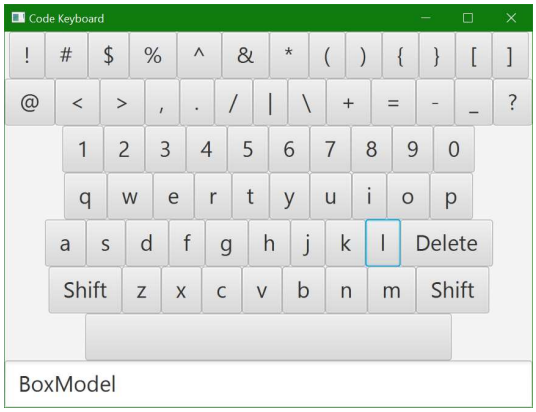
CMPT 381 Assignment 5: Text Prediction

Due: Monday, April 10, 11:55pm ***(*Note later due date because of final exam*)***

Overview

In this assignment, you will work with soft keyboards and text models to develop a predictive keyboard for Java coding, using JavaFX.

Part 1: A Coding Soft Keyboard in JavaFX



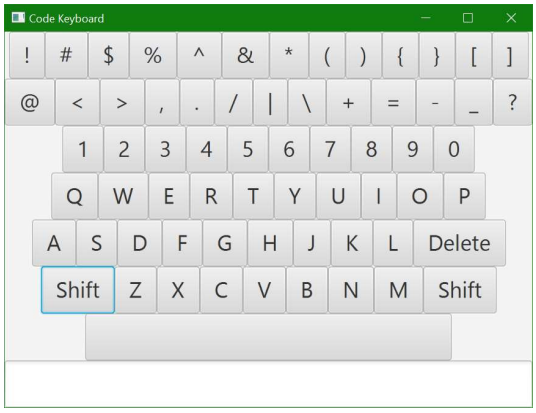
Using JavaFX, build the on-screen keyboard pictured at right.

Interface requirements:

- The UI consists of several JavaFX Buttons (one for each keyboard key), and one entry box below the keyboard.

Interaction requirements:

- When the user clicks on a key button, that symbol is added to the entry box at the bottom of the window (unless the key is Delete or Shift)
- When the user clicks Delete, the rightmost character is deleted
- When the user clicks Shift, all letters change to upper case (see lower picture); the letters stay in upper case until the Shift key is clicked again.
- Shift has no effect on the number or symbol keys



Software requirements:

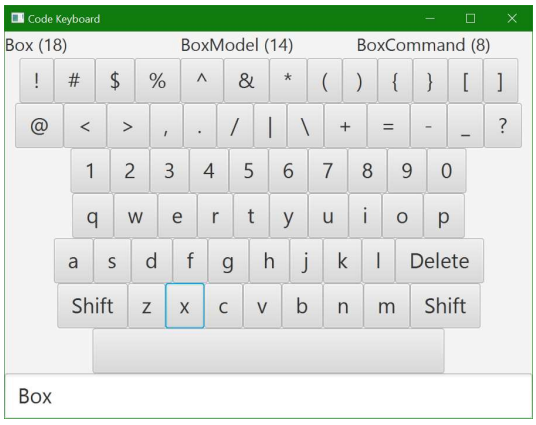
- All keyboard keys should be implemented as JavaFX Buttons
- Although it is possible to do the keyboard layout one button at a time, it will be much simpler to do it algorithmically
- Build the keyboard UI, and handle button events, within the main JavaFX application class
- Implement method “changeCase( )” to change from upper to lower case
- You do *not* need to use an MVC architecture for the application

Resources for part 1:

- JavaFX layout tutorial: docs.oracle.com/javafx/2/layout/builtin_layouts.htm

Part 2: Modeling and Predicting Java Code

In the second part of the assignment, you will add capabilities to your system from Part 1 that process a directory of Java source files (to build a model of Java code), and that provide word-level predictions in the interface based on what the user has typed.



Additional interface requirements:

- Add UI components to the top of the interface to show the top three suggestions from the predictor (see picture at right)

Additional interaction requirements:

- As the user types on the soft keyboard, the system generates predictions based on what has been typed so far. For example, in the system shown at right, the user has typed “Box” on the soft keyboard, and the top predictions are shown above the keyboard (“Box”, “BoxModel”, and “BoxCommand”). In addition, the suggestion UI shows the number of times that the suggestion appears in the trie structure (see below) – for example, “Box” was seen 18 times in the training data.

- Any change to the entered text (including changes resulting from the Delete button) regenerate a new prediction.
- Clicking on any of the suggestions copies the suggestion into the entry box

***Additional software requirements*:**

- Add classes Trie and TrieNode, based on the example in the moodle Examples folder. The trie structure will hold the model of Java code that will allow you to make predictions
- To build the trie:
    - Read in all of the Java source files from the BoxFX example (on moodle)
    - Split the text into words using the following characters as delimiters: space, round brackets, period, and comma.
    - Add each word to the trie
- Add class Candidate to hold a candidate word and its frequency
- Write method `public ArrayList<Candidate> getCandidates(String prefix)` in class Trie (you will also need subsidiary methods in class TrieNode):
    - Traverse the trie using the current prefix string (i.e., what the user has typed so far)
    - Create Candidate objects for each complete word below this point in the trie (with their frequencies)
- Write method `public void getSuggestions(String prefix)` in the application class. This method will be called whenever the text in the entry box changes, and will put the top three suggestions (sorted by frequency) into the suggestions UI

Resources for part 2:

- Trie example code: on the moodle in the Examples directory
- BoxFX code: on the moodle in the Examples directory
- Trie on Wikipedia: https://en.wikipedia.org/wiki/Trie

This assignment is to be completed individually; each student will hand in an assignment.

What and Where to hand in

- JavaFX: a zip file of your NetBeans project folder for **either** part 1 (if that is as much as you have completed) **or** part 2. If you have completed part 2, you do not have to hand in a project file for part 1.
- A readme.txt file that indicates exactly what the marker needs to do to run your code. (Systems for 381 should never require the marker to install external libraries).
- Hand in your two files (one zip and one readme.txt) to the link on the course Moodle.

Evaluation

Marks will be given for producing a system that meets the requirements above, and compiles and runs without errors. Note that no late assignments will be allowed, and no extensions will be given, without medical reasons.