Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет»

ПНИПУ

**Лабораторная работа по ООП**

**«№»**

Выполнил:

студент группы РИС-23-2б

Ившин Максим Сергеевич

Проверила:

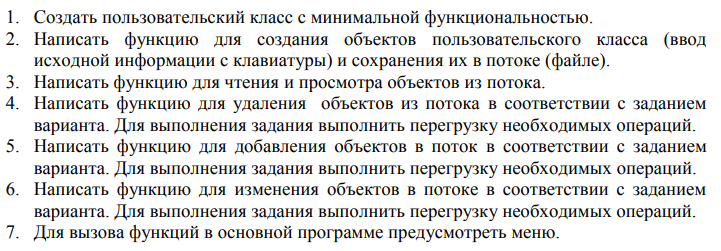
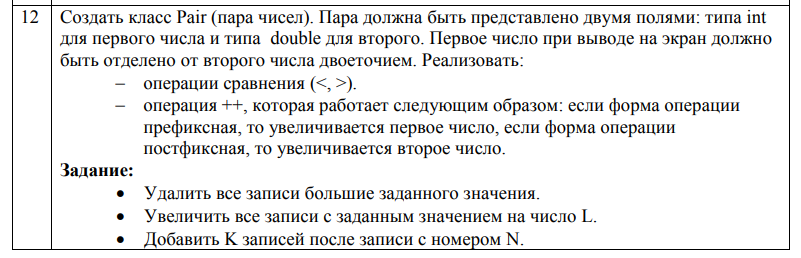
доцент кафедры ИТАС

О.А. Полякова

2024 г.

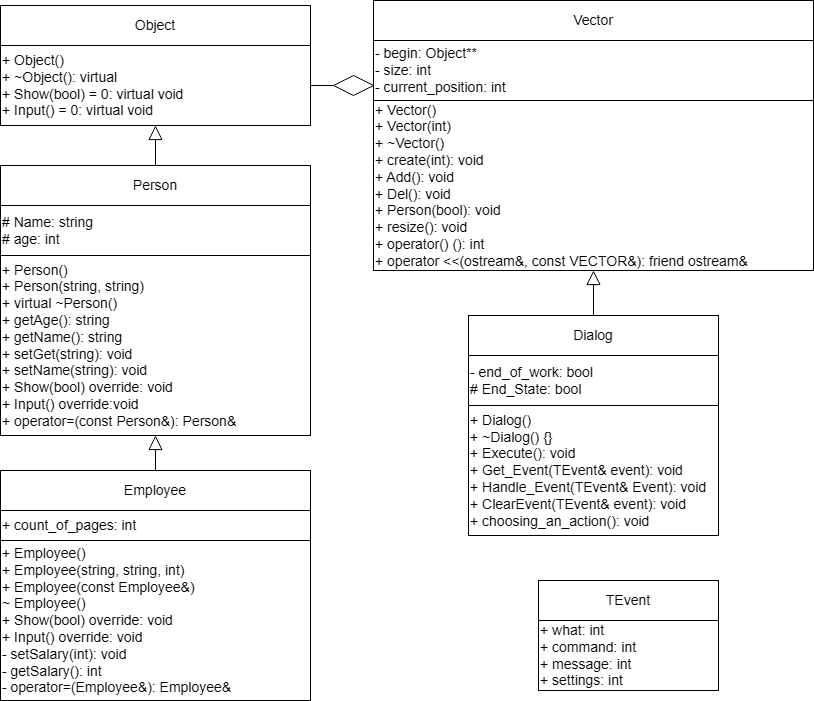
**Разработка алгоритма**

**Постановка задачи:**

****

**Анализ задачи:**

1. Необходимо определить иерархию пользовательских классов, где во главе стоит абстрактный класс Object с чисто виртуальными методами для ввода и вывода информации об атрибутах объектов.
2. Для класса Person, наследуемого от Object, необходимо реализовать конструкторы, деструктор, операцию присваивания, селекторы и модификаторы.
3. Определить класс Employee на основе структуры Person.
4. Для Employee и Person реализовать конструкторы, деструктор, методы для добавления и удаления элементов в группу, метод для просмотра группы, перегрузить операцию для получения информации о размере группы.
5. Определить класс Dialog – наследника группы, в котором реализовать методы для обработки событий.
6. Добавить методы для обработки событий группой и объектами пользовательских классов.



Код на C++:

#include <iostream>

using namespace std;

#include "Includes.cpp"

int main()

{

    // system("chcp 1251>null");

    Dialog dialog;

    dialog.Execute();

    return 0;

}

Classes.h:

#pragma once

#include <iostream>

#include <string>

#include "Event.h"

using namespace std;

class Object

{

public:

    Object() {}

    virtual ~Object() {}

    virtual void Show(bool) = 0;

    virtual void Input() = 0;

};

class Person : public Object

{

protected:

    string Name;

    int age;

public:

    Person();

    Person(string, int);

    Person(const Person &);

    virtual ~Person() {}

    void Show(bool) override;

    void Input() override;

    Person &operator=(const Person &);

    void setAge(int &age) { this->age = age; }

    void setName(string &name) { this->Name = name; }

    int getAge() { return this->age; }

    string getName() { return this->Name; }

};

class Employee : public Person

{

protected:

    float salary;

    string post;

public:

    Employee();

    Employee(string, int, float, string);

    Employee(const Employee &);

    ~Employee() {}

    void Show(bool) override;

    void Input() override;

    Employee &operator=(const Employee &);

    void setPages(float &salary) { this->salary = salary; }

    float getPages() { return this->salary; }

};

Dialog.h:

#pragma once

#include "Vector.h"

#include "Event.h"

#include "Classes.h"

class Dialog : public Vector

{

protected:

    int EndState;

public:

    Dialog();

    ~Dialog() {}

    void GetEvent(TEvent &);

    bool is\_command(char);

    int Execute();

    void HandleEvent(TEvent &);

    void ClearEvent(TEvent &);

    int Valid();

    void EndExec();

};

Dialog.cpp:

#pragma once

#include "Dialog.h"

Dialog::Dialog() : Vector()

{

    this->EndState = 0;

}

void Dialog::GetEvent(TEvent &event)

{

    cout << "Выберите действие\n"

         << "m: Создать группу (Пример команды: m10)\n"

         << "+: Добавить элемент в группу\n"

         << "-: Удалить элемент из группы\n"

         << "s: Вывести информацию об элементах группы\n"

         << "z: Вывести информацию о названиях всех элементов группы\n"

         << "q: Конец работы\n";

    cout << "-> ";

    string temp\_command, temp\_params;

    cin >> temp\_command;

    char code = temp\_command[0];

    if (is\_command(code))

    {

        event.what = EvMessage;

        switch (code)

        {

        case 'm':

            event.command = CmCreateGroup;

            break;

        case '+':

            event.command = CmAddToGroup;

            break;

        case '-':

            event.command = CmRemoveFromGroup;

            break;

        case 's':

            event.command = CmShow;

            break;

        case 'z':

            event.command = CmShowName;

            break;

        case 'q':

            event.command = CmExit;

            break;

        }

        if (temp\_command.size() > 1)

        {

            temp\_params = temp\_command.substr(1, temp\_command.size() - 1);

            int settings = atoi(temp\_params.c\_str());

            event.settings = settings;

        }

    }

    else

    {

        event.what = EvNothing;

    }

}

bool Dialog::is\_command(char action)

{

    int i = 0;

    string temp\_command = "m+-szq";

    while (i < int(temp\_command.size()))

    {

        if (action == temp\_command[i])

        {

            return true;

        }

        i++;

    }

    return false;

}

int Dialog::Execute()

{

    TEvent event;

    do

    {

        EndState = 0;

        GetEvent(event);

        cout << "\n";

        HandleEvent(event);

    } while (!Valid());

    return EndState;

}

void Dialog::HandleEvent(TEvent &event)

{

    if (event.what == EvNothing)

    {

        cout << "Такой команды нет.\n";

        return;

    }

    switch (event.command)

    {

    case CmCreateGroup:

        create(event.settings);

        ClearEvent(event);

        break;

    case CmAddToGroup:

        Add();

        ClearEvent(event);

        break;

    case CmRemoveFromGroup:

        Del();

        ClearEvent(event);

        break;

    case CmShow:

        print(false);

        ClearEvent(event);

        break;

    case CmShowName:

        print(true);

        ClearEvent(event);

        break;

    case CmExit:

        EndExec();

        ClearEvent(event);

        break;

    default:

        break;

    }

}

void Dialog::ClearEvent(TEvent &event)

{

    event.what = EvNothing;

}

int Dialog::Valid()

{

    if (EndState == 0)

    {

        return 0;

    }

    return 1;

}

void Dialog::EndExec()

{

    EndState = 1;

}

Event.h:

#pragma once

const int EvNothing = 0; // ничего не делать

const int EvMessage = 100; // получена комманда

const int CmCreateGroup = 1; //m создать группу

const int CmAddToGroup = 2; //+ добавить в группу элемент

const int CmRemoveFromGroup = 3; //- удалить элемент из группы

const int CmShow = 4; //s Вывести информацию об элементах группы

const int CmShowName = 5; //z Вывести информацию о названиях всех элементов группы

const int CmExit = 6; //q Конец работы

struct TEvent

{

    int what;

    union

    {

        int command;

        struct

        {

            int message;

            int settings;

        };

    };

};

Employee.cpp:

#pragma once

#include "Classes.h"

Employee::Employee() : Person()

{

    this->salary = 0;

    this->post = "";

}

Employee::Employee(string name, int age, float salary, string post) : Person(name, age)

{

    this->age = age;

    this->post = post;

}

Employee::Employee(const Employee &other\_employee)

{

    this->age = other\_employee.age;

    this->Name = other\_employee.Name;

    this->salary = other\_employee.salary;

}

void Employee::Show(bool names = false)

{

    if (!names)

    {

        cout << "\nИмя: " << this->Name << " Возраст: " << this->age

             << "\nЗарплата: " << this->salary << " Должность: " << this->post << "\n\n";

    }

    else

    {

        cout << "\nИмя: " << this->Name << "\n\n";

    }

}

void Employee::Input()

{

    cout << "\nВведите данные работника:\n";

    cout << "Название: ";

    cin.seekg(cin.eof());

    getline(cin, this->Name);

    cout << "Возраст: ";

    cin >> this->age;

    cout << "Зарплата: ";

    cin >> this->salary;

    cout << "Должность: ";

    cin.seekg(cin.eof());

    getline(cin, this->post);

}

Employee &Employee::operator=(const Employee &other\_employee)

{

    if (this == &other\_employee)

    {

        return \*this;

    }

    Person::operator=(other\_employee);

    this->salary = other\_employee.salary;

    this->post = other\_employee.post;

    return \*this;

}

**GitHub:** *https://github.com/geroineee/PNRPU*