МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное учреждения высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт компьютерных технологий и информационной безопасности

Кафедра математического обеспечения и применения ЭВМ

**ЛАБОРАТОРНАЯ РАБОТА № 4**

по дисциплине

**«Объектно-ориентированное программирование»**

на тему:

**«Контейнеры STL»**

*Вариант № 9*

Выполнил:

Студент группы

КТбо2-7

Шубенков С. С.

Проверил:

Тарасов С. А.

Оценка

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Таганрог 2020

# **1 ФОРМУЛИРОВКА ЗАДАНИЯ**

Создать класс “Общежитие (Hostel)” с полями: ФИО студента, номер группы, номер комнаты, срок окончания проживания. Вывести информацию о студентах задаваемой группы, срок проживания которых заканчивается в задаваемом году.

Исходным материалом является текстовый файл, в котором записана некоторая информация.

# **2 СПЕЦИФИКАЦИЯ КЛАССОВ**

class Hostel

{

public:

Hostel();

Hostel(std::string& name, int numberGroup, int roomNumber, int endYearResidence);

const std::string& GetName() const;

int GetNumberGroup() const;

int GetRoomNumber() const;

int GetEndYearResidence() const;

int GetId() const;

void SetId(int id);

private:

int \_id = 0;

std::string \_name;

int \_numberGroup;

int \_roomNumber;

int \_endYearResidence;

};

class HostelStorage

{

public:

~HostelStorage();

std::vector<Hostel\*> GetHostelByGroup(int group);

std::vector<Hostel\*> GetHostelByRoom(int room);

std::vector<Hostel\*> GetHostelByYear(int year);

std::vector<Hostel\*> GetHostelByName(std::string& name);

std::vector<Hostel\*> GetHostelByGroupAndYear(int group, int year);

const std::map<int, Hostel\*>& GetHostels() const;

const Hostel& GetHostelById(int id) const;

void AddHostel(Hostel\* hostel);

private:

std::map<int, Hostel\*> \_hostelById;

std::map<int, std::set<int>> \_hostelByGroup;

std::map<int, std::set<int>> \_hostelByRoom;

std::map<int, std::set<int>> \_hostelByYear;

std::map<std::string, std::set<int>> \_hostelByName;

int \_currectId = 0;

};

class HostelParse

{

public:

static void Parse(HostelStorage\* hos);

private:

constexpr static const char\* \_file = "Hostel.txt";

};

class Console

{

public:

void Run();

private:

HostelStorage \_hosStorage;

int \_menuItemNumber = -1;

static const int \_numPointMenu = 8;

const string \_menu[\_numPointMenu] =

{

"Add new hostel",

"Search by name",

"Search by group",

"Search by room",

"Search by expiration date",

"Searching in a group with an expiring residency",

"Print all hostels",

"Complete the program"

};

const string \_errInput = "Incorrect input. Try again\n\n";

void Input(int& v);

void Input(string& v);

void PrintHostel(Hostel& hos);

void PrintAllHostel();

void PrintTitle();

void PrintGroupHostel(vector<Hostel\*>& hos);

void SearchName();

void SearchGroup();

void SearchRoom();

void SearchYear();

void SearchGroupAndYear();

void CreateHostel();

};

# **4 ДИАГРАММА КЛАССОВ**

UML диаграмма классов изображена на рисунке 1.

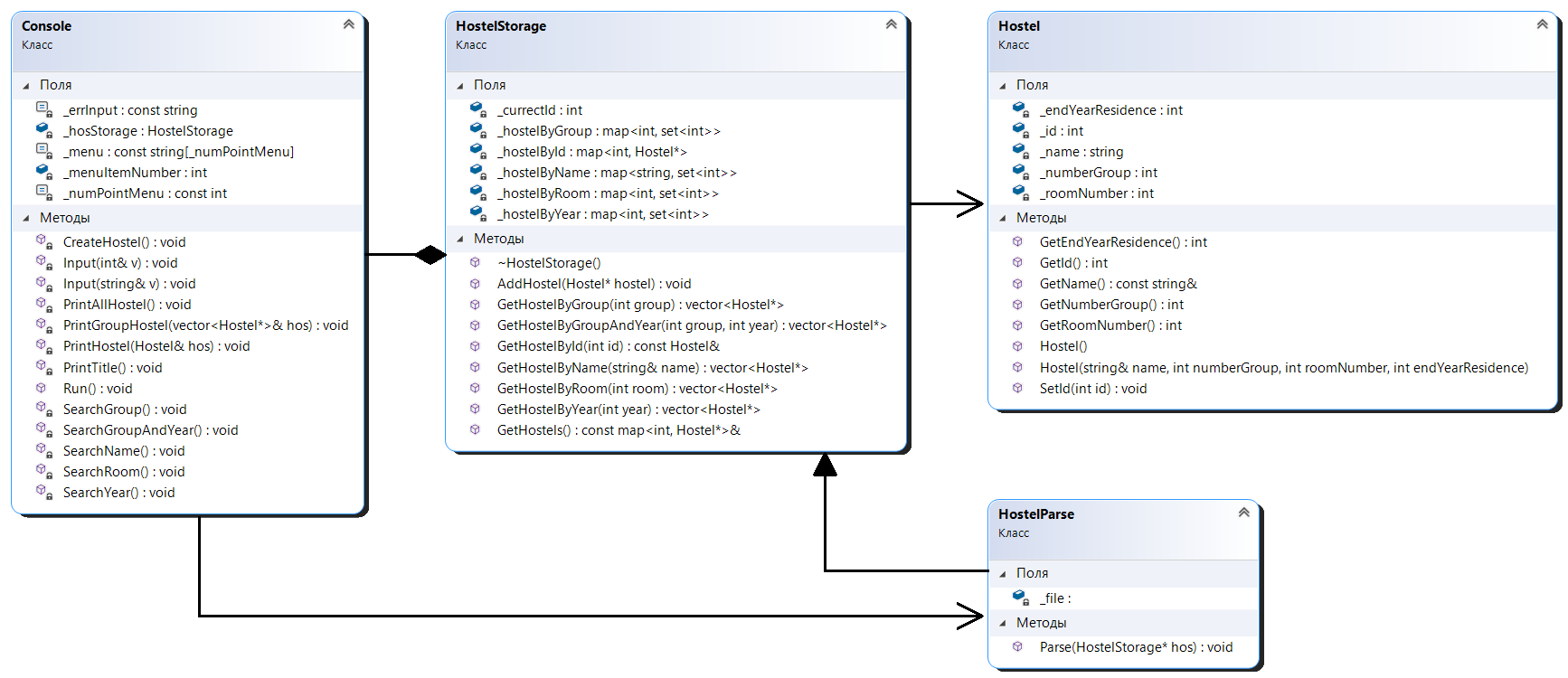


Рисунок 1 – Диаграмма классов

**ЛИСТИНГ ПРОГРАММЫ**

Hostel.h

#pragma once

#include <string>

class Hostel

{

public:

Hostel() = default;

Hostel(std::string& name, int numberGroup, int roomNumber, int endYearResidence)

{

if (name.length() == 0)

throw std::exception("Empty name");

if (numberGroup <= 0)

throw std::exception("Impossible group number");

if (roomNumber <= 0)

throw std::exception("Impossible room number");

if (endYearResidence <= 1980)

throw std::exception("Impossible end year residence");

\_name = name;

\_numberGroup = numberGroup;

\_roomNumber = roomNumber;

\_endYearResidence = endYearResidence;

}

const std::string& GetName() const { return \_name; }

int GetNumberGroup() const { return \_numberGroup; }

int GetRoomNumber() const { return \_roomNumber; }

int GetEndYearResidence() const { return \_endYearResidence; }

int GetId() const { return \_id; }

void SetId(int id) { \_id = id; }

private:

int \_id = 0;

std::string \_name; //имя студента

int \_numberGroup; //номер группы

int \_roomNumber; //номер комнаты

int \_endYearResidence; //год, до которого оплачено общежитие

};

HostelStorage.h

#pragma once

#include <map>

#include <set>

#include <vector>

#include "Hostel.h"

class HostelStorage

{

public:

~HostelStorage()

{

for (auto& elem : \_hostelById)

delete elem.second;

}

std::vector<Hostel\*> GetHostelByGroup(int group);

std::vector<Hostel\*> GetHostelByRoom(int room);

std::vector<Hostel\*> GetHostelByYear(int year);

std::vector<Hostel\*> GetHostelByName(std::string& name);

std::vector<Hostel\*> GetHostelByGroupAndYear(int group, int year); //поиск по группе и дате окончания проживания

const std::map<int, Hostel\*>& GetHostels() const;

const Hostel& GetHostelById(int id) const;

void AddHostel(Hostel\* hostel);

private:

std::map<int, Hostel\*> \_hostelById; //по айди

std::map<int, std::set<int>> \_hostelByGroup; //по группам

std::map<int, std::set<int>> \_hostelByRoom; //по комнатам

std::map<int, std::set<int>> \_hostelByYear; //по дате оплаты рожтвания

std::map<std::string, std::set<int>> \_hostelByName; //по имени

int \_currectId = 0;

};

HostelStorage.cpp

#include "HostelStorage.h"

void HostelStorage::AddHostel(Hostel\* hostel)

{

hostel->SetId(\_currectId);

\_hostelByGroup[hostel->GetNumberGroup()].insert(\_currectId);

\_hostelById[\_currectId] = hostel;

\_hostelByName[hostel->GetName()].insert(\_currectId);

\_hostelByRoom[hostel->GetRoomNumber()].insert(\_currectId);

\_hostelByYear[hostel->GetEndYearResidence()].insert(\_currectId);

\_currectId++;

}

const Hostel& HostelStorage::GetHostelById(int id) const

{

auto iter = \_hostelById.find(id);

if (iter == \_hostelById.end())

{

throw std::exception("Id does not exist");

}

return \*iter->second;

}

const std::map<int, Hostel\*>& HostelStorage::GetHostels() const

{

return \_hostelById;

}

std::vector<Hostel\*> HostelStorage::GetHostelByName(std::string& name)

{

auto iter = \_hostelByName.find(name);

if (iter == \_hostelByName.end())

{

throw std::exception("Name does not exist");

}

std::vector<Hostel\*> hos;

for (auto& elem : iter->second)

{

hos.push\_back(\_hostelById[elem]);

}

return hos;

}

std::vector<Hostel\*> HostelStorage::GetHostelByGroup(int group)

{

auto iter = \_hostelByGroup.find(group);

if (iter == \_hostelByGroup.end())

{

throw std::exception("Group does not exist");

}

std::vector<Hostel\*> hos;

for (auto& elem : iter->second)

{

hos.push\_back(\_hostelById[elem]);

}

return hos;

}

std::vector<Hostel\*> HostelStorage::GetHostelByRoom(int room)

{

auto iter = \_hostelByRoom.find(room);

if (iter == \_hostelByRoom.end())

{

throw std::exception("Room does not exist");

}

std::vector<Hostel\*> hos;

for (auto& elem : iter->second)

{

hos.push\_back(\_hostelById[elem]);

}

return hos;

}

std::vector<Hostel\*> HostelStorage::GetHostelByYear(int year)

{

auto iter = \_hostelByYear.find(year);

if (iter == \_hostelByYear.end())

{

throw std::exception("Year does not exist");

}

std::vector<Hostel\*> hos;

for (auto& elem : iter->second)

{

hos.push\_back(\_hostelById[elem]);

}

return hos;

}

std::vector<Hostel\*> HostelStorage::GetHostelByGroupAndYear(int group, int year)

{

auto iter = \_hostelByGroup.find(group);

if (iter == \_hostelByGroup.end())

{

throw std::exception("Group does not exist");

}

std::vector<Hostel\*> hos;

for (auto& elem : iter->second)

{

if (\_hostelById[elem]->GetEndYearResidence() <= year)

{

hos.push\_back(\_hostelById[elem]);

}

}

if (hos.empty())

{

throw std::exception("There are no expiring students in this group");

}

return hos;

}

HostelParse.h

#pragma once

#include <string>

#include <fstream>

#include <sstream>

#include "HostelStorage.h"

class HostelParse

{

public:

static void Parse(HostelStorage\* hos);

private:

constexpr static const char\* \_file = "Hostel.txt";

};

HostelParse.cpp

#include "HostelParse.h"

void HostelParse::Parse(HostelStorage\* hos)

{

std::ifstream infile(\_file);

if (!infile.is\_open())

{

throw std::exception("File does not exist");

}

std::string line;

std::string name;

int numberGroup;

int roomNumber;

int endYearResidence;

while (std::getline(infile, line))

{

std::istringstream in(line);

in >> name;

in >> numberGroup;

in >> roomNumber;

in >> endYearResidence;

hos->AddHostel(new Hostel(name, numberGroup, roomNumber, endYearResidence));

}

}

Console.h

#pragma once

#include <iostream>

#include "HostelParse.h"

using namespace std;

class Console

{

public:

void Run();

private:

HostelStorage \_hosStorage;

int \_menuItemNumber = -1;

static const int \_numPointMenu = 8;

const string \_menu[\_numPointMenu] =

{

"Add new hostel",

"Search by name",

"Search by group",

"Search by room",

"Search by expiration date",

"Searching in a group with an expiring residency",

"Print all hostels",

"Complete the program"

};

const string \_errInput = "Incorrect input. Try again\n\n";

void Input(int& v);

void Input(string& v);

void PrintHostel(Hostel& hos);

void PrintAllHostel();

void PrintTitle();

void PrintGroupHostel(vector<Hostel\*>& hos);

void SearchName();

void SearchGroup();

void SearchRoom();

void SearchYear();

void SearchGroupAndYear();

void CreateHostel();

};

Console.cpp

# #include "Console.h"

void Console::Run()

{

try

{

HostelParse::Parse(&\_hosStorage);

}

catch (exception e)

{

cout << "Error. " << e.what();

\_menuItemNumber = \_numPointMenu;

}

while (\_menuItemNumber != \_numPointMenu)

{

for (int i = 0; i < \_numPointMenu; i++)

{

cout << i + 1 << ". " << \_menu[i] << endl;

}

Input(\_menuItemNumber);

system("cls");

switch (\_menuItemNumber)

{

case 1:

CreateHostel();

break;

case 2:

SearchName();

break;

case 3:

SearchGroup();

break;

case 4:

SearchRoom();

break;

case 5:

SearchYear();

break;

case 6:

SearchGroupAndYear();

break;

case 7:

PrintAllHostel();

break;

case 8:

break;

default:

cout << \_errInput;

}

}

}

void Console::Input(int& v)

{

cout << "> ";

cin >> v;

if (cin.fail())

{

cin.clear();

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

}

}

void Console::Input(string& v)

{

cout << "> ";

cin >> v;

if (cin.fail())

{

cin.clear();

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

}

}

void Console::PrintHostel(Hostel& hos)

{

cout.width(5);

cout << left << hos.GetId();

cout.width(12);

cout << left << hos.GetName();

cout.width(7);

cout << left << hos.GetNumberGroup();

cout.width(7);

cout << left << hos.GetRoomNumber();

cout.width(15);

cout << right << hos.GetEndYearResidence();

cout << '\n';

}

void Console::PrintAllHostel()

{

PrintTitle();

for (auto& elem : \_hosStorage.GetHostels())

{

PrintHostel(\*elem.second);

}

cout << '\n';

}

void Console::PrintTitle()

{

cout.width(5);

cout << left << "ID";

cout.width(12);

cout << left << "Name";

cout.width(7);

cout << left << "Group";

cout.width(7);

cout << left << "Room";

cout.width(15);

cout << left << "Expiration date";

cout << '\n';

}

void Console::PrintGroupHostel(vector<Hostel\*>& hos)

{

PrintTitle();

for (auto& elem : hos)

{

PrintHostel(\*elem);

}

cout << '\n';

}

void Console::SearchName()

{

cout << "Enter a name (no spaces)\n";

string name;

Input(name);

vector<Hostel\*> hos;

system("cls");

try

{

hos = \_hosStorage.GetHostelByName(name);

}

catch(exception e)

{

cout << "Error. " << e.what() << "\n\n";

return;

}

PrintGroupHostel(hos);

}

void Console::SearchGroup()

{

cout << "Enter the number group\n";

int group;

Input(group);

vector<Hostel\*> hos;

system("cls");

try

{

hos = \_hosStorage.GetHostelByGroup(group);

}

catch (exception e)

{

cout << "Error. " << e.what() << "\n\n";

return;

}

PrintGroupHostel(hos);

}

void Console::SearchRoom()

{

cout << "Enter the number room\n";

int room;

Input(room);

vector<Hostel\*> hos;

system("cls");

try

{

hos = \_hosStorage.GetHostelByRoom(room);

}

catch (exception e)

{

cout << "Error. " << e.what() << "\n\n";

return;

}

PrintGroupHostel(hos);

}

void Console::SearchYear()

{

cout << "Enter the year of the end of residency\n";

int year;

Input(year);

vector<Hostel\*> hos;

system("cls");

try

{

hos = \_hosStorage.GetHostelByYear(year);

}

catch (exception e)

{

cout << "Error. " << e.what() << "\n\n";

return;

}

PrintGroupHostel(hos);

}

void Console::SearchGroupAndYear()

{

cout << "Enter the number group\n";

int group;

Input(group);

cout << "Enter the year of the end of residency\n";

int year;

Input(year);

vector<Hostel\*> hos;

system("cls");

try

{

hos = \_hosStorage.GetHostelByGroupAndYear(group, year);

}

catch (exception e)

{

cout << "Error. " << e.what() << "\n\n";

return;

}

PrintGroupHostel(hos);

}

void Console::CreateHostel()

{

cout << "Enter a name (no spaces)\n";

string name;

Input(name);

cout << "Enter the number group\n";

int group;

Input(group);

cout << "Enter the number room\n";

int room;

Input(room);

cout << "Enter the year of the end of residency\n";

int year;

Input(year);

system("cls");

Hostel\* hos = nullptr;

try

{

hos = new Hostel(name, group, room, year);

}

catch (exception e)

{

cout << "Error. " << e.what() << "\n\n";

return;

}

\_hosStorage.AddHostel(hos);

}

main.cpp

#include <vld.h>

#include "Console.h"

int main()

{

Console con;

con.Run();

return 0;

}