ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»  
(РУТ (МИИТ))

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

ОТЧЁТ  
О ПРАКТИЧЕСКОЙ РАБОТЕ № 3

По дисциплине «Языки программирования»

Выполнил: ст. гр. ТКИ – 241

Хугаева М. М.

Проверил: к.т.н., доц.

Васильева М. А.

Москва 2023

Код:

#include "Node.h"

#pragma once

#include <memory>

template <typename Type>

struct Node

{

public:

using pointer = std::unique\_ptr<Node>;

Type data;

pointer next;

Node(Type value) : data{ std::move(value) }, next{ nullptr } {};

};

#include "SingleList.h"

#include "Node.h"

#pragma once

#include "Node.h"

#include <iostream>

#include <string>

#include <memory>

template <typename Type>

class SingleList

{

using linkPointer = typename Node<Type>::pointer;

private:

linkPointer head = {};

Node<Type>\* tail = nullptr;

public:

/\*

\* @brief Пустой объект односвязного списка

\*/

SingleList();

/\*

\* @brief Деструктор

\*/

~SingleList();

/\*

\* @brief Преобразование в строку

\*/

std::string toString() const noexcept;

/\*

\* @brief Вставка элемента

\*/

void pushFront(const Type value);

/\*

\* @brief Показать весь список

\*/

void show() const;

/\*

\* @brief Проверка наличия элементов в списке

\* @return true - есть элементы, иначе false

\*/

bool hasElements() const noexcept;

/\*

\* @brief Проверка отсутствия элементов в списке

\* @return true - нет элементов, иначе false

\*/

bool isEmpty() const noexcept;

linkPointer getHead();

};

template <typename Type>

SingleList<Type>::SingleList()

: head{ nullptr }

{

}

template <typename Type>

SingleList<Type>::~SingleList()

{

while (head != nullptr)

{

head = std::move(head->next);

}

}

template<typename Type>

std::string SingleList<Type>::toString() const noexcept

{

std::stringstream buffer;

auto current = head.get();

while (head->next != nullptr)

{

buffer << current->data;

current = current->next.get();

}

return buffer.str();

}

template<typename Type>

inline void SingleList<Type>::pushFront(Type value)

{

auto newLink = new Node<Type>(std::move(value));

if (tail != nullptr)

{

tail->next.reset(newLink);

}

else

{

head.reset(newLink);

}

tail = newLink;

}

template<typename Type>

bool SingleList<Type>::hasElements() const noexcept

{

return (this->isEmpty);

}

template<typename Type>

bool SingleList<Type>::isEmpty() const noexcept

{

return this->head == nullptr;

}

template<typename Type>

typename Node<Type>::pointer SingleList<Type>::getHead()

{

return head;

}

template<typename Type>

void SingleList<Type>::show() const

{

auto current = head.get();

while (current != nullptr)

{

std::cout << current->data << std::endl;

current = current->next.get();

}

}

Тесты:

#include "pch.h"

#include "CppUnitTest.h"

#include "/Users/yanmi/source/repos/SmartPtr/SmartPtr/SmartPtr/SingleList.h"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace TestSmartPtr

{

TEST\_CLASS(TestSmartPtr)

{

public:

TEST\_METHOD(ValidDataLIsts)

{

SingleList<int> listone;

listone.pushFront(0);

SingleList<int> listtwo;

listtwo.pushFront(0);

Assert::IsTrue(listone.toString() == listtwo.toString());

}

TEST\_METHOD(ValidDataLIst)

{

SingleList<int> listone;

listone.pushFront(0);

Assert::IsFalse(listone.toString() == "1");

}

};

}

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 1 – Выполнение тестов

UML-диаграмма:

Изображение выглядит как текст, диаграмма, снимок экрана, линия

Автоматически созданное описание



Рисунок 2 – UML-диаграмма

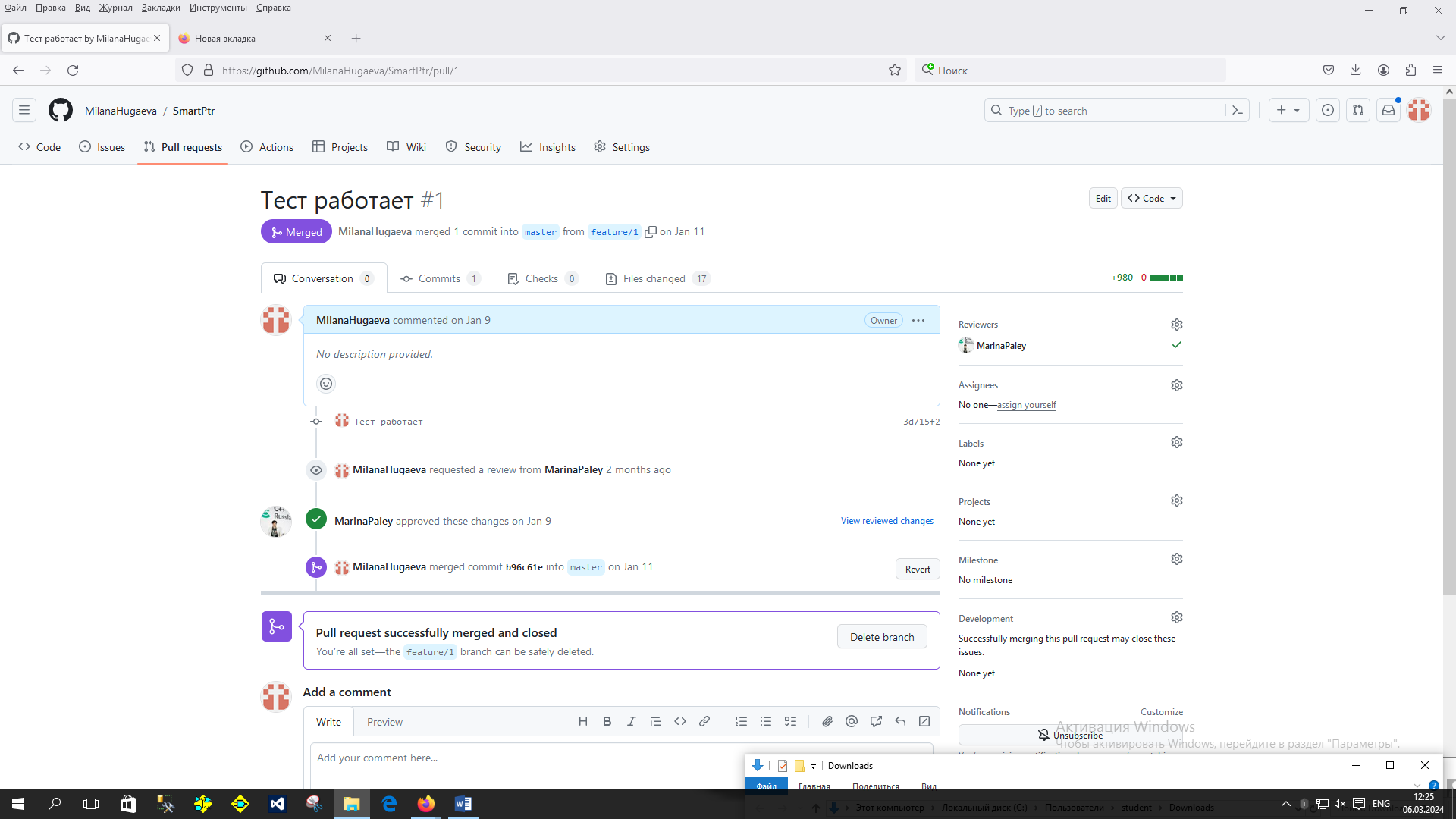


Рисунок 3 – Аппрув задания