The University of Western Ontario

Faculty of Engineering

Department of Electrical and Computer Engineering

**Software Engineering for Human-Computer Interface Design**

**Prof. Roy Eagleson**

***Simple Calculator User Interface project assignment***

Consider the Design process for creating a user interface: The Software Engineering methodology will always proceed from informal requirements specifications, to implemented physical executable systems. We will now explore these transformations in detail, by following a classical pathway from requirements to implementation and testing.

The following set of case studies will involve the design of a Human-Computer Interface that will be a tool that will allow users to perform standard arithmetic calculations. This design will be challenging, since there will be a wide range of individuals (from novice to expert) who will be identified as users. In addition, the tool should be usable for generic arithmetic tasks, as well as for specific complex tasks.

As a review: There are several possible implementations of systems to assist with calculations (Abacus and Slide Rules notwithstanding -- not to mention paper-and-pencil). Yet, today we can focus on the over-arching concept of keypress-based user interfaces, with simple text-based output displays representing the digits corresponding to the input and output numbers. Examples of simple arithmetical problems, provided as test cases for the uses of such an interface, are given in the appendix of this document – but will be expanded later through discussions in class, as evaluative test cases.

We will be considering the design of three versions:

1. A simple 4-function calculator, of the kind that can be bought in a dollar store, which has buttons 0-9, and the function buttons ‘+’,’-‘,’x’, ‘/’, along with ‘=’ . This calculator will receive inputs from the user in normal “INFIX” mode, but without any operator precedence. In other words, if the user presses the following sequence of inputs { ‘5’ ‘+’ ‘3’ ‘x’ ‘2’ ‘=’ } the user will see ‘5’,’3’,’8’,’2’and then the answer should be ‘16’. (not ‘11’)
2. A simple 4-function calculator, which operates in “RPN” mode. It has the same numbers 0-9, and the function buttons ‘+’,’-‘,’x’, ‘/’, but instead of ‘=’ it has an ‘enter’ key. The mode of entry for the same calculation would be { ‘5’ ‘enter’ ‘3’ ‘+’ ‘2’ ‘x’ } which are the keystrokes that will result in ‘5’,’3’,’2’and finally ‘16’.
3. A 4-function calculator, which obeys the order of operations given when multiplication and division take precedence over addition and subtraction. Also, as a convenience to the user, bracketing keys ‘(‘ and ‘)’ are provided, so that the same calculation would be entered as ‘(‘ ‘5’ ‘+’ ‘3’ ‘)’ ‘x’ ‘2’ ‘=’ , the user will see ‘5’,’3’,’8’,’2’and then the answer should be ‘16’. But, if they type ‘5’ ‘+’ ‘3’ ‘x’ ‘2’ ‘=’ they will see ‘5’, ‘3’, ‘2’, ‘11’

There are a few extensions which may arise naturally. In particular, the addition of a decimal point ‘.’ (for now, let’s not include ‘scientific notation ‘E’, although it would be another natural extension). A ‘clear entry’ and ‘clear’ key “CE/C” is fairly easy to add. Another possible addition would be the memory store and memory recall keys “M” and “MR”.

Over and above these natural extensions, a question still arises. For any particular user interface mode (1) (2) or (3), it is critical to compare the usability of each mode. In particular, for any set of test cases, which interaction mode allows the user to perform their calculations with minimal errors? Which interaction mode allows them to perform their calculations more rapidly? Which do they find more satisfying? Which are unintuitive? In other words, which are more usable?

To answer these questions, a simple functionality can be added, and that is, to log each keypress from the user, and timestamp them while writing to a logfile. A simple .csv file usually suffices, where, for each keypress, the system clock time (with millisecond accuracy), along with the particular keystroke, is logged on a line-by-line basis into the .csv file, for each user, and each training session. It helps with analysis of the logfiles if the username, test case number, and date is recorded encoded within the filename, or within the .csv file itself.

Deadlines for this assignment will be as discussed in class, and posted on OWL.

Technical note: in order to simplify this assignment, do not implement ‘divide by zero’ error checking, nor any limitation on the number of digits entered, nor any functionality for entering negative numbers. These additions would not be difficult, but by omitting them, this assignment will serve as an introduction to the steps needed to proceed through all stages of the Requirements, Analysis, Design, Implementation and Testing phases.

Appendix: Sample Test Cases for your Calculator Interface

For each of the three calculator modes (1) (2) and (3), investigate the Utility and Usability of these three interfaces as tools to assist with the following calculations: [further test cases will be provided in class, through discussion ]

1. What is the difference between the following two numbers: 57 times 13 -- and 161 divided by 7.
2. Double the number that results from the difference between 1023 and 127
3. What is the area of a rectangle that has one side 789cm and the other side is 148cm longer than that amount ?
4. If the tax on gasoline is 9 cents per litre, and last week you consumed 135 litres of gasoline, but this week, you only consumed 117 litres of gasoline, then what is the total amount of tax that you saved?
5. If two numbers differ by 154, but the ratio of these two numbers is 15, then what is the smaller of the two numbers?