Programming Languages and Techniques (CIS120)

Lecture 37

Swing IV: Paint Revisited

Chapter 31

Announcements

- HW9: Game – Due Monday, December 9th at 11:59pm

- Final Exam:
  - Tuesday, December 17th 6:00-8:00PM
  - Room assignments TBA
  - Coverage: comprehensive, but emphasizing material since Midterm 2
    - Dynamic dispatch, Exceptions, IO, OO concepts
  - See example exams

- Makeup exam offered Weds. December 18th
  - required by registrar for exam conflicts
  - see Piazza (soon) for details

How far along are you on HW09: Make Your Own Game?

I haven't started yet

I have the basic design implemented

I'm about halfway done, I think

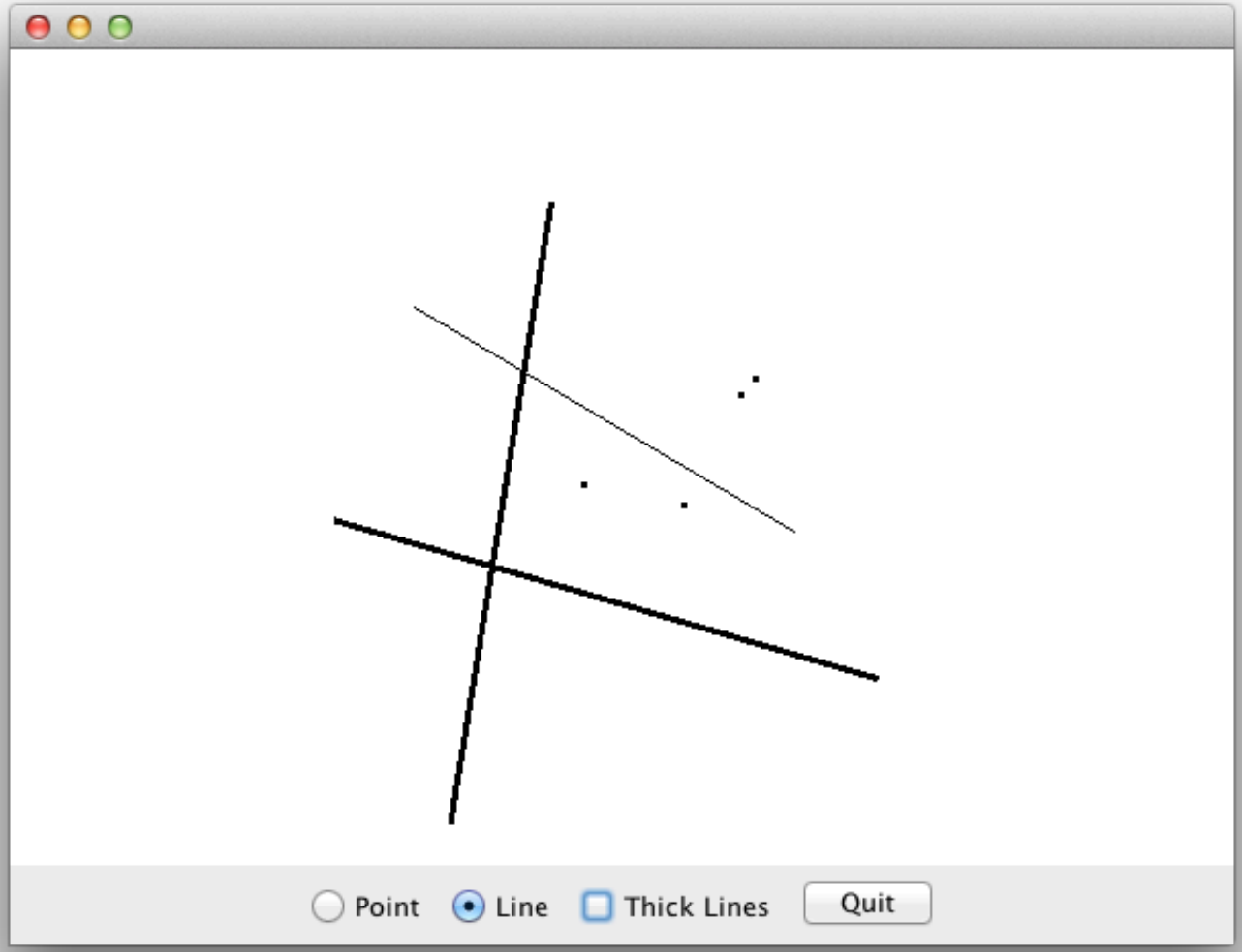
I'm nearly finished

I'm done!

Paint Revisited

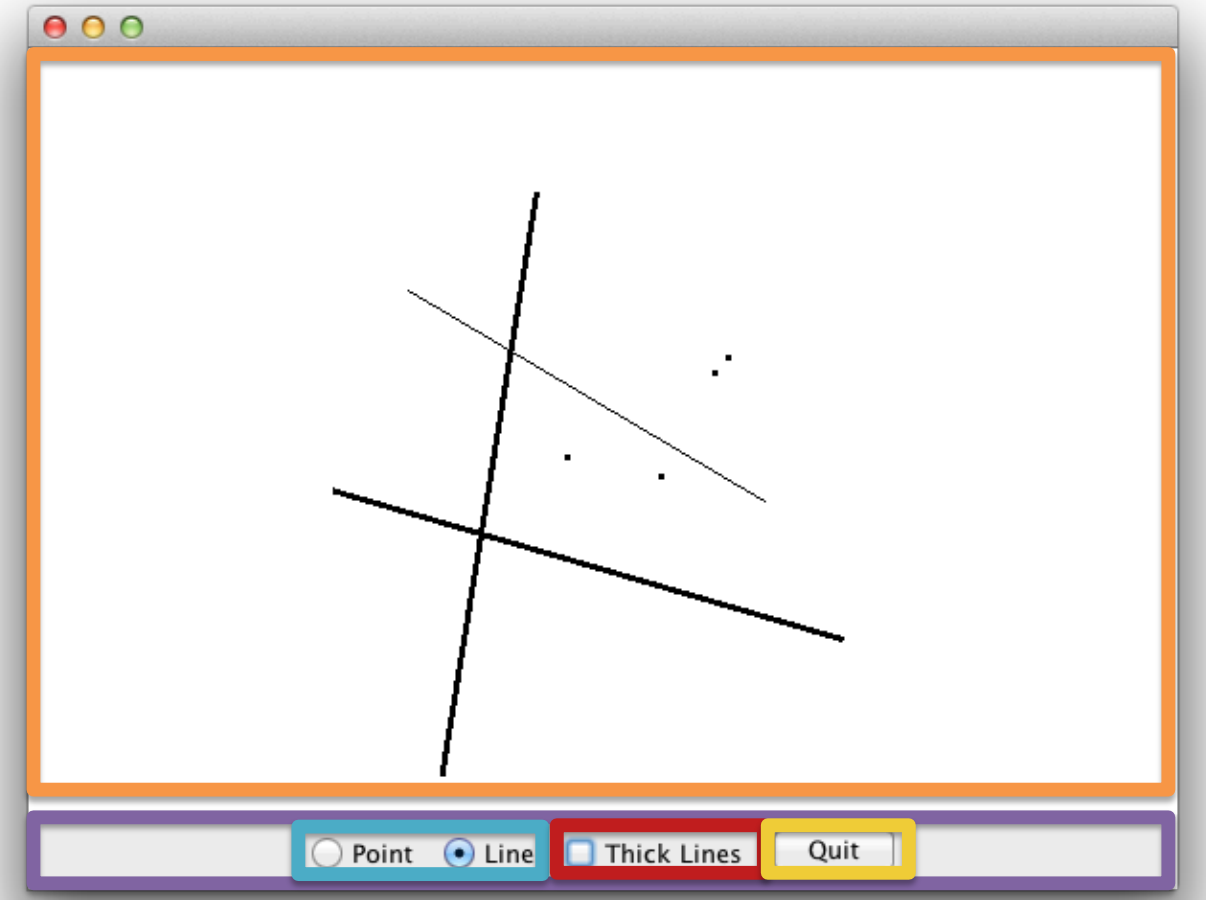
Using Anonymous Inner Classes

Refactoring for OO Design



What layout would you use for this app? What components would you use?

Canvas subclass of JPanel (canvas)



JPanel (toolbar)

JRadioButton (point, line)

JCheckbox (thick)

JButton (quit)

Paint Revisited
(thoroughly discussed in Chap 31)

Using Anonymous Inner Classes

Refactoring for OO Design

(See PaintA.java … PaintE.java)

What OO feature can we use to eliminate the use of Enum/switch from Paint.java?

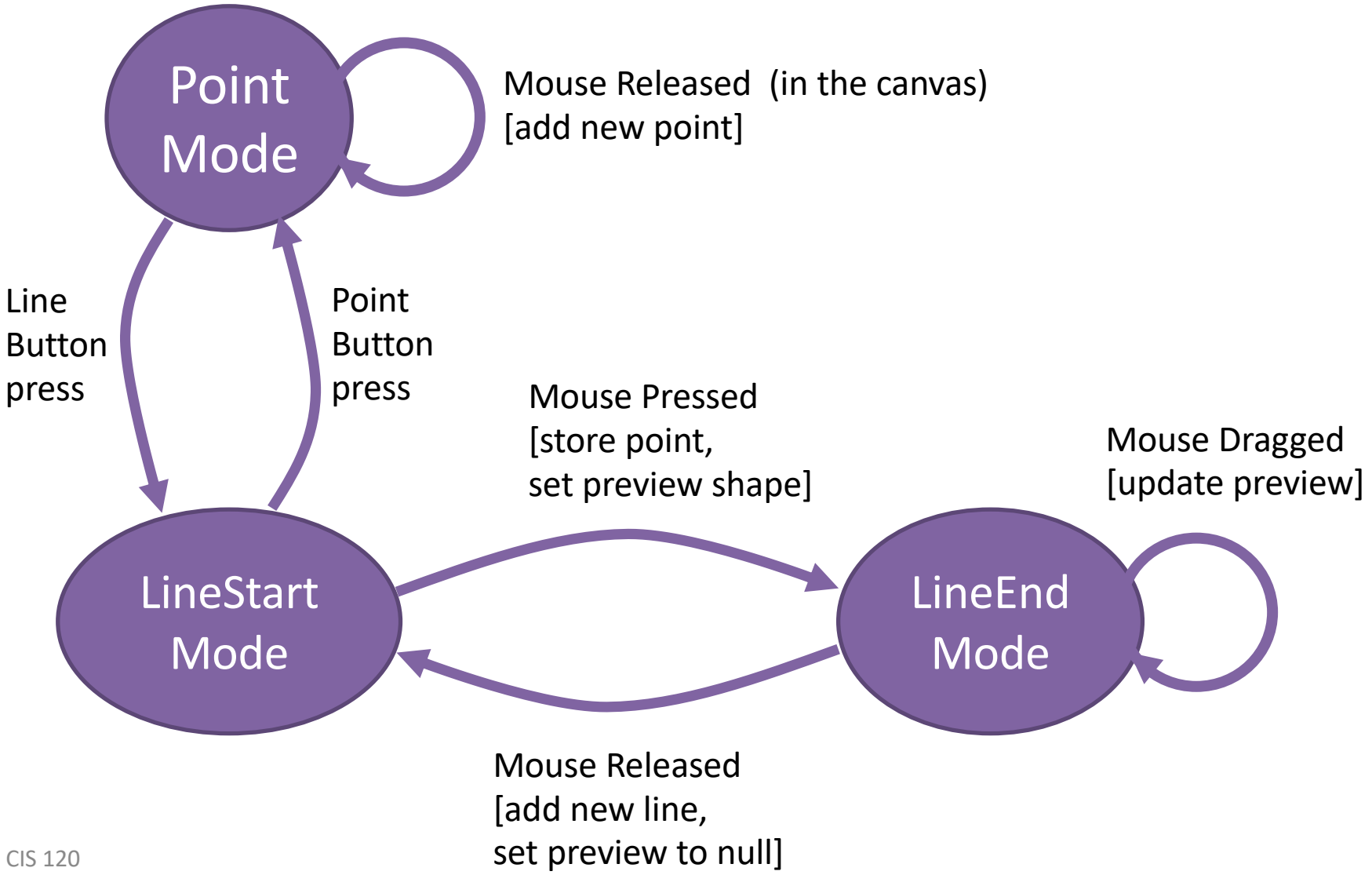
Generics

Inheritance

Dynamic Dispatch

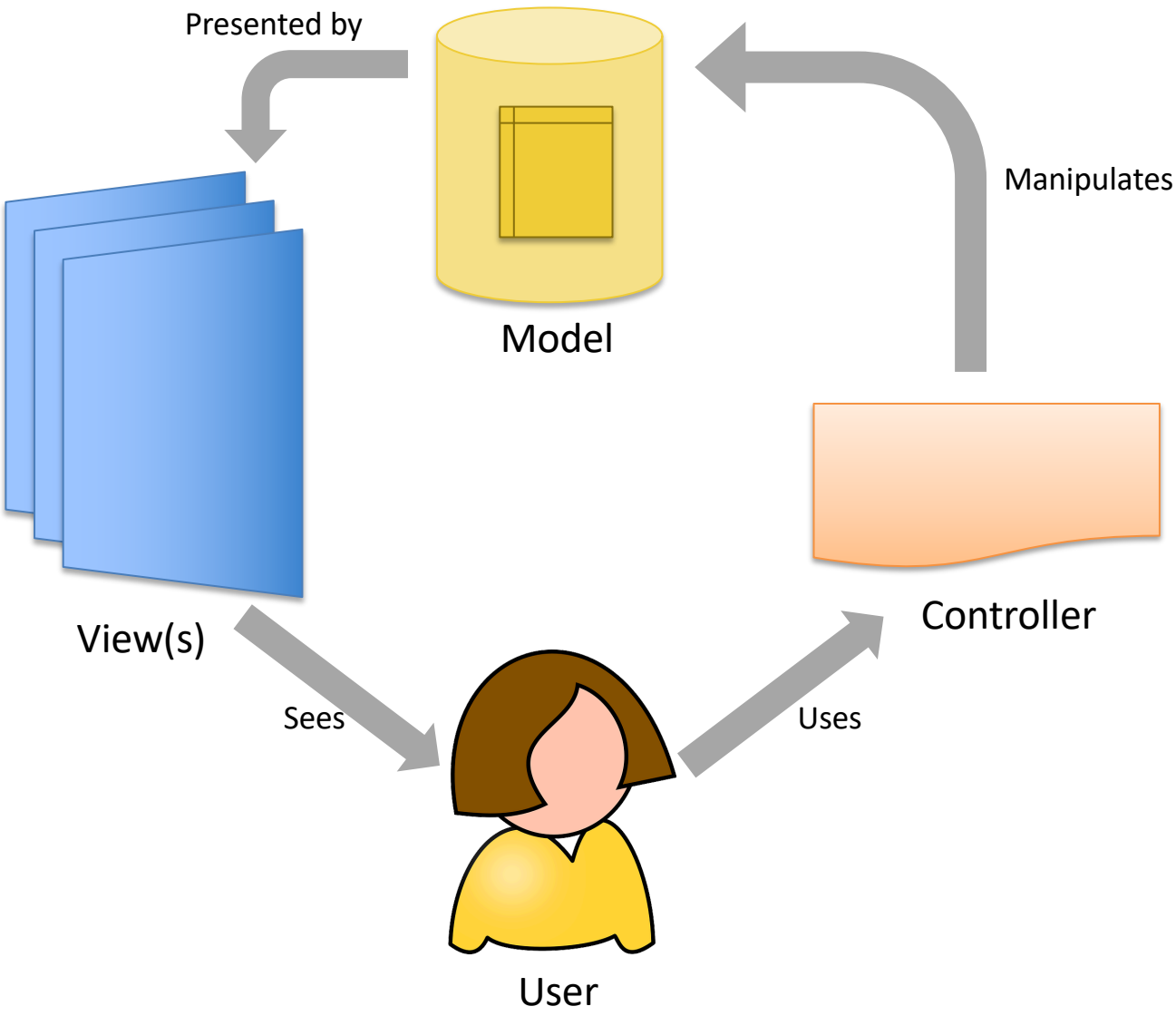
Anonymous Inner Classes

Mouse Interaction in Paint

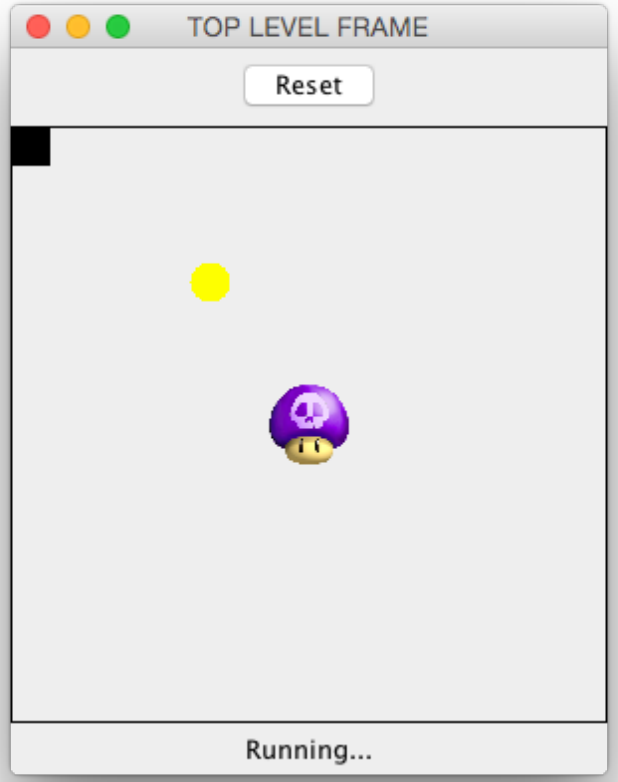


Model View Controller Design Pattern

MVC Pattern


Presented by
Model
Manipulates
View(s)
Controller
Sees
Uses
User

Example 1: Mushroom of Doom



Example: MOD Program Structure

- GameCourt, GameObj + subclass local state
  - object location & velocity
  - status of the game (playing, win, loss)
  - how the objects interact with eachother (tick)

Model

- Draw methods
  - paintComponent in GameCourt
  - draw methods in GameObj subclasses
  - status label

View

- Game / GameCourt
  - Reset button (updates model)
  - Keyboard control (updates square velocity)

Controller

Example: Paint Program Structure

- Main frame for application (class Paint)
  - List of shapes to draw
  - The current color
  - The current line thickness

Model

- Drawing panel (class Canvas, inner class of Paint)

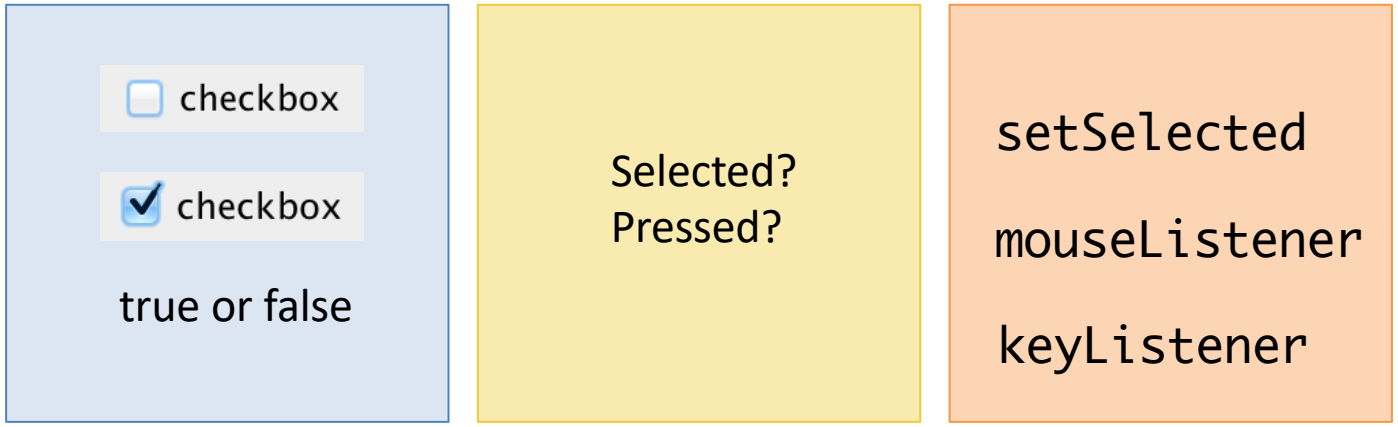
View

- Control panel (class JPanel)
  - Contains radio buttons for selecting shape to draw
  - Line thickness checkbox, undo and quit buttons

Controller

- Connections between Preview shape (if any…)
  - Preview Shape: View <-> Controller
  - MouseAdapter: Controller <-> Model

Example: CheckBox

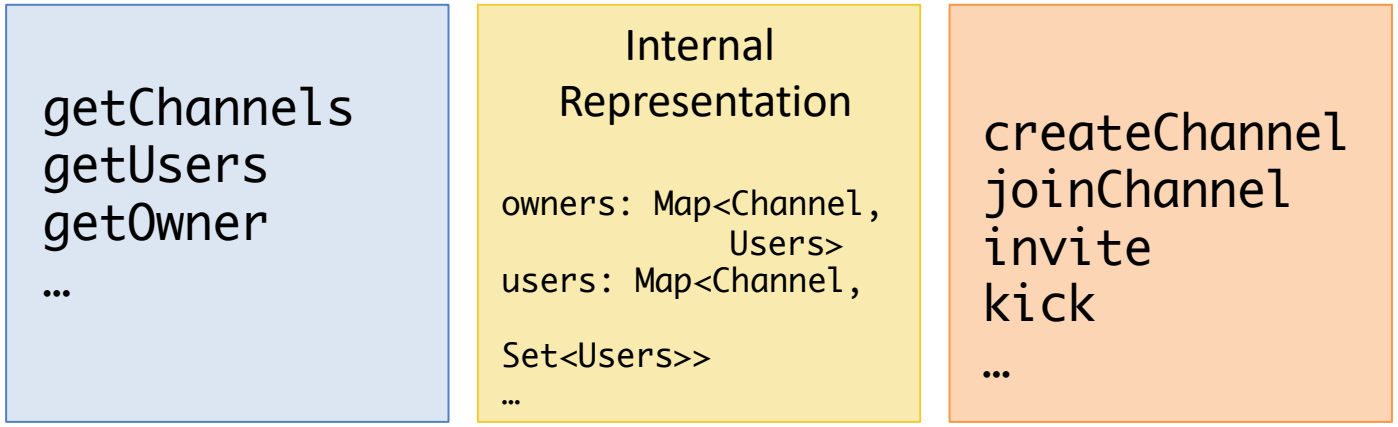



```
Class JToggleButton.ToggleButtonModel

boolean   isSelected()
void   setPressed(boolean b)
void   setSelected(boolean b)
```

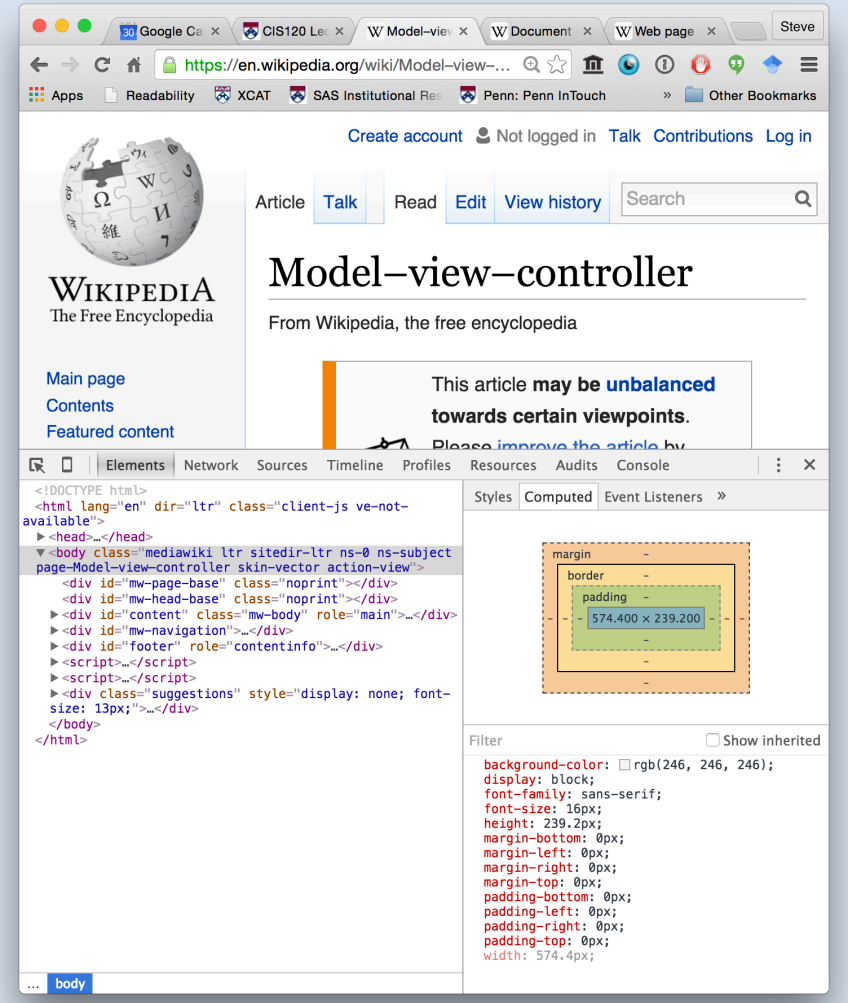
Checks if the button is selected.
Sets the pressed state of the button.
Sets the selected state of the button.

Example: Chat Server



ServerModel

Example: Web Pages



Views

Internal Representation: DOM (Document Object Model)

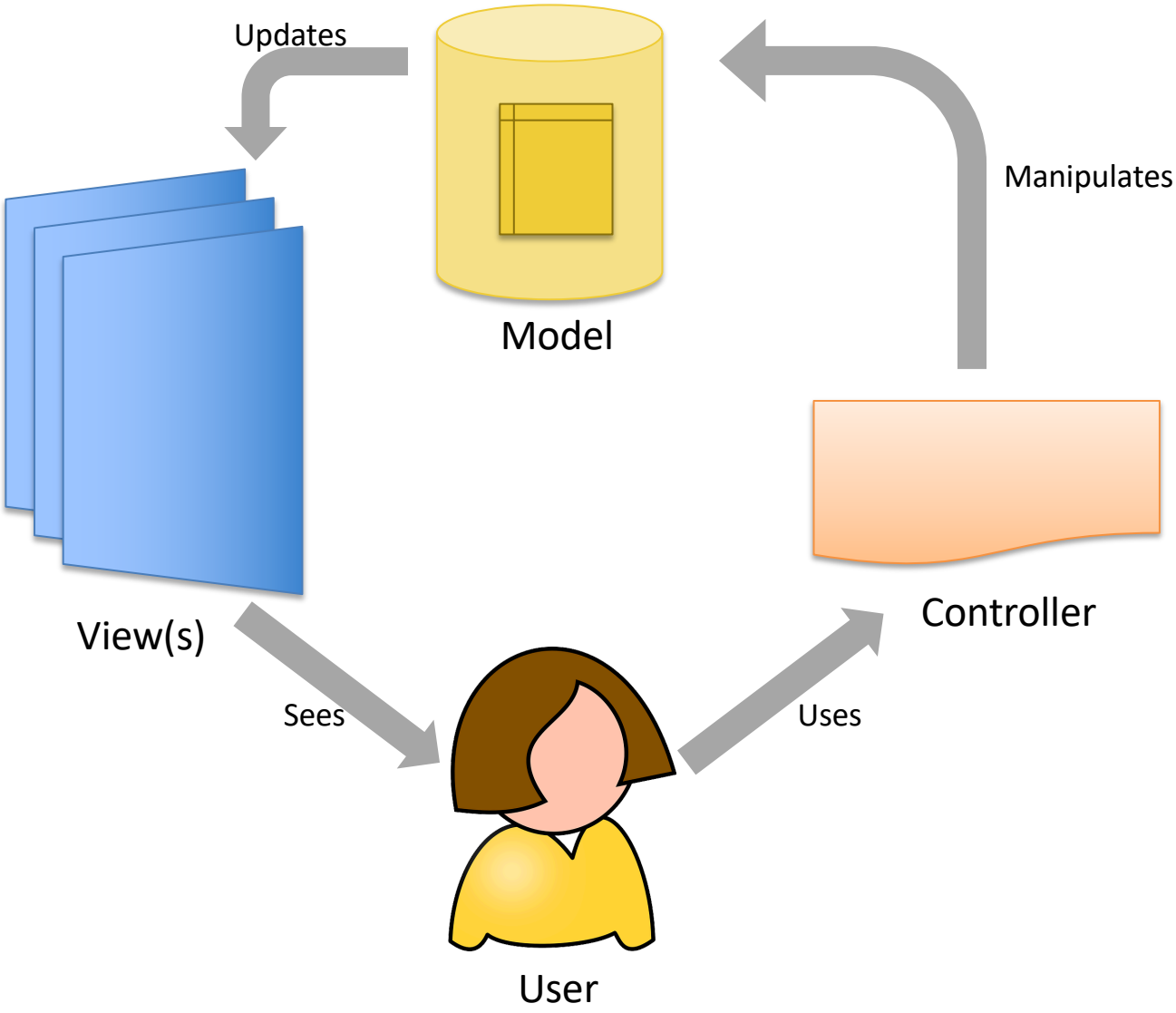
Model

JavaScript API

```
document.
addEventListener()
```

Controllers

MVC Pattern


Updates
Model
Manipulates
View(s)
Controller
Sees
Uses
User

MVC Benefits?

- Decouples important "model state" from how that state is presented and manipulated
  - Suggests where to insert interfaces in the design
  - Makes the model testable independent of the GUI

- Multiple views
  - e.g. from two different angles, or for multiple different users

- Multiple controllers
  - e.g. mouse vs. keyboard interaction

MVC Variations

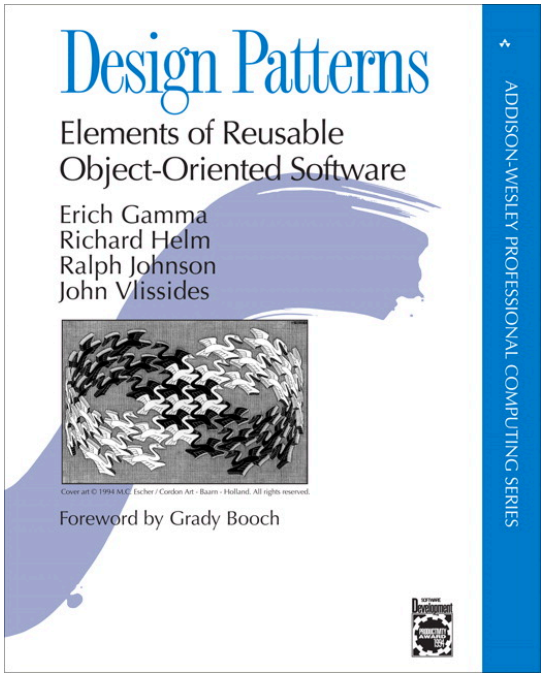
- Many variations on MVC pattern

- Hierarchical / Nested
  - As in the Swing libraries, in which JComponents often have a "model" and a "controller" part

- Coupling between Model / View or View / Controller
  - e.g. in MOD the Model and the View are coupled because the model carries most of the information about the view

Design Patterns

- Design Patterns
  - Influential OO design book published in 1994 (so a bit dated)
  - Identifies many common situations and "patterns" for implementing them in OO languages



- Some we have seen explicitly:
  - e.g. *Iterator* pattern
- Some we've used but not explicitly described:
  - e.g. The Broadcast class from the Chat HW uses the *Factory* pattern
- Some are workarounds for OO's lack of some features:
  - e.g. The *Visitor* pattern is like OCaml's fold + pattern matching