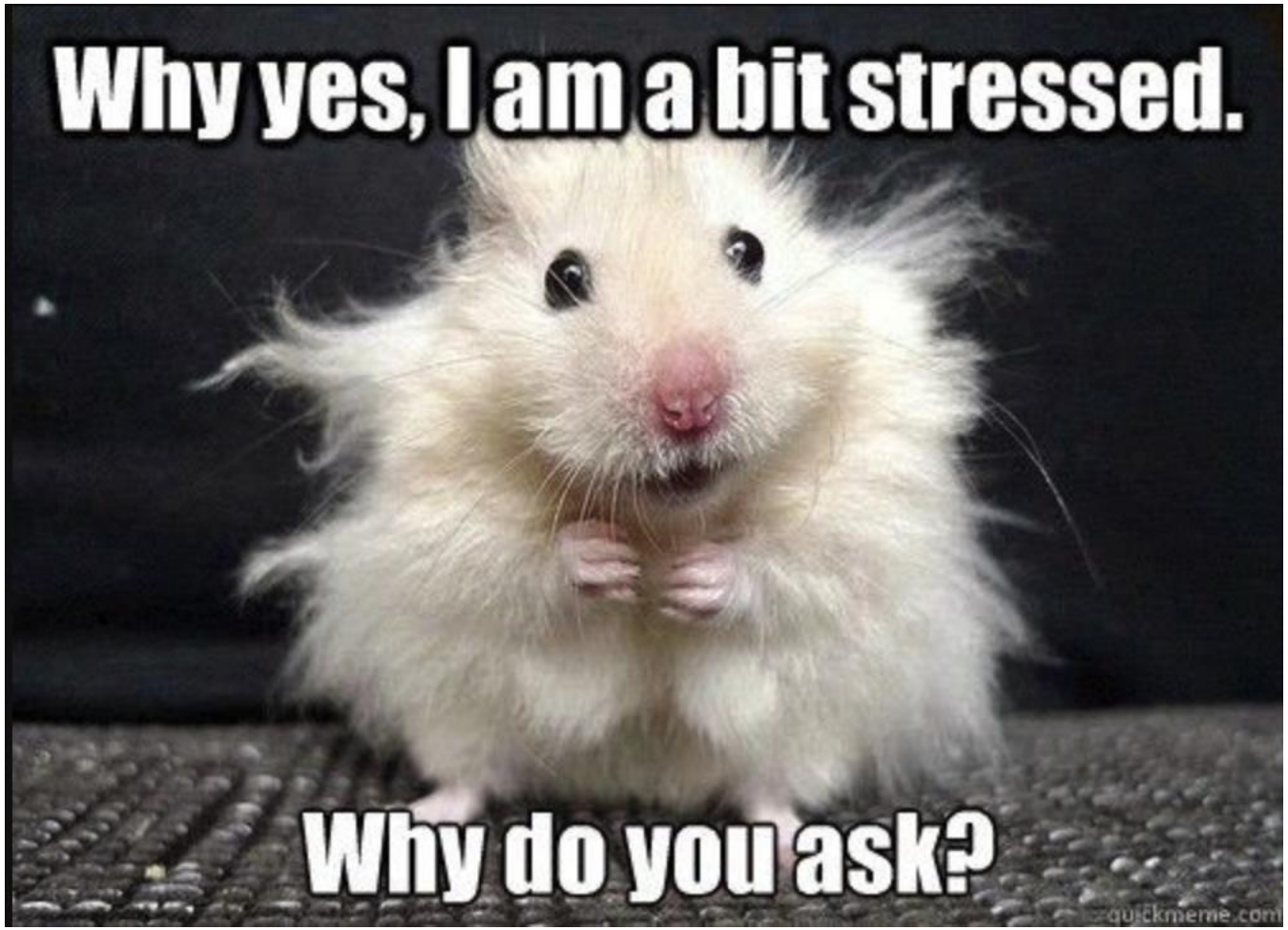
Temperature Check

On a scale of 1-10, where 10 is the most stressed out, how are you doing today?

PollEv.com/leahp
text **leahp** to **22333**

No correct answer. 7, 8, and 11 were the most common answers. Don't forget to focus on self care during this difficult time of the semester!



Picture: https://elgl.org/treat-yourself-caring-for-you-during-covid-19/stressed-meme/

Lecture 25: Event Driven Programming

CS 2110, Matt Eichhorn and Leah Perlmutter

November 20, 2025

Roadmap

Java, Complexity, OOP

- start– 9/30

ADTs I

- List, Stack, Queue, Iteration
  - 10/2 – 10/16

ADTs II

- Trees, Set, Map, Hash Table, Graph
  - Tues 10/21- 11/13

Beyond ADTs

- **Graphical User Interfaces & Event-Driven Programming**
  - 11/18, 11/20
- Parallel Programming
  - 11/25, 12/2
- Data Structures and Social Implications
  - 12/4

Swing Resources

- Swing Documentation – make use of the search bar!
- SwingSet2 – interactive gallery of different widgets, with source code demonstrating how to create them
- A Visual Guide to Layout Managers – tutorial explaining the different layouts available
- Swing Tutorial

Overview of today

Introduction (Tuesday) ✅

TicTacToeGraphical (Tuesday) ✅

- Review code for game logic
- Visual components

TicTacToe (Thursday)

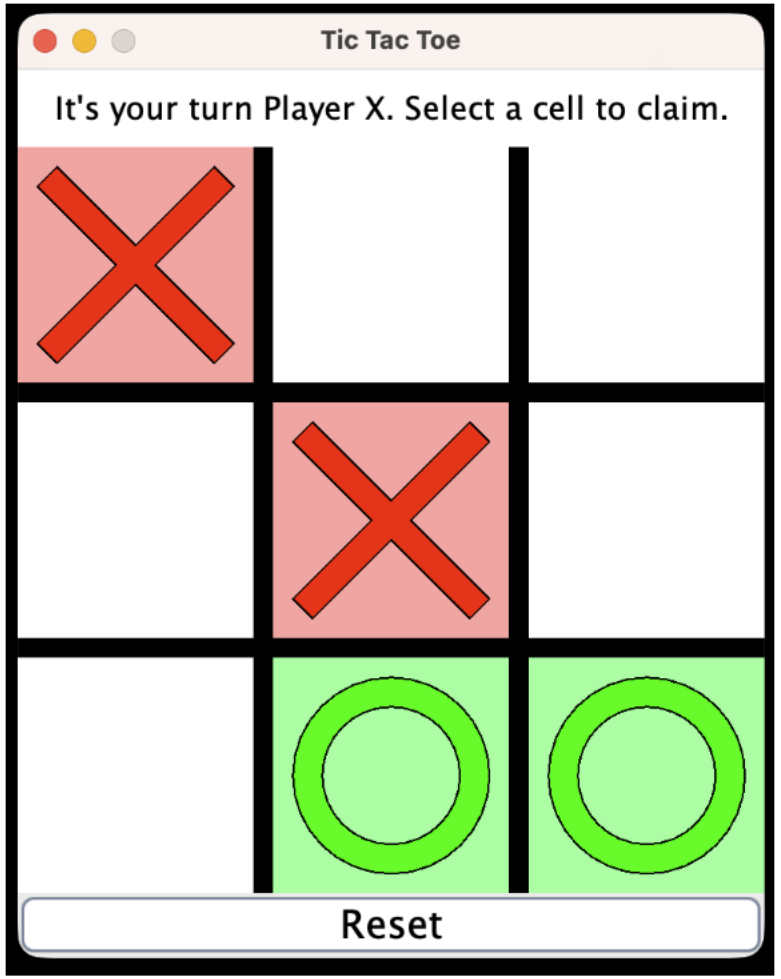
- Event-Driven Programming
- Swing Event Loop
- Implementing Listeners and Callbacks

Tic Tac Toe Application

Ingredients for Graphical Applications

What's in a GUI?

- Accept user input, interaction - Controller
- Reset the game - Controller
- Display images in color - View
- Backend that stores state of game, connected to visual component - Model
- Display text - View



Model View Controller (MVC)

Model (backend)

- objects underlying application state, including methods that access and modify state

View (frontend)

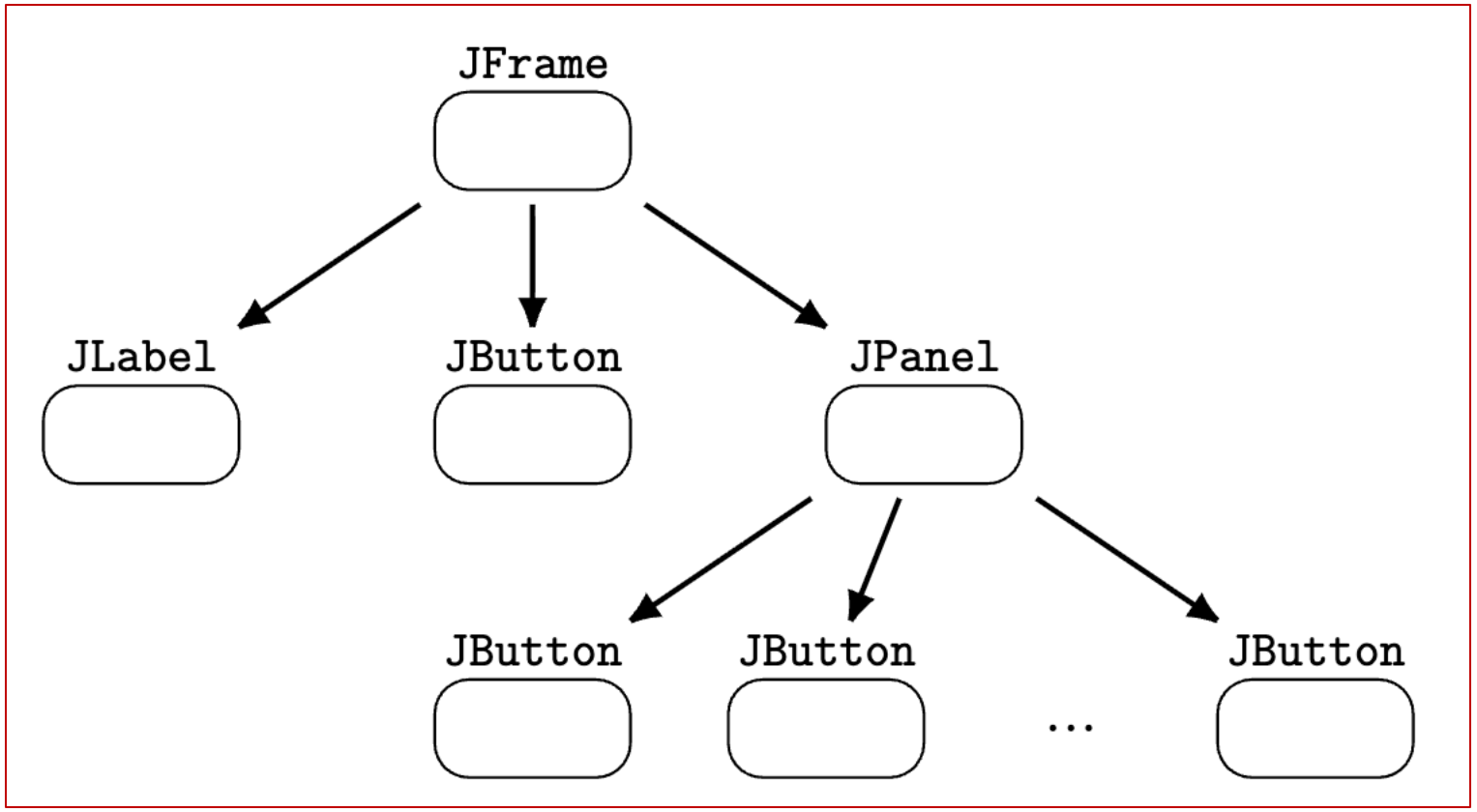
- stuff shown on the screen for the application

Controller

- logic that responds to user inputs (aka events) by updating model and view

The model, view, and controller must connect to each other for the application to work, but the more we can **minimize coupling**, making each component as self-contained as possible, the easier it is to maintain our program.

Hierarchical design



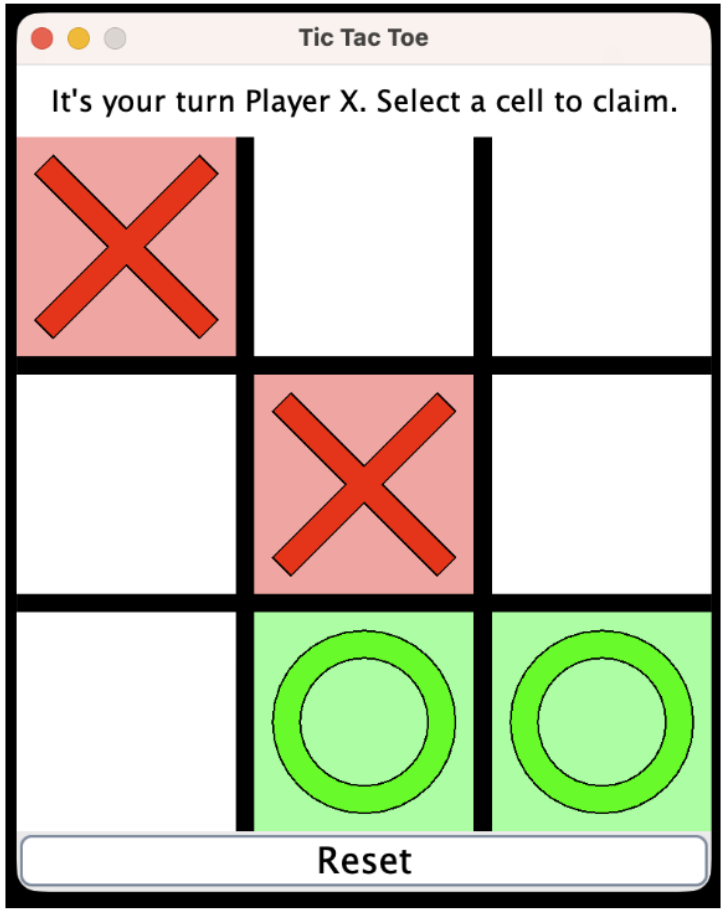
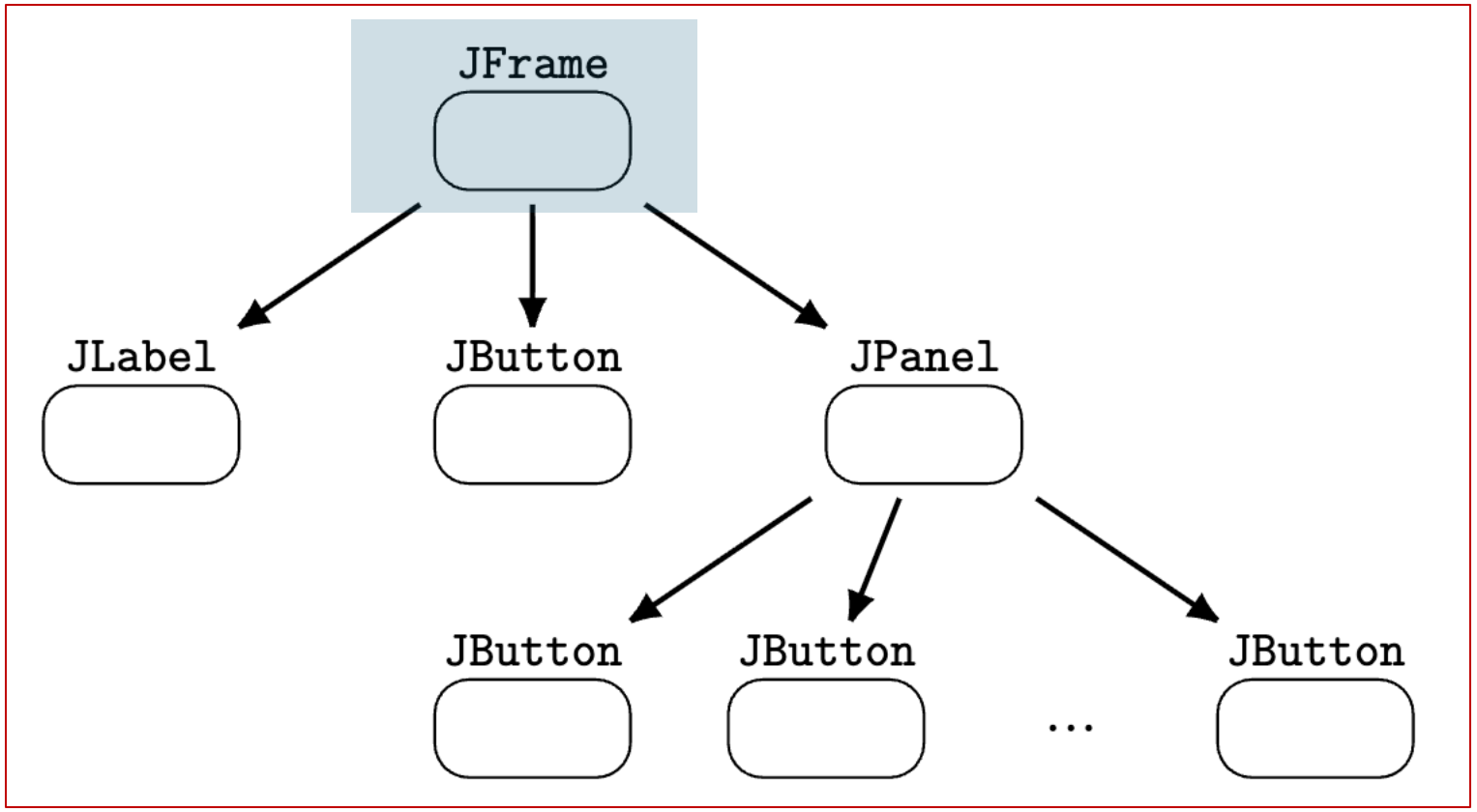


Illustration of how components are hierarchically nested in our TicTacToe application. (This is not an inheritance diagram.)

Hierarchical design



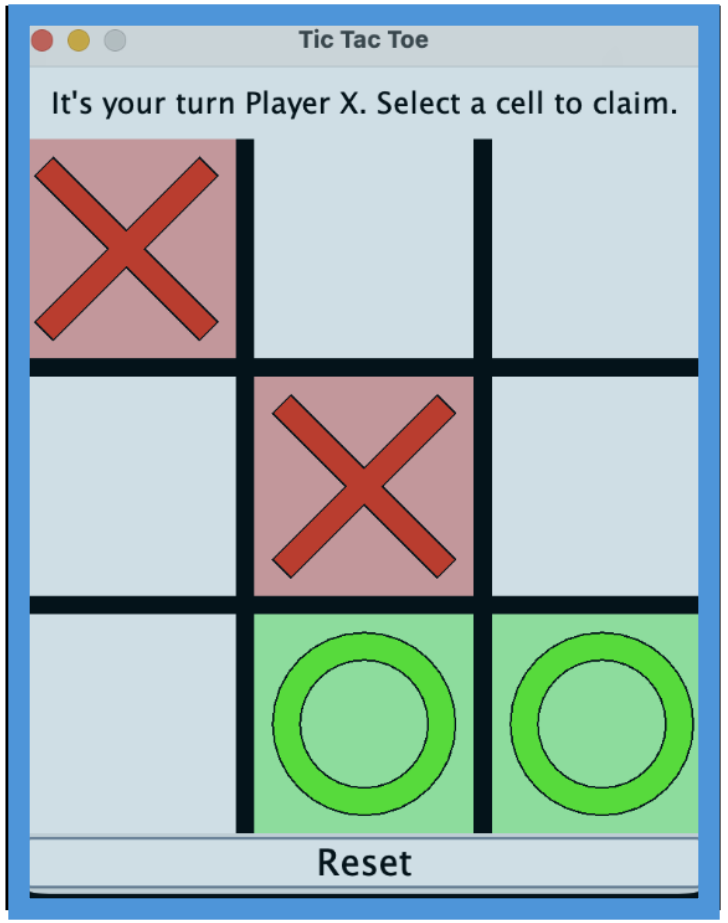
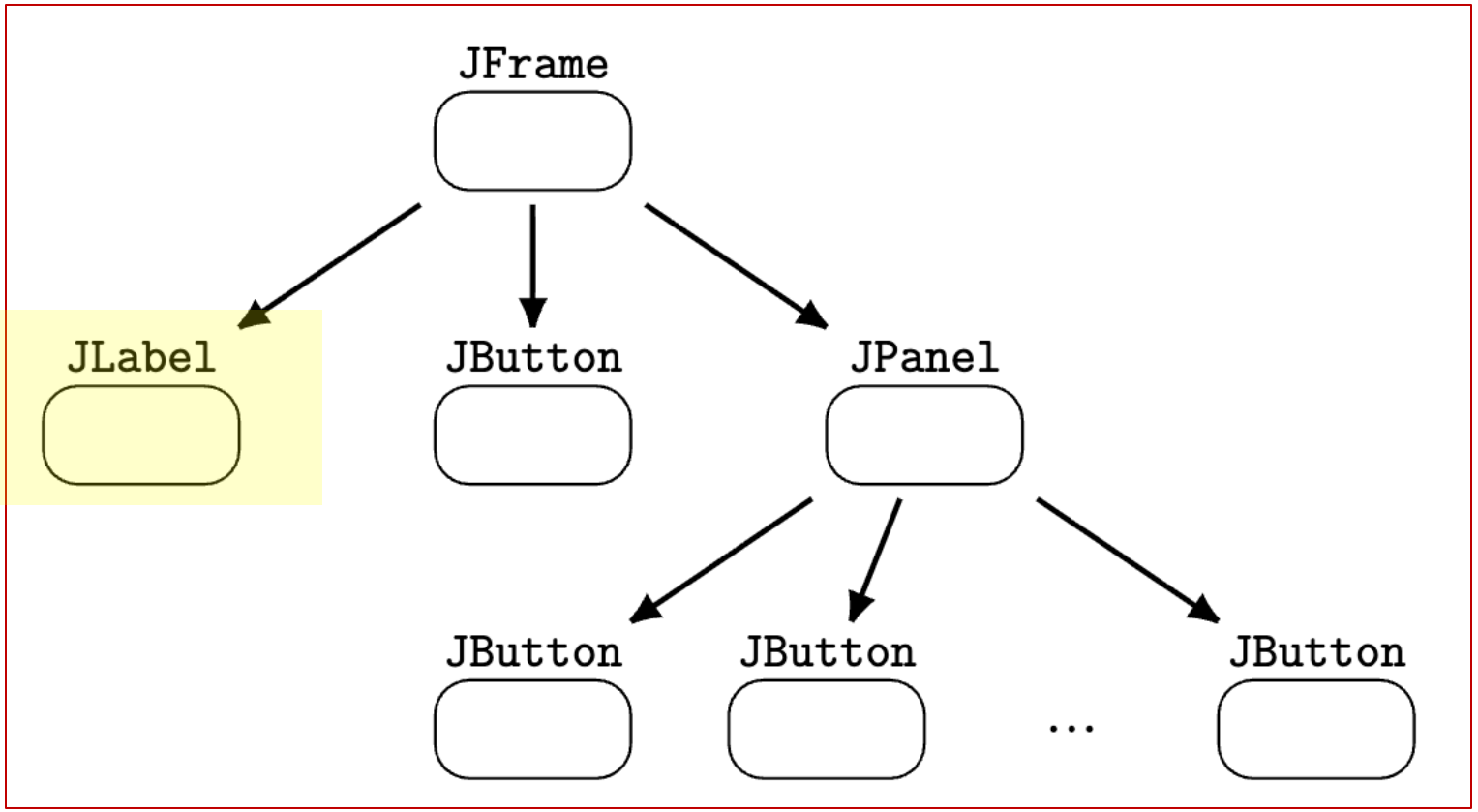


Illustration of how components are hierarchically nested in our TicTacToe application. (This is not an inheritance diagram.)

Hierarchical design



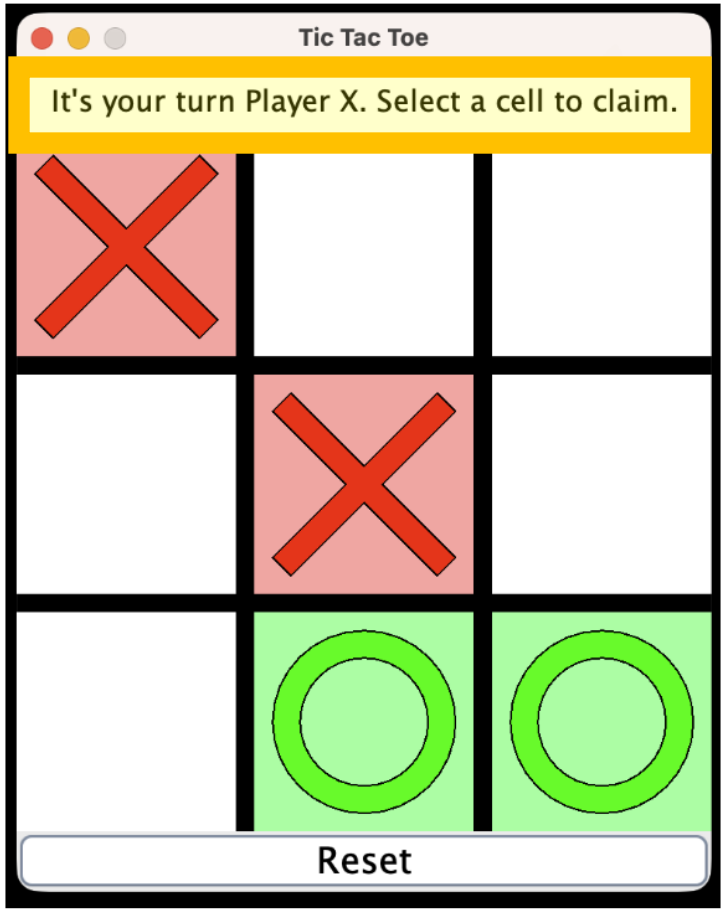
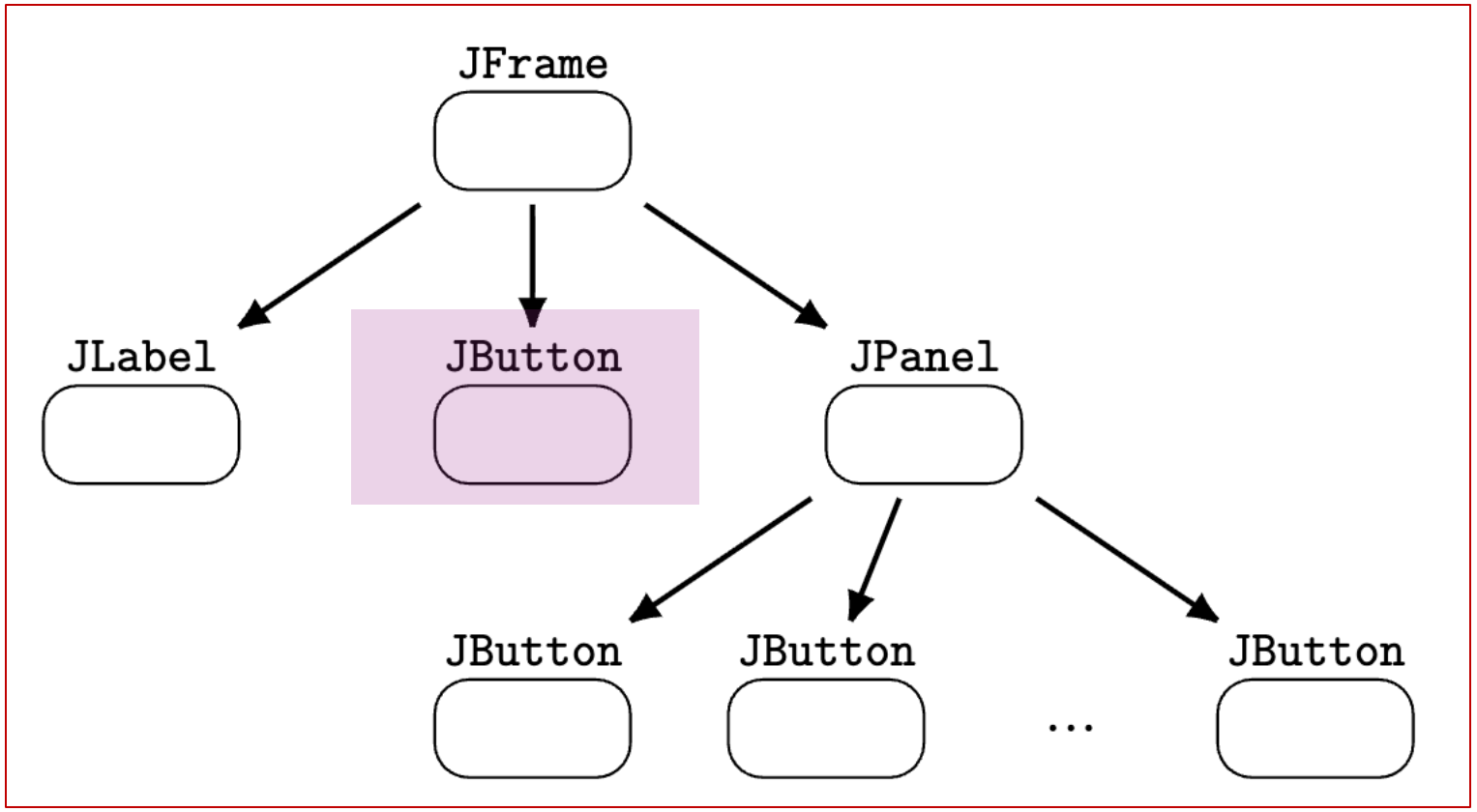


Illustration of how components are hierarchically nested in our TicTacToe application. (This is not an inheritance diagram.)

Hierarchical design



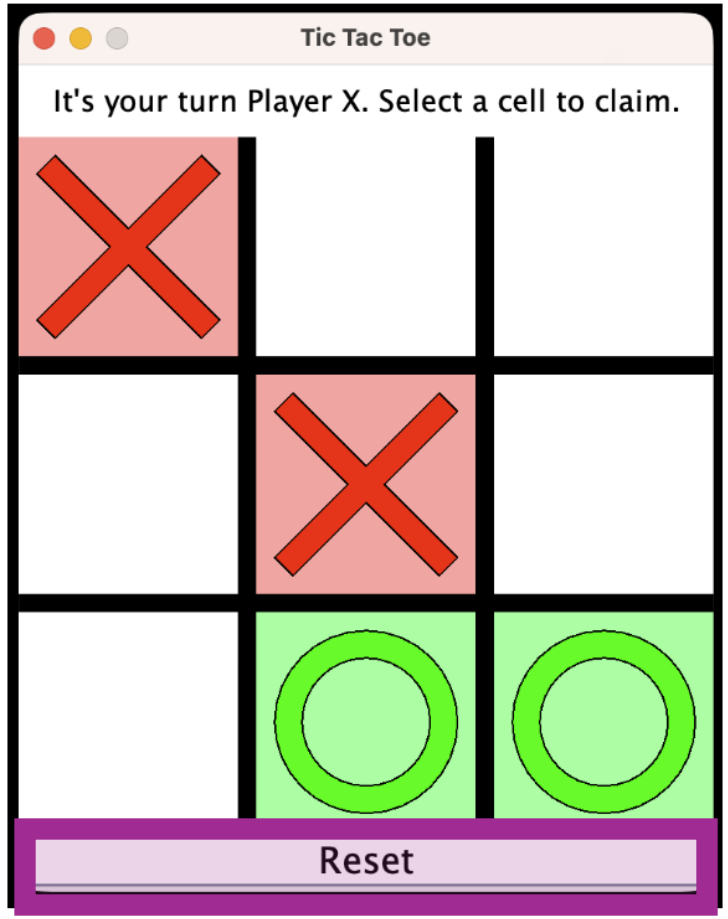
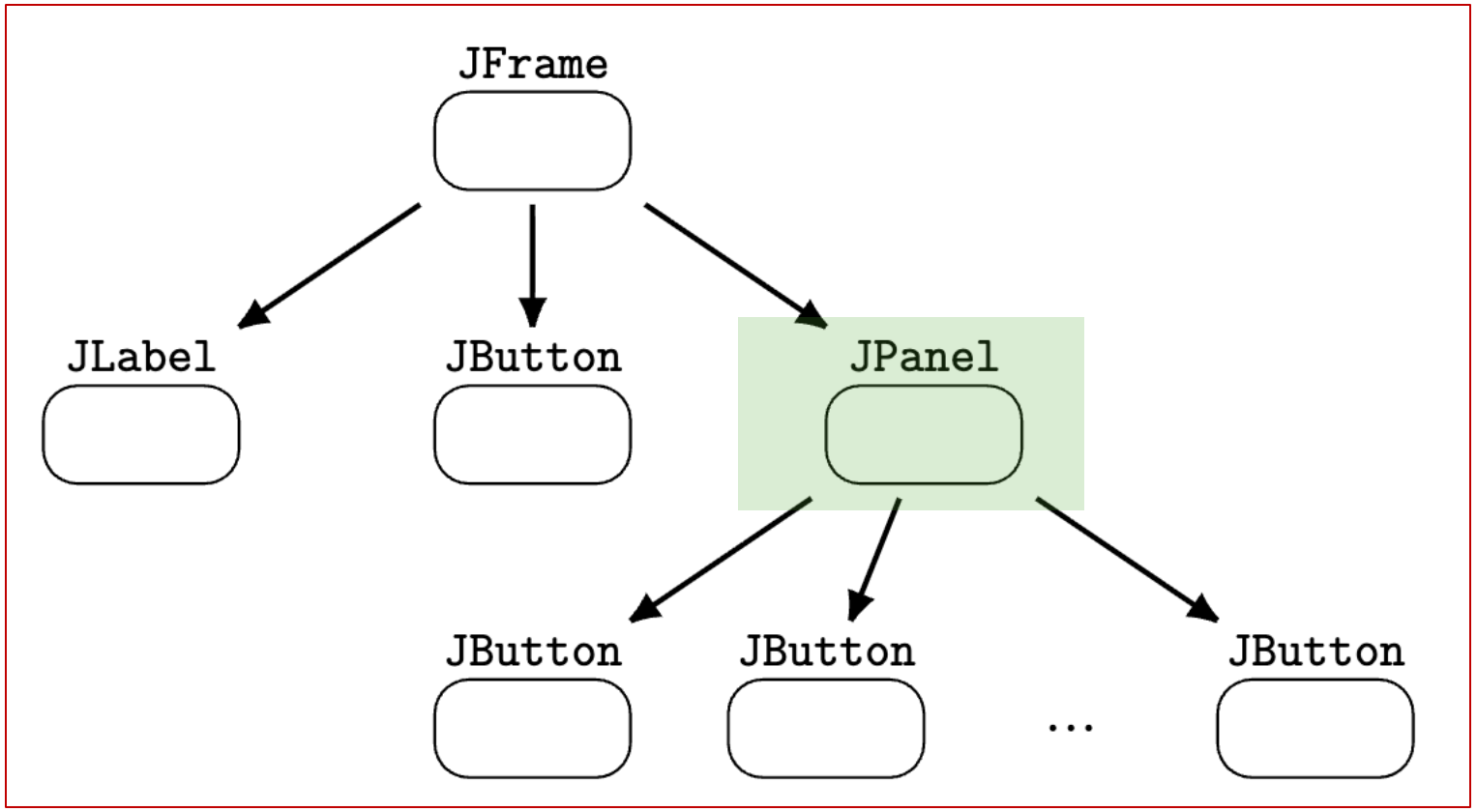


Illustration of how components are hierarchically nested in our TicTacToe application. (This is not an inheritance diagram.)

Hierarchical design



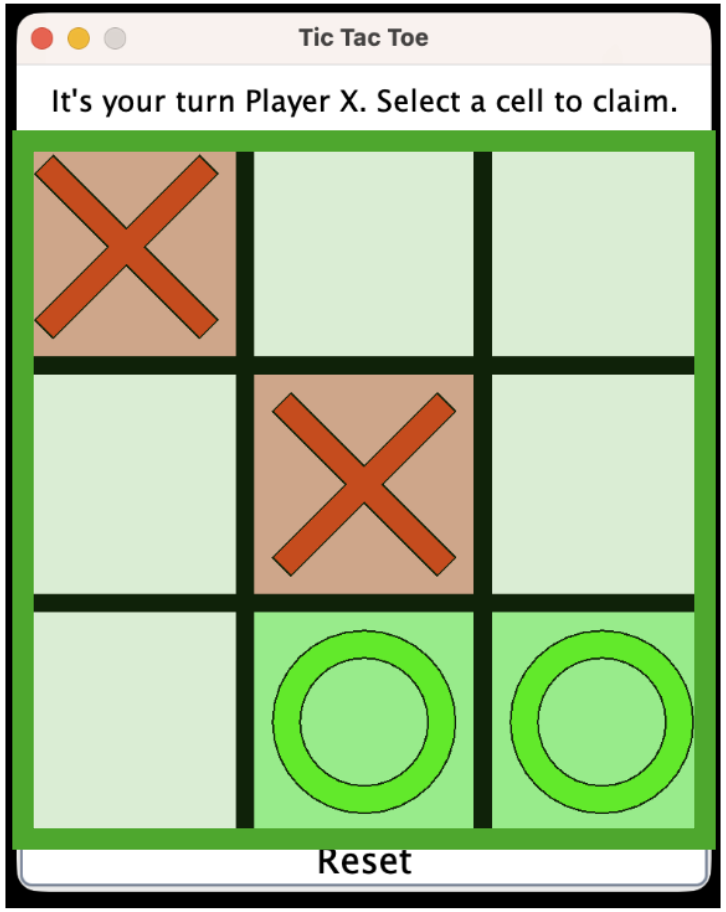
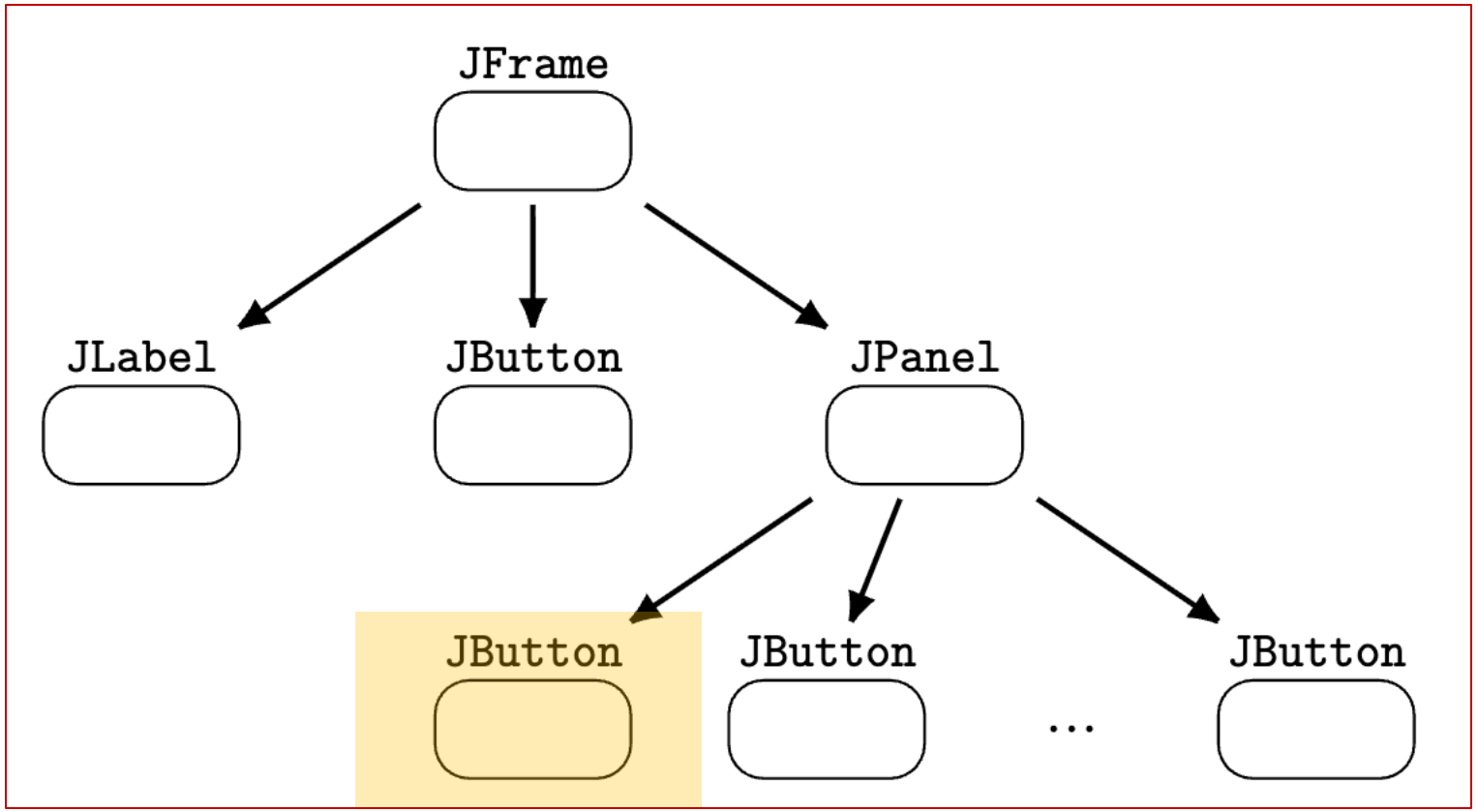


Illustration of how components are hierarchically nested in our TicTacToe application. (This is not an inheritance diagram.)

Hierarchical design



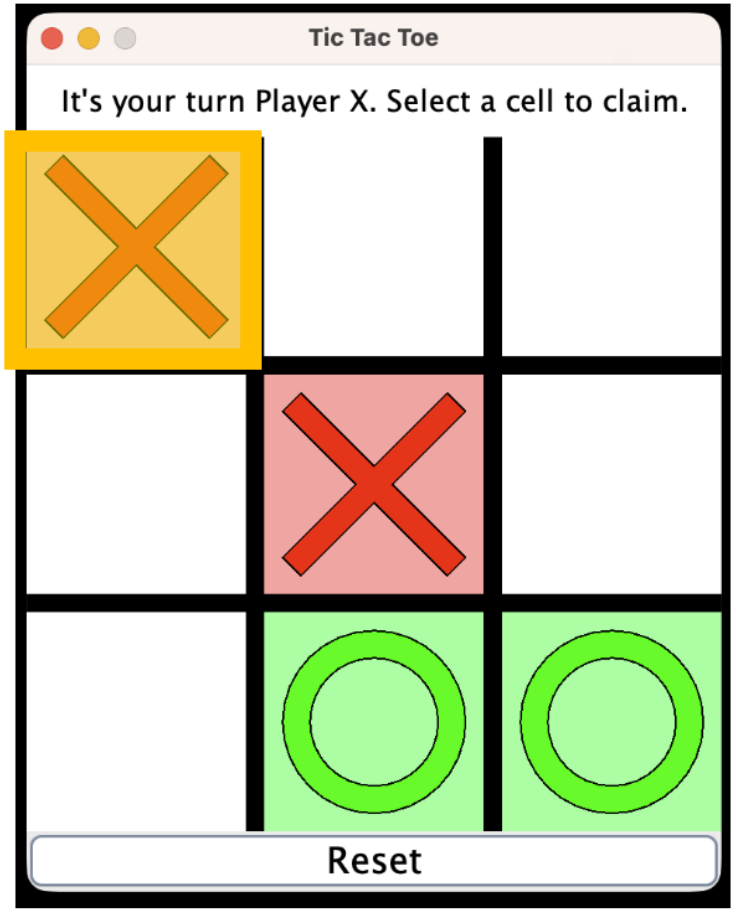
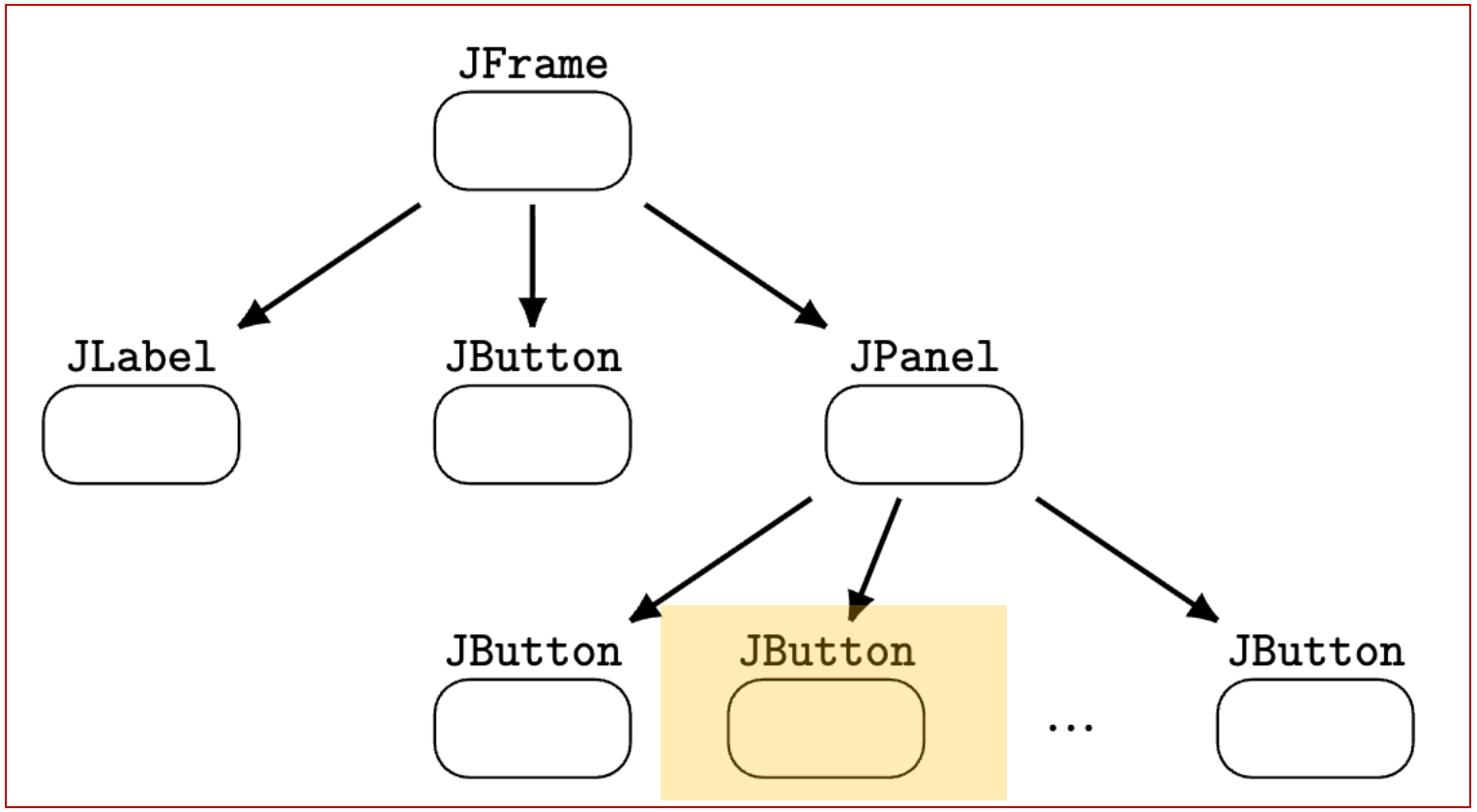


Illustration of how components are hierarchically nested in our TicTacToe application. (This is not an inheritance diagram.)

Hierarchical design



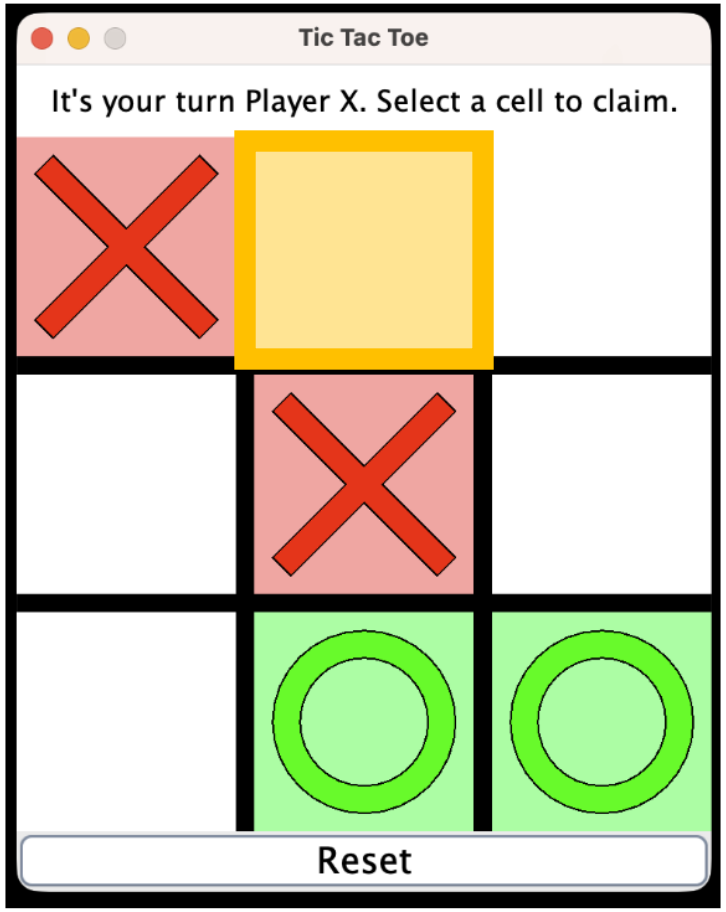
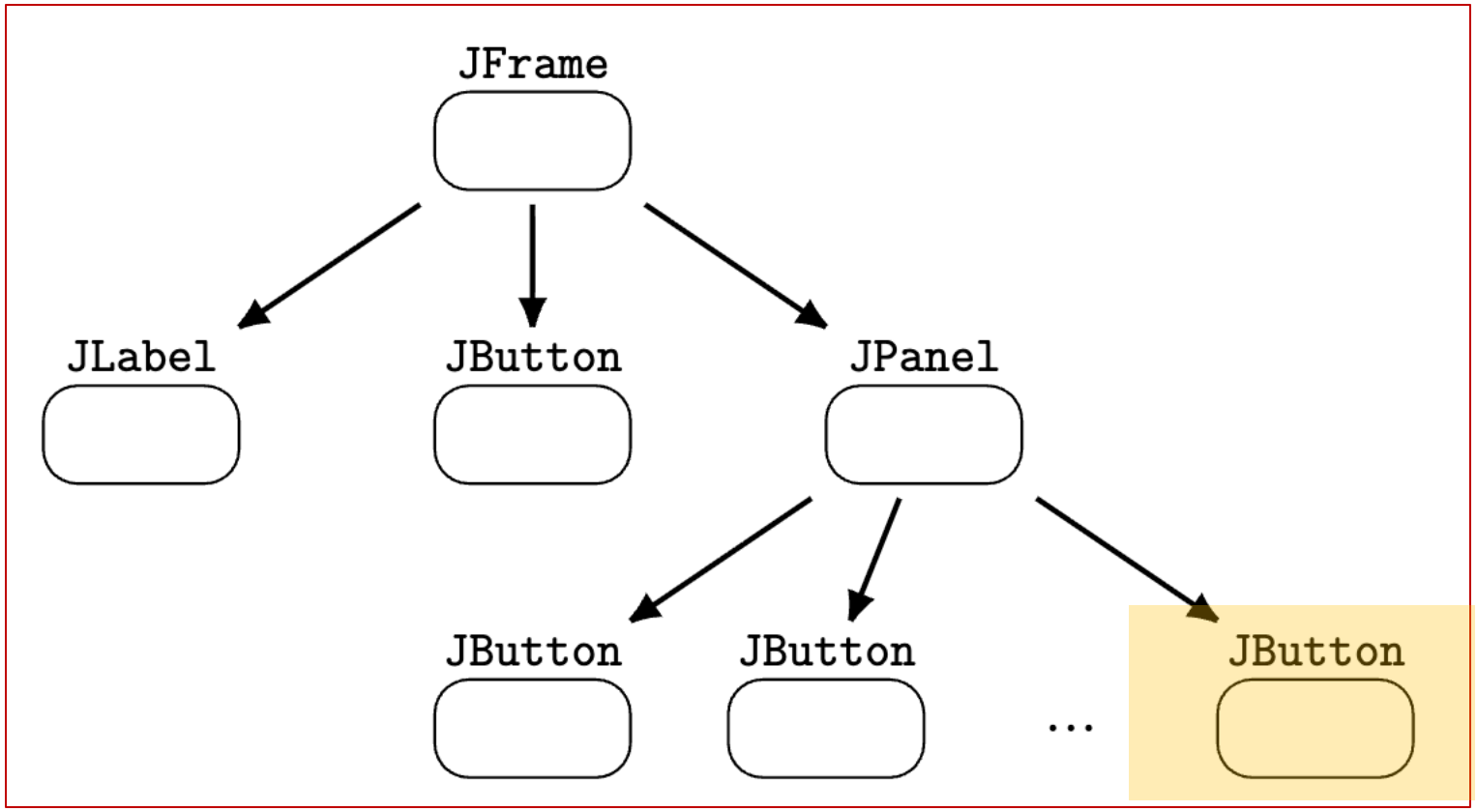


Illustration of how components are hierarchically nested in our TicTacToe application. (This is not an inheritance diagram.)

Hierarchical design



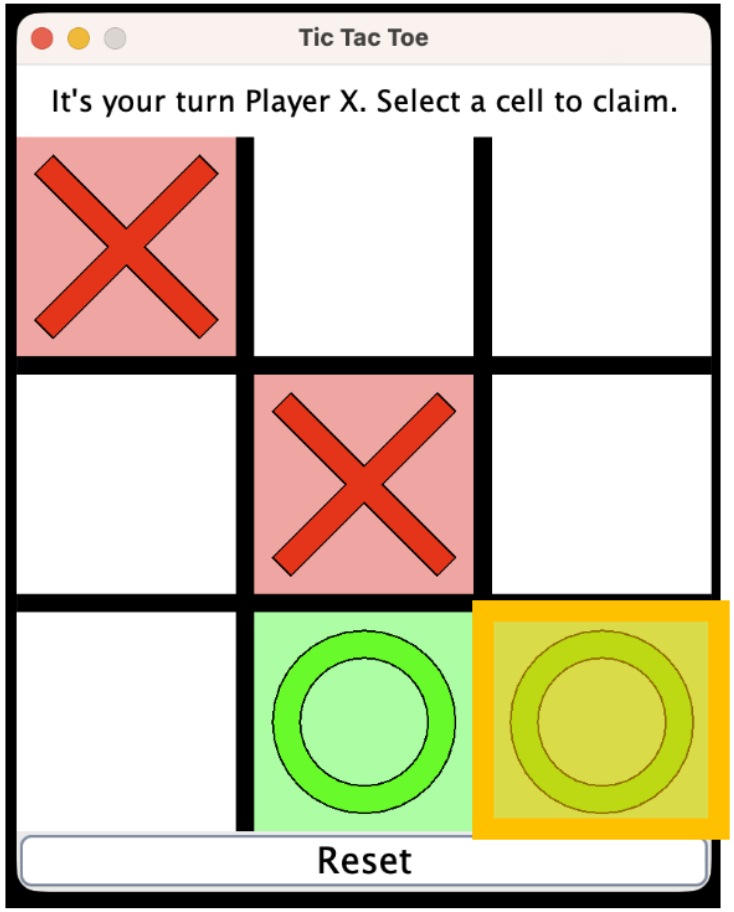


Illustration of how components are hierarchically nested in our TicTacToe application. (This is not an inheritance diagram.)

Event-Driven Programming

Declarative Programming

Imperative Programming

- Most of 2110 up until now
- Implementer specifies which instructions should be executed in which order
- At runtime, instructions are followed to produce the program's output or achieve its desired side-effects

"Do X then do Y then do Z"

Declarative Programming

- Implementer specifies the desired outcome of their code
- At runtime, the way to achieve these outcomes is determined and carried out behind the scenes

"When you do this, here's how it must come out"

Event-Driven Programming

Event

- An occurrence that could necessitate a change in program state

Event-Driven Program (aka Responsive Program)

- Program that executes certain code in response to events

Ingredients for Event-Driven Programming

Events

- An occurrence that could necessitate a change in program state

Callbacks

- Code to update the model and view under certain conditions
- Can be passed around and invoked later

Associations between Events and Callbacks

- Code that records which kinds of events should cause each callback to be called

Event Loop

- Code that detects events and executes the associated callbacks

Events invoked by the user

- activating a view Component, e.g. by clicking it
- keyboard events, e.g. Ctrl+Q
- mouse events, e.g. entering or exiting a certain region of the screen

Events invoked from within the code

- methods called with invokeLater()

Event loop needs to execute very fast or the view will freeze up

- Any callback that takes a "long time" needs to happen concurrently with the execution of the event loop (Concurrency: coming soon!)

Ingredients for Event-Driven Programming

Events

- An occurrence that could necessitate a change in program state

Callbacks

- Code to update the model and view under certain conditions
- Can be passed around and invoked later

Associations between Events and Callbacks

- Code that records which kinds of events should cause each callback to be called

Event Loop

- Code that detects events and executes the associated callbacks

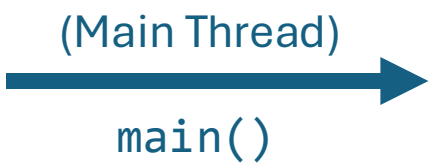
Inversion of control

- custom subroutines allow an external entity (such as a framework's event loop) to control when they are executed

Observer pattern

- Units of code called **observers** register with a central module which notifies them of relevant state changes by calling one of their methods

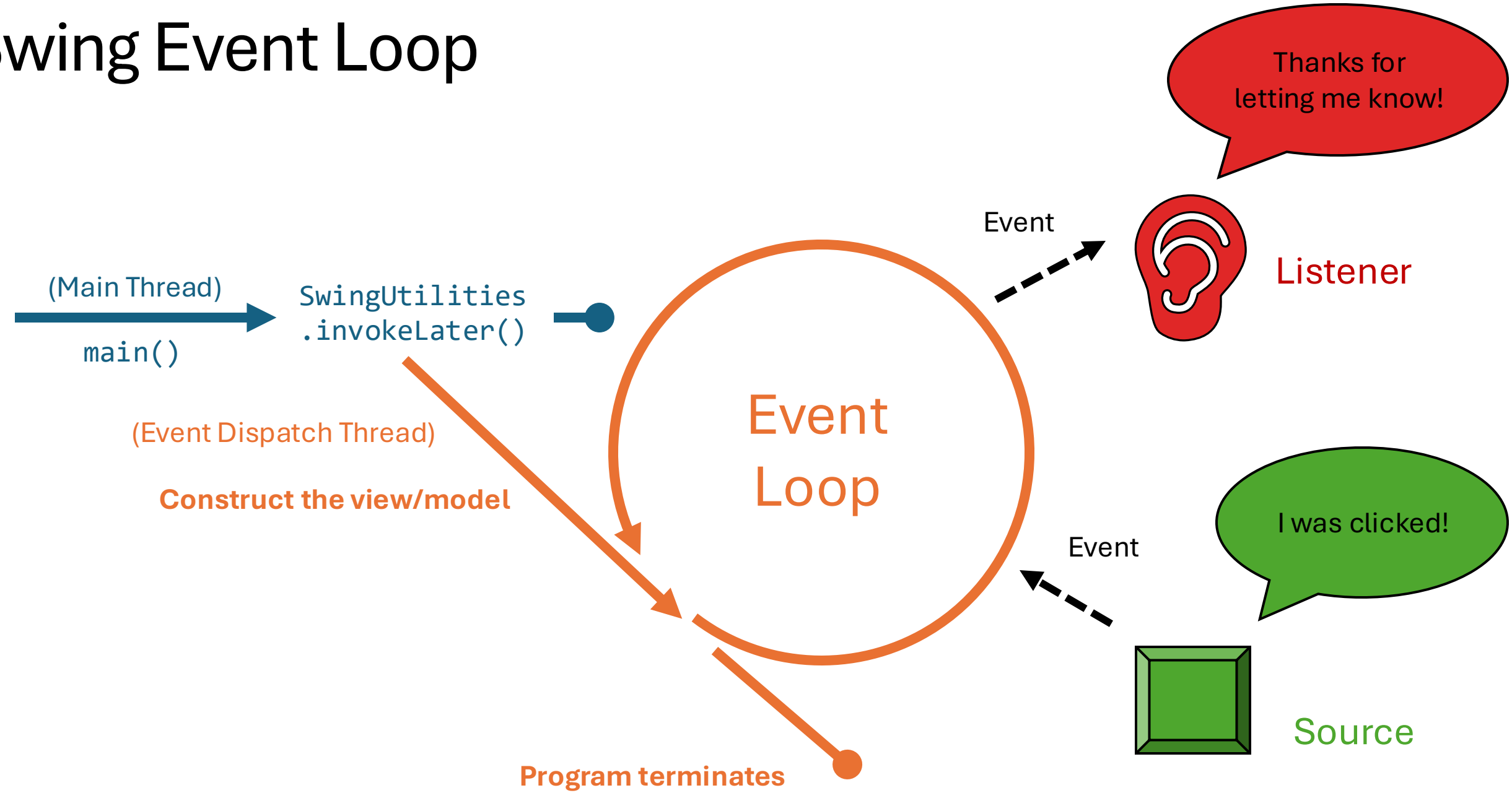
Swing Event Loop



Thread

- an entity that executes a sequence of instructions
- can operate concurrently with other threads (in parallel / at the same time as)
- more on this next week!

Swing Event Loop



Ingredients for Event-Driven Programming

Events

- An occurrence that could necessitate a change in program state

Callbacks

- Code to update the model and view under certain conditions
- Can be passed around and invoked later

Associations between Events and Callbacks

- Code that records which kinds of events should cause each callback to be called

Event Loop ✅

- Code that detects events and executes the associated callbacks

Events, Listeners, and Components

Event and source	Example Listener Type	Callbacks to Implement	Type that can have this kind of listener added to it
KeyEvent from keyboard	KeyListener	void keyPressed(KeyEvent e) void keyReleased(KeyEvent e) void keyTyped(KeyEvent e)	Component
MouseEvent from mouse or touchpad	MouseMotionListener	void mouseDragged(MouseEvent e) void mouseMoved(MouseEvent e)	Component
ActionEvent from Button	ActionListener	void actionPerformed(ActionEvent e)	JButton

Creating Listeners and Callbacks

Demo: Reset Listener

Variable Capture -- An anonymous class or lambda expression can *capture* a variable declared within a surrounding scope, making the value of that variable accessible within

Ingredients for Event-Driven Programming

Events ✅

- An occurrence that could necessitate a change in program state

Callbacks ✅

- Code to update the model and view under certain conditions
- Can be passed around and invoked later

Associations between Events and Callbacks ✅

- Code that records which kinds of events should cause each callback to be called

Event Loop ✅

- Code that detects events and executes the associated callbacks

Poll: Action Listeners

Which line associates an event with a callback?

A) 10
B) 2
C) 6
D) None of the above

PollEv.com/leahp
text **leahp** to **22333**

```
1 | class App extends JFrame
2 |             implements ActionListener {
2 |
3 |   public App() {
4 |     JButton button = new JButton("B");
5 |     add(button);
6 |     button.addActionListener(this);
7 |   }
8 |
9 |   @Override
10|   public void actionPerformed(ActionEvent e) {
11|     print("Got " + e.getActionCommand()
12|       + " from " + e.getSource());
13|   }
14|}
```

Poll: Action Listeners

Which line associates an event with a callback?

A) 10

B) 2

C) 6

D) None of the above

As a result of line 10, any `ActionEvent` associated with `button` will cause `App.actionPerformed()` to get called with the event passed in as its argument.

```
1 | class App extends JFrame
2 |           implements ActionListener {
2 |
3 |   public App() {
4 |     JButton button = new JButton("B");
5 |     add(button);
6 |     button.addActionListener(this);
7 |   }
8 |
9 |   @Override
10|   public void actionPerformed(ActionEvent e) {
11|     print("Got " + e.getActionCommand()
12|       + " from " + e.getSource());
13|   }
14|}
```

Cell Action Listener

Code Demo

Whose turn is it?

Problem: the turnLabel needs to know when the turn changes

Solution: Property Changes

Exercise

How could we make the available cells turn light gray when the mouse is over them?

- which events to listen for?
- how to set up the listeners?
- any additional state?
- how to turn a component gray?

Use the documentation to figure it out!

Ingredients

- MouseListener: implement mouseEntered() and mouseExited()
- hover field of cell, initialize to false, create setter
- create Color field HOVER_COLOR
- cell.paintComponent() -- check for hover to determine background color
- when adding symbol to cell, set hover to false before repainting

Metacognition

- Take 1 minute to write down a brief summary of what you have learned today

Thanks and have a great day!