**Institute of Technology Tralee**

**Computing Department**

**Object Oriented Programming 1**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Practical 10 Extra – Input Validation**

This extra lab sheet just poses a few more input validation problems for you to try out, taken from past assessments.

**Exercise 1**

A Java program is required that will read in the username and password combinations of 5 people, using a **for** loop. You can take it that the user will enter a valid username each time, but each password must be fully validated to ensure it complies with the following criteria:

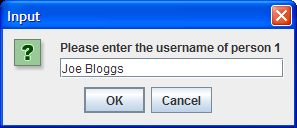
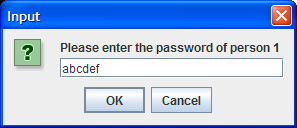
* must have a minimum of 8 and a maximum of 15 characters in total
* must only contain letters or digits
* must contain at least 3 digits

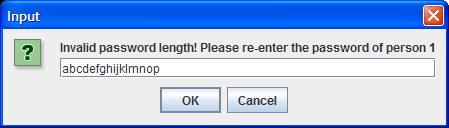
The user will keep getting issued with a suitable error message until they enter a valid password. The error message will indicate a reason why the password was invalid.

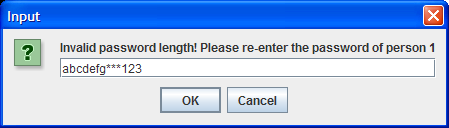
Once all 5 valid username/password combinations have been entered, they will then be displayed neatly aligned via a text-area on a message dialog. You can take here it that the longest username will be 25 characters. On your text-area, you can display the information using a “monospaced” plain font with a point size of 11.

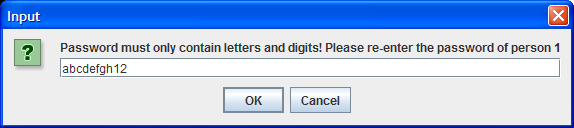
**Sample Screenshots**

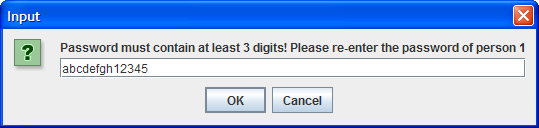
**The main loop begins by asking the user to enter the first username and password. Here, several invalid passwords are entered and an appropriate message gets issued to the user indicating the problem, asking them to re-enter.**

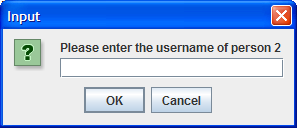




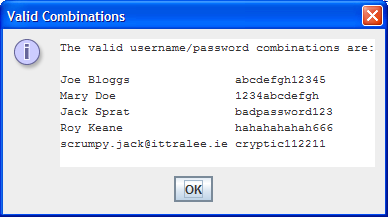




**Finally, a valid password is supplied and the main loop progresses to ask for the second username.**



**…. more input and validation follows and finally, when the main loop has finished, the program displays the following message dialog:**



**Exercise 2**

You must write a Java program that validates a Visa debit card number that has been issued by the AIB bank in Tralee. In order to be valid, the debit card number must

* Be 16 characters long
* Begin with the digit 4
* Contain all digits
* Satisfy the “golden rule”

In order to demonstrate the “golden rule”, take the following sample debit card number:

4417123456789113

1. The first step is to double every second number beginning with the first number, so we get (numbers spaced out for clarity, with doubled values in bold):

**8** 4 **2** 7 **2** 2 **6** 4 **10** 6 **14** 8 **18** 1 **2** 3

In order to convert the characters in the debit card from a character to its numeric equivalent, you can use the **Character.getNumericValue()**method. You simply pass in the character you have extracted as an argument and it will return the numeric equivalent e.g.

Character.getNumericValue(‘4’) would return the integer value 4, which you can then double.

1. Now simply add **all the digits** of all the numbers above (note that 10 becomes 1+0 = 1 and 14 becomes 1+4 = 5 – hint: you can use the remainder operator **%** to help convert these 2 digit numbers to 1 digit numbers quickly)

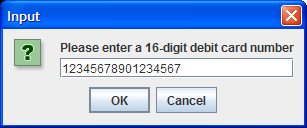
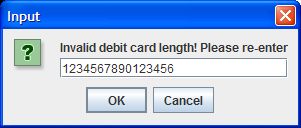
The total of all these digits is 70

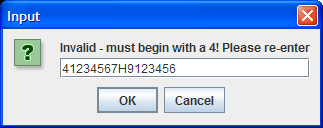
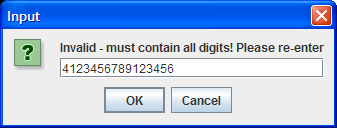
1. Check to see if the total of all the digits is divisible by 10. If it is then the “golden rule” has been satisfied and the debit card number must be valid.

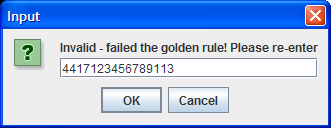
If the debit card number is found to be invalid, your program should issue a suitable error message to the user indicating what was wrong with it and ask the user to re-enter.

**Sample Screenshots**

**The user gets asked to enter a debit card number. Several invalid ones are entered and the user gets asked to re-enter each time, being told what was wrong with the supplied value:**



**… eventually a valid debit card number is entered and the user gets a confirmation message dialog**

