M7 (b) – Inversion of Control

Jin L.C. Guo

Image Source: https://c1.staticflickr.com/9/8363/29350436510_efe6626995_b.jpg

Objective

- Be able to Use Callback to achieve decoupling
- Be able to use the Observer design pattern effectively;
- Event Handling in GUI applications
- Understand the concept of an application framework;
- Understand the Model-View-Controller Decomposition

Objective

- Be able to Use Callback to achieve decoupling
- Be able to use the Observer design pattern effectively;
- Event Handling in GUI applications
- Understand the concept of an application framework;
- Understand the Model-View-Controller Decomposition

Event

- A notification that something interesting has happened.
- Examples in Graphic Interface?

Move a mouse

User click a button

Press a key

Mouse press and drag

Menu item is selected

Popup window is hidden

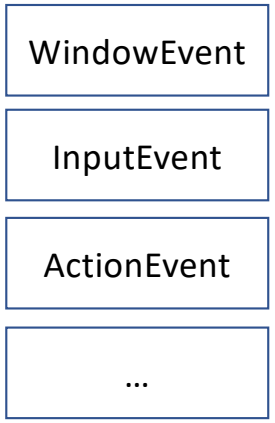
Window is closed

How to capture event and act accordingly

- Define an event handler

implement

Interface EventHandler<T extends Event>



void handle(T event) **<= Callback method**

Invoked when a specific event of the type for which this handler is registered happens.

```
Public class MyEventHandler implements EventHandler<ActionEvent>
{
   @Override
   public void handle(ActionEvent event)
   {
       //Event Handling steps
   }
}
```

How to capture event and act accordingly

- Instantiate and register the event handler

```
Button btn = new Button();
btn.setOnAction(eventHandler);
```

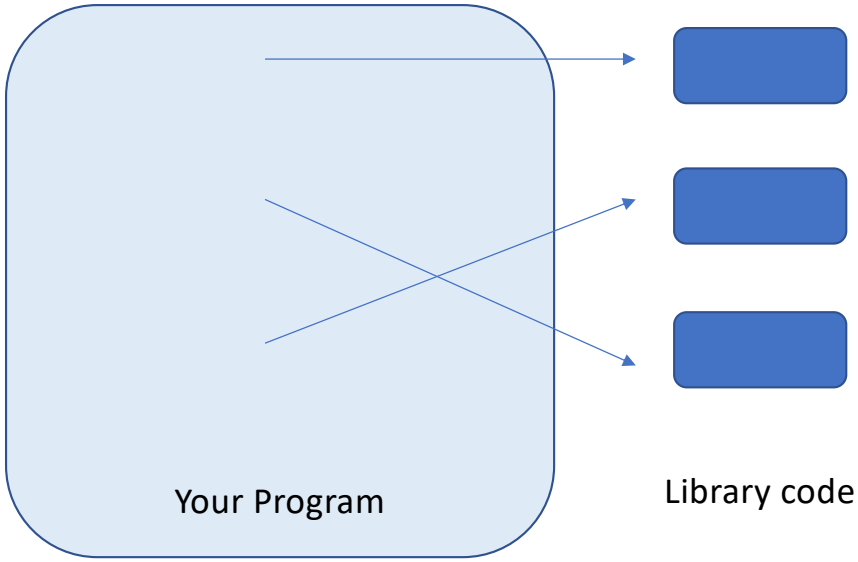
Instance of MyEventHandler

How to capture event and act accordingly

- Instantiate and register the event handler

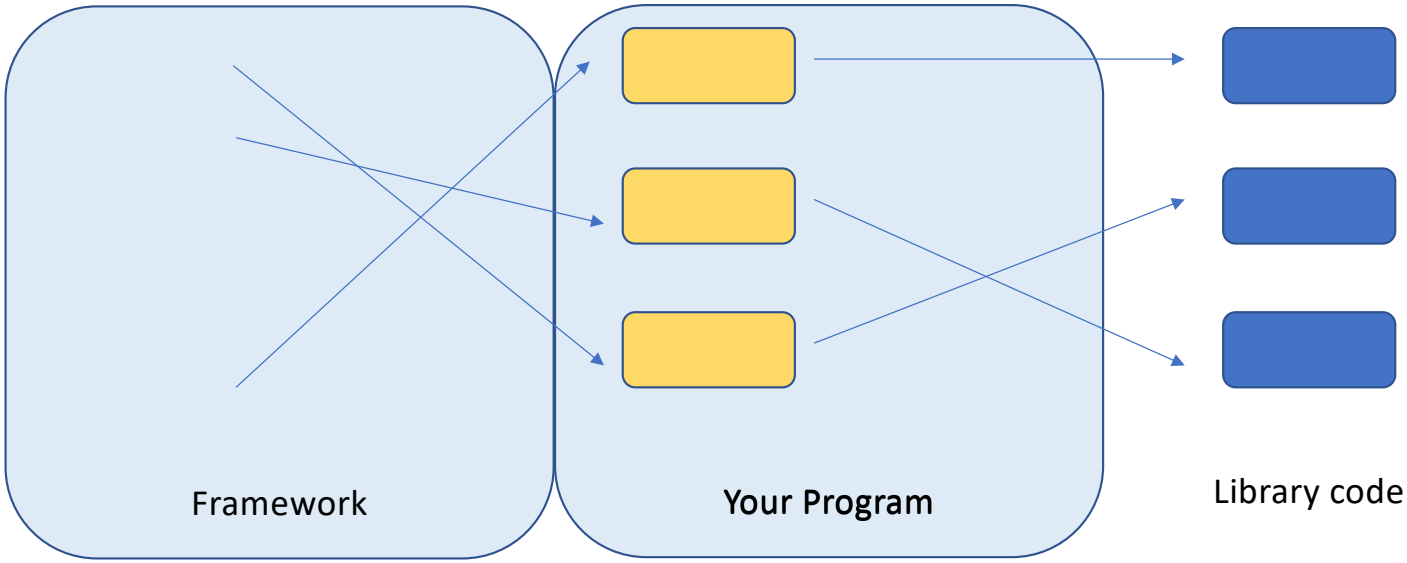

```
Button btn = new Button();

btn.setOnAction(new EventHandler<ActionEvent>() {
            public void handle(ActionEvent event) {
            //Event Handling steps
            }
        });
```

Library vs Framework



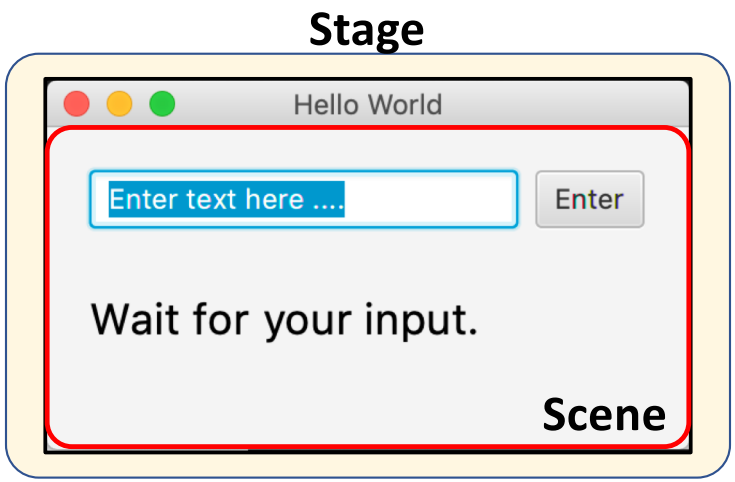

```
public static void main(String[] arg)
```

Library vs Framework

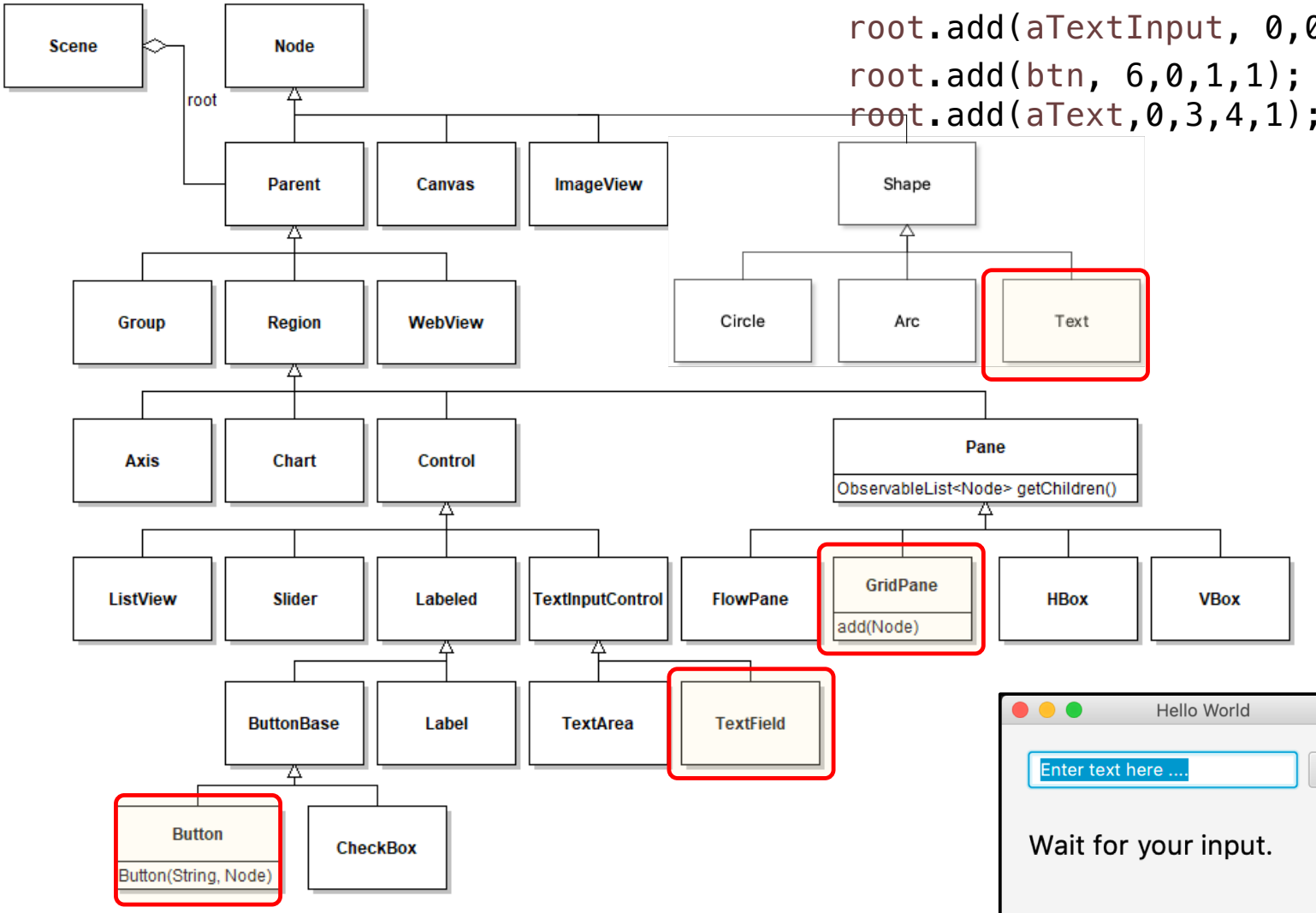


Launch JavaFX framework

```
public class MyApplication extends Application
{
    /**
    * Launches the application.
    * @param pArgs This program takes no argument.
    */
    public static void main(String[] pArgs)
    {
        launch(pArgs);
    }

    @Override
     public void start(Stage pPrimaryStage)
     {
         //Setup the stage
          pPrimaryStage.show();
     }
}
```




```
primaryStage.setScene(scene);
```

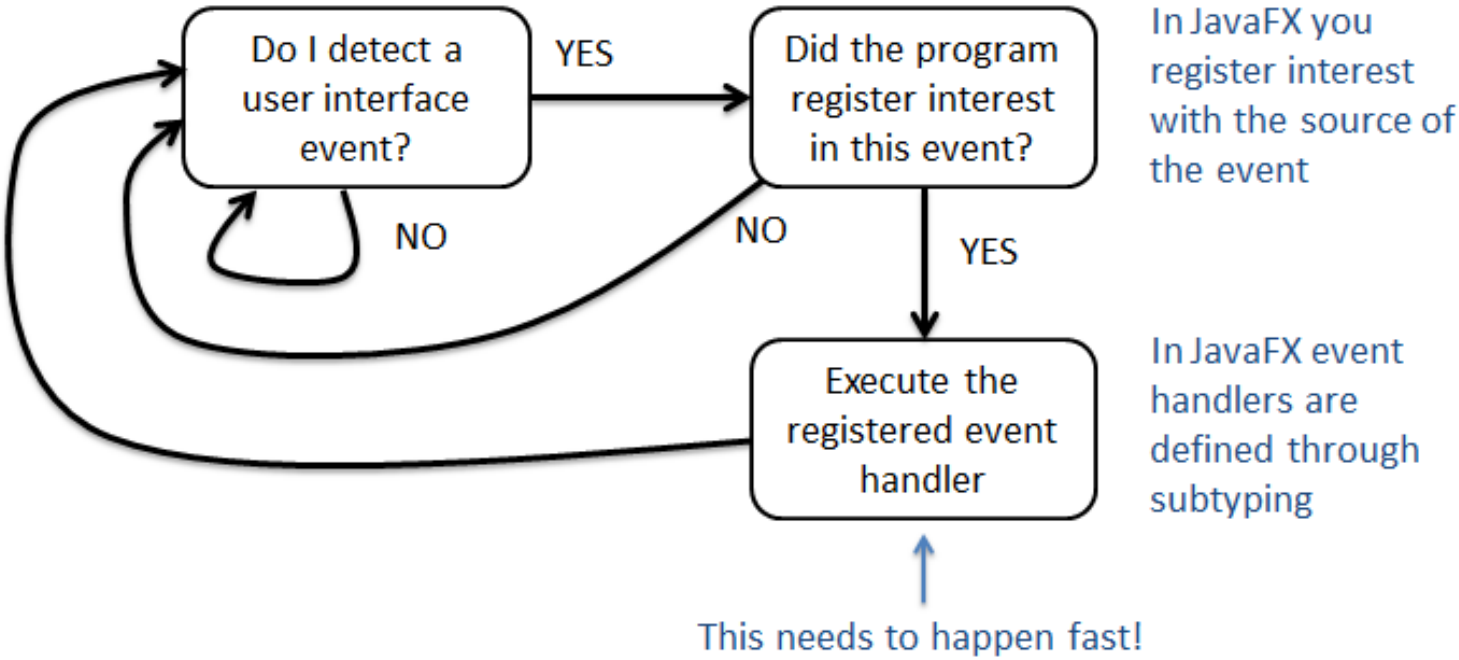



```
root.add(aTextInput, 0,0,2,1);
root.add(btn, 6,0,1,1);
root.add(aText,0,3,4,1);
```



```
Scene scene = new Scene(root, Width, Height);
```

When does event handling happen?



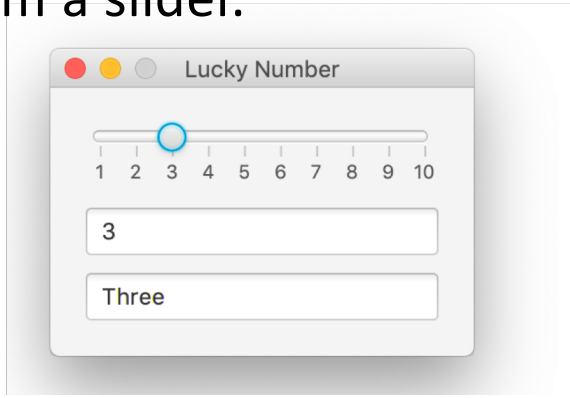
Text Display Demo

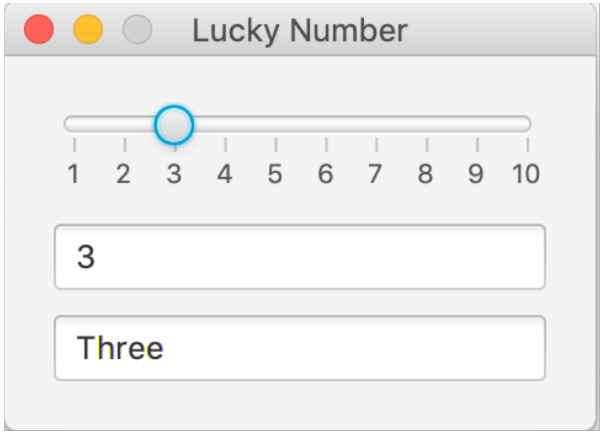
Lucky Number Example

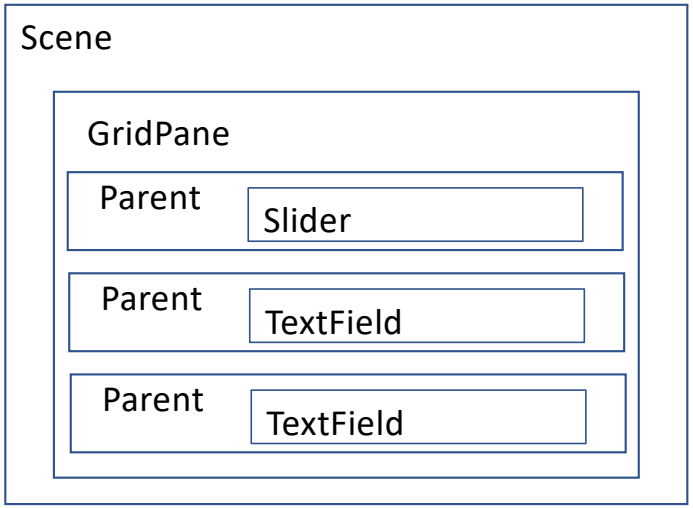
The user should be able to select a number between 1 and 10 inclusively.

The current selection should also be able to viewed in the integer and text fields and the slider.

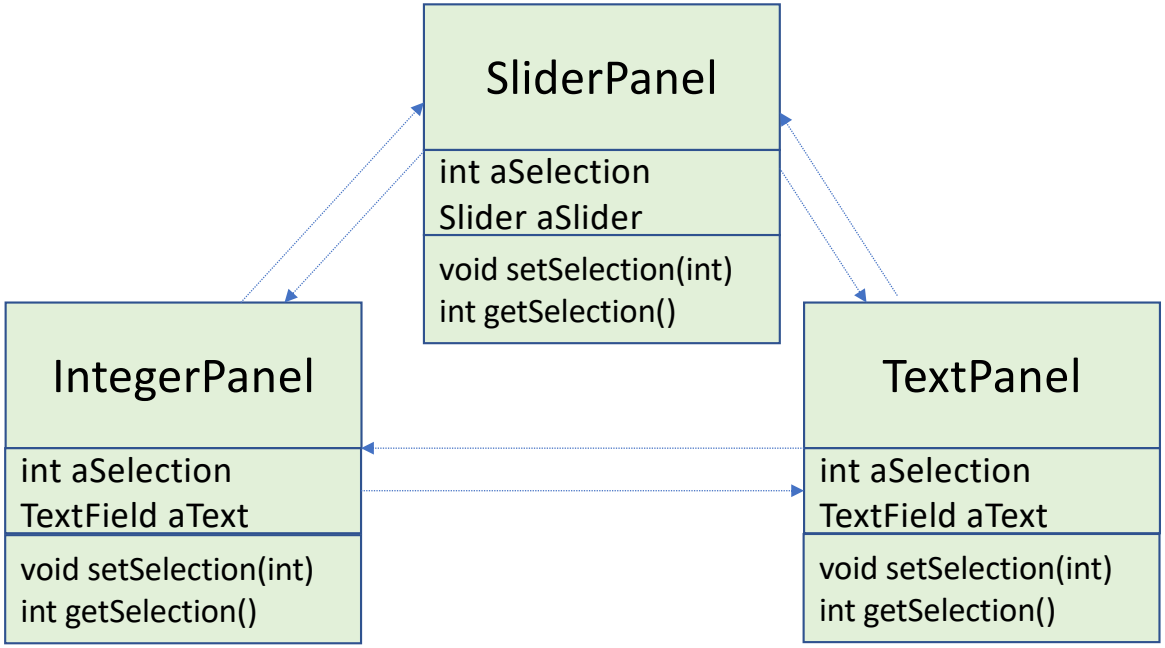
The selection should be performed through either typing it, writing it out in the corresponding fields, or selecting it from a slider.




Lucky Number
1 2 3 4 5 6 7 8 9 10
3
Three


Scene
GridPane
Parent
Slider
Parent
TextField
Parent
TextField

Problem Decomposition



High Coupling

Components are inter-dependent

Low Extensibility

hard to add/remove selection mechanism

MVC Decomposition

Model – View – Controller

Design pattern

Architectural pattern

Guideline to separate concerns

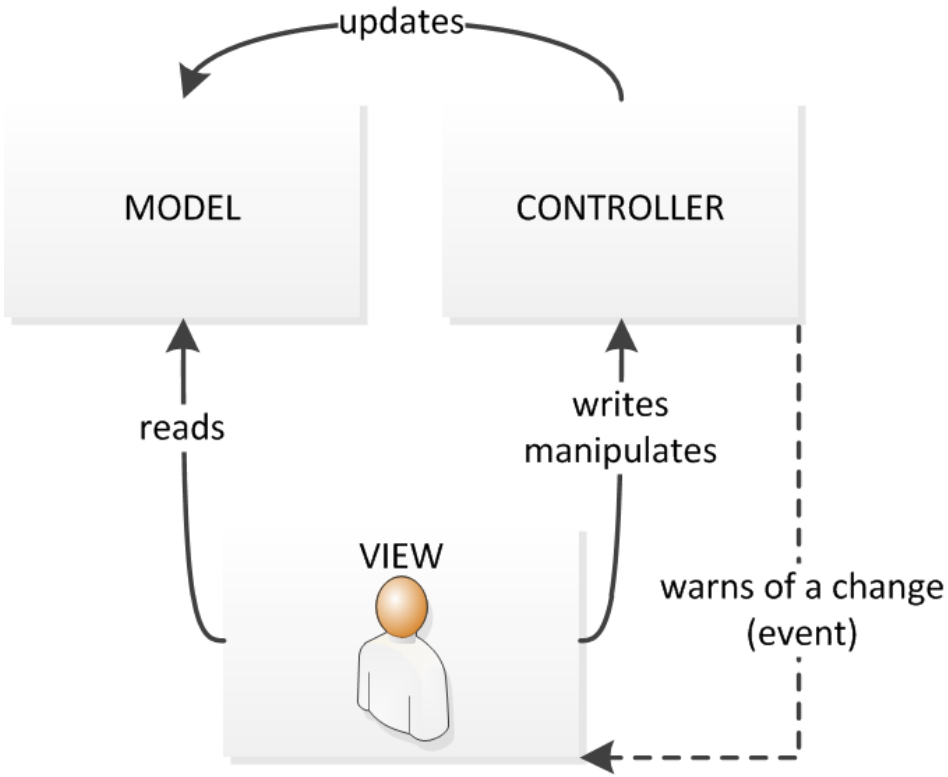
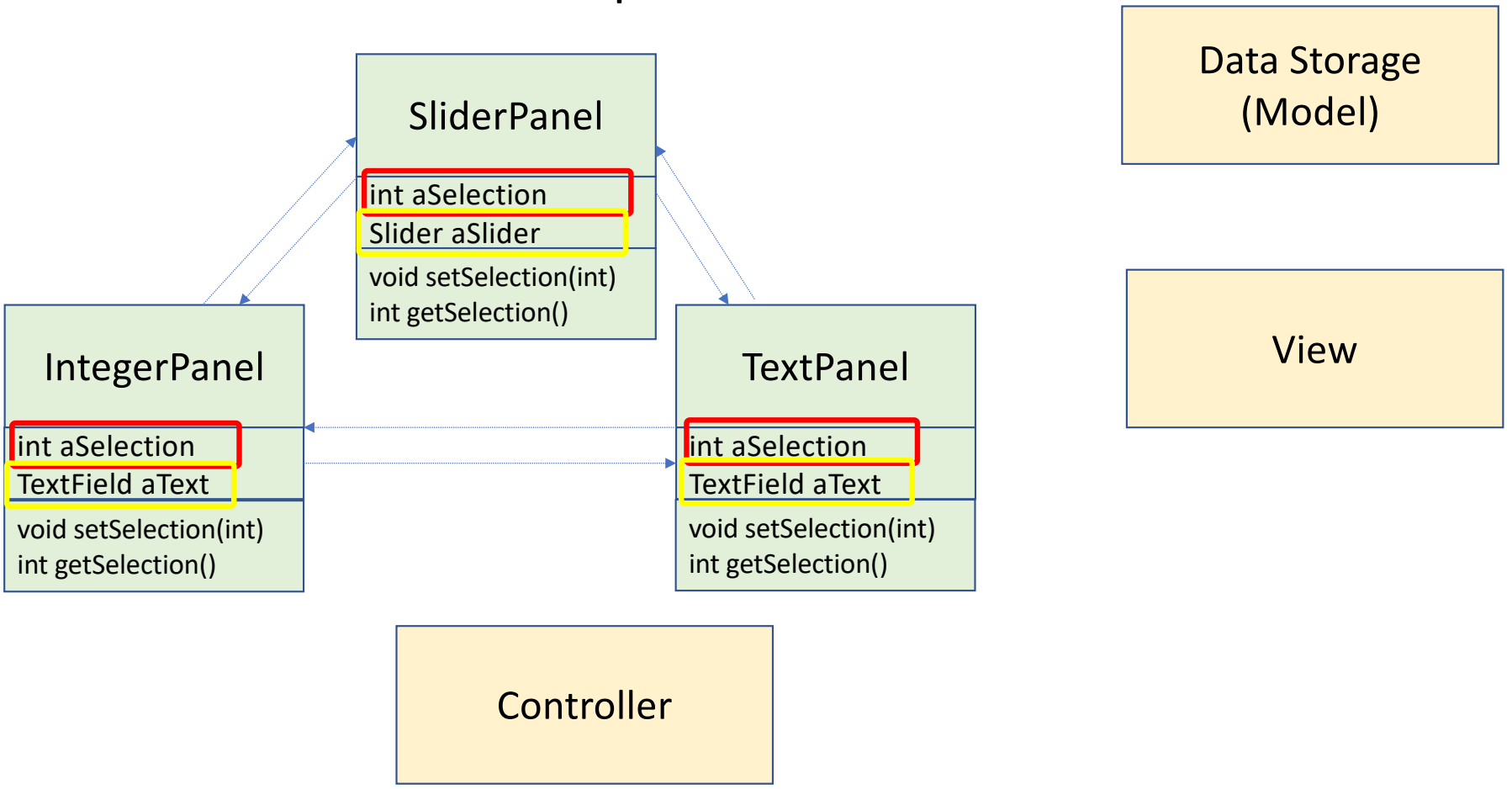


Image Source: https://upload.wikimedia.org/wikipedia/commons/6/63/ModeleMVC.png

Problem Decomposition

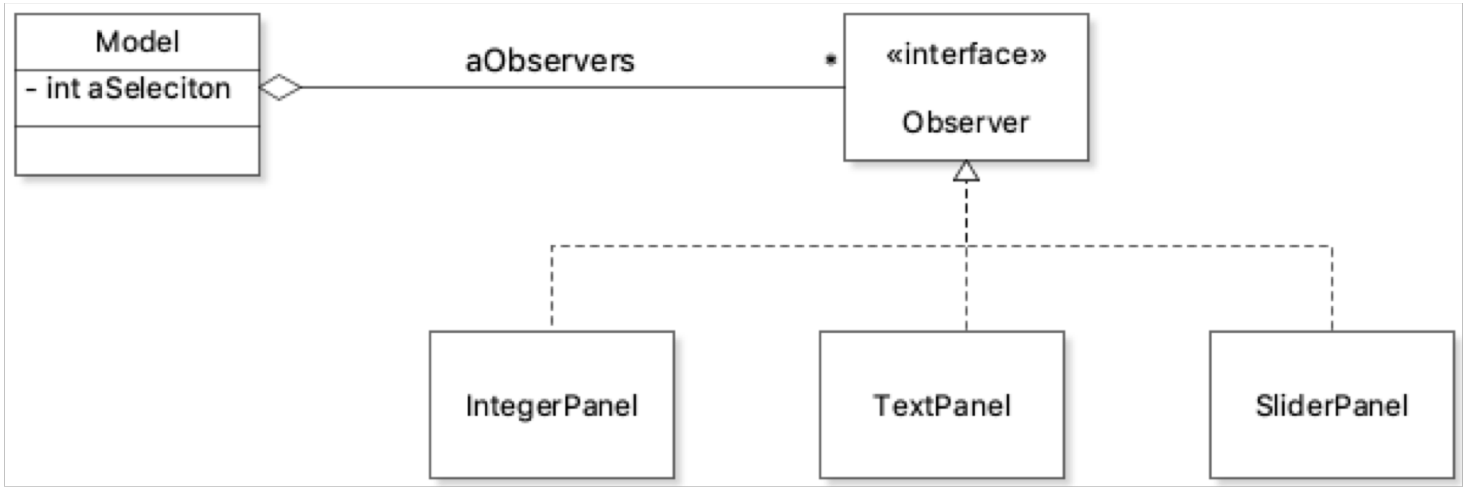


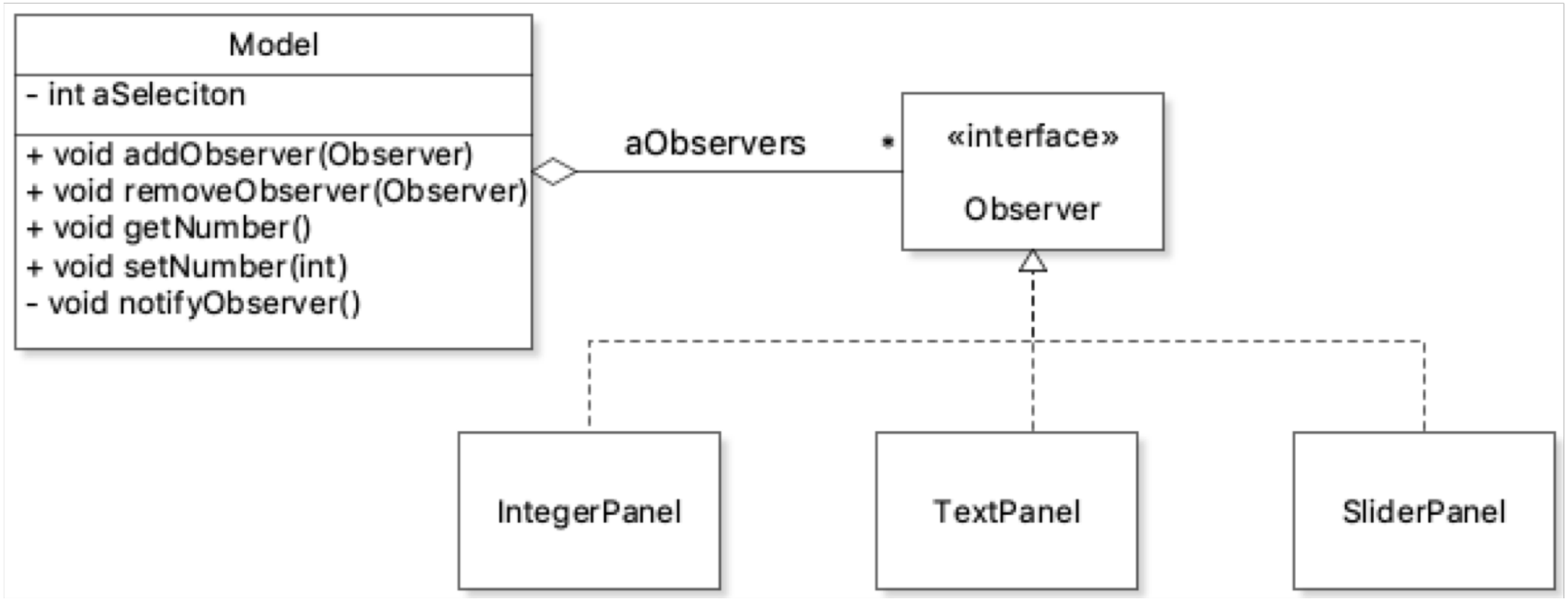
Activity

- Improve the design using Observer Pattern and MVC decomposition.

Activity: Applying Observer in MVC

- What methods should be included in Model?




Model
- int aSeleciton
+ void addObserver(Observer)
+ void removeObserver(Observer)
+ void getNumber()
+ void setNumber(int)
- void notifyObserver()
aObservers
*
«interface»
Observer
IntegerPanel
TextPanel
SliderPanel