Министерство высшего образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет» (ПНИПУ)**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ

Тема: «Лабораторная работа №8»

Выполнил

Студент группы РИС-22-2б

Прядеин И.А.

Проверил доцент кафедры

ИТАС

Полякова О. А.

Пермь 2023

Постановка задачи

1. Определить иерархию пользовательских классов. Во главе иерархии должен стоять абстрактный класс с чисто виртуальными методами ввода и вывода информации об атрибутах объектов.
2. Реализовать конструкторы, деструктор, операцию присваивания, селекторы и модификаторы.
3. Определить класс-группу на основе структуры, указанной в варианте.
4. Для группы реализовать конструкторы, деструктор, методы для добавления и удаления элементов в группу, метод просмотра группы, перегрузить операцию для получения информации о размере группы.
5. Определить класс Диалог - наследника группы, в котором реализовать методы для обработки событий.
6. Добавить методы для обработки событий группой и объектами пользовательских классов.
7. Написать тестирующую программу.
8. Нарисовать диаграмму классов и диаграмму объектов.

**Вариант 8:**

Базовый класс:

ЧЕЛОВЕК (Person)

Имя - string

Возраст - int

Производный класс

АБИТУРИЕНТ (ABITURIENT)

Количество баллов - int

Специальность - string

Группа - Список (List).

Комманды:

* Создать группу (формат команды: m количество элементов группы).
* Добавить элемент в группу (формат команды: +)
* Удалить элемент из группы (формат команды: -)
* Вывести информацию об элементах группы (формат команды: s)
* Вывести информацию о среднем возрасте (формат команды: z)
* Конец работы (формат команды: q)

**Исходный код программы:**

**Файл “Lab\_8.cpp”:**

#include "Dialog.cpp"

int main() {

// Person \*A = new Person;

// A->Input();

// Abiturient \*B = new Abiturient;

// B->Input();

// List list(3);

// list.Show();

// Object\* p = A;

// list.Push\_back(p);

// p = B;

// list.Push\_back(p);

// list.Show();

// list.Pop\_back();

// list.Show();

Dialog D;

D.Execute();

return 0;

}

**Файл “Object.h”:**

#pragma once

class Object {

public:

Object();

virtual void Show() = 0;

virtual void Input() = 0;

virtual ~Object();

};

**Файл “Object.cpp”:**

#include "Object.h"

Object::Object() {

}

Object::~Object() {

}

**Файл “Person.h”:**

#include "Object.cpp"

#include <string>

#include <iostream>

using namespace std;

class Person : public Object {

public:

Person();

virtual ~Person();

void Show();

void Input();

Person(string, int);

Person(const Person&);

string Get\_name() {

return name;

}

int Get\_age() {

return age;

}

void Set\_name(string n) {

name = n;

}

void Set\_age(int a) {

age = a;

}

Person& operator =(const Person&);

protected:

string name;

int age;

};

**Файл “Person.cpp”:**

#include "Person.h"

Person::Person() {

name = "";

age = 0;

}

Person::~Person() {

}

Person::Person(string n, int a) {

name = n;

age = a;

}

Person::Person(const Person& other) {

name = other.name;

age = other.age;

}

Person& Person::operator =(const Person& other) {

if (&other == this)

return \*this;

name = other.name;

age = other.age;

return \*this;

}

void Person::Show() {

cout << "Name: " << name;

cout << "\nAge: " << age;

cout << "\n";

}

void Person::Input() {

cout << "Name: ";

cin >> name;

cout << "Age: ";

cin >> age;

cout << "\n";

}

**Файл “Abiturient.h”:**

#include "Person.cpp"

class Abiturient : public Person {

public:

Abiturient();

Abiturient(string, int, int, string);

~Abiturient();

Abiturient(const Abiturient&);

void Show();

void Input();

int Get\_Number\_of\_points() {

return Number\_of\_points;

}

string Get\_speciality() {

return speciality;

}

void Set\_Number\_of\_points(int np) {

Number\_of\_points = np;

}

void Set\_speciality(string s) {

speciality = s;

}

Abiturient& operator =(const Abiturient&);

protected:

int Number\_of\_points;

string speciality;

};

**Файл “Abiturient.cpp”:**

#include "Abiturient.h"

Abiturient::Abiturient() : Person() {

Number\_of\_points = 0;

speciality = "";

}

Abiturient::Abiturient(string n, int a, int np, string s) : Person(n, a) {

Number\_of\_points = np;

speciality = s;

}

Abiturient::~Abiturient() {

}

Abiturient::Abiturient(const Abiturient& other) {

name = other.name;

age = other.age;

Number\_of\_points = other.Number\_of\_points;

speciality = other.speciality;

}

Abiturient& Abiturient::operator =(const Abiturient& other) {

if (&other == this)

return \*this;

name = other.name;

age = other.age;

Number\_of\_points = other.Number\_of\_points;

speciality = other.speciality;

return \*this;

}

void Abiturient::Show() {

cout << "Name: " << name;

cout << "\nAge: " << age;

cout << "\nNumber of points: " << Number\_of\_points;

cout << "\nSpeciality: " << speciality;

cout << "\n";

}

void Abiturient::Input() {

cout << "Name: ";

cin >> name;

cout << "Age: ";

cin >> age;

cout << "Number of points: ";

cin >> Number\_of\_points;

cout << "Speciality: ";

cin >> speciality;

cout << "\n";

}

**Файл “List.h”:**

#include "Abiturient.cpp"

struct Node {

Object\* data;

Node\* next;

Node\* prev;

Node (Object\* data, Node\* next = nullptr, Node\* prev = nullptr) {

this->data = data;

this->next = next;

this->prev = prev;

}

};

class List {

private:

Node\* head;

Node\* tail;

int size;

public:

List();

List(int);

~List();

bool isEmpty();

void clear();

void Pop\_back();

void Push\_back();

void Push\_back(Object\*);

void Show();

void Average\_age();

};

**Файл “List.cpp”:**

#include "List.h"

List::List() {

head = nullptr;

tail = nullptr;

size = 0;

}

List::~List() {

clear();

}

void List::clear() {

while (!isEmpty())

Pop\_back();

}

void List::Pop\_back() {

if (isEmpty())

throw runtime\_error("List is empty");

Object\* data = tail->data;

Node\* temp = tail;

tail = tail->prev;

delete temp;

if (tail == nullptr)

head = nullptr;

else

tail->next = nullptr;

size--;

}

List::List(int size) {

if (size < 0)

throw invalid\_argument("Size cannot be negative");

head = nullptr;

tail = nullptr;

this->size = 0;

for (int i = 0; i < size; i++)

Push\_back();

}

void List::Push\_back() {

Object \*p;

cout << "1.Person" << endl;

cout << "2.Abiturient" << endl;

int y;

cin >> y;

if (y == 1) {

Person\* a = new Person;

a->Input();

p = a;

Node\* newNode = new Node(p);

if (isEmpty()) {

head = newNode;

tail = newNode;

}

else {

newNode->prev = tail;

tail->next = newNode;

tail = newNode;

}

size++;

}

else {

if (y == 2) {

Abiturient\* b = new Abiturient;

b->Input();

p = b;

Node\* newNode = new Node(p);

if (isEmpty()) {

head = newNode;

tail = newNode;

}

else {

newNode->prev = tail;

tail->next = newNode;

tail = newNode;

}

size++;

}

else

return;

}

}

void List::Push\_back(Object\* p) {

Node\* newNode = new Node(p);

if (isEmpty()) {

head = newNode;

tail = newNode;

}

else {

newNode->prev = tail;

tail->next = newNode;

tail = newNode;

}

size++;

}

bool List::isEmpty() {

return size == 0;

}

void List::Show() {

Node\* current = head;

cout << "List:\n";

while (current != nullptr) {

Object\* p = current->data;

p->Show();

cout << endl;

current = current->next;

}

cout << "\n";

}

**Файл “Dialog.h”:**

#include "Event.h"

#include "list.cpp"

class Dialog : public List {

public:

Dialog();

virtual ~Dialog();

virtual void GetEvent(TEvent& event);

virtual int Execute();

virtual void HandleEvent(TEvent& event);

virtual void ClearEvent(TEvent& event);

int Valid();

void EndExec();

protected:

int EndState;

};

**Файл “Dialog.cpp”:**

#include "Dialog.h"

Dialog::Dialog() : List() {

EndState = 0;

}

Dialog::~Dialog() {

}

void Dialog::GetEvent(TEvent& event) {

string OpInt = "+-sqm";

string s;

string param;

char code;

cout << '>';

cin >> s;

code = s[0];

if (OpInt.find(code) >= 0) {

event.what = evMessage;

switch (code) {

case 'm':

event.command = cmMake;

break;

case '+':

event.command = cmAdd;

break;

case '-':

event.command = cmDel;

break;

case 's':

event.command = cmShow;

break;

case 'q':

event.command = cmQuit;

break;

}

if (s.length() > 1) {

param = s.substr(1, s.length()-1);

int A = atoi(param.c\_str());

event.a = A;

}

}

else

event.what = evNothing;

}

int Dialog::Execute() {

TEvent event;

do {

EndState = 0;

GetEvent(event);

HandleEvent(event);

}

while(!Valid());

return EndState;

}

int Dialog::Valid() {

if (EndState == 0)

return 0;

else

return 1;

}

void Dialog::ClearEvent(TEvent& event) {

event.what = evNothing;

}

void Dialog::EndExec() {

EndState = 1;

}

void Dialog::HandleEvent(TEvent& event) {

if (event.what == evMessage) {

switch (event.command)

{

case cmMake:

if (event.a < 0)

throw invalid\_argument("Size cannot be negative");

for (int i = 0; i < event.a; i++)

Push\_back();

ClearEvent(event);

break;

case cmAdd:

Push\_back();

ClearEvent(event);

break;

case cmDel:

Pop\_back();

ClearEvent(event);

break;

case cmShow:

Show();

ClearEvent(event);

break;

case cmQuit:

EndExec();

ClearEvent(event);

break;

}

}

}

**Файл “Event.h”:**

#pragma once

const int evNothing = 0;

const int evMessage = 100;

const int cmAdd = 1;

const int cmDel = 2;

const int cmShow = 3;

const int cmMake = 4;

const int cmQuit = 101;

struct TEvent {

int what;

union {

int command;

struct {

int message;

int a;

};

};

};

**UML диаграмма:**

**Скриншот результата выполнения программы**

**Ответы на вопросы:**

1. Что такое класс-група? Привести примеры таких классов.
2. Привести пример описания класса-группы Список (List).
3. Привести пример конструктора (с параметром, без параметров, копирования) для класса-группы Список.
4. Привести пример деструктора для класса-группы Список.
5. Привести пример метода для просмотра элементов для класса-группы Список.
6. Какой вид иерархии даёт группа?
7. Почему во главе иерархии классов, содержащихся в группе объектов должен находиться абстрактный класс?
8. Что такое событие? Для чего используются события?
9. Какие характеристики должно иметь событие-сообщение?
10. Привести пример структуры, описывающей событие.
11. Задана структура события

struct TEvent {

int what;

union {

MouseEventType mouse;

KeyDownEvent keyDown;

MessageEvent message;

}

};

Какие значения, и в каких случаях присваиваются полю what?

1. Задана структура события

struct TEvent {

int what;

union {

int command;

struct {

int message;

int a;

};

};

};

Какие значения, и в каких случаях присваиваются полю command?

1. Задана структура события

struct TEvent {

int what;

union {

int command;

struct {

int message;

int a;

};

};

};

Для чего используется поля a и message?

1. Какие методы необходимы для организации обработки сообщений?
2. Какой вид имеет главный цикл обработки событий-сообщений?
3. Какую функцию выполняет метод ClearEvent()? Каким образом?
4. Какую функцию выполняет метод HandleEvent()? Каким образом?
5. Какую функцию выполняет метод GetEvent()?
6. Для чего используется поле EndState? Какой класс (объект) содержит это поле?
7. Для чего используется функция Valid()?