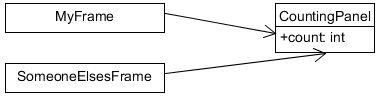
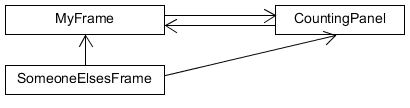
Observer Live coding

In the **problem package**, the students are presented with two JFrames that contain CountingPanels:



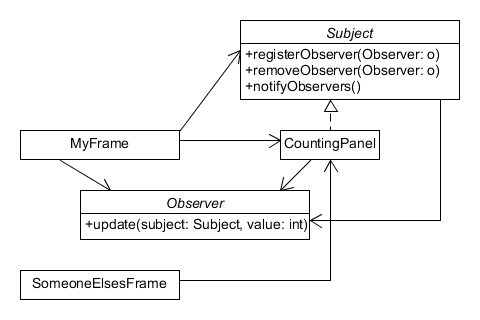
When you click a button in a CountingPanel, its count goes up by one. One of the frames, MyFrame, wants to sum up all of the counts and display the sum on the screen. The way the code is currently written, MyFrame is polling each of its CountingPanels at regular intervals. But polling is inefficient, but the big problem is that we rely on implementation details of the CountingPanels because we access the count variable to get the current count. This access creates pretty tight coupling from the MyFrame to the CountingPanels. Can we make a better design?

Our first intuition is to use our CSSE220 design skills to tell CountingPanels about our MyFrame. This way, the CountingPanels can push updates to our MyFrame whenever there is an update. As the **problem\_attempt1** package shows, this instead makes the coupling even worse because now we have tight coupling in both directions:



The problem manifests spectacularly for poor old SomeoneElsesFrame, which also uses CountingPanels. As the comments indicate, the last thing they want is to know about is a redundant MyFrame. But now, to avoid compile errors, these suckers have to use a MyFrame anyway! Indeed, this code will dump nasty errors to the console if you click one of the buttons at index >2, because MyFrame only knows about 2 buttons. The double arrows pointing between MyFrame and CountingPanel are a code smell that indicates tight coupling. Let’s fix that with the Observer pattern.

The **solution** package follows Freeman’s observer pattern to decouple MyFrame from CountingPanels in both directions. We add two interfaces: Subject and Observer. CountingPanels become Subjects. We create Observers in MyFrame and pass them to the CountingPanels to update the sums:



This indirection decouples the two classes, making SomeoneElsesFrame happy, while still giving us push capability.