LEBANESE AMERICAN UNIVERSITY

Electrical and Computer Engineering Dept

COE 312
Data Structures

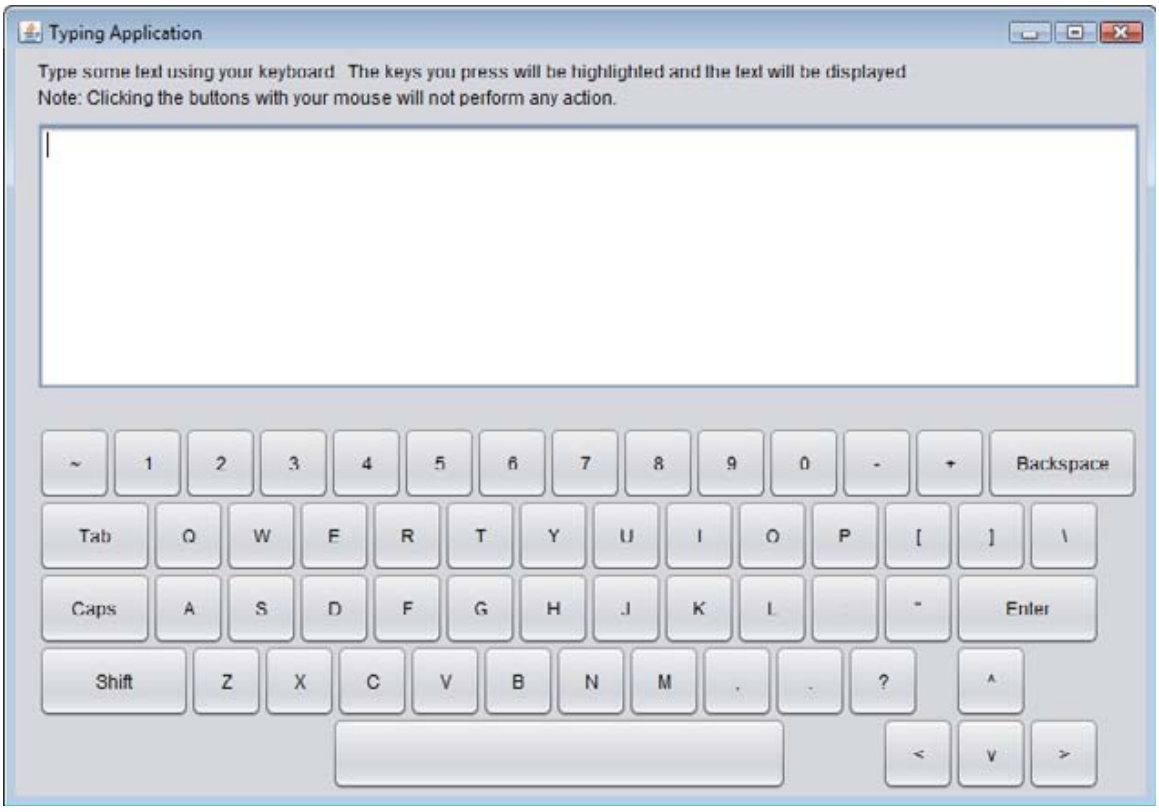
Fall 2013

W. FAWAZ

Project II

I. Objective

Typing quickly and correctly is an essential skill for working effectively with computers and the Internet. In this project, you will develop a GUI-based application that can help users learn to "**touch type**" (i.e., type correctly without looking at the keyboard). The application should display a **virtual keyboard** like the one given below.



The keyboard should allow the user to watch what he or she is typing on the screen without looking at the actual keyboard. Use `JButtons` to represent the keys. As the user presses each key, the application **highlights the corresponding** `JButton` on the GUI and **adds the character** to a `JTextArea` that shows what the user has typed so far.

II. Pangram

You can test your program by having your end user type a **pangram** – a phrase that contains every letter of the English alphabet at least once – such as “the quick brown fox jumped over a lazy dog”. Other pangrams can be found on the web and you are **strongly encouraged** to get as many pangrams as you can and integrate them into your program.

To make the program interesting, your application is required to support the additional functionalities delineated below.

III. Error reporting

The first feature that you are required to equip your application with is related to **error reporting**. Specifically, your application has to be able to monitor the user’s accuracy as follows. While the user is typing one of the **randomly generated pangrams** that are pre-stored in your application and that you display on the screen above the virtual keyboard, the application should keep track of how many keystrokes the user **types correctly** and how many are **typed incorrectly**. This would enable your application to keep track of which keys the user is having difficulty with and display a report showing those keys.

IV. Timer function

The second bit of functionality to be supported by your application is the addition of **a timer function** that gives the user a limited amount of time to type an entire pangram.

V. Scoring System

Once you have introduced the timer functionality to your application, augment your implementation with the following additional feature. Make sure that your application is capable of **keeping track of the user’s score**. The score assigned to the user depends on the number of correct and incorrect keystrokes. In particular, the score can be derived from these numbers by subtracting the number of correct keystrokes from the number of incorrect ones.

VI. Levels and Bonus

Finally, finalize the implementation of your application by including up to three levels referred to as “Easy”, “Medium”, and “Difficult” respectively. The degree of difficulty associated with each level should be tied with both the **number of mistyped keystrokes** your end user is entitled to and the **amount of time** within which the entire pangram is to be typed.

For example, you can envisage imposing the following restrictions on the user during the course of the “Easy” level. The user can make up to five mistakes during the said level and get away with it as long as the pangram is type fully within 60 seconds for instance. Eventually, the expectations will become higher when the user advances to the “Medium” level. At this level, the user cannot make more than 3 mistakes and he will have to fully type the pangram within a shorter amount of time of say 30 seconds. More stringent requirements are imposed on the user for the “Difficult”

level as your application gets less tolerant at this stage. So, you can require the user to make at most one typing mistake under stricter timing requirements of say 20 seconds.

To further improve the end user experience and make the application even more exciting, your last task would be to introduce bonuses to each one of the three levels as illustrated in what follows. In the Easy level, you can reward your end user by adding 10 extra seconds to the timer whenever he correctly types 9 consecutive letters. As for the Medium level, the reward can be the same but with a higher expectation in terms of the number of correctly typed consecutive letters. A value of 13 sounds a proper value for this parameter. As for the Hard level, wait until the user types correctly 20 consecutive letters before awarding a bonus of just 3 seconds.

What to turn in?

This project is due at the beginning of class on the due date. You have to turn in the following material in both **hard** and **soft** copies.

Criteria	Percentage
Documentation of your solution including explanations and illustrations in one or two pages along with short write-up of questions and/or problems that you encountered while doing this assignment.	2 pts (10%)
Source code that contains an appropriate amount of comments. Well-organized and correct code receives 16 pts, messy yet working code receives 10 pts, code with bugs receives 2 pts, and incomplete code receives 1 pt.	16 pts (80 %)
Execution output such as a snapshot of the contents of standard output. A correct output receives 2 pts, the one with minor errors receives 1 pt, and an incomplete output receives 0 pts.	2 pts (10%)
Total	20 pts (100%)

Good Luck!