*CSE 102*

**JavaFX && SceneBuilder**

* 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. Which of the following is not an absolute necessity in a JavaFX application?
2. Scene
3. Stage
4. BorderPane
5. Main method
6. What kind of role does a CSS file play in a JavaFX app?
7. Describes the style
8. Configures the compilation process
9. Lays out components into their places
10. Expresses the application logic
11. The EventHandler interface specifies a method named …………. .
12. fire
13. add
14. remove
15. handle
16. You use the class …………. to actually display an image.
17. ImageView
18. Image
19. handle
20. False
21. All JavaFX applications must extend the class …………. .
22. Node
23. Application
24. Pane
25. Layout
26. Inspect the following simple app. A blank window is what we see when it is run, what is the problem?

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

Pane root = **new** Pane();

Scene scene = **new** Scene(root, 300, 300);

Label l = **new** Label("Hello World!");

l.setStyle("-fx-font-size: 20px");

root.getChildren().add(l);

stage.show();

}

1. A Pane object cannot be the root node of a scene graph.
2. The object returned by the call to getChildren() is unmodifiable.
3. A Label object does not have a method called setStyle().
4. Stage object is unaware of the Scene object.
5. The following app employs an AnimationTimer object, which calls its handle() method with a frequency of around 60Hz. The input to the handle() method is a long integer which is a timestamp measured in nanoseconds (difference in timestamps of two different calls gives the time elapses between them as nanoseconds). Modify this app so that it counts one tick per second.

**int** counter = 0;

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

Pane root = **new** Pane();

Scene scene = **new** Scene(root, 300, 300);

Label l = **new** Label("Count: 0");

l.setStyle("-fx-font-size: 20px");

root.getChildren().add(l);

AnimationTimer timer = **new** AnimationTimer() {

@Override

**public** **void** handle(**long** now) {

counter++;

l.setText("Count: " + counter);

}

};

timer.start();

stage.setScene(scene);

stage.show();

}

1. Write the simplest drawing app with minimum functionality.

**ANSWERS:**

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

**long** prevTimestamp = 0;

**…**

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

**…**

**public** **void** handle(**long** now) {

**if**(now - prevTimestamp > 1000000000L) {

prevTimestamp = now;

counter++;

l.setText("Count: " + counter);

}

}

**…**

}

**double** prevX;

**double** prevY;

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

Pane root = **new** Pane();

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

Canvas canvas = **new** Canvas(300, 300);

root.getChildren().add(canvas);

GraphicsContext g = canvas.getGraphicsContext2D();

canvas.setOnMouseMoved(e -> {

prevX = e.getX();

prevY = e.getY();

});

canvas.setOnMouseDragged(e -> {

**double** x = e.getX();

**double** y = e.getY();

g.strokeLine(prevX, prevY, x, y);

prevX = x;

prevY = y;

});

stage.setScene(scene);

stage.show();

}