

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	<p>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.</p>
----	---

Текст программы  
файл Lab\_Main.cpp

```
1  #include <string>
2  #include <iostream>
3  #include "Object.h"
4  #include "Person.h"
5  #include "Student.h"
6  #include "Vector.h"
7
8  using namespace std;
9
10  int main(){
11      system ("chcp 1251>null");
12
13      Vector v(5);
14      Person a;
15      cin >> a;
16      Student b;
17      cin >> b;
18      Object* p = &a;
19      v.Add(p);
20      p = &b;
21      v.Add(p);
22      cout << v;
23
24      return 0;
25  }
```

## Файл Object.h

```
1  #pragma once
2  #include <iostream>
3  #include <string>
4  using namespace std;
5
6  class Object {
7  public:
8      Object() {};
9      ~Object() {};
10     virtual void Show() = 0;
11 };

```

## Файл Person.cpp

```
1  #include "Person.h"
2
3  Person::Person(void) {
4      name = "";
5      age = 0;
6  }
7  Person::Person(string n, int a) {
8      name = n;
9      age = a;
10 }
11 Person::Person(const Person& p) {
12     name = p.name;
13     age = p.age;
14 }
15 Person::~Person(){}
16 void Person::set_name(string n) {
17     name = n;
18 }
19 void Person::set_age(int a) {
20     age = a;
21 }
22
23 Person& Person::operator=(const Person& p) {
24     if (this == &p) {
25         return *this;
26     }
27     name = p.name;
28     age = p.age;
29     return *this;
30 }
31
32 ostream& operator >>(ostream& in, Person& p) {
33     cout << "name: "; in >> p.name;
34     cout << "age: "; in >> p.age;
35     return in;
36 }
37 ostream& operator <<(ostream& out, const Person& p) {
38     out << "\nname: " << p.name;
39     out << "\nage: " << p.age << " y.o.";
40     return out;
41 }
42
43 void Person::Show() {
44     cout << "\nname: " << name;
45     cout << "\nage: " << age << " y.o.\n";
46 }
```

## Файл Person.h

```
1  #pragma once
2  #include<iostream>
3  #include<string>
4  #include "Object.h"
5
6  using namespace std;
7
8  class Person: public Object {
9  protected:
10     string name;
11     int age;
12 public:
13     void Show();
14     Person();
15     Person(string, int);
16     Person(const Person&);
17     virtual ~Person();
18     string get_name() { return name; };
19     int get_age() { return age; }
20
21     void set_name(string);
22     void set_age(int);
23
24     Person& operator=(const Person&);
25
26     friend istream& operator >>(istream& in, Person& p);
27     friend ostream& operator <<(ostream& out, const Person& p);
28 };

```

## Файл Student.cpp

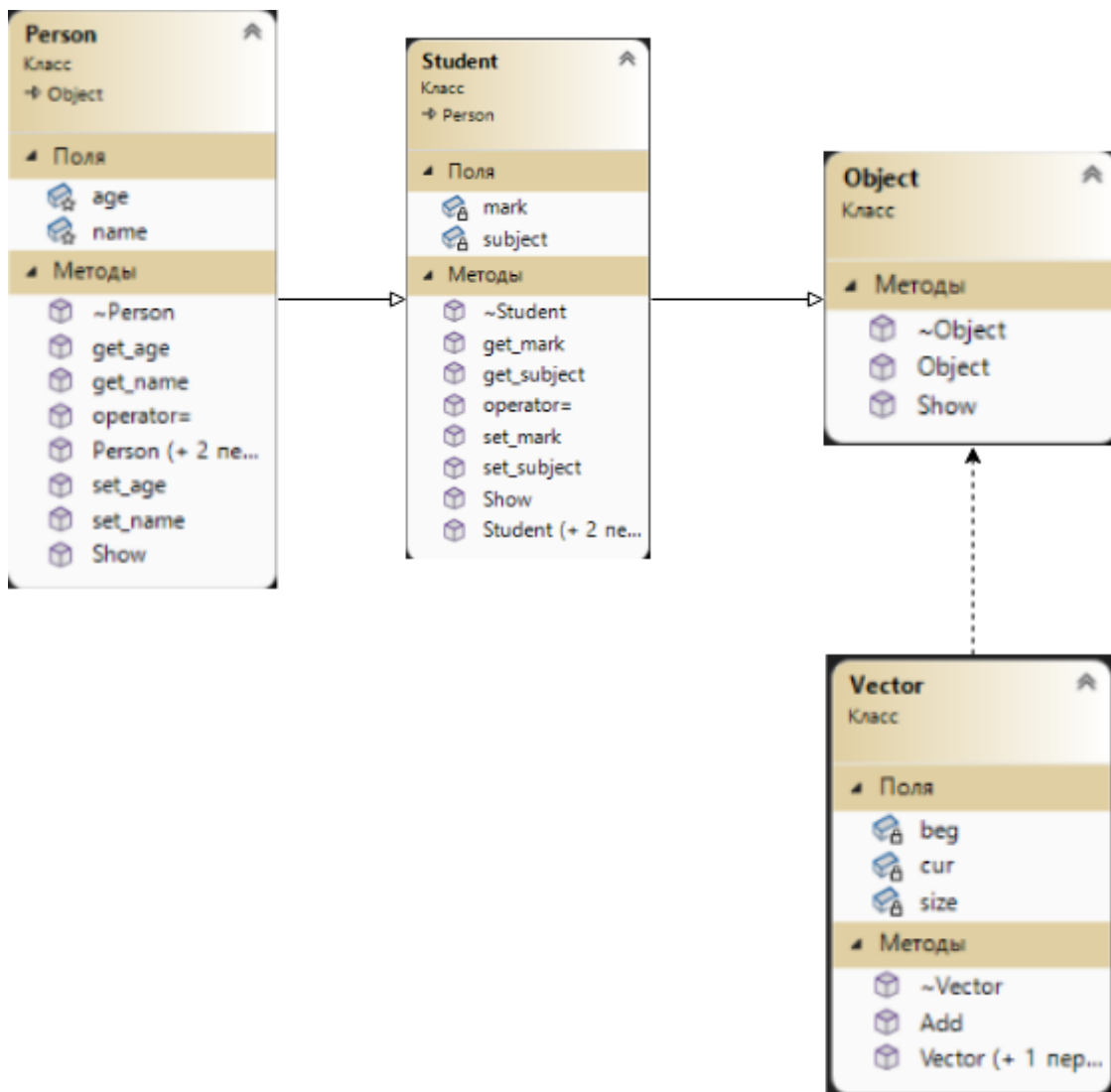
```
1  #include "Student.h"
2
3  Student::Student():Person(){
4      subject = "";
5      mark = 0;
6  }
7  Student::Student(string n, int a, string s, int m):Person(n,a) {
8      subject = s;
9      mark = m;
10 }
11 ✓ Student::Student(const Student& s) {
12     name = s.name;
13     age = s.age;
14     subject = s.subject;
15     mark = s.mark;
16 }
17
18 Student& Student::operator=(const Student& s) {
19     if (&s == this) {
20         return *this;
21     }
22     name = s.name;
23     age = s.age;
24     subject = s.subject;
25     mark = s.mark;
26     return *this;
27 }
28
29 istream& operator >>(istream& in, Student& p) {
30     cout << "name: "; in >> p.name;
31     cout << "age: "; in >> p.age;
32     cout << "subject: "; in >> p.subject;
33     cout << "mark: "; in >> p.mark;
34     return in;
35 }
36 ostream& operator <<(ostream& out, const Student& p) {
37     out << "\nname: " << p.name;
38     out << "\nage: " << p.age << " y.o.";
39     out << "\nsubject: " << p.subject;
40     out << "\nmark: " << p.mark;
41     return out;
42 }
43 ✓ void Student::Show() {
44     cout << "\nname: " << name;
45     cout << "\nage: " << age << " y.o.";
46     cout << "\nsubject: " << subject;
47     cout << "\nmark: " << mark << endl;
48 }
```

## Файл Student.h

```
1  #pragma once
2  #include "Person.h"
3
4  class Student : public Person {
5  private:
6      string subject;
7      int mark;
8  public:
9      Student();
10     Student(string, int, string, int);
11     Student(const Student&);
12     ~Student(){}
13
14     string get_subject() { return subject; }
15     int get_mark() { return mark; }
16
17     void set_subject(string s) { subject = s; }
18     void set_mark(int m) { mark = m; }
19
20     Student& operator=(const Student&);
21
22     friend istream& operator >>(istream& in, Student& p);
23     friend ostream& operator <<(ostream& out, const Student& p);
24
25     void Show();
26
27     };
```



## UML-диаграмма



## Тест

```
name: dfhdfh  
age: 24  
name: dfhdfh  
age: 34  
subject: dfhdfg  
mark: 2
```

```
name: dfhdfh  
age: 24 y.o.
```

```
name: dfhdfh  
age: 34 y.o.  
subject: dfhdfg  
mark: 2
```

C:\Users\MOkASiH\Desktop\Test\x64\Debug\Test.exe (процесс 15268) завершил работу с кодом 0.  
Нажмите любую клавишу, чтобы закрыть это окно...

## 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.