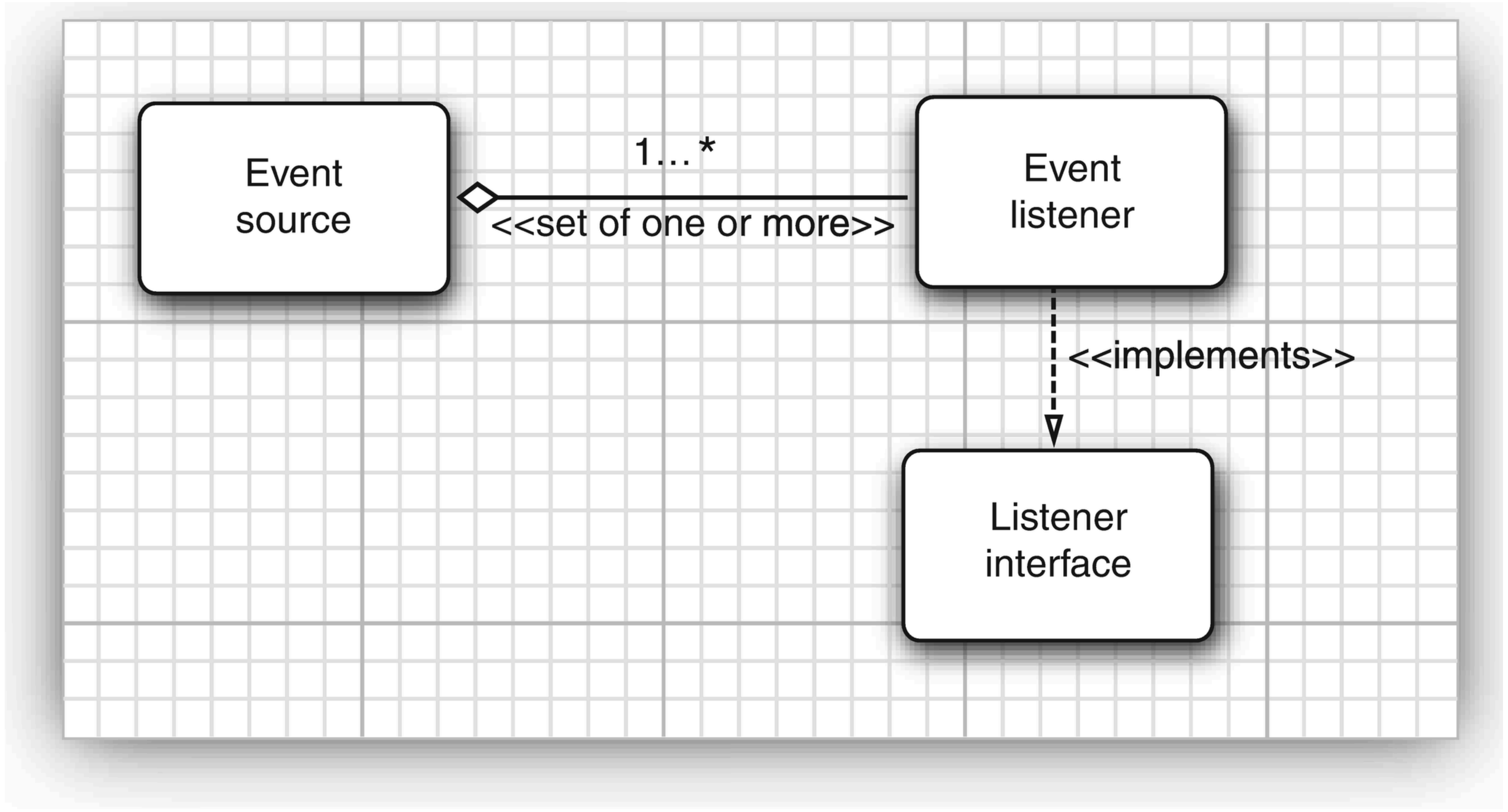
Basics of Event Handling

- Any operating environment that supports GUIs constantly monitors events such as keystrokes or mouse clicks.
- The operating environment reports these events to the programs that are running
- events are transmitted from the *event sources* (such as buttons or scrollbars) to *event listeners*.
- You can designate *any* object to be an event listener—in practice, you pick an object that can conveniently carry out the desired response to the event.

- Event sources have methods that allow you to register event listeners with them.
- When an event happens to the source, the source sends a notification of that event to all the listener objects that were registered for that event.

- A listener object is an instance of a class that implements a special interface called (naturally enough) a *listener interface*.
- An event source is an object that can register listener objects and send them event objects.
- The event source sends out event objects to all registered listeners when that event occurs.
- The listener objects will then use the information in the event object to determine their reaction to the event.

Relationship between event sources and listeners

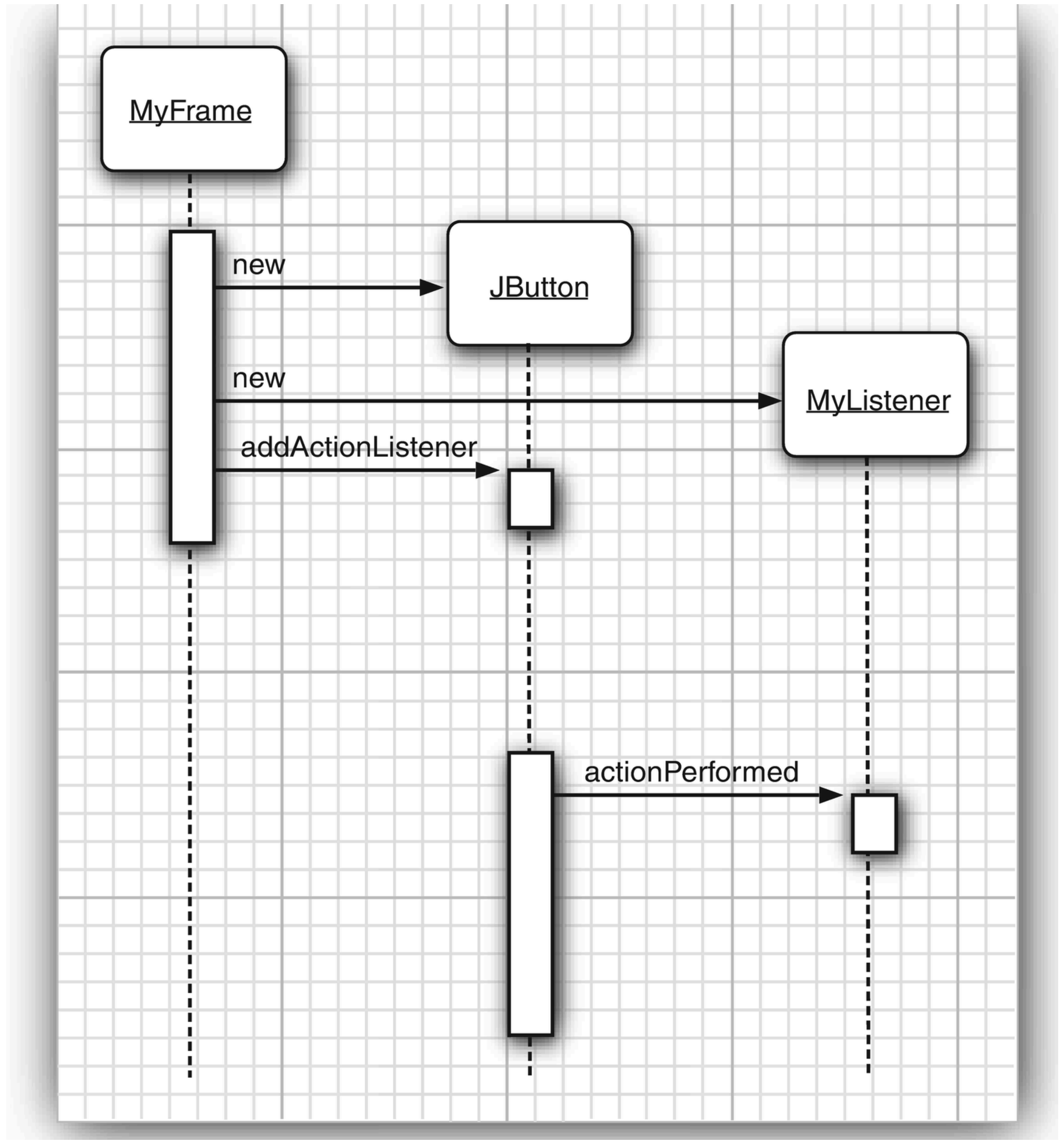


- ActionListener listener = . . .;
- JButton button = new JButton("OK");
- button.addActionListener(listener);
- listener object is notified whenever an “action event” occurs in the button.
- For buttons, an action event is a button click.

To implement the ActionListener interface, the listener class must have a method called actionPerformed that receives an ActionEvent object as a parameter.

```
class MyListener implements ActionListener
{
. . .
public void actionPerformed(ActionEvent event)
{
// reaction to button click goes here
. . .
}
}
}
```

interaction between the event source, event listener, and event object



- JButton yellowButton = new JButton("Yellow");
- JButton blueButton = new JButton("Blue");
- JButton redButton = new JButton("Red");
- To add it to the container
  - buttonPanel.add(yellowButton);
  - buttonPanel.add(blueButton);
  - buttonPanel.add(redButton);

- class ColorAction implements ActionListener
- {
- private Color backgroundColor;
- public ColorAction(Color c)
- {
- backgroundColor = c;
- }
- public void actionPerformed(ActionEvent event)
- {
- // set panel background color
- . . .
- }
- }

- ColorAction yellowAction = new ColorAction(Color.YELLOW);
- ColorAction blueAction = new ColorAction(Color.BLUE);
- ColorAction redAction = new ColorAction(Color.RED);
- yellowButton.addActionListener(yellowAction);
- blueButton.addActionListener(blueAction);
- redButton.addActionListener(redAction);