ПНИПУ

Федеральное государственное автономное образовательное учреждение высшего образования «Пермский национальный исследовательский политехнический университет»

Лабораторная работа №5.

Наследование. Виртуальные функции. Полиморфизм.

Выполнил студент группы РИС-23-3Б

Буковский Денис Владимирович

Проверила доцент кафедры ИТАС О.А. Полякова

2024

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

Базовый класс: ЧЕЛОВЕК (PERSON)

Имя (name) — string

Возраст (age) — int

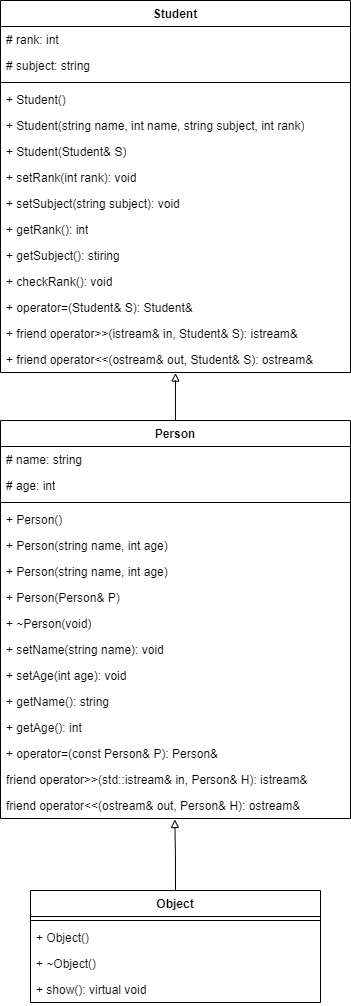
Определить методы изменения полей. Создать производный класс STUDENT, имеющий поля Предмет — string и Оценка — int.

Определить методы изменения полей и метод, выдающий сообщение о неудовлетворительной оценке.

1. Анализ задачи

Создадим абстрактный класс base, в нём определим виртуальный метод show(). Унаследуем от него класс Student со всеми соответствующими ему методами и определим методы в соответствии с заданием. Протестируем работу программы в функции main()

1. Блок-схема



1. Код

People.h

#pragma once

#include<iostream>

class Object {

public:

Object(void);

~Object(void);

virtual void show() = 0; // Чисто виртуальный метод

};

class Person : public Object {

protected:

std::string name;

int age;

public:

Person();

Person(const Person& T);

Person(std::string name, int age);

~Person(void);

void setName(std::string name);

void setAge(int age);

std::string getName();

int getAge();

void show() override;

Person& operator=(const Person& H);

friend std::istream& operator>>(std::istream& in, Person& H);

friend std::ostream& operator<<(std::ostream& out, Person& H);

};

class Student : public Person {

protected:

int rank;

std::string subject;

public:

Student();

Student(Student& S);

Student(std::string name, int age, std::string subject, int rank);

~Student();

Student& operator=(Student& S);

void setRank(int rank);

int getRank();

void setSubject(std::string subject);

std::string getSubject();

void checkRank();

void show() override;

friend std::istream& operator>>(std::istream& in, Student& H);

friend std::ostream& operator<<(std::ostream& out, Student& S);

};

People.cpp

#include <iostream>

#include "People.h"

Object::Object(void) {

std::cout << "Abstract base object was created" << std::endl;

};

Object::~Object(void) {

std::cout << "Abstract base object was destroyed" << std::endl;

};

Person::Person() {

name = "";

age = 0;

std::cout << "Person object was created" << std::endl;

}

Person::Person(const Person& T) {

name = T.name;

age = T.age;

std::cout << "Person object was created" << std::endl;

}

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

this->name = name;

this->age = age;

std::cout << "Person object was created" << std::endl;

}

Person::~Person(void) {

std::cout << "Person object was destroyed" << std::endl;

}

void Person::setName(std::string name) {

this->name = name;

}

void Person::setAge(int age) {

if (age > 0) this->age = age;

}

std::string Person::getName() {

return this->name;

}

int Person::getAge() {

return this->age;

}

void Person::show() {

std::cout << "Name: " << this->name << std::endl << "Age: " << this->age << std::endl;

}

Person& Person::operator=(const Person& H) {

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

name = H.name;

age = H.age;

return \*this;

}

std::istream& operator>>(std::istream& in, Person& H) {

std::cout << "Enter name: ";

in >> H.name;

std::cout << "Enter age: ";

in >> H.age;

std::cout << std::endl;

return in;

}

std::ostream& operator<<(std::ostream& out, Person& H) {

out << "Name: " << H.name << std::endl << "Age: " << H.age << std::endl;

return out;

}

Student::Student() {

name = "";

age = 0;

subject = "";

rank = 0;

std::cout << "Student object was created" << std::endl;

}

Student::Student(Student& S) {

this->name = S.name;

this->age = S.age;

this->subject = S.subject;

this->rank = S.rank;

std::cout << "Student object was created" << std::endl;

}

Student::Student(std::string name, int age, std::string subject, int rank) {

this->name = name;

this->age = age;

this->subject = subject;

this->rank = rank;

std::cout << "Student object was created" << std::endl;

}

Student::~Student() {

std::cout << "Student object was destroyed" << std::endl;

}

Student& Student::operator=(Student& S) {

this->name = S.name;

this->age = S.age;

this->subject = S.subject;

this->rank = S.rank;

return \*this;

}

void Student::setRank(int rank) {

this->rank = rank;

}

int Student::getRank() {

return this->rank;

}

void Student::setSubject(std::string subject) {

this->subject = subject;

}

std::string Student::getSubject() {

return this->subject;

}

void Student::checkRank() {

switch (rank) {

case 0: {

std::cout << "Нет оценок по предмету " << subject << "!" << std::endl;

break;

}

default:

std::cout << "Оценка по предмету " << subject << " - " << rank << std::endl;

}

}

void Student::show(){

std::cout << "Name: " << this->name << std::endl << "Age: " << this->age << std::endl << "Subject: " << this->subject << std::endl << "Rank: " << this->rank << std::endl;

}

std::istream& operator>>(std::istream& in, Student& H) {

std::cout << "Enter name: ";

in >> H.name;

std::cout << "Enter age: ";

in >> H.age;

std::cout << "Enter subject: ";

in >> H.subject;

std::cout << "Enter rank: ";

in >> H.rank;

return in;

}

std::ostream& operator<<(std::ostream& out, Student& S) {

out << "Name: " << S.name << std::endl << "Age: " << S.age << std::endl << "Subject: " << S.subject << std::endl << "Rank: " << S.rank << std::endl;

return out;

}

Main.cpp

#include <iostream>

#include "People.h"

using namespace std;

int main() {

Student student1;

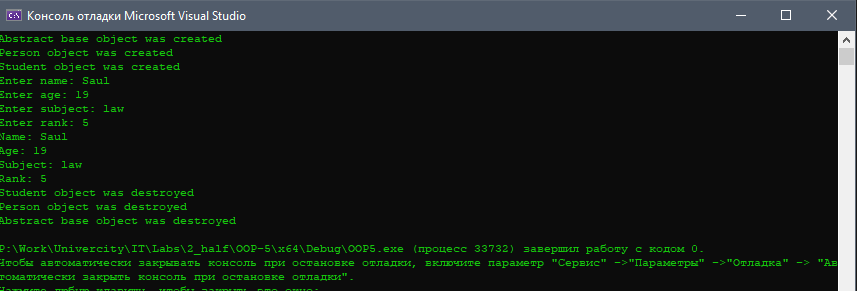
cin >> student1;

student1.show();

return 0;

}

1. Вывод программы



1. Ответы на контрольные вопросы

1. Чистый виртуальный метод – это виртуальный метод, который обязательно должен быть переопределён в классе-наследнике. Он служит для создания абстрактных классов.

2. Абстрактный класс – это класс который содержит чистые виртуальные методы и атрибуты, присущие любому наследнику.

3. Абстрактные классы создаются для поддержания полиморфизма.

4. Полиморфные функции – это функции, которые описаны в абстрактном классе-родителе и переопределены в наследниках.

5. Полиморфизм отличается от принципа подстановки тем, что при полиморфизме классы-наследники могут иметь новые методы или атрибуты, не присущие классу-родителю.

6. Базовый класс – Человек – Студент.

7. Вывод информации – show, присущ всем 3 классам.

8. Механизм позднего связывания используется для связи между классом-родителем и классами-наследниками.