Swing Layouts and Dialogs

Overview

In this lab, you will improve the Swing application you created in the previous lab. Specifically, you will use layout managers to take control over how the components are laid out in the window. You'll also add some dialog boxes to the application, to give the user some visual feedback when they perform various actions (e.g. removing an item).

Source folders

Student project: `StudentSwingLayoutsDialogs`
Solution project: `SolutionSwingLayoutsDialogs`

Roadmap

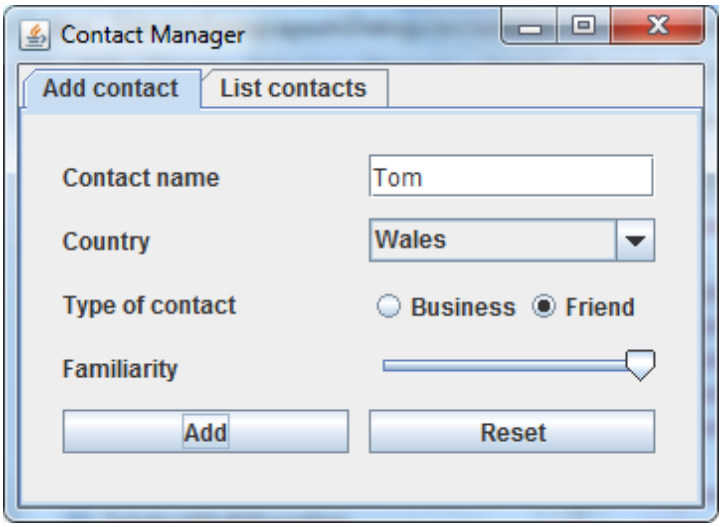
There are 4 exercises in this lab, of which the last exercise is "if time permits". Here is a brief summary of the tasks you will perform in each exercise; more detailed instructions follow later:

1. Familiarization with the solution application
2. Using a layout manager in the *Add contact* tab
3. Using a layout manager in the *List contacts* tab
4. Additional suggestions

Exercise 1: Familiarization with the solution application

Open the *solution* project. The functionality of the application is the same as in the previous lab, but we've used layout managers to improve the user interface.

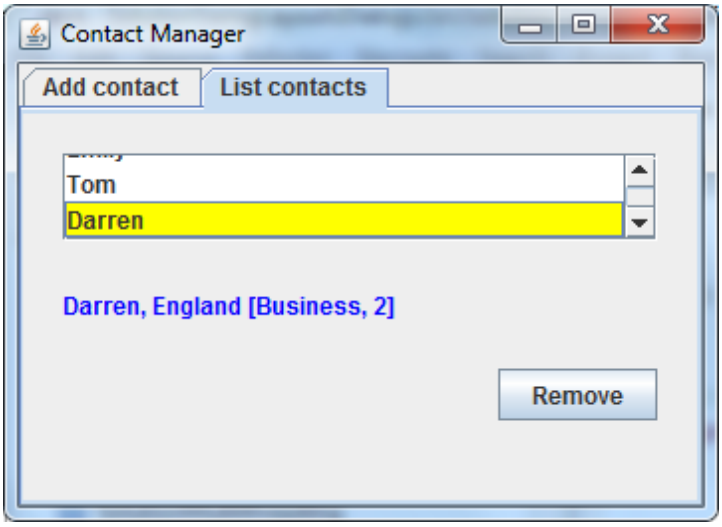
This is how the *Add contact* tab appears:



Note the following points:

- The components are laid out in a 2-column grid, via a `GridLayout` in the panel.
- There's a horizontal gap of 10 pixels between each row, and a vertical gap of 10 pixels between each row.
- Column 0 contains labels (plus the *Add* button).
- Column 1 contains input components (plus the *Reset* button).
- The *Business* and *Friend* radio buttons have been wrapped up in their own panel. A `FlowLayout` is applied to the panel, and the panel is then added to the grid.

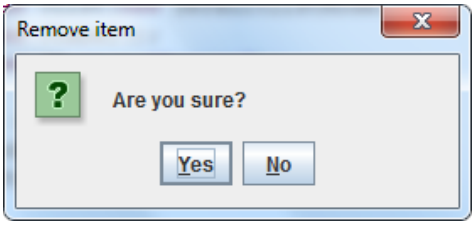
This is how the *List contacts* tab appears:



Note the following points:

- The components are laid out in a 1-column grid, via a `GridLayout` in the panel.
- The *Remove* button has been wrapped up in its own panel. A right-aligned `FlowLayout` is applied to the panel, and the panel is then added to the grid.

If you select an item in the list box, and then click the *Remove* button, a confirmation box appears as follows:



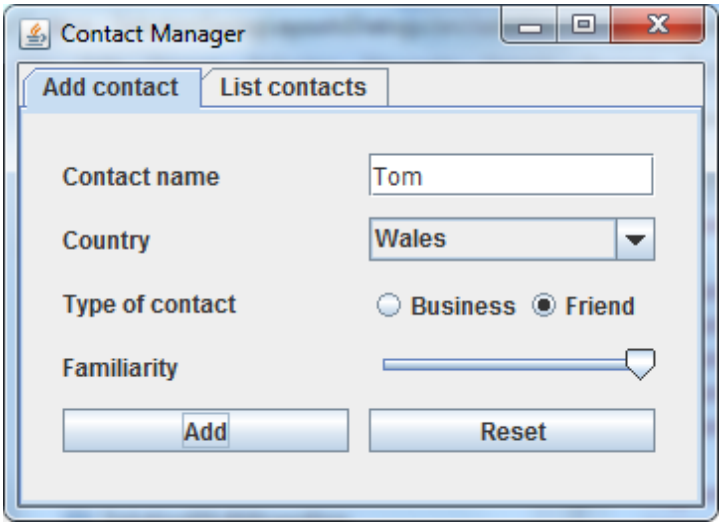
Conversely, if there is no item selected and you try to click the *Remove* button, and error message box appears as follow:

When you're happy with how everything works, close the application.

Exercise 2: Using a layout manager in the *Add contact* tab

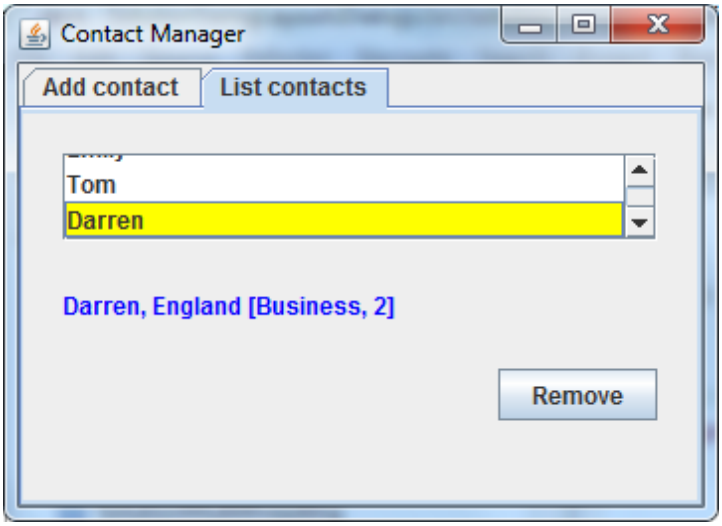
Switch to the *student* project and open the `ContactManager` class.

Locate the `createAddContactPanel()` method and refactor the code so that it displays the *Add contact* tab as per the solution application. Here's a reminder of the UI you are aiming to reproduce:



Exercise 3: Using a layout manager in the *List contacts* tab

Locate the `createListContactsPanel()` method and refactor the code so that it displays the *List contacts* tab as per the solution application. Here's a reminder of the UI you are aiming to reproduce (also, remember the confirmation / error message boxes for the *Remove* button):



Exercise 4 (If time permits): Additional suggestions

- Add an "edit" capability to the application, so that the user can double-click an item in the list box and edit the contact details. Hints: handle the mouse double-click event on the list box, and display the currently selected `Contact` object in a custom dialog box (you can reuse some panel-creation code from the `createAddContactPanel()` method).

- Experiment using a `GridBagLayout` rather than a `GridLayout`, to gain absolute control over how components are displayed in a grid.