Министерство образования и науки РФ

Федеральное государственное автономное образовательное учреждение

высшего профессионального образования «НИТУ «МИСиС»

Институт ИТАСУ

Кафедра ИК

Отчёт по курсовой работе

по дисциплине

«Технологии программирования»

на тему приложение «Симулятор трейдера»

Выполнил

студент группы БПМ-19-1

Балаев А. А.

Проверил:

Полевой Д. В.

Москва 2020 г.

Оглавление

Задача

Написать пользовательское приложение являющееся симулятором спекуляции на инвестиционной бирже, в котором есть возможность покупки ценных бумаг (акций и облигаций) и их последующей продаже. Доход вычисляется из разности цен на момент продажи и покупки (купоны и дивиденды в этой программе не учитываются).

Техническое задание

Приложение осуществляет работу с 9:00 до 19:00, в остальное время возможность продажи и покупки заблокирована. Приложение может хранить данные об инвестициях пользователя, то есть количество определенный ценных бумаг их стоимость на текущий момент и на момент покупки. Также хранится информация о самих бумагах, их название, и цена. Помимо этого у пользователя есть возможность ввести свое имя и фамилию. Эти данные сохранятся в файл и используются при дальнейшей работе с приложением. Цена всех бумаг устанавливается автоматически, генерируясь случайным образом на каждую минуту времени работы «биржи» на 1 день. Программа высчитывает потенциальную выгоду при

покупке и весь доход с продаж. Помимо этого, можно посмотреть график изменений цен на каждую из доступных бумаг.

Пользовательская документация

После запуска программы перед пользователем открывается окно «Рынок», в котором есть 2 таблицы с облигациями и акциями. Цены на каждый из видов ценных бумаг автоматически обновляются каждую минуту. Для получения информации в данный момент времени пользователю необходимо нажать на кнопку «Обновить». Так же в этом поле есть ComboBox в котором можно выбрать любую из ценных бумаг и купить их в количестве введеным пользователем, предварительно внести число в поле editLine, а после нажав на кнопку «Купить». Или можно посмотреть график изменения цены выбранной бумаги до данного момента времени.

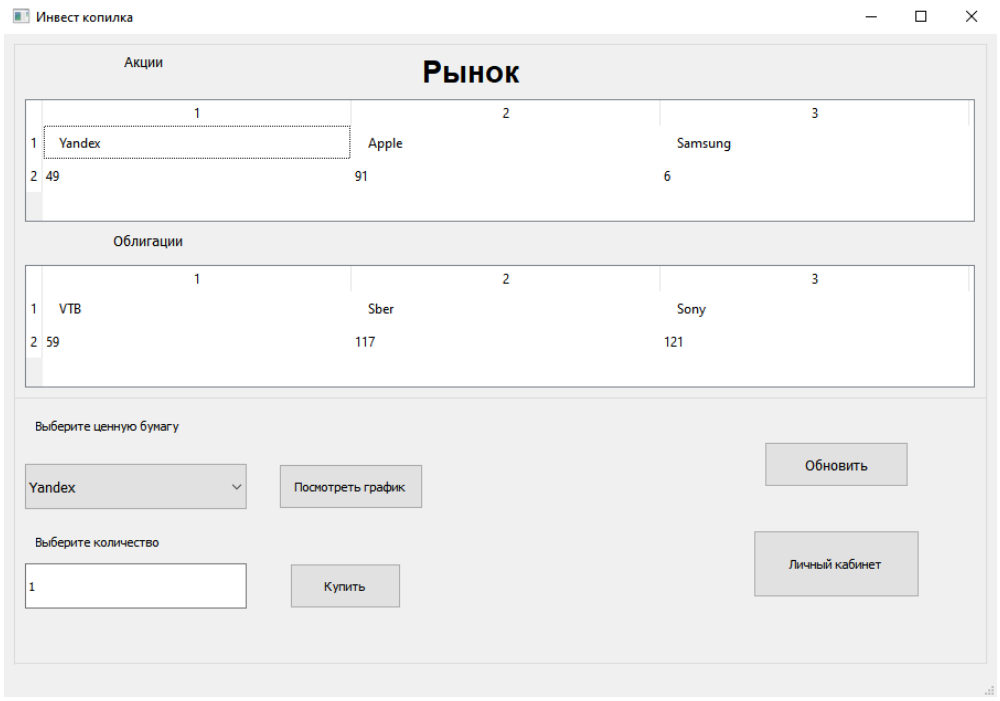


Рис.1. Окно «Рынок»

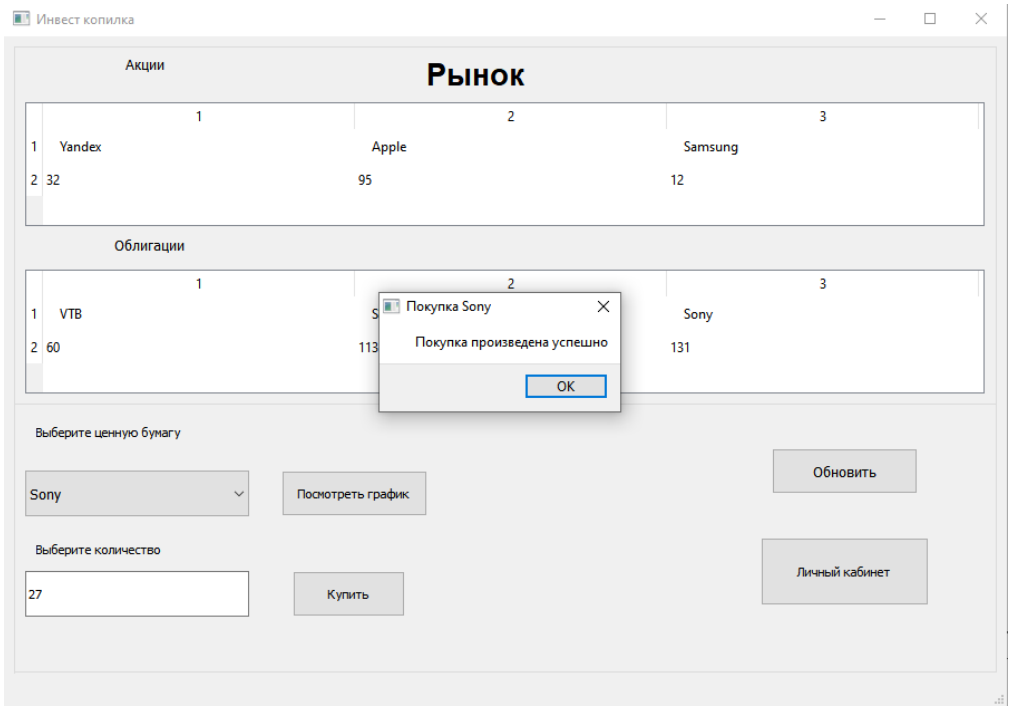


Рис.2. Осуществление покупки

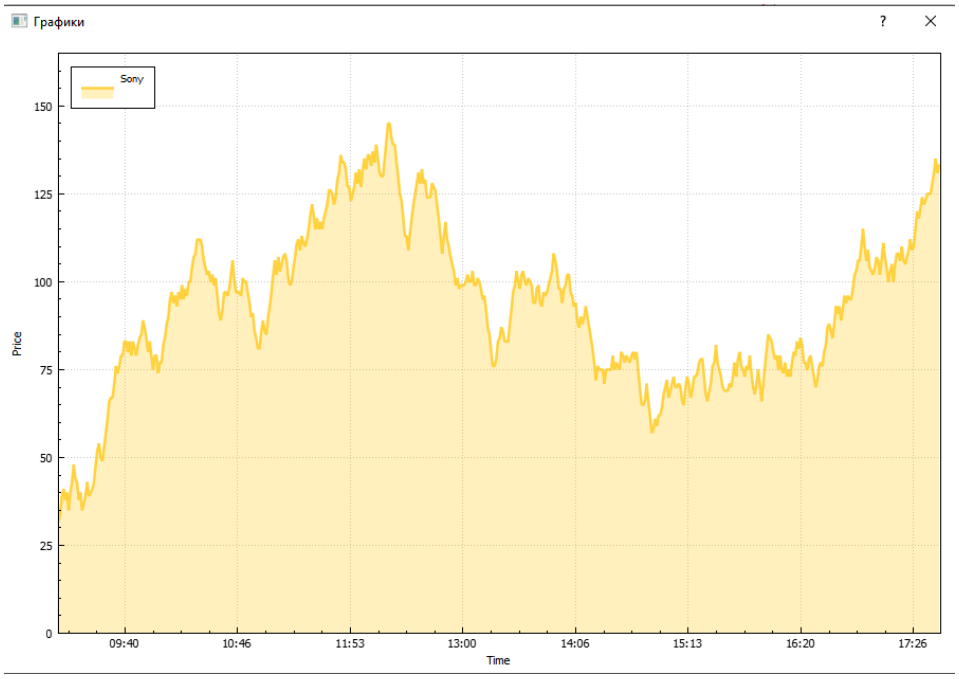


Рис.3. Просмотр графика изменения цен

Также на этом окне можно перейти в личный кабинет, нажав на одноименную кнопку.

В окне личный кабинет Пользователь может ввести свое имя и фамилию в специальном окне нажав на кнопку «Изменить данные пользователя» и чтобы они были внесены в программу необходимо нажать на кнопку «Обновить». Также в данном окне есть два lable в которых указывается потенциальная выгода при совершении сделки. И вся копилка продаж.

Можно также вернуться в предыдущее окно нажав на кнопку «Рынок».

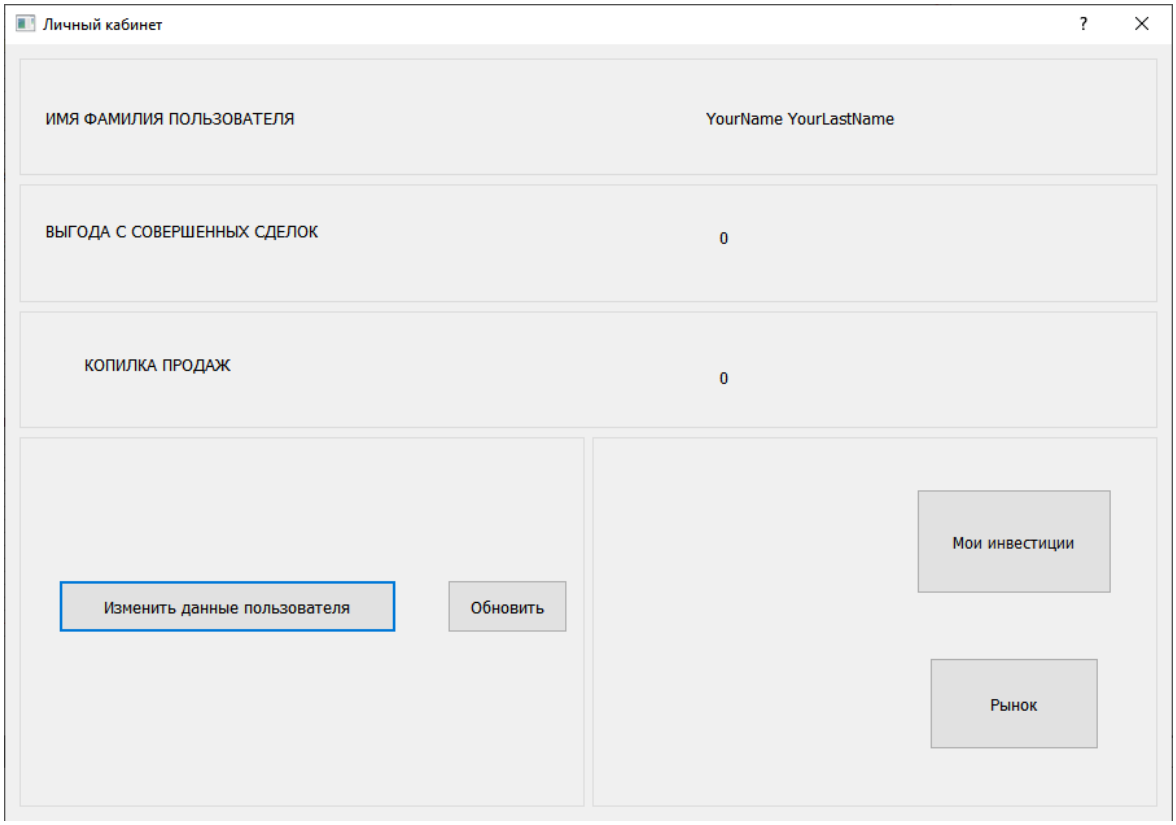


Рис.4. Окно «Личный кабинет»

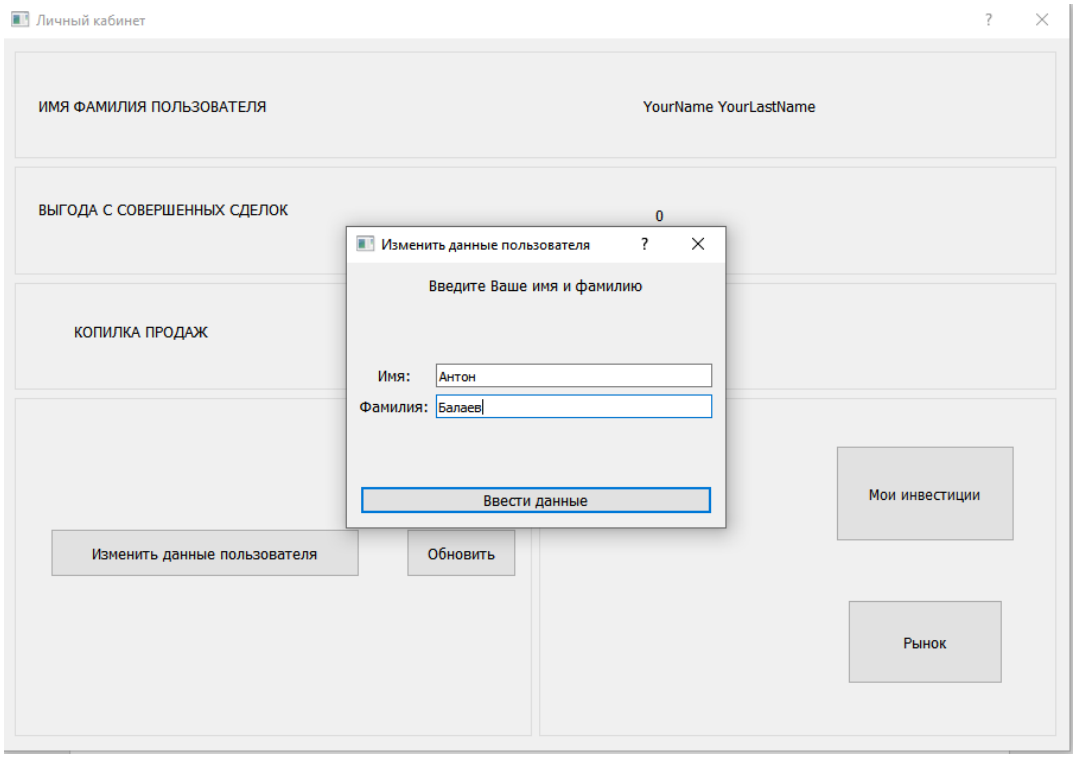


Рис.5. Изменение данных о пользователе

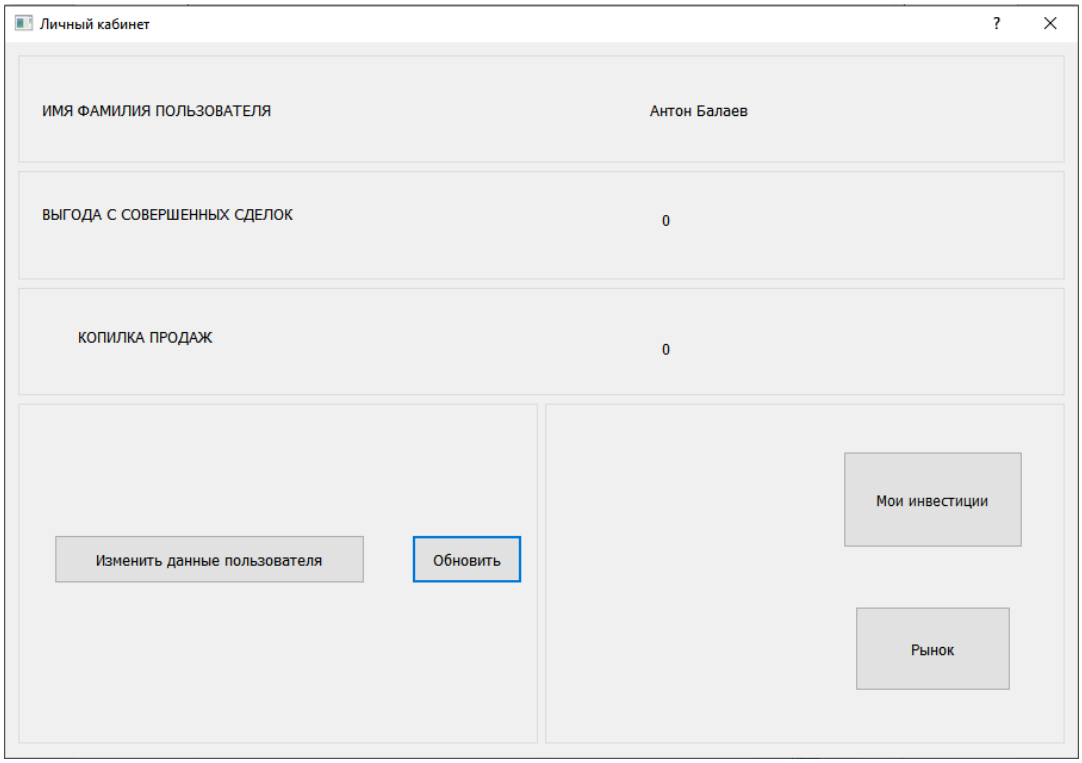


Рис.6. Внесение изменений

Нажав на кнопку «Мои инвестиции» пользователь перейдет в новое окно, в котором показана таблица где отображаются все купленные бумаги и в каком объеме. В этом же окне пользователь может осуществить продажи купленных бумаг, выбрав бумагу и количество. Чтобы внести изменения в программу после продаже бумаги необходимо нажать на кнопку «Обновить».

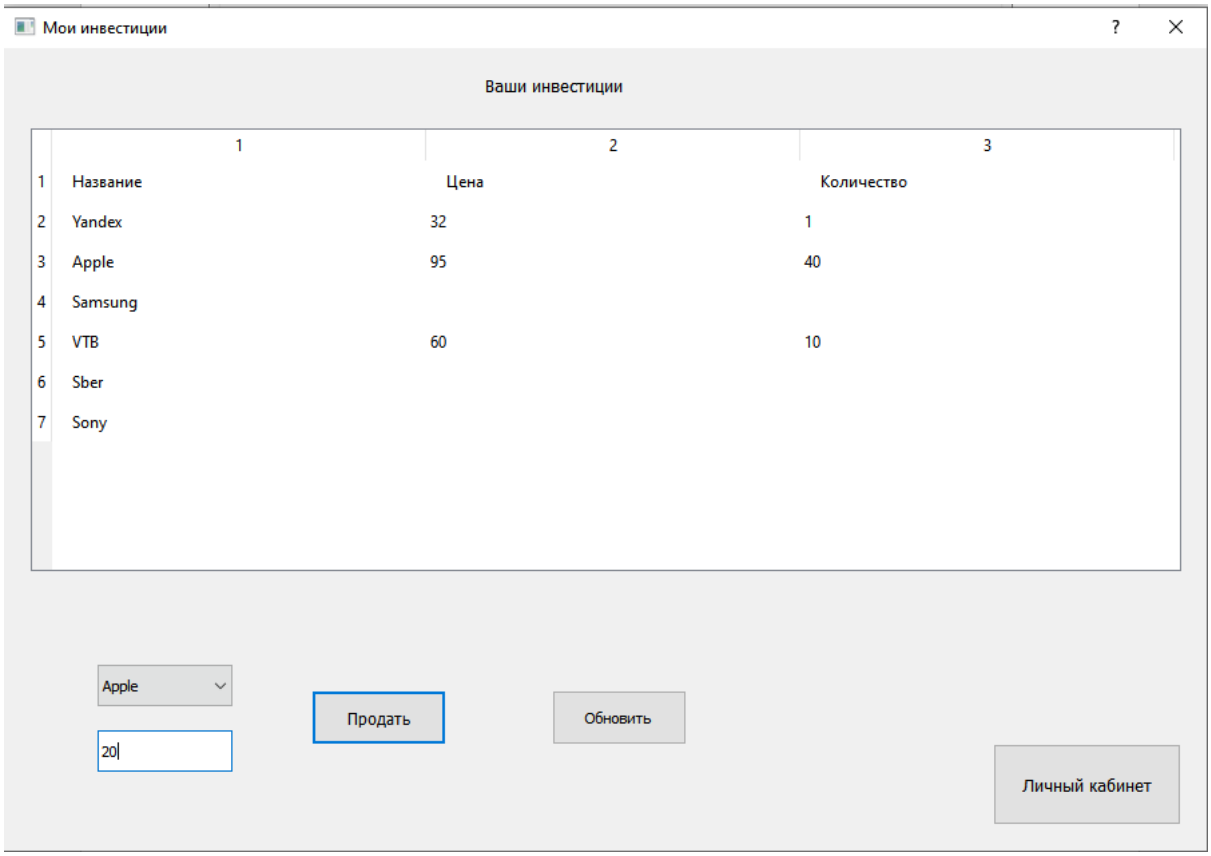


Рис.7. Окно «Мои инвестиции»

Вся финансовая биржа работает с 9:00 по 19:00, в остальное время цены на ценные бумаги не доступны, как и графики их изменения.

Техническое описание

Приложение реализовано на основе фреймворка Qt.

Основные функции

- Автоматическое заполнение ячеек таблицы с ценой акций и облигаций.
- Построения графика исходя из данных занесенных в файл.
- Осуществление покупки и продажи ценных бумаг.

- Внесение данных пользователя (имени и фамилии) с возможностью сохранения или изменения этих данных.
- Подсчет текущего дохода с купленных бумаг
- Подсчет дохода с продаж

Документация

Класс BoughtCollection

Класс описывает коллекцию купленных бумаг

#include <BoughtCollection.h>

Открытые члены

	BoughtCollection (**PaperCollection** *papers, **User** *user)
	Конструктор принимающий массив бумаг, которые существуют и информацию о пользователе
int	**getCount** () const
	Метод возвращающий количество купленных бумаг
Investment	**getInvestment** (int index) const
	Метод возвращающий экземпляр класса **Investment** по индексу
Investment	**getInvestment** (QString name) const
	Метод возвращающий экземпляр класса **Investment** по имени
int	**getNumberInvesments** (QString name) const
	Метод возвращающий количество купленных бумаг по ее имени

void	**deleteInvestment** (int n)
	Метод удаляющий инвестиции при их продаже по индексу

void	**deleteInvestment** (QString name)
	Метод удаляющий инвестиции при их продаже по индексу

void	**setInvestments** (QVector< QStringList > data)
	Метод заносящий информацию об инвестициях полученную из файла

void	**setInvestmentByName** (double price, double sum, int number, QString name)
	Метод обновл¤ющий данные существующих инвестиций

void	**setCountInvestment** (int num)
	Метод устанавливающий количество ценных бумаг

void	**addInvestment** (const **Investment** &invest)
	Метод добавляющий инвестиции

void	**sellInvestment** (QString name, int number)
	Метод осуществляющий продажу инвестиции

void	**profit** ()
	Метод подсчитывающий выгоду с совершенной сделки

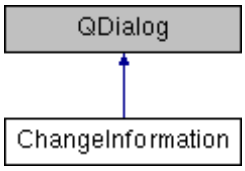
bool	**checkHaveInvestmentByName** (QString name)
	Метод проверяющий наличие инвестиции по ее имени

Класс ChangeInformation

Класс описывающий изменения данных об имени и фамилии пользователя

#include <changeinformation.h>

Граф наследования: ChangeInformation:



Открытые члены

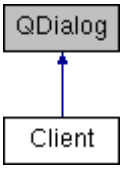
	ChangeInformation (QWidget *parent=nullptr, **User** *user=nullptr)
	Конструктор принимающий родительский класс и информацию о пользователе

Класс Client

Класс описывает окно "Личный кабинет".

#include <client.h>

Граф наследования: Client:



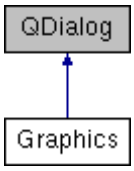
Открытые члены

	Client (QWidget *parent=nullptr, **User** *user=nullptr, **BoughtCollection** *investments=nullptr, **Paper Collection** *papers=nullptr)
	Конструктор принимающий родительский класс, данные о пользователе, информацию о купленных бумагах и о существующих ценных бумагах

Класс Graphics

Класс описывающий построение графиков в окне "Графики". Подробнее...
#include <**graphics.h**>

Граф наследования: Graphics:



Открытые члены

	Graphics (QWidget *parent=nullptr, int index=0)
	Конструктор принимающий родительский класс и индекс бумаги

Класс Investment

Класс хранящий информацию об инвестициях Подробнее...

#include <**Investment.h**>

Открытые члены

	Investment (double sum=0, int num=0, double price=0, QString paperName="")
	Конструктор класса, принимающий сумму, кол-во, цену и имя бумаги инвестиции

	Investment (const **Investment** &other)
	Копирующий конструктор

double	**getSum** () const
	Метод, возращающий сумму инвестиции

int	**getNumberPapers** () const
	Метод, возращающий кол-во купленных бумаг инвестиции

double	**getPrice** () const
	Метод, возращающий цену бумаг инвестиции

QString	**getPaperName** () const
	Метод, возращающий имя бумаг инвестиции

void	**setSum** (double sum)
	Метод, устанавливающий сумму инвестиции

void	**setNumberPapers** (int num)
	Метод, устанавливающий кол-во бумаг инвестиции

void	**setPrice** (double priceBuy)
	Метод, устанавливающий цену бумаги инвестиции

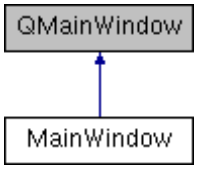
void	**setPaperName** (QString paperName)
	Метод, устанавливающий имя бумаги инвестиции

Класс MainWindow

Класс описывает логику работы основного окна. Основное окно содержит 2 вкладки: для работы с клиентами и для работы с услугами/тарифами.

#include <**mainwindow.h**>

Граф наследования: MainWindow:



Открытые члены

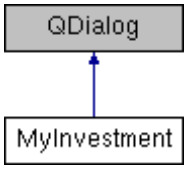
	MainWindow (QWidget *parent=nullptr)

Класс MyInvestment

Класс содержащий данные о купленных бумагах пользователя.

#include <myinvestment.h>

Граф наследования: MyInvestment:



Открытые члены

	MyInvestment (QWidget *parent=nullptr, **BoughtCollection** *investments=nullptr, **User** *user=nullptr, **PaperCollection** *papers=nullptr)
	Конструктор класса, принимающий родительский класс, купленные бумаги, информацию о пользователе и существующих бумагах

Класс PaperCollection

Класс описывающий коллекцию всех ценных бумаг.

#include <PaperCollection.h>

Открытые члены

	PaperCollection ()
	Конструктор по умолчанию
int	**getCount** () const
	Метод, возвращающий количество ценных бумаг

SecurityPaper	getPaper (QString name) const
	Метод, возвращающий экземпляр класса SecurityPaper по имени бумаги
SecurityPaper	getPaper (int index) const
	Метод, возвращающий экземпляр класса SecurityPaper по индексу бумаги
double	getPriceByName (QString name) const
	Метод, возвращающий цену бумаги по ее имени
void	setPrice (QString name, double sum)
	Метод, устанавливающий цену бумаги
void	addPaper (SecurityPaper *paper)
	Метод добавляющий бумагу в коллекцию
void	deletePaper (int n)
	Метод удаляющий ценную бумагу из коллекции

Класс SecurityPaper

Класс содержащий информацию об ценных бумагах.

#include <SecurityPaper.h>

Открытые члены

	SecurityPaper (QString name="Paper", double price=0)
	Конструктор класса, принимающий имя бумаги и ее цену

	SecurityPaper (**SecurityPaper** &other)
	Копирующий конструктор

QString	**getName** ()
	Метод, возвращающий имя бумаги

double	**getPrice** ()
	Метод, возвращающий цену бумаги

void	**setName** (QString name)
	Метод, устанавливающий имя бумаги

void	**setPrice** (double price)
	Метод, устанавливающий цену бумаги

Класс User

Класс содержащий данные о пользователе Подробнее...

#include <**User.h**>

Открытые члены

	User (QString firstName="Name", QString lastName="lastName", double profit=0, double moneyBox=0, int numberPapers=0)
	Конструктор пользователя, приимающий его имя, фамилию, выгоду с купленных бумаг, копилку продаж и кол-во имеющихся бумаг

double	**getProfitNow** ()
	Метод, возращающий выгоду с купленных бумаг

double	**getMoneyBox** ()
	Метод, возращающий сумму в копилке продаж
QString	**getFirstName** ()
	Метод, возращающий имя пользователя
QString	**getLastName** ()
	Метод, возращающий фамилию пользователя
int	**getNumberPapers** ()
	Метод, возращающий кол-во купленных бумаг
void	**setProfitNow** (double profit)
	Метод, устанавливающий кол-во бумаг
void	**addToMoneyBox** (double sum)
	Метод, добавляющий новую сумму в копилку продаж
void	**setFirstName** (QString firstName)
	Метод, устанавливающий имя пользователя
void	**setMoneyBox** (double moneyBox)
	Метод, устанавливающий сумму копилки продаж пользователя
void	**setLastName** (QString lastName)
	Метод, устанавливающий фамилию пользователя
void	**setNumberPapers** (int numberPapers)
	Метод, устанавливающий кол-во бумаг пользователя

Класс utils

Класс содержащий методы для упрощения работы Подробнее...

#include <**utils.h**>

Открытые статические члены

static std::string	**qStringStdString** (QString string)
	Метод, переводящий строку формата QString в формат StdString.
static QString	**StdStringtoQString** (std::string string)
	Метод, переводящий строку формата StdString в формат QString.
static void	**StrToMassive** (const std::string s, std::vector< double > *massive)
	Метод, преобразующий строку полученную из файла в вектор
static std::vector< std::string >	**parseStr** (std::ifstream &file)
	Метод, парсящий строки из файла пользователя и сохраняющий их в ветор строк
static std::vector< std::vector< double > >	**parseData** (std::ifstream &file)
	Метод, парсящий строки из файла с данными о измененях цен бумаг и сохраняющий из вектор строк
static void	**createUserFile** ()
	Метод, создающий файл пользователя, если он еще не создан

static std::vector< std::string >	**getDataFromUserFile** ()
	Метод, возвращающий данные о пользователе, которые он прочитал из файла, с помощью метода ParseStr.
static int	**whatTimeIndex** ()
	Метод, возращающий минуту с начала открытия биржи, которая соответсвует индесу в массиве с ценнами на бумаги в эту минуту
static void	**newPrice** ()
	Метод, генерирующий цены на бумаги
static void	**addUserDataToFile** (QString name, QString lastName, double profit, double moneyBox, int numberPapers)
	Метод, обновляющий информацию о пользователя в файле
static void	**createInvestmentFile** ()
	Метод, создающий файл инвестиций, если он еще не создан
static void	**addInvestmentFile** (**BoughtCollection** *investments)
	Метод, добавляющий в файл новую инвестицию
static QVector< QStringList >	**getInvestmentsFromFile** ()
	Метод, возвращающий вектор, хранящий информацию о инвестициях

static bool	**isEmpty** (std::ifstream &pFile)
	Метод, проверяющий пустой ли файл

Руководство по сборке

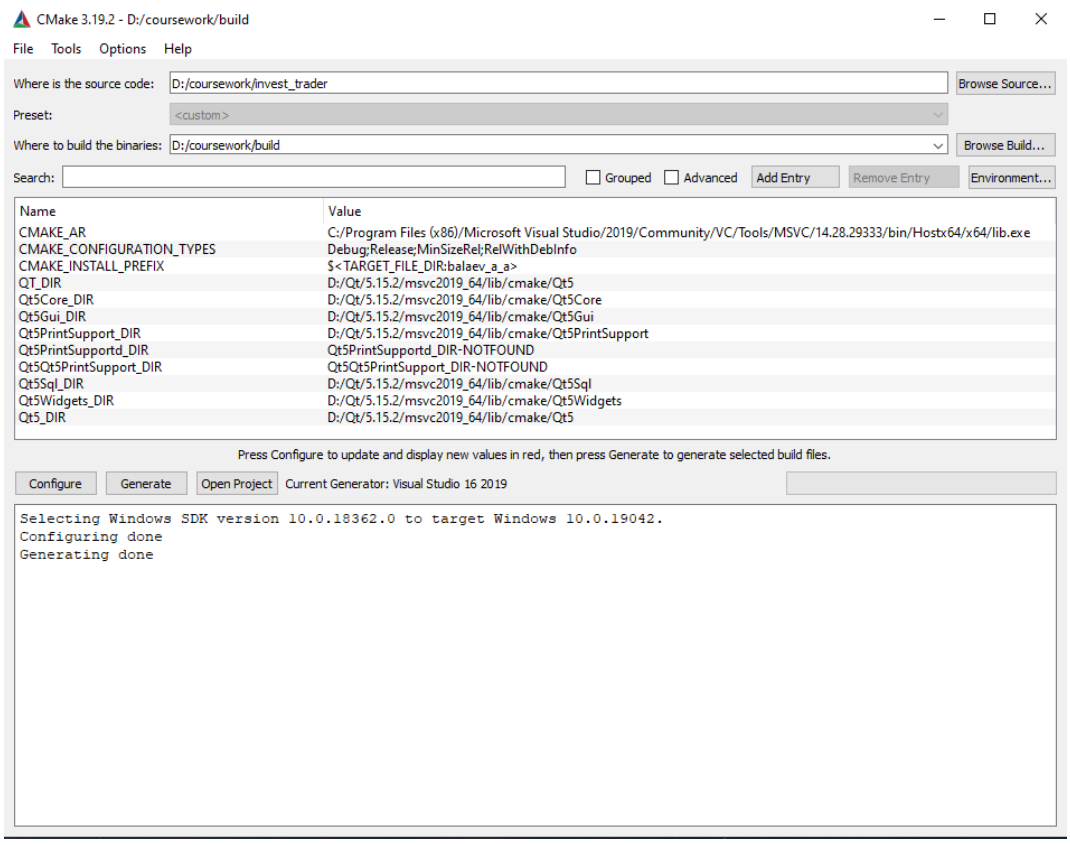
1. Загрузите данные с репозитория:

https://mysvn.ru/Antuanidze/coursework/

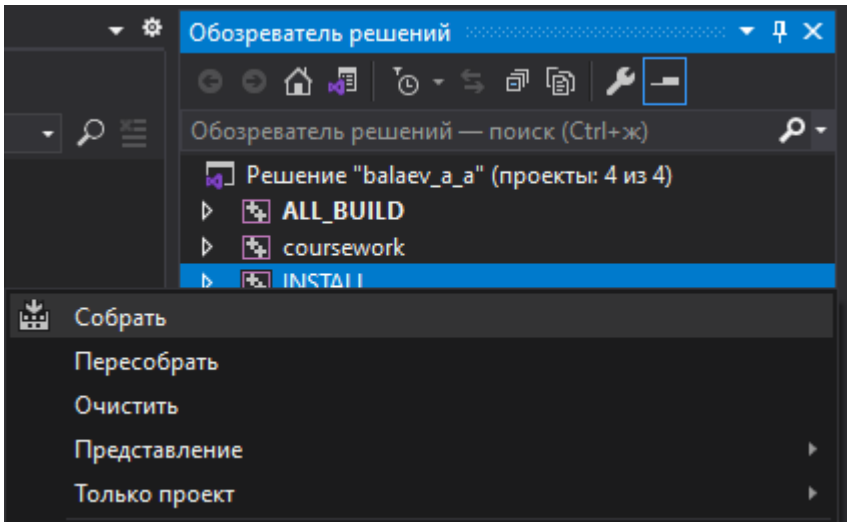
2. В папке, которая содержит скаченную папку, создайте папку build.

3. В файле CMakeLists.txt при необходимости можно указать CMAKE_INSTALL_PATH. Если при сборке не удается найти директорию Qt (не переменная среда), то также необходимо указать путь до папки с библиотеками.

4. Запустите CMake GUI. В первую строку введите адрес скачанной папки. Во вторую строку адрес папки build.



5. Через IDE (Visual Studio) собрать INSTALL часть проекта.



6. Теперь можно переходить в директорию установки и пользоваться приложением (coursework.exe).