Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

направление подготовки: 09.03.01 - «Информатика и вычислительная техника»

**О Т Ч Е Т**

**по лабораторной работе “Классы” №5**

**по дисциплине**

**«Основы алгоритмизации и программирования» семестр 2**

Выполнил студент гр. ИВТ-21-1б

Ипатов Дмитрий Сергеевич

Проверил:

Ст. Преподаватель кафедры ИТАС

Яруллин Д.В.

(оценка) (подпись)

г. Пермь-2022

**Постановка задачи:**

* + 1. Определить абстрактный класс.
    2. Определить иерархию классов, в основе которой будет находиться абстрактный класс (см. лабораторную работу №4).
    3. Определить класс Вектор, элементами которого будут указатели на объекты иерархии классов.
    4. Перегрузить для класса Вектор операцию вывода объектов с помощью потоков.
    5. В основной функции продемонстрировать перегруженные операции и полиморфизм Вектора.

**Анализ задачи:**

Класс, содержащий хотя бы один чисто виртуальный метод, называется *абстрактным.*

Чисто виртуальный метод содержит признак = 0 вместо тела, например:

virtual void f(int) = 0;

Чисто виртуальный метод должен переопределяться в производном классе (возможно, опять как чисто виртуальный).

Абстрактные классы предназначены для представления общих понятий, которые предполагается конкретизировать в производных классах. Абстрактный класс может использоваться *только в качестве базового* для других классов — объекты абстрактного класса создавать нельзя, поскольку прямой или косвенный вызов чисто виртуального метода приводит к ошибке при выполнении.

**Код программы:**

#include <iostream>

#include <string>

#include <Windows.h>

using namespace std;

class creature

{

public:

creature(void);

~creature(void);

virtual void Show() = 0;

};

class Person :public creature

{

protected:

string name;

int age;

public:

Person();

Person(string, int);

Person(const Person&);

~Person() {}

string GetName() { return name; }

int GetAge() { return age; }

void SetName(string name) { this->name = name; }

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

virtual Person& operator= (const Person&);

void Show();

friend istream& operator>>(istream& in, Person& p);

friend ostream& operator<<(ostream& out, const Person& p);

};

Person::Person()

{

this->name = "-";

this->age = 0;

}

Person::Person(string newName, int newAge)

{

this->name = newName;

this->age = newAge;

}

Person::Person(const Person& t)

{

this->name = t.name;

this->age = t.age;

}

Person& Person::operator=(const Person& t)

{

if (&t == this) return \*this;

name = t.name;

age = t.age;

return \*this;

}

istream& operator>>(istream& in, Person& p)

{

cout << "Имя: "; in >> p.name;

cout << "Возраст: "; in >> p.age;

return in;

}

ostream& operator<<(ostream& out, const Person& p) { return (out << "Имя: " << p.name << "\nВозраст: " << p.age << "\n\n"); }

void Person::Show()

{

cout << "Имя: " << name << "\nВозраст: " << age << "\n\n";

}

class Employee : public Person

{

protected:

string post;

double salary;

public:

Employee(void);

Employee(string, double, string, int);

Employee(const Employee&);

~Employee();

void SetSalary(double salary) { this->salary = salary; }

void SetPost(string post) { this->post = post; }

double GetSalary() { return salary; }

string GetPost() { return post; }

virtual Employee& operator=(const Employee&);

void Show();

double cntsalary() { return salary \* 1.1; }

friend istream& operator>>(istream& in, Employee& t);

friend ostream& operator<<(ostream& out, const Employee& t);

};

Employee::Employee()

{

this->name = "-";

this->age = 0;

this->post = "-";

this->salary = 0;

}

Employee::Employee(string post, double salary, string name, int age)

{

this->post = post;

this->salary = salary;

this->name = name;

this->age = age;

}

Employee::Employee(const Employee& t)

{

this->post = t.post;

this->salary = t.salary;

this->name = t.name;

this->age = t.age;

}

Employee::~Employee() {}

istream& operator>>(istream& in, Employee& t)

{

cout << "Пост: "; in >> t.post;

cout << "Оклад: "; in >> t.salary;

return in;

}

ostream& operator<<(ostream& out, const Employee& t) { return (out << "Имя: " << t.name << "\nВозраст: " << t.age << "\nПост: " << t.post << "\nОклад: " << t.salary << "\n\n"); }

void Employee::Show()

{

cout << "Имя: " << name << "\nВозраст: " << age << "\nПост: " << post << "\nОклад: " << salary << "\n\n";

}

Employee& Employee::operator=(const Employee& t)

{

if (&t == this) return \*this;

post = t.post;

salary = t.salary;

return \*this;

}

void f1(Person& c)

{

c.SetName("Джон");

cout << c << '\n';

}

Person f2()

{

Employee l("Слесарь", 25000, "Жора Букер", 34);

return l;

}

class Vector

{

private:

creature\*\* beg;

int size;

int cur;

public:

Vector(void);

Vector(int);

~Vector();

void Add(creature\*);

friend ostream& operator<<(ostream& out, const Vector&);

};

Vector::Vector(void)

{

beg = 0;

size = 0;

cur = 0;

}

Vector::Vector(int n)

{

beg = new creature \* [n];

cur = 0;

size = n;

}

void Vector::Add(creature\* p)

{

if (cur < size)

{

beg[cur] = p;

cur++;

}

}

ostream& operator<<(ostream& out, const Vector& v)

{

if (v.size == 0) out << "Empty" << endl;

creature\*\* p = v.beg;

for (int i = 0; i < v.cur; i++)

{

(\*p)->Show();

p++;

}

return out;

}

Vector::~Vector(void)

{

if (beg != 0) delete[] beg;

beg = 0;

}

int main()

{

setlocale(LC\_ALL, "rus");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Vector v(5);

Person chel1;

cin >> chel1;

Employee chel2("Актриса", 100000, "Анна Тэйлор Джой", 25);

creature\* p = &chel1;

v.Add(p);

p = &chel2;

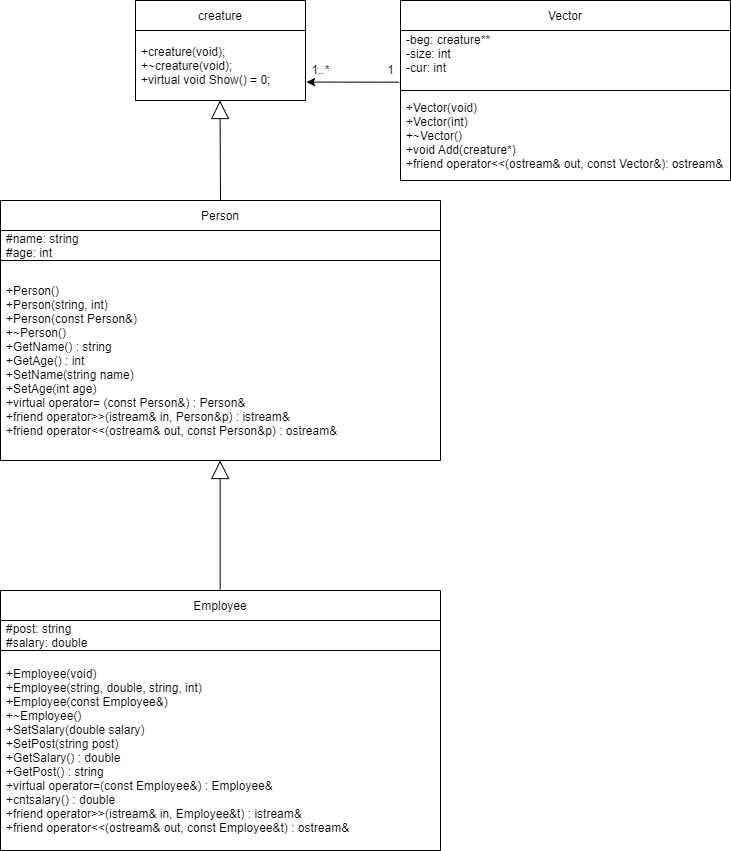
v.Add(p);

cout << v;

return 0;

}

**UML:**

****

**Контрольные вопросы:**

* 1. Какой метод называется чисто виртуальным? Чем он отличается от виртуального метода?

Чисто виртуальный методсодержит признак = 0 вместо тела, например: virtual void f(int) = 0; А виртуальный метод в отличие от чисто виртуального еще и описывается.

* 1. Какой класс называется абстрактным?

Класс, который имеет хотя бы один чисто виртуальный метод.

* 1. Для чего предназначены абстрактные классы?

Абстрактные классы предназначены для представления общих понятий, которые предполагается конкретизировать в производных классах.

* 1. Что такое полиморфные функции?

Функции, которые способны работать с объектом любого типа в пределах одной иерархии

* 1. Чем полиморфизм отличается от принципа подстановки?

Принцип подстановки отвечает за спецификацию классов, а полиморфизм за их поведение.

* 1. Привести примеры иерархий с использованием абстрактных классов.

Есть абстрактный класс оружие и у него есть чистая виртуальная функция “use”, которая отвечает за его использование, также существуют классы нож, пистолет, взрывчатка и все они являются наследниками абстрактного класса оружие, так как в самом классе оружие нельзя описать действие для каждого наследника (точнее это займет очень много кода), мы описываем метод use в наследнике с ключевым словом override, т.е. задаем новое поведение для метода use в зависимости от класса и логики взаимодействия с объектом ( как было написано пистолет стреляет, нож режет, взрывчатка взрывается и у неё может быть таймер).