

Gots ta move dat data

... and not trash your threads...

Lecture 26, Nov 27

Administrivia

- Reminder:
 - Send me checkpoint materials with/soon after your milestones
 - Current code base/tests
 - Current design docs
 - Current documentation
 - Presentation materials

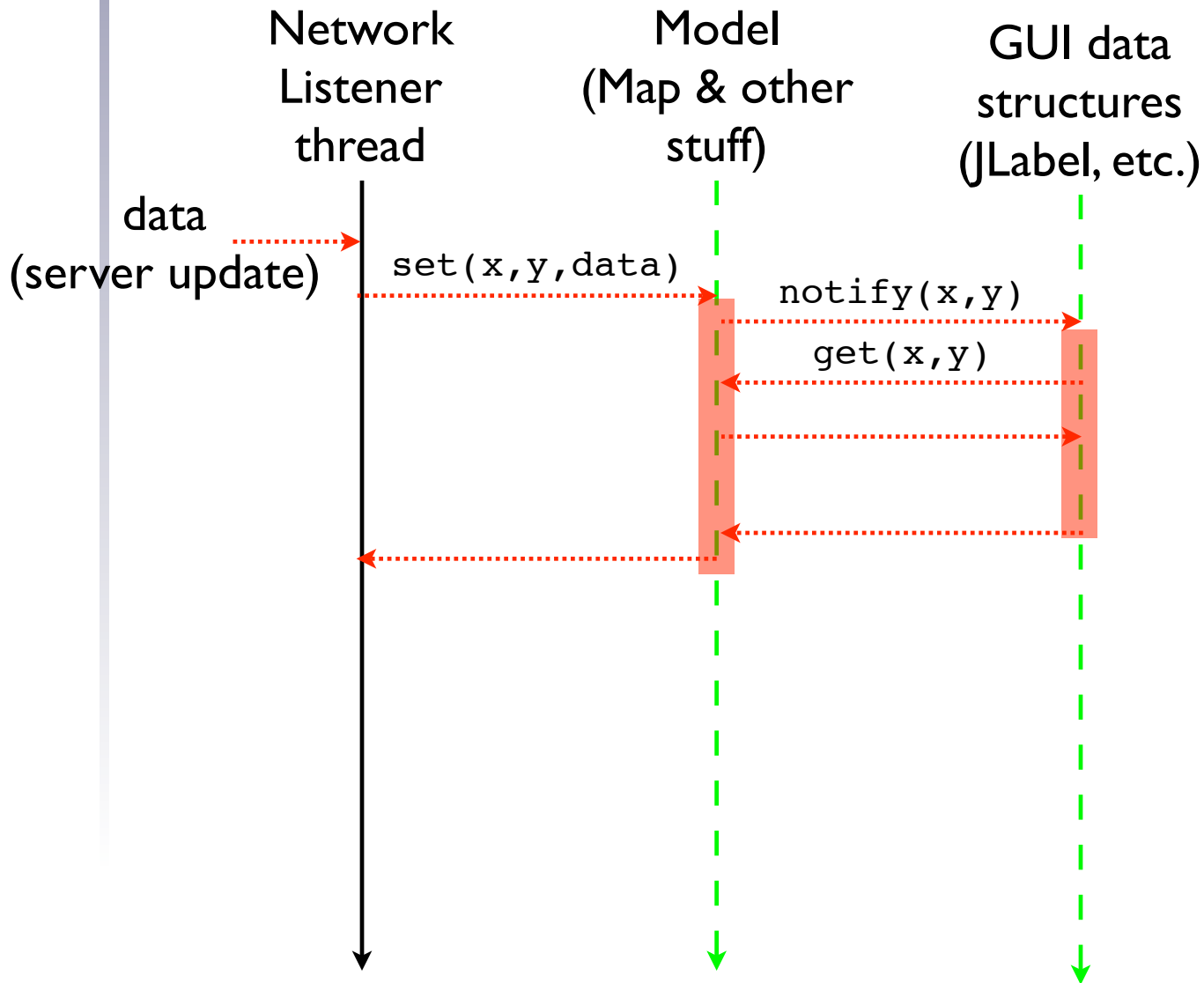
Time keeps on rollin'

- Last time:
 - Thanksgiving!
 - Mmmmm... Tryptophan... [zzzzzz]
- Before that:
 - Null Terminators P3M3 delivery
- Today:
 - Thud Muffins P3M3 delivery
 - Communicating with the Swing thread (maybe)

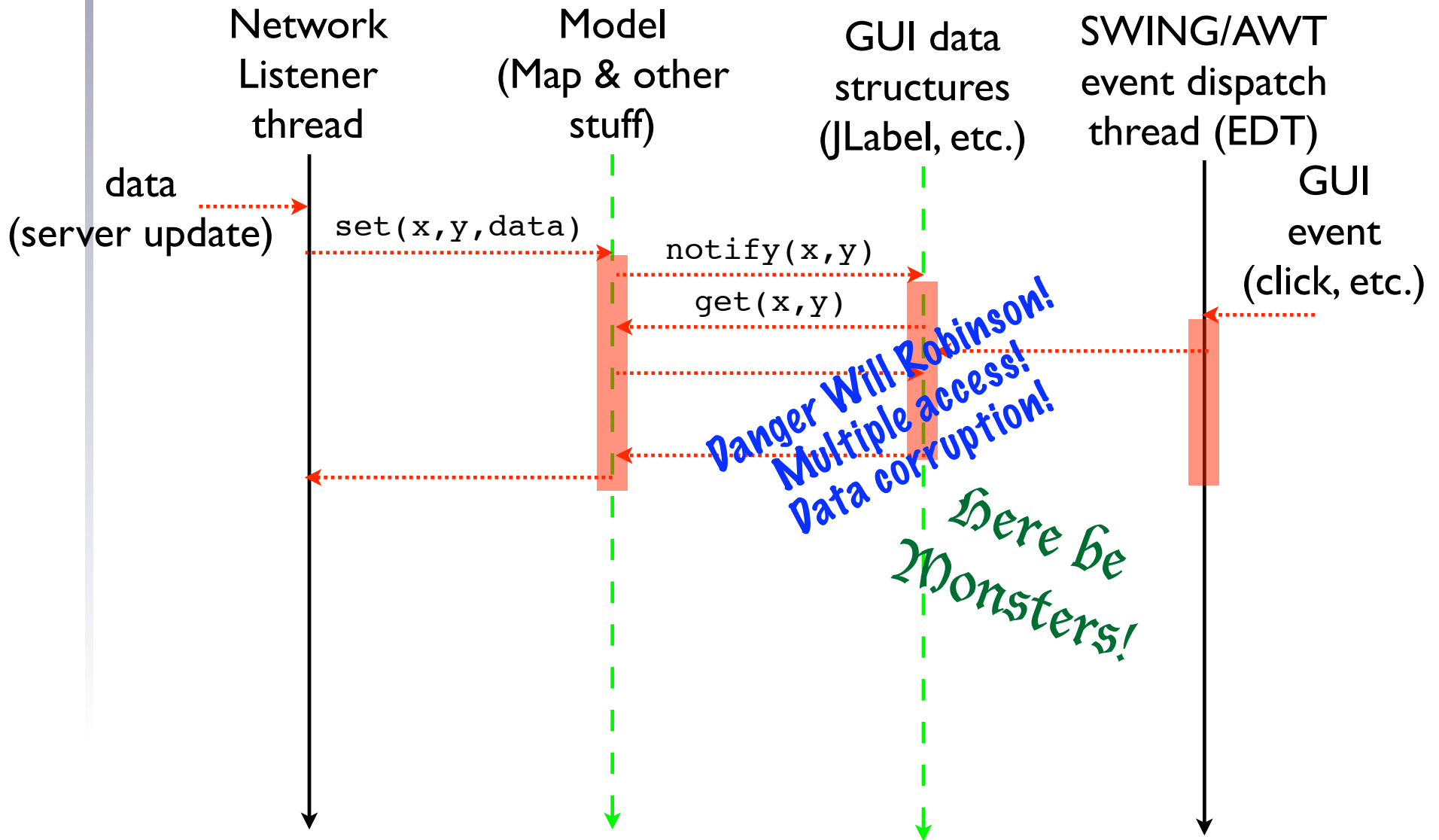
Client-side communications

- Issue:
 - Data enters client through network listener thread
 - GUI being processed on Swing/AWT event queue
 - “Event Dispatch Thread” (EDT)
 - Need to transfer data between them
 - Need to handle synchronization...

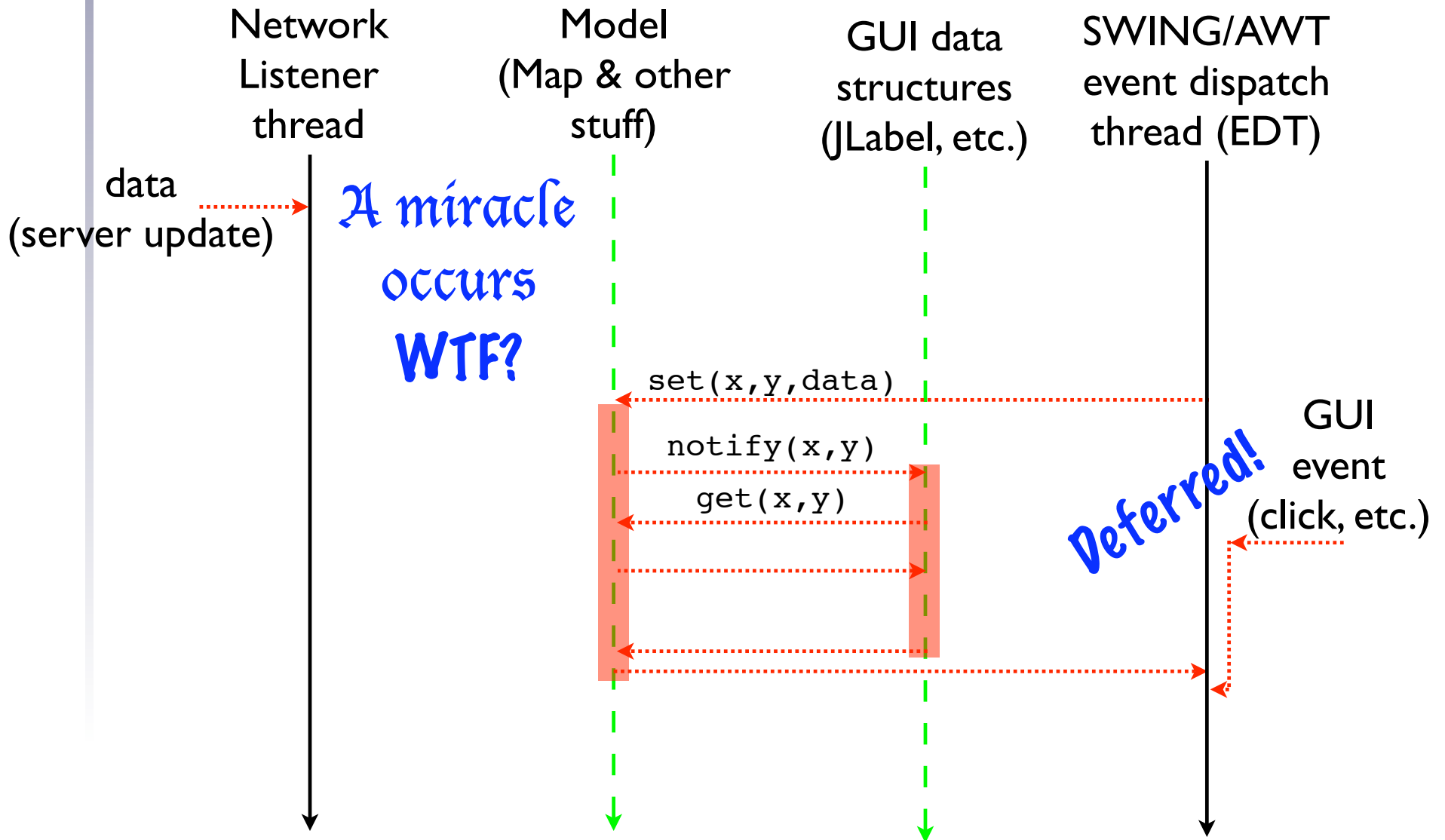
Classic MVC design



Classic MVC design



What you want



How to get there...

- Need some way for listener thread to:
 - Store the incoming data temporarily
 - Notify the EDT: “Hey! There’s some new data! Come deal with it!”
- Requires:
 - Synchronized access to temp data store
 - Rapid turnaround in listener thread

How to get there...

- Clever, clever SWING designers thought of this...
- `javax.swing.SwingUtilities.invokeLater()`
 - Takes a `Runnable`
 - EDT executes `Runnable.run()` “when it’s convenient”
 - After rest of outstanding AWT events have cleared

How to get there...

- Clever, clever SWING designers thought of this...
- `javax.swing.SwingUtilities.invokeLater()`
- Immediately returns control to calling thread (network listener)
- Executes `Runnable.run()` *once*
- Does *not* create a new thread

Network listener code

```
public void listenToNetwork(Socket s) {  
    while (!toStop) {  
        Message data= // read from network  
        synchronizedBuffer.add(data);  
        SwingUtilities.invokeLater(new _msgHandler());  
    }  
}
```

Network listener code

```
public void listenToNetwork(Socket s) {
    while (!toStop) {
        Message data= // read from network
        synchronizedBuffer.add(data);
        SwingUtilities.invokeLater(new _msgHandler());
    }
}

private static class _msgHandler
implements Runnable {
    public void run() {
        Message m=synchronizedBuffer.remove();
        while (m!=null) {
            m.execute(model);
            m=synchronizedBuffer.remove();
        }
    }
} }
```

Alternately...

```
public void listenToNetwork(Socket s) {
    while (!toStop) {
        Message data= // read from network
        SwingUtilities.invokeLater(new _msgHandler(data));
    }
}

private static class _msgHandler
    implements Runnable {
    public _msgHandler(Message m) { _data=m; }
    public void run() {
        m.execute(model);
    }
    private final Message _data;
}
}
```