**GUI exercise in Java**

Søren Lauesen 04-12-2015

Write a Java program that shows this Form (*Stage* in Javanese):



There is a button labeled *Move 10 down* and an array of circles, alternating blue and red.

* When you click a circle, it will print on System.out: *I am 0* (or 1, 2 . . .)
* When you click the button, circle 0 will move 10 pixels down.

Hints

1. With NetBeans, create a project of type JavaFX.
2. The JavaFX project has a *start* method that contains code for creating a *scene*, a root *StackPane* and a button attached to the pane. Try to run it. It should show a Form with a Hello World button at the center. Try to click it. If it doesn't print anything on *out*, there is something wrong with your Java/Netbeans/JDK installation.
3. Change the StackPane to a simple Pane. **Try it.** It makes the button move to the top-left corner. Why? The simple Pane allows you to define the X-Y positions of the visual components. The default X-Y of the Button is 0, 0. So it moves to the top-left.
4. Now declare an array of 5 Circle shapes.   
   **Warning:** NetBeans can suggest that you import a Circle file. Do so, but make sure you import a javaFX file, not a javaAWT.
5. Make a *for( j=0 . . .)* loop. Inside the loop, create a Circle shape in the same way as the button is created. Let the array element refer to the circle. Add the circle to the root pane with  
    getChildren().add  
   in the same way as the button.
6. Set the centerX and centerY position of the Circles. The Y position is constant and the X position is calculated based on *j*. You can either set the position with setters (see below) or use a circle constructor that has centerX and centerY as parameters.
7. Try it out. If you don't see any circles, it may be because you have imported a javaAWT file rather than a javaFX file. The javaAWT file doesn't show anything in a JavaFX project. It may also be because you set the position wrongly or forgot to add the circle to the *root.*
8. Set the border color and fill color of the circles (see setters below). The fill color must depend on *j*.
9. Change the event handler of the Button so that it doesn't print anything, but increases the Y-pos of the first circle using the getters and setters for CenterY. **Try it out**.

Now attach an event handler to each Circle. In lambda notation it will look like this:  
  
 circle.setOnMouseClicked( e -> { // Attach the event handler.

System.out.println("I am " + ???);

});

1. But how does the circle know its own number? You cannot just write *j*. At run time the event handler has no access to *j*. The for loop was executed long ago. The trick is to use a field in Circle called UserData. It can be any object or simple value. We can save the *j* value there and the Circle's event handler can find its own *j* there. The code looks like this:

circle.setUserData(j); // Allow the event handler to get its own *j*.

circle.setOnMouseClicked(e -> { // Attach the event handler.

System.out.println("I am " + circle.getUserData()); // Find the *j* in UserData.

});

}

1. When creating a Circle or another Shape, you will need to set various fields/properties of the Shape. There are a lot to choose from. Unfortunately, few of them work for any Shape. The most important ones are:

setFill( Color. ... ) defines the inner color of most Shapes.

setStroke(Color. ...) defines the border color of most Shapes.  
 setCenterX( ), setCenterY( ), setRadius( ) define position and property for a circle.  
 setX( ), setY( ), setWidth( ), setHeight( ) define the position and size of many Shapes.

setTranslateX(), setTranslateY() define the position for a Button.

1. **Warning:** We have seen cases where a Shape is inserted correctly in the getChildren list, but doesn't show on the Form. It helped using setStroke( ... ) or for texts, making the text non-null.
2. **Warning:** Sometimes NetBeans behaves strangely. It often helps to right-click the project icon and use Clean and Build.