МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА ИНФОРМАЦИОННО-СЕТЕВЫХ ТЕХНОЛОГИЙ

КУРСОВАЯ РАБОТА (ПРОЕКТ)   
ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| асс. |  |  |  | Рождественская К.Н. |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ |
| ПАТТЕРНЫ |
| по дисциплине: ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 7541ВЦ |  |  |  | А.В. Смирнов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2018

Содержание

[ПАТТЕРНЫ - 1 -](#_Toc515117997)

[1. Постановка задачи. - 3 -](#_Toc515117998)

[2. Формализация. - 3 -](#_Toc515117999)

[3. Описание паттерна. - 3 -](#_Toc515118000)

[4. Работа программы. - 4 -](#_Toc515118001)

[5. Листинг программы. - 6 -](#_Toc515118002)

[5.1. Код main.cpp - 6 -](#_Toc515118003)

[5.2. Код ticketwindow.h - 6 -](#_Toc515118004)

[5.3. Код ticketwindow.cpp - 7 -](#_Toc515118005)

[5.4. Код train.h - 8 -](#_Toc515118006)

[5.5. Код train.cpp - 8 -](#_Toc515118007)

[5.6. Код plane.h - 9 -](#_Toc515118008)

[5.7. Код plane.cpp - 9 -](#_Toc515118009)

[6. Выводы. - 9 -](#_Toc515118014)

# Постановка задачи.

Создать систему запросов «Касса → Поезд → Самолет». Пользователя интересует покупка билета, которая сопровождается требованиями: дата, стоимость, пункт назначения. Пользователь использует кассу для нахождения подходящего транспорта. Касса обращается сначала к билетам поездов, если доступных вариантов нет, то к билетам самолетов. Использовать паттерн «Цепочка обязанностей». Данные о доступных днях, маршрутах и стоимости для поезда и самолета берутся из внешних входных файлов «train», «plane» соответственно. Все купленные билеты сохраняются в выходном файле.

# Формализация.

* Использовать паттерн «Цепочка обязанностей»
* Разделить классы на заголовочные и исполняемые файлы
* Данные берутся из внешних файлов
* Результаты должны записываться в файл
* Программа должна быть написана на С++

# Описание паттерна.

Назначение паттерна Chain of Responsibility:

* Паттерн Chain of Responsibility позволяет избежать жесткой зависимости отправителя запроса от его получателя, при этом запрос может быть обработан несколькими объектами. Объекты-получатели связываются в цепочку. Запрос передается по этой цепочке, пока не будет обработан.
* Вводит конвейерную обработку для запроса с множеством возможных обработчиков.
* Объектно-ориентированный связанный список с рекурсивным обходом.

Паттерн Chain of Responsibility связывает в цепочку объекты-получатели, а затем передает запрос-сообщение от одного объекта к другому до тех пор, пока не достигнет объекта, способного его обработать. Число и типы объектов-обработчиков заранее неизвестны, они могут настраиваться динамически. Механизм связывания в цепочку использует рекурсивную композицию, что позволяет использовать неограниченное число обработчиков.

Паттерн Chain of Responsibility упрощает взаимосвязи между объектами. Вместо хранения ссылок на всех кандидатов-получателей запроса, каждый отправитель хранит единственную ссылку на начало цепочки, а каждый получатель имеет единственную ссылку на своего преемника - последующий элемент в цепочке.

# Работа программы.

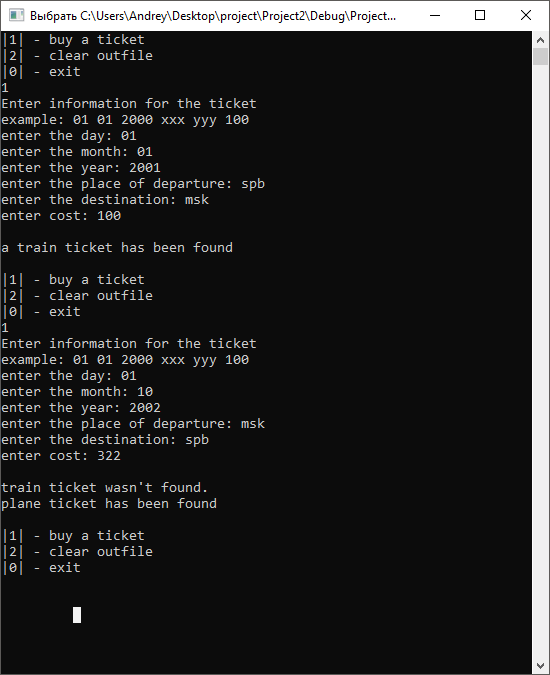


Рисунок 1 – Ввод информации и поиск в файле.

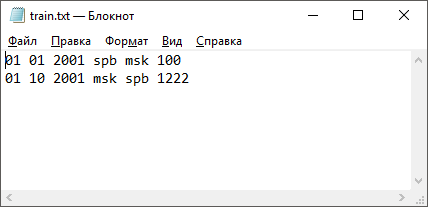


Рисунок 2 – Файл с билетами на поезд.

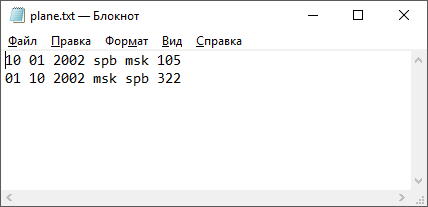


Рисунок 3 – Файл с билетами на самолет.

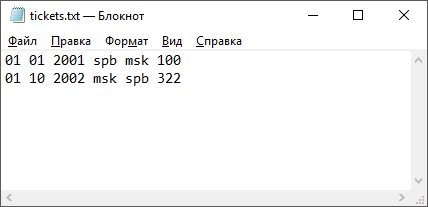


Рисунок 4 – Файл с купленными билетами.

# Листинг программы.

# main.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <fstream>

#include "ticketwindow.h"

#include "train.h"

#include "plane.h"

using namespace std;

char ticketInformation[100];

int main() {

train first;

plane second;

first.add(&second);

second.setNext(&first);

int button = 1;

while (button) {

cout << "|1| - buy a ticket\n";

cout << "|2| - clear outfile\n";

cout << "|0| - exit\n";

cin >> button;

switch (button) {

case 1: {

first.setInformation();

first.handle(1);

break;

}

case 2: {ofstream file("files\\tickets.txt", ios\_base::trunc); file.close(); break; }

case 0: 0;

}

}

return 0;

}

# ticketwindow.h

#ifndef TICKETWINDOW\_H

#define TICKETWINDOW\_H

extern char ticketInformation[100];

class ticketwindow {

ticketwindow \*next;

private:

char buffer[20];

public:

ticketwindow();

void add(ticketwindow \*n);

void setInformation();

void setNext(ticketwindow \*n);

virtual void handle(int i);

~ticketwindow();

};

#endif TICKETWINDOW\_H

# ticketwindow.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include "ticketwindow.h"

using namespace std;

ticketwindow::ticketwindow() {

next = 0;

}

void ticketwindow::add(ticketwindow \*n) {

if (next)

next->add(n);

else

next = n;

}

void ticketwindow::setInformation() {

cout << "Enter information for the ticket\nexample: 01 01 2000 xxx yyy 100\n";

cout << "enter the day: ";

cin >> ticketInformation;

cout << "enter the month: ";

cin >> buffer;

strcat(ticketInformation, " ");

strcat(ticketInformation, buffer);

cout << "enter the year: ";

cin >> buffer;

strcat(ticketInformation, " ");

strcat(ticketInformation, buffer);

cout << "enter the place of departure: ";

cin >> buffer;

strcat(ticketInformation, " ");

strcat(ticketInformation, buffer);

cout << "enter the destination: ";

cin >> buffer;

strcat(ticketInformation, " ");

strcat(ticketInformation, buffer);

cout << "enter cost: ";

strcat(ticketInformation, " ");

cin >> buffer;

strcat(ticketInformation, buffer);

}

void ticketwindow::setNext(ticketwindow \*n) {

next = n;

}

void ticketwindow::handle(int i) {

next->handle(i);

}

ticketwindow::~ticketwindow() {}

# train.h

#define \_CRT\_SECURE\_NO\_WARNINGS

#ifndef TRAIN\_H

#define TRAIN\_H

#include "ticketwindow.h"

class train : public ticketwindow {

private:

char ticketInformationFromFile[100];

public:

train();

virtual void handle(int i);

~train();

};

#endif TRAIN\_H

# train.cpp

#include <iostream>

#include <fstream>

#include "train.h"

#include "ticketwindow.h"

using namespace std;

train::train() {};

void train::handle(int i) {

ifstream file("files\\train.txt");

if (!file.is\_open())

cout << "file not found" << endl;

else {

while (!file.eof()) {

file.getline(ticketInformationFromFile, sizeof(ticketInformationFromFile), '\n');

if (strcmp(ticketInformationFromFile, ticketInformation) == 0) {

cout << "\na train ticket has been found\n" << endl;

ofstream fout("files\\tickets.txt", ios\_base::app);

fout << ticketInformationFromFile;

fout << "\n";

fout.close();

break;

}

}

if ((file.eof() == 1) && (strcmp(ticketInformationFromFile, ticketInformation) != 0)) {

cout << "\ntrain ticket wasn't found." << endl;

ticketwindow::handle(i);

}

}

file.close();

}

train::~train() {}

# plane.h

#define \_CRT\_SECURE\_NO\_WARNINGS

#ifndef PLANE\_H

#define PLANE\_H

#include "ticketwindow.h"

class plane : public ticketwindow {

private:

char ticketInformationFromFile[100];

public:

plane();

virtual void handle(int i);

~plane();

};

#endif PLANE\_H

# plane.cpp

#include <iostream>

#include <fstream>

#include "plane.h"

#include "ticketwindow.h"

using namespace std;

plane::plane() {};

void plane::handle(int i) {

ifstream file("files\\plane.txt");

if (!file.is\_open())

cout << "file not found" << endl;

else {

while (!file.eof()) {

file.getline(ticketInformationFromFile, sizeof(ticketInformationFromFile), '\n');

if (strcmp(ticketInformationFromFile, ticketInformation) == 0) {

cout << "plane ticket has been found\n" << endl;

ofstream fout("files\\tickets.txt", ios\_base::app);

fout << ticketInformationFromFile;

fout << "\n";

fout.close();

break;

}

}

if ((file.eof() == 1) && (strcmp(ticketInformationFromFile, ticketInformation) != 0)) {

cout << "plane ticket wasn't found.\n" << endl;

}

}

file.close();

}

plane::~plane() {}

# Выводы.

В проделанной работе был изучен паттерн «Цепочка обязанностей».