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

#### Кафедра МОСТ

##### Направление подготовки: «Программная инженерия»

##### Профиль подготовки: «Разработка программно-информационных систем»

## ОТЧЕТ

### По лабораторной работе №8

##### по дисциплине «Технология программирования мобильных систем»

**Выполнил:** студент группы 381908-4

Кургузиков Кирилл

#### Руководитель:

##### доцент кафедры программной инженерии

##### Борисов Николай Анатольевич

##### Нижний Новгород 2022

**Содержание**

[1. Цель 3](#_Toc97973273)

[2. Постановка задачи 4](#_Toc97973274)

[3. Решение поставленной задачи 6](#_Toc97973275)

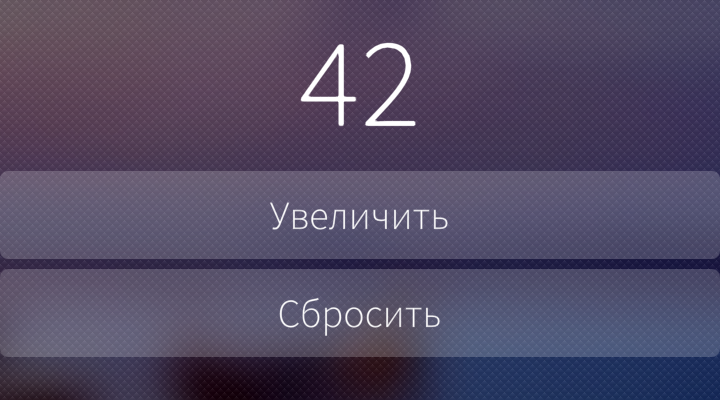
[4. Руководство пользователя 8](#_Toc97973276)

[Приложение 13](#_Toc97973277)

# Цель

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.

# Решение поставленной задачи

1. Класс-счётчик

class **Counting** : public QObject

{

Q\_OBJECT

Q\_PROPERTY(int count READ count WRITE setCount NOTIFY countChanged)

public:

explicit **Counting**(QObject \*parent = nullptr);

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

Q\_INVOKABLE void **increaseCount**() { m\_count++; };

Q\_INVOKABLE void **zeroingCount**() { m\_count = 0; };

int **count**() {return m\_count; };

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

signals:

void **countChanged**();

private:

int m\_count = 0;

};

2. Использование мета-объект класса-счётчика

Counting {

id: *cntg*

}

3. Приложение с текстовым полем и двумя кнопками. Использование класс-счётчика в QML

Column {

Label {

id: *label*;

text: *cntg*.getCount();

}

Button {

text: "Увеличить"

onClicked: {

*cntg*.increaseCount();

*label*.text = *cntg*.getCount();

}

}

Button {

text: "Обнулить"

onClicked: {

*cntg*.zeroingCount();

*label*.text = *cntg*.getCount();

}

}

}

4. Поле со значением счётчика как свойство

Q\_PROPERTY(int count READ count WRITE setCount NOTIFY countChanged)

5. Класс, содержащий список из строк

class **ListStrings** : public QObject

{

Q\_OBJECT

public:

explicit **ListStrings**(QObject \*parent = nullptr);

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

Q\_INVOKABLE void **removeBackElem**()

{

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

m\_data.pop\_back();

}

};

Q\_INVOKABLE QString **getAllList**()

{

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;

};

6. Приложение, использующее класс-список

ListStrings {

id: *listst*

}

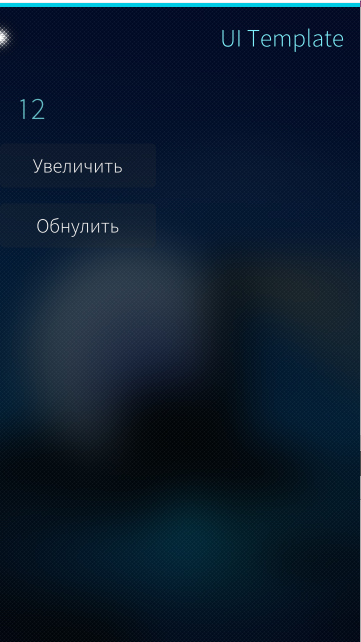
7. Свойство только для чтения, которое позволяет получить список всех строк в виде одной

private:

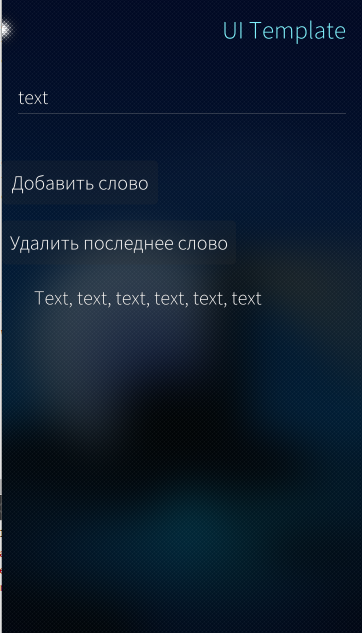
QList<QString> m\_data;

# Руководство пользователя

1. Использование класса-счетчика



2. Использование класса-списка



# Приложение

liststrings.h

#ifndef LISTSTRINGS\_H

#define LISTSTRINGS\_H

#include <QObject>

#include <QString>

class **ListStrings** : public QObject

{

Q\_OBJECT

public:

explicit **ListStrings**(QObject \*parent = nullptr);

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

Q\_INVOKABLE void **removeBackElem**()

{

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

m\_data.pop\_back();

}

};

Q\_INVOKABLE QString **getAllList**()

{

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 // LISTSTRINGS\_H

task1.h

#ifndef TASK1\_H

#define TASK1\_H

#include <QObject>

#include <QString>

class **Counting** : public QObject

{

Q\_OBJECT

Q\_PROPERTY(int count READ count WRITE setCount NOTIFY countChanged)

public:

explicit **Counting**(QObject \*parent = nullptr);

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

Q\_INVOKABLE void **increaseCount**() { m\_count++; };

Q\_INVOKABLE void **zeroingCount**() { m\_count = 0; };

int **count**() {return m\_count; };

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

signals:

void **countChanged**();

private:

int m\_count = 0;

};

#endif // TASK1\_H

lab\_8.cpp

#include <sailfishapp.h>

#include <QtQuick>

#include <QScopedPointer>

#include "task1.h"

#include "liststrings.h"

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

{

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

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

qmlRegisterType<Counting>("com.counting", 1, 0, "Counting");

qmlRegisterType<ListStrings>("com.liststrings", 1, 0, "ListStrings");

v->setSource(SailfishApp::pathTo("qml/lab\_8.qml"));

v->show();

return app->exec();

}

liststrings.cpp

#include "liststrings.h"

ListStrings::**ListStrings**(QObject \*parent) : QObject(*parent*)

{

}

task1.cpp

#include "task1.h"

Counting::**Counting**(QObject \*parent) : QObject(*parent*)

{

}

FirstPage.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

import com.counting 1.0

Page { // @disable-check M301

id: *page*

allowedOrientations: Orientation.All

SilicaFlickable {

anchors.fill: *parent*

PullDownMenu { // @disable-check M301

MenuItem { // @disable-check M301

text: *qsTr*("Show Page 2")

onClicked: *pageStack*.animatorPush(*Qt*.resolvedUrl("SecondPage.qml"))

}

}

Counting {

id: *cntg*

count: 10;

}

Column {

id: *column*

width: *page*.width

spacing: Theme.paddingLarge

PageHeader {

title: *qsTr*("UI Template")

}

Label { // @disable-check M301

id: *label*;

x: Theme.horizontalPageMargin

text: *cntg*.getCount();

color: Theme.secondaryHighlightColor

font.pixelSize: Theme.fontSizeExtraLarge

}

Button { // @disable-check M301

text: "Увеличить"

onClicked: {

*cntg*.increaseCount();

*label*.text = *cntg*.getCount();

}

}

Button {

text: "Обнулить"

onClicked: {

*cntg*.zeroingCount();

*label*.text = *cntg*.getCount();

}

}

}

}

}

SecondPage.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

import com.liststrings 1.0

Page { // @disable-check M301

id: *page*

allowedOrientations: Orientation.All

SilicaFlickable {

anchors.fill: *parent*

PullDownMenu { // @disable-check M301

MenuItem { // @disable-check M301

text: *qsTr*("Show Page 1")

onClicked: pageStack.animatorPush(*Qt*.resolvedUrl("FirstPage.qml"))

}

}

ListStrings {

id: *listst*

}

Column {

id: *column*

width: *page*.width

spacing: Theme.paddingLarge

PageHeader {

title: *qsTr*("UI Template")

}

TextField { // @disable-check M301

id: *tf*;

placeholderText: "Введите слово"

}

Button { // @disable-check M301

text: "Добавить слово"

onClicked: {

*listst*.addInList(*tf*.text)

*label*.text = *listst*.getAllList();

}

}

Button { // @disable-check M301

text: "Удалить последнее слово"

onClicked: {

*listst*.removeBackElem();

*label*.text = *listst*.getAllList();

}

}

TextField { // @disable-check M301

id: *label*;

width: *parent*.width;

x: Theme.horizontalPageMargin

text: *listst*.getAllList()

readOnly: true;

}

}

}

}

Lab\_8.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

import com.counting 1.0

import com.liststrings 1.0

import "pages"

ApplicationWindow {

initialPage: Component { FirstPage { } } // @disable-check M301

cover: *Qt*.resolvedUrl("cover/CoverPage.qml")

allowedOrientations: defaultAllowedOrientations

}