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 № 2

По дисциплине «Языки программирования»

Выполнил: ст. гр. ТКИ – 241

Борцов Ю. Д.

Проверил: к.т.н., доц.

Васильева М. А.

Москва 2023

**Задание:** Написать архитектуру Школы, в которой содержатся классы учебный класс, ученик, преподаватель, с импользованием Smart-pointers

**Код программы:**

**#include "classroom.h"**

**#include "student.h"**

**#include "teacher.h"**

**void Classroom::addStudent(const std::shared\_ptr<Student>& student)**

**{**

**students.push\_back(student);**

**student->currentClassroom(shared\_from\_this());**

**}**

**void Classroom::addTeacher(const std::shared\_ptr<Teacher>& teacher)**

**{**

**this->teacher = teacher;**

**}**

**std::string Classroom::displayClassInfo()**

**{**

**std::ostringstream streamBuffer;**

**streamBuffer << "Class Info:\n";**

**if (teacher != nullptr)**

**{**

**streamBuffer << "\n";**

**streamBuffer << teacher->displayInfo();**

**}**

**streamBuffer << "Students:";**

**streamBuffer << "\n";**

**for (std::vector<std::weak\_ptr<Student>>::iterator temp = students.begin(); temp != students.end(); ++temp)**

**{**

**if (std::shared\_ptr<Student> sharedStudent = temp->lock())**

**{**

**streamBuffer << sharedStudent->displayInfo();**

**streamBuffer << "\n";**

**}**

**}**

**return streamBuffer.str();**

**}**

**#include <memory>**

**#include <vector>**

**#include <iostream>**

**#include <sstream>**

**class Student;**

**class Teacher;**

**class Classroom : public std::enable\_shared\_from\_this<Classroom>**

**{**

**private:**

**std::vector<std::weak\_ptr<Student>> students;**

**std::shared\_ptr<Teacher> teacher;**

**public:**

**/\*\***

**\* @brief Добавляет студента в класс.**

**\* @param student Указатель shared\_ptr на добавляемого студента.**

**\*/**

**void addStudent(const std::shared\_ptr<Student>& student);**

**/\*\***

**\* @brief Добавляет учителя в аудиторию.**

**\* @param teacher Указатель shared\_ptr на добавляемого учителя.**

**\*/**

**void addTeacher(const std::shared\_ptr<Teacher>& teacher);**

**/\*\***

**\* @brief Отображает информацию об аудитории, включая учителя и студентов.**

**\* @return Информация об учебном классе в строковом типе.**

**\*/**

**std::string displayClassInfo();**

**};**

**#include "student.h"**

**#include "classroom.h"**

**Student::Student(const std::string& name, const int age) : name(name), age(age) {}**

**void Student::currentClassroom(const std::shared\_ptr<Classroom>& classroom)**

**{**

**myClassroom = classroom;**

**}**

**std::string Student::displayInfo()**

**{**

**std::ostringstream streamBuffer;**

**streamBuffer << "Name: " << name << ", Age: " << age << "\n";**

**return streamBuffer.str();**

**}**

**#include <memory>**

**#include <iostream>**

**#include <sstream>**

**class Classroom;**

**class Student : public std::enable\_shared\_from\_this<Student>**

**{**

**private:**

**std::string name;**

**const int age;**

**std::shared\_ptr<Classroom> myClassroom;**

**public:**

**Student(const std::string& name, int age);**

**/\*\***

**\* @brief Определяет студенту аудиторию.**

**\* @param classroom Указатель shared\_ptr на аудиторию, к которой присоединяется студент.**

**\*/**

**void currentClassroom(const std::shared\_ptr<Classroom>& classroom);**

**/\*\***

**\* @brief Отображает информацию о студенте.**

**\*/**

**std::string displayInfo();**

**/\*\***

**\* @brief Получает указатель на текущую аудиторию студента.**

**\* @return Указатель shared\_ptr на текущую аудиторию студента.**

**\*/**

**std::shared\_ptr<Classroom> getClassroom()**

**{**

**return myClassroom;**

**}**

**};**

**#include "teacher.h"**

**#include "classroom.h"**

**Teacher::Teacher(const std::string& name) : name(name) {}**

**void Teacher::currentClassroom(const std::shared\_ptr<Classroom>& classroom)**

**{**

**myClassroom = classroom;**

**}**

**std::string Teacher::displayInfo()**

**{**

**std::ostringstream streamBuffer;**

**streamBuffer << "Teacher: " << name << "\n";**

**return streamBuffer.str();**

**}**

**#include <memory>**

**#include <iostream>**

**#include <sstream>**

**class Classroom;**

**class Teacher : public std::enable\_shared\_from\_this<Teacher>**

**{**

**private:**

**std::string name;**

**std::weak\_ptr<Classroom> myClassroom;**

**public:**

**/\*\***

**\* @brief Конструктор класса учитель.**

**\*/**

**Teacher(const std::string& name);**

**/\*\***

**\* @brief Присоединяет учителя к аудитории.**

**\* @param classroom Указатель shared\_ptr на аудиторию, к которой присоединяется учитель.**

**\*/**

**void currentClassroom(const std::shared\_ptr<Classroom>& classroom);**

**/\*\***

**\* @brief Отображает информацию о учителе.**

**\*/**

**std::string displayInfo();**

**/\*\***

**\* @brief Получает указатель на текущую аудиторию учителя.**

**\*/**

**std::shared\_ptr<Classroom> getClassroom()**

**{**

**return myClassroom.lock();**

**}**

**};**

**Тесты:**

#include "gtest/gtest.h"

#include "classroom.h"

#include "student.h"

#include "teacher.h"

TEST(ClassroomTest, AddStudent)

{

auto classroom = std::make\_shared<Classroom>();

auto student = std::make\_shared<Student>("John", 20);

classroom->addStudent(student);

EXPECT\_EQ(classroom->displayClassInfo(), "Class Info:\nStudents:\nName: John, Age: 20\n\n");

}

TEST(ClassroomTest, AddTeacher)

{

auto classroom = std::make\_shared<Classroom>();

auto teacher = std::make\_shared<Teacher>("Mr. Smith");

classroom->addTeacher(teacher);

EXPECT\_EQ(classroom->displayClassInfo(), "Class Info:\n\nTeacher: Mr. Smith\nStudents:\n");

}

TEST(StudentTest, DisplayInfo)

{

auto student = std::make\_shared<Student>("Alice", 22);

EXPECT\_EQ(student->displayInfo(), "Name: Alice, Age: 22\n");

}

TEST(TeacherTest, DisplayInfo)

{

auto teacher = std::make\_shared<Teacher>("Mrs. Johnson");

EXPECT\_EQ(teacher->displayInfo(), "Teacher: Mrs. Johnson\n");

}

TEST(StudentTest, CurrentClassroom)

{

auto classroom = std::make\_shared<Classroom>();

auto student = std::make\_shared<Student>("Bob", 25);

student->currentClassroom(classroom);

EXPECT\_EQ(student->getClassroom(), classroom);

}

TEST(TeacherTest, CurrentClassroom)

{

auto classroom = std::make\_shared<Classroom>();

auto teacher = std::make\_shared<Teacher>("Ms. Davis");

teacher->currentClassroom(classroom);

EXPECT\_EQ(teacher->getClassroom(), classroom);

}

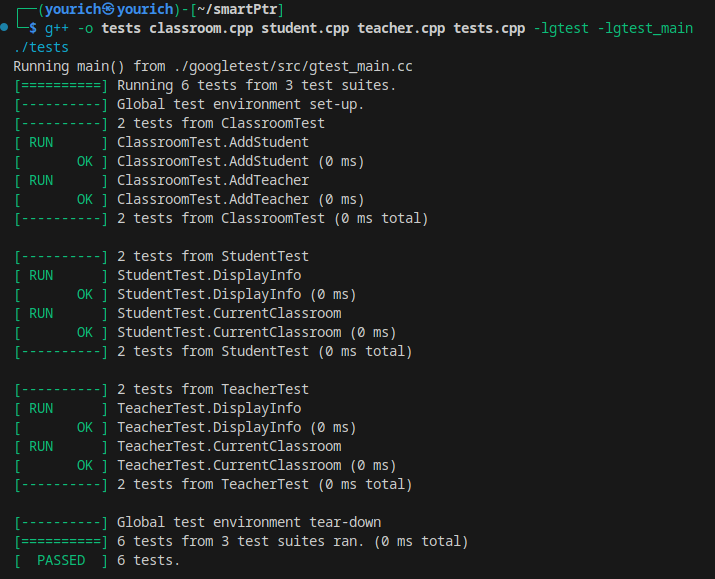


Рисунок 1 – Прохождение тестов

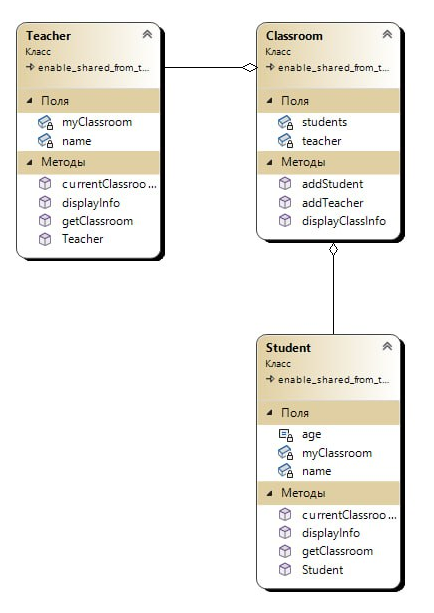


Рисунок 2 – UML-диаграмма проекта

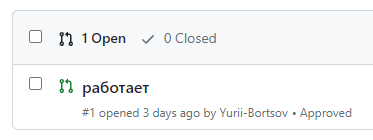


Рисунок 3 – Approve задания