**湖州师范学院信息工程学院**

**学生实验报告**

课程名称： 面向对象程序设计（C++）

班 级： 20240823

实验内容： 继承和模板

指导教师： 高强

实验时间： 2025.5.15

实验地点： 30-409

学生姓名： 李佳诚

得 分：

**湖州师范学院信息工程学院制**

# **一、实验目的**

掌握类的继承和模板的定义和使用。

# **二、实验内容**

**People.h**

#pragma once

#include <string>

class People {

protected:

std::string name;

int age;

public:

People(const std::string& name, int age);

People(const People& other);

virtual ~People() = default;

std::string getName() const;

int getAge() const;

void displayInfo() const;

};

**People.cpp**

#include "People.h"

#include <iostream>

People::People(const std::string& name, int age) : name(name), age(age) {}

People::People(const People& other) : name(other.name), age(other.age) {}

std::string People::getName() const { return name; }

int People::getAge() const { return age; }

void People::displayInfo() const {

std::cout << name << " " << age << std::endl;

}

**Soldier.h**

#pragma once

#include "People.h"

class Soldier : virtual public People {

protected:

int serviceYears;

public:

Soldier(const std::string& name, int age, int serviceYears);

Soldier(const Soldier& other);

~Soldier() = default;

void fight() const;

int getServiceYears() const;

};

**Soldier.cpp**

#include "Soldier.h"

#include <iostream>

Soldier::Soldier(const std::string& name, int age, int serviceYears)

: People(name, age), serviceYears(serviceYears) {}

Soldier::Soldier(const Soldier& other)

: People(other), serviceYears(other.serviceYears) {}

void Soldier::fight() const {

std::cout << getName() << "正在战斗" << std::endl;

}

int Soldier::getServiceYears() const { return serviceYears; }

**Doctor.h**

#pragma once

#include "People.h"

enum class Title { ATTENDING, ASSOCIATE, SENIOR };

class Doctor : virtual public People {

protected:

Title title;

public:

Doctor(const std::string& name, int age, Title title);

Doctor(const Doctor& other);

~Doctor() = default;

void treat() const;

Title getTitle() const;

};

**Doctor.cpp**

#include "Doctor.h"

#include <iostream>

Doctor::Doctor(const std::string& name, int age, Title title)

: People(name, age), title(title) {}

Doctor::Doctor(const Doctor& other)

: People(other), title(other.title) {}

void Doctor::treat() const {

std::cout << getName() << "正在治疗" << std::endl;

}

Title Doctor::getTitle() const { return title; }

**MilitaryDoctor.h**

#pragma once

#include "Soldier.h"

#include "Doctor.h"

class MilitaryDoctor : public Soldier, public Doctor {

private:

int taskCount;

public:

MilitaryDoctor(const std::string& name, int age, Title title, int serviceYears);

MilitaryDoctor(const MilitaryDoctor& other);

~MilitaryDoctor() = default;

void performDuty();

};

**MilitaryDoctor.cpp**

#include "MilitaryDoctor.h"

#include <iostream>

MilitaryDoctor::MilitaryDoctor(const std::string& name, int age, Title title, int serviceYears)

: People(name, age),

Soldier(name, age, serviceYears),

Doctor(name, age, title),

taskCount(0) {}

MilitaryDoctor::MilitaryDoctor(const MilitaryDoctor& other)

: People(other),

Soldier(other),

Doctor(other),

taskCount(other.taskCount) {}

void MilitaryDoctor::performDuty() {

taskCount++;

std::cout << getName() << "正在执行第" << taskCount << "次任务" << std::endl;

}

**RoleManager.h**

#pragma once

#include <vector>

template<typename T>

class RoleManager {

private:

std::vector<T> roles;

public:

RoleManager() = default;

void addRole(const T& role);

double getAverageAge();

void displayAll() const;

int getRoleCount() const;

T& getRole(int index);

};

template<typename T>

void RoleManager<T>::addRole(const T& role) {

roles.push\_back(role);

}

template<typename T>

double RoleManager<T>::getAverageAge() {

if (roles.empty()) return 0.0;

int total = 0;

for (const auto& role : roles) {

total += role.getAge();

}

return static\_cast<double>(total) / roles.size();

}

template<typename T>

void RoleManager<T>::displayAll() const {

for (const auto& role : roles) {

role.displayInfo();

}

}

template<typename T>

int RoleManager<T>::getRoleCount() const {

return roles.size();

}

template<typename T>

T& RoleManager<T>::getRole(int index) {

return roles[index];

}

**main.cpp**

#include <iostream>

#include "People.h"

#include "Soldier.h"

#include "Doctor.h"

#include "MilitaryDoctor.h"

#include "RoleManager.h"

int main() {

Soldier s1("张三", 28, 6);

Doctor d1("李四", 35, Title::ASSOCIATE);

MilitaryDoctor md1("王五", 40, Title::SENIOR, 15);

std::cout << "[Soldier 信息]" << std::endl;

s1.displayInfo();

s1.fight();

std::cout << "\n[Doctor 信息]" << std::endl;

d1.displayInfo();

d1.treat();

std::cout << "\n[MilitaryDoctor 信息]" << std::endl;

md1.displayInfo();

md1.performDuty();

std::cout << "\n[RoleManager 测试]" << std::endl;

RoleManager<People> manager;

manager.addRole(s1);

manager.addRole(d1);

manager.addRole(md1);

std::cout << "角色数量: " << manager.getRoleCount() << std::endl;

std::cout << "平均年龄: " << manager.getAverageAge() << std::endl;

std::cout << "所有角色信息：" << std::endl;

manager.displayAll();

return 0;

}