# SYSC 2004
# Winter 2018
# Lab 9

## Objectives

- First Experience with JavaFX GUIs, including basic layouts and event handling
- Increased independence in program development: There are no step-by-step instructions; problem-solving is part of the implied work.

Submission Requirements: (Exact names are required by the submit program)

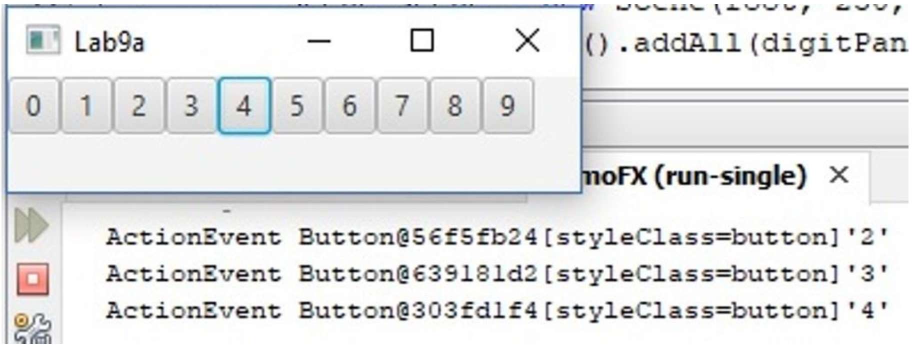
- **Without a submission you will not get a mark:** Lab9a.java, Lab9b.java

### Part A – First Simple GUI

You are to create the simple GUI application shown below: A simple horizontal array of 10 buttons representing the ten digits from zero to nine. When a button is pressed, the application shall print out the digit that was pressed (See sample output below).

- Your code should correspond to the UML class diagram given below. Notice that there are **no** instance variables. All the variables that you use should be local variables within the `start`() method.
- The `EventHandler` (called **Lab9aHandler**) should be implemented as a **non-public** class, inside the same file as **Lab9a**. In this way, you only have one file to submit for this part.

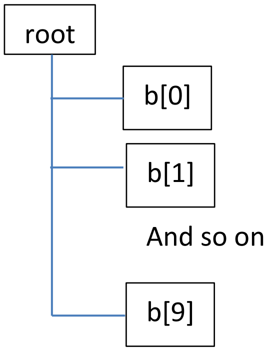
Please use the examples in the lecture slides to complete this part of the lab.

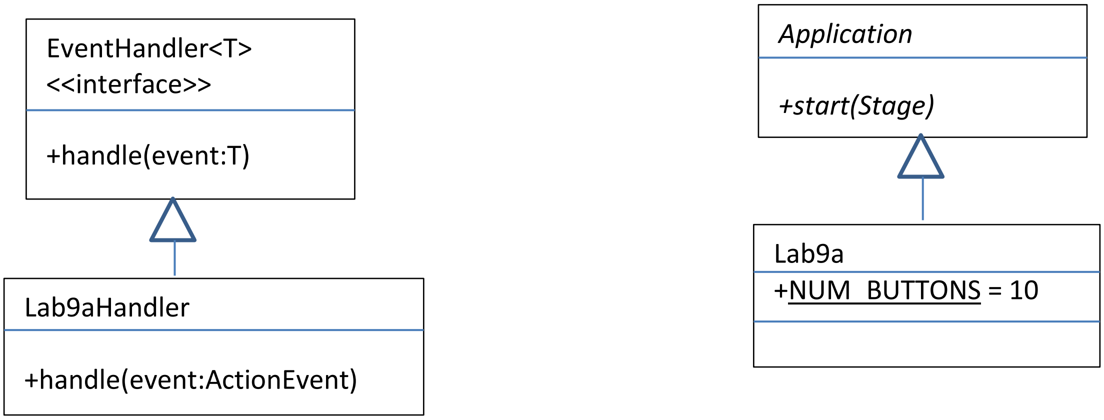


```
().addAll(digitPan
```

```
ActionEvent Button@56f5fb24[styleClass=button]'2'
ActionEvent Button@639181d2[styleClass=button]'3'
ActionEvent Button@303fd1f4[styleClass=button]'4'
```

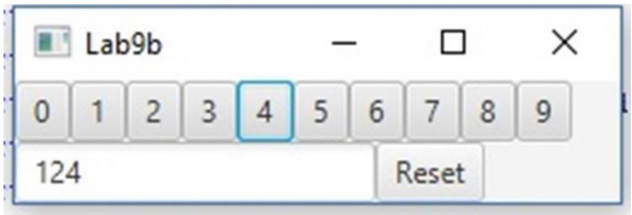
Scene Graph





## Part B – Second Complicated GUI

You are going to enlarge your GUI so that it now looks like the image below.

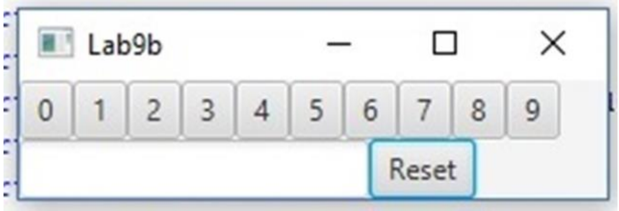


124

Underneath your original row of buttons, there is a second row containing two controls: a `Textfield` and a **Reset** `Button`. As a user presses buttons in the upper row, the button’s digit should be added to the growing `number` displayed in the `Textfield`. In the example shown, three digits have been pressed: ‘1’ then 2’ and now ‘4’.

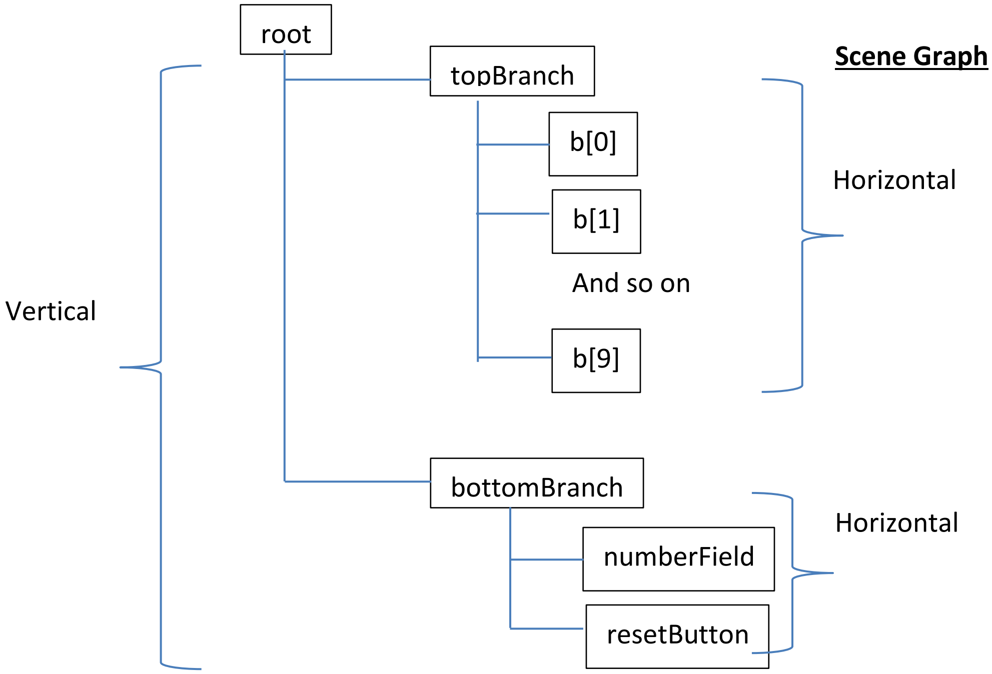
- The `Textfield` should be made `noneditable`, meaning that it will display text but a user cannot type in the box. (See JavaFX API Button: `setNonEditable() method`)

The **Reset** `Button` will clear the **number** `Textfield`, when pressed; shown below.



There are two challenges in completing this portion of the lab

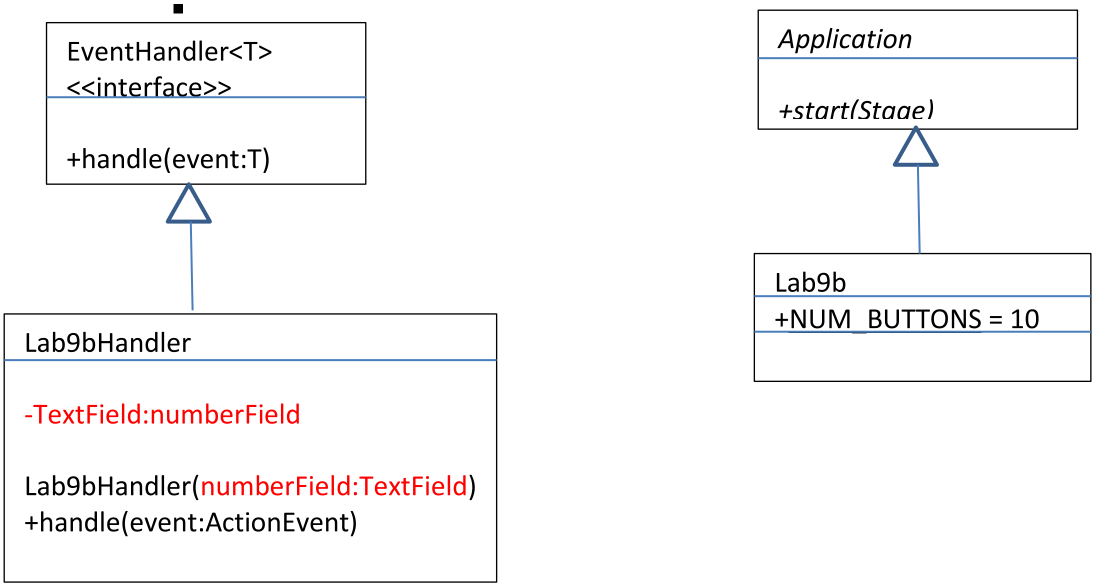
1. **Layout** – The scene graph needs to make use of two container-branches. The top branch contains your original layout of 10 buttons. The bottom branch contains our two new controls. The top and bottom branch are arranged vertically.



## 2. Sharing Object References between the Application and the EventHandler

The code for the `EventHandler` (**Lab9bHandler**) becomes more complicated in this example because:

1. The `handle`() method  must distinguish between presses of the buttons for the digits versus presses of the **Reset** button.  Different actions are needed for these two sets of buttons.
    - One solution is to use the label of the button to distinguish between multiple possible sources of a button-press.
        - The `handle`() method's argument called `event` inherits a method called `getSource`() which returns a reference to the control object which was clickedOn.
        - In our program, only buttons are clickable, so the originating control object must/will be a Button object.
        - The `Button` class provides a method called `getText`() which returns the label of a button object.
2. The `handle`() method must change the text within the **number** `TextField` (to either add a digit or to clear it completely).  Doing so involves invoking methods on the number `TextField's` object which, in turn, means that the `handle`() method needs this object's reference. Yet this object reference is in the `Application` class (**Lab9a**) not in the `EventHandler` class (**Lab9bHandler**). The design solution is:
    - There is a new constructor in `Lab9bHandler` that requires an argument.  This argument will be used to initialize the instance variable `numberField`.
    - The `start`() method in `Lab9b` will first create the number `TextField` (temporarily storing its reference in a local variable) and then create the `Lab9bHandler` object, passing in the object reference.
    - The `handle`() method in `Lab9bHandler` will use the instance variable to alter the text displayed in the `numberField`.
        - Example: `String existingText = numberField.getText();`
        - Example: `numberField.setText( "new Value" );`
        - Example: `numberField.clear();`
        -


EventHandler<T>
<<interface>>
+handle(event:T)
Lab9bHandler
-TextField:numberField
Lab9bHandler(numberField:TextField)
+handle(event:ActionEvent)
Application
+start(Stage)
Lab9b
+NUM_BUTTONS = 10