Objective: Exercise skill in developing and managing multiple collaborations among multiple, overlapping groups of objects.

Description: In lab 7, you developed a version of the graphical browser that enables users to browse items supplied by a data service over the network. In this lab, the instructors provided a class called GUIManager, which:

- services mouse and keyboard events (invoking application callbacks as appropriate); and
- periodically “takes a breath” by notifying objects that implement the GUIManagerEventListener interface.

By “taking a breath”, we mean that the GUI manager temporarily cedes control to any listening objects before continuing to monitor the mouse and keyboard. In the lab, we had you extend class (BufferedNetworkStream) so that when the GUI manager takes a breath, the BufferedNetworkStream object is invoked to check the network for new browsable data. Conceptually, this is a nice solution; however such a design is very sensitive to transmission delays from the server to the browser. To see this, set the Perlisim server to emit data with minimum delay (i.e., 0.0 in the value slider) and observe the responsiveness of the browser to scrolling actions, such as rapidly dragging the slider handle up and down. Then go set the delay to maximum and repeat the GUI interaction. You should notice that the browser GUI behaves much more erratically in this case. This choppiness occurs because GUI manager cedes control to the BufferedNetworkStream object by invoking a function (serviceNetwork) on this object, and this function call blocks until there is data to retrieve from the network. With a long transmission delay, this blocking becomes noticeable.

The only way to really solve this problem is to use a multi-threaded browser, such that “the GUI” runs in a thread of control that is separate from and concurrent with a thread that waits for data over the network. For this lab, you must extend your solution to Lab 7 in this fashion. Be forewarned: With the use of multiple threads comes the potential for new classes of run-time errors (e.g., race conditions and deadlocks) that can be very difficult to track down. While it can be very instructive to diagnose and fix such bugs, we recommend that you follow some design guidelines that will help to prevent their introduction. First, before you even think about adding the new thread, you should refactor your solution to Lab 7 so that the GUI manager and the BufferedNetworkStream object do not interact directly. Rather, the interaction of these objects should be managed by a mediator.
object, which listens for breaths from the GUI manager and for the notification of new data from the network stream. When notified of new data by the streaming manager, the mediator should not immediately update the GUI objects. Rather, it should record the salient facts about the new data and then wait for notification of a GUI breath before updating any GUI objects. By waiting for a GUI breath, the mediator can be sure that the GUI manager is not (concurrently) modifying any of the GUI objects, which means that it will be safe to update them.

Note, you will need to worry about race conditions in the design of the mediator itself. You should use the same solution for synchronizing access to the mediator that you used in lab 8 to synchronize access to the BufferedNetworkStream. In this lab, you may use all of the materials that were available to you for lab 7. Specifically, you can use all of the files in /user/cse370/Labs/Lab7.

**Tasks**

1. Invent a class GUIStreamMediator, which mediates the GUIManager and BufferedNetworkStream objects. Refactor your solution to lab 7 so that all communication between the network stream and the GUI goes through this mediator, and then show your solution to the TA.

2. Rewrite your main application so that it creates a new thread in which to run the BufferedNetworkStream. **Note:** This solution will have race conditions, which means that your program may fail or produce unexpected results on some runs. We will rectify this situation in next week’s lab.