Activity 3-1

Toss a Die

1. Create a new Java project and a new class ***TestDie*** with ***main()***.
2. Create a new ***class Die***, in this same Java file. It has,
   - a data field ***maxFace***
   - a constructor with an ***int*** parameter to set the value of ***maxFace*** to the input value
   - a method ***roll*** which does not require parameter but randomly generates and returns an integer from ***1 to maxFace***
3. In ***main(),***
   - create a ***Die*** object with ***`maxFace = 6`***
   - call the ***roll*** method of the object to get a random number and ***print*** it out.

Activity 3-2

CAD

1. Design a class named **TwoDPoint** to represent a point with x and y coordinates.
2. The class contains:
   - Two private data fields **x** and **y** that represent the coordinates
   - A no-arg constructor that constructs a point (0,0)
   - A constructor that constructs a point with specific coordination
   - Two **get** methods for data x and y, respectively
   - A method named **getDistance** that returns the distance from current point to another point of the **TwoDPoint** type. The method signature is:

```
public double getDistance(TwoDPoint remotePoint)
```

3. Write a test program with appropriate test data.

Activity 3-3

Printers

1. Design a class named **PrintMachine.java** with
   - public **static** variable **int *totalNoOfCopy*,** which records the total number of copies made from all the print machines
   - a method with header (below) which will update ***totalNoOfCopy*** value and return a String array with all its element value being ***strText*** and array size being i***ntNos***.

```
public String[] copy(String strText, int intNos)
```

2. Design a program **RunPrintMachine.java**:
   - Create an object called ***Canon*** from class ***PrintMachine***
   - Create another object called ***Fujitsu*** from class ***PrintMachine.***
   - Copy ***x6*** "***Flying!***" on the ***Canon*** machine
   - Copy ***x8*** "***High!***" on ***Fujitsu*** machine
   - Show all the copies' content from the 2 print machines to user and print out the total number of copies generated from the 2 print machines.

Activity 3-4

PC and Components

1. Given the 2 classes below.
2. Redesign the class PC to adopt the DIP.
3. Write the test program with appropriate data.

```
class PC {
  SSD ssd;
  CPU cpu;
  PC() {
     Ssd = new SSD (“Crucial T705”, “2TB”);
     cpu = new CPU(“Intel”);
   }
   public String toString() {
      return (“PC with CPU “ + cpu.brand + “ and SSD “ + ssd.brand);
   }
}
```

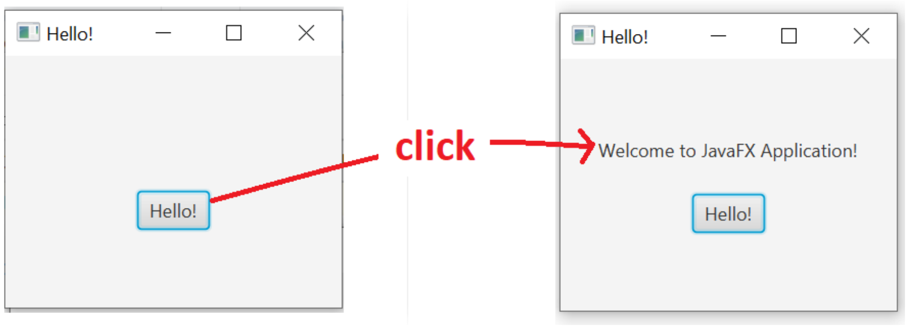
```
class CPU {
   String brand;
    CPU(String brand) {
      this.brand = brand;
   }
}
```

```
class SSD {
   String brand;
   String capacity;
   SSD(String brand, String capacity) {
     this.brand = brand;
     this.capacity = capacity;
   }
}
```

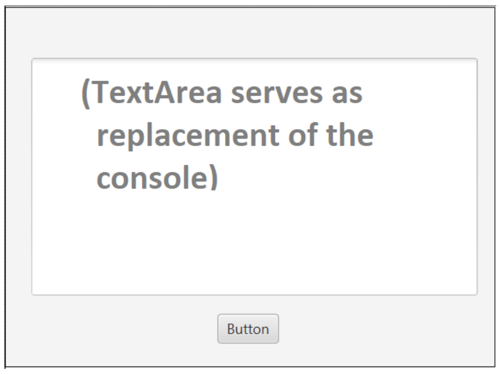
Activity 3-5

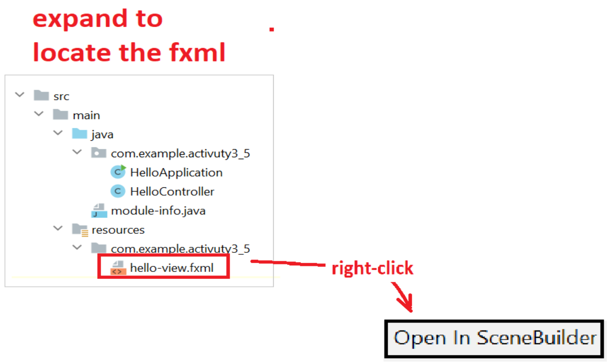
Build your own Window version of "Console output" with JavaFx (Desktop Application)

1. Refer to ***Installation*** guide.
2. Follow the steps to create a default desktop ***JavaFX*** window application.
3. Running the default "***HelloApplication***" program will give a window with a "***Hello***" button. When the button is clicked, a text "***Welcome to JavaFX Application***" will appear.

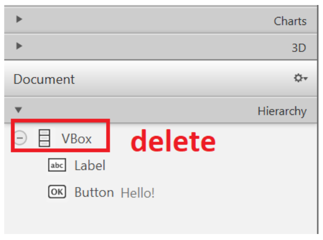


4. We are going to modify this default user interface to the one below with the help of ***Scene Builder***. It has only 2 components: a ***TextArea*** and a ***Button***.

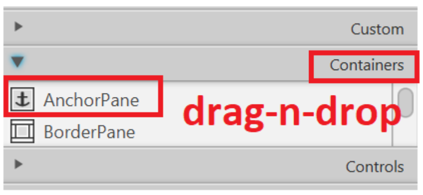


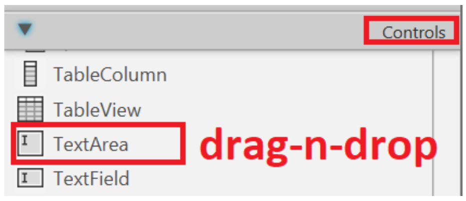


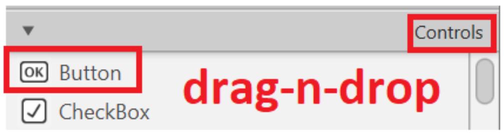
5. In ***Scene Builder***. Delete the default controls.



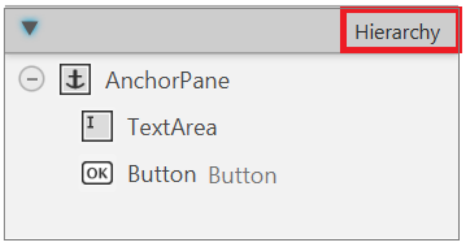
6. Next, drag and add these controls (in sequence) into ***Hierarchy*** and arrange them accordingly.



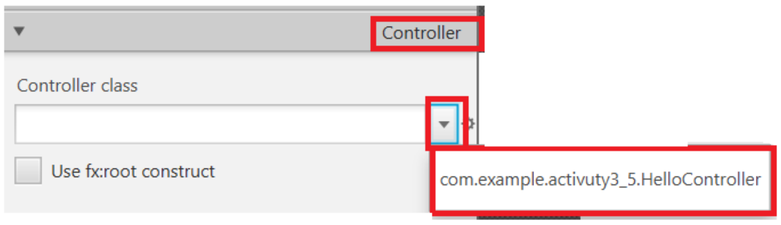




The ***Hierarchy*** panel should look like this:



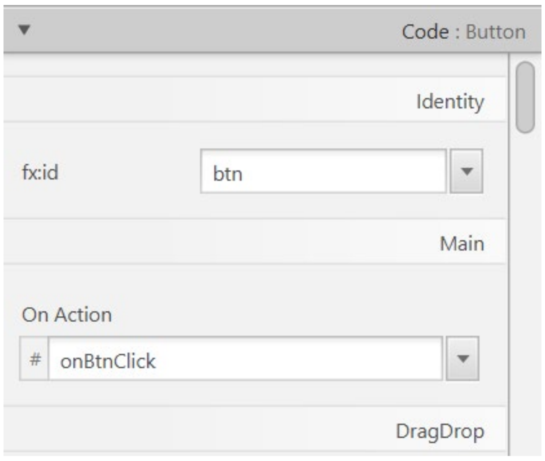
In the ***Controller*** panel, select the default.



Save.

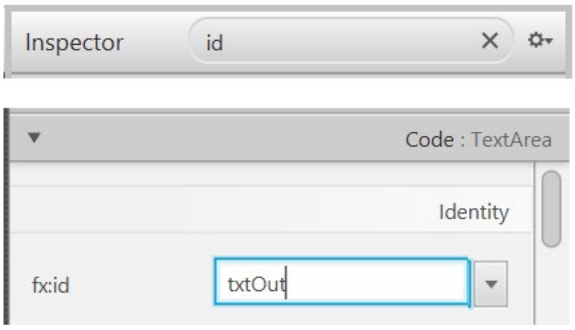
7. Next, assign ***IDs*** to the ***TextArea*** and ***Button***. These ***IDs*** will be used in the code in ***HelloController***.java.

   In ***Scene Builder***:
   - Select the ***TextArea*** in the Hierarchy panel.
   - Check the Inspector panel on the right.
   - Set ***fx:id*** to ***txtOut***.
   - Select the ***Button***.
   - Set ***fx:id*** to ***btn***. In addition, under the ***On Action***, type in ***onBtnClick***. This is going to be the ***callback*** method in the Java code later for the button.



fx:id btn

On Action # onBtnClick

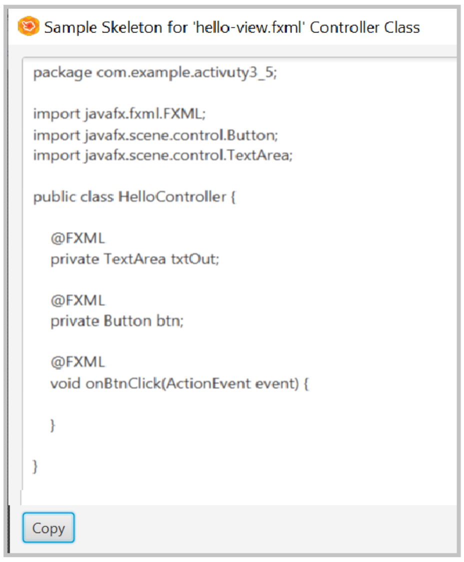


fx:id txtOut

   Save.

8. The changes in user interface in ***Scene Builder*** will be propagated back to the ***fxml*** file in IntelliJ. However, the code will not be ported over automatically.
   The skeleton code for the controller has to be manually copied and paste over in ***IntelliJ***.

   In ***Scene Builder***:
   - Select ***View*** menu.
   - Select ***Show Sample Controller Skeleton***. A window will pop up. Click ***Copy***.

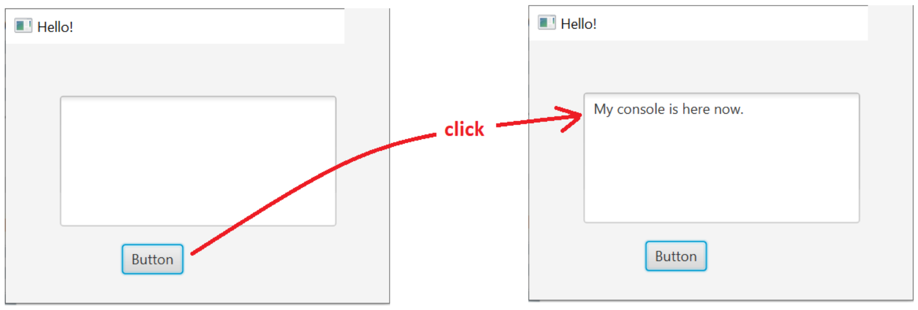


```
package com.example.activuty3_5;

import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.TextArea;

public class HelloController {

    @FXML
    private TextArea txtOut;

    @FXML
    private Button btn;

    @FXML
    void onBtnClick(ActionEvent event) {

    }

}
```

9. Back to ***IntelliJ***:
   - Open ***HelloController.java***, paste the copied code from ***Scene Builder***.
   - Import ***ActionEvent*** to get rid of the errors.
   - Edit the ***callback*** method ***onBtnClick*** to:

```
@FXML
void onBtnClick(ActionEvent event) {
    txtOut.setText ("My console is here now.");
}
```

- Run the ***HelloApplication.java***. The window shall look like:



My console is here now.

setText() – method to display a text in the ***TextArea***.

Now, you can just add you code in the ***callback*** method to display any results, in ***String*** type, in the ***TextArea***.

For instance, to display output in a console program:

```
System.out.println (“Total cost is “ + cost );
```

To display output in the ***TextArea***:

```
txtOut.setText (“Total cost is“ + cost );
```