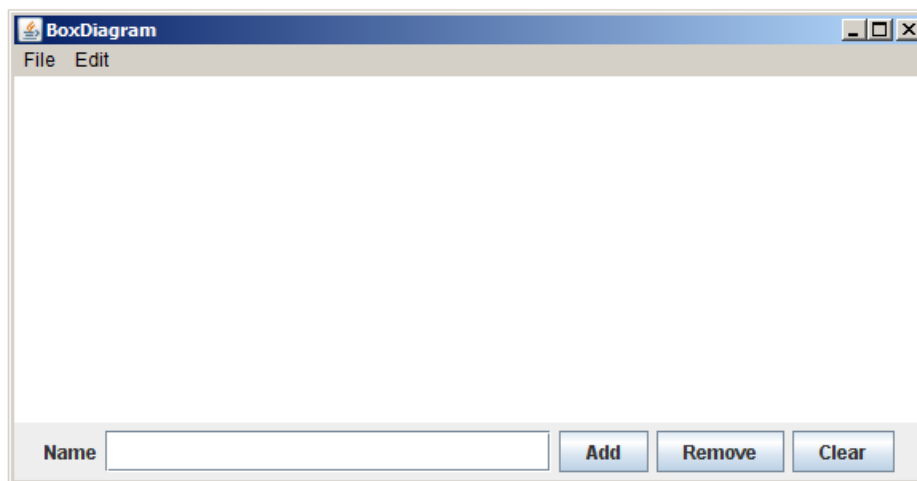


Section Handout #7: Using Interactors

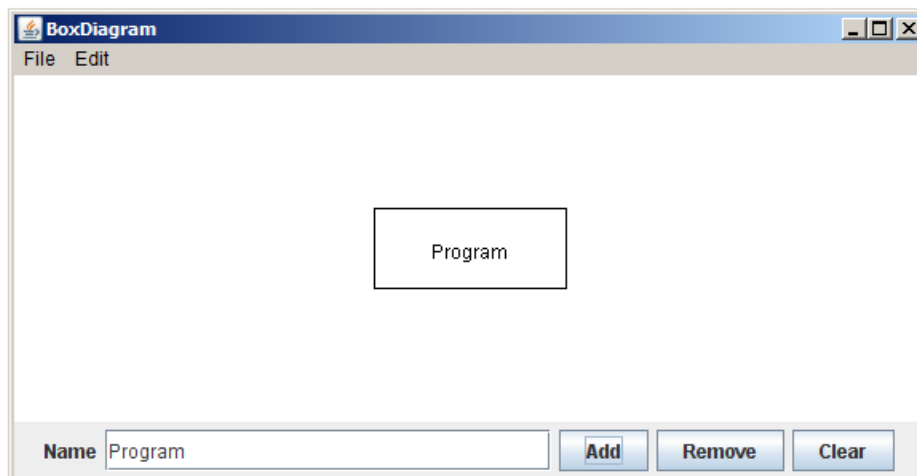
Based on a handout by Eric Roberts

The purpose of this section is to give you some practice using the kind of interactors you need for the NameSurfer project in Assignment #6.

The specific example is to build a framework for an interactive design tool that allows the user to arrange boxes with labels on the window. The starting configuration for the program presents an empty graphics canvas and a control bar containing a `JLabel`, a `JTextField`, and three `JButtons`. The window as a whole then looks like this:



The most important operation in the program is to be able to add a new box to the screen, which you do by typing the name of the box in the `JTextField` and clicking the `Add` button. Doing so creates a new labeled box with that name in the center of the window. For example, if you entered the string `Program` in the `JTextField` and clicked `Add`, you would see the following result:

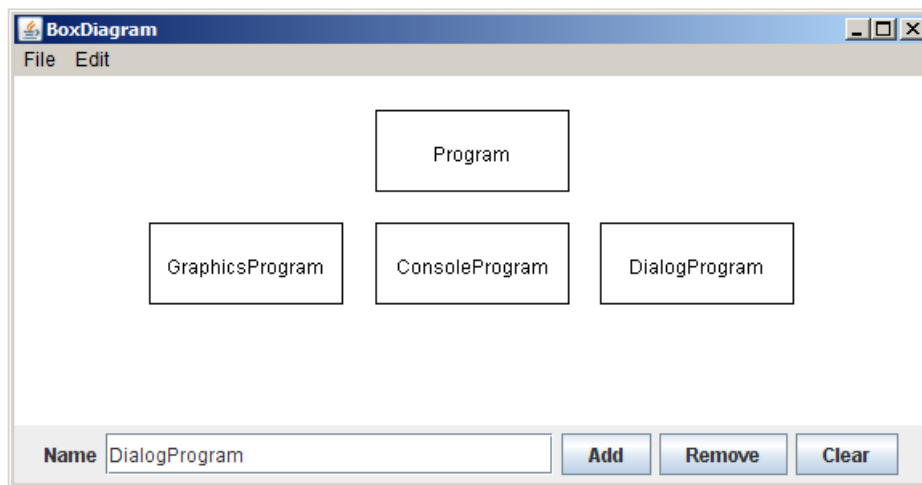


For this simple version of the application, you may assume that the box always has constant dimensions, as specified by the following constant definitions:

```
private static final double BOX_WIDTH = 120;
private static final double BOX_HEIGHT = 50;
```

Once you have created a labeled box, your program should allow you to move the box as a unit by dragging it with the mouse. Because the outline and the label must move together, it makes sense to combine the `GRect` and `GLabel` into a `GCompound` and then use code similar to that in Figure 10-4 in the book to implement the dragging operation.

The ability to create new boxes and drag them to new positions makes it possible to draw box diagrams containing an arbitrary number of labeled boxes. For example, you could add three more boxes with names `GraphicsProgram`, `ConsoleProgram`, and `DialogProgram` respectively, and drag them around to create this diagram of the `Program` class hierarchy:



The other two buttons in the control strip are **Remove** and **Clear**. The **Remove** button should delete the box whose name appears in the `JTextField`; the **Clear** button should remove all of the boxes. Note that you do not need to worry about the user trying to create multiple boxes with the same name—you can assume that all box names are unique.

While the operations in this program are conceptually simple, they influence the design in the following ways:

- The fact that you may need to remove a box by name forces you to keep track of the objects that appear in the window in some way that allows you to look up a labeled box given the name that appears inside it. You therefore need some structure—and there is an obvious choice in the Java Collection Framework—that keeps track of all the boxes on the screen.
- If the only objects in the window were labeled boxes, you could implement the **Clear** button by removing everything from the `GCanvas`. While that would work for this assignment, you might want to extend the program so if there were other objects on the screen that were part of the application itself, they would remain on the screen. In that case, you would want to implement **Clear** by iterating through the set of boxes on the screen and removing each one. Remember, you can assume all box names are unique.