Halit Özsoy, Yusuf Kalaycı
2016400141, 2016400357

20.12.2017
CMPE 230 – HOMEWORK 3

REPORT / DOCUMENTATION FOR HOMEWORK 1

Problem:

Develop a QT program that implements a currency converter that should look like the following:

TL :			100 TL
Rate:			0.25854 TL/USD
Amount:			25.85 USD
USD	EUR	GBP	CNY

Our Solutions & Challenges We Faced

Firstly, we decided to implement the program in one main file and a reader class for network communication. We've created everything except setting the connect methods in the main file. Then, when we started using connect methods, we realized that building classes for qobjects would make more sense.

From that point on, we restarted the project (reused most of the previous codes) and created qobjects. As a result, we succeeeded in creating the whole project in a CurrencyConverter class, and also created a CurrencyButton class which sends a custom signal when clicked.

What's more, we've added a few more improvements to the project. Firstly, instead of fetching currency rates from api.fixer.io every time it converts a money, we decided to cache the rates at the beginning of the project. We found out that we could get both the rates for USD/EUR/GBP/CNY by one http request, too (which would be super fast). So, we implemented the program in that way.

Note: We cached the results not only we wanted but also did the fixer.io on their website by telling "Please cache results whenever possible."

Out reader class sends a get request to "https://api.fixer.io/latest?base=TRY" to get all the results on base TRY, then extracts the USD, EUR, GBP, CNY rates from the json response by using regexes.

There was a problem on Halit's hotspot internet, that it did not connect to api.fixer.io, so he created a simple website http://fixer.corupta.net which curled this address and returned a custom fallback rate when the connection failed. But, we found out that the problem was not on api.fixer.io but only on Halit's internet. So, he used his website for testing.

What's more, after caching the results, we would also be able to calculate conversion not only when USD/EUR/GBP/CNY buttons were clicked, but also when the input value for TL amount changed, too. So, we've also implemented that. (Doing something similar, would broke the program (would create lag) without caching results, since there would be a new http request everytime the value changed even a little bit causing too much http requests.)

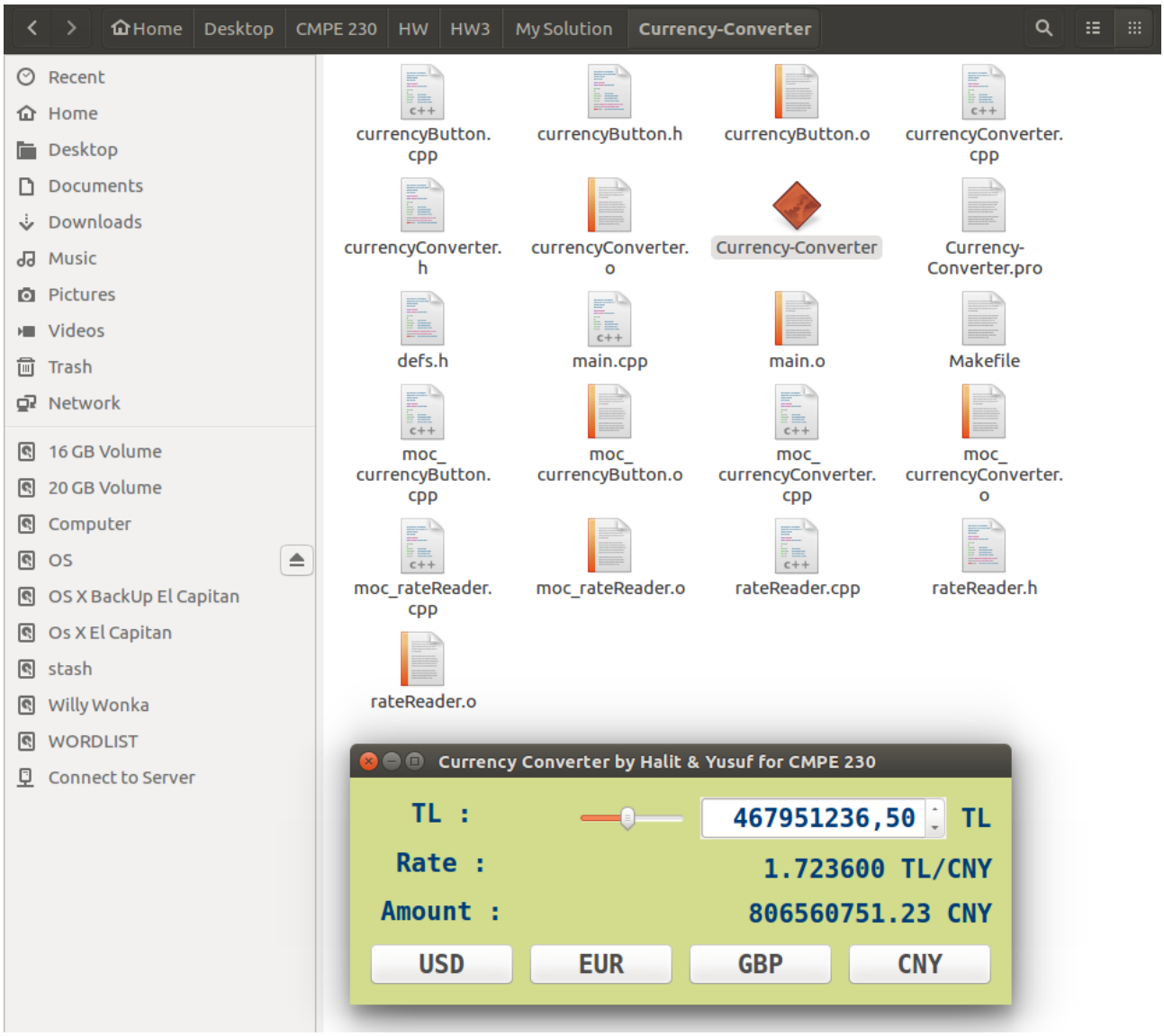
After finishing the project we've checked out the QColor and QFont components. Experimented a little bit with them and decided on a font and some colors, we've somehow liked. Also, we used some margin and spacing to improve the program visually.

Finally, we've added a slider next to the input bar to make it a more useful program. However, in order to make the program take a range from 0 to 1 billions, and also 2 decimal precision, We would need a range of 10^11 on the slider. Although, the slider would take an integer value only as the range argument, we found a way to fix this issue. Each tiny bit change of the slider would mean 0.50 not 0.01, and that way its range would be from 0 to 2 billions which would fit in an integer. To do so, we had to implement a custom function which takes the changed value from the slider and multiplies it with 0.50 to get the real Turkish lira amount.

What We Learned

We have gained some practice on QT, and learned how to build QT objects and why. We've also dealt with http requests and json parsing in cpp, for the first time. We've researched and learned about some useful QT classes, such as QDoubleSpinBox.

Screenshot of the Program



Halit Özsoy, Yusuf Kalaycı
2016400141, 2016400357

20.12.2017
CMPE 230 – HOMEWORK 3

Documentation

main.cpp

GNU nano 2.5.3 File: main.cpp

```cpp
#include <QWidget>
#include <QApplication>
#include <QtGui>
#include "currencyConverter.h"
#include <iostream>

int main(int argc, char *argv[]) {
        QApplication app(argc, argv);

        CurrencyConverter currencyConverter;
        // program works in the constructor of this class

        return app.exec();
}
```

CurrencyConverter Class (currencyConverter.h)

GNU nano 2.5.3 File: currencyConverter.h

```cpp
#include <QtGui>
#include "rateReader.h"
#include "currencyButton.h"

// that is the main class
class CurrencyConverter : public QObject {
        Q_OBJECT

private:
        // current / last entered amount in turkish liras - default is 0
        double tryAmount;
        // current / last chosen currency type (an enum either of USD/EUR/GBP/CNY) - default is USD
        CurrencyType currentCurrency;

        // a class to get exchange rates for try from the api.fixer.io
        RateReader *myReader;

        // a slider to input tryAmount
        QSlider* inputSlider;
        // an input box to input tryAmount
        QDoubleSpinBox *inputNumber;
        // QLabels in which the results of the exchange to be presented
        // ratenumber is the rate of the chosen currency type over turkish liras
        // amountnumber is the tryAmount converted to the chosen currency type from turkish liras
        QLabel *rateNumber, *amountNumber;

        // the four buttons in buttom (USD/EUR/GBP/CNY)
        // since currency type is an enum of 0=USD, 1=EUR, 2=GBP, 3=CNY
        // button[USD] is the button with the label USD
        CurrencyButton* buttons[4];

        // a method that creates and returns a QLabel with the given label string, aligment and width
        static QLabel* createLabel(QString s, Qt::Alignment alignment, int width = 0);

private slots:
        // called when tryAmount changed / currentCurrency is changed / getting rates from the api succeeded.
        // rateNumber and amountNumber is calculated and presented.
        void calculateResult();
        // called when tryAmount is changed via spin box => changes tryAmount and calls calculateResult
        void updateInputAmount(double newValue);
        // called when tryAmount is changed via slider => multiplies the slider change by 0.50 and calls updateInputAmount with this value.
        // 0.50 is a number multiplier to have the slider(uses int) work with 1 billion range on a double with 2 decimals.
        void divideUpdateInputAmount(int intValue);
        // called when one of the buttons on the bottom are pressed
        // changes currentCurrency to the pressed button's currency and calls calculateResult
        void updateSelectedButton(CurrencyType newCurrency);

public:
        // constructor, where the whole widget is prepared and ran.
        CurrencyConverter(QWidget *parent = 0);
};
```

Halit Özsoy, Yusuf Kalaycı
2016400141, 2016400357

CurrencyButton Class (currencyButton.h)

```
 GNU nano 2.5.3                                          File: currencyButton.h

#include <QPushButton>
#include "defs.h"

// a qpushbutton class that recieves the width, currency type and currency name
// instead of the classic clicked signal, it gives a signal with the currency type of the button
   currency type is an enum that is either of USD/EUR/GBP/CNY
   currency name is its string representation
class CurrencyButton : public QPushButton {
        Q_OBJECT

        private:
                // currencyType of this button
                CurrencyType currency;

        private slots:
                // function to be called when clicked, and emits a signal with the currency of this button
                void doWhenClicked();

        signals:
                // the signal to be emitted when this button is clicked
                void currencySelected(CurrencyType currency);
        public:
                // constructor that constructs a QPushButton, using the currencyName and width arguments
                // also sets the currency of this button class
                CurrencyButton(CurrencyType currency, QString currencyName, int width = 0, QWidget* parent = 0);

};
```

RateReader Class (rateReader.h)

```
 GNU nano 2.5.3                                          File: rateReader.h

#include <QtGui>
#include <QNetworkAccessManager>
#include <QNetworkRequest>
#include <QNetworkReply>
#include "defs.h"

// NOTE: instead of fetching rates on each button click, we've decided to cache the results and use it on all button clicks
// In fact, after caching the results, we were able to have the program calculate the conversion results not only when buttons were clicked
// but also when the input for turkish liras were changed too. Since the program doesn't need to send a new http request for each conversion
// which would create a big overhead, thus require less conversion (only on button clicks).
// Also, fixer.io specifically asked for "Please cache results whenever possible."
class RateReader : public QObject {
        Q_OBJECT

        public:
                // constructor that sets connecting to 0, and makes the rate request to api.fixer.io
                // requests api.fixer.io/latest?base=TRY to fetch the rates for all currency types from TRY by one http request
                // so that both api.fixer.io's servers and this program would need less internet connection & work faster.
                // (good for fixer.io and they have specifically wanted such usage by telling "Please cache results whenever possible.")
                RateReader();
                // a function that returns the rate for a given currency type (USD/EUR/GBP/CNY)
                double getRate(CurrencyType);
                // returns the name of a given currency type (USD/EUR/GBP/CNY) as a string (USD -> "USD", etc.)
                QString getCurrencyName(CurrencyType currency);
                // true until the connection & getting rates from api.fixer.io is completed.
                bool connecting;

        signals:
                // a signal emitted when the connection is completed => so that the main object can call calculation functions from now on.
                void connectionCompleted();

        private slots:
                // called when the network connection finished with a reply (api.fixer.io)
                // extract USD, EUR, GBP, CNY rates from the json reply, using regexes.
                void replyFinished(QNetworkReply *reply);

        private:
                // an array to store fetched rates for currencies (USD/EUR/GBP/CNY)
                // note that CurrencyType is an enum where USD=0, EUR=1, GBP=2, CNY=3
                double rates[4];
                // NetworkAccessManager used to communicate via api.fixer.io
                QNetworkAccessManager *manager;
};
```