МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования   
**«Национальный исследовательский   
Нижегородский государственный университет им. Н.И. Лобачевского»**

**(ННГУ)**

**Институт информационных технологий, математики и механики**

**Отчет по практическому заданию для лекции №8**

**Выполнила:**

студентка группы 382006-2

Кулёва Анна Андреевна

**Проверил:**

Карчков Денис Александрович

Рецензент:

Нижний Новгород  
2023

# Содержание

[Содержание 2](#_Toc149486933)

[1. Цель практического занятия 3](#_Toc149486934)

[2. Постановка задачи 4](#_Toc149486935)

[3. Руководство пользователя 6](#_Toc149486936)

[4. Руководство программиста 8](#_Toc149486937)

[Заключение 12](#_Toc149486938)

[Приложение 13](#_Toc149486939)

# Цель практического занятия

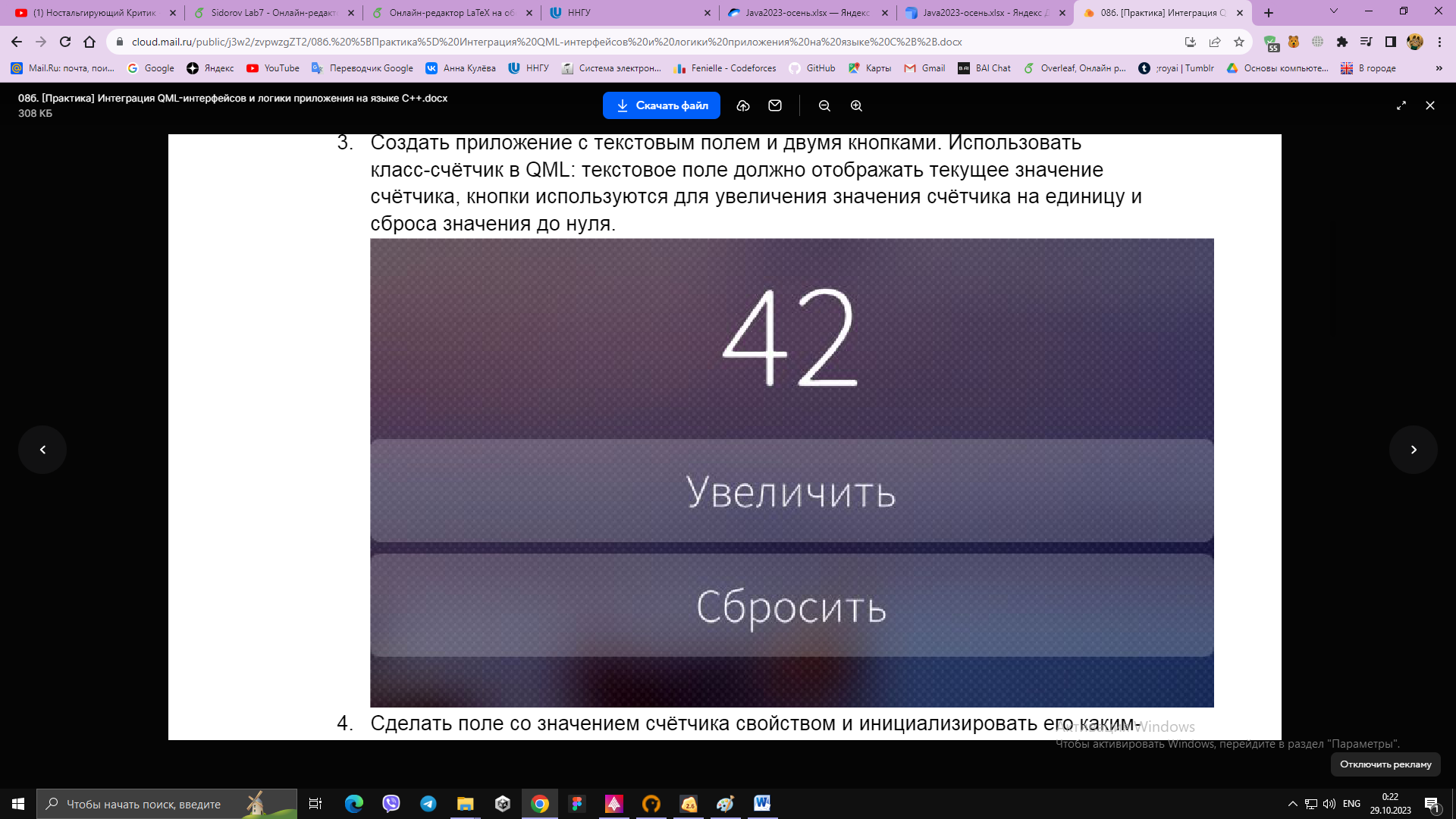
Цель данного практического занятия состоит в том, чтобы научиться использовать C++ классы в QML, научиться писать собственные QML компоненты на языке C++ и использовать их в приложении.

# 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.

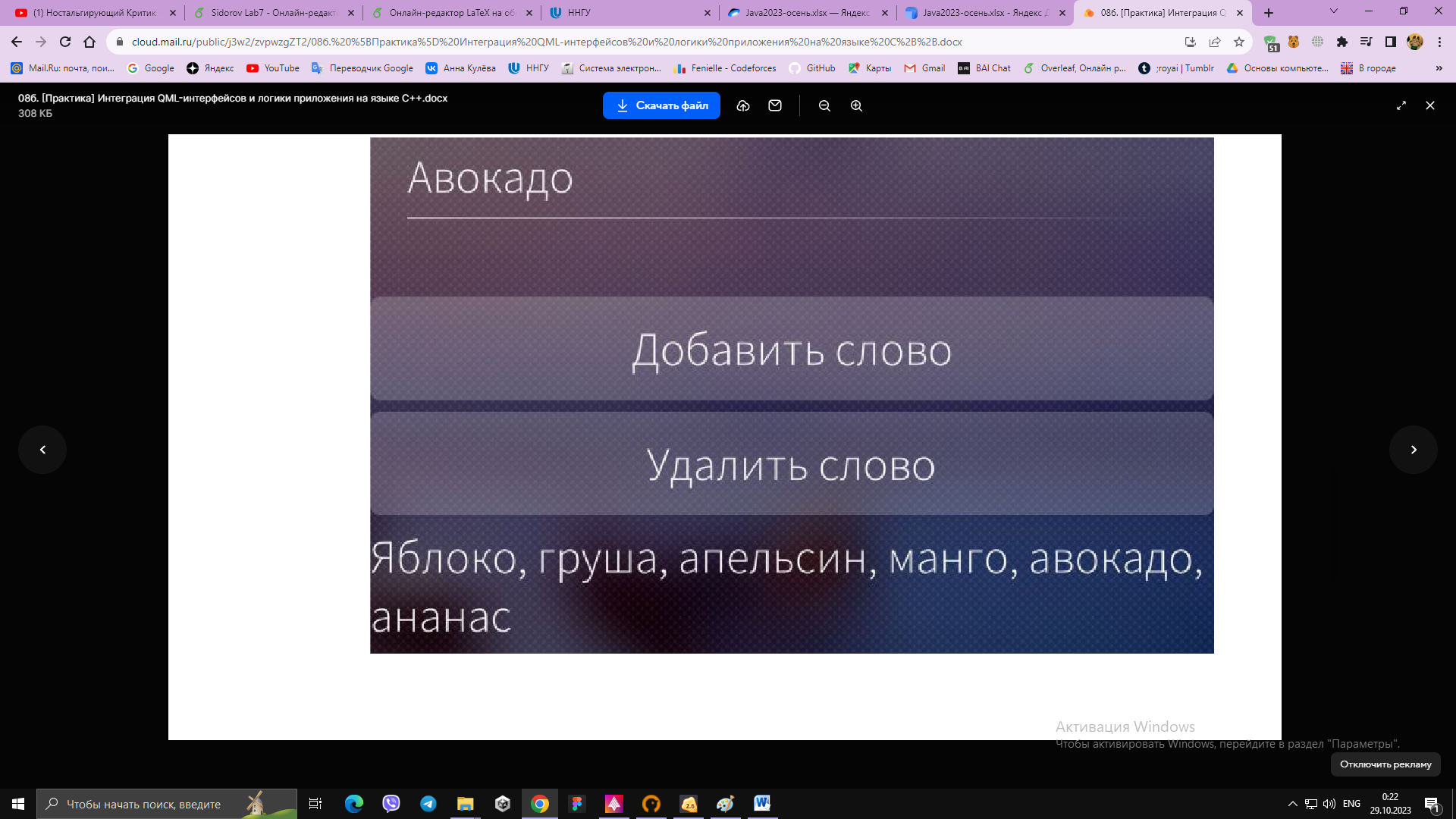


4. Сделать поле со значением счётчика свойством и инициализировать его каким-либо значением при создании объекта в QML.

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.



# Руководство пользователя

При запуске программы пользователь увидит главную страницу с кнопками, по которым сможет перейти к 1 и 2 заданиям. На странице с заданием 1 пользователь сможет взаимодействовать с двумя кнопками: «Увеличить» - увеличивает значение переменной на 1 и «Сбросить» - сбрасывает значение до 0 (рисунок 1). Значение переменной отображается в текстовом поле.

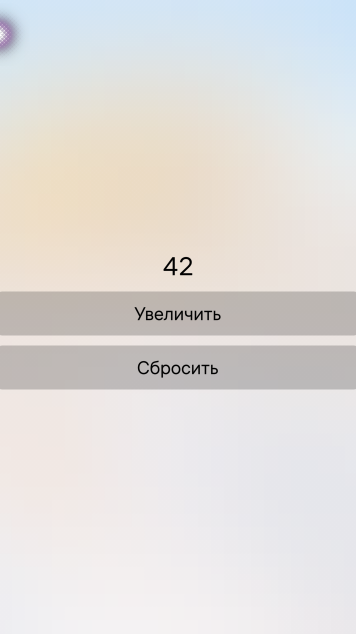


Рисунок 1. Счётчик

На странице с заданием 2 текстовое поле, куда пользователь может ввести слово. Это слово можно добавить в конец списка с помощью кнопки «Добавить слово». Также по кнопке «Удалить слово» можно удалить последнее слово списка (рисунок 2). Слова записываются через запятую, первое – с заглавной буквы.

(рисунок 2).

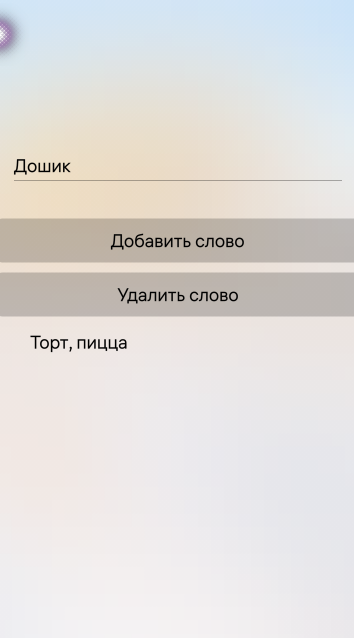


Рисунок 2. Список

# Руководство программиста

Программа реализована на языке программирования QML.

1. Реализуем класс-счетчик в файле Counter.h с полем для хранения текущего значения int m\_count и методами для увеличения значения на единицу Q\_INVOKABLE void inc() и сброса до нуля Q\_INVOKABLE void reset:

*class* Counter : *public* QObject

{

Q\_OBJECT

Q\_PROPERTY(int m\_count READ getCount WRITE setCount NOTIFY countChanged)

*public*:

Counter(){}

Q\_INVOKABLE int getCount() { *return* m\_count; };

Q\_INVOKABLE void inc() { m\_count++; emit countChanged(); };

Q\_INVOKABLE void reset() { m\_count = 0; emit countChanged(); };

Q\_INVOKABLE void setCount(*const* int temp) { m\_count = temp; emit countChanged(); };

signals:

void countChanged();

*private*:

int m\_count = 0;

};

1. Создаём мета-объект класса-счетчика Counter с id: counter:

Counter {

id: *counter*

m\_count: 42

}

1. Создаём приложение с текстовым полем Label и двумя кнопками Button, которые при нажатии вызывают counter.inc() и counter.reset() соответственно. При создании приложения используется класс-счетчик в QML:

Column {

id: *column*

width: *parent*.width

anchors.centerIn: *parent*

spacing: 20

Label {

anchors.horizontalCenter: *parent*.horizontalCenter

id: *l1*

text: *counter*.m\_count

font.pixelSize: Theme.fontSizeExtraLarge

}

Button {

anchors.horizontalCenter: *parent*.horizontalCenter

width: *parent*.width

text: "Увеличить"

onClicked: {

*counter*.inc();

}

}

Button {

anchors.horizontalCenter: *parent*.horizontalCenter

width: *parent*.width

text: "Сбросить"

onClicked: {

*counter*.reset()

}

}

1. Создаём поле со значением счётчика свойством:

Q\_PROPERTY(int m\_count READ getCount WRITE setCount NOTIFY countChanged)

1. Создаём класс, содержащий список из строк. Он содержит методы для добавления строки в список Q\_INVOKABLE void add(QString temp) и удаления последней добавленной строки Q\_INVOKABLE void popBack() :

class StringList : public QObject

{

Q\_OBJECT

public:

StringList() {}

Q\_INVOKABLE void add(QString temp) { m\_data << temp; };

Q\_INVOKABLE void popBack()

{

if (!m\_data.isEmpty()) {

m\_data.pop\_back();

}

};

private:

QList<QString> m\_data;

};

return temp;

};

private:

QList<QString> m\_data;

};

1. Создаём приложение, позволяющее добавить введённое слово и удалить последнее добавленное с использованием данного класса в QML:

Page {

StringList {

id: stringList

}

Column {

id: column

width: parent.width

anchors.horizontalCenter: parent.horizontalCenter

spacing: 20

y: 300

TextField {

id: textField;

placeholderText: "Введите слово"

}

Button {

text: "Добавить слово"

width: parent.width

onClicked: {

stringList.add(textField.text)

label.text = stringList.getAll();

}

anchors.horizontalCenter: parent.horizontalCenter

}

Button {

text: "Удалить слово"

width: parent.width

onClicked: {

stringList.popBack();

label.text = stringList.getAll();

}

anchors.horizontalCenter: parent.horizontalCenter

}

TextField {

id: label;

width: parent.width;

x: Theme.horizontalPageMargin

text: stringList.getAll()

readOnly: true;

wrapMode: TextInput.Wrap

}

}

1. Реализуем 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:

Q\_INVOKABLE QString getAll()

{

QString temp;

for (int i = 0; i < m\_data.length(); i++)

{

if (i == 0) {

QString t = m\_data[i];

t[0] = t[0].toUpper();

temp += t;

} else {

temp += m\_data[i].toLower();

}

if (i != m\_data.length()-1){

temp += ", ";

}

}

return temp;

};

# Заключение

В данной лабораторной работе я научилась использовать C++ классы в QML, научиться писать собственные QML компоненты на языке C++ и использовать их в приложении. Также были выполнены все шаги практического задания.

# Приложение

**Counter.h**

#ifndef COUNTER\_H

#define COUNTER\_H

#include <QObject>

#include <QString>

*class* **Counter** : *public* QObject

{

Q\_OBJECT

Q\_PROPERTY(int m\_count READ getCount WRITE setCount NOTIFY countChanged)

*public*:

*//* *explicit* *Counter(QObject* *\*parent* *=* *nullptr);*

**Counter**(){}

Q\_INVOKABLE int **getCount**() { *return* m\_count; };

Q\_INVOKABLE void **inc**() { m\_count++; emit countChanged(); };

Q\_INVOKABLE void **reset**() { m\_count = 0; emit countChanged(); };

Q\_INVOKABLE void **setCount**(*const* int temp) { m\_count = temp; emit countChanged(); };

signals:

void **countChanged**();

*private*:

int m\_count = 0;

};

#endif *//* *COUNTER\_H*

**StringList.h**

#ifndef STRINGLIST\_H

#define STRINGLIST\_H

#include <QObject>

#include <QString>

*class* **StringList** : *public* QObject

{

Q\_OBJECT

*public*:

*//* *explicit* *StringList(QObject* *\*parent* *=* *nullptr);*

**StringList**() {}

Q\_INVOKABLE void **add**(QString temp) { m\_data << temp; };

Q\_INVOKABLE void **popBack**()

{

*if* (!m\_data.isEmpty()) {

m\_data.pop\_back();

}

};

Q\_INVOKABLE QString **getAll**()

{

QString temp;

*for* (int i = 0; i < m\_data.length(); i++)

{

*if* (i == 0) {

QString t = m\_data[i];

t[0] = t[0].toUpper();

temp += t;

} *else* {

temp += m\_data[i].toLower();

}

*if* (i != m\_data.length()-1){

temp += ", ";

}

}

*return* temp;

};

*private*:

QList<QString> m\_data;

};

#endif *//* *STRINGLIST\_H*

**main.cpp**

#include <QScopedPointer>

#include <QGuiApplication>

#include <QQuickView>

#include "counter.h"

#include "StringList.h"

#include "Calc.h"

#include <sailfishapp.h>

int **main**(int argc, char \*argv[])

{

QScopedPointer<QGuiApplication> application(SailfishApp*::application(argc,* *argv)*);

application->setOrganizationName(QStringLiteral("ru.auroraos"));

application->setApplicationName(QStringLiteral("Lab7"));

qmlRegisterType<Counter>("harbour.appname.counter", 1, 0, "Counter");

qmlRegisterType<StringList>("harbour.appname.stringlist", 1, 0, "StringList");

qmlRegisterType<Calc>("harbour.appname.calc", 1, 0, "Calc");

QScopedPointer<QQuickView> view(SailfishApp*::createView()*);

view->setSource(SailfishApp::pathTo(QStringLiteral("qml/Lab7.qml")));

view->show();

Counter myCounter;

int cnt = myCounter.getCount();

*//printf("m\_count:* *%d",* *cnt);*

qDebug() << cnt << "\n";

myCounter.inc();

cnt = myCounter.getCount();

*//printf("m\_count:* *%d",* *cnt);*

qDebug() << cnt << "\n";

myCounter.reset();

*return* application->exec();

}

**Page1.qml**

*import* QtQuick 2.0

*import* Sailfish.Silica 1.0

*import* harbour.appname.counter 1.0

Page {

id: *page*

allowedOrientations: Orientation.All

Counter {

id: *counter*

m\_count: 42

}

Column {

id: *column*

width: *parent*.width

anchors.centerIn: *parent*

spacing: 20

Label {

anchors.horizontalCenter: *parent*.horizontalCenter

id: *l1*

text: *counter*.m\_count

font.pixelSize: Theme.fontSizeExtraLarge

}

Button {

anchors.horizontalCenter: *parent*.horizontalCenter

width: *parent*.width

text: "Увеличить"

onClicked: {

*counter*.inc();

}

}

Button {

anchors.horizontalCenter: *parent*.horizontalCenter

width: *parent*.width

text: "Сбросить"

onClicked: {

*counter*.reset()

}

}

}

}

**Page2.qml**

*import* QtQuick 2.0

*import* Sailfish.Silica 1.0

*import* harbour.appname.stringlist 1.0

Page {

StringList {

id: *stringList*

}

Column {

id: *column*

width: *parent*.width

anchors.horizontalCenter: *parent*.horizontalCenter

spacing: 20

y: 300

TextField {

id: *textField*;

placeholderText: "Введите слово"

}

Button {

text: "Добавить слово"

width: *parent*.width

onClicked: {

*stringList*.add(*textField*.text)

*label*.text = *stringList*.getAll();

}

anchors.horizontalCenter: *parent*.horizontalCenter

}

Button {

text: "Удалить слово"

width: *parent*.width

onClicked: {

*stringList*.popBack();

*label*.text = *stringList*.getAll();

}

anchors.horizontalCenter: *parent*.horizontalCenter

}

TextField {

id: *label*;

width: *parent*.width;

x: Theme.horizontalPageMargin

text: *stringList*.getAll()

readOnly: true;

wrapMode: TextInput.Wrap

}

}