Object-oriented programming

Session 9

10th of April 2019
Ljubov Jaanuska
Last time

• JavaFX

• Created
  – stage
  – scene
    • scene graph
      – root
Grouped buttons

VBox root = new VBox();
RadioButton b1 = new RadioButton("First button");
b1.setSelected(true);
RadioButton b2 = new RadioButton("Second button");

    ToggleGroup bGroup = new ToggleGroup();
    b1.setToggleGroup(bGroup);
    b2.setToggleGroup(bGroup);

root.getChildren().add(b1);
root.getChildren().add(b2);
Scene scene = new Scene(root, 300, 300);
primaryStage.setTitle("Radio buttons");
primaryStage.setScene(scene);
primaryStage.show();
Grouped buttons

• Radio buttons
  – Class RadioButton
  

• Group
  – Class ToggleGroup
  
Events

Button → Event → Handler
Event

• Steps
  – something happens
  – inform the object
  – react – handle the event (*handling*)
Event

• Class Event (or its subclass) are in classes
  – AWT
  – Swing
  – JavaFX
Event

• `javafx.event.Event`
  – many subclasses

• Constructor parameters
  – event type
  – source
  – target
Event types

https://docs.oracle.com/javase/8/javafx/events-tutorial/processing.htm
Event type: MouseEvent

`EventType<MouseEvent>`

<table>
<thead>
<tr>
<th>Modifier and Type</th>
<th>Field and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static <code>EventType&lt;MouseEvent&gt;</code></td>
<td>ANY</td>
</tr>
<tr>
<td>static <code>EventType&lt;MouseEvent&gt;</code></td>
<td>DRAG_DETected</td>
</tr>
<tr>
<td>static <code>EventType&lt;MouseEvent&gt;</code></td>
<td>MOUSE_CLICKED</td>
</tr>
<tr>
<td>static <code>EventType&lt;MouseEvent&gt;</code></td>
<td>MOUSE_DRAGGED</td>
</tr>
<tr>
<td>static <code>EventType&lt;MouseEvent&gt;</code></td>
<td>MOUSE_ENTERED</td>
</tr>
</tbody>
</table>

https://openjfx.io/javadoc/11/javafx.base/javafx/event/EventType.html
https://openjfx.io/javadoc/11/javafx.graphics/javafx/scene/input/MouseEvent.html
r1.addEventListener(KeyEvent.KEY_PRESSED, eventHandler)
Event handler

- Event handler
  - interface `EventHandler`
    - method `handle`

- Event handler is added to a node

Handler in a separate class

```java
public class MyHandler implements EventHandler<MouseEvent> {

    public void handle(MouseEvent event) {
        System.out.println("Target" + event.getTarget());
        System.out.println("Source" + event.getSource());
        System.out.println("Handler");
    }
}
```
// create an instance
EventHandler<MouseEvent> myHandler =
    new MyHandler();

// apply handler
r1.addEventHandler(MouseEvent.MOUSE_ENTERED, myHandler);
Convenience methods

`setOnEvent-type(EventHandler<event-class> value)`

– Makes an event handler creation shorter
  • `setOnKeyPressed(eventHandler)`
  • `setOnMouseEntered(eventHandler)`
  • `setOnAction(eventHandler)`

• Only for the most common events
• Described in class `Node`

https://docs.oracle.com/javase/8/javafx/events-tutorial/convenience_methods.htm
Add handler

```java
EventHandler<MouseEvent> myHandler = new MyHandler();

// convenience method
r1.setOnMouseEntered(myHandler);
```

- No need to specify the event type
- Event handler
public class Test extends Application {
    public void start(Stage primaryStage) {

        rings.getChildren().add(r1);

        class MyHandler implements EventHandler<MouseEvent> {
            public void handle(MouseEvent event) {
                System.out.println("Handler inside the class");
            }
        }

        EventHandler<MouseEvent> myHandler = new MyHandler();
        r1.setOnMouseEntered(myHandler);

        ...
    }
}
EventHandler<MouseEvent> myHandler =
    new EventHandler<MouseEvent>(){

        public void handle(MouseEvent event){
            System.out.println("Anonymous handler");
        }
    };

r1.setOnMouseEntered(myHandler);
Instance without a name

r1.setOnMouseEntered(

    new EventHandler<MouseEvent>() {

        public void handle(MouseEvent event) {
            System.out.println("Target" + event.getTarget());
            System.out.println("Source" + event.getSource());
            System.out.println("Anonymous handler without a name");
        }
    }

);
Different opportunities

• Three steps
  – Handler class is
    • a separate class
    • an inner class
    • an anonymous inner class
  – Handler instance
    • with a name
    • without a name
  – Bind with a node
    • `r1.addEventHandler(MouseEvent.MOUSE_ENTERED, handler);`
    • convenience methods
      `r1.setOnMouseEntered(handler);`
One more possibility: lambda

```java
r1.setOnMouseEntered((event) ->
    r1.setFill(Color.CHARTREUSE));
```

```java
r1.setOnMouseEntered(e ->
    {System.out.println("Handler, lambda");
        r1.setFill(Color.CRIMSON);}
);
```
Add events

r1.setOnMouseExited(
    new EventHandler<MouseEventArgs>(){
        public void handle(MouseEventArgs event) {
            System.out.println("Target" +
                event.getTarget());
            System.out.println("Source" +
                event.getSource());
            System.out.println(event);
            System.out.println("Handler");
            if (r1.setFill() == Color.Crimson)
                r1.setFill(Color.DeepSkyBlue);
            else
                r1.setFill(Color.Firebrick);
        }
    });
Grouped buttons

r2.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        Label lbl = new Label("Correct answer!");
        root.getChildren().add(lbl);
    }
});
Event for the scene

```
scene.addEventHandler(MouseEvent.ANY, new EventHandler<MouseEvent>() {
    public void handle(MouseEvent event) {
        System.out.println("Handler");
        rec.setFill(Color.PERU);
    }
});
```
Event for the stage

```java
primaryStage.addEventHandler(MouseEvent.ANY,
    new EventHandler<MouseEvent>(){
        public void handle(MouseEvent event){
            System.out.println(event);
            System.out.println("Handler");
            rec.setFill(Color.PERU);
        }
    });
```
Listeners

• InterfaceChangeListener
  – method changed