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

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

**ОТЧЕТ**

**Тема:** Лабораторная работа №5 по ООП

Семестр: 2

Выполнил студент ИВТ-22-2б:

Мельников Глеб Владимирович

(дата, подпись)

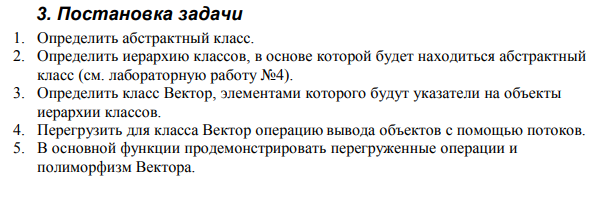
Проверила:

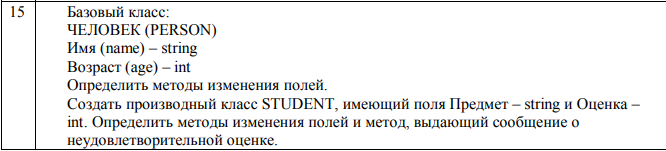
Полякова Ольга Андреевна

(дата, подпись)

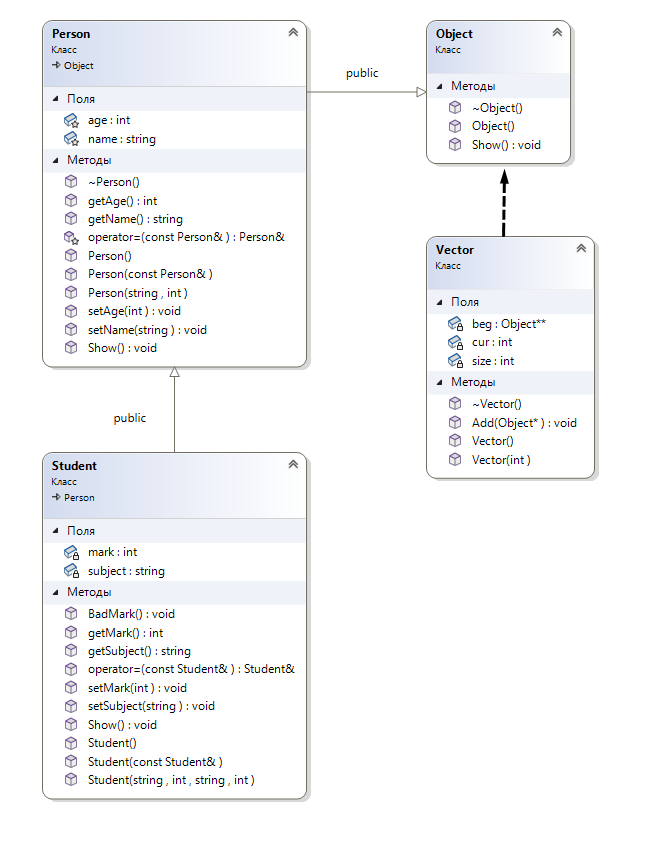
Пермь 2023

**Задание**





**UML-Диаграмма**



**Программное решение**

**Object.h**

#pragma once

#include <iostream>

#include <string>

using namespace std;

class Object

{

public:

Object() {};

~Object() {};

virtual void Show() = 0;

};

**Vector.h**

#pragma once

#include "Object.h"

class Vector {

public:

Vector();

Vector(int);

~Vector();

void Add(Object\*);

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

private:

Object\*\* beg;

int size;

int cur;

};

**Vector.cpp**

#include "Vector.h"

Vector::Vector() {

beg = 0;

size = 0;

cur = 0;

}

Vector::~Vector() {

if (beg != 0) {

delete[] beg;

}

beg = 0;

}

Vector::Vector(int n) {

beg = new Object \* [n];

cur = 0;

size = n;

}

void Vector::Add(Object\* p) {

if (cur < size) {

beg[cur] = p;

cur++;

}

}

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

if (v.size == 0) {

out << "Вектор пуст!\n";

}

Object\*\* p = v.beg;

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

(\*p)->Show();

p++;

}

return out;

}

**Student.h**

#pragma once

#include <iostream>

#include "Person.h"

using namespace std;

class Student : public Person {

private:

string subject;

int mark;

public:

void Show();

Student();

Student(string, int, string, int);

Student(const Student&);

void setSubject(string);

void setMark(int);

string getSubject();

int getMark();

void BadMark();

friend istream& operator>>(istream&, Student&);

friend ostream& operator<<(ostream&, const Student&);

Student& operator=(const Student&);

};

**Student.cpp**

#include "Student.h"

void Student::Show() {

cout << name << ' ' << age << ' ' << subject << ' ' << mark << '\n';

}

Student::Student() {

name = "";

subject = "";

age = 0;

mark = 0;

}

Student::Student(string name, int age, string subject, int mark) {

this->name = name;

this->age = age;

this->subject = subject;

this->mark = mark;

}

Student::Student(const Student& copied) {

name = copied.name;

age = copied.age;

subject = copied.subject;

mark = copied.mark;

}

void Student::setSubject(string subject) {

this->subject = subject;

}

void Student::setMark(int mark) {

this->mark = mark;

}

string Student::getSubject() {

return subject;

}

int Student::getMark() {

return mark;

}

void Student::BadMark() {

if (mark < 4) {

cout << this->name << " имеет плохую оценку\n";

}

}

istream& operator>>(istream& in, Student& cur) {

cout << "Введите данные в формате:\nимя возвраст предмет оценка\n";

return (in >> cur.name >> cur.age >> cur.subject >> cur.mark);

}

ostream& operator<<(ostream& out, const Student& cur) {

return (out << cur.name << ' ' << cur.age << ' ' << cur.subject << ' ' << cur.mark << '\n');

}

Student& Student::operator=(const Student& m\_student) {//перегрузка оператора присваивания

if (&m\_student != this) {

name = m\_student.name;

age = m\_student.age;

subject = m\_student.subject;

mark = m\_student.mark;

}

return \*this;

}

**Person.h**

#pragma once

#include <iostream>

#include <string>

#include "Object.h"

using namespace std;

class Person : public Object {

protected:

string name;

int age;

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

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

Person& operator=(const Person&);

public:

void Show();

void setName(string);

void setAge(int);

string getName();

int getAge();

Person();

Person(string, int);

Person(const Person&);

~Person() {};

};

**Person.cpp**

#include "Person.h"

Person::Person() {

this->name = "";

this->age = 0;

}

Person::Person(string name, int age) {

this->name = name;

this->age = age;

}

Person::Person(const Person& copied) {

age = copied.age;

name = copied.name;

}

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

cout << "Введите данные в формате:\nимя возвраст\n";

return (in >> p.name >> p.age);

}

ostream& operator<<(ostream& out, const Person& p) {

return (out << p.name << ' ' << p.age << '\n');

}

void Person::Show() {

cout << name << ' ' << age << '\n';

}

void Person::setName(string name) {

this->name = name;

}

void Person::setAge(int age) {

this->age = age;

}

string Person::getName() {

return name;

}

int Person::getAge() {

return age;

}

Person& Person::operator=(const Person& m\_person) {//перегрузка оператора присваивания

if (&m\_person != this) {

name = m\_person.name;

age = m\_person.age;

}

return \*this;

}

**OOPLab5.cpp**

#include <iostream>

#include "Object.h"

#include "Person.h"

#include "Student.h"

#include "Vector.h"

using namespace std;

int main() {

system("chcp 1251 > NULL");

Vector v(5);

Person p;

cin >> p;

Student s;

cin >> s;

s.BadMark();

Object\* o = &p;

v.Add(o);

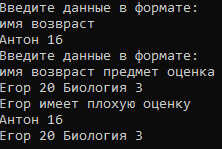
o = &s;

v.Add(o);

cout << v;

return 0;

}



**Контрольные вопросы**

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

чисто виртуальный метод — метод, не имеющий определения в базовом классе. Отличается от виртуального отсутствием тела. (Вместо тела пишется = 0)

1. Какой класс называется абстрактным?

Класс, в котором объявлена хотя бы одна чисто виртуальная функция.

1. Для чего предназначены абстрактные классы?

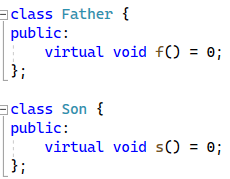
Для создания конкретных производных классов.

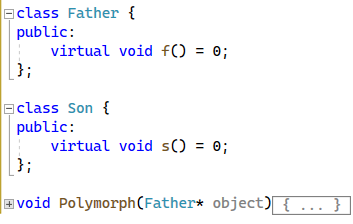
1. Что такое полиморфные функции?

Функции, которые могут принимать в качестве параметров как данные класса родителя, так и данные классов потомков.

1. Чем полиморфизм отличается от принципа подстановки?

Принцип подстановки — принцип полиморфизма.

1. Привести примеры иерархий с использованием абстрактных классов. 
2. Привести примеры полиморфных функций.



1. В каких случаях используется механизм позднего связывания?

Когда компилятор сам должен выбрать какую реализацию функции стоит вызвать при работе программы, т. е. динамически.