Министерство науки и высшего образования Российской Федерации

Федеральное государственное образовательное учреждение

высшего образования

«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

Отчет защищен с отметкой\_\_\_\_\_

Преподаватель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

« \_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022г.

Отчет

по лабораторной работе №12

**«Наследование + Полиморфизм»**

по дисциплине «Программирование - 3 семестр»

Студент группы ПИ-02 Немчинов В. И.

Преподаватель доцент, к.т.н. Троцкий В.

Барнаул 2022 г.

**Задание:**

Модифицировать ваши проекты на С++, C# и Java из предыдущих ЛР.

1) В main продемонстрировать работу с одно и двумерными массивами объектов вашего класса;

!) Git обязателен;

!) В отчете по ЛР должна быть ссылка на Githab.

***Репозиторий:*** https://github.com/Zireael-of-Hope/Lab12

Код программы:

С++:

Lab12\_C++:

#include <iostream>

#include <conio.h>

#include <Color.h>

#include <Weapon.h>

#include <Human.h>

int main()

{

Weapon\* M4A1;

M4A1 = new AssaultRifle(Rarity::Standard, "TOP", 100, 800, 40, 30, 1.5, AmmoType::AMMO\_5\_56);

AssaultRifle AK\_105(Rarity::Advanced, "Peacemaker", 100, 800, 40, 30, 1.5, AmmoType::AMMO\_7\_62);

Handgun Glock17(Rarity::High\_End, "CrappyThing", 55, 350, 40, 17, 2.5, AmmoType::AMMO\_9);

Knife Bayonet(Rarity::Elite, "");

Bayonet.SetSkin("Bloody Moon");

Bayonet.GetStatus();

Bayonet.SetSkin("Green Sky", 1);

Bayonet.GetStatus();

//Если не использовать using Class::Method в Knife

//То будет доступен лишь один методов SetSkin()

//Какой из методов доступен - определяет Тип Bayonet

Human MyBuddy("Hamilton", 78, 180, 37);

SWAT\_Operator Operator("Johnson", 87, 190, 27, Rank::LIEUTENANT);

Operator = MyBuddy;

cout << Operator;

Operator.UseWeapon(\*M4A1);

cout << "\n"; Operator.ExampleCallVirtualMethod();

cout << endl;

Human \*Buddy, \*Buddy2;

Buddy = new SWAT\_Operator("Bottas", 76, 174, 2, Rank::DEPUTY);

Buddy2 = new Human("Norris", 81, 185, 24);

Buddy->UseWeapon(\*M4A1);

cout << endl;

Buddy2->UseWeapon(Glock17);

cout << endl;

//Не виртуальная функция

AK\_105.LevelUp(); //AssaultRifle::LevelUp();

AK\_105.GetStatus();

M4A1->LevelUp(); //Weapon::LevelUp();

M4A1->GetStatus();

M4A1->LevelUp();

M4A1->GetStatus();

Glock17.Weapon::LevelUp();

Glock17.GetStatus();

}

Human.h:

#pragma once

#include <iostream>

#include <Weapon.h>

using namespace std;

enum class Rank {

DEPUTY,

SUPERVISOR,

SERGEANT,

LIEUTENANT

};

class Human

{

protected:

string name;

double weight;

double height;

int age;

public:

Human(string name, double weight, double height, int age);

string GetName();

double GetWeight();

double GetHeight();

int GetAge();

virtual void UseWeapon(Weapon& weapon);

friend ostream& operator<< (ostream& out, const Human& human);

void ExampleCallVirtualMethod();

private:

virtual string ExampleVirtualMethod();

};

class SWAT\_Operator final : public Human

{

private:

Rank rank;

public:

SWAT\_Operator(string name, double weight, double height, int age, Rank rank);

void UseWeapon(Weapon& weapon) override final;

SWAT\_Operator& operator=(Human& other);

};

Human.cpp:

#include "Human.h"

SWAT\_Operator::SWAT\_Operator(string name, double weight, double height, int age, Rank rank)

: Human(name, weight, height, age)

{

this->rank = rank;

}

void SWAT\_Operator::UseWeapon(Weapon& weapon)

{

cout << "Officer using deadly force! ";

weapon.Attack();

}

SWAT\_Operator& SWAT\_Operator::operator=(Human& other)

{

if (this == &other)

return \*this;

age = other.GetAge();

height = other.GetHeight();

name = other.GetName();

weight = other.GetWeight();

}

Human::Human(string name, double weight, double height, int age)

{

this->age = age;

this->height = height;

this->name = name;

this->weight = weight;

}

string Human::GetName()

{

return name;

}

double Human::GetWeight()

{

return weight;

}

double Human::GetHeight()

{

return height;

}

int Human::GetAge()

{

return age;

}

void Human::UseWeapon(Weapon& weapon)

{

cout << "Suspect using deadly force! ";

weapon.Attack();

}

void Human::ExampleCallVirtualMethod()

{

cout << ExampleVirtualMethod();

}

string Human::ExampleVirtualMethod()

{

return "Hello World";

}

ostream& operator<<(ostream& out, const Human& human)

{

out << "Info: " << human.name << " Age " << human.age << " Weight " << human.weight << " Height " << human.height << endl;

return out;

}

Weapon.h:

#pragma once

#include <iostream>

#include <Color.h>

using namespace std;

enum class Rarity {

FOR\_EXAMPLE, //присвоится при вызове повышения уровня из дочернего класса AssaultRifle

Standard,

Advanced,

High\_End,

Elite

};

enum class AmmoType {

AMMO\_45,

AMMO\_9,

AMMO\_5\_56,

AMMO\_7\_62

};

class Weapon

{

protected:

Rarity rare;

int id;

int damage;

string skin;

Weapon();

Weapon(Rarity rare, string skin);

virtual void InspectWeapon() = 0;

virtual void UpdateDueToRarity() = 0;

public:

~Weapon();

virtual void Attack() = 0;

void SetSkin(string skin\_name);

bool LevelUp();

void GetStatus();

};

class Knife : public Weapon

{

private:

virtual void UpdateDueToRarity() override final;

public:

Knife(Rarity rare, string skin);

using Weapon::SetSkin;

void SetSkin(string skin\_name, int special\_skin);

virtual void Attack() override final;

virtual void InspectWeapon() override final;

};

class Firearm : public Weapon

{

protected:

Firearm(Rarity rare, string skin, int effective\_range, int fire\_rate, int damage, int magazine, double reload\_time, AmmoType ammo\_type);

int effective\_range;

int fire\_rate;

int magazine;

double reload\_time;

AmmoType ammo\_type;

void FirearmMechanic(AmmoType ammo\_type, int effective\_range);

};

class Handgun final : public Firearm

{

private:

virtual void UpdateDueToRarity() override final;

public:

Handgun(Rarity rare, string skin, int effective\_range, int fire\_rate, int damage, int magazine, double reload\_time, AmmoType ammo\_type);

virtual void Attack() override;

virtual void InspectWeapon() override;

};

class AssaultRifle final : public Firearm

{

private:

virtual void UpdateDueToRarity() override final;

public:

AssaultRifle(Rarity rare, string skin, int effective\_range, int fire\_rate, int damage, int magazine, double reload\_time, AmmoType ammo\_type);

virtual void Attack() override;

virtual void InspectWeapon() override;

bool LevelUp();

};

Weapon.cpp:

#include "Weapon.h"

Weapon::Weapon()

{

rare = Rarity::Standard;

skin = "Standard";

}

Weapon::Weapon(Rarity rare, string skin)

{

this->rare = rare;

this->skin = skin;

}

Weapon::~Weapon()

{

}

void Weapon::SetSkin(string skin\_name)

{

skin = skin\_name;

}

void Knife::Attack()

{

cout << "\*cutting stuff\*" << endl;

}

bool Weapon::LevelUp()

{

switch (rare) {

case Rarity::Standard:

rare = Rarity::Advanced;

UpdateDueToRarity();

return true;

case Rarity::Advanced:

rare = Rarity::High\_End;

UpdateDueToRarity();

return true;

case Rarity::High\_End:

rare = Rarity::Elite;

UpdateDueToRarity();

return true;

default: return false;

}

}

void Knife::InspectWeapon()

{

cout << "\*inspecting knife\*" << endl;

}

void Weapon::GetStatus()

{

switch (rare) {

case Rarity::FOR\_EXAMPLE:

SetColor(4, 0);

break;

case Rarity::Standard:

SetColor(2, 0);

break;

case Rarity::Advanced:

SetColor(9, 0);

break;

case Rarity::High\_End:

SetColor(5, 0);

break;

case Rarity::Elite:

SetColor(10, 0);

break;

}

switch (id) {

case 0:

cout << "Knife: ";

break;

case 1:

cout << "Handgun: ";

break;

case 2:

cout << "Assault Rifle: ";

break;

case 3:

cout << "Sniper Rifles: ";

break;

}

SetColor(15, 0);

cout << "Damage: " << damage << " | Skin: " << skin << endl;

}

Knife::Knife(Rarity rare, string skin) : Weapon(rare, skin)

{

id = 0;

damage = 55;

UpdateDueToRarity();

}

void Knife::SetSkin(string skin\_name, int special\_skin)

{

skin = skin\_name + "\_KNIFE";

}

void Knife::UpdateDueToRarity()

{

switch (rare) {

case Rarity::FOR\_EXAMPLE:

cout << "\n\nAssaultRifle LevelUp methos has been called!\n" << endl;

break;

case Rarity::Standard:

damage += 0;

break;

case Rarity::Advanced:

damage += 5;

break;

case Rarity::High\_End:

damage += 10;

break;

case Rarity::Elite:

damage += 15;

break;

}

}

Handgun::Handgun(Rarity rare, string skin, int effective\_range, int fire\_rate, int damage, int magazine, double reload\_time, AmmoType ammo\_type)

: Firearm(rare, skin, effective\_range, fire\_rate, damage, magazine, reload\_time, ammo\_type)

{

}

void Handgun::Attack()

{

cout << "\*shooting with pistol: ";

FirearmMechanic(ammo\_type, effective\_range);

cout << "\*";

}

void Firearm::FirearmMechanic(AmmoType ammo\_type, int effective\_range)

{

cout << "BANG with ";

switch (ammo\_type) {

case AmmoType::AMMO\_9:

cout << "9mm ";

break;

case AmmoType::AMMO\_45:

cout << "45.ACP ";

break;

case AmmoType::AMMO\_5\_56:

cout << "5.56 ";

break;

case AmmoType::AMMO\_7\_62:

cout << "7.62 ";

break;

}

cout << "at " << effective\_range << " meters";

}

void Handgun::InspectWeapon()

{

cout << "\*inspecting handgun\*" << endl;

}

void Handgun::UpdateDueToRarity()

{

switch (rare) {

case Rarity::Standard:

damage += 0;

effective\_range += 0;

reload\_time = reload\_time - 0;

break;

case Rarity::Advanced:

damage += 4;

effective\_range += 10;

reload\_time = reload\_time - 0.3;

break;

case Rarity::High\_End:

damage += 8;

effective\_range += 20;

reload\_time = reload\_time - 0.6;

break;

case Rarity::Elite:

damage += 12;

effective\_range += 30;

reload\_time = reload\_time - 0.9;

break;

}

}

AssaultRifle::AssaultRifle(Rarity rare, string skin, int effective\_range, int fire\_rate, int damage, int magazine, double reload\_time, AmmoType ammo\_type)

: Firearm(rare, skin, effective\_range, fire\_rate, damage, magazine, reload\_time, ammo\_type)

{

}

void AssaultRifle::Attack()

{

cout << "\*automatic shooting with assault rifle: ";

FirearmMechanic(ammo\_type, effective\_range);

cout << "\*";

}

void AssaultRifle::InspectWeapon()

{

cout << "\*inspecting assault rifle\*" << endl;

}

bool AssaultRifle::LevelUp()

{

rare = Rarity::FOR\_EXAMPLE;

return true;

}

void AssaultRifle::UpdateDueToRarity()

{

switch (rare) {

case Rarity::Standard:

break;

case Rarity::Advanced:

damage += 5;

effective\_range += 13;

reload\_time = reload\_time - 0.3;

break;

case Rarity::High\_End:

damage += 10;

effective\_range += 26;

reload\_time = reload\_time - 0.6;

break;

case Rarity::Elite:

damage += 15;

effective\_range += 39;

reload\_time = reload\_time - 0.9;

break;

}

}

Firearm::Firearm(Rarity rare, string skin, int effective\_range, int fire\_rate, int damage, int magazine, double reload\_time, AmmoType ammo\_type)

: Weapon(rare, skin)

{

id = 2;

this->effective\_range = effective\_range;

this->fire\_rate = fire\_rate;

this->damage = damage;

this->magazine = magazine;

this->reload\_time = reload\_time;

this->ammo\_type = ammo\_type;

}

C#:

Program.cs:

using System;

using System.Collections.Generic;

using System.Linq;

namespace Lab12\_CS // Note: actual namespace depends on the project name.

{

public class Program

{

public static void Main(string[] args)

{

Weapon M4A1 = new AssaultRifle(Rarity.Elite, "TOP", 100, 800, 40, 30, 1.5, AmmoType.AMMO\_5\_56);

AssaultRifle AK\_105 = new AssaultRifle(Rarity.Advanced, "Peacemaker", 100, 800, 40, 30, 1.5, AmmoType.AMMO\_7\_62);

Handgun Glock17 = new Handgun(Rarity.High\_End, "CrappyThing", 55, 350, 40, 17, 2.5, AmmoType.AMMO\_9);

Knife Bayonet = new Knife(Rarity.Elite, "");

Bayonet.SetSkin("Bloody Moon");

Bayonet.GetStatus();

Bayonet.SetSkin("Green Sky", 1);

Bayonet.GetStatus();

Human MyBuddy = new Human("Hamilton", 78, 180, 37);

Human MuBuddyCopy = new Human(MyBuddy);

SWAT\_Operator Operator = new SWAT\_Operator("Johnson", 87, 190, 27, Rank.LIEUTENANT);

Console.WriteLine(MyBuddy.toString());

Console.WriteLine(MuBuddyCopy.toString());

Console.WriteLine(Operator.toString());

Operator.UseWeapon(M4A1);

Console.WriteLine(""); Operator.ExampleCallVirtualMethod();

Console.WriteLine("");

Human Buddy, Buddy2;

Buddy = new SWAT\_Operator("Bottas", 76, 174, 2, Rank.DEPUTY);

Buddy2 = new Human("Norris", 81, 185, 24);

Buddy.UseWeapon(M4A1);

Console.WriteLine("");

Buddy2.UseWeapon(Glock17);

Console.WriteLine("");

}

}

}

Human.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab12\_CS

{

enum Rank

{

DEPUTY,

SUPERVISOR,

SERGEANT,

LIEUTENANT

};

internal class Human : IHuman

{

protected String name;

protected double weight;

protected double height;

protected int age;

public Human(String name, double weight, double height, int age) {

this.age = age;

this.height = height;

this.name = name;

this.weight = weight;

}

public Human(Human other) {

name = other.name;

weight = other.weight;

height = other.height;

age = other.age;

}

public String GetName() {

return name;

}

public double GetWeight() {

return weight;

}

public double GetHeight() {

return height;

}

public int GetAge() {

return age;

}

public virtual void UseWeapon(Weapon weapon) {

Console.Write("Suspect using deadly force! ");

weapon.Attack();

}

public String toString() {

return ("Info: " + name + " Age " + age + " Weight " + weight + " Height " + height);

}

public void ExampleCallVirtualMethod() {

Console.Write(ExampleVirtualMethod());

}

private String ExampleVirtualMethod() {

return "Hello World";

}

}

class SWAT\_Operator : Human

{

private Rank rank;

public SWAT\_Operator(String name, double weight, double height, int age, Rank rank)

: base(name, weight, height, age)

{

this.rank = rank;

}

public override void UseWeapon(Weapon weapon) {

Console.Write("Officer using deadly force! ");

weapon.Attack();

}

}

}

Weapon.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab12\_CS

{

enum Rarity

{

FOR\_EXAMPLE, //присвоится при вызове повышения уровня из дочернего класса AssaultRifle

Standard,

Advanced,

High\_End,

Elite

};

enum AmmoType

{

AMMO\_45,

AMMO\_9,

AMMO\_5\_56,

AMMO\_7\_62

};

internal abstract class Weapon

{

protected Rarity rare;

protected int id;

protected int damage;

protected String skin;

protected Weapon()

{

rare = Rarity.Standard;

skin = "Standard";

}

protected Weapon(Rarity rare, String skin)

{

this.rare = rare;

this.skin = skin;

}

protected abstract void UpdateDueToRarity();

public abstract void InspectWeapon();

public abstract void Attack();

public void SetSkin(String skin\_name)

{

skin = skin\_name;

}

public bool LevelUp()

{

switch (rare)

{

case Rarity.Standard:

rare = Rarity.Advanced;

UpdateDueToRarity();

return true;

case Rarity.Advanced:

rare = Rarity.High\_End;

UpdateDueToRarity();

return true;

case Rarity.High\_End:

rare = Rarity.Elite;

UpdateDueToRarity();

return true;

default: return false;

}

}

public void GetStatus()

{

switch (rare)

{

case Rarity.Standard:

Console.Write("Standard: ");

break;

case Rarity.Advanced:

Console.Write("Advanced: ");

break;

case Rarity.High\_End:

Console.Write("High\_End: ");

break;

case Rarity.Elite:

Console.Write("Elite: ");

break;

}

switch (id)

{

case 0:

Console.Write("Knife: ");

break;

case 1:

Console.Write("Handgun: ");

break;

case 2:

Console.Write("Assault Rifle: ");

break;

case 3:

Console.Write("Sniper Rifles: ");

break;

}

Console.WriteLine("Damage: " + damage + " | Skin: " + skin);

}

}

class Knife : Weapon

{

protected override void UpdateDueToRarity()

{

switch (rare)

{

case Rarity.FOR\_EXAMPLE:

Console.WriteLine("\n\nAssaultRifle LevelUp methos has been called!\n");

break;

case Rarity.Standard:

break;

case Rarity.Advanced:

damage += 5;

break;

case Rarity.High\_End:

damage += 10;

break;

case Rarity.Elite:

damage += 15;

break;

}

}

public Knife(Rarity rare, String skin)

: base(rare, skin)

{

id = 0;

damage = 55;

UpdateDueToRarity();

}

public void SetSkin(String skin\_name, int special\_skin)

{

skin = skin\_name + "\_KNIFE";

}

public override void Attack()

{

Console.WriteLine("\*cutting stuff\*");

}

public override void InspectWeapon()

{

Console.WriteLine("\*inspecting knife\*");

}

};

abstract class Firearm : Weapon

{

protected Firearm(Rarity rare, String skin, int effective\_range, int fire\_rate, int damage, int magazine, double reload\_time, AmmoType ammo\_type)

: base(rare, skin)

{

id = 2;

this.effective\_range = effective\_range;

this.fire\_rate = fire\_rate;

this.damage = damage;

this.magazine = magazine;

this.reload\_time = reload\_time;

this.ammo\_type = ammo\_type;

}

protected int effective\_range;

protected int fire\_rate;

protected int magazine;

protected double reload\_time;

protected AmmoType ammo\_type;

protected void FirearmMechanic(AmmoType ammo\_type, int effective\_range)

{

Console.Write("BANG with ");

switch (ammo\_type)

{

case AmmoType.AMMO\_9:

Console.Write("9mm ");

break;

case AmmoType.AMMO\_45:

Console.Write("45.ACP ");

break;

case AmmoType.AMMO\_5\_56:

Console.Write("5.56 ");

break;

case AmmoType.AMMO\_7\_62:

Console.Write("7.62 ");

break;

}

Console.Write("at " + effective\_range + " meters");

}

};

class Handgun : Firearm

{

protected override void UpdateDueToRarity()

{

switch (rare)

{

case Rarity.Standard:

reload\_time = reload\_time - 0;

break;

case Rarity.Advanced:

damage += 4;

effective\_range += 10;

reload\_time = reload\_time - 0.3;

break;

case Rarity.High\_End:

damage += 8;

effective\_range += 20;

reload\_time = reload\_time - 0.6;

break;

case Rarity.Elite:

damage += 12;

effective\_range += 30;

reload\_time = reload\_time - 0.9;

break;

}

}

public Handgun(Rarity rare, String skin, int effective\_range, int fire\_rate, int damage, int magazine, double reload\_time, AmmoType ammo\_type)

: base(rare, skin, effective\_range, fire\_rate, damage, magazine, reload\_time, ammo\_type)

{

}

public override void Attack()

{

Console.Write("\*shooting with pistol: ");

FirearmMechanic(ammo\_type, effective\_range);

Console.Write("\*");

}

public override void InspectWeapon()

{

Console.WriteLine("\*inspecting handgun\*");

}

};

class AssaultRifle : Firearm

{

protected override void UpdateDueToRarity()

{

switch (rare)

{

case Rarity.Standard:

break;

case Rarity.Advanced:

damage += 5;

effective\_range += 13;

reload\_time = reload\_time - 0.3;

break;

case Rarity.High\_End:

damage += 10;

effective\_range += 26;

reload\_time = reload\_time - 0.6;

break;

case Rarity.Elite:

damage += 15;

effective\_range += 39;

reload\_time = reload\_time - 0.9;

break;

}

}

public AssaultRifle(Rarity rare, String skin, int effective\_range, int fire\_rate, int damage, int magazine, double reload\_time, AmmoType ammo\_type)

: base(rare, skin, effective\_range, fire\_rate, damage, magazine, reload\_time, ammo\_type)

{

}

public override void Attack()

{

Console.Write("\*automatic shooting with assault rifle: ");

FirearmMechanic(ammo\_type, effective\_range);

Console.Write("\*");

}

public override void InspectWeapon()

{

Console.WriteLine("\*inspecting assault rifle\*");

}

}

}

IHuman.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab12\_CS

{

internal interface IHuman

{

public void UseWeapon(Weapon weapon);

public int GetAge();

public String GetName();

public double GetWeight();

public double GetHeight();

}

}

Java:

Main.Java:

package com.company;

public class Main {

public static void main(String[] args) {

Weapon M4A1 = new AssaultRifle(Rarity.Elite, "TOP", 100, 800, 40, 30, 1.5, AmmoType.AMMO\_5\_56);

AssaultRifle AK\_105 = new AssaultRifle(Rarity.Advanced, "Peacemaker", 100, 800, 40, 30, 1.5, AmmoType.AMMO\_7\_62);

Handgun Glock17 = new Handgun(Rarity.High\_End, "CrappyThing", 55, 350, 40, 17, 2.5, AmmoType.AMMO\_9);

Knife Bayonet = new Knife(Rarity.Elite, "");

Bayonet.SetSkin("Bloody Moon");

Bayonet.GetStatus();

Bayonet.SetSkin("Green Sky", 1);

Bayonet.GetStatus();

Human MyBuddy = new Human("Hamilton", 78, 180, 37);

Human MuBuddyCopy = new Human(MyBuddy);

SWAT\_Operator Operator = new SWAT\_Operator("Johnson", 87, 190, 27, Rank.LIEUTENANT);

System.out.println(MyBuddy.toString());

System.out.println(MuBuddyCopy.toString());

System.out.println(Operator.toString());

Operator.UseWeapon(M4A1);

System.out.println(""); Operator.ExampleCallVirtualMethod();

System.out.println("");

Human Buddy, Buddy2;

Buddy = new SWAT\_Operator("Bottas", 76, 174, 2, Rank.DEPUTY);

Buddy2 = new Human("Norris", 81, 185, 24);

Buddy.UseWeapon(M4A1);

System.out.println("");

Buddy2.UseWeapon(Glock17);

System.out.println("");

}

}

Weapon.Java:

package com.company;

enum Rarity {

FOR\_EXAMPLE, //присвоится при вызове повышения уровня из дочернего класса AssaultRifle

Standard,

Advanced,

High\_End,

Elite

};

enum AmmoType {

AMMO\_45,

AMMO\_9,

AMMO\_5\_56,

AMMO\_7\_62

};

public abstract class Weapon {

protected Rarity rare;

protected int id;

protected int damage;

protected String skin;

protected Weapon() {

rare = Rarity.Standard;

skin = "Standard";

}

protected Weapon(Rarity rare, String skin) {

this.rare = rare;

this.skin = skin;

}

protected abstract void UpdateDueToRarity();

public abstract void InspectWeapon();

public abstract void Attack();

public void SetSkin(String skin\_name) {

skin = skin\_name;

}

public boolean LevelUp() {

switch (rare) {

case Standard:

rare = Rarity.Advanced;

UpdateDueToRarity();

return true;

case Advanced:

rare = Rarity.High\_End;

UpdateDueToRarity();

return true;

case High\_End:

rare = Rarity.Elite;

UpdateDueToRarity();

return true;

default: return false;

}

}

public void GetStatus() {

switch (rare) {

case Standard:

System.out.print("Standard: ");

break;

case Advanced:

System.out.print("Advanced: ");

break;

case High\_End:

System.out.print("High\_End: ");

break;

case Elite:

System.out.print("Elite: ");

break;

}

switch (id) {

case 0:

System.out.print("Knife: ");

break;

case 1:

System.out.print("Handgun: ");

break;

case 2:

System.out.print("Assault Rifle: ");

break;

case 3:

System.out.print("Sniper Rifles: ");

break;

}

System.out.println("Damage: " + damage + " | Skin: " + skin);

}

};

class Knife extends Weapon {

@Override

protected void UpdateDueToRarity() {

switch (rare) {

case FOR\_EXAMPLE:

System.out.println("\n\nAssaultRifle LevelUp methos has been called!\n");

break;

case Standard:

break;

case Advanced:

damage += 5;

break;

case High\_End:

damage += 10;

break;

case Elite:

damage += 15;

break;

}

}

public Knife(Rarity rare, String skin) {

super(rare, skin);

id = 0;

damage = 55;

UpdateDueToRarity();

}

public void SetSkin(String skin\_name, int special\_skin) {

skin = skin\_name + "\_KNIFE";

}

@Override

public void Attack() {

System.out.println("\*cutting stuff\*");

}

@Override

public void InspectWeapon() {

System.out.println("\*inspecting knife\*");

}

};

abstract class Firearm extends Weapon {

protected Firearm(Rarity rare, String skin, int effective\_range, int fire\_rate, int damage, int magazine, double reload\_time, AmmoType ammo\_type) {

super(rare, skin);

id = 2;

this.effective\_range = effective\_range;

this.fire\_rate = fire\_rate;

this.damage = damage;

this.magazine = magazine;

this.reload\_time = reload\_time;

this.ammo\_type = ammo\_type;

}

protected int effective\_range;

protected int fire\_rate;

protected int magazine;

protected double reload\_time;

protected AmmoType ammo\_type;

protected void FirearmMechanic(AmmoType ammo\_type, int effective\_range) {

System.out.print("BANG with ");

switch (ammo\_type) {

case AMMO\_9:

System.out.print("9mm ");

break;

case AMMO\_45:

System.out.print("45.ACP ");

break;

case AMMO\_5\_56:

System.out.print("5.56 ");

break;

case AMMO\_7\_62:

System.out.print("7.62 ");

break;

}

System.out.print("at " + effective\_range + " meters");

}

};

class Handgun extends Firearm {

@Override

protected void UpdateDueToRarity() {

switch (rare) {

case Standard:

reload\_time = reload\_time - 0;

break;

case Advanced:

damage += 4;

effective\_range += 10;

reload\_time = reload\_time - 0.3;

break;

case High\_End:

damage += 8;

effective\_range += 20;

reload\_time = reload\_time - 0.6;

break;

case Elite:

damage += 12;

effective\_range += 30;

reload\_time = reload\_time - 0.9;

break;

}

}

public Handgun(Rarity rare, String skin, int effective\_range, int fire\_rate, int damage, int magazine, double reload\_time, AmmoType ammo\_type) {

super(rare, skin, effective\_range, fire\_rate, damage, magazine, reload\_time, ammo\_type);

}

@Override

public void Attack() {

System.out.print("\*shooting with pistol: ");

FirearmMechanic(ammo\_type, effective\_range);

System.out.print("\*");

}

@Override

public void InspectWeapon() {

System.out.println("\*inspecting handgun\*");

}

};

class AssaultRifle extends Firearm {

@Override

protected void UpdateDueToRarity() {

switch (rare) {

case Standard:

break;

case Advanced:

damage += 5;

effective\_range += 13;

reload\_time = reload\_time - 0.3;

break;

case High\_End:

damage += 10;

effective\_range += 26;

reload\_time = reload\_time - 0.6;

break;

case Elite:

damage += 15;

effective\_range += 39;

reload\_time = reload\_time - 0.9;

break;

}

}

public AssaultRifle(Rarity rare, String skin, int effective\_range, int fire\_rate, int damage, int magazine, double reload\_time, AmmoType ammo\_type) {

super(rare, skin, effective\_range, fire\_rate, damage, magazine, reload\_time, ammo\_type);

}

@Override

public void Attack() {

System.out.print("\*automatic shooting with assault rifle: ");

FirearmMechanic(ammo\_type, effective\_range);

System.out.print("\*");

}

@Override

public void InspectWeapon() {

System.out.println("\*inspecting assault rifle\*");

}

}

Human.Java:

package com.company;

enum Rank {

DEPUTY,

SUPERVISOR,

SERGEANT,

LIEUTENANT

};

public class Human implements IHuman {

protected String name;

protected double weight;

protected double height;

protected int age;

public Human(String name, double weight, double height, int age) {

this.age = age;

this.height = height;

this.name = name;

this.weight = weight;

}

public Human(Human other) {

name = other.name;

weight = other.weight;

height = other.height;

age = other.age;

}

@Override

public String GetName() {

return name;

}

@Override

public double GetWeight() {

return weight;

}

@Override

public double GetHeight() {

return height;

}

@Override

public int GetAge() {

return age;

}

@Override

public void UseWeapon(Weapon weapon) {

System.out.print("Suspect using deadly force! ");

weapon.Attack();

}

public String toString() {

return ("Info: " + name + " Age " + age + " Weight " + weight + " Height " + height);

}

public void ExampleCallVirtualMethod() {

System.out.print(ExampleVirtualMethod());

}

private String ExampleVirtualMethod() {

return "Hello World";

}

};

class SWAT\_Operator extends Human {

private Rank rank;

public SWAT\_Operator(String name, double weight, double height, int age, Rank rank) {

super(name, weight, height, age);

this.rank = rank;

}

@Override

public void UseWeapon(Weapon weapon) {

System.out.print("Officer using deadly force! ");

weapon.Attack();

}

}

IHuman.Java:

package com.company;

enum Rank {

DEPUTY,

SUPERVISOR,

SERGEANT,

LIEUTENANT

};

public class Human implements IHuman {

protected String name;

protected double weight;

package com.company;

public interface IHuman {

public void UseWeapon(Weapon weapon);

public int GetAge();

public String GetName();

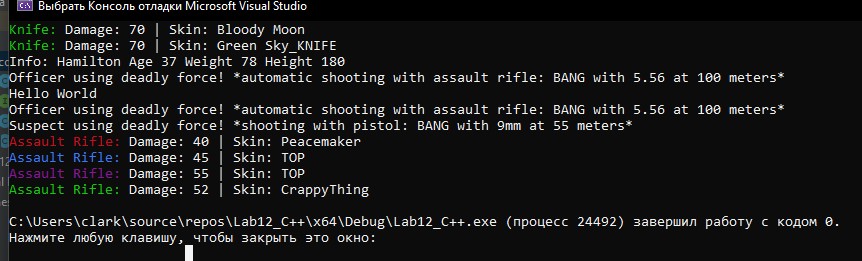
public double GetWeight();

public double GetHeight();

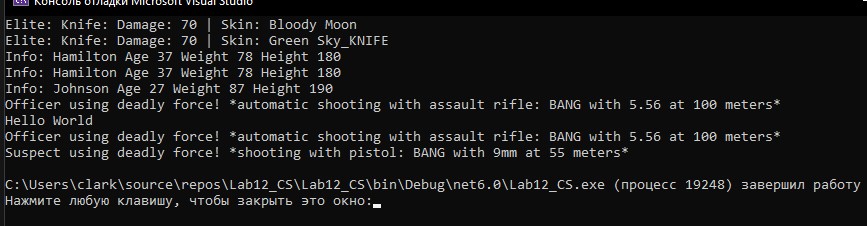
}

Результаты работы программы:

C++



C#



Java

