Objectives:

1. Exercise skill in designing reusable collaborations.
2. Use mediators to synthesize a complex collaboration from simpler ones.

Description: In this project, you will develop an interactive itinerary-planning tool, which allows clients to plan trips to various North American cities using a variety of modes of transportation (e.g., air, rail, bus). Users should be able to plan a trip by specifying origin and destination cities, price ranges, and desired modes of transport. The tool should treat each consecutive choice (e.g., choice of a city of origin, choice of a price range, etc) as a constraint that limits the remaining choices and displays to the user only those remaining choices that are consistent with previous choices. For example, if a user specifies that he or she wishes to travel from Lansing to St Louis, then the tool should only allow the user to choose modes of transportation and price ranges that pertain to travel from Lansing to St Louis. By contrast, if a user specifies that he or she wishes to travel from Lansing to an unspecified location, but only wants to spend $200.00, then the tool should only display the destinations and modes of transportation that fit this budget.

The user interface of this tool should contain widgets (e.g., buttons, list boxes, and text fields) for specifying choices in three major categories—cities, mode, and pricing. Specifically:

Cities contains two list boxes, which allow the selection of origin and destination cities;

Mode contains 3 check boxes (labeled "bus", "train", and "air" respectively) and a list box for selecting the carrier that provides the selected mode of transportation. For example, if the "air" button is depressed, the list box will list air carriers (e.g., Northwest, United, Delta); whereas if the "bus" button is depressed, the list box will list van lines (e.g., Greyhound, Trailways, etc). If no button is depressed, the list box should be disabled.

Pricing contains two text-entry boxes that depict the lower and upper bounds of a price range and a drop-down menu that displays all of the available prices.

As mentioned previously, choices made in one category may constrain the choices in the remaining categories. For example, when an origin and destination are selected, the mode-of-travel and price categories should reflect only the available modes and prices for service between the selected cities. By contrast, when a mode of travel (and possibly a particular carrier) is selected, the list of destination cities (and price ranges) should reflect only those that are serviced by that mode (or carrier) from the city of origin. Finally, when a price range is selected, the modes of travel, carriers, and destination cities should reflect only those that are within that price range.

The user interface should reflect constraints by disabling any choices that (if made) would violate the constraint. Choices are disabled in different ways, depending upon the widget type used to make the choice. For example, a mode is disabled by deactivating the check box that corresponds to this mode; a choice of city (resp. carrier) is disabled by removing its name from the appropriate list box; and a choice of price is disabled by removing it from the drop-down menu and adjusting the lower and upper bounds of the price range (if appropriate). In addition to these selection categories,
the dialog should contain a button for finalizing the itinerary. This button should only be enabled when a complete and unique itinerary has been specified. When pressed, this button will cause the itinerary to be printed to standard output. The dialog should also contain a reset button, which when pressed will reset each selection category (thus removing all constraints) so that the user may begin anew.

**Tasks:** For this project, you must:

1. Design and implement role classes and collaborations to support each of the different widget types that are required to implement this example (check-box panels, list boxes, drop-down menus, and text fields).

2. Design an itinerary-planning widget\(^1\) by synthesizing the collaborations you designed in step 1 and introducing mediators where appropriate to reduce the number of interactions.

Item 1 will require you to design classes that wrap the existing fltk classes to support a role-based design. Note that fltk was *not* designed using the principles of role-based design, which is why in the labs, we define classes such as ViewPort and Button that encapsulate the fltk classes and provide configuration/collaboration operations. Before designing your itinerary manager, you must first go build analogous collaborations for the kinds of widgets needed in this example:

**Check-box panel** is a group of check boxes with a label that describes their function. An example is the *mode-of-transportation* panel, which comprises 3 check boxes (labeled “air”, “rail”, and “bus”) and a label “Mode of transportation” that describes the function of these buttons. It should be possible to enable and deactivate an entire check-box panel, and it should be possible to deactivate individual buttons within the panel.

**List box** is a scrollable viewport whose lines correspond to selectable items. The user selects an item by clicking anywhere on the line that contains the item.

**Drop-down menu** is a button that, when pressed, opens a menu with a fixed set of selectable items.

**Text box** is a field that allows the display and editing of text strings.

Each of these widgets must be designed as a reusable collaboration, which might involve multiple role classes, some of which are abstract. To be reusable, these collaborations must not contain any information that is specific to the itinerary-management problem. **You will be graded heavily on this point.** Moreover, the code that implements your itinerary-management tool should be designed by synthesizing these collaborations and should not reference or use the primitive fltk classes directly.\(^2\) **You will be graded heavily on this point as well.**

Implementing these widgets will require you to learn a little bit about some existing classes in the fltk user-interface toolkit. To get at the fltk documentation, point your browser to the URL:

```
http://www.fltk.org/doc-1.1/toc.html
```

\(^1\)similar to the dialog box illustrated the Gamma chapter on Mediators

\(^2\)The one exception is that your dialog box may inherit from Fl_Group.
You should look specifically to how Fltk currently handles check boxes, list boxes, menus, and text boxes\textsuperscript{3}.

Note: We will post a file of available routes, including mode of transportation, source and destination cities, and price. Your program should read in this file and use these data.

\textbf{Deliverables}: You must submit all of your source files and the Makefile to the handin system. It should be possible for the TAs to just type ‘make’ without any additional editing or copying of files into the working directory. All of these files are due by 11:59PM on Sunday, November 9th.

\textsuperscript{3}The \texttt{IvalText} class that you’ve used in the labs might be a good starting point.