CSC 243 - Java Programming

JavaFX Event Driven Programming

Events

- An event is an object created from an event source
- An event object contains the properties relevant to the event
- An event handler object is an object that is registered with an event source object to process event objects
- Events are fired when some relevant action is performed, such as pressing a button

Basic JavaFX Event Handling

- For an object to be an event handler, the following conditions must be met:
  1. The object must implement the `EventHandler<T extends Event>` interface
  2. The `EventHandler` object must be registered with the event source object using the method `setOnAction`

Some JavaFX User Actions

Action	**Source Object**	**Event**	**Registration Method**
Click a Button	`Button`	`ActionEvent`	`setOnAction`
Submit Text	`TextField`	`ActionEvent`	`setOnAction`
Mouse Clicked	`Node, Scene`	`MouseEvent`	`setOnMouseClicked`
Mouse Pressed	`Node, Scene`	`MouseEvent`	`setOnMousePressed`
Mouse Released	`Node, Scene`	`MouseEvent`	`setOnMouseReleased`
Mouse Dragged	`Node, Scene`	`MouseEvent`	`setOnMouseDragged`
Key Pressed	`Node, Scene`	`KeyEvent`	`setOnKeyPressed`
Key Released	`Node, Scene`	`KeyEvent`	`setOnKeyReleased`
Key Typed	`Node, Scene`	`KeyEvent`	`setOnKeyTyped`

Convenient Syntax for Event Handlers

- Inner Classes
- Anonymous Inner Classes
- Lambda Expressions

Inner Class Example

```
public void start(Stage primaryStage) {
    // ...

    button.setOnAction(new ButtonHandler);
}

class ButtonHandler
        implements EventHandler<ActionEvent> {
    public void handle(ActionEvent e) {
        // handle event
    }
}
```

Anonymous Inner Class Example

```
public void start(Stage primaryStage) {
    // ...

    button.setOnAction(
        new EventHandler<ActionEvent>() {
            public void handle(ActionEvent e) {
                // handle event
            }
        }
    );
}
```

Anonymous Inner Class Rules

- An anonymous inner class must always extend a superclass or implement an interface, but cannot have an explicit `extends` or `implements` clause
- An anonymous inner class must implement all of the abstract methods
- An anonymous inner class uses the zero argument constructor of the superclass

Lambda Expression Example

```
public void start(Stage primaryStage) {
    // ...

    button.setOnAction(e -> {
        // handle event
    });
}
```