1 CS1OOP Assessed JavaFX Lab

1.1 Welcome

This is the worksheet for the CS1OOP assessed JavaFX lab. Your submission to Blackboard will be worth 40% of the continuous assessment marks of term 2.

1.1.1 Introduction

In this assessed lab you will implement a simple GUI. All material needed for the lab can be found on Blackboard.

You should already have practised creating a simple GUI from the prep lab. You must complete this work on your own. You must upload your code to Blackboard by the specified deadline.

You will have unlimited attempts to re-submit until the deadline, so feel free to submit early and often.

1.1.2 What you should already know

You should have completed the preparatory lab first. You should be able to:

- Use the Scene Builder to create scenes in FXML format.
- Write Java code to load the FXML files from your application.
- Use JavaFX containers to position and dimension components.
- Write JavaFX controller classes to associate behaviour with GUI elements.

1.1.3 What you'll need

You will need a computer with this software installed:

- A Java 11 or newer JDK: download it from Oracle or AdoptOpenJDK
- Eclipse for Java Developers
- Oracle JavaFX Scene Builder 2.0, which requires registering for a free Oracle account. If this is a problem for you, try the Gluon SceneBuilder, but please do not use any of the Gluon-specific components as these are not included with standard JavaFX 11 distributions.

If you are using a lab computer or a student VDI machine, launch Eclipse 2019-06 through AstonApps (you will need to launch Java 11.0.5 first). This should also set up the JavaFX SceneBuilder 2.0 for you: to start the Scene Builder, go to the Windows Start menu and search for "Scene Builder".

1.1.4 What you'll do

As a quick summary, you will start from a model solution of the JavaFX preparatory lab that is on Blackboard, and make these improvements:

- Create an FXML file for a new "Create Video" dialog according to a screenshot.
- Use the appropriate JavaFX containers to ensure the controls resize and reposition themselves according to another screenshot.
- Add a new "Add" button to the main window that shows this new dialog.
- Add a controller class that creates the new `Video` object and adds it to the `Database` if the user clicks on "Confirm".

Check the next sections for the details!

1.2 Importing the project

Download the starting code from Blackboard. You don't need to unpack the `.zip` file.

Import the project by using "File - Import - Existing Projects into Workspace", then pick "Archive file" and select the `.zip` file you downloaded.

One project called `uk.ac.aston.oop.javafx.assessed` should come up. Press OK to import this project.

The project is a model solution of the preparatory lab: you should already be familiar with its structure.

Try running the `Main` class now, so you can be sure that the program works before you start changing it!

If you are using a Mac and the code runs but nothing is shown, go to "Run - Run Configurations… - Main" and make sure that on the "Arguments" tab, the option "Use the -XstartOnFirstThread argument when launching with SWT" is unchecked.

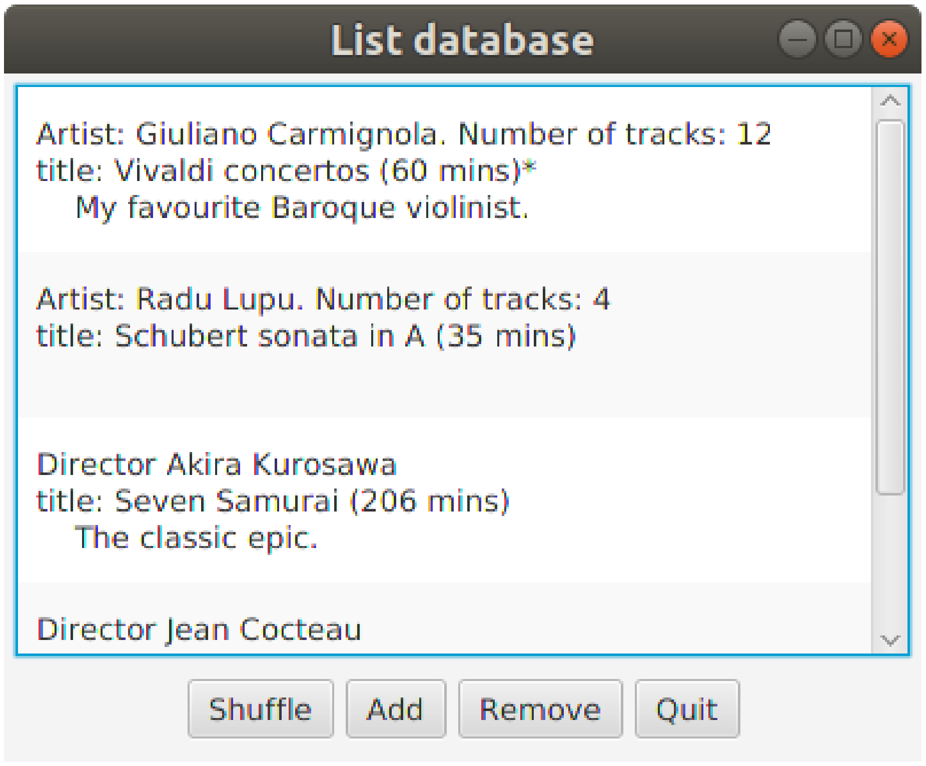
1.3 Task to complete

The task is to add an "Add" button to the GUI, which will show a dialog for entering the information of a new `Video`. If the user fills in the details and confirms the addition, then a new `Video` should be added at the end of the items in the `Database` and should be listed in the main window. If the user cancels the dialog, the dialog should simply close itself without changing the `Database`.

Further guidelines on how to accomplish some of these requirements are shown in the next section, and staff will be available during scheduled lab sessions to help with tooling issues and clarify the worksheet.

1.3.1 Task details

The main dialog should end up looking like this:



Artist: Giuliano Carmignola. Number of tracks: 12
title: Vivaldi concertos (60 mins)*
    My favourite Baroque violinist.

Artist: Radu Lupu. Number of tracks: 4
title: Schubert sonata in A (35 mins)

Director Akira Kurosawa
title: Seven Samurai (206 mins)
    The classic epic.

Director Jean Cocteau

Figure 1: New main dialog with Add button

The "Create Video" dialog (which should go into a `CreateVideo.fxml` file in the same folder as the other `.fxml` files - use exactly this name, including uppercase and lowercase letters) should look like this:

You are free to use any combination of JavaFX containers to achieve the above layout. You must ensure that the text field for the title is to the right of the "Title:" label, that the text field for the director is to the right of the "Director:" label, that the slider for the playing time is to the right of the "Playing time:" label, and that the components are vertically ordered as shown on the picture (title, director, playing time slider, own, comment, and buttons).

The "Own" checkbox reflects whether the `Video` is owned or not: `Item` has a pair of `getOwn`/`setOwn` methods to retrieve and store this information.

You should make it so the "Create Video" dialog can be resized, stretching as shown here:

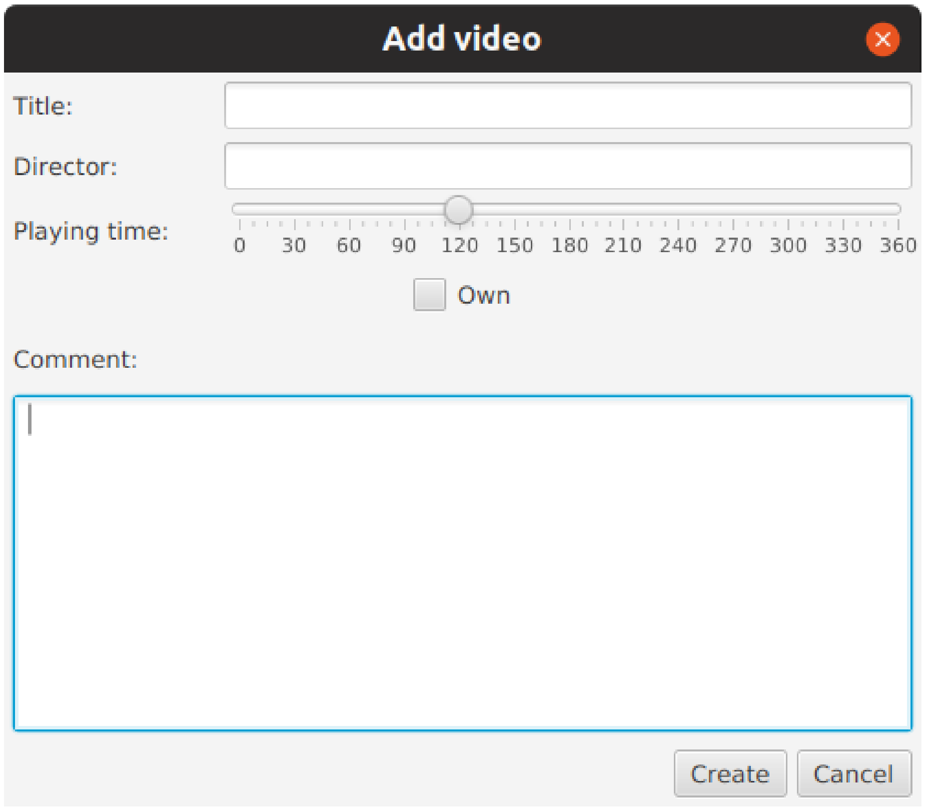


Figure 2: New Create Video dialog

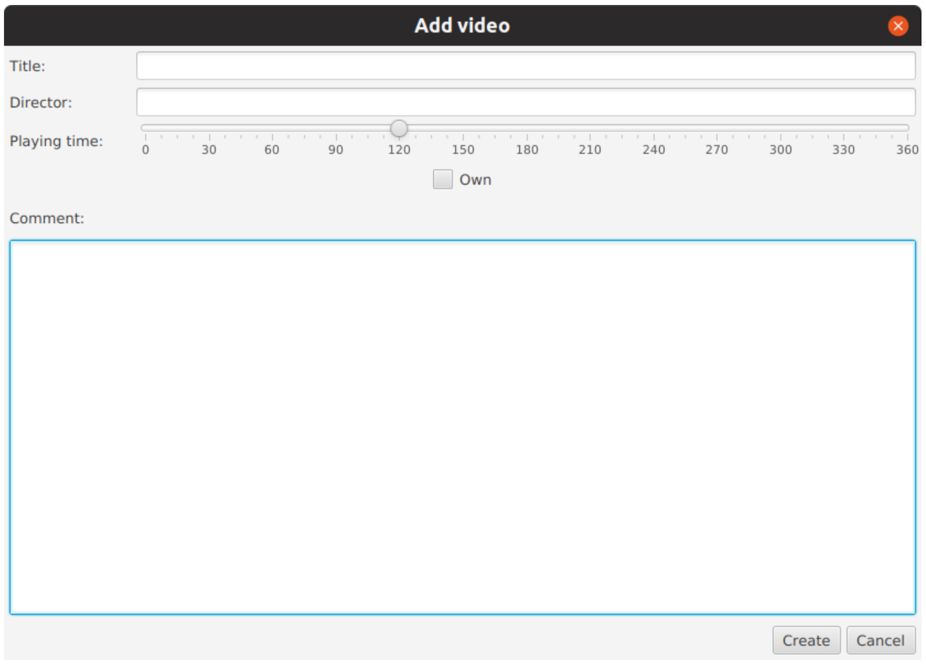


Figure 3: Resized Create Video dialog

Our tests will check that your text fields and sliders grow to take on the additional horizontal space, and that making the window taller will result in the text area becoming taller, and the buttons moving further down but everything else staying in place.

1.3.2 Marking scheme

The marking scheme is out of 14, as follows:

- "Add" button in main window: 1 mark
- "Add" button shows "Create Video" dialog: 1 mark
- "Create Video" dialog has all the components: 2 marks
- "Create Video" components have the expected relative positioning: 2 marks
- "Create Video" resizes horizontally correctly: 1 mark
- "Create Video" resizes vertically correctly: 1 mark
- "Create Video" has slider ticks for "Playing time": 1 mark
- "Create Video" dialog is application modal: 1 mark
- "Create" button in "Create Video" works when "Own" is not checked: 1.5 marks
- "Create" button in "Create Video" works when "Own" is checked: 1.5 marks

- "Cancel" button in "Create Video" works: 1 mark

1.4 Guidelines

You should already have a general idea from the preparatory lab. Regardless, here are some guidelines to help you along.

1.4.1 Extend the main scene with an "Add" button

The first task is to extend the main window with the "Add" button. Edit the `ListScene.fxml` file with the Scene Builder to do so.

Next, you should extend the `ListController` so it can handle the event when the user clicks on the button. As a quick test, just have it print something on the console for now.

1.4.2 Design the "Create Video" scene

Once the "Add" button can print something, work on a first version of the FXML file for the "Create Video" dialog. Some pointers:

- First of all, think of which containers you might want to use. You will generally want to combine different containers. Hint: our model solution uses three different ones.
- Once you have the containers figured out, place the various components, without worrying too much about the exact looks:
  - 2 buttons
  - 1 checkbox
  - 4 labels
  - 1 slider
  - 1 text field
  - 1 text area
- After that, tweak the options in the slider. You will want to change the minimum/maximum values, the major tick units and the minor tick counts between each pair of major ticks.
- Add `fx:id`s for the text fields, the checkbox, the slider, and the text area.
- Add "onAction" handlers to the "Create" and "Cancel" buttons.

Your JavaFX Scene Builder preview should show a window that roughly looks like the one from the previous screenshots. You will have time to make it look nicer later. It is more important to add behaviour right now.

1.4.3 Write the skeleton of the “Create Video” controller

Create a skeleton of the controller class for your new window. To do this, the easiest way is to go through your `.fxml` file:

- For each `<Type ... fx:id="abc">`, add an `@FXML private Type abc;` field.
- For each `onAction="#yourMethod"`, add an `@FXML public void yourMethod() {}` method.

It is a good idea to copy and paste names from the `.fxml` file to your Java code, to avoid wasting time on typos. Leave the event handler methods empty for now.

1.4.4 Making “Add” display the “Create Video” dialog

Go back to `ListController` and revise the event handler for the “Add” button, having it create a new `Stage` that displays the “Add” scene you just designed. The new window should be modal, i.e. it should not be possible to interact with the main dialog until it is closed. You may want to consult the code of the `removePressed` method in the `ListController` for pointers on how to do this.

Check now that you can run the program, click on the “Add” button and see the window appear. Try clicking on the “Remove” button while “Add” is open: it shouldn’t activate. Try closing the “Add” window and then clicking on “Remove”: it should work now. Close the program for now.

1.4.5 Creating the Video when confirmed

Time to do the next bit: fill in the event handlers for the “Create” and “Cancel” buttons in the “Add” dialog.

The “Create” button should create the `Video` object from the current values in the form, add it to the `Database` and hide the window. Remember to use the checked/unchecked status of the “Own” checkbox (`Item` has a `setOwn` method).

The “Cancel” button should simply hide the window, without making any changes to the `Database`.

You should be able to see the new Video in the main window immediately: the `ListView` should have reacted to the change in the `itemsProperty()` of the`Database` automatically.

1.4.6 Fine tuning the resizing of the “Create Video” dialog

The last part is to improve the layout and styling of your “Create Video” dialog, so it looks closer to the picture.

Open your `.fxml` file again in the Scene Builder and fine tune the way your components react to changes in the size of the window. Consider alignment, horizontal/vertical growth, margins, padding, spacing and preferred/maximum sizes. Do not worry about minimal sizes for this lab.

Use the “Preview” menu and try resizing the window. The text fields and sliders should take up any additional space, and the buttons should stay at the bottom of the window.

1.5 Submit to AutoFeedback and Blackboard

Now that you have completed the task, you should use the `Submit Assessed JavaFX Lab to AutoFeedback` file that corresponds to your operating system to submit your code to AutoFeedback.

For the details on how to use AutoFeedback, please refer to the last section of Lab 1.

In this lab, we didn’t ask you to write any tests: instead, AutoFeedback has its own tests that check the various things you should have done as part of this lab. Keep in mind that many of the tests are brought over from the preparatory lab: those do not give you marks. The ones that count for marks are marked accordingly, and they are mostly in `CreateVideoFXMLTest`, `ListSceneFXMLTest`, and `MainWindowTest`. These tests add up to the same marks as the marking scheme: you mark will be the same as on AutoFeedback unless otherwise noted.

Make sure that you submit your code to Blackboard by the deadline, even if you do not pass all the tests. You can still get partial marks: see the marking scheme earlier in the worksheet.