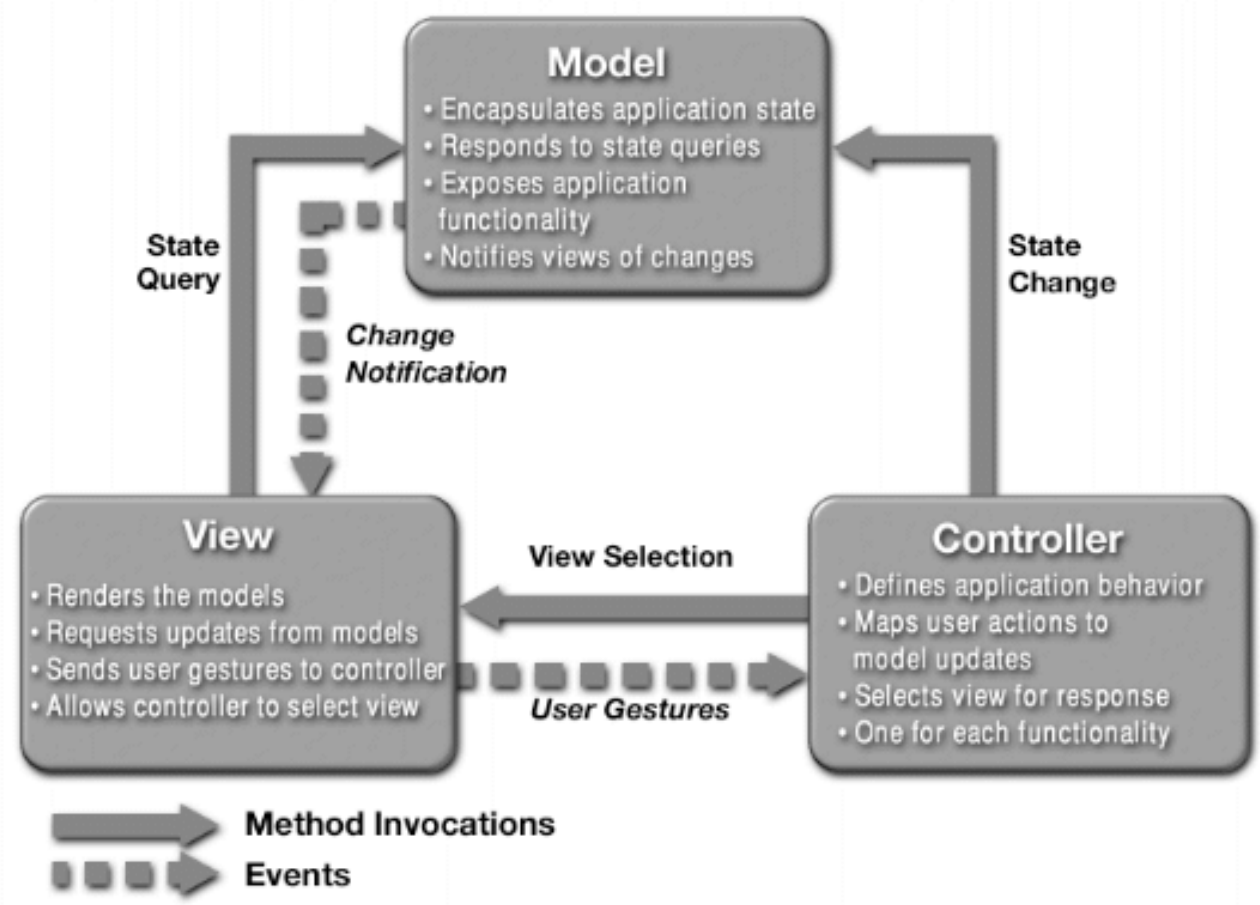
# MVC

COMP 401 Fall 2018

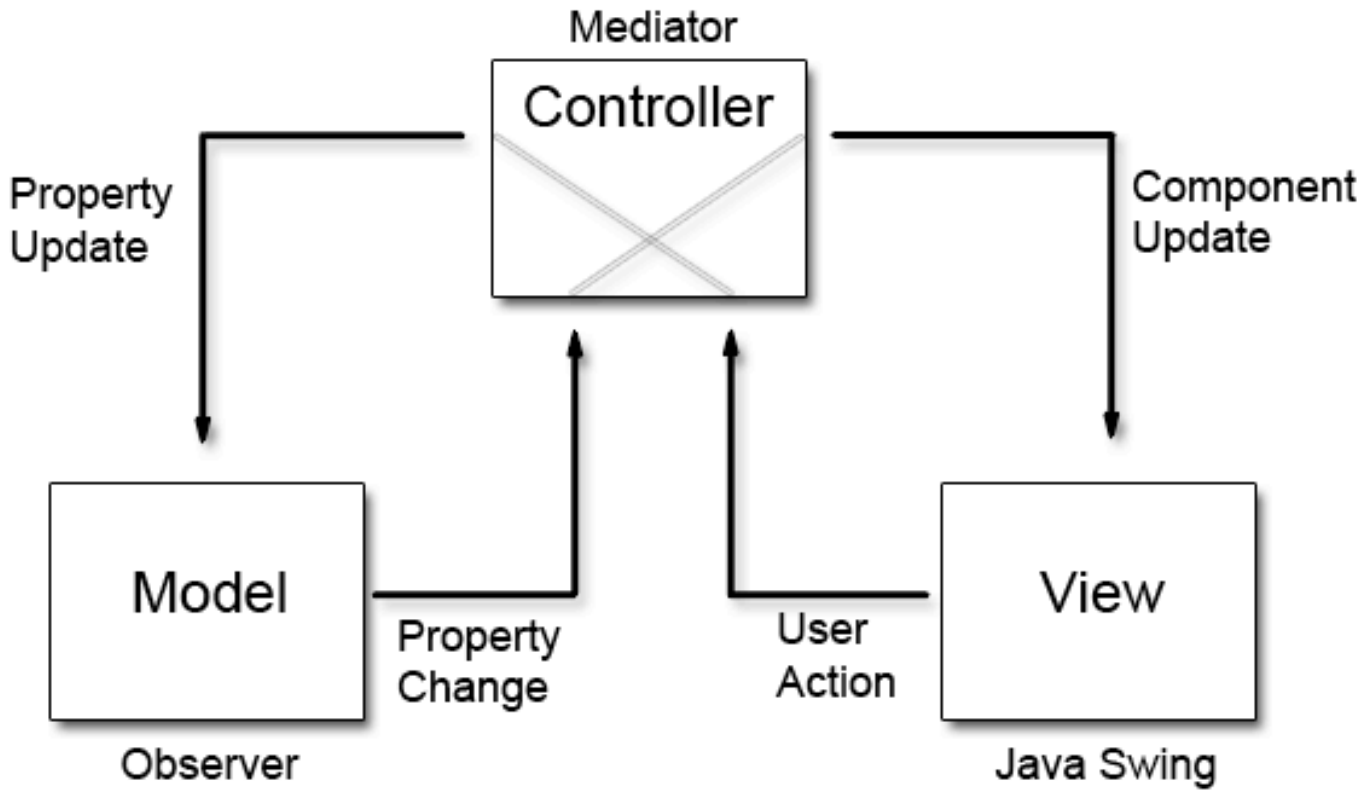
Lectures 19 and 20

# Classic MVC



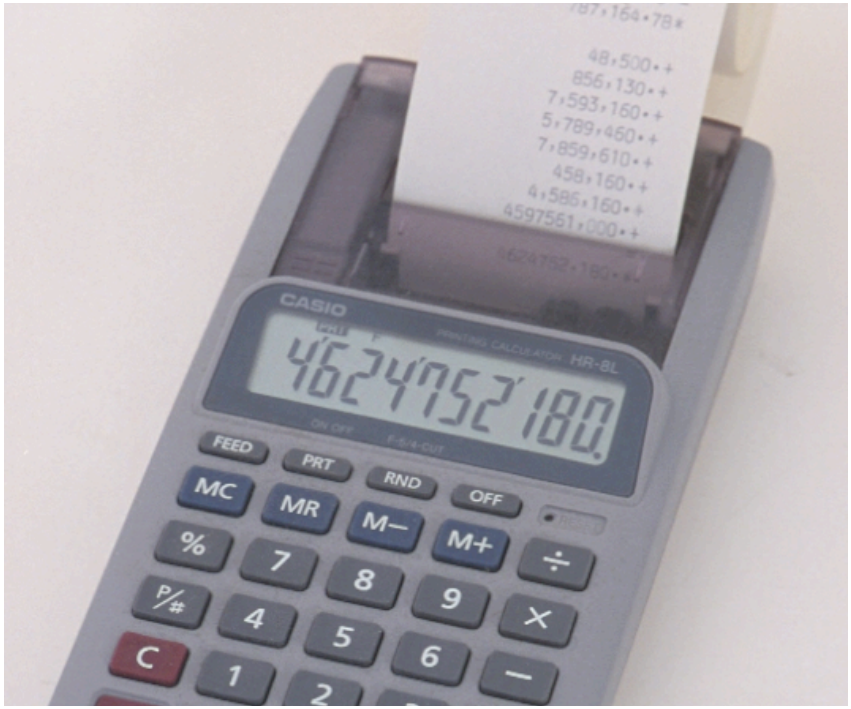

http://www.oracle.com/technetwork/articles/javase/mvc-136693.html

# Alternate MVC




http://www.oracle.com/technetwork/articles/javase/mvc-136693.html

# MVC Calculator Design




Start with the model. What does the model need to encapsulate?

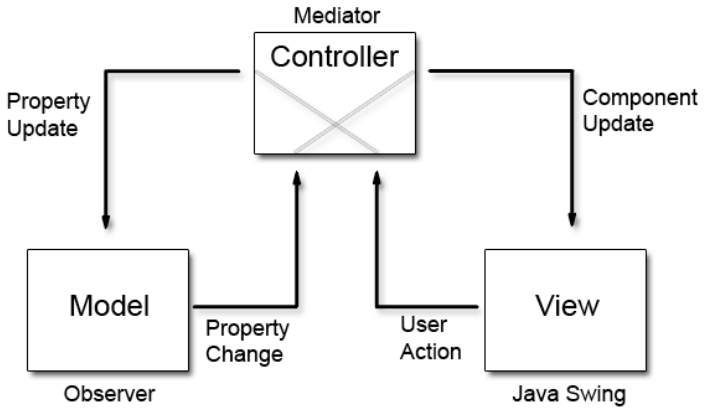
# lec20.v01

- CalculatorModel
  - Encapsulates a history (i.e., a sequence) of operations.
  - A current value on which the next operation will occur.
  - Observable
    - Notifies observers when model changes which in this case is whenever a new operation occurs.
    - CalculatorObserver is interface implemented by registered observers.
- Operation
  - Specifies an operation
    - Notice it doesn't specify the value that the operation applied to or the result.
      - Why doesn't that matter?

# MVC Calculator Design

- Now design the view
- lec20.v02

# Controller




- Controller sits between model and view
  - Responds to what happens in the view
    - In our case: button pushes
  - Instructs view to change when necessary
    - In our case: what is in the display, what is on the tape
  - Manipulates the model
    - Registers operations to affect the value
  - Responds to model changes
    - Causes registered operations to appear on the tape

# Controller

- We need a way for controller to register itself with all of the view's buttons.
  - lec20.v03
  - Notice that controller encapsulates references to model and view.
    - Model and view, however, don't know about each other or the controller.

# Handling Digits

- What should happen when a digit is pressed?
  - Depends on what has happened before.
    - If we are starting a new number, we should replace the display with this digit.
    - If we have already started a number, we should append the digit to the number already in the display.
  - Keeping track of this kind of logic and doing the right thing is exactly the job of the controller.
- Things we need to add:
  - A way for the controller to get/set the display
  - A field in controller to keep track of whether we are starting a new number or not and corresponding logic
  - lec20.v04

# Handling Decimal Point

- Should only allow one decimal point so needs to check to see if one is already there.
- Special case for beginning of number
- What we need:
  - Indicator of whether or not decimal point has already been pressed.
  - Logic to handle the button
  - lec20.v05

# Handling Operations

- What happens when an operator is pressed?
  - The operator indicates which operation is now in progress.
    - But we can't actually do the operation until another number is entered.
    - If an operation was already in progress, that operation should now complete using number in the display
- Things we need to add:
  - Field in controller to keep track of operation in progress.
  - Logic to apply operator in progress to model at the right time.
    - Right time = when operand is complete and next operator is known.
    - To get operand, we need to add method in view object to retrieve current display and interpret it as a number.
  - lec20.v06

# Seeing the Output

- Controller now handling most buttons and sending operations to model
  - Need to react to changes in the model
- What we need:
  - Implement CalculatorObserver
  - Attach to model
  - Handle observed changes to model.
  - lec20.v07

# Four Bugs and An Inversion

- Delay in seeing result on tape when pressing =
  - No need to wait for next number when = is current operation
- Tape doesn't scroll
  - Just need to put it in a ScrollPane
- Can't change mind about operation
  - To fix: if still at start of number, just replace operation.
- Can't make a zero
  - Logic for starting a number incorrect
  - Need to separate case of starting a number after operation was pressed that allows 0
- Still need to handle +/-
  - Need to watch out for fact that first character of this button is same as addition operation.
- lec20.v08

# Reconsidering Design

- Things work (for the most part)
- But something is wrong with our design…
  - … think about what we would need to do in order to support keyboard numbers.
  - … think about what we would need to do in order to support buttons with image-based icons
- Current interaction between controller and view requires controller to know about view internals.
  - Instead of exposing raw underlying UI events, we should translate them into something more abstraction-specific.
  - lec20.v09
- And now we can add keyboard input
  - Only changes the view object. Controller remains the same.
  - lec20.v10