МИНПРОСВЕЩЕНИЯ РОССИИ

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Нижегородский государственный педагогический университет

имени Козьмы Минина»

Факультет информационных технологий

Кафедра информационных систем и цифровых сервисов в управлении

**ОТЧЕТ**

по Индивидуальному проекту **на тему «Классы» и «Наследование»**

Направление подготовки: **Информационные системы и технологии**

Профиль подготовки: **Информационные системы и технологии**

**Выполнила:** Гусева Екатерина Александровна

**Группа:** ИСТ-23-2

г. Нижний Новгород

2025 год

**Введение**

**Ход работы:**

**Задание:**

Вариант 8

Описать класс «поезд», содержащий следующие закрытые поля:

• название пункта назначения;

• номер поезда (может содержать буквы и цифры);

• время отправления.

Предусмотреть свойства для получения состояния объекта.

Описать класс «вокзал», содержащий закрытый массив поездов. Обеспечить следующие возможности:

• вывод информации о поезде по номеру с помощью индекса;

• вывод информации о поездах, отправляющихся после введенного с клавиатуры

времени;

• перегруженную операцию сравнения, выполняющую сравнение времени отправления

двух поездов;

• вывод информации о поездах, отправляющихся в заданный пункт назначения.

Информация должна быть отсортирована по времени отправления. Написать программу, демонстрирующую все разработанные элементы классов.

**Код:**

#define NOMINMAX

#include <iostream>

#include <vector>

#include <string>

#include <algorithm>

#include <sstream>

#include <iomanip>

#include <locale.h>

#include <windows.h>

using namespace std;

class Train {

private:

string destination;

string number;

int departure\_time;

int parseTime(const string& time) {

int hours, minutes;

char colon;

istringstream iss(time);

if (!(iss >> hours >> colon >> minutes) || colon != ':' ||

hours < 0 || hours >= 24 || minutes < 0 || minutes >= 60) {

throw invalid\_argument("Неверный формат времени");

}

return hours \* 60 + minutes;

}

public:

Train(const string& dest, const string& num, const string& time)

: destination(dest), number(num) {

departure\_time = parseTime(time);

}

string getDestination() const { return destination; }

string getNumber() const { return number; }

int getDepartureTime() const { return departure\_time; }

string getTimeString() const {

ostringstream oss;

oss << setw(2) << setfill('0') << departure\_time / 60 << ":"

<< setw(2) << setfill('0') << departure\_time % 60;

return oss.str();

}

bool operator<(const Train& other) const {

return departure\_time < other.departure\_time;

}

};

class Station {

private:

vector<Train> trains;

public:

void addTrain(const Train& train) {

trains.push\_back(train);

}

void printByIndex(size\_t index) const {

if (index >= trains.size()) {

cout << "Неверный индекс!" << endl;

return;

}

const Train& t = trains[index];

cout << "Поезд №" << t.getNumber() << " в " << t.getDestination()

<< " отправляется в " << t.getTimeString() << endl;

}

void printAfterTime(const string& time) const {

vector<Train> filtered;

try {

int target = Train("", "", time).getDepartureTime();

for (const auto& t : trains) {

if (t.getDepartureTime() > target) filtered.push\_back(t);

}

}

catch (...) {

cout << "Ошибка формата времени!" << endl;

return;

}

sort(filtered.begin(), filtered.end());

if (filtered.empty()) {

cout << "Нет подходящих поездов" << endl;

return;

}

cout << "Поезда после " << time << ":\n";

for (const auto& t : filtered) {

cout << "- №" << t.getNumber() << " в " << t.getDestination()

<< " (" << t.getTimeString() << ")\n";

}

}

void printByDestination(const string& dest) const {

vector<Train> result;

for (const auto& t : trains) {

if (t.getDestination() == dest) result.push\_back(t);

}

sort(result.begin(), result.end());

if (result.empty()) {

cout << "Нет поездов в " << dest << endl;

return;

}

cout << "Поезда в " << dest << ":\n";

for (const auto& t : result) {

cout << "- №" << t.getNumber() << " в " << t.getTimeString() << endl;

}

}

const Train& operator[](size\_t index) const {

return trains.at(index);

}

void printAllWithIndexes() const {

if (trains.empty()) {

cout << "\nНа вокзале нет поездов!\n";

return;

}

cout << "\nСписок доступных поездов:\n";

cout << "============================================\n";

for (size\_t i = 0; i < trains.size(); ++i) {

cout << " Индекс: " << i << endl;

cout << " Номер поезда: " << trains[i].getNumber() << endl;

cout << " Пункт назначения: " << trains[i].getDestination() << endl;

cout << " Время отправления: " << trains[i].getTimeString() << endl;

cout << "=========================================\n";

}

}

size\_t getTotalTrains() const {

return trains.size();

}

};

int getIntInput(const string& prompt) {

int value;

while (true) {

cout << prompt;

if (cin >> value) break;

cin.clear();

cin.ignore((numeric\_limits<streamsize>::max)(), '\n');

cout << "Ошибка ввода! Введите целое число.\n";

}

cin.ignore();

return value;

}

string getStringInput(const string& prompt) {

string value;

cout << prompt;

getline(cin, value);

return value;

}

void interactiveTrainViewer(Station& station) {

while (true) {

cout << "\nВыберите действие:\n";

cout << "1. Просмотреть поезд по индексу\n";

cout << "2. Вернуться в главное меню\n";

cout << "Ваш выбор: ";

int choice;

cin >> choice;

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

if (choice == 2) break;

if (choice == 1) {

int index = getIntInput("Введите индекс поезда: ");

if (index >= 0 && index < station.getTotalTrains()) {

cout << "\nИнформация о выбранном поезде:\n";

station.printByIndex(index);

}

else {

cout << "Ошибка: неверный индекс!\n";

}

}

else {

cout << "Неверный выбор!\n";

}

cout << "\nНажмите Enter для продолжения...";

cin.get();

}

}

int main() {

setlocale(LC\_ALL, "Russian");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Station station;

// Добавление тестовых данных

station.addTrain(Train("Москва", "101", "08:30"));

station.addTrain(Train("Санкт-Петербург", "234", "14:15"));

station.addTrain(Train("Москва", "321", "09:45"));

station.addTrain(Train("Казань", "445", "23:50"));

// Главное меню

while (true) {

cout << "\nГлавное меню:\n";

cout << "1. Просмотр поездов по индексам\n";

cout << "2. Поиск поездов по времени\n";

cout << "3. Сравнение времени отправления\n";

cout << "4. Поиск по пункту назначения\n";

cout << "5. Выход\n";

cout << "Выберите действие: ";

int choice;

cin >> choice;

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

if (choice == 5) break;

switch (choice) {

case 1:

interactiveTrainViewer(station);

break;

case 2: {

string time = getStringInput("Введите время (HH:MM): ");

station.printAfterTime(time);

break;

}

case 3: {

int index1 = getIntInput("\nВведите первый индекс для сравнения: ");

int index2 = getIntInput("Введите второй индекс для сравнения: ");

if (index1 >= station.getTotalTrains() || index2 >= station.getTotalTrains()) {

cout << "Ошибка: неверные индексы!\n";

}

else {

const Train& t1 = station[index1];

const Train& t2 = station[index2];

cout << "\nСравнение времени отправления:\n";

cout << "Поезд " << index1 << " (" << t1.getTimeString() << ") "

<< (t1 < t2 ? "раньше" : "позже")

<< " чем поезд " << index2 << " (" << t2.getTimeString() << ")\n";

}

break;

}

case 4: {

string dest = getStringInput("Введите пункт назначения: ");

station.printByDestination(dest);

break;

}

default:

cout << "Неверный выбор!\n";

}

cout << "\nНажмите Enter для продолжения...";

cin.get();

}

return 0;

}