COP 2251 – Chapter 15 – Event-Driven Programming & Animations (JavaFX)

Introduction

* User interaction with an object (the event source object) in a GUI program generates an event.
* The handle method of an event handler object is coded to respond to the event, but the handler must first have been registered with the event source object.

Requirements for an event handler

* The event handler must be an instance of interface **EventHandler<T extends Event>**.
* The handler for an ActionEvent is registered with the event source object using the **source.setOnAction(handler)**.

Try HandleEvent from the zipped examples.

* Note the two inner classes that implement the EventHandler interface by coding the handle() method.
* Here is the code that creates the handlers and registers them with the buttons:

OKHandlerClass handler1 = **new** OKHandlerClass();

btOK.setOnAction(handler1);

CancelHandlerClass handler2 = **new** CancelHandlerClass();

btCancel.setOnAction(handler2);

Events and Event Sources

* Events are objects created from event source objects.
* The **EventObject** class is the parent class for event subclasses. See Figure 15.4 on page 596.
* The event object’s **getSource()** method will return the name of the event source object.
* See Table 15.1 on page 597 for the types of events fired by user actions with GUI objects.
* Note how actions and events are named in the table below:

|  |  |  |  |
| --- | --- | --- | --- |
| User Action | Source Object | Event Type | Event Registered By |
| Button Click | Button,CheckBox, RadioButton | **Action**Event | setOn**Action**(EventHandler<**Action**Event>) |
| Mouse **Entered** | Any Node | MouseEvent | setOn**MouseEntered**(EventHandler<**Mouse**Event>) |
| Key **Pressed** | Any Node | **Key**Event | setOn**KeyPressed**(EventHandler<**Key**Event>) |

Registering Handlers and Handling Events

* An event handler object must be registered by name with the event source object.
* The handler object must be an instance of the correct interface type to ensure that it contains a method that can handle the event.
* See Figure 15.5.

Try ControlCircleWithoutEventHandling from the zipped examples.

Try ControlCircle from the zipped examples. Note that the example contains a class CirclePane and that an instance variable of it named circlePane is defined in line 15, making it referenceable anywhere in class ControlCircle.

Inner Classes

* An inner class is a class defined inside another class. **It cannot be public**.
* Inner classes are often used as event handlers and helper classes.
* ControlCircle has several inner classes. Class EnlargeHandler was used as an event handler.
* See the features of inner classes in the bullet list on page 602.
* Inner classes are also useful for reducing the number of source code files in a project and for avoiding naming conflicts.

Anonymous Inner Class Handlers

* See an example in the graphic at the top of page 603. Coding is simplified.
* Note that instead of creating a named instance that implements EventHandler, an anonymous inner class is created by running the EventHandler constructor for the desired event.
* See the bullet list on page 603 for the features of anonymous inner classes.

Try AnonymousHandlerDemo from the zipped examples. It creates four anonymous inner classes as handlers for the four buttons.

**Exercise**: Add an anonymous handler to output “Entered New” when the mouse is moved over New.

Simplifying Event Handling Using Lambda Expressions

* This feature introduced in Java 8 can simplify event handling in your programs.
* Compare parts (a) and (b) at the bottom of page 605
* Enclosing parentheses can be omitted if there is only one parameter and no explicit data type.
* A lambda expression is treated by the compiler like an object from an anonymous inner class.

e -> {

…

}

* In the code above, the compiler understands that **e** is an **xEvent** (ActionEvent or MouseEvent or KeyEvent) parameter and that the statements inside the braces are for the **handle** method.
* There can be **only one** abstract method in the interface. This is known as a **functional interface**, or **Single Abstract Method (SAM)**.
* Using lambda expressions for event handlers allows for shorter, cleaner, and clearer code.

Try LambdaHandlerDemo from the zipped examples. Note that line 37 illustrates the simplest syntax for a lambda function…no braces and no internal semicolon.

**Exercise**: Add a lambda expression to output “Moused over Print” when the mouse enters Print.

Case Study: Loan Calculator

* NOTE: this example requires that Loan.java be added to the project. It was.

Try LoanCalculator from the zipped examples. Note the lambda expression in line 51 calls a method that is defined later in the program.

Mouse Events

* A **MouseEvent** object is created when the mouse is moved, pressed, dragged, clicked, or released over a node or scene.
* See the methods of MouseEvent in the UML diagram in Figure 15.10 on page 611.
* Four constants of MouseButton identify the individual mouse buttons:
  + PRIMARY, SECONDARY, MIDDLE, and NONE
* The **getButton()** method can detect which button was pressed.

Try MouseEventDemo (try dragging the text node) from the zipped examples.

**Exercise**: Add a line to the lambda to display the X and Y coordinates of the text in the console.

Key Events

* Pressing, releasing a key on a node or scene fires a **KeyEvent**.
* See the methods of KeyEvent in the UML diagram in Figure 15.12 on page 613.
* The KeyEvent **getCode()** method returns a key code constant defined in the **enum** **KeyCode**.
* See the KeyCode constants in Table 15.2 on page 613.

Try KeyEventDemo from the zipped examples.

Try a new lower case letter and use the cursor keys to move it.

Note that the switch cases refer to the KeyCode class constants for the cursor keys.

Note that the focus must be on the character and this is achieved by line 35.

Try ControlCircleWithMouseAndKey (try up / down cursor keys, left / right mouse buttons) from the zipped examples.

Listeners for Observable Objects

* An observable object is an instance of class **Observable**.
* Its **addListener()** method can specify a listener to listen for a value change in the object.
* The listener class must implement the **InvalidatonListener** interface by overriding its **invalidated()** method.
* Every binding property is an instance of Observable.

Try ObservablePropertyDemo from the zipped examples.

**Exercise**: Comment out the lines 11-16 and add a lambda expression as on 617. Display the output of ov.toString(), too.

Try DisplayResizableClock (try resizing the window) from the zipped examples. Note that clock width and height are made Observable objects by being bound to the dimensions of the pane.

Animation

* JavaFX provides the abstract class **Animation** for …um, animations.
* See the UML diagram about class Animation in Figure 15.16 on page 619.
* The paragraph immediately below this UML diagram explains the Animation class very well.

PathTransition

* This subtype of Animation moves a node along a path over a time defined by class **Duration**.
* The duration can be specified in milliseconds (ms).

Try PathTransitionDemo from the zipped examples. Pressing and releasing the mouse over the circle will pause and resume the animation, respectively.

Try FlagRisingAnimation from the zipped examples. Try a different value for Duration.millis().

FadeTransition

* This subclass of animation animates a change in opacity in a node over time.
* See the UML diagram in Figure 15.19.

Try FadeTransitionDemo from the zipped examples. Reset line 33 to ft.setToValue(0.**0**);

Timeline

* This subclass of Animation can execute any animation in KeyFrame intervals.

Try TimelineDemo from the zipped examples. Try Duration.millis(**5000**).

Try ClockAnimation from the zipped examples.

Case Study: Bouncing Ball

* Examine classes BallPane and BounceBallControl on pages 627-629.

Try BounceBallControl from the zipped examples. Use cursor up to increase the ball speed and cursor down to slow it down.

Try Example2.java and Example3.java.

The **animate** package has additional examples that might interest you. Try the them all.