Темнохуд Михаил, ПЗПИ-16-2

Лабораторная работа №4

# Задание:

Для обраної проблемної області необхідно спроектувати від 3 і більше класів. Визначити ієрархію: базовий (і) клас (и) і клас (и) наслідники, їх методи (відповідно до концепції та з поведінкою об'єктів реального світу). Можливі одиночне (для оцінки «добре») або множинне наслідування (для оцінки «відмінно») двох видів: 1) базові класи класу спадкоємця не є спорідненими один одному; 2) базові класи класу спадкоємець мають спільного предка. Головна програма повинна демонструвати їх роботу та взаємозв’язок. Для класів необхідно обґрунтувати вибір: специфікаторів доступу до даних, вид спадкування, (не)використання віртуальних базових класів, абстрактних базових класів, чистих віртуальних функцій. Необхідно продемонструвати роботу віртуальних функцій.

# Код программы:

// lab4.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include "GunAPI.h"

void PrintTitle(LPCTSTR caption);

int main()

{

Gun\* g1 = new M16A4(\_T("M16A4 #1234"), 30, { \_T("5.56"), 1000 });

AutoRifle\* g2 = new M416(\_T("M416 #5678"), 30, { \_T("5.56"), 900 });

Gun\* g3 = new Kar98K(\_T("German Kar98K"), 5, { \_T("5.56"), 1500 });

Gun\* g4 = new Gun(\_T("Ordinary gun"), 20, { \_T("5.56"), 1000 });

PrintTitle(\_T("M16A4 TEST"));

g1->Shoot();

g1->Reload();

while (g1->HasBullets()) { ((M16A4\*)g1)->ShootBurst(); }

PrintTitle(\_T("M416 TEST"));

while (g2->HasBullets()) { g2->Shoot(); }

g2->Shoot();

g2->Reload();

while (g2->HasBullets()) { g2->Shoot(); }

g2->Reload();

g2->Shoot(25);

g2->Shoot(25);

g2->Reload();

g2->Shoot(29);

g2->Shoot();

g2->Shoot();

PrintTitle(\_T("Kar98K TEST"));

g3->Reload();

g3->Shoot();

for(int i = 0; i < 4; i++)

{

g3->Shoot();

g3->Reload();

}

system("pause");

delete g1;

delete g2;

delete g3;

return 0;

}

void PrintTitle(LPCTSTR caption)

{

\_tprintf(\_T("========================================\r\n"));

\_tprintf(\_T("=====%10s=====\r\n"), caption);

\_tprintf(\_T("========================================\r\n"));

}

// Gun.h : Defines the entry point for the console application.

//

#pragma once

#include "GunAPI.h"

enum

{

GUN\_OK = 0,

GUN\_ERROR = 1

};

struct Bullet

{

LPCTSTR Caliber;

int Power;

};

class Gun

{

protected:

TCHAR\* \_name;

Bullet \_bulletType;

int \_bulletTotal;

int \_bulletRemaining;

bool \_ready;

public:

Gun();

Gun(LPCTSTR name, int bulletTotal, Bullet bulletInfo);

Gun(const Gun &);

Gun& operator=(const Gun &gun);

virtual int Shoot();

virtual int Reload();

void ActionShoot();

void ActionReload();

void ActionReloadNeed();

void ActionReloadNoNeed();

bool Ready();

bool IsFull();

bool HasBullets();

int GetBulletsTotal();

int GetBulletsRemaining();

Bullet GetCaliber();

virtual ~Gun() {}

};

// IAuto.h : Defines the entry point for the console application.

//

#pragma once

#include "GunAPI.h"

class IAuto

{

protected:

virtual int Shoot(int count) = 0;

};

// IBurst.h : Defines the entry point for the console application.

//

#pragma once

#include "GunAPI.h"

class IBurst : public IAuto

{

protected:

int ShootBurst() { return Shoot(3); }

};

// AutoRifle.h : Defines the entry point for the console application.

//

#pragma once

#include "Gun.h"

class AutoRifle : public Gun, public IAuto

{

public:

AutoRifle()

:Gun() {}

AutoRifle(LPCTSTR name, int bulletTotal, Bullet bulletInfo)

: Gun(name, bulletTotal, bulletInfo) {}

AutoRifle(const Gun &gun) : Gun(gun) {}

virtual int Shoot();

virtual int Shoot(int count);

virtual int Reload();

~AutoRifle() {}

};

// Kar98K.h : Defines the entry point for the console application.

//

#pragma once

#include "GunAPI.h"

class Kar98K : public Gun

{

public:

Kar98K()

:Gun() {}

Kar98K(LPCTSTR name, int bulletTotal, Bullet bulletInfo)

: Gun(name, bulletTotal, bulletInfo) {}

Kar98K(const Gun &gun) : Gun(gun) {}

int Shoot();

int Reload();

~Kar98K() {}

};

// M416.h : Defines the entry point for the console application.

//

#pragma once

#include "GunAPI.h"

class M416 : public AutoRifle

{

public:

M416()

:AutoRifle() {}

M416(LPCTSTR name, int bulletTotal, Bullet bulletInfo)

: AutoRifle(name, bulletTotal, bulletInfo) {}

M416(const Gun &gun) : AutoRifle(gun) {}

int Shoot();

int Shoot(int count);

int Reload();

~M416() {}

};

// M16A4.h : Defines the entry point for the console application.

//

#pragma once

#include "GunAPI.h"

class M16A4 : public AutoRifle, public IBurst

{

public:

M16A4()

:AutoRifle() {}

M16A4(LPCTSTR name, int bulletTotal, Bullet bulletInfo)

: AutoRifle(name, bulletTotal, bulletInfo) {}

M16A4(const Gun &gun) : AutoRifle(gun) {}

int Shoot();

int Shoot(int count);

int ShootBurst();

int Reload();

~M16A4() {}

};

// GunAPI.h : Defines the entry point for the console application.

//

#pragma once

#include "Gun.h"

#include "IAuto.h"

#include "IBurst.h"

#include "AutoRifle.h"

#include "M416.h"

#include "M16A4.h"

#include "Kar98k.h"

// Gun.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include "GunAPI.h"

Gun::Gun()

{

\_name = new TCHAR[4];

\_tcscpy(\_name, \_T("Gun"));

\_bulletType = { \_T(""), 0 };

\_bulletTotal = 0;

\_bulletRemaining = 0;

\_ready = false;

}

Gun::Gun(LPCTSTR name, int bulletTotal, Bullet bulletInfo)

{

\_name = new TCHAR[\_tcslen(name)];

\_tcscpy(\_name, name);

\_bulletType = bulletInfo;

\_bulletTotal = bulletTotal;

\_bulletRemaining = 0;

\_ready = false;

}

Gun::Gun(const Gun &obj)

{

\_name = new TCHAR[\_tcslen(obj.\_name)];

\_tcscpy(\_name, obj.\_name);

\_bulletType = obj.\_bulletType;

\_bulletTotal = obj.\_bulletTotal;

\_bulletRemaining = obj.\_bulletRemaining;

\_ready = obj.\_ready;

}

Gun& Gun::operator=(const Gun &gun) {

if (this != &gun)

{

\_name = new TCHAR[\_tcslen(gun.\_name)];

\_tcscpy(\_name, gun.\_name);

\_bulletType = gun.\_bulletType;

\_bulletTotal = gun.\_bulletTotal;

\_bulletRemaining = gun.\_bulletRemaining;

\_ready = gun.\_ready;

}

return \*this;

}

int Gun::Shoot()

{

int result = GUN\_ERROR;

if (!HasBullets() || !\_ready)

{

ActionReloadNeed();

}

else

{

\_ready = false;

\_bulletRemaining--;

ActionShoot();

result = GUN\_OK;

}

\_tprintf(\_T("\r\n"));

return result;

}

int Gun::Reload()

{

int result = GUN\_ERROR;

if (IsFull() && \_ready)

{

ActionReloadNoNeed();

}

else

{

\_ready = true;

if (!HasBullets())

{

\_bulletRemaining = \_bulletTotal;

}

ActionReload();

result = GUN\_OK;

}

\_tprintf(\_T("\r\n"));

return result;

}

void Gun::ActionShoot()

{

\_tprintf(\_T("%s BDISH %d\r\n"), \_name, (\_bulletTotal - \_bulletRemaining));

}

void Gun::ActionReload()

{

\_tprintf(\_T("%s RELOADED, %d\r\n"), \_name, \_bulletRemaining);

}

void Gun::ActionReloadNeed()

{

\_tprintf(\_T("%s NEEDS TO BE RELOADED\r\n"), \_name);

}

void Gun::ActionReloadNoNeed()

{

\_tprintf(\_T("%s DOESNT NEED TO BE RELOADED\r\n"), \_name);

}

bool Gun::Ready()

{

return \_ready;

}

bool Gun::IsFull()

{

return \_bulletRemaining == \_bulletTotal;

}

bool Gun::HasBullets()

{

return \_bulletRemaining > 0;

}

int Gun::GetBulletsTotal()

{

return \_bulletTotal;

}

int Gun::GetBulletsRemaining()

{

return \_bulletRemaining;

}

Bullet Gun::GetCaliber()

{

return \_bulletType;

}

// AutoRifle.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include "GunAPI.h"

int AutoRifle::Shoot()

{

return Shoot(1);

}

int AutoRifle::Shoot(int count)

{

int result = GUN\_ERROR;

if (!HasBullets() || !\_ready)

{

ActionReloadNeed();

}

else

{

for (int i = 0; i < count && \_bulletRemaining != 0; i++)

{

\_bulletRemaining--;

ActionShoot();

result = GUN\_OK;

}

}

\_tprintf(\_T("\r\n"));

return result;

}

int AutoRifle::Reload()

{

int result = GUN\_ERROR;

if (IsFull() && \_ready)

{

ActionReloadNoNeed();

}

else

{

\_ready = true;

\_bulletRemaining = \_bulletTotal;

ActionReload();

result = GUN\_OK;

}

\_tprintf(\_T("\r\n"));

return result;

}

// Kar98K.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include "GunAPI.h"

int Kar98K::Shoot()

{

return Gun::Shoot();

}

int Kar98K::Reload()

{

return Gun::Reload();

}

// M416.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include "GunAPI.h"

int M416::Shoot()

{

return AutoRifle::Shoot(1);

}

int M416::Shoot(int count)

{

return AutoRifle::Shoot(count);

}

int M416::Reload()

{

\_tprintf(\_T("God bless US"));

return AutoRifle::Reload();

}

// M16A4.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include "GunAPI.h"

int M16A4::Shoot()

{

return AutoRifle::Shoot(1);

}

int M16A4::Shoot(int count)

{

return AutoRifle::Shoot(count);

}

int M16A4::ShootBurst()

{

return AutoRifle::Shoot(3);

}

int M16A4::Reload()

{

return AutoRifle::Reload();

}

# Результат выполнение:





