Lab 4: Master of Ceremonies

Programming Fundamentals 2

30th March 2021

Goals

✯ Analyzing the needs of a client.

✯ Decomposing a specification into small tasks, and make different versions.

✯ Implementing an app with blurry specifications.

- This laboratory must be done **alone**[1].
- **Deadline: Saturday 10th** April 08:00AM

Deliverables

1. The code on your Github repository generated by clicking here: `https://classroom.github.com/a/MsS1_hSi`
2. The tags representing the various versions of the software (see explanations below).
3. The UML class diagram of the code.
4. A presentation video of 3' minutes on FLIPGRID: `https://flipgrid.com/a23756d1`
    - Use your @student.uni.lu email to connect.
    - Share your screen and comment your work on this lab.

[1] This is to prepare you for the exam.

1 Preliminary

It is very important to read the whole exercise before starting. The explanations of your friend are not necessarily given in the right "implementation order" (*e.g.*, the first implementation step might come last). This exercise is not very long in terms of code. Try to make the design as neat as possible helped by the Java concepts we learnt this semester. Be careful at the code you write (small functions, indentation, names, ...), we will be more strict on these aspects. Here some other important points:

- You must set up a *Maven* project, so `mvn compile` must work.
- Make sure your code pass the clean coding checklist. Passing this checklist will be very important to succeed the exam.
- Windows user: The sound is not working natively in WSL, so you might want to use an IDE (IntelliJ or Netbeans for instance). Open your project from the IDE as a Maven project. Alternatives are: (i) install Ubuntu on your computer (that's actually quite easy), (ii) run Ubuntu in a VirtualMachine, (iii) Compile as a `.jar` (using `mvn package`) and execute on Windows.

2 Master of Ceremonies

You friend, a well-known master of ceremonies, wants to organise a huge party post-covid. To make this party even more interesting, he wants all the tracks to be developed with a cutting-edge computer-assisted composition (CAC) software. However, existing softwares are too expensive, so he's asking you to develop a small CAC software, just for his needs. In the next paragraph, your friend describes the expected functionalities of his app.

I already developed a proof-of-concept using the amazing language Java, that you can find in the Github repository. I can already play some notes using the class `Synth.java` and I've shown how to use this class in the function `demo` in `MC.java`. My main goal is to have a system in which I can compose a *musical score* and play it. In particular, you must know that a musical note is basically a pitch, *e.g.*, how low or high is the sound, basically the frequency. For the piano, we have a sequence of (usually) 88 white and black keys, each representing a note with a specific frequency. The notes are grouped in *octaves*, an octave being 12 notes, that we write $C, C^\sharp, D, D^\sharp, E, F, F^\sharp, G, G^\sharp, A, A^\sharp, B$[2]. A key on a piano is uniquely identified by its note (*e.g.*, C or $F^\sharp$) and its octave (*e.g.*, 4 or 5). We write $C4$ the note C in the fourth octave. In Java, I am using the format MIDI, where a note is represented by a number, *e.g.*, the middle note of a piano is $C4 = 60$. Sometimes, it sounds much better if we group notes together (played at the same time) to form a *chord*. An *arpeggio* is a special way to play a chord such that each note in the chord is played individually. We must talk about the *note value*, which is the relative duration of a note. A *whole note* has a relative value of 1, a *half note* has a relative value of $1/2$, which means you can play two half notes during the time of a single half note. Similarly a *quarter note* has a relative value of $1/4$, so you can play four quarter notes during a whole note. Although I'd already be happy with these three note values, you can find them all on Wikipedia[3]. Crucially, note value implies that a note is not specifically bound to a duration in (milli)seconds. If we decide a whole note is 1 second, then a quarter note will be 250ms. This mapping between notes and time is called the tempo or BPM (beats per minutes). For instance, a BPM of 100 on the half note means that we can play 100 half note during one minute, thus a half note lasts for 0.6 seconds. From there, we can compute the duration of other notes as well. Oh, I forgot to say, but we can also have *rest* instead of notes, which are moment of silent. Therefore, a rest has no pitch, but it has a "note value" as well. I'd be happy to have a custom BPM, but if it is fixed by you, I think I can survive!

Your friend wants to test the newest functionalities every day, therefore you must ensure that you are adding one thing at a time, and that at any time, you can compile and present your software! To achieve that, you will, as usual, **decompose your software** into small tasks, easily achievable.
For each new functionality, you will create a version tag on Git:

```
// do the git commit and git push as usual, then to mark a version:
git tag -a v1.1 -m "Support for half note"
git push origin v1.1
```

[2]This is the English notation, for French, Italian,... it would be do, re, mi, ...
[3]`https://en.wikipedia.org/wiki/Note_value`

Obviously, the explanations given by your friend are blurry, and you'll need to make up for it. Questions can be answered on Discord, but keep in mind that in real life you cannot usually meet or ask for information to the client very frequently.

Hint: Chapter 5 on subtype polymorphism can be useful to represent collection of heterogeneous elements (e.g., notes, chords, ...).

3 Competitive track

Those interested in the competitive track must register here (you can join anytime): `https://docs.google.com/spreadsheets/d/1KMZx58SoE08g-l4usphtaLFnPhKzBDhTpa9PgixOok8/edit?usp=sharing`.

See Laboratory 1 for the submission instructions. This time, the exercises are:

- *11742 - Social Constraints* (iterative algorithm): `https://onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=24&page=show_problem&problem=2842`
- *624 - CD* (recursive backtracking): `https://onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=24&page=show_problem&problem=565`
- *10341 - Solve It* (binary search): `https://onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=24&page=show_problem&problem=1282`
- *10226 - Hardwood Species* (`TreeMap` or `TreeSet`): `https://onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=24&page=show_problem&problem=1167`
- *11629 - Ballot evaluation* (`TreeMap` or `TreeSet`): `https://onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=24&page=show_problem&problem=2676`

Good luck!