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 Математического обеспечения и суперкомпьютерных технологий**

Направление подготовки: «Фундаментальная информатика и информационные технологии»

Профиль подготовки: «Инженерия программного обеспечения»

**ОТЧЕТ**

**По лабораторной работе №7**

по дисциплине «Технология программирования мобильных систем»

**Выполнила:** студентка группы 381906-3

Сорокина Екатерина Вячеславовна

**Руководитель:**

Доцент, кандидат технических наук

Борисов Николай Анатольевич

Нижний Новгород  
2022

**Содержание**

[1. Цель 3](#_Toc123147554)

[2. Постановка задачи 4](#_Toc123147555)

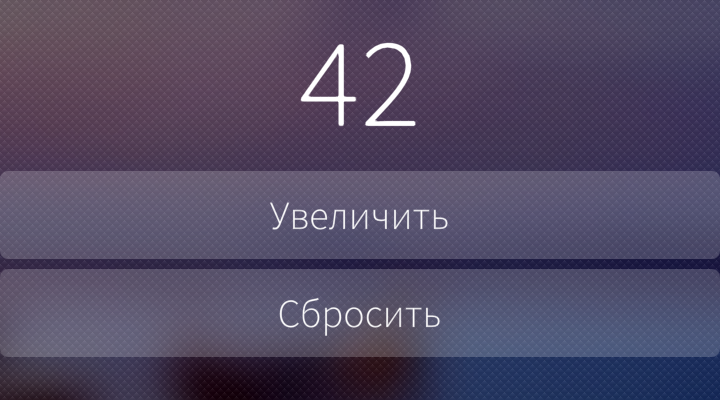
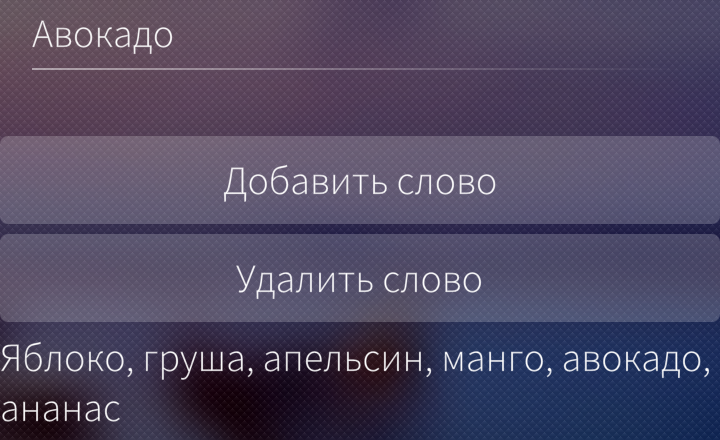
[3. Решение поставленной задачи 5](#_Toc123147556)

[4. Руководство пользователя 9](#_Toc123147557)

[Приложение 11](#_Toc123147558)

1. Цель

Цель: научиться использовать C++ классы в QML, научиться писать собственные QML компоненты на языке C++ и использовать их в приложении.

1. Постановка задачи
2. Создать класс-счётчик с полем для хранения текущего значения и методами для увеличения значения на единицу и сброса до нуля.
3. Использовать мета-объект класса-счётчика для создания объекта и вызова его методов (использовать функцию main, результат изменения состояния проверять выводом на консоль).
4. 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.
5. Сделать поле со значением счётчика свойством и инициализировать его каким-либо значением при создании объекта в QML.
6. Создать класс, содержащий список из строк. Класс должен содержать методы для добавления строки в список и удаления последней добавленной строки.
7. Создать приложение, позволяющее добавить введённое слово и удалить последнее добавленное с использованием данного класса в QML. Слова сохраняются в нижнем регистре.
8. 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.  
     
   
9. Решение поставленной задачи

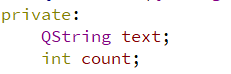
Создаём проект со стандартной заготовкой приложения.

**Задание 1.**

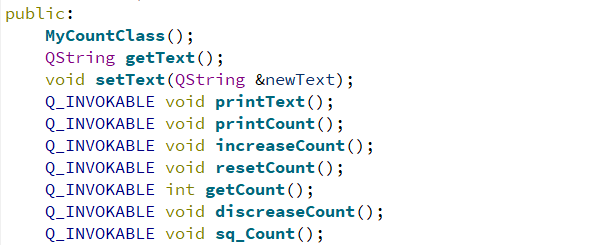
Макросы Q\_Object и Q\_PROPERTY для доступа к свойствам с помощью системы мета-объектов



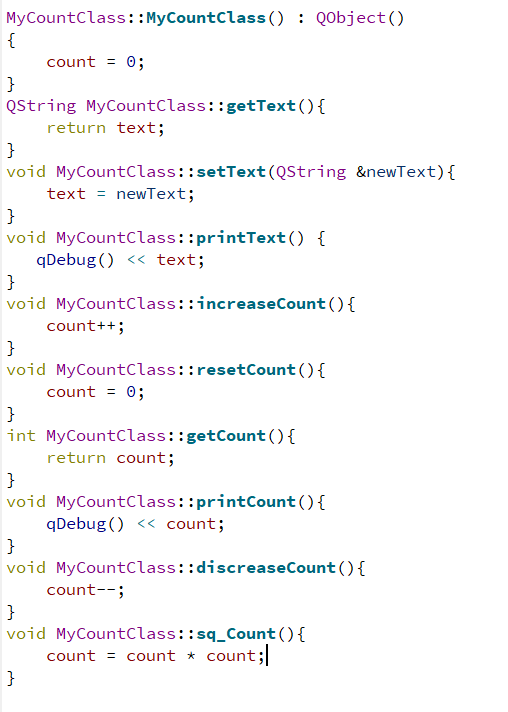
Переменные



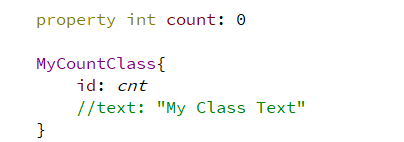
Методы доступные с помощью макроса Q\_INVOKABLE



Описание методов для инициализации счетчика, его увеличения, сброса и получения значения



Вызов класса



Вызов методов для изменения счетчика с помощью кнопок

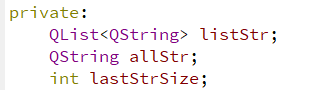


**Задание 2.**

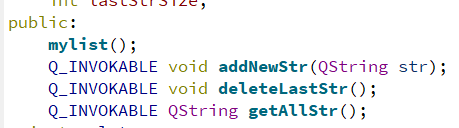
Макросы Q\_Object и Q\_PROPERTY для доступа к свойствам с помощью системы мета-объектов



Переменные



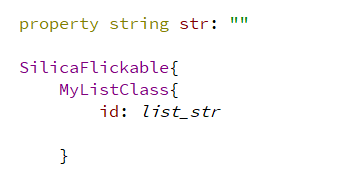
Методы доступные с помощью макроса Q\_INVOKABLE



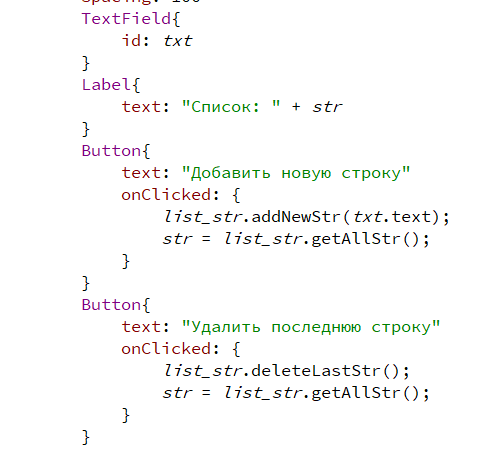
Описание методов для добавления новой записи и удаления последней



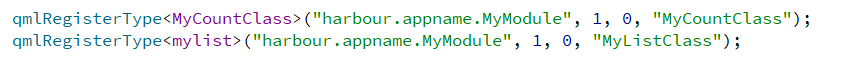
Вызов класса



Вызов методов класса с помощью кнопок



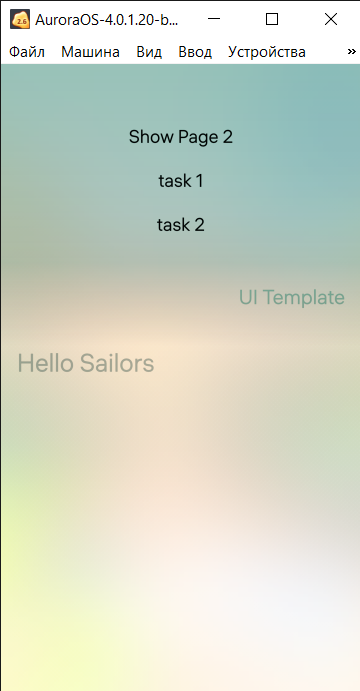
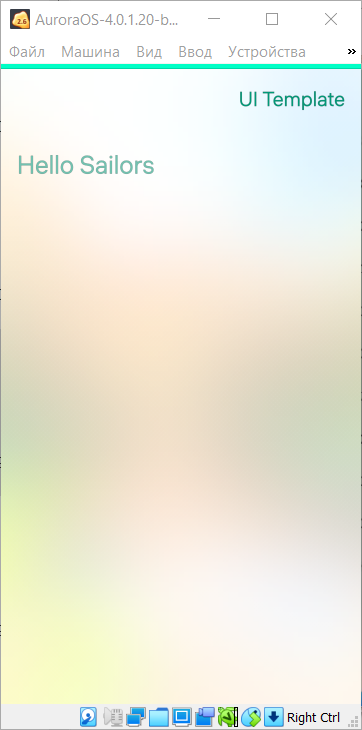
Подключение наших классов в главном файле проекта

Осталось только сохранить все изменения и запустить эмулятор.

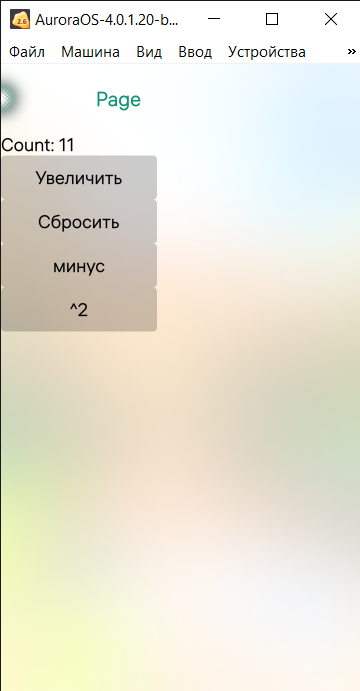
1. Руководство пользователя

При запуске приложения пользователь попадает на страницу FirstPage. Потянув вниз, открывается меню с нашими страницами.

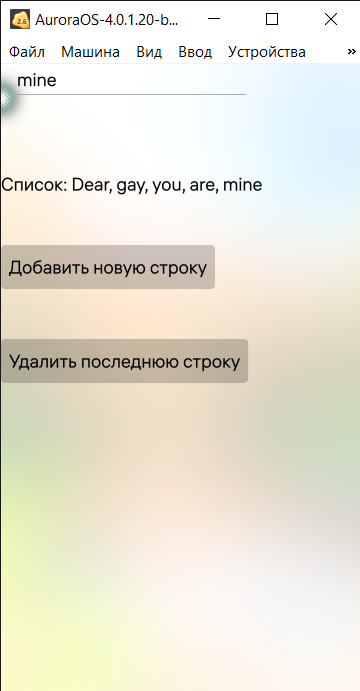


При нажатии на каждую из страниц, она загрузит нам наше задание.

Задание 1:



Задание 2:



Приложение

// FirstPage.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page*

// The effective value will be restricted by ApplicationWindow.allowedOrientations

allowedOrientations: Orientation.All

// To enable PullDownMenu, place our content in a SilicaFlickable

SilicaFlickable {

anchors.fill: *parent*

// PullDownMenu and PushUpMenu must be declared in SilicaFlickable, SilicaListView or SilicaGridView

PullDownMenu {

MenuItem {

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

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

}

MenuItem{

text: *qsTr*("task 1")

onClicked: *pageStack*.push(*Qt*.resolvedUrl("task\_1.qml"))

}

MenuItem{

text: *qsTr*("task 2")

onClicked: *pageStack*.push(*Qt*.resolvedUrl("task\_2.qml"))

}

}

// Tell SilicaFlickable the height of its content.

contentHeight: *column*.height

// Place our content in a Column. The PageHeader is always placed at the top

// of the page, followed by our content.

Column {

id: *column*

width: *page*.width

spacing: Theme.paddingLarge

PageHeader {

title: *qsTr*("UI Template")

}

Label {

x: Theme.horizontalPageMargin

text: *qsTr*("Hello Sailors")

color: Theme.secondaryHighlightColor

font.pixelSize: Theme.fontSizeExtraLarge

}

}

}

}

// task\_1.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

import harbour.appname.MyModule 1.0

Page {

id: *page1*

property int count: 0

MyCountClass{

id: *cnt*

//text: "My Class Text"

}

SilicaFlickable {

Column {

//anchors.fill: parent

PageHeader { title: "Page"}

Label { text: "Count: " + *count* }

Button {

//width: parent.width

text: "Увеличить"

onClicked: {

*cnt*.increaseCount();

*cnt*.printCount();

*count* = *cnt*.getCount();

}

}

Button {

//width: parent.width

text: "Сбросить"

onClicked: {

*cnt*.resetCount();

*cnt*.printCount();

*count* = *cnt*.getCount();

}

}

Button{

text: "минус"

onClicked: {

*cnt*.discreaseCount();

*cnt*.printCount();

*count* = *cnt*.getCount();

}

}

Button{

text: "^2"

onClicked: {

*cnt*.sq\_Count();

*cnt*.printCount();

*count* = *cnt*.getCount();

}

}

}

}

}

//mycount.h

#ifndef MYCOUNT\_H

#define MYCOUNT\_H

#include <QObject>

class **MyCountClass** : public QObject

{

Q\_OBJECT

Q\_PROPERTY(QString getText READ getText WRITE setText NOTIFY TextChanged)

private:

QString text;

int count;

public:

**MyCountClass**();

QString **getText**();

void **setText**(QString &newText);

Q\_INVOKABLE void **printText**();

Q\_INVOKABLE void **printCount**();

Q\_INVOKABLE void **increaseCount**();

Q\_INVOKABLE void **resetCount**();

Q\_INVOKABLE int **getCount**();

Q\_INVOKABLE void **discreaseCount**();

Q\_INVOKABLE void **sq\_Count**();

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

private slots:

signals:

void **TextChanged**();

};

#endif // MYCOUNT\_H

//mycount.cpp

#include "mycount.h"

#include <QDebug>

MyCountClass::**MyCountClass**() : QObject()

{

count = 0;

}

QString MyCountClass::**getText**(){

return text;

}

void MyCountClass::**setText**(QString &newText){

text = newText;

}

void MyCountClass::**printText**() {

qDebug() << text;

}

void MyCountClass::**increaseCount**(){

count++;

}

void MyCountClass::**resetCount**(){

count = 0;

}

int MyCountClass::**getCount**(){

return count;

}

void MyCountClass::**printCount**(){

qDebug() << count;

}

void MyCountClass::**discreaseCount**(){

count--;

}

void MyCountClass::**sq\_Count**(){

count = count \* count;

}

//task\_2.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

import harbour.appname.MyModule 1.0

Page {

id:*page2*

property string str: ""

SilicaFlickable{

MyListClass{

id: *list\_str*

}

Column{

spacing: 100

TextField{

id: *txt*

}

Label{

text: "Список: " + *str*

}

Button{

text: "Добавить новую строку"

onClicked: {

*list\_str*.addNewStr(*txt*.text);

*str* = *list\_str*.getAllStr();

}

}

Button{

text: "Удалить последнюю строку"

onClicked: {

*list\_str*.deleteLastStr();

*str* = *list\_str*.getAllStr();

}

}

// Button{

// text: "размер списка"

// onClicked: {

// console.log()

// }

// }

}

}

}

//mylist.h

#ifndef MYLIST\_H

#define MYLIST\_H

#include <QObject>

class **mylist** : public QObject{

Q\_OBJECT

Q\_PROPERTY(QList<QString> listStr STORED true USER true)

private:

QList<QString> listStr;

QString allStr;

int lastStrSize;

public:

**mylist**();

Q\_INVOKABLE void **addNewStr**(QString str);

Q\_INVOKABLE void **deleteLastStr**();

Q\_INVOKABLE QString **getAllStr**();

private slots:

signals:

// void listChanged();

};

#endif // MYLIST\_H

//mylist.cpp

#include "mylist.h"

#include <QDebug>

mylist::**mylist**()

{

lastStrSize = 0;

}

void mylist::**addNewStr**(QString str){

if(listStr.empty()){

str.replace(0,1,str[0].toUpper());

}

else{

str.push\_front(", ");

}

allStr += str;

listStr.append(str);

qDebug()<<"element of list = "<<listStr.at(listStr.size()-1);

lastStrSize = str.size();

qDebug()<<"size of last str= "<<lastStrSize;

}

QString mylist::**getAllStr**(){

return allStr;

}

void mylist::**deleteLastStr**(){

if(!listStr.empty()){

listStr.pop\_back();

}

// lastStrSize = listStr.at(listStr.size()-1).size();

// qDebug()<<"size of last str= "<<lastStrSize;

// allStr.remove(allStr.size()-lastStrSize,lastStrSize);

allStr.clear();

for (int i =0 ; i<listStr.size();i++){

allStr += listStr[i];

}

}

//classes\_components.qml

#include <sailfishapp.h>

#include "mycount.h"

#include "mylist.h"

#include <QtQuick>

#include <QScopedPointer>

#include <QGuiApplication>

#include <QQuickView>

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

{

// SailfishApp::main() will display "qml/classes\_components.qml", if you need more

// control over initialization, you can use:

//

// - SailfishApp::application(int, char \*[]) to get the QGuiApplication \*

// - SailfishApp::createView() to get a new QQuickView \* instance

// - SailfishApp::pathTo(QString) to get a QUrl to a resource file

// - SailfishApp::pathToMainQml() to get a QUrl to the main QML file

//

// To display the view, call "show()" (will show fullscreen on device).

qmlRegisterType<MyCountClass>("harbour.appname.MyModule", 1, 0, "MyCountClass");

qmlRegisterType<mylist>("harbour.appname.MyModule", 1, 0, "MyListClass");

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

app->setApplicationName("classes\_components");

app->setOrganizationName("ru.test");

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

view->setSource(SailfishApp::pathToMainQml());

view->showFullScreen();

return app->exec();

}