**Institute of Technology Tralee**

**Computing Department**

**Structured Programming 2**

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

**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. Read the questions very carefully and then have a good go at them.

**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:

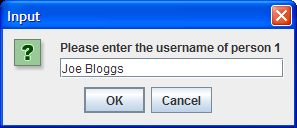
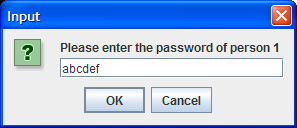
* must have a minimum of 8 and a maximum of 15 characters in total
* must only contain letters or digits
* must contain at least 3 digits

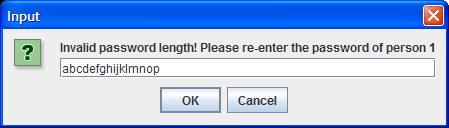
The user will keep getting issued with a suitable error message until they enter a valid password. The error message will indicate a reason why the password was invalid.

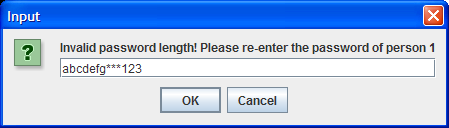
Once all 5 valid username/password combinations have been entered, they will then be displayed neatly aligned via a text-area on a message dialog. You can take here it that the longest username will be 25 characters. On your text-area, you can display the information using a “monospaced” plain font with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

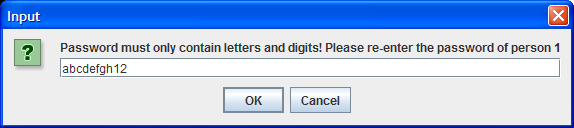
**Sample Screenshots**

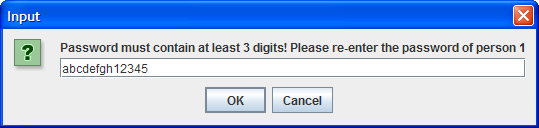
**The main loop begins by asking the user to enter the first username and password. Here, several invalid passwords are entered and an appropriate message gets issued to the user indicating the problem, asking them to re-enter.**

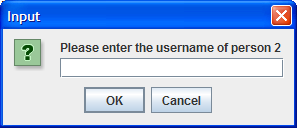




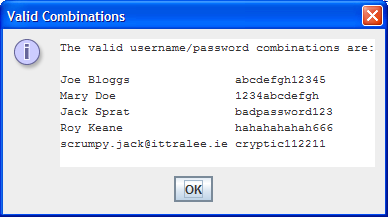




**Finally, a valid password is supplied and the main loop progresses to ask for the second username.**



**…. 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)

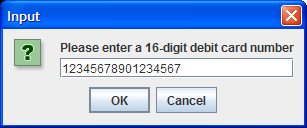
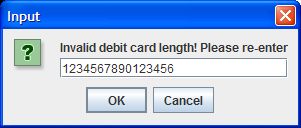
The total of all these digits is 70

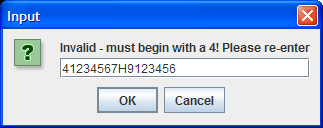
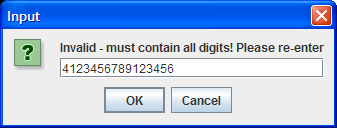
1. Check to see if the total of all the digits is divisible by 10. If it is then the “golden rule” has been satisfied and the debit card number must be valid.

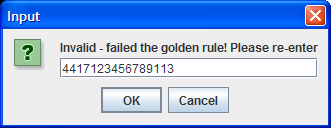
If the debit card number is found to be invalid, your program should issue a suitable error message to the user indicating what was wrong with it and ask the user to re-enter.

**Sample Screenshots**

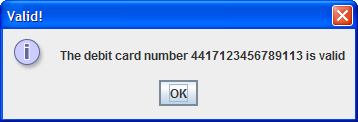
**The user gets asked to enter a debit card number. Several invalid ones are entered and the user gets asked to re-enter each time, being told what was wrong with the supplied value:**



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



**Exercise 3**

In counties Kerry, Limerick and Cork, the landline telephone number format is (0xy)zzzzzzz - note there are **no spaces** here. x must be the digit 2 or 6, y can be any digit in the range 1-9 inclusive and z can be any digit in the range 0-9 inclusive. Therefore a valid telephone number from one of these counties must

* Be exactly 12 characters long
* Begin with “(0”
* The 3rd character must be a 2 or a 6
* The 4th character must be within the range 1-9 inclusive
* The 5th character must be a closing parenthesis i.e. )
* The last 7 characters must all be digits (use a **for loop** in this part)

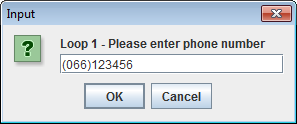
A Java program must be written that reads in exactly 5 telephone numbers. Each piece of data entered must be checked to ensure it constitutes a valid telephone number.

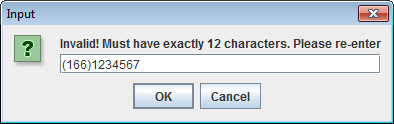
Once each piece of data is entered, it is validated to ensure it complies with the criteria outlined above. You should perform the tests in the order indicated above, starting in each case by testing the number of characters in the string entered. As long as the value entered remains invalid, it should be rejected with the user given the reason for failure along with a prompt to re-enter.

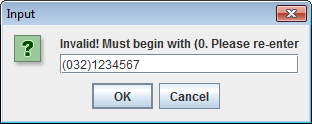
Once the main loop has finished, the program should display a message dialog showing a list of the valid phone numbers that were entered.

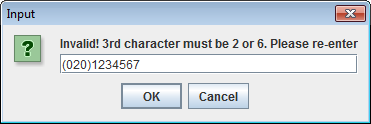
**Sample Screenshots**

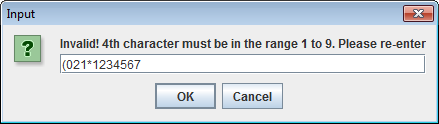
**In this first iteration of the main loop, the user begins by entering several invalid phone number values. The user is told what is wrong with the value supplied each time and the validation loop only quits once a valid phone number has been supplied.**

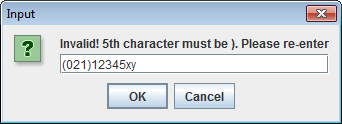


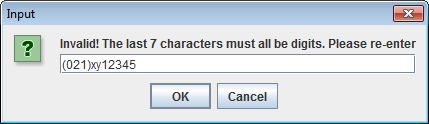


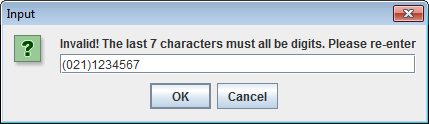




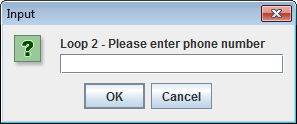




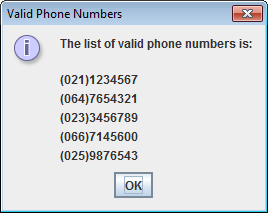




**At last a valid phone number value is supplied so the validation loop exits and the user gets asked for another phone number value – this, and subsequent values, will then be validated in the same way as before**



**When the last valid phone number value has been supplied and the main loop exits, a message dialog appears showing the list of 5 valid phone numbers that had been entered**



**Exercise 4**

All current registration plates for cars purchased in Ireland must conform to the following criteria:

* It must be between 6 and 12 characters long (inclusive)
* The first 2 characters of the registration plate must be digits that represent the year of purchase
* The third character must be a dash symbol,
* The next 2 characters must either be 2 uppercase letters or else a single uppercase letter followed by a dash, to represent the county in which the car was purchased. If the 4th and 5th characters were both uppercase letters, then the 6th character must be a dash, but if the 4th and 5th characters were an uppercase letter followed by a dash then the 6th character must be a digit
* Beyond the 6th character, all the remaining characters in the registration plate must consist of digits - as there could be any number of digits, up to a maximum of 6 digits, for this part of the registration plate, you will need a **loop** to process these characters

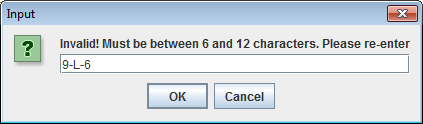
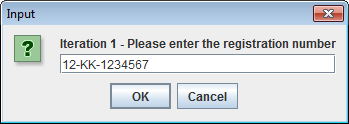
Examples of valid registration plates would be 08-KY-1234 and 97-L-985671, while examples of invalid ones would be 2008-C-12, 89-LKK-234 and 67\*WD\*6537

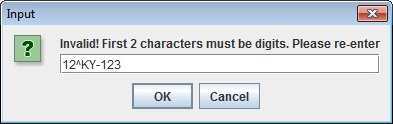
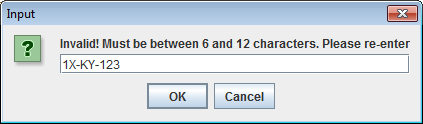
You must write a Java program that processes exactly 4 registration numbers. You can use any structure you like for the main loop iteration process here. As each registration number is entered, it must be fully validated so that it complies with the criteria outlined in the previous paragraphs. The tests should be performed in the order indicated by the screenshots below, starting with the length of the registration number string. As long as the registration number entered remains invalid, it should be rejected with the user given the reason for failure along with a prompt to re-enter.

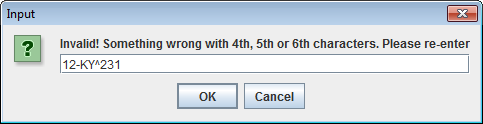
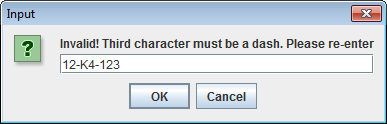
Once the main loop has finished, the program should just display all 4 valid registration numbers on a message dialog.

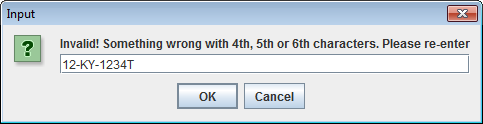
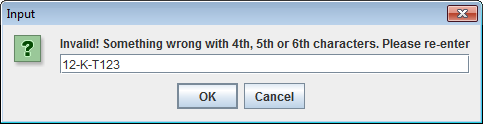
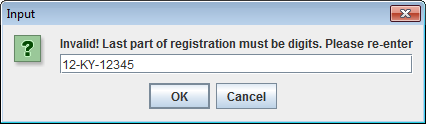
**Sample Screenshots**

**In this first iteration of the main loop, the user begins by entering several invalid registration number values. The user is told what is wrong with the value supplied each time and the validation loop only quits once a valid reg number has been supplied.**

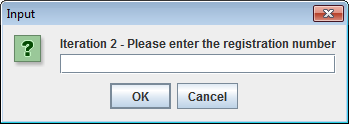






**At last a valid registration number is supplied so the user gets asked for another registration number – this will be validated also**



**When the 4th valid registration number has been supplied, a message dialog appears to display the 4 valid values**

