*CSE 102*

**JavaFX - III**

* So far we developed some small GUI apps and got a sense of GUI programming. As you may have noticed, our code mixed up two conceptually very different parts: logic and design. The design part could further be divided into layout design and purely decorative design.
* A better strategy would be to separate those logically distinct parts into distinct code units. With that aim in mind, we will make that separation from now on. The application logic should stay in the .java source file(s). Layout design should be expressed with FXML markup language (a modified version of XML), and the decorative design should be expressed as CSS files (the same standard people use to decorate websites).
* SceneBuilder is a utility program for designing layouts without writing any code. The program allows you to drag-n-drop components and to lay them out as you want using your mouse and then converts your work into an FXML file.
* [Here](https://docs.oracle.com/javafx/scenebuilder/1/installation_1-1/jsbpub-installation_1-1.htm) is a tutorial on how to download and install SceneBuilder and [here](https://docs.oracle.com/javafx/scenebuilder/1/use_java_ides/sb-with-eclipse.htm) is a tutorial on how to integrate it with Eclipse IDE.

**Exercises:**

1. What is the main functional difference between ToggleButton and RadioButton?
2. RadioButton cannot be a member of a ToggleGroup while ToggleButton can.
3. When grouped with a ToggleGroup you can select multiple ToggleButtons while you cannot do the same thing with RadioButtons.
4. RadioButtons cannot be deselected while ToggleButtons can.
5. They are identical functionally, only their appearance differ.
6. What is the main functional difference between ChoiceBox and ComboBox?
7. ChoiceBox allows single selection while ComboBox allows multiple selections.
8. The order of options is adjustable in ChoiceBox but not in ComboBox.
9. ComboBox takes a generic type argument while ChoiceBox does not.
10. ComboBox has an option which allows user to enter a custom value besides the predefined options.
11. What does showOpenDialog() method of a FileChooser object returns?
12. A File object associated with the chosen file.
13. Contents of the file as a byte array.
14. Name of the file as a String object.
15. Contents of the file as a String object.
16. What is wrong with the following code? Assume that a ChoiceBox and a ComboBox is properly defined in the associated *.fxml* file and their *fx:id* property is set to “choice” and “combo”, respectively.

**public** **class** SampleController **implements** Initializable {

@FXML

ChoiceBox<String> choice;

@FXML

ComboBox<String> combo;

@Override

**public** **void** initialize(URL location, ResourceBundle resources) {

choice = **new** ChoiceBox<>();

combo = **new** ComboBox<>();

choice.getItems().addAll("hello", "world");

combo.getItems().add("option");

}

}

1. Multiple usage of @FXML annotation causes error.
2. The method name *getItems* is wrongly spelled, it should be *getOptions*.
3. Assigning new objects to *choice* and *combo* breaks the connection with .fxml document.
4. The method *addAll* cannot take multiple arguments.
5. I defined a Button in the corresponding .fxml file and set its *fx:id* property to “button”. However the following controller code still doesn’t work. What could be the problem?

**public** **class** SampleController {

**public** **void** buttonAction() {

System.***out***.println("hey");

}

}

1. The handler method *buttonAction* must take an argument of type *ActionEvent*.
2. The *onAction* property of Button in .fxml file may not be set appropriately.
3. Event handlers must return a value.
4. Using a @FXML annotation a Button reference must be put in *SampleController* clas
5. Is there any difference between *MousePressed* event and *MouseClicked* event? If so, what’s the difference?
6. No, they have the same functionality.
7. Yes, *MousePressed* event is fired whenever a mouse button is pressed while *MouseClicked* event is fired when a mouse button is pressed and released on the same node.
8. Yes, *MousePressed* event is fired when any mouse button is pressed while *MouseClicked* event is fired only when the primary mouse button is pressed.
9. Yes, *MousePressed* event is fired when any mouse button is pressed and released while *MouseClicked* event is fired only when the primary mouse button is pressed and released.
10. Write a JavaFX app which displays random images when user clicks a button (gets images from internet).
11. Write a JavaFX app which displays three radio buttons, more than one of which cannot be selected at the same time.

**ANSWERS:**

1. C
2. D
3. A
4. C
5. B
6. B

**public** **void** start(Stage stage) {

BorderPane root = **new** BorderPane();

String imgUrl = "https://picsum.photos/500";

Button button = **new** Button("change");

ImageView img = **new** ImageView(imgUrl);

root.setCenter(img);

root.setBottom(button);

BorderPane.*setAlignment*(button, Pos.***CENTER***);

button.setOnAction(e -> {

img.setImage(**new** Image(imgUrl));

});

Scene scene = **new** Scene(root);

stage.setScene(scene);

stage.show();

}

**public** **void** start(Stage stage) {

RadioButton b1 = **new** RadioButton("button1");

RadioButton b2 = **new** RadioButton("button2");

RadioButton b3 = **new** RadioButton("button3");

// adding buttons to the same toggle group

ToggleGroup g = **new** ToggleGroup();

g.getToggles().addAll(b1, b2, b3);

HBox box = **new** HBox(b1, b2, b3);

box.setSpacing(20);

box.setPadding(**new** Insets(50));

Scene scene = **new** Scene(box);

stage.setScene(scene);

stage.show();

}