Assignment Tutorial Letter 2025

Advanced Programming
COS3711

Year module

Computer Science Department

Assignment 1 Questions

BARCODE

Assignment 1

1. Introduction

Please note that Qt Designer should not be used, and you are expected to manually set up GUIs to ensure that you properly handle memory using Qt's parent-child functionality.

Marks will also be awarded for following good programming practice, for example, naming conventions, code layout, using forward class declarations and initialiser lists in constructors, and GUI handling like setting focus, tabbing and clearing input widgets (like text input fields being cleared and spin boxes being returned to some default value), and providing appropriate user feedback. Your code should build and run without any warnings.

2. Question 1

Write a console application that can be run from the command line using the following forms:

```
count                                   // run without any parameters
count file1.txt                         // pass one file name
count file1.txt fileTwo.txt             // pass more than one file name
count –a –b file1.txt    fileTwo.txt    //   pass   flags   to   change
behaviour
count –ab –c file1.txt              // pass flags in an alternative way
```

If no arguments are provided, then print a message describing what arguments should be included.

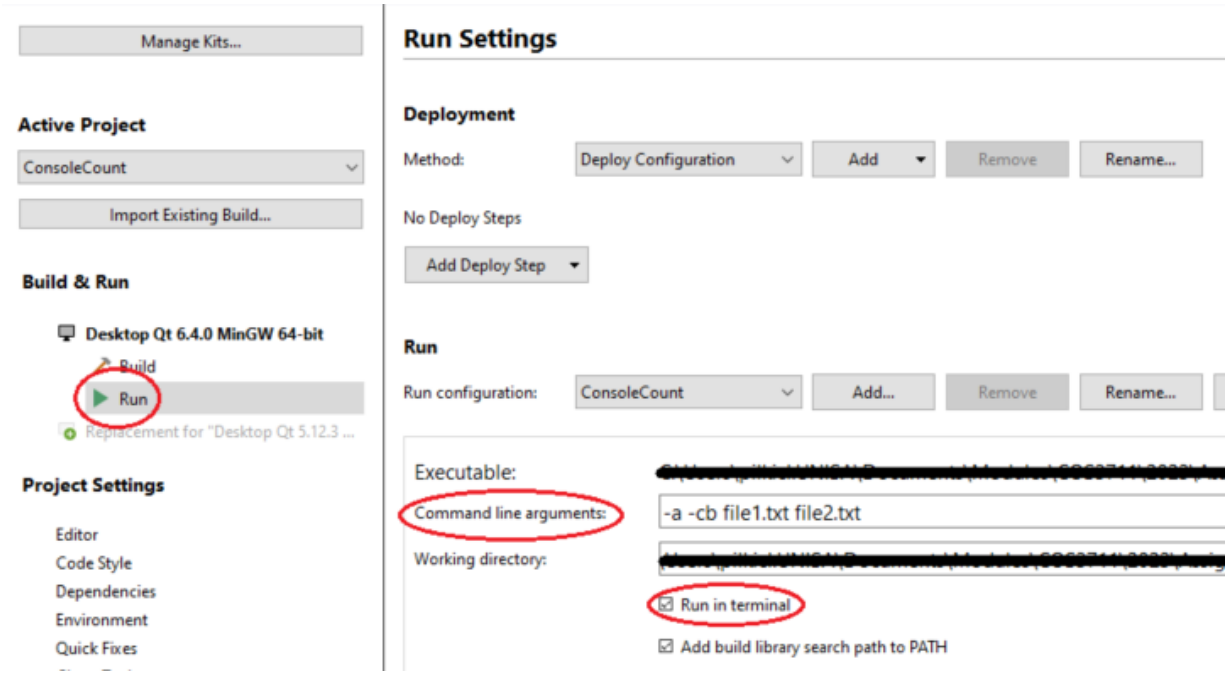
The application should, using regular expressions, count the number of occurrences of each of the following in the text files given.

- If the `–a` flag is set, count the number of words longer than 4 characters that start with a capital letter. There may be further capital letters in the word.
- If the `–b` flag is set, count the number of words that are hyphenated. This hyphen should not be at the start or end of a word.
- If the `–c` flag is set, count the number of words that start and end on the same character.
- If the `–d` flag is set, count the number of words that do not start with a vowel. Note that these words can start with any character, and do not just have to start with alphabetic characters.

If no flags are provided, it is assumed that all counts should be performed.

It is suggested that you remove all whitespace at the start and end of the document, as well are removing all punctuation (`.,?!:;` and so on) – try doing this using a regular expression. Assume that there is only 1 space between words.

Remember that to set command line arguments in Qt Creator, click on *Projects* in the left menu, click on the *Run* menu and enter the arguments in the *Command line arguments* field. You can also click the *Run in terminal* check box.



See the screenshot below for an example of the output for the arguments that were used above.

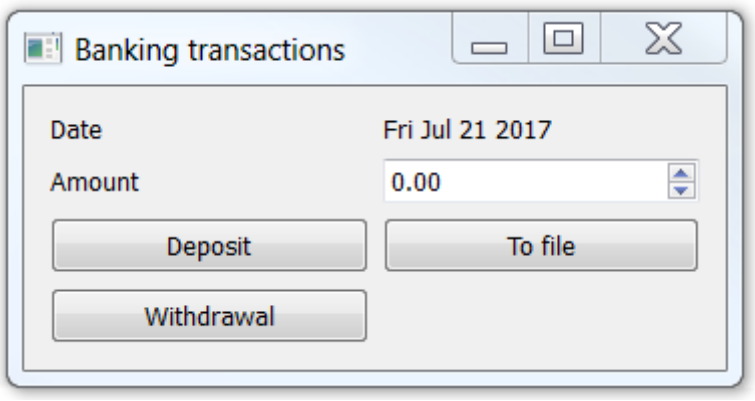
```
***file1.txt***
Number of words longer than 4 letters that start with a capital: 3
Number of hyphenated words: 0
Number of words that start and end on the same letter: 2

***file2.txt***
Number of words longer than 4 letters that start with a capital: 2
Number of hyphenated words: 2
Number of words that start and end on the same letter: 1


Press <RETURN> to close this window...
```

Question 2

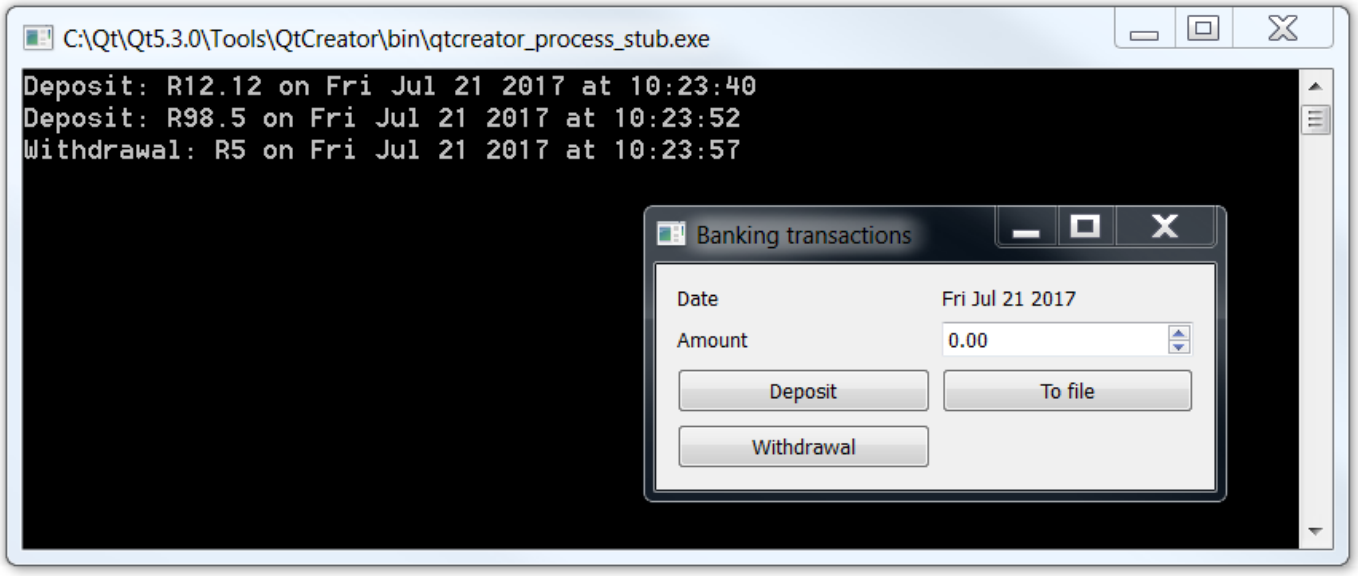
Create a graphical user interface that allows a user to deposit or withdraw an amount. Display the current date on the interface. Below is an example of such an interface.



Create a `Transaction` class that stores the date, time, amount, and type of transaction (deposit or withdrawal, implemented as an `enum`). `Transaction`s should be handled as pointers.

Create a `TransactionList` class that keeps a single list of transactions. There should be only one instance of this list allowed, so implement it using the Singleton design pattern.

When the user clicks the `Deposit` or `Withdrawal` button, the transaction should be written to the transaction list. Display the transaction in a console window as well (this is as a check that the transaction is actually being created.




```
Deposit: R12.12 on Fri Jul 21 2017 at 10:23:40
Deposit: R98.5 on Fri Jul 21 2017 at 10:23:52
Withdrawal: R5 on Fri Jul 21 2017 at 10:23:57
```

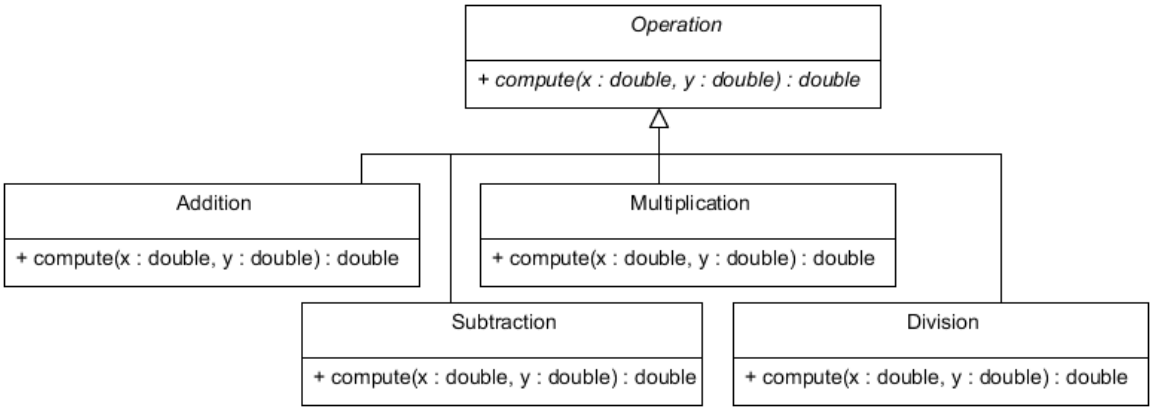
When the user clicks on the `To file` button, the list of transactions should be written to a human-readable text file.

Ensure that your graphical user interface class handles only the interface, and does not contain code that is used to manage the transaction list; create a separate class for this purpose.

Question 3

In this exercise, you will design a simple calculator with four basic operations, namely addition, subtraction, multiplication, and division. Implementation of this simple calculator should separate the code for user interface and mathematical operations.

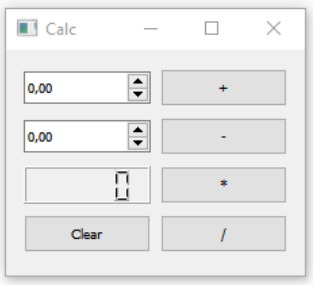
The code for performing mathematical operations should be designed using the following class hierarchy:



The `Operation` class hierarchy contains the logic for performing various mathematical operations. A concrete class in this hierarchy represents one mathematical operation, which is performed in its `compute()` function that returns the result of an operation.

Implement the Factory method design pattern using one factory class named `OperationFactory` so that an instance of a class in the `Operation` class hierarchy can be created based on the requested operation (`+`, `-`, `*`, `/`). Additionally, `OperationFactory` should also be an implementation of Singleton.

The user interface for the calculator should make use of five `QPushbutton`s, two `QDoubleSpinBox`es and one `QLCDNumber` as shown below:



`QDoubleSpinBox`es allow user to enter numbers and `QLCDNumber` is for displaying the result of the calculation. To use the calculator, the user should first enter two numbers and then select one of the four (`+`, `-`, `*`, `/`) buttons. The result of the operation is displayed in `QLCDNumber`. The button `clear` is used to clear `QDoubleSpinBox`es and `QLCDNumber`.

Note the following:

- The signals emitted by the four buttons (`+`, `-`, `*`, `/`) should be handled in a single slot.
- The user interface code should make use of the `Operation` class hierarchy to get the result of the requested operation.
- Handle overflow in `QLCDNumber` display appropriately.
- Handle division by 0 appropriately.