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 №5 по классам

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

_____

Проверил:
Доцент кафедры ИТАС
Яруллин Д.В.

_____

_____

Пермь 2024

Задача:

15	Базовый класс: ЧЕЛОВЕК (PERSON) Имя (name) – string Возраст (age) – int Определить методы изменения полей. Создать производный класс STUDENT, имеющий поля Предмет – string и Оценка – int. Определить методы изменения полей и метод, выдающий сообщение о неудовлетворительной оценке.

Текст программы
файл Lab_Main.cpp

```cpp
#include <string>
#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 a;
    cin >> a;
    Student b;
    cin >> b;
    Object* p = &a;
    v.Add(p);
    p = &b;
    v.Add(p);
    cout << v;

    return 0;
}
```

Файл Object.h

```cpp
#pragma once
#include <iostream>
#include <string>
using namespace std;

class Object {
public:
        Object() {};
        ~Object() {};
        virtual void Show() = 0;
};
```

Файл Person.cpp

```cpp
#include "Person.h"

Person::Person(void) {
	name = "";
	age = 0;
}
Person::Person(string n, int a) {
	name = n;
	age = a;
}
Person::Person(const Person& p) {
	name = p.name;
	age = p.age;
}
Person::~Person(){}
void Person::set_name(string n) {
	name = n;
}
void Person::set_age(int a) {
	age = a;
}

Person& Person::operator=(const Person& p) {
	if (this == &p) {
		return *this;
	}
	name = p.name;
	age = p.age;
	return *this;
}

istream& operator >>(istream& in, Person& p) {
	cout << "name: "; in >> p.name;
	cout << "age: "; in >> p.age;
	return in;
}
ostream& operator <<(ostream& out, const Person& p) {
	out << "\nname: " << p.name;
	out << "\nage: " << p.age << " y.o.";
	return out;
}

void Person::Show() {
	cout << "\nname: " << name;
	cout << "\nage: " << age << " y.o.\n";
}
```

Файл Person.h

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

using namespace std;

class Person: public Object {
protected:
        string name;
        int age;
public:
        void Show();
        Person();
        Person(string, int);
        Person(const Person&);
        virtual ~Person();
        string get_name() { return name; };
        int get_age() { return age; }

        void set_name(string);
        void set_age(int);

        Person& operator=(const Person&);

        friend istream& operator >>(istream& in, Person& p);
        friend ostream& operator <<(ostream& out, const Person& p);
};
```

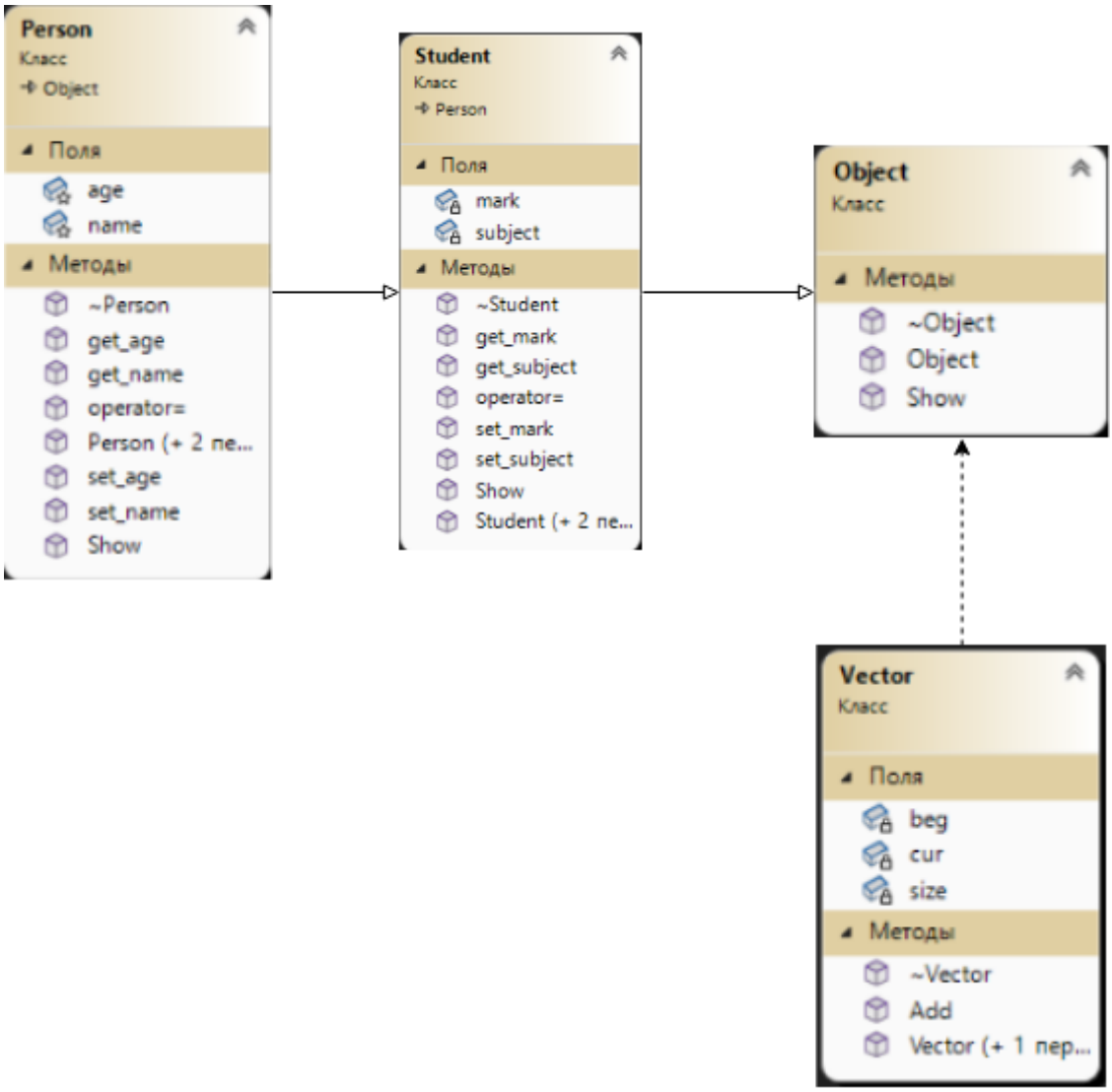
Файл Student.cpp

```cpp
#include"Student.h"

Student::Student():Person(){
	subject = "";
	mark = 0;
}
Student::Student(string n, int a, string s, int m):Person(n,a) {
	subject = s;
	mark = m;
}
Student::Student(const Student& s) {
	name = s.name;
	age = s.age;
	subject = s.subject;
	mark = s.mark;
}

Student& Student::operator=(const Student& s) {
	if (&s == this) {
		return* this;
	}
	name = s.name;
	age = s.age;
	subject = s.subject;
	mark = s.mark;
	return *this;
}

istream& operator >>(istream& in, Student& p) {
	cout << "name: "; in >> p.name;
	cout << "age: "; in >> p.age;
	cout << "subject: "; in >> p.subject;
	cout << "mark: "; in >> p.mark;
	return in;
}
ostream& operator <<(ostream& out, const Student& p) {
	out << "\nname: " << p.name;
	out << "\nage: " << p.age <<" y.o.";
	out << "\nsubject: " << p.subject;
	out << "\nmark: " << p.mark;
	return out;
}
void Student::Show() {
	cout << "\nname: " << name;
	cout << "\nage: " << age << " y.o.";
	cout << "\nsubject: " << subject;
	cout << "\nmark: " << mark << endl;
}
```

Файл Student.h

```
#pragma once
#include "Person.h"

class Student : public Person {
private:
        string subject;
        int mark;
public:
        Student();
        Student(string, int, string, int);
        Student(const Student&);
        ~Student(){}

        string get_subject() { return subject; }
        int get_mark() { return mark; }

        void set_subject(string s) { subject = s; }
        void set_mark(int m) { mark = m; }

        Student& operator=(const Student&);

        friend istream& operator >>(istream& in, Student& p);
        friend ostream& operator <<(ostream& out, const Student& p);

        void Show();

};
```

UML-диаграмма



Ответы на вопросы

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

   Ответ: чисто виртуальный метод — метод, не имеющий определения в базовом классе. Отличается от виртуального отсутствием тела. (Вместо тела пишется = 0)
2. Какой класс называется абстрактным?

   Ответ: класс, в котором объявлена хотя бы одна чисто виртуальная функция.
3. Для чего предназначены абстрактные классы?

   Ответ: для создания конкретных производных классов.
4. Что такое полиморфные функции?

   Ответ: функции, которые могут принимать в качестве параметров как данные класса родителя, так и данные классов потомков.
5. Чем полиморфизм отличается от принципа подстановки?

   Ответ: принцип подстановки — принцип полиморфизма.

6. Привести примеры иерархий с использованием абстрактных классов.

```
class Father {
public:
    virtual void f() = 0;
};

class Son {
public:
    virtual void s() = 0;
};
```

7. Привести примеры полиморфных функций.

```
class Father {
public:
    virtual void f() = 0;
};

class Son {
public:
    virtual void s() = 0;
};

void Polymorph(Father* object) { ... }
```

8. В каких случаях используется механизм позднего связывания?

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