get the duration in seconds, we need a unit conversion:

```
duration = beats/tempo * 60
```

Following this approach, we can easily adjust the tempo of the song when producing ("rendering") the audio file.

Turning Songs into Audio

The main work of your program will be done by another class called a `SongPlayer`. An outline of this class is provided in `songplayer.py`. You should read the specifications in that file carefully to see what it does. As an example, here's a "quick and dirty" way to listen to 5 seconds of audio from "rendering" the first two parts of the loyalty using a plain vanilla string synthesizer:

```
from songplayer import SongPlayer
from stringsynth import StringSynth

player = SongPlayer("loyalty.sng")
synth = StringSynth()
player.render_part(0, synth)
player.render_part(1, synth)
player.play_audio(5)
```

Of course we can get much more interesting results by using different synthesizers (or at least different synthesizer settings) for various parts. You can also render a single part multiple times using different synthesizers to produce more "textured" sounds.

Assignment

Your assignment is to implement the `SongPlayer` class and also write a program that generates a `loyalty4.wav` file with a nice 4-part rendering of the Wartburg loyalty. You should turn in 3 files:

```
songplayer.py -- The completed SongPlayer class
loyalty.py    -- Your program that generates ``loyalty4.wav``
loyalty4.wav  -- Your audio file
```

Bonus points will be awarded for the most interesting wav files. I'm especially interested in ones that sound particularly nice. Note you should make sure to submit exactly the version of the `loyalty.py` that produced the file. If running your program with MY class produces a result that sounds different from your file, that means your SongPlayer does not match the specification.

Have fun!