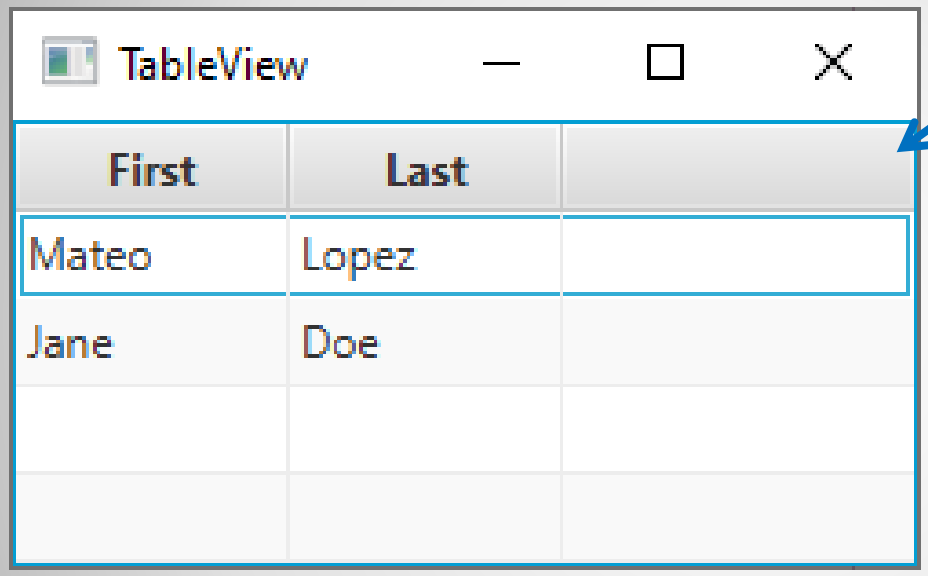
Java Programming

Arthur Hoskey, Ph.D.
Farmingdale State College
Computer Systems Department

Today's Lecture

- JavaFX - TableView

- Display rows and columns of data.



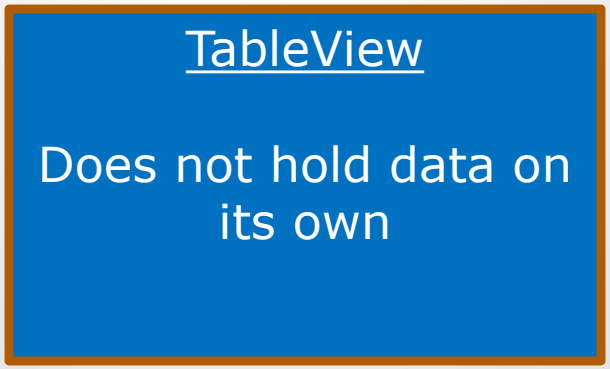
TableView

In Scene Builder you can drag and drop a TableView control on to the window

TableView

TableView

- Component used to display rows and columns of data.
- Does not inherently hold data itself.
- Uses another class to actually hold its data.



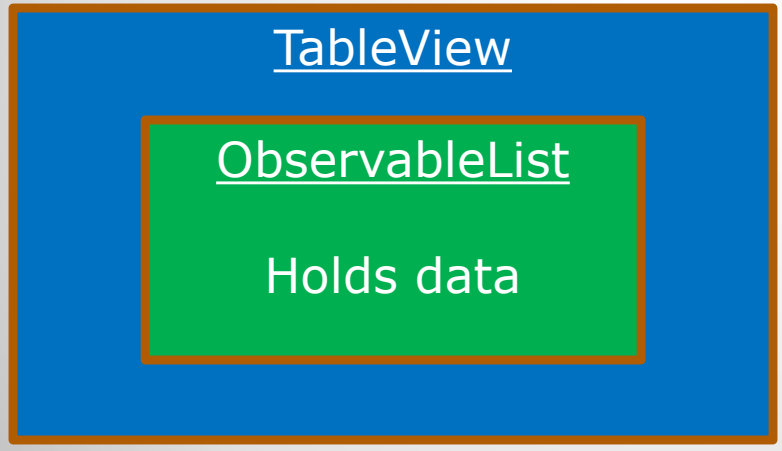
TableView

ObservableList

- Used by TableView to store data.
- Just a collection of data.
- Does not display data on its own (it is a member of the TableView class).

ObservableList

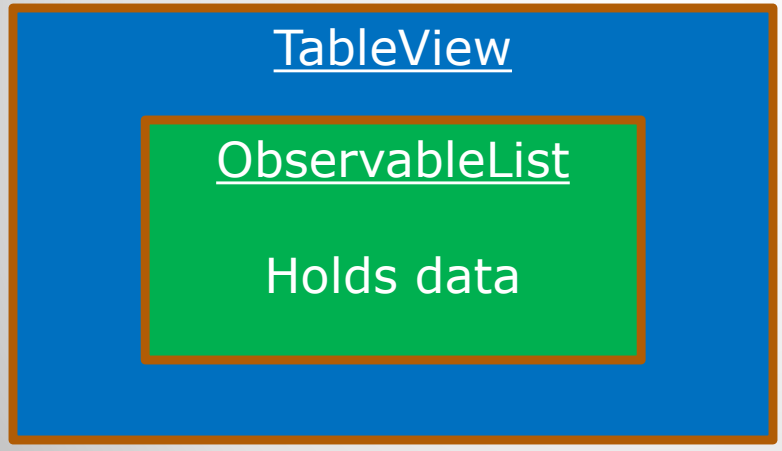
- TableView needs to be associated with an ObservableList.
- The ObservableList will hold the data for the TableView.
- TableView "has an" ObservableList (aggregation relationship).



How do we associate a ObservableList with a TableView?

TableView "has a" ObservableList

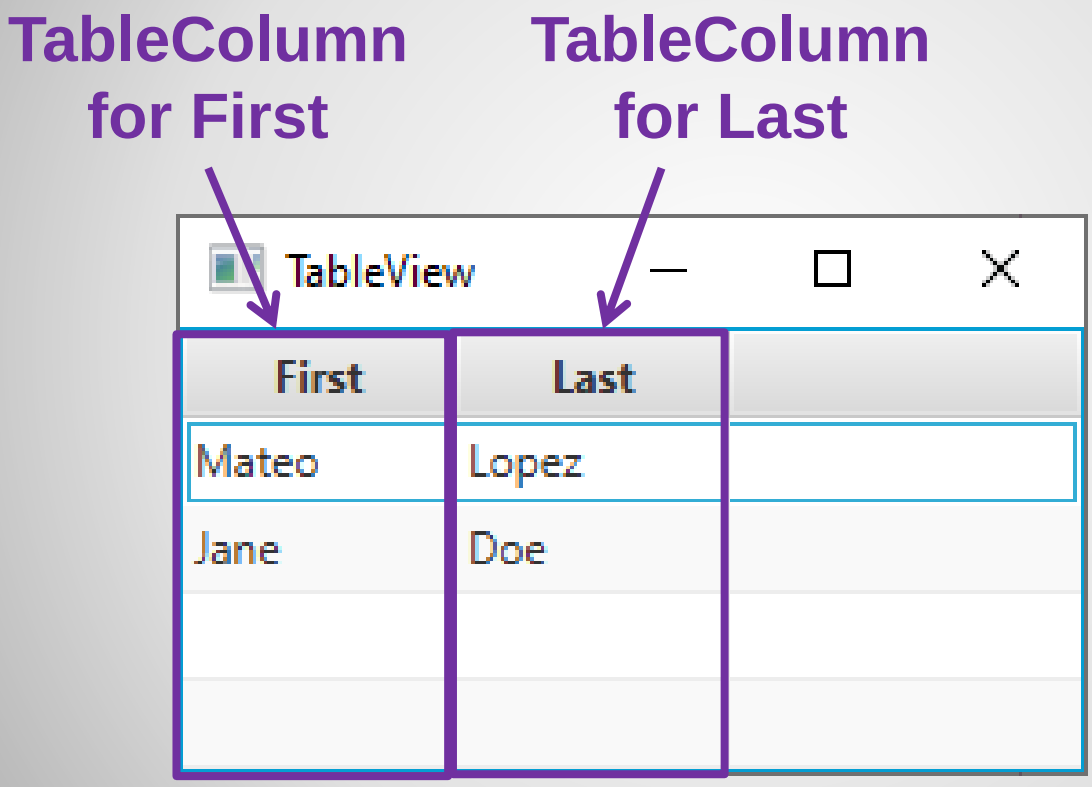
- There is an Items member inside of the TableView.
- This Items member stores data that the TableView displays.
- For example:



The ObservableList has the TableView's data

TableView "has an" ObservableList

- A TableView contains multiple TableColumns.



TableColumn

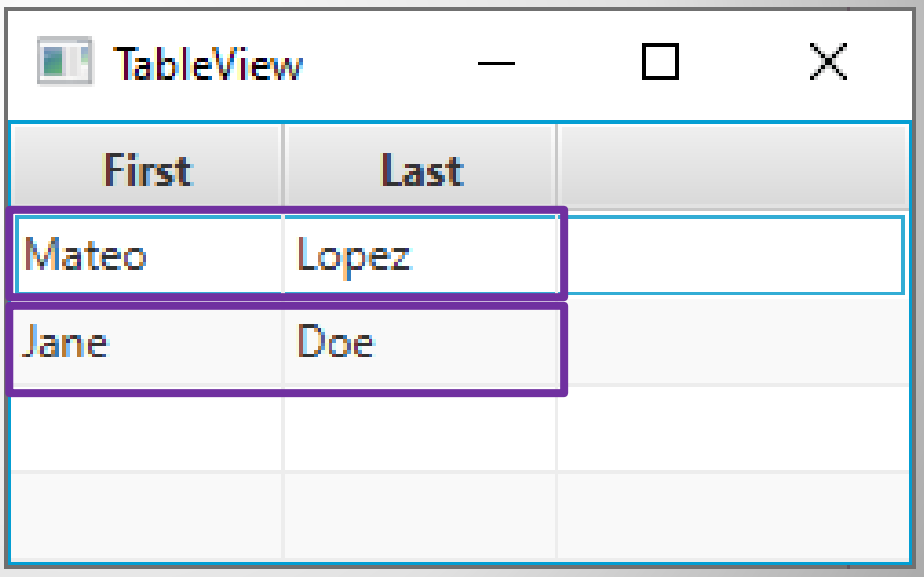
TableView Row Data

- TableView row data is stored in class instances.
- You must create a class that will be used to store row data.
- For example, a Person class with first and last name can be used by this TableView.
- The TableView will have an ObservableList of the class that stores the row data (ObservableList<Person> in this example).

Person Class

```
public class Person {
    private String first;
    private String last;
    // other code here…
}
```

Each row has its own instance of Person



TableView Setup Overview

1. Create a class that will represent each row being displayed in the TableView. Do this in code.
2. Add TableView to layout. Do this in Scene Builder.
3. Add columns to the TableView. Do this in Scene Builder.
4. Add member variables for the TableView and each TableColumn to the controller class (most likely PrimaryController).
5. Set each columns cell value factory. Do this in code.
6. Add data to the TableView.

TableView Setup Overview

1. Create Class for Rows

- Create a new class that will be used to store data for one row.
- For example:

```
public class Person {
    private String first;
    private String last;

    public Person(String first, String last) {
        this.first = first;
        this.last = last;
    }

    public String getFirst() { return first; }
    public void setFirst(String first) { this.first = first; }

    public String getLast() { return last; }
    public void setLast(String last) { this.last = last; }
}
```

Create Class for Rows

2. Add TableView

- Add a TableView to the layout in Scene Builder.
- TableViews are in the Controls section (left side) in Scene Builder.
- Set the fx:id of the TableView.

Add TableView

3. Add Columns to TableView

- Do this in Scene Builder.
- TableColumns are in the Controls section (left side) in Scene Builder.
- Set TableColumn header for each column by setting the value of its Text property.
- Set the fx:id of each TableColumn.

Add TableColumns

4. Add Member Variables for TableView and TableColumns

- Do this in code (in the controller class).
- For example:

```
@FXML
private TableView tableViewPersons;
```


```
@FXML
private TableColumn<Person, String> tableColumnFirst;

@FXML
private TableColumn<Person, String> tableColumnLast;
```

Add TableColumns

5. Set TableColumn Cell Factories (associate columns and vars)

- Do this in code (in the controller class).
- This associates a field on the class to a TableColumn.
- The best place to put this code is in the controller's initialize method.
- For example:

Associate tableColumnFirst with the "first" field on the Person class. This will cause it to use setFirst/getFirst on the Person instance to get/set data for that column.

```
public void initialize() {
    tableColumnFirst.setCellValueFactory(
      new PropertyValueFactory<Person,String>("first"));

    tableColumnLast.setCellValueFactory(
      new PropertyValueFactory<Person,String>("last"));

    // other code goes here…
}
```

Set TableColumn Cell Factories

6. Add Data to the TableView

- Do this in code.
- Get the ObservableList associated with the TableView.
- Add instances of the row class to the ObservableList (Person in this example). For example:

Get the ObservableList from the TableView

```
ObservableList<Person> olPersons = tableViewPersons.getItems();

Person p1 = new Person("Mateo", "Lopez");
olPersons.add(p1);

Person p2 = new Person("Jane", "Doe");
olPersons.add(p2);
```

Add Data to the TableView

- Handle TableView Item Selection (controller code)

```
@FXML
private TableView<Person> tableViewPersons;

@FXML
protected void handleTableViewItemsMouseClick(MouseEvent event) {

   ObservableList<Person> observableList =
      (ObservableList<Person>) tableViewPersons.getItems();


   int selectedIndex = tableViewPersons.getSelectionModel().getSelectedIndex();


   if (selectedIndex >= 0 && selectedIndex < observableList.size()) {
      System.out.println("Selected Item: " + observableList.get(selectedIndex));
   }
}
```

TableView FXML Id

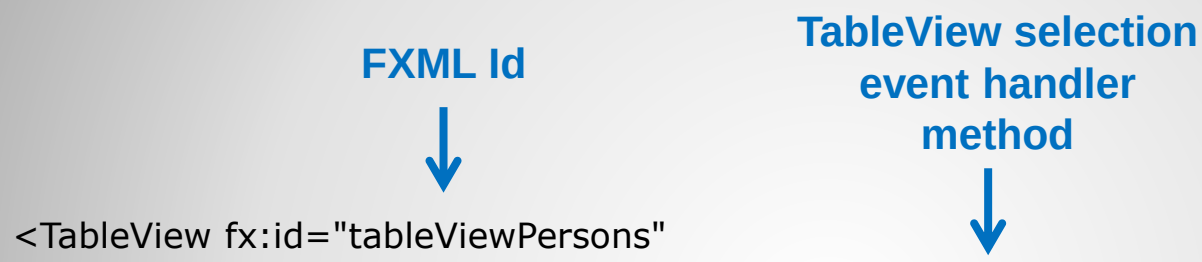
Get the background collection

Get the selected item index

Get the selected item from the collection

TableView Selected Item

- Handle TableView Item Selection (FXML code)



TableView Selected Item

End of Slides