**Технологии конструирования программного обеспечения**

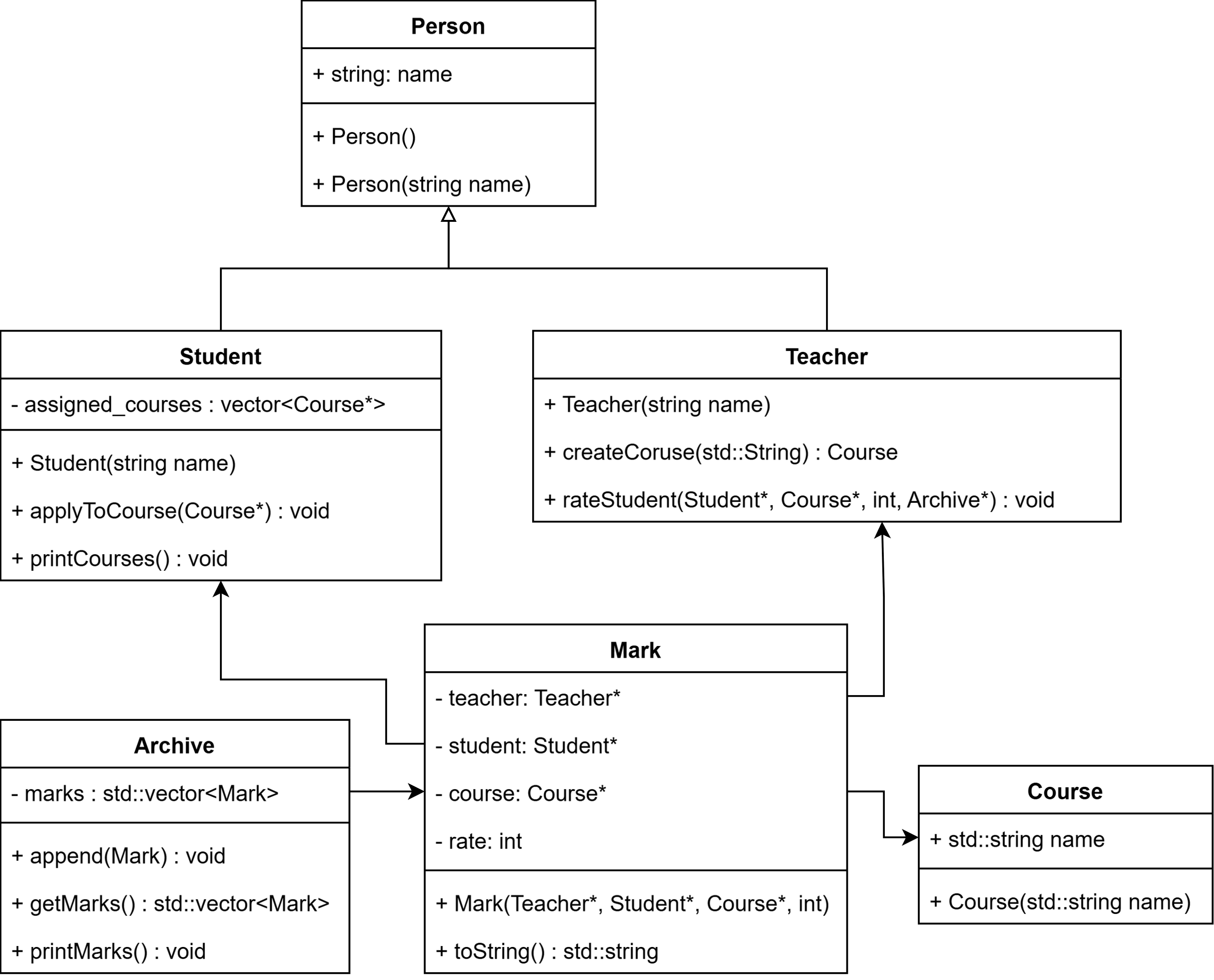
**Отчет по лабораторной работе № 1**

**Группа:** 221-3210 **Студент:** Обухов Алексей Сергеевич

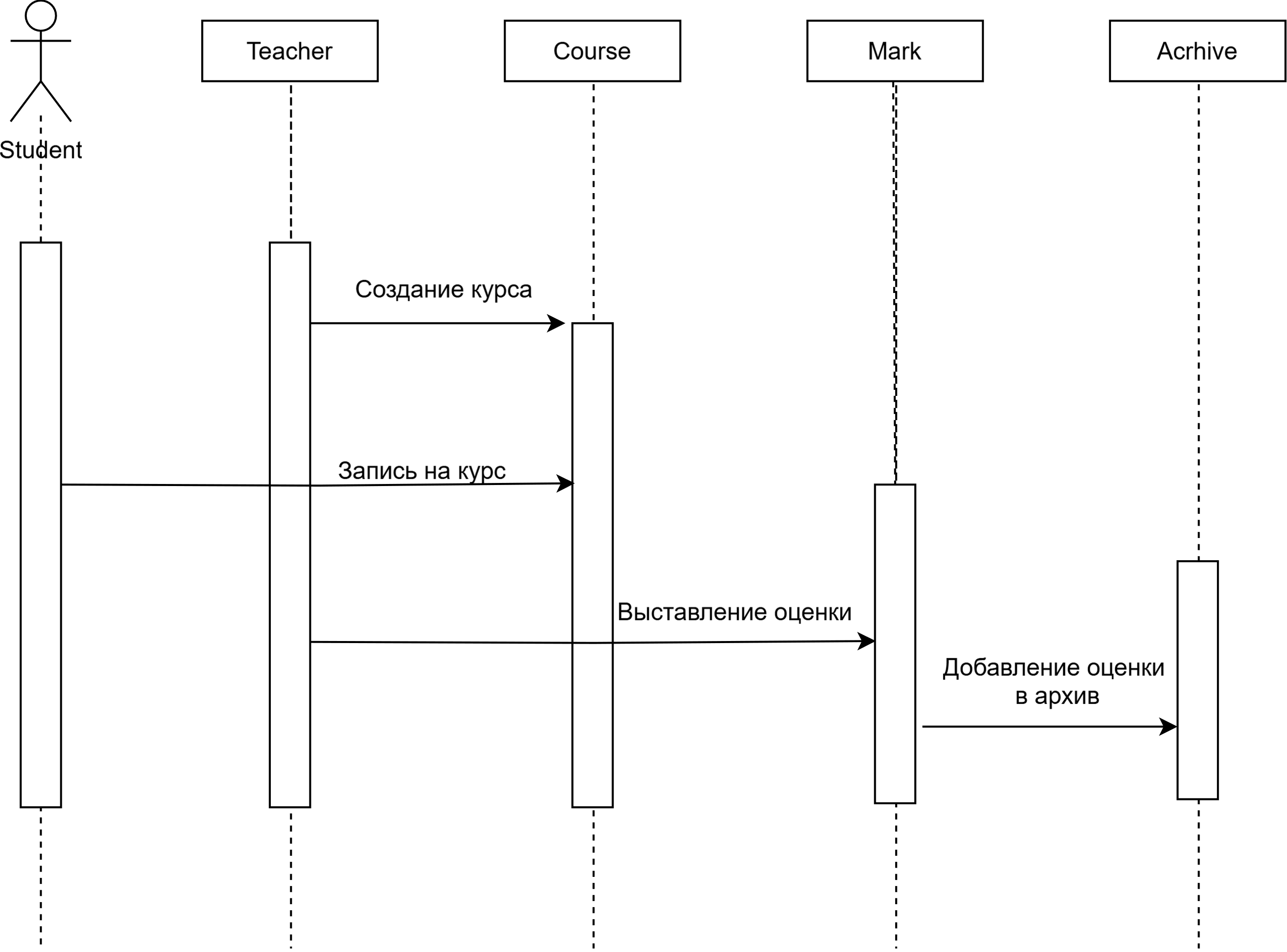
**Задание на лабораторную работу**

1. Система Факультатив. Преподаватель объявляет запись на Курс. Студент записывается на Курс, обучается и по окончании Преподаватель выставляет Оценку, которая сохраняется в Архиве. Студентов, Преподавателей и Курсов при обучении может быть несколько.

**Диаграмма классов**

****

**Диаграмма последовательности**

****

**Исходный код программы**

#include <iostream>

#include <string>

#include <vector>

// Класс для описания курса

class Course {

public:

// Название курса

std::string name;

Course(std::string name) {

this->name = name;

}

};

// Класс для описания личности

class Person {

public:

// Имя личности

std::string name;

Person() {

this->name = "NoName";

};

Person(std::string name) {

this->name = name;

}

};

// Класс для описания студента

class Student : public Person {

// Курсы, на которые записан студент

std::vector<Course\*> assigned\_courses;

public:

Student(std::string name) {

this->name = name;

}

// Фукнция записи на курс

void applyToCourse(Course\* course);

// Вывод всех курсов, на которые записан студент

void printCourses();

};

// Предварительная декларация классов

class Mark;

class Archive;

// Класс для описания преподавателя

class Teacher : public Person {

public:

Teacher(std::string name) {

this->name = name;

}

// Фукнция для создания курса

Course createCourse(std::string name);

// Фукнция для оценки студента по курсу

void rateStudent(Student\* student, Course\* course, int rate, Archive\* archive);

};

// Класс для описания оценки

class Mark {

private:

// Преподаватель, выставивший оценку

Teacher\* teacher;

// Студент, получивший оценку

Student\* student;

// Курс, по которому была выставлена оценка

Course\* course;

// Оценка

int rate;

public:

Mark(Teacher\* teacher, Student\* student, Course\* course, int rate);

// Преобразование оценки в строку

std::string toString();

};

// Класс для описания архива

class Archive {

private:

// Оценки, хранящиеся в архиве

std::vector<Mark> marks;

public:

// Добавление оценки в архив

void append(Mark mark);

// Получение копии архива

std::vector<Mark> getMarks();

// Вывод содержимого архива

void printMarks();

};

// Функция записи на курс

void Student::applyToCourse(Course\* course) {

assigned\_courses.push\_back(course);

}

// Функция вывода курсов, на которые записан студент

void Student::printCourses() {

std::cout << "Студент " + this->name + " записан на следующие курсы:\n";

for (int i = 0; i < assigned\_courses.size(); i++) {

std::cout << "#" << i << " " + assigned\_courses[i]->name << "\n";

}

}

// Функция создания курса

Course Teacher::createCourse(std::string name) {

return Course(name);

}

// Фукнция выставления оценки студенту по курсу

void Teacher::rateStudent(Student\* student, Course\* course, int rate, Archive\* archive) {

Mark mark = Mark(this, student, course, rate);

archive->append(mark);

}

Mark::Mark(Teacher\* teacher, Student\* student, Course\* course, int rate) {

this->teacher = teacher;

this->student = student;

this->course = course;

this->rate = rate;

}

// Функция преобразования класса в строку для удобного вывода

std::string Mark::toString() {

return "Оценка " + std::to\_string(rate) + " по курсу " + course->name + " была выставлена студенту "

+ student->name + " преподавателем " + teacher->name;

}

// Добавление оценки в архив

void Archive::append(Mark mark) {

marks.push\_back(mark);

}

// Получение копии архива

std::vector<Mark> Archive::getMarks() {

return marks;

}

// Вывод содержимого архива

void Archive::printMarks() {

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

std::cout << "#" << i << " " + marks[i].toString() << "\n";

}

}

int main() {

setlocale(LC\_ALL, "Russian");

// Создание студентов

Student studentA(std::string("Иван"));

Student studentB(std::string("Петр"));

// Создание преподавателей

Teacher teacherA(std::string("Васечкин Михаил Валерьевич"));

Teacher teacherB(std::string("Васильев Василий Васильевич"));

// Созание курсов

Course courseA = teacherA.createCourse(std::string("Проектирование ИС"));

Course courseB = teacherB.createCourse(std::string("Управление разработкой ПО"));

// Создание архива

Archive archive;

// Запись студентов на курсы

studentA.applyToCourse(&courseA);

studentA.applyToCourse(&courseB);

studentB.applyToCourse(&courseB);

// Вывод курсов, на которые записаны студенты

studentA.printCourses();

std::cout << "\n";

studentB.printCourses();

std::cout << "\n";

// Выставление оценок студентам

teacherA.rateStudent(&studentA, &courseA, 5, &archive);

teacherA.rateStudent(&studentA, &courseB, 4, &archive);

teacherA.rateStudent(&studentB, &courseB, 3, &archive);

// Вывод содержимого архива

archive.printMarks();

return 0;

}