МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Нижегородский государственный университет им. Н.И. Лобачевского»
Национальный исследовательский университет

Институт информационных технологий, математики и механики
Кафедра программной инженерии

ОТЧЕТ

по дисциплине «Разработка мобильных приложений»

«Интеграция QML-интерфейсов и логики приложения на языке C++»

Выполнил:
студент группы 381706-1
Денисов В. Л.

Проверил:
доцент кафедры программной инженерии
Борисов Н. А.

Нижний Новгород
2020.

Содержание

1 Цели

Целью данной лабораторной работы является изучение C++ классов в QML, написания собственных QML-компонентов на языке C++ и использования их в приложении.

2 Постановка задачи

1. Создать класс-счётчик с полем для хранения текущего значения и методами для увеличения значения на единицу и сброса до нуля.
2. Использовать мета-объект класса-счётчика для создания объекта и вызова его методов (использовать функцию main, результат изменения состояния проверять выводом на консоль).
3. Создать приложение с текстовым полем и двумя кнопками. Использовать класс-счётчик в QML: текстовое поле должно отображать текущее значение счётчика, кнопки используются для увеличения значения счётчика на единицу и сброса значения до нуля.
4. Сделать поле со значением счётчика свойством и инициализировать его каким-либо значением при создании объекта в QML.
5. Создать класс, содержащий список из строк. Класс должен содержать методы для добавления строки в список и удаления последней добавленной строки.
6. Создать приложение, позволяющее добавить введённое слово и удалить последнее добавленное с использованием данного класса в QML. Слова сохраняются в нижнем регистре.
7. Реализовать свойство только для чтения, которое позволяет получить список всех строк в виде одной, перечисленных через запятую и использовать это свойство для вывода добавленных строк на экран. Свойство должно моментально реагировать на изменение содержимого списка, первое слово начинается с заглавной буквы.

3 Решение поставленных задач

Создадим проект со стандартной заготовкой приложения, где файлом главной страницы приложения будет являться *FirstPage.qml*, для каждого отдельного задания из постановки задачи тоже будет свой файл.

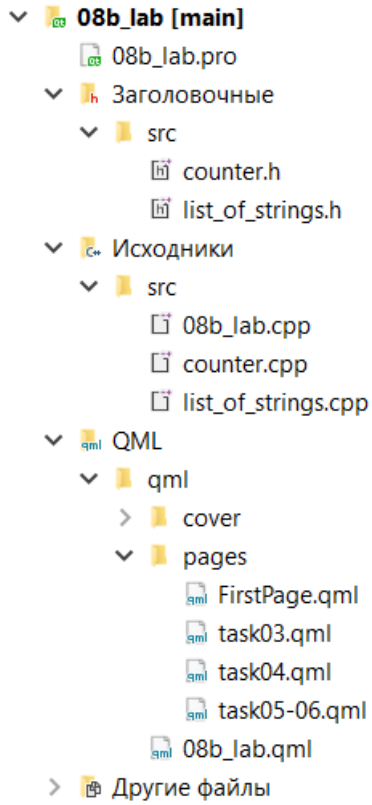

Рисунок 1 Структура проекта.

Вызов демонстрации каждого задания будем производить путем нажатия соответствующей кнопки, расположенной на *FirstPage*. Страница с выбранным заданием будет отправлена в *pageStack*.

1. Класс-счётчик с полем для хранения текущего значения и методами для увеличения значения на единицу и сброса до нуля опишем в заголовочном файле *counter.h*. Реализация класса будет представлена в файле *counter.cpp*. Для дальнейшего использования класса зарегистрируем его в приложении путем вызова метода *qmlRegisterType<Counter>("custom.Counter", 1, 0, "Counter")* в *main()*-функции приложения.

2. Создание мета-объекта (назовём его *metaCounter*) осуществляется путем использования конструкции *Counter::staticMetaObject*, где *Counter* – название класса-счётчика. Затем у мета-объекта *metaCounter* вызываем метод *newInstance()* для создания нового экземпляра класса, результат работы присваиваем указателю *obj* на базовый класс

объектов в *QT* (*QObject*). Выполняем связь сигнала изменения значения счётчика *countChange()* и слота *print()*, который будет выполнять вывод на консоль. По полученному указателю *obj* вызываем методы класса-счётчика: *metaCounter.invokeMethod(obj, "название_метода")*.

```
11:55:35: Запускается /usr/bin/08b_lab ...
[D] unknown:0 - QML debugging is enabled. Only use this in a safe environment.
[D] unknown:0 - Using Wayland-EGL
[D] Counter::print:24 - Counter:  1
[D] Counter::print:24 - Counter:  2
[D] Counter::print:24 - Counter:  3
[D] Counter::print:24 - Counter:  0
```

Рисунок 2 Вызов метода класса-счётчика через мета-объект.

3. Использование класса-счётчика непосредственно на странице в приложении осуществляется следующим образом. Импортируем зарегистрированный функционал *import custom.Counter 1.0*. Создаем элемент *Counter{id: counter}*. У соответствующих кнопок делаем обработчики события *onClicked*, вызывающих методы *counter.increment()* – увеличение значения, или *counter.reset()* – сброс счётчика.

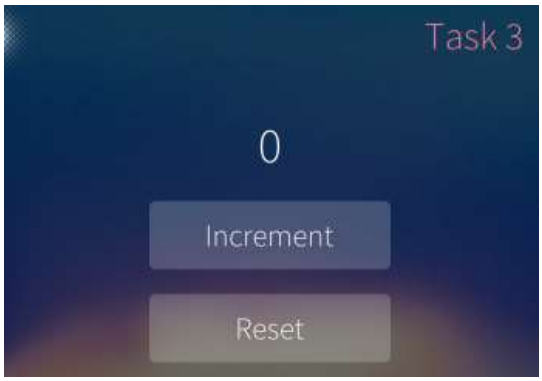

Рисунок 2 Счётчик без первичной инициализации.

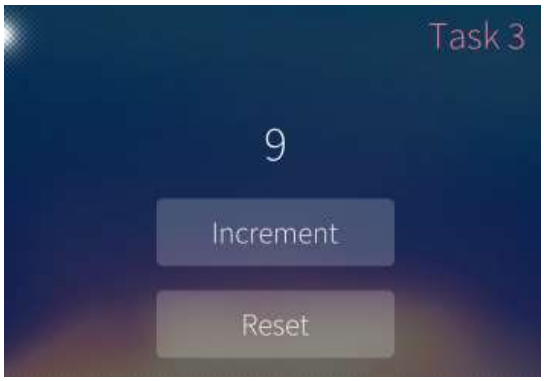

Рисунок 3 Несколько раз выполнено увеличение счётчика.

4. Подход аналогично описанному в пункте 3 за исключением того, что у элемента *Counter* помимо *id* будет устанавливаться атрибут *count* (например, *count: 3*).

Рисунок 4 Счётчик с первичной инициализацией.

5. Создание класса, содержащего список из строк, аналогично пункту 1. Заголовочный файл – *list_of_strings.h*, реализация – *list_of_strings.cpp*, регистрация в приложении – *qmlRegisterType<ListOfStrings>("custom.ListOfStrings", 1, 0, "ListOfStrings").*

6, 7 Класс со списком строк будет построен на основе встроенного класса *QStringList* – объект этого класса будет использоваться для хранения строк. В методе добавления элементов в наш список будем использовать функцию *str.toLower()* для преобразования строки *str* в нижний регистр. А в методе отображения элементов списка к первой букве применим *toUpper()* – переведём её обратно в верхний регистр.

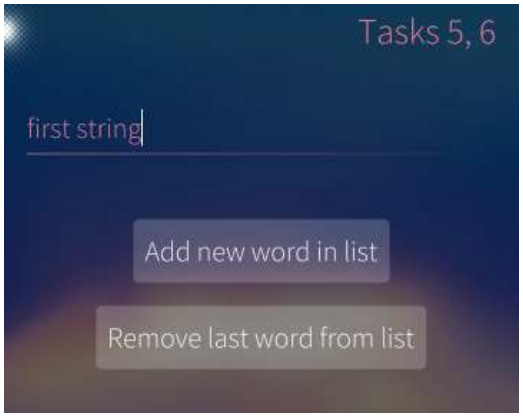

Рисунок 5 Добавление первой строки в список.

Рисунок 6 Первая строка добавлена. Добавление второй строки.

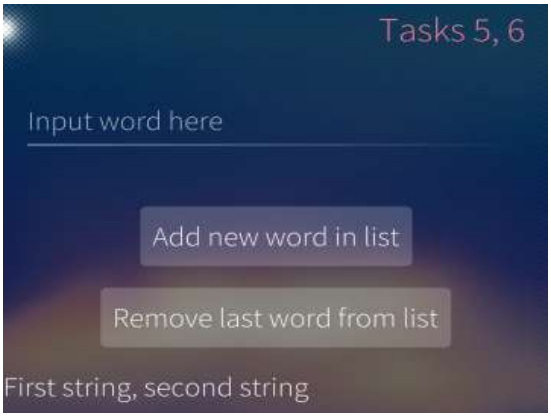

Рисунок 7 В списке две строки.

4 Приложение

4.1 Файл FirstPage.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    id: page

    SilicaFlickable {
        anchors.fill: parent
        contentHeight: column.height

        Column {
            id: column
            width: page.width
            spacing: Theme.paddingLarge
            PageHeader {
                title: qsTr("Main Page")
            }
            Button {
                text: qsTr("Counter without initialization")
                onClicked: pageStack.push(Qt.resolvedUrl("task03.qml"))
                anchors.horizontalCenter: parent.horizontalCenter
            }
            Button {
                text: qsTr("Counter with initialization")
                onClicked: pageStack.push(Qt.resolvedUrl("task04.qml"))
                anchors.horizontalCenter: parent.horizontalCenter
            }
            Button {
                text: qsTr("List of strings")
                onClicked: pageStack.push(Qt.resolvedUrl("task05-06.qml"))
                anchors.horizontalCenter: parent.horizontalCenter
            }

        } // Column
    } // SilicaFlickable
} // Page
```

4.2 Файл 08b_lab.cpp

```
#ifdef QT_QML_DEBUG
#include <QtQuick>
#include "counter.h"
#include "list_of_strings.h"
#endif

#include <sailfishapp.h>

int main(int argc, char *argv[])
{
    // main
    QGuiApplication *app = SailfishApp::application(argc, argv);
    QQuickView *view = SailfishApp::createView();
    view->setSource(SailfishApp::pathTo("qml/08b_lab.qml"));
    qmlRegisterType<Counter>("custom.Counter", 1, 0, "Counter");
    qmlRegisterType<ListOfStrings>("custom.ListOfStrings", 1, 0,
"ListOfStrings");
    view->show();
```

```
    // Demo task 2
    const QMetaObject metaCounter = Counter::staticMetaObject;
    QObject *obj = metaCounter.newInstance();
    QObject::connect(obj, SIGNAL(countChange()), obj, SLOT(print()));
    metaCounter.invokeMethod(obj,"increment");
    metaCounter.invokeMethod(obj,"increment");
    metaCounter.invokeMethod(obj,"increment");
    metaCounter.invokeMethod(obj,"reset");
    // end Demo task 2

    return app->exec();
}
```

4.3 Файл counter.h

```
#ifndef COUNTER_H
#define COUNTER_H

#include <QObject>
#include <QMetaObject>
#include <QDebug>

class Counter : public QObject
{
    Q_OBJECT
    Q_PROPERTY(int count READ getCount WRITE setCount NOTIFY countChange)
private:
    int count = 0;

public:
    Q_INVOKABLE Counter() : QObject() {}

    Q_INVOKABLE void increment();
    Q_INVOKABLE void reset();
    int getCount();
    void setCount(int _count);

signals:
    void countChange();

private slots:
    void print();
};

#endif // COUNTER_H
```

4.4 Файл counter.cpp

```
#include "counter.h"


Q_INVOKABLE void Counter::increment() {
    count++;
    emit countChange();
}

Q_INVOKABLE void Counter::reset() {
    count = 0;
    emit countChange();
}

int Counter::getCount() {
    return count;
```

```
}

void Counter::setCount(int _count) {
    count = _count;
}

// slot
void Counter::print() {
    qDebug() << "Counter: " << count;
}
```

4.5 Файл task03.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0
import custom.Counter 1.0


Page {
    id: page

    Counter {
        id: counter
    }

    Column {
        id: column
        width: page.width
        spacing: Theme.paddingLarge

        PageHeader { title: "Task 3"}

        Label {
            text: counter.count
            anchors.horizontalCenter: parent.horizontalCenter
            font.pixelSize: 64
        }

        Button {
            text: "Increment"
            anchors.horizontalCenter: parent.horizontalCenter
            onClicked: counter.increment()
        }

        Button {
            text: "Reset"
            anchors.horizontalCenter: parent.horizontalCenter
            onClicked: counter.reset()
        }
    }
}
```

4.6 Файл task04.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0
import custom.Counter 1.0


Page {
    id: page

    Counter {
```

```
        id: counter
        count: 3
    }

    Column {
        id: column
        width: page.width
        spacing: Theme.paddingLarge

        PageHeader { title: "Task 4"}

        Label {
            text: counter.count
            anchors.horizontalCenter: parent.horizontalCenter
            font.pixelSize: 64
        }

        Button {
            text: "Increment"
            anchors.horizontalCenter: parent.horizontalCenter
            onClicked: counter.increment()
        }

        Button {
            text: "Reset"
            anchors.horizontalCenter: parent.horizontalCenter
            onClicked: counter.reset()
        }
    }
}
```

4.7 Файл list_of_strings.h

```
#ifndef LIST_OF_STRINGS_H
#define LIST_OF_STRINGS_H

#include <QObject>
#include <QMetaObject>
#include <QDebug>
#include <QString>
#include <QList>


#include <QObject>
#include <QString>
#include <QStringList>


class ListOfStrings : public QObject
{
    Q_OBJECT
    Q_PROPERTY(QString list READ getText NOTIFY listChanged)

private:
    QStringList list;

public:
    ListOfStrings() : QObject() {
        list = QStringList();
    }

    Q_INVOKABLE void push(QString str);
    Q_INVOKABLE void pop();
```

```
    QString getText();

signals:
    void listChanged();
};

#endif // LIST_OF_STRINGS_H
```

4.8 Файл list_of_strings.cpp

```
#include "list_of_strings.h"


Q_INVOKABLE void ListOfStrings::push(QString str) {
    list.append(str.toLower());
    emit listChanged();
}

Q_INVOKABLE void ListOfStrings::pop() {
    list.removeLast();
    emit listChanged();
}

QString ListOfStrings::getText() {
    QString text = list.join(", ");
       return text.replace(0, 1, text[0].toUpper());
}
```

4.9 Файл task05-06.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0
import custom.ListOfStrings 1.0


Page {
    id: page

    ListOfStrings {
        id: list_strings
    }

    Column {
        id: column
        width: page.width
        spacing: Theme.paddingLarge

        PageHeader { title: "Tasks 5, 6"}

        TextField {
            id: text_field
            width: parent.width
            placeholderText: "Input string here"
        }

        Button {
            text: "Add new string in list"
            anchors.horizontalCenter: parent.horizontalCenter

            onClicked: {
                list_strings.push(text_field.text);
                text_field.text = "";
            }
```

```
        }

        Button {
            text: "Remove last string from list"
            anchors.horizontalCenter: parent.horizontalCenter

            onClicked: list_strings.pop()
        }

        Label {
            id: label_list_strings
            width: page.width
            wrapMode: "Wrap"
            text: list_strings.list
        }
    }
}
```

5 Используемая литература

1. Документация QT – https://doc.qt.io/qt-5/qmake-project-files.html