steCSE 213 – Homework 4 Rubric

Student Name: Marisa Loraas

Grader Name: **Steven Aque**

Problem	Score	Total
Style Guidelines (7 points)	5	**5**
Submission is named `cse213_<firstname>_<lastname>_hw4.tar.gz`	1	1
Packages are named: `oop.<firstinitial><lastname>.hw4.<number>` where <number> is the problem number	1	1
Code follows Google's style guide reasonably well, and uses four-space indentation	3	3
Pitch.java	6	**6**
enum class differentiates 12 different notes: C, CSHARP, ...,B	2	2
getOffset() returns -9, -8, ..., 2 for C, CSHARP, ..., ; this can be done with a private attribute, or just **this.ordinal()** - 9	2	2
toString() returns a string with the format "C#"	2	2
Beat.java	10	**10**
enum class differentiates 8 different durations: WHOLE, THREEQUARTER, ..., SIXTEENTH	2	2
getBeats() returns 4, 3, 2, 3/2, 1, 3/4, 1/2, 1/4 for WHOLE, THREEQUARTER, ..., SIXTEENTH	2	2
getDuration(tempo) returns (60.0 x beets) / tempo	3	3
toString() returns a string in the form "(1/4)"; the fraction of a WHOLE note, wrapped in parentheses	3	3
Note.java	50	**65**
Default constructor sets **pitch** to C, **octave** to 4, and **length** to QUARTER	3	3
new Note(pitch, octave, beat) checks: 1. if pitch is G#, A, A#, or B then -1 ≤ octave ≤ 8 2. Otherwise, -1 ≤ octave ≤ 9 If the octave is out of range, sets the pitch to C and the octave to 4 • **Does not set defaults on invalid input (-2 points)** **Note: instead of checking octave == 9, you should be checking octave > 8, but I won't be taking off any points**	3	5
new Note(spn, beat) parses the string, spn to get the pitch and octave (see setSPN()). If the string is invalid, sets the pitch to C and the octave to 4. • **Does not set defaults on invalid input (-2 points)**	3	5

new Note(midi, beat) converts the MIDI number to a pitch and octave, (see setMIDI()). If the MIDI number is not between 0 and 127 (inclusive), sets the pitch to C and the octave to 4 • **Does not set defaults on invalid input (-2 points)**	3	5
New Note(frequency, beat) converts the frequency to the given note and octave (see setFrequency()). If the frequency is out of range, sets the pitch to C and the octave to 4. • **Does not set defaults on invalid input (-2 points)**	3	5
getSPN() returns a string, containing the pitch followed by the octave (i.e. "C4" or "D#-1")	3	3
setSPN() parses a string to set both the pitch and octave: 1. If the string is invalid, do nothing 2. If the second character is '#', the pitch is given by spn.substring(0, 2) and the octave is given by spn.substring(2) 3. Otherwise, the pitch and octave are given by spn.substring(0, 1) and spn.substring(1) • **Octave setting method, while creative, does not account for "-1" as a valid octave. Minus sign will be removed by replacement. (-1 point)**	5	6
getMIDI() converts the pitch and octave to the corresponding MIDI number: 69 + pitch.getOffset() + 12 x (octave - 4)	6	6
setMIDI() sets the pitch and octave using a MIDI number: 1. If the MIDI number is < 0 or > 127, do nothing 2. Otherwise, the pitch has the value: midi % 12, (this is the index in Pitch.values()) 3. And the octave is given by: midi / 12 - 1	6	6
getFrequency() computes the midi number, and converts it to the corresponding frequency: $f = 440 \times 2^{(midi - 69)/12}$	5	5
setFrequency() converts the frequency to a midi number: $midi = 12 \times \log_2(f/440) + 69$ and uses it to set the pitch and octave, (see setMIDI())	5	5
setOctave() checks that: 1. If the pitch is G#, A, A#, or B, then the new octave is between -1 and 8 2. Otherwise, $-1 \leq octave \leq 9$ If the input is invalid, this method does not set the octave. • **No error checking (-3 points)**	1	4
setPitch() checks if the octave is 9; if it is, and the input is G#, A, A#, or B, then it does not set the pitch • **No error checking (-3 points)**	1	4
toString() returns the SPN followed by a space and the length, (i.e. "C4 (1/4)" or "D#-1 (3/16)")	3	3
PlaySong.java	9	**9**
playSong() takes an ArrayList of Notes and a tempo. For each note, it gets:	4	4

1. The frequency (note.getFrequecy()) 2. And the duration, (note.getLength().getDuration(tempo)) And passes these as arguments to Tone.playTone()		
main() opens song.txt and reads each line. The second word on each line is converted to a Beat value; the first is an SPN string.	2	2
Each line of the input is converted to a Note, and added to an ArrayList	2	2
The list is passed to playSong(), with a tempo of 120. The result should sound like *Scarborough Fair by Simon and Farfunkel!*	1	1
JUnit Tests	5	**5**
Student creates a JUnit Test Suite class to run all other unit tests	0.5	0.5
Student creates a testing class to test their Pitch class that tests: 1. Pitch.A.getOffset() returns 0 (0.5 pts) 2. Pitch.C.getOffset() returns -9 (0.5 pts)	1	1
Student creates a testing class to test their Beat class that tests: 1. getDuration() w/ WHOLE and a tempo of 120 returns 2 (0.5 pts) 2. getDuration() w/ SIXTEENTH w/ a tempo of 120 returns 0.125 (0.5 pts)	1	1
Student creates a testing class to test their Note class that tests: 1. (Constructor 1) A new object is created with C 4 quarter note. 2. (Constructor 2) The maximum octave of the new Note object returned is 8 if given Pitch is A # . 3. (Constructor 3) The pitch and octave of a valid note is set accordingly. 4. (Constructor 4) The note and octave of a valid midi range is set accordingly. 5. (Constructor 5) The new object sets the correct pitch and octave from nearest MIDI number. (0.5 pts) for each constructor test	2.5	2.5
Total Score	85	**100**
Comments:		