**Lesson 7 – Bringing it all Together**

* Need to make the GUI respond to user activity
* We clumped our code into 3 basic groups
  + **Model**
  + **View**
  + **Startup**
* **Model**
  + Everywhere we manipulate our data
  + Describes how the program functions
  + Methods are used to manipulate data stored in instance variables
* **View**
  + All the display (no manipulation)
  + This code is separate from the Model
  + Responsible for displaying both the components for user interaction and the information about the state of the data contained in the Model
  + Methods are used to create and update the display
    - Update Method accesses attributes from the Model Class
* **Startup**
  + Very limited functionality
  + Creates Instance of the View and Model and links them so they can work together
  + This is the Class with a Main Method
  + A Frame containing the View is displayed
* **Events**
  + Events generally work the same way for all components
  + When something happens to a JComponent an event object is created describing the event
    - When it occurred
    - What keys (if any) were pressed
    - Which JComponent created it
  + We need to create a new class that requires this event object as a parameter
    - This class then updates the data in the model which indirectly updates the data in the view
      * This Class (or set of classes) are called the Controller Class
* **Controller Class**
  + A part of the GUI that contains code that responds to user actions/events
  + A separate class is created for every JComponent that will be manipulated
  + Must extend Object class and implement a Listener specific to the type of interaction it responds to
    - Usually ActionListener or MouseListener
  + Contains instance variables for the model and JComponent and one (or more) method(s) that respond(s) to the event(s)
  + Name of the Controller Class is named for the JComponent that is being manipulated
  + Pass in the Model and JComponent to the Class Constructor
  + The Methods in the class are those that are required by the Listener’s interface
    - These methods run at any time the JComponent is being manipulated
    - They are passed the event object that was generated by the JComponent
      * Example: ActionEvent Object
  + Before we can make this work, we must register the Controller
    - This is done by using the add(type listener)Listener to a JComponent
      * Provides an instance of the controller as an argument
    - Since most GUI’s have several controllers the View Class should have a helper method to add listeners
      * Must be called in the View’s Constructor
* Since the Controller calls methods in the Model to manipulate the data there needs to be a way for the Model to inform the View that data has been changed
  + This is done by adding an update() method in the View Class
    - This method redraws the GUI using accessor views from the Model Class
    - All methods in the Model that modify data must end by calling the View’s update() method
* We must now link the View and Model Class
  + This is done in the Startup Class
* **Startup Class**
  + Creates an instance of the Model and View Class
    - Creates a JFrame to display the View
  + The Model must have an attribute for the View
    - The Model Class contains a public method to instantiate the View
      * Allows the creation of multiple views for the same model
  + View Class
    - Has an attribute for the Model
      * Constructor instantiates the model and calls its setGUI method so the Model knows which View its working with