Earlier, we showed how to register an anonymous function to listen to a click on a JButton. For a button declared like this:

**private** JButton eastButton= **new** JButton("east");

to register an anonymous function that will call buttonClick to process a click on the east button, use

1. eastButton.addActionListener(e -> buttonClick(e));

However, if the same method buttonClick is to be called for several buttons, it is better to create only *one* anonymous function and use it several times, as show below. Parameter type ActionEvent is unnecessary, but we put it there so you know what the type is.

1. ActionListener al= (ActionEvent e) -> buttonClick(e);
2. westButton.addActionListener(al);
3. eastButton.addActionListener(al);

Now, for any other even on some other component, we need to give you the same kind of information: The type of the function (ActionListener for a JButton) and the type of parameter e (above, ActionEvent). We can do that by showing you line (1), which tells you the type of the anonymous function and what its parameter is. With that, you can decide to register a listener using something like lines (2..3) or using something like line (1).

Keyboard clicks

JTextField. The following listens to enter/return hits.

ActionListener jtfl= (ActionEvent e) -> makeUpperCaseField(e);

JComboBox

ItemListeners are notified when ever the state of the button is changed, whether through a user interacting with the button or programmatically (via the setSelected method). ActionListeners on the other hand will be called when a user interacts with the button (but can be simulated programmatically via the onClick method).

Note that a user interacting with the button such as clicking or hitting the space bar will also change the state of the button and raise an item event as well as an action event. Generally, you will want to define either one or the other, don't listen for both action events and item events on the button.

The difference is that ActionEvent is fired when the action is performed on the JCheckBox that is it's state is changed either by clicking on it with the mouse or with a space bar or a mnemonic. It does not **really** listen to change events whether the JCheckBox is selected or deselected.

For instance, if JCheckBox c1 (say) is added to a ButtonGroup. Changing the state of other JCheckBoxes in the ButtonGroup will not fire an ActionEvent on other JCheckBox, instead an ItemEvent is fired.

**Final words:** An ItemEvent is fired even when the user deselects a check box by selecting another JCheckBox (when in a ButtonGroup), however ActionEvent is not generated like that instead ActionEvent only listens whether an action is performed on the JCheckBox (to which the ActionListener is registered only) or not. It does not know about ButtonGroup and all other selection/deselection stuff.