MVC Architecture (A high-level organization for GUIs)

Model: stores and manipulates "core" program data

- "Essential" data and operations representing program state, independent of how this is shown to the user
  - » e.g. a single chess game model object might be hooked up to a text-based or GUI based interface.

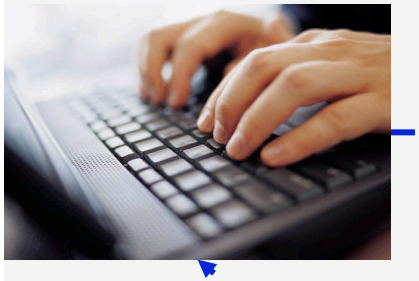
View: visualizes state of the model

- Filtering: what parts of the model are visualized?
- Visualization: how the chosen elements are drawn
- A single model may be used with *multiple* views

Controller: collects user and system events

- Interaction with the view and/or model
  - » e.g. user selects what to show (e.g. selecting a tab)
  - » e.g. user replaces a value in a spreadsheet (updated model as well as view)

MVC Diagram: Manipulating a circle viewed in a GUI



User/system events

Controller
Collects button, key, timer events etc.

View update
(e.g. user changes color)

Model update
(e.g. new circle radius)

Rendered
GUI Appearance
(shown in GUI Window(s))

View

Model
(Circle attributes)

View update
(e.g. new circle radius)

Note: In Java Swing (and many other GUI toolkits), View/Controller are often merged (e.g. JButton)

Simple MVC in Swing

Model

Implemented using an object from a class with event listener and event listener *registration* methods

View(s)

Implemented as listeners registered with a model object; notified when the model (state of the "core" program) is updated.

Controller

Implemented using widgets with registered listeners that invoke model and/or view operations as needed.

Model-View-Controller (MVC) Example

CircleModel.java

CircleView.java

CircleController.java

MVCDemo.java

(In Liang)