Object-Oriented Programming

Event Handling

Outline

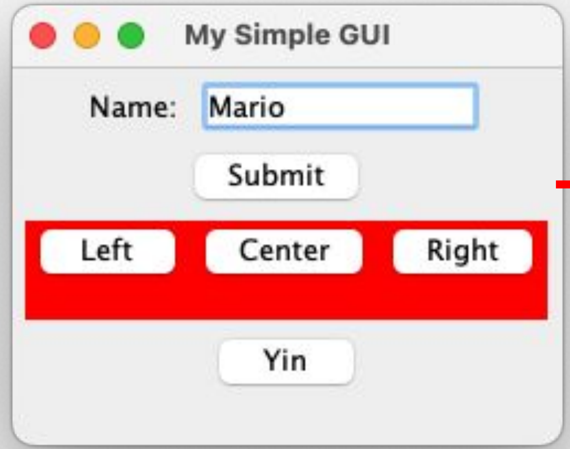
- Defining an event
- Components of an event
- Types of (AWT) Events

What is an event?

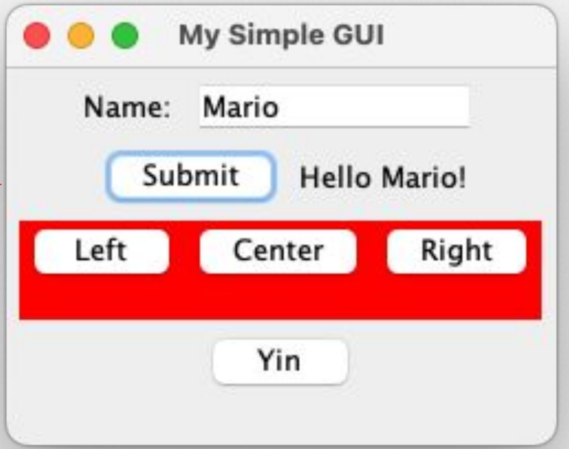
Event

- Represents a change in state
    - A button was pressed
    - A text field received another character
    - A key was pressed
    - A download finished
- Can happen in the foreground (e.g. UI) or in the background (e.g. running service)

What we've observed so far…



Click on submit button



Listener waits for an event to happen

Action object is passed

```
JButton greetingBtn = new JButton("Submit");
greetingBtn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        greetingsLbl.setText("Hello " + greetingNameTf.getText() + "!");
    }
});
```

Components in Event Handling

- Event source
    - An object is created when an event occurs
        - Contains information about the action and source
    - In our previous example, this was an ActionEvent
        - Let’s take a look at ActionEvent quickly…

Source: https://www.thoughtco.com/event-2034091

What we’ve observed so far…

With **ActionEvent**, we can…

- getActionCommand() – returns string associated to the action
- getWhen() – timestamp of action
- getSource() – object where event occurred; typecast appropriately
- And others…

For example:

```
((JButton) e.getSource()).setEnabled(false);
```

```
JButton greetingBtn = new JButton("Submit");
greetingBtn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        greetingsLbl.setText("Hello " + greetingNameTf.getText() + "!");
    }
});
```

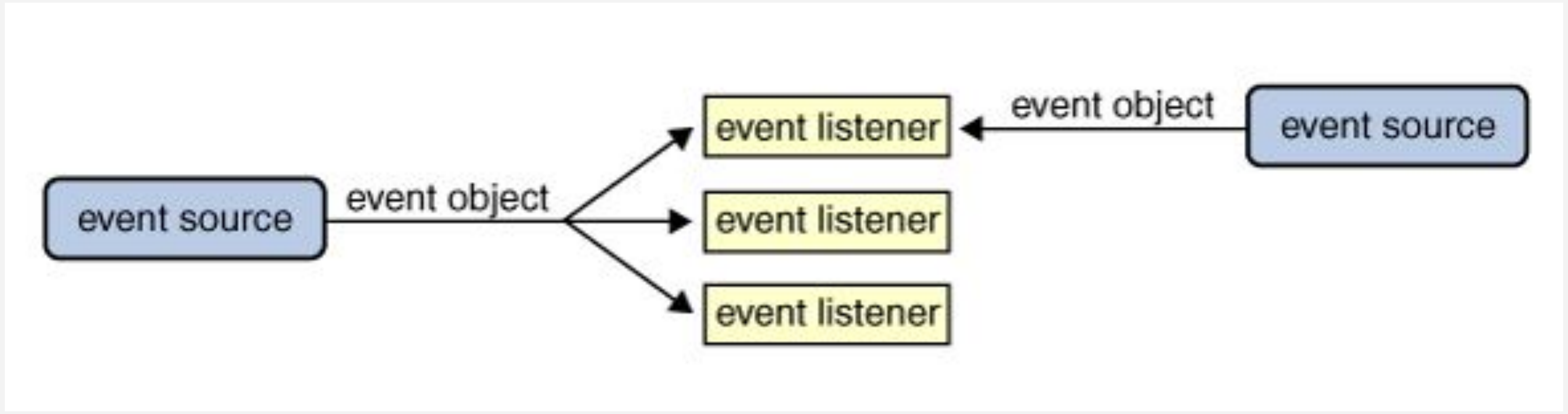
Questions?

Components in Event Handling

- Event **source**
  - An object is created when an event occurs
    - Contains information about the action and source
  - In our previous example, this was an ActionEvent
- Event listener
  - Actively listens or waits for an event to take place
  - Receives the event object upon an occurrence of an event

Source: https://www.thoughtco.com/event-2034091

Components in Event Handling



Notice how a source can have multiple event listeners.

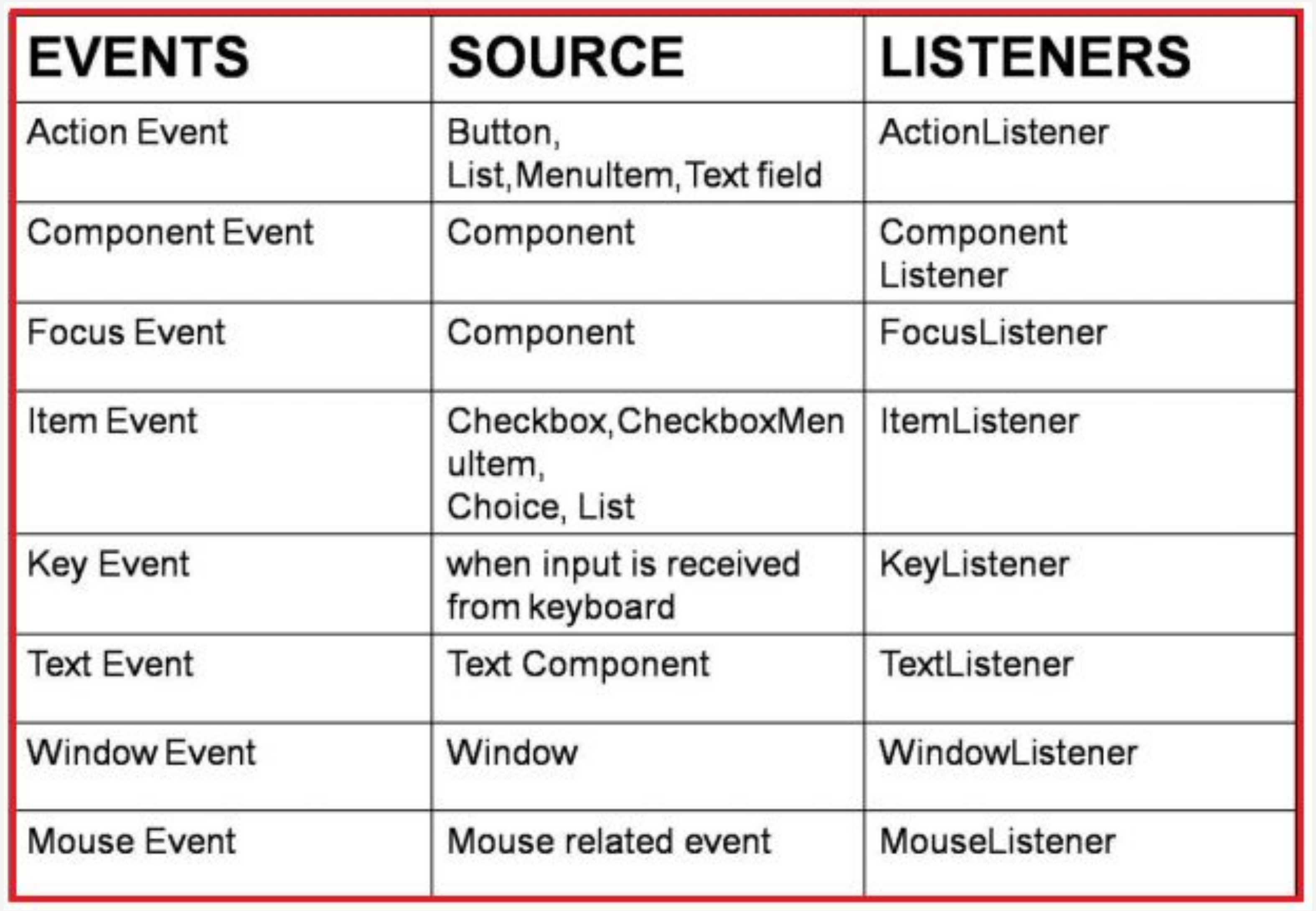
Additionally, a single event listener can also be associated to multiple sources (assuming the listener can be placed in the first place)

Image source: https://docs.oracle.com/javase/tutorial/uiswing/events/intro.html

Types of Events

Notice most in the table are GUI related

This is mainly because we're associating events with GUI interactions



EVENTS	SOURCE	LISTENERS
Action Event	Button, List, MenuItem, Text field	ActionListener
Component Event	Component	Component Listener
Focus Event	Component	FocusListener
Item Event	Checkbox, CheckboxMenuItem, Choice, List	ItemListener
Key Event	when input is received from keyboard	KeyListener
Text Event	Text Component	TextListener
Window Event	Window	WindowListener
Mouse Event	Mouse related event	MouseListener

You can always utilize the more generic **EventListener** and **EventObject** to accommodate your needs

Source: https://dotnettutorials.net/lesson/event-handling-in-java/

This code snippet is from an Android project (still written in Java)

```
public void setImage(String imageName) {
    // With the storageReference, get the image based on its name
    StorageReference imageRef = this.storageRef.child(imageName);
    // Download the image and display via Picasso accordingly
    imageRef.getDownloadUrl().addOnCompleteListener(new OnCompleteListener<Uri>() {
        @Override
        public void onComplete(@NonNull Task<Uri> task) {
            if(task.isSuccessful()) {
                Log.d("Debug", "onComplete: got image");
                Picasso.get()
                        .load(task.getResult())
                        .error(R.mipmap.ic_launcher)
                        .placeholder(R.mipmap.ic_launcher)
                        .into(imageIv);
            } else {
                Log.d("Debug", "onComplete: did not get image");
            }
        }
    });
}
```

Event listener

Source of event

Event object

What's happening here is that we're loading an image into a GUI element using the image's name. The string name is queried on a server and the image is eventually downloaded and inserted into the respective GUI element.

While this is a little more complicated than what we're used to, notice the same pattern being used

Questions?

Keep learning…