Правительство Российской Федерации

Федеральное государственное автономное образовательное учреждение
высшего образования «Национальный исследовательский университет
«Высшая школа экономики»

Кафедра «Компьютерная безопасность»

ОТЧЕТ
К ЛАБОРАТОРНОЙ РАБОТЕ №78

по дисциплине

«Языки программирования»

Работу выполнил
студент группы СКБ-222 _____ Д.А. Спиридонов
подпись, дата

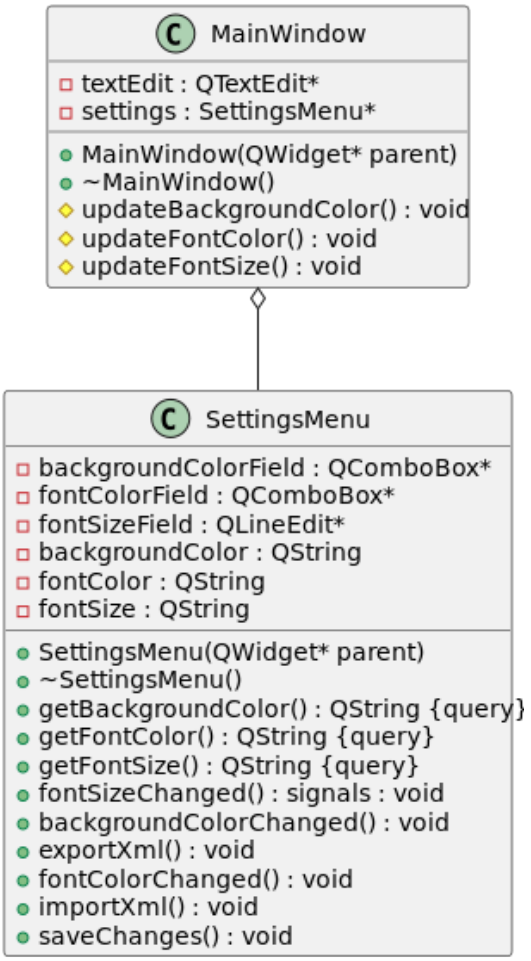
Работу проверил _____ С.А. Булгаков
подпись, дата

Москва 2023

Содержание

Приложение А

А.1 UML-диаграммы классов



Приложение В

В.1 Исходный код MainWindow.h

```
#ifndef __MAINWINDOW_H__
#define __MAINWINDOW_H__

#include <QMainWindow>
#include <QMenuBar>
#include <QWidget>
#include <QTextEdit>
#include <QLabel>

#include "SettingsMenu.h"


class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    MainWindow(QWidget* parent = nullptr);
    ~MainWindow() = default;
protected slots:
    void updateFontSize();
    void updateFontColor();
    void updateBackgroundColor();
private:
    SettingsMenu* settings;
    QTextEdit*    textEdit;
};

#endif // __MAINWINDOW_H__
```

B.2 Исходный код MainWindow.cpp

```
#include "MainWindow.h"


MainWindow::MainWindow(QWidget* parent)
    : QMainWindow(parent)
{
    this->setWindowTitle(tr("Notepad"));

    QTabWidget* tabs = new QTabWidget(this);
    this->setCentralWidget(tabs);

    settings = new SettingsMenu;
    textEdit = new QTextEdit;

    tabs->addTab(textEdit, "Editor");
    tabs->addTab(settings, "Settings");

    QAction* importSettingsAction = new QAction("Import settings");
    QAction* exportSettingsAction = new QAction("Export settings");

    auto fileMenu = menuBar()->addMenu(tr("File"));
    fileMenu->addAction(importSettingsAction);
    fileMenu->addAction(exportSettingsAction);

    connect(importSettingsAction, SIGNAL(triggered()), settings, SLOT(importXml()));
    connect(exportSettingsAction, SIGNAL(triggered()), settings, SLOT(exportXml()));

    connect(settings, SIGNAL(fontSizeChanged()), this, SLOT(updateFontSize()));
    connect(settings, SIGNAL(fontColorChanged()), this, SLOT(updateFontColor()));
    connect(settings, SIGNAL(backgroundColorChanged()), this, SLOT(updateBackgroundColor()));

    this->updateFontSize();
    this->updateFontColor();
    this->updateBackgroundColor();

    this->resize(640, 480);
}


void MainWindow::updateFontSize()
{
    QFont font = textEdit->currentFont();
    font.setPointSize(settings->getFontSize().toInt());
    textEdit->setFont(font);
    textEdit->update();
}


void MainWindow::updateFontColor()
{
    QString tmp = textEdit->toPlainText();
    textEdit->clear();
    textEdit->setTextColor(QColor(settings->getFontColor()));
    textEdit->insertPlainText(tmp);
    textEdit->update();

}


void MainWindow::updateBackgroundColor()
{
    QPalette p = textEdit->palette();
    p.setColor(QPalette::Base, settings->getBackgroundColor());
```

```
    textEdit->setPalette(p);
    textEdit->update();

}
```

B.3 Исходный код SettingsMenu.h

```
#ifndef __SETTINGS_MENU_H__
#define __SETTINGS_MENU_H__

#include <QVBoxLayout>
#include <QHBoxLayout>
#include <QFormLayout>
#include <QFileDialog>
#include <QPushButton>
#include <QLineEdit>
#include <QComboBox>
#include <QLabel>
#include <QWidget>

#include <QXmlStreamReader>
#include <QXmlStreamWriter>
#include <QDebug>

class SettingsMenu : public QWidget
{
    Q_OBJECT
public:
    SettingsMenu(QWidget* parent = nullptr);
    ~SettingsMenu() = default;

    QString getFontSize() const;
    QString getFontColor() const;
    QString getBackgroundColor() const;
public slots:
    void saveChanges();
    void importXml();
    void exportXml();
signals:
    void fontSizeChanged();
    void fontColorChanged();
    void backgroundColorChanged();
private:
    QString fontSize = "14";
    QString fontColor = "Black";
    QString backgroundColor = "White";

    const QStringList colors = {
        "White", "Black","Red",
        "Dark red", "Green", "Dark green",
        "Blue", "Dark blue", "Cyan",
        "Dark cyan", "Magenta", "Dark magenta",
        "Yellow", "Dark yellow", "Gray",
        "Dark gray", "Light gray"
    };

    QLineEdit* fontSizeField;
    QComboBox* fontColorField;
    QComboBox* backgroundColorField;
};

#endif // __SETTINGS_MENU_H__
```

B.4 Исходный код SettingsMenu.cpp

```
#include "SettingsMenu.h"


SettingsMenu::SettingsMenu(QWidget* parent)
    : QWidget(parent)
{
    QVBoxLayout* main_layout = new QVBoxLayout(this);

    QFormLayout *formLayout = new QFormLayout(this);

    fontSizeField = new QLineEdit;
    QLabel* fontSizeLabel = new QLabel("Font size: ");
    fontSizeField->setValidator( new QIntValidator(6, 100, this));
    fontSizeField->insert(fontSize);
    formLayout->addRow(fontSizeLabel, fontSizeField);

    QLabel* fontColorLabel = new QLabel("Font color: ");
    fontColorField = new QComboBox;
    fontColorField->addItems(colors);
    fontColorField->setCurrentText(fontColor);
    formLayout->addRow(fontColorLabel, fontColorField);

    QLabel* backgroundColorLabel = new QLabel("Background color");
    backgroundColorField = new QComboBox;
    backgroundColorField->addItems(colors);
    backgroundColorField->setCurrentText(backgroundColor);
    formLayout->addRow(backgroundColorLabel, backgroundColorField);

    main_layout->addLayout(formLayout);

    QPushButton* save_button = new QPushButton(tr("Save changes"));
    connect(save_button, SIGNAL(clicked()), this, SLOT(saveChanges()));
    main_layout->addWidget(save_button);

    main_layout->setAlignment(Qt::AlignCenter);
    this->setLayout(main_layout);
}


QString SettingsMenu::getFontSize() const
{
    return fontSize;
}


QString SettingsMenu::getFontColor() const
{
    return fontColor;
}


QString SettingsMenu::getBackgroundColor() const
{
    return backgroundColor;
}


void SettingsMenu::saveChanges()
{
    fontSize = fontSizeField->text();
    fontColor = fontColorField->currentText();
    backgroundColor = backgroundColorField->currentText();
```

```
    emit fontSizeChanged();
    emit fontColorChanged();
    emit backgroundColorChanged();
}


void SettingsMenu::importXml()
{
    auto filename = QFileDialog::getOpenFileName(this, tr("Open setting‘s file"), "./", tr("Settings (*.xml)

    QFile* file = new QFile(filename);
    if (!file->open(QIODevice::ReadOnly | QIODevice::Text))
    {
        return;
    }
    QXmlStreamReader xml(file);

    xml.readNext();


    while (!xml.atEnd() && !xml.hasError())
    {
        QXmlStreamReader::TokenType token = xml.readNext();
        if (token == QXmlStreamReader::StartDocument)
        {
            continue;
        }
        if(token == QXmlStreamReader::Invalid)
        {
            qDebug() << xml.errorString();
        }
        if (token == QXmlStreamReader::StartElement)
        {
            if (xml.name() == "FontSize")
            {
                xml.readNext();
                fontSize = xml.text().toString();
                fontSizeField->insert(fontSize);
            }
            else if (xml.name() == "FontColor")
            {
                xml.readNext();
                fontColor = xml.text().toString();
                fontColorField->setCurrentText(fontColor);
            }
            else if (xml.name() == "BackgroundColor")
            {
                xml.readNext();
                backgroundColor = xml.text().toString();
                backgroundColorField->setCurrentText(backgroundColor);
            }
            xml.readNext();
        }
    }

    emit fontSizeChanged();
    emit fontColorChanged();
    emit backgroundColorChanged();
}


void SettingsMenu::exportXml()
{
    QFile* file = new QFile("settings.xml");
```

```
    file->open(QIODevice::Text|QIODevice::WriteOnly);
    QXmlStreamWriter xmlWriter(file);

    xmlWriter.setAutoFormatting(true);
    xmlWriter.writeStartDocument();
    xmlWriter.writeStartElement("Settings");

    xmlWriter.writeTextElement("FontSize", fontSize);
    xmlWriter.writeTextElement("FontColor", fontColor);
    xmlWriter.writeTextElement("BackgroundColor", backgroundColor);

    xmlWriter.writeEndElement();
    xmlWriter.writeEndDocument();
    file->close();
}
```

B.5 Исходный код main.cpp

```
#include <QApplication>
#include "MainWindow.h"

int main(int argc, char** argv)
{
    QApplication app(argc, argv);
    MainWindow mainwindow;
    mainwindow.show();
    return app.exec();
}
```