Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

Дисциплина: «Основы алгоритмизации и программирования»

Тема: «Полиморфизм»

Семестр 2

Выполнил работу

Студент группы ИВТ-22-2б

Устюгова Полина Дмитриевна

Проверил

Доцент кафедры ИТАС

Полякова Ольга Андреевна

Г. Пермь-2023

**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.

**Описание класса**

Базовый класс:

ЧЕЛОВЕК (PERSON)

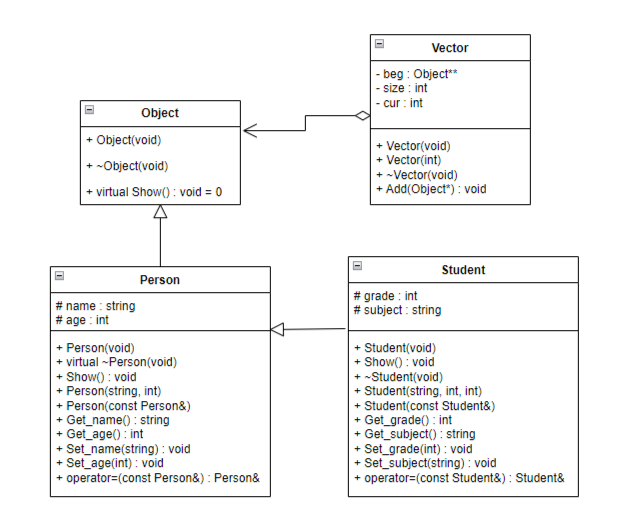
Имя (name) – string Возраст (age) – int

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.

**UML - Диаграмма**



**Рис. 1 – UML-диаграмма классов Person и Student**

**Контрольные вопросы**

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

Чисто виртуальный метод в базовом класса содержит признак равенства 0 вместо тела, например: virtual void f(int) = 0. Он должен переопределяться в производном классе (с помощью спецификатора override, который не является обязательным).

1. Какой класс называется абстрактным?

Абстрактный класс - класс, который содержит или наследует хотя бы одну чисто виртуальную функцию без переопределения.

1. Для чего предназначены абстрактные классы?

Абстрактные классы используются в качестве обобщения концепций, на основе которые можно создавать более конкретные производные классы.

Создание объектов абстрактного класса невозможно (компилятор выдаст ошибку), но можно использовать указатели и ссылки на типы абстрактных классов. Также, этот класс нельзя использовать при явном приведении типов (тип переменной, тип возвращаемого значения и т.п.)

class Object

{

public:

Object(void); public:

~Object(void);

virtual void Show() = 0;//чисто виртуальная функция

};

1. Что такое полиморфные функции?

Полиморфизм - взаимозаменяемость объектов.

В контексте реализации программы полиморфизм - возможность использования экземпляра класса-наследника там, где требуется экземпляр базового класса.

class Base // Абстрактный класс

{

virtual void print() = 0; // чисто виртуальная функция

};

class Derived : Base

{

public:

virtual void print(Base\* p)

{

cout << "It's all right";

}

};

void main()

{

Derived\* t = new Derived;

t.print(t); // в функцию передается базовый файл в качестве аргумента.

}

Абстрактный класс становится здесь шаблоном.

1. Чем полиморфизм отличается от принципа подстановки?

Полиморфизм - это возможность классов иметь разную реализацию одного и того же функционала в основном с помощью наследования. Это механизм работы языка программирования.

Принцип подстановки является “правилом хорошего кода". В хорошем коде все методы класса-потомка должны быть применимы к объекту класса родителя.

1. Привести примеры иерархий с использованием абстрактных классов.

class Object // Класс объекта

{

public:

virtual double weight() const = 0; // вес

virtual double volume() const = 0; // объём

};

class Sphere : public Object // класс сферы

{

int radius;

double pi = 3.14;

int density;

public:

Sphere(int R, double pi);

double volume() const override // переопределение функции

{

return 1.333\*(pi\*pow(radius, 3));

}

double weight() const override // переопределение функции

{

return volume() \* density;

}

};

1. Привести примеры полиморфных функций.

class Book

{

protected:

string title;

string author;

public:

Book(string t, string a)

{

title = t;

author = a;

}

virtual void show()

{

cout << title << endl << author;

}

};

class Page : public Book

{

protected:

int charter;

int page;

public:

Page(string t, string a, int ch, int p) : Book(t, a)

{

charter = ch;

page = p;

}

void show()

{

cout << title << endl << author;

cout << charter << endl << page;

}

};

1. В каких случаях используется механизм позднего связывания?

Если в иерархии унаследованных классов объявляется виртуальный элемент, то выполняется позднее связывание.

При позднем связывании вызов метода происходит на основании типа объекта, а не типа ссылки на базовый класс. Позднее связывание используется, если нужно реализовать полиморфизм.