

M7017E Lab 1

Audio recorder



Team

- Flore Diallo
- Hervé Loeffel
- Clément Notin

Table of Contents

<u>Team</u>	1
<u>1.Problem Specification</u>	3
<u>2.Usage and user's guide</u>	3
<u>3.Systems Description</u>	5
Technical stack.....	5
Packages and classes.....	5
Code design guideline.....	6
Methods description.....	6
Data types and structures.....	6
Version control.....	7
<u>4.Algorithm Description</u>	7
<u>5.Problems and Reflections</u>	7
<u>6.Contribution</u>	7
Clément Notin.....	8
Hervé Loeffel.....	8
Code of honour.....	8

1. Problem Specification

Thanks to *Gstreamer*, a library designed to handle streams and more specifically media, we can create a multimedia system which allows us to record and play some media files such as video or audio. It is composed of five sets of plugins (Core, Base, Good, Bad and Ugly) which contain all the elements we need. We are allowed to use the Base, Core and Good plugins. The computers, in the lab room, on which we are working on allow us to work with all the *Gstreamer* plugins sets except the Ugly one.

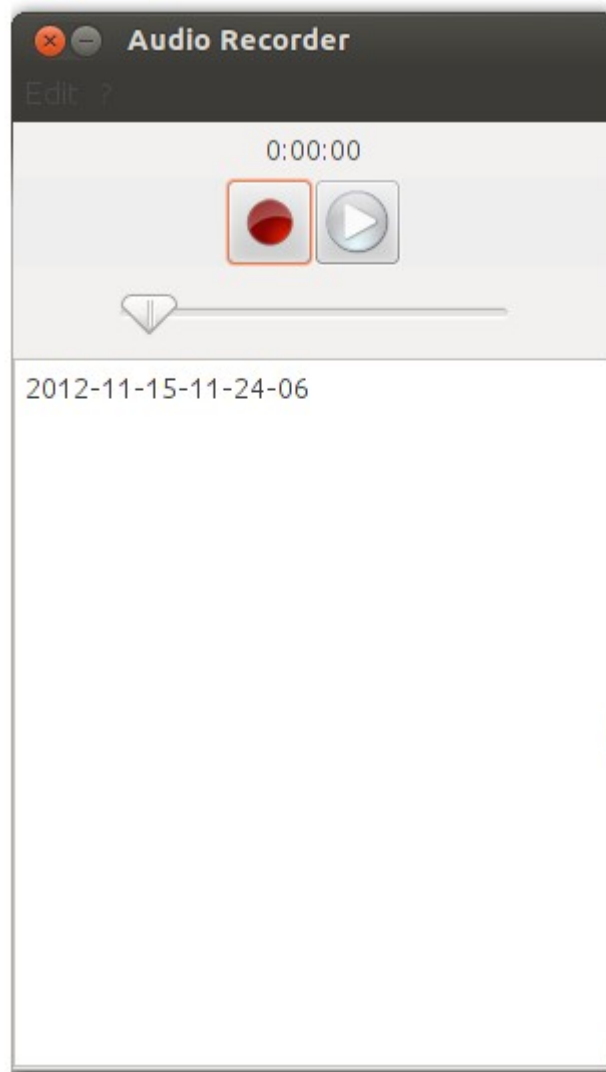
We decided to focus our project on the audio part and created an Audio Recorder which allows the user to record an audio clip and to play it afterwards. To develop our application, we supposed that the user is quite familiar this kind of application and that he knows the symbolic used. Therefore we decided to develop a really simple user interface, with only a few buttons and a few menus, but this interface is really intuitive for any user with basic knowledge in computer. The application also provides some basic functionalities that can be found in every media player, such as seeking into the file while playing, the ability to pause and see a slider and a time indication to know where you currently are in the file. Concerning the media files, we decided to use the OGG format with the Vorbis encoder. The use of MP3 files is not allowed since we would have to use the Ugly set of plugins from *Gstreamer* (for licensing reasons). That way, we only use the Good and Base plugins.

Once these basics tasks resolved, we have added some useful functionalities that an advanced user would be glad to have. First, the user has an access to all the previous recordings (or stored in the same folder) through a list. He can change the selected file while playing another, the new file will be read immediately.

We also decided to allow the user to set his own preferences for the use of the application. It is possible for him to change the folder where the files are stored (or read from) and he can also change the quality and the name of the recording.

The recorder so obtained can be used and understood thanks to the following user guide.


2. Usage and user's guide



The audio recorder can be launched from its executable JAR file on any machine with *GStreamer* installed. The use of the software is quite simple. There are two main buttons:


- The *recording button*:



You just have to press this button to start the recording. When this button has the following icon  it means that it is still recording. To stop the recording, you just have to click again on this button. Then a dialog box will appear to edit the name and to save this file. If you prefer to discard the recording you just have to press the "Cancel" button of the box.

- The *button to play a file*:



This button plays the selected file in the list. To select a file you just have to click on its name in the list, thus the selected file is highlighted. If no file is selected the button will do nothing. You can press on the same button (with the icon ) to pause, and then you can continue the play by clicking again on this

button. You can also play a file simply by double clicking on its name in the list. You can also use the slider under the buttons to go forward or backward in the file.

You also have a *menu bar* with some options (the way it looks in the screenshot is not normal, due to a bug in Gnome theme in the latest Ubuntu). By clicking on Edit -> Preferences, a box appear to configure the folder where the files will be saved to and played from, and the quality of the recording. The button "?" of the menu bar shows a standard about box with some information on the Audio Recorder.

The recorded files are saved as ".ogg" files and our software will only play .ogg files. The other kinds of audio files will not be recognized by the Audio Recorder and will not appear in the list of files. Even though it would not be very complicated to implement other formats thanks to GStreamer plugins' versatility, the user is not supposed to play (even if possible) files other than the recordings, thus restricting to this format seems reasonable.

3. Systems Description

Technical stack

Our project works with Java >1.6.

The UI is created with Swing and tries to use the system's native look'n'feel (the screenshots in this document were taken using Ubuntu Unity graphical manager).

The build and the dependences of the project are managed by Maven which is an industry standard for Java development. We advise to use the corresponding IDE plugin (m2eclipse for Eclipse) or the command line tool "mvn". One can generate the executable JAR for the program with all necessary resources with "\$ mvn clean compile assembly:single" at the root of the project and find the result in "target" folder.

We also used [Lombok](#) to avoid generating getters, setters, default constructors etc. This does not add any runtime overhead but it must be installed in the IDE to remove errors about missing methods. You can see it for example when we use the annotations @Getter, @Setter, @AllArgsConstructor, ...

The dependencies (automatically resolved and downloaded by Maven) are:

- gstreamer-java: 1.5
- junit for the unit tests: 4.11-beta-1
- lombok: 0.11.6

Packages and classes

There are two packages:

- `se.ltu.M7017E.lab1`
 - `App` is the most important class. It exposes all the necessary API methods for the UI (User Interface), like `startPlayer(filename)` or `stopRecorder()`.
 - `Main`, just to hold the startup method (`Main`). Creates the `App` and the UI.
 - `Recorder`, is a full-featured pipeline for audio recording from the source to a compressed (Vorbis algorithm) OGG file. Easily reusable for the next lab for example.
 - `Settings`, uses the [Java standard Preferences API](#) to manage user settings.
- `se.ltu.M7017E.lab1.ui`
 - `Gui` is the main class for the User Interface with the main window (frame).
 - `PreferencesDialog` is the window for user settings.
 - `SpringUtilities` is a borrowed utility class from <http://docs.oracle.com/javase/tutorial/uiswing/examples/layout/SpringGridProject/src/layout/SpringUtilities.java> to ease the creation of Spring layouts

There is only one unit test (for the Preferences API) because it is hard to code tests for a UI, and it was also difficult to test the backend due to its multimedia nature. But we did extensive integration tests by ourselves.

Code design guideline

The program is mainly separated in two parts: the UI and the backend. The backend is in charge of the multimedia and the user preferences. The guiding principle was to really separate both: the UI depends on the backend but not the opposite (you can delete the UI package without any consequence, this is a one-way relation). Therefore the code is clear and well organized and if needed another UI (for command line or remote control over network) can easily replace the graphical one.

This is a best practice in software engineering when creating an application with a UI.

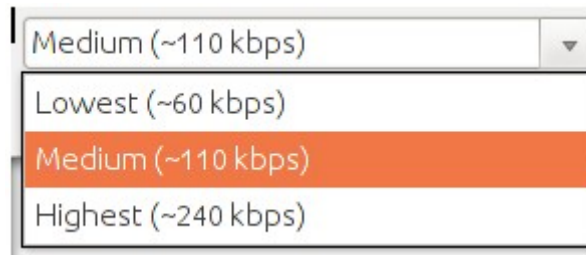
Methods description

The methods are documented with Javadoc comments. Please refer to it.

Data types and structures

Our software does not need specific data types other than the ones already provided by Java.

One minor exception though, in `PreferencesDialog` we needed to show the `JComboBox` (dropdown list) with understandable (textual) label while associating to each a float value:



Here for example "Lowest" is associated to the quality 0.1 and "Highest" to the quality 1.0 (they relate to properties of the [vorbisenc filter](https://xiph.org/vorbis/doc/vorbisenc_filter.html)). To do this we created a class associating a label and a value as advised by <http://stackoverflow.com/a/5661669>.

Version control

We have used Git and [Github](https://github.com) to share our files within the team.

4. Algorithm Description

There is no specifically clever algorithm in our application to explain. Even the list of the audio files available to play is generated through Java File API call <http://docs.oracle.com/javase/6/docs/api/java/io/File.html#listFiles%28java.io FilenameFilter%29> and by passing a FilenameFilter accepting only "*.ogg" files.

5. Problems and Reflections

We decided to keep this application simple for the user. Therefore, we did not implement some features such as deleting or renaming a file, since it can be done by using the standard file explorer. This application is so dedicated to record and play an audio file.

It took us some time to make the application and the user interface work well together. Like explained above, the user interface is not interfering with the application and can be easily replaced or modified.

We also tried to apply some automatic gain, as used in some famous applications which use the microphone. This automatic gain is supposed to adjust the level of the sound while recording, for example if you are too far away from your microphone. We tried to use some *Gstreamer* elements (volume and level in the Base and Good plugins), but we would have needed more time to make it work good enough.

6. Contribution

Clément Notin

Technical leader, architect and developer on every class.

Hervé Loeffel

Developer, especially on the file playing/pausing, saving and events management (double click,etc..) parts.

Code of honour

We state that our project is compliant with the code of honour. All the production is our original team work, except for the external libraries (JAR files) and explicitly marked code (SpringUtilities class).