# **ITEC 3150 Exam 2 – Coding Problem**

## Description

This part of the exam is worth 75% of the exam grade. It is open book, open notes. You may not communicate with anyone else during the exam, except, of course, the instructor.

You can ask the instructor one phone-a-friend question.

Submit a single IntelliJ project zip with your solutions, when you are through.

**Please start with the provided starter project.**

*If you cannot get the starter project to run in your environment, create a new JavaFX project in Intellij and copy the java sources (AggregateSum.java, KeyMasterFX.java, Roller.java and StreetWithNames.java) found in the starter project to your package folder (where the HelloApplication.java file is found in the new project.) You will also need to copy streetnames.txt into the top-level folder for your project.*

### Problem 1 (25 points)

Modify the JavaFX GUI in Roller.java so that a button click on *btTurn* results in the *lblResult* being updated. The updated value will be a random number between 2 and 12, inclusive.

The following statement can be used to generate a random number between 2 and 12:

long roll = Math.*round*(10 \* Math.*random*()) + 2;

*lblResult* should use the *verdana* font, with a BOLD weight and REGULAR posture. The font size should be set to 35.

You should also set the scene size and window caption as described below in the rubric.

A successful solution appears as follows, after the Roll button has been clicked:

Graphical user interface, text, application, chat or text message

Description automatically generated

File(s) supplied: Roller.java

Rubric

10 *btTurn*, when clicked, updates *lblResult* with a new random number between 2 and 12, inclusive

5 The font properties for *lblResult* are set correctly (*verdana*, BOLD weight, REGULAR posture and 35 point size).

5 The application title is set to “Feeling Lucky!”

5 The scene used is set to 200 pixels wide by 100 pixels high

### Problem 2 (25 Points)

Your colleague is building a game and wants to put in some hot keys for rapid navigation. You offer to build a prototype to demonstrate the functionality. Modify the provided KeyMasterFX class so that when the A,S,W and D keys are pressed, the Label (named *label*) updates to read *Left*, *Backward*, *Forward* and *Right*, respectively. Modify *label* so that it displays the proper text with an Arial font, bold text, regular posture, and a point-size of 60.

The app should appear as shown when A,S,W and D are typed:

Graphical user interface, text, chat or text message

Description automatically generatedGraphical user interface, text, application, chat or text message

Description automatically generated Graphical user interface, text, application, chat or text message

Description automatically generated Graphical user interface, text, application, chat or text message

Description automatically generated

Rubric

10 Modify the provided KeyMasterFX class so that when the A,S,W and D keys are pressed, the Label (named label) updates to read Left, Backward, Forward and Right, respectively.

7.5 Modify label so that it displays the proper text with an Arial font, bold text, regular posture, and a point-size of 60.

7.5 Modify label such that the label’s text color is red for a backwards move and black otherwise.

File(s) supplied: KeyMasterFX.java

### Problem 3 (25 Points)

Your colleague has code that retrieves a list of all named streets within a specific geography. In this example, we will process the results of retrieving street names from OpenStreetMap for Gwinnett County. The contents are in the starter project and named streetnames.txt. Modify StreetsWithNames.java to start FIVE threads. Each thread should count the number of entries that contain the string *Gwinnett,* by examining a subset of the entire list. The counting should be case insensitive, counting each of *gwinnett*, *Gwinnett* or *gwiNNeTt* as a valid occurrence. Each thread should update the AggregateSum object. Make sure that AggregateSum is modified to make it thread-safe.

File(s) supplied: AggregateSum.java, StreetsWithNames.java, streetnames.txt

Rubric

10 FIve threads are created. Each thread counts the number of occurrences of *peachtree* in it’s own sublist, which is one fifth of the entire list.

10 AggregateSum has been updated to make it thread-safe

5 The aggregate sum is printed after all thread have concluded their processing. The aggregate sum is correct!

## **Other Notes**

No expository video is required.