Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования

Пермский национальный исследовательский политехнический университет

Факультет электротехнический

Кафедра ИТАС

ОТЧЁТ

о лабораторной работе №8 по классам

Выполнил:
Студент группы ИВТ-23-1Б
Пискунов Д. А.

_____

Проверил:
Доцент кафедры ИТАС
Яруллин Д.В.

_____

_____

Пермь 2024

Задача:

15	Базовый класс: ПЕЧАТНОЕ_ИЗДАНИЕ(PRINT) Название– string Автор – string Производный класс ЖУРНАЛ (MAGAZIN) Количество страниц - int Группа – Дерево (Tree). Команды: • 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 названиях всех элементов группы (формат команды : z) Конец работы (формат команды: q)

Текст программы
файл Lab_Main.ccp

```cpp
#include<iostream>
#include <string>
#include"Print.h"
#include"Object.h"
#include"Magazin.h"
#include"Dialog.h"

using namespace std;

int main() {
	system("chcp 1251 > NULL");

	cout << "m: Создать группу\n+: Добавить элемент\n-: Удалить элимент\ns: Информация о элементах\nz: Информация (Названия)\nq: Выход\n";
	Dialog D;
	D.Execute();

	return 0;
}
```

Файл Dialog.cpp

```cpp
#include "Dialog.h"
#include "Event.h"
#include "Tree.h"
#include <iostream>
using namespace std;

Dialog::Dialog() : Tree() {
        EndState = 0;
}

Dialog::~Dialog(void) {}

void Dialog::GetEvent(TEvent& Event) {
        string OpInt = "+m-szq";
        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;
                case 'z':Event.command = cmGet; break;
                }
        }
        else Event.what = evNothing;
}

void Dialog::Execute() {
        TEvent Event;
        do {
                EndState = 0;
                GetEvent(Event);
                HandleEvent(Event);
        } while (Valid());
}

bool Dialog::Valid() { return (EndState == 0); }

void Dialog::ClearEvent(TEvent& Event) {
        Event.what = evNothing;
```

```
46              Event.what = evNothing;
47      }
48      void Dialog::EndExec() {
49              EndState = 1;
50      }
51
52      void Dialog::HandleEvent(TEvent& Event) {
53              if (Event.what == evMessage) {
54                      switch (Event.command) {
55                      case cmMake:
56                              cout << "Ââåäèòå ðàçìåð: ";
57                              cin >> size;
58                              beg = new Object * [size];
59                              cur = 0;
60                              ClearEvent(Event);
61                              break;
62                      case cmAdd:
63                              Add();
64                              ClearEvent(Event);
65                              break;
66                      case cmDel:Del();
67                              ClearEvent(Event);
68                              break;
69                      case cmShow:Show();
70                              ClearEvent(Event);
71                              break;
72                      case cmQuit:EndExec();
73                              ClearEvent(Event);
74                              break;
75                      default:
76                              Tree::HandleEvent(Event);
77                              break;
78                      };
79              };
80      }
```

Файл Dialog.h

```
#pragma once
#include "Tree.h"
#include "Event.h"

class Dialog : public Tree {
public:
        Dialog();
        virtual ~Dialog(void);
        virtual void GetEvent(TEvent& Event);
        virtual void Execute();
        virtual void HandleEvent(TEvent& Event);
        virtual void ClearEvent(TEvent& Event);
        bool Valid();
        void EndExec();
protected:
        int EndState;
};
```

Файл Event.h

```
#pragma once
using namespace std;

const int evNothing = 0;
const int evMessage = 100;
const int cmAdd = 1;
const int cmDel = 2;
const int cmGet = 3;
const int cmShow = 4;
const int cmMake = 6;
const int cmQuit = 101;

struct TEvent {
        int what;
        int command;
};
```

Файл Magazin.cpp

```cpp
#include "Magazin.h"
Magazin::Magazin() {
	page_count = 0;
};
Magazin::Magazin(int p) {
	page_count = p;
}
Magazin::Magazin(const Magazin& m) {
	page_count = m.page_count;
}
Magazin::~Magazin(){}
void Magazin::Set_page_count(int p) {
	page_count = p;
}
int Magazin::Get_page_count() {
	return page_count;
}
Magazin& Magazin::operator=(const Magazin& m) {
	if (&m == this) {
		return *this;
	}
	page_count = m.page_count;
	return *this;
}
void Magazin::Show() {
	cout << "\nName: " << name;
	cout << "\nAuthor: " << author;
	cout << "\nPage_count: " << page_count << endl;
}
void Magazin::Input() {
	cout << "\nName: "; cin >> name;
	cout << "\nAuthor: "; cin >> author;
	cout << "\nPage_count: "; cin >> page_count;
}
```

Файл Magazin.h

```h
#pragma once
#include "Print.h"
class Magazin:
    public Print{
protected:
    int page_count;
public:
    Magazin();
    Magazin(int);
    Magazin(const Magazin&);
    ~Magazin();
    void Set_page_count(int);
    int Get_page_count();
    Magazin& operator=(const Magazin&);
    void Show();
    void Input();
};
```

Файл Object.h

```cpp
#pragma once
#include"Event.h"
class Object{
public:
        Object() {}
        virtual void Show() = 0;
        virtual void Input() = 0;
        virtual void HandleEvent(const TEvent&) = 0;
};
```

Файл Print.cpp

```cpp
#include "Print.h"
Print::Print() {
        name = "";
        author = "";
}
Print::Print(string n, string a) {
        name = n;
        author = a;
}
Print::Print(const Print& p) {
        name = p.name;
        author = p.author;
}
Print::~Print() {}
void Print::Set_name(string n) {
        name = n;
}
void Print::Set_author(string a) {
        author = a;
}
string Print::Get_name(){
        return name;
}
string Print::Get_author() {
        return author;
}
```

```cpp
        return author;
}
Print& Print::operator=(const Print& p) {
        if (&p == this) {
                return *this;
        }
        name = p.name;
        author = p.author;
        return *this;
}
void Print::Show() {
        cout << "\nName: "<<name;
        cout << "\nAuthor: " << author << endl;
}
void Print::Input() {
        cout << "\nName: "; cin >> name;
        cout << "\nAuthor: "; cin >> author;
}
void Print::HandleEvent(const TEvent& e) {
        if (e.what == evMessage) {
                switch (e.command) {
                case cmGet:
                        cout << "Íàçâàíèå: " << Get_name() << '\n';
                        break;
                }
        }
}
```

Файл Print.h

```
#pragma once
#include<iostream>
#include "Object.h"


using namespace std;

class Print:
        public Object{
protected:
        string name;
        string author;
public:
        Print();
        Print(string, string);
        Print(const Print&);
        ~Print();
        void Set_name(string);
        void Set_author(string);
        string Get_name();
        string Get_author();
        Print& operator=(const Print&);
        void Show();
        void Input();
        void HandleEvent(const TEvent& e);
};
```

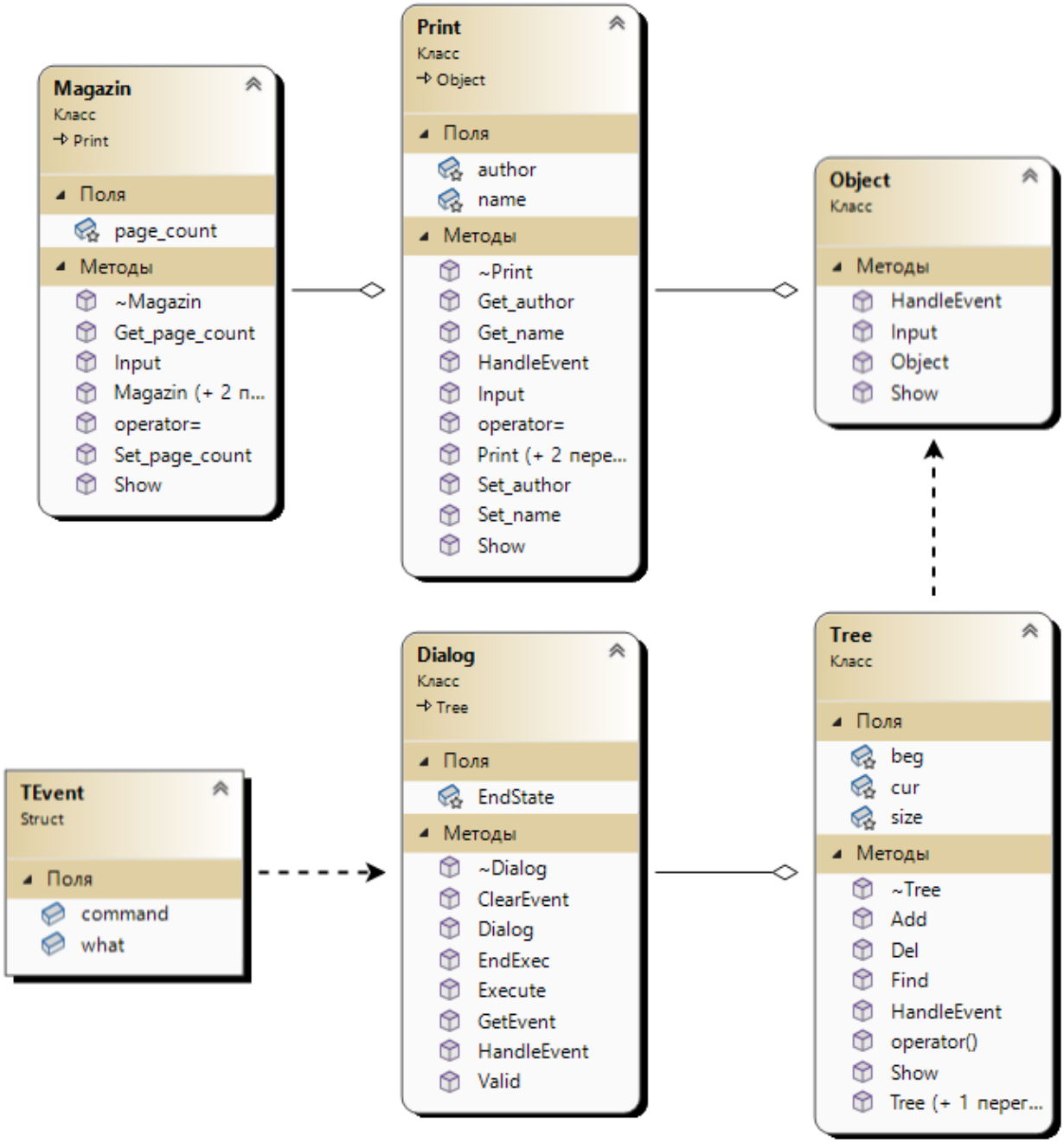
Файл Tree.cpp

```cpp
#include "Tree.h"
#include<iostream>
#include"Dialog.h"

Tree::Tree() {
	beg = nullptr;
	size = 0;
	cur = 0;
}
Tree::Tree(int n) {
	beg = new Object * [n];
	cur = 0;
	size = n;
}
Tree::~Tree(void) {
	if (beg != 0) { delete[] beg; }
	beg = 0;
}
void Tree::Add() {
	Object* p;

	cout << "\n1.Print\n";
	cout << "2.Magazin\n";
	int choise;
	cin >> choise; cout << "\n";

	if (choise == 1) {
		Print* a = new (Print);
		a->Input();
		p = a;
		if (cur < size) {
			beg[cur] = p;
			cur++;
		}
	}
	else if (choise == 2) {
		Magazin* b = new (Magazin);
		b->Input();
		p = b;
		if (cur < size) {
			beg[cur] = p;
			cur++;
		}
	}
	else return;
}
void Tree::Show()
{
	if (cur == 0) {
		cout << "\nEmpty" << endl;
	}
	Object** p = beg;
	for (int i = 0; i < cur; i++) {
		(*p)->Show();
		p++;
	}
}
int Tree::operator()() {
	return cur;
}
void Tree::Del(void) {
	if (cur == 0) {
		return;
	}
	cur--;
}
void Tree::Find(int tmp) {
	Object** p = beg;
	for (int i = 0; i < cur; i++) {
		if (i == tmp - 1) {
			(*p)->Show();
		}
		p++;
	}
}
void Tree::HandleEvent(const TEvent& event) {
	if (event.what == evMessage) {
		Object** p = beg;
		for (int i = 0; i < cur; i++) {
			(*p)->Show();
			++p;
		}
	}
}
```

Файл Tree.h

```h
#pragma once
#include<iostream>
#include"Object.h"
#include"Print.h"
#include"Magazin.h"

class Tree {
protected:
        Object** beg;
        int size;
        int cur;
public:
        Tree();
        Tree(int);
        ~Tree();
        void Add();
        void Del();
        void Show();
        void Find(int);
        int operator()();
        void HandleEvent(const TEvent& event);
};
```

UML-диаграмма



Тест

```
m: Создать группу
+: Добавить элемент
-: Удалить элимент
s: Информация о элементах
z: Информация (Названия)
q: Выход
>m
Введите размер: 10
>+

1.Print
2.Magazin
1


Name: gggg

Author: wwef
>-
>s

Empty
>z
>q

C:\Users\MOkASiH\Desktop\Test\x64\Debug\Test.exe (процесс 19644) завершил работу с кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно…
```

Ответы на вопросы

1. Что такое класс-группа? Привести примеры таких классов.

Ответ: класс-группа — объект, в который включены другие объекты.

2. Привести пример описания класса-группы Список (List).

```
class List
{
public:
Iterator beg, *end;
Node *head, *tail;
...
};
```

3. Привести пример конструктора (с параметром, без параметров, копирования) для класса-группы Список.

```
List::List() : head(nullptr), tail(nullptr), beg(nullptr), end(nullptr) {}
List::List(Node head, Node *tail) : head(head), tail(tail), beg(head), end(nullptr) {}
List::List(const List& other) : List(other.head, other.tail) {}
```

4. Привести пример деструктора для класса-группы Список.

```
~List() {
    while (head != NULL)
        pop_front();
}
```

5. Привести пример метода для просмотра элементов для класса-группы Список.

```
void getElements(List list)
{
    for (List::Iterator it = list.begin(); it != list.end(); it++)
        cout << *it;
}
```

6. Какой вид иерархии дает группа?

Иерархию объектов

7. Почему во главе иерархии классов, содержащихся в группе объектов должен находится абстрактный класс?

Для удобного расширения кода

8. Что такое событие? Для чего используются события?

Событие — информация, полученная в ходе действий пользователя. События используют для взаимодействия пользователя с интерфейсом программы.

9. Какие характеристики должно иметь событие-сообщение?

Событие-сообщение должно иметь код сообщения.

10. Привести пример структуры, описывающей событие.

```
struct tEvent
{
    EventType type; // EventType – enum
    T data;
};
```

11. Задана структура события:

```
struct TEvent
{
	int what;
	union
	{
		MouseEventType mouse;
		KeyDownEvent keyDown;
		MessageEvent message;
	}
};
```

Какие значения, и в каких случаях присваиваются полю *what*?

Полю what присваиваются значения пришло ли событие.

12. В: Задана структура события:

```
struct TEvent
{
		int what;
		union
	{
	int command;
	struct
	{
			int message;
			int a;
	};
}
};
```

Какие значения, и в каких случаях присваиваются полю *command*?

Полю command присваиваются значения типа события в случае требуемого what

13. Задана структура события:

```
struct TEvent
{
        int what;
        union
        {
                int command;
                struct
                {
                        int message;
                        int a;
                };
        }
};
```

Для чего используются поля a и message?

message – данные

14. Какие методы необходимы для организации обработки сообщений?

Метод получения события и метод обработчик сообщений

15. Какой вид имеет главный цикл обработки событий-сообщений?

```
do {
    EndState = 0;
    GetEvent(Event);
    HandleEvent(Event);
} while (Valid());
```

16. Какую функцию выполняет метод ClearEvent()? Каким образом?

Устанавливает значения по умолчанию для данного экземпляра

17. Какую функцию выполняет метод HandleEvent()? Каким образом?

Обрабатывает событие

18. Какую функцию выполняет метод GetEvent ()?

Получает новое событие

19. Для чего используется поле EndState? Какой класс (объект) содержит это поле?

EndState – это поле для отображения завершения обработки

20. Для чего используется функция Valid()?

Проверка значения EndState