Model-View

COMP 401 Fall 2018

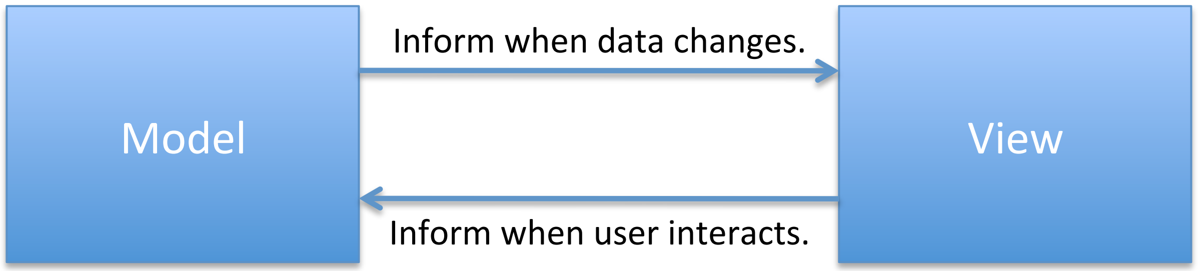
Lecture 18

ColorChooser Widget Review

- Color object embedded within the widget
  - Fine for a small piece of immutable information
  - Not so great for a complex abstraction
    - UI may not always be needed

Model – View Pattern

- Keep object representing the data separate from widget providing the UI
  - Model: Object representing abstraction within your application.
  - View: Object (widget) representing UI to model.



Model-View Example

- Playlist
  - This is the model object
  - Manages a list of Song objects
- PlaylistView
  - This is the view object
  - Provides a UI for interacting with Playlist

lec18.v1

- Playlist

lec18.v2

- PlaylistView
  - Encapsulates a reference to Playlist object
  - Builds a simple view of the playlist

lec18.v3

- AddSongWidget
  - An interface for adding a new song to the Playlist

lec18.v4

- Connecting model to view
  - Model needs to inform the view whenever it changes so that the view can rebuild to reflect any changes.
- My approach here is to make the model observable.
  - And to register PlaylistView as an observer
- Notes:
  - revalidate() required whenever contents of a Java Swing container are changed

lec18.v5

- Notice duplication of code in constructor when list is first built and when rebuilt when responding to model update.
  - Refactor into a helper method.
- JScrollPane
  - Wraps another Swing component that may change size and provides scrolling view as needed.

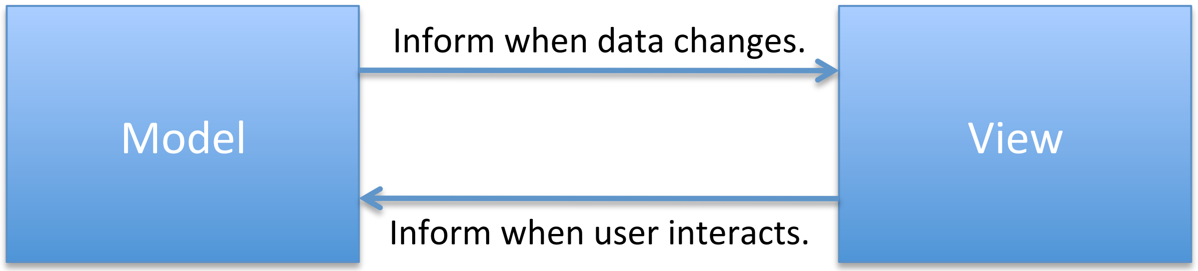
lec18.v6

- Add a shuffle button

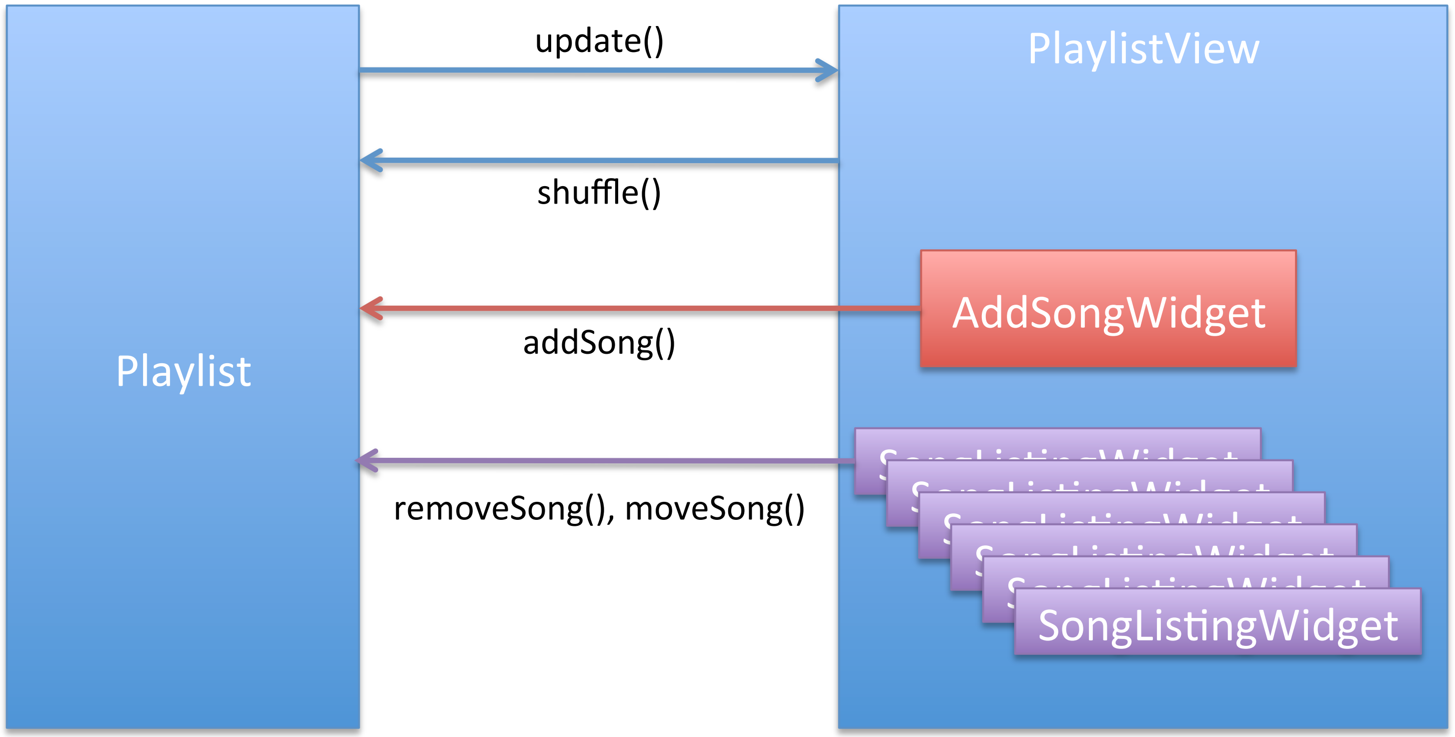
lec18.v7

- Create more interesting song listing
  - SongListingWidget
    - Includes a buttons for removing the song from the playlist and moving them up and down the list.
- BorderFactory
  - Swing class for creating borders for components
- Unicode characters
  - Lots of glyphs available
  - Changed Song to use star glyph for rating
- Added method to playlist for moving songs

Model / View


Model
Inform when data changes.
View
Inform when user interacts.

Playlist / PlaylistView



Model / View Summary

- Model provides access to an abstraction.
  - Any manipulation / modification done via public methods provided by model.
- View builds an interface for rendering current state of the model.
  - Interactions with interface translated to appropriate calls on model methods.

Model-View Inadequacies

- Good for simple, direct UI elements
  - Just one model object driving UI view object
  - UI reflects model state directly
  - User interactions with view have direct mapping to updates/public methods of model.
- Complex behavior harder to express
  - Dynamic user interfaces
  - Complex models
  - Contextual user interactions