CISC 3120

C16a: Model-View-Controller and JavaFX Styling

Hui Chen

Department of Computer & Information Science

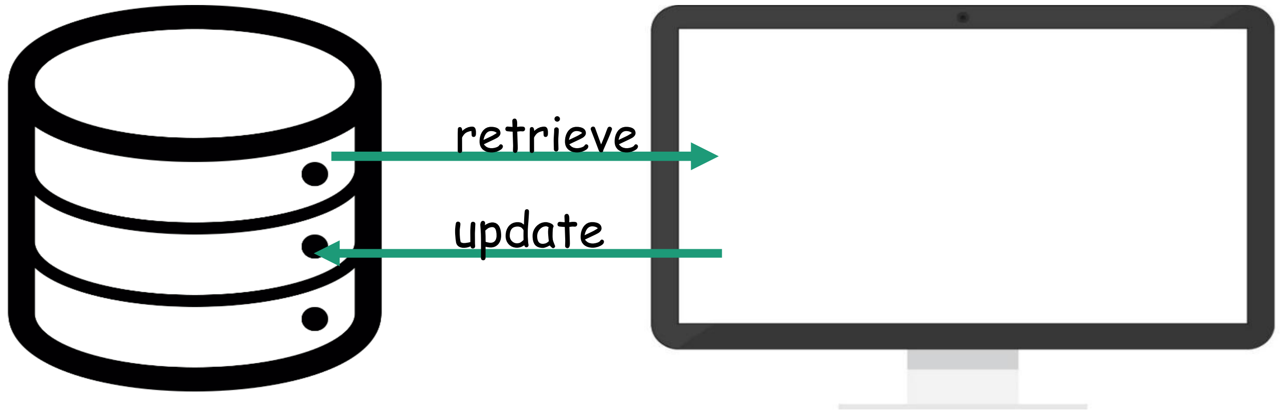
CUNY Brooklyn College

Outline

- Recap and issues
- Model-View-Controller pattern
  - FXML and Scene Builder
- Styling user interface with CSS
- Brief introduction to graphics, media, and charts

Applications: Data and View

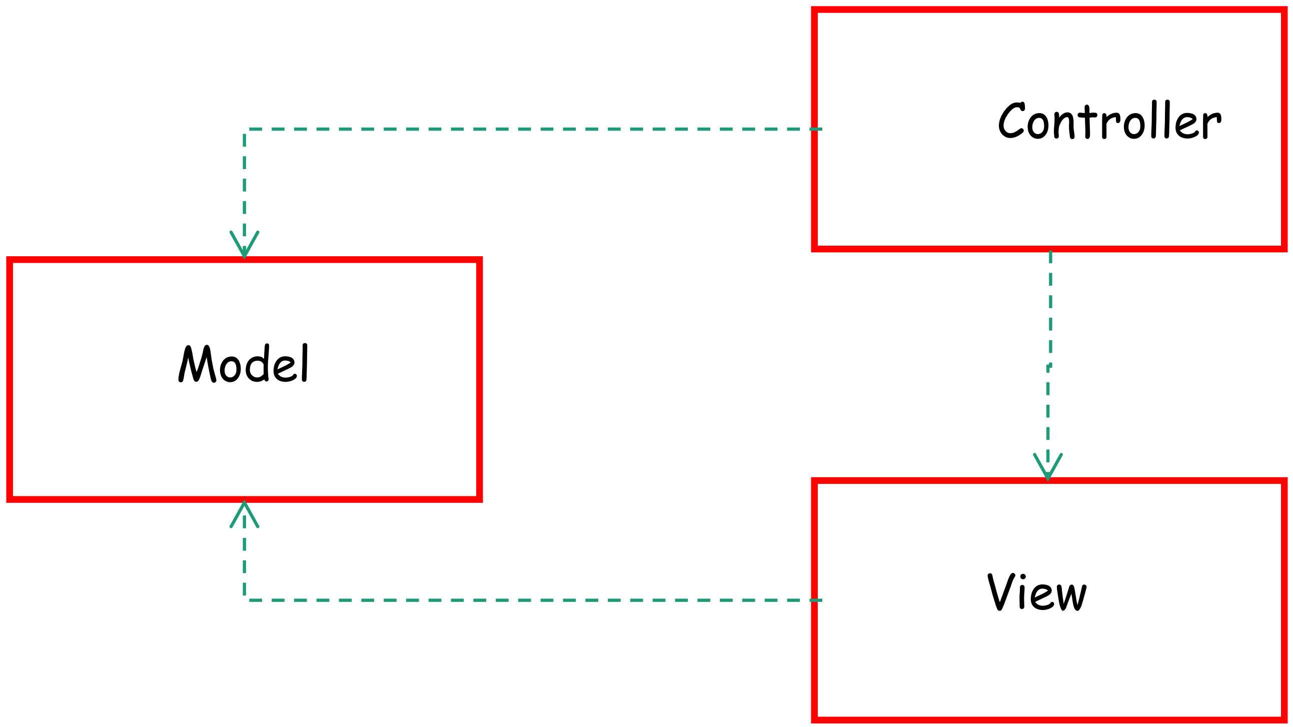
- Many computer systems are to display data and update data in a data store
- Two sets of terms
  - Business logic: the modeling of the application domain (model, data, business logic)
  - User interfaces: the visual feedback to the users (view, presentation)



Need to Separate Concerns

- User interfaces often changes more frequently than business logic
- Applications may display the same set of data differently
- User interface design and application logic design require different skill sets
  - User interface design and user interface development are two different concepts
- User interface code tends to be more device-dependent than business logic
- Create automatic tests for user interfaces is generally more difficult and time-consuming than business logic

Model-View-Controller



Model-View-Controller

- It separates an application three separated components/classes,
  - (Model) the modeling of the application domain
  - (View) the presentation,
  - (Controller) and the actions based on user input
- *A* fundamental design pattern for the separation of user interface from business logic.

Model

- Model is independent of view and controller
- Manages the behavior and data of the application domain
- Responds to requests for information about its state (usually from the view)
- Responds to instructions to change state (usually from the controller).

View

- Depends on model, but is independent of the controller
- Manages the display of information.

Controller

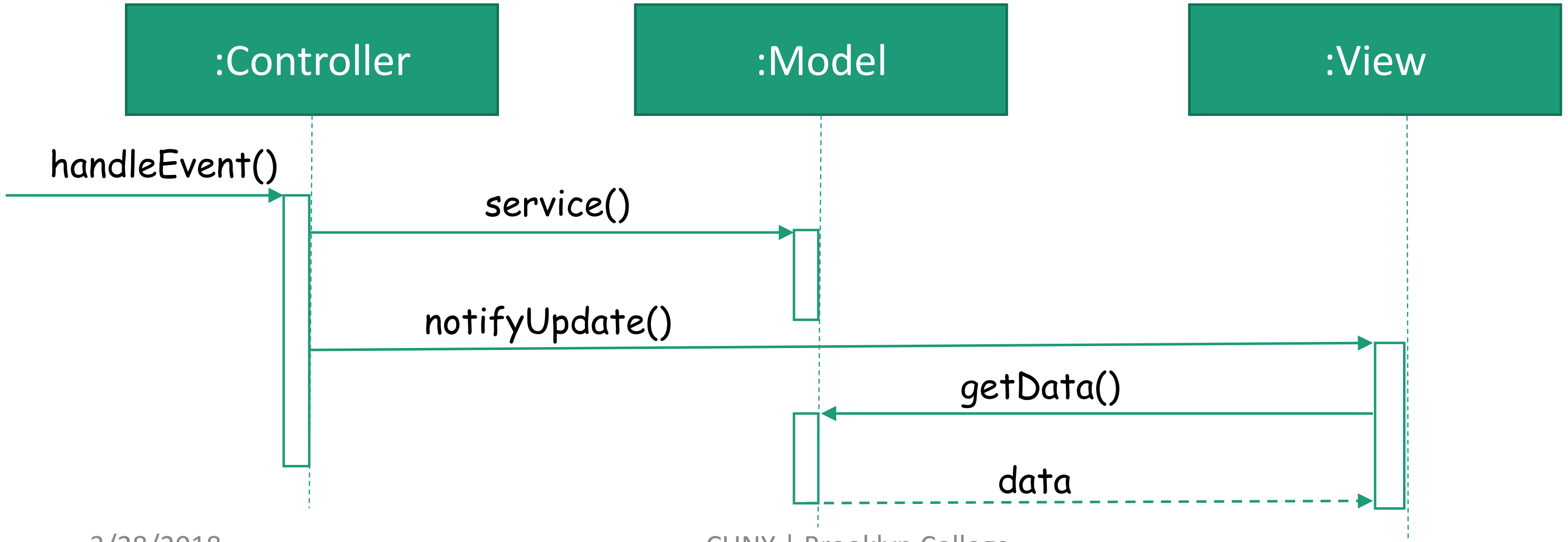
- Depends on both model and view
- Interprets the mouse and keyboard inputs from the user
- Inform the model and/or the view to change as appropriate.

MVC Models

- Passive model
- Active model

Passive MVC Model

- One controller manipulates the model exclusively
  - updates the model (data)
  - inform the view that the model has changed and request the view to refresh

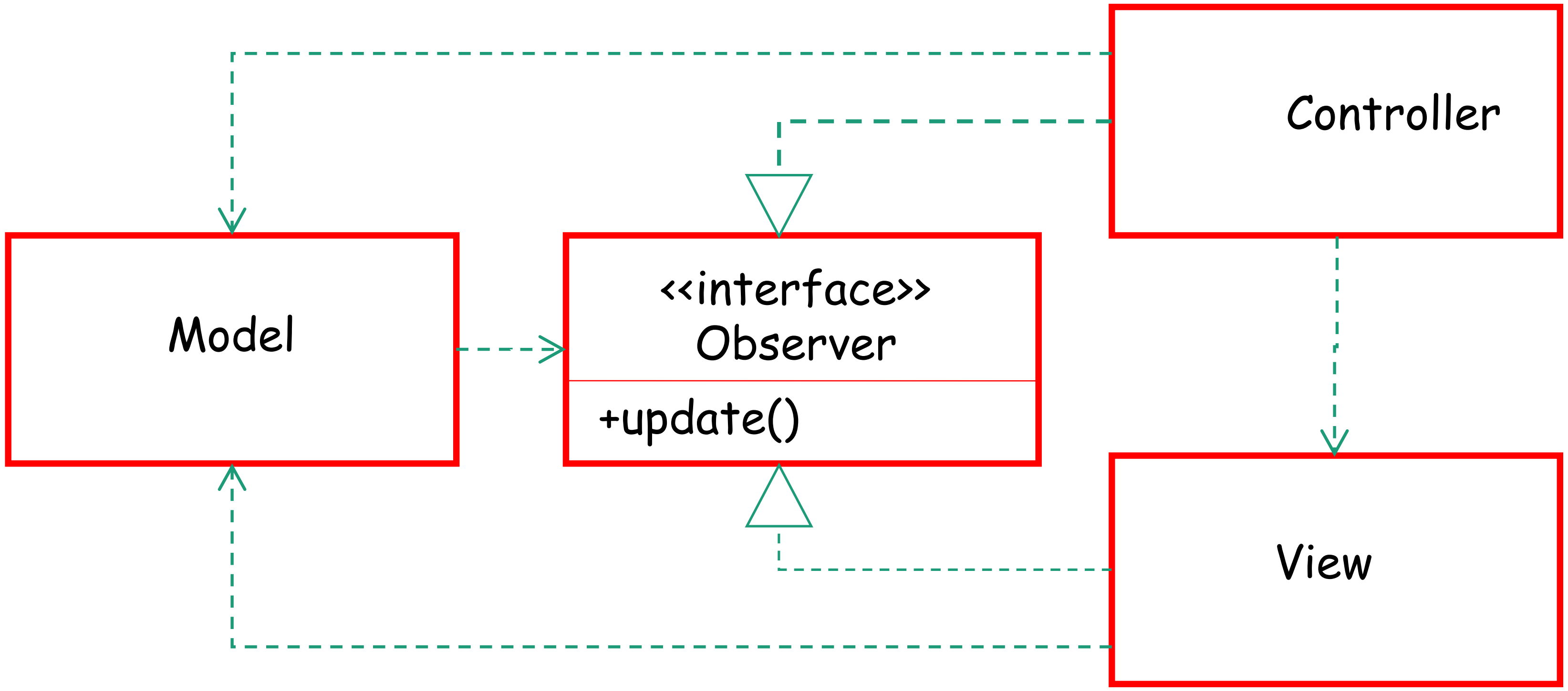


Passive MVC Model: Discussion

- Model is passive
  - Model is completely independent of View and Controller
  - Model does not notify View or Controller any changes on it
  - Controller is responsible for updating model, and for requesting view to refresh
- Often realized via dependency injection
- Limitation
  - If model can be updated from multiple controllers, the view may be out-of-date

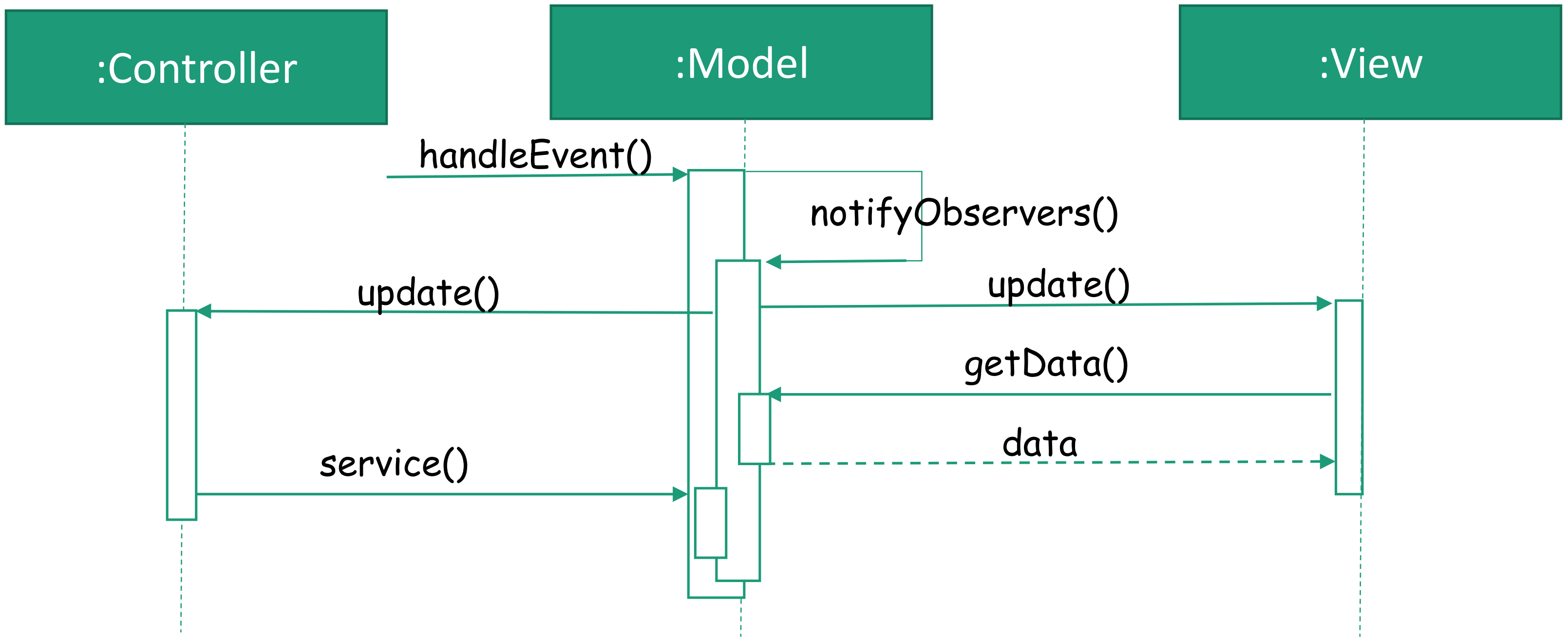
Active MVC Model

- Introduce an observer



Model and View Interaction

- Separation of concerns: code/logic are separated, but the objects interact



Active MVC Model Discussion

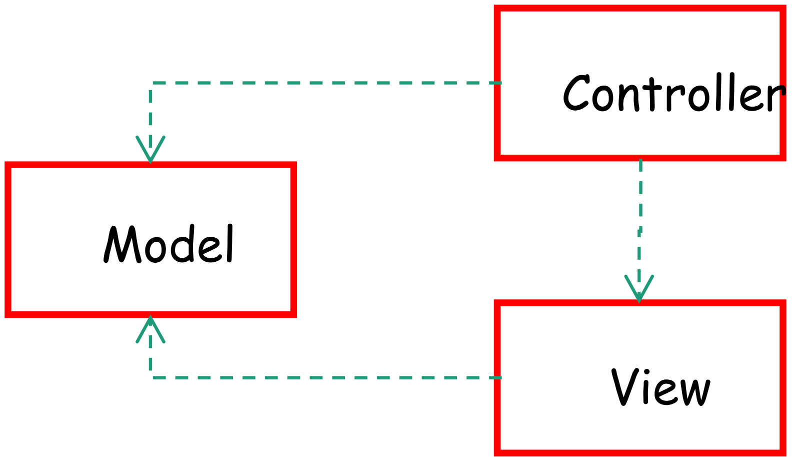
- Model is active
  - Model may change state without controller's involvement
    - e.g., in particular, when two or more sources may result in model update
- How do we separate model from view when model is active?
  - Model updates view
  - Realized via the Observer pattern
    - Model is an observable that notifies view or controller that is an observer
    - The model never requires specific information about any views
    - Controller or model implements the Observer whenever necessary
- Also called: the publish-subscribe pattern

Publish-Subscribe Pattern

- Subject: a subset of Observables in the model
- Subscriber: a subject's observer
- In an application that has multiple views, we often have multiple subjects
  - Each describe a specify type of model change
  - Each view can then subscribe only types of changes that are relevant to the view

Realizing MVC: Passive Model

- Three (categories) of classes
  - Controller class
  - Model class
  - View class



Passive MVC: Dependency Injection

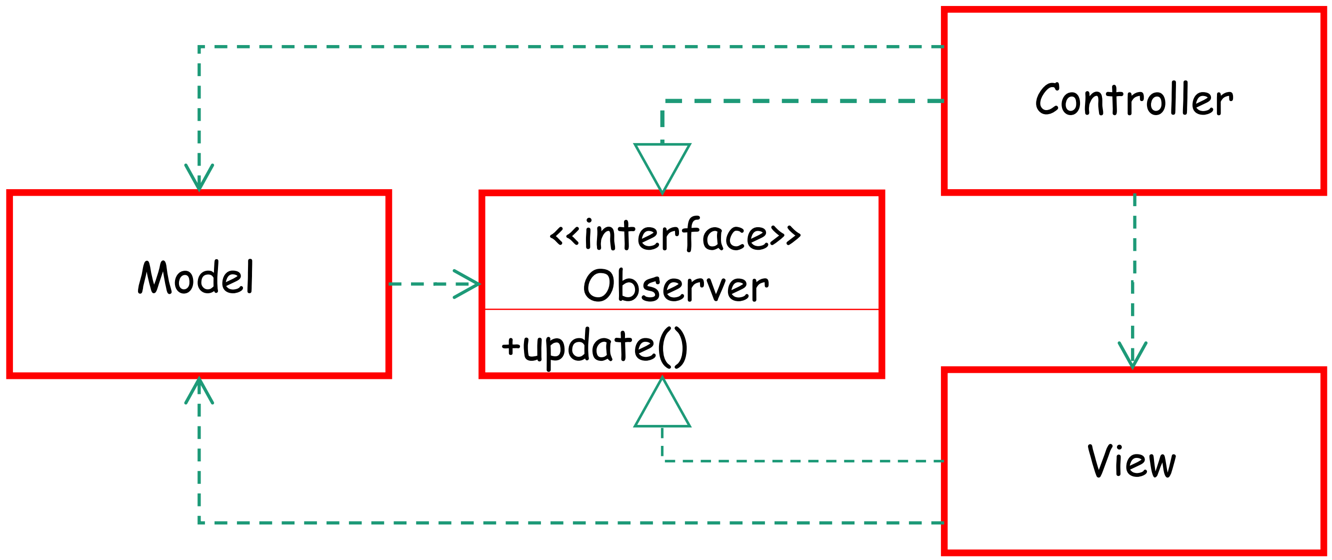
- Interpret dependency as association
- Model class is independent completely
  - No reference to either the Controller or the View class at all
- View depends on Model
  - The View class has instance variable that references to Model
- Controller depends on both View and Model
  - The Controller class has instance variables that reference to the Model and the View, respectively

Passive MVC: Dependency Injection

- Interpret dependency as a weaker dependency relationship than the association
- Model class is independently completely
    - No reference to either the Controller or the View class at all
- View depends on Model
    - The View class may not have instance variable that references to Model
    - A method of View has a parameter of the Model type (e.g., getData(Model m))
- Controller depends on both View and Model
    - The Controller class may not have instance variables that reference to either the Model or the View.
    - It has methods that requires parameters of either the Model or the View type, e.g., service(Model m), notifyUpdate(View v).

Active MVC: Publish-Subscribe

- Three (types) of classes: Model, View, and Controller
  - Model has instance variables to observables (so the Model is related to the Observer, via the platform)
  - View and controller implements the Observer interface



Active MVC: Publish-Subscribe

- Observer pattern via JavaFX Properties
- Model
  - Has instance variables references to subjects (JavaFX properties, or classes that wrap JavaFX properties)
- View and Controller
  - Has event listener either as instance method parameter or instance variables to listen to changes in Model
- Controller
  - Has references to model either as instance method parameter or instance variables (for update model)

Computer Science Quotes App

- Model (or Application domain)
  - A list of strings (computer science authors and what they said)
- View
  - The interface shows the quotes
- Controller
  - Intercept users' mouse clicks
  - Inform model (or domain) about quote to display
  - Inform view to update the quote to be displayed

Questions?

- Concept of Model-View-Controller pattern
- Concept of Publish-subscribe pattern
- Realization of MVC
  - Dependency injection
  - Observer pattern

Building View via FXML

- Help enforce the constraints imposed by the Model-View-Controller pattern
  - Separate application logic from user interface by expressing user interface in XML
    - Construct scene graph without writing code (in contrast to constructing scene graphs in procedural code)
- Some consider it a convenient way to express View in an XML file
  - The hierarchical structure of an XML document closely parallels the structure of the JavaFX scene graph.
  - It has tool support (JavaFX Scene Builder)

Tool Support: Scene Builder

- JavaFX Scene Builder 2.0
  - Oracle does not offer the binary any more
  - Source code is distributed with the OpenJFX project
  - Three options
    - Download & install from a reputable 3rd party provider
    - Down the source code, build it, and install it yourself
    - (Not recommended) Download & install JavaFX Scene Builder 1.x

Using Eclipse for JavaFX FXML Project

- If from scratch
  - Create a Maven project
  - Plan, plan, plan, plan, plan, and plan …
  - Create Controller class (always name it as a Controller)
    - Use @FXML to annotate fields and methods (injecting dependency on View to the Controller)
  - Create FXML file (View)
    - You can create it using the Scene Builder 2.0
    - Specify controller for the view
  - Create classes for your application logic (Model)
    - Either passive or active MVC
- Follow the MVC pattern, and the guideline discussed

FXML: Overview

- Starting with
  - <?xml version=*"1.0" encoding="UTF-8"?>*
- Two major parts
  - Import dependencies, e.g.,
    - <?import javafx.scene.control.Button?>
  - Scene graph starting from a root node, e.g.,
    - <GridPane> … </GridPane>
    - Insert nodes within the tag representing the root node

Specify Controller for FXML View

- At attribute and value pair to the root element of the scene graph
  - fx:controller="YourControllerClass"
- You may use fully qualified class name (including package name)

Injecting Dependency via @FXML Annotation

- Variable name in the Controller class must match the value of attribute "fx:id", e.g.,
  - In Controller
    - @FXML
    - TextField outputTextField
  - In FXML View
    - <TextField fx:id=*"outputTextField*">
- Event Handler method in the Controller class must also match value of the event handler after prefixed with a "#" sign, e.g.,
  - In Controller
    - @FXML
    - void processNumberKeys(ActionEvent event) { … }
  - In FXML View
    - <Button *onAction="#processNumberKeys*">

Questions?

- Motivation and concept of FXML
- Big picture
  - Tool to build FXML views
  - A high-level guideline for a JavaFX FXML application project

User Interface Design with FXML

- FXML
  - XML-based language
  - XML = Extensible Markup Language
- Help build a user interface separated from the application logic

Example: CS Quotes in JavaFX with FXML

- Define the Model
  - CsQuotesModel.java
- Define the View
  - fxml_mainview.fxml
- Define the Controller
  - CsQuotesController.java

Example: Instantiating the View from FXML File

- Entry Point of the Application

```
private final static String APP_TITLE = "Quotations in Computer Science";
  private final static String MAIN_VIEW_FXML = "fxml_mainview.fxml";
  @Override
  public void start(Stage primaryStage) throws IOException {
     Pane mainPane =
            (Pane)FXMLLoader.load(getClass().getResource(MAIN_VIEW_FXML));
     Scene mainScene = new Scene(mainPane);
      primaryStage.setTitle(APP_TITLE);
     primaryStage.setScene(mainScene);
     primaryStage.show();
  }
```

Questions

- Express Views in FXML
- Example application

Skin JavaFX Application with CSS

- Control appearance of JavaFX interface using Cascading Style Sheets
- Cascading Style Sheets (CSS)
  - A World-Wide-Web Consortium (W3C) standard
  - Originally designed as a simple mechanism for adding style (e.g., fonts, colors, spacing) to Web documents
  - See https://www.w3.org/Style/CSS/
  - CSS level 1, 2, and 3 (some still under development)
- JavaFX CSS (JavaFX 8)
  - Based on W3C CSS level 2.1 with some addition on current work on CSS level 3
  - Aimed at providing a uniform method to style both desktop and web applications

An Example of JavaFX CSS

Selector

```
.root {
    -fx-font-size: 16pt;
    -fx-font-family: "Courier New";
    -fx-base: rgb(132, 145, 47);
    -fx-background: rgb(225, 228, 203);
    -fx-background-image: url("background.jpg");
    -fx-background-repeat: no-repeat;
    -fx-background-size: cover;
}
```

Styles in {}

A style is written as a property and value pair, and the property name and its value is separated by a ":", and ended with a ";".

JavaFX property names are prefixed with a vendor extension of "-fx-".

Apply Styles

- Styles are applied (but not necessarily selected for) to Nodes in the Scene graph
  - First applied to the parent, then to its children
- A node is styled after it is added to the scene graph.
- A node is re-styled
  - when the following changes made to the node's pseudo-class state, style-class, id, inline style, or parent
    - Pseudo-class state: e.g., MouseEvent.MOUSE_ENTERED
  - When stylesheets are added to or removed from the scene.

CSS Selectors

- CSS selectors are used to match styles to scene-graph nodes
  - Type selector
  - Class selector
  - ID selector
  - Context selector

Type Selector

- Select based on type name returned by Node's getTypeSelector method
- Analogous to a CSS type selector
- See style and code example in
  - CssDemoFX

Class Selector

- Select based on the value of the styleClass property of the Node
  - A Node can have multiple style classes
- Analogous to a CSS class selector
- See style and code example in
  - CssDemoFX

ID Selector

- Select based on the ID of the Node
  - The ID of a Node can be set using Node's setId method
  - ID is should be unique
- Analogous to a CSS ID selector
- See style and code example in
  - CssDemoFX

Context Selector

- Selection based on contextual information
- Example:
  - #brooklyn-orange-next-quote Text { … }
  - matches a Node whose type name is "Text" and the Node is a descendent of the Node whose ID is #brooklyn-orange-next-quote
  - See CSS 3 Selectors for more
    - https://www.w3.org/TR/css3-selectors/

Grow your skills & knowledge

- CISC 3620 Computer Graphics
  - 2D and 3D graphics
- CISC 3320 Operating Systems
  - Concurrency, processes, and threads

Assignments

- Practice