Міністерство освіти та науки України

Дніпровський державний технічний університет

### Кафедра програмного забезпечення систем

**Звіти до лабораторних робіт**

із дисципліни

„Моделювання та аналіз програмного забезпечення”

варіант №3

Розробив \_\_\_\_\_\_\_\_\_\_\_\_\_\_ ст. гр. ПЗ-19-1-ду

(підпис) Боюн Артур Валерійович

Перевірив \_\_\_\_\_\_\_\_\_\_\_\_\_\_ к.т.н, доцент каф. ПЗС (підпис) Ялова К.М.

Дати здавання:

ЛР1 \_\_\_\_\_\_\_\_\_\_\_\_\_

ЛР2 \_\_\_\_\_\_\_\_\_\_\_\_\_

ЛР3 \_\_\_\_\_\_\_\_\_\_\_\_\_

ЛР4 \_\_\_\_\_\_\_\_\_\_\_\_\_

ЛР5 \_\_\_\_\_\_\_\_\_\_\_\_\_

ЛР6 \_\_\_\_\_\_\_\_\_\_\_\_\_

ЛР7 \_\_\_\_\_\_\_\_\_\_\_\_\_

ЛР8 \_\_\_\_\_\_\_\_\_\_\_\_\_

Кам’янське

2020

***Лабараторна робота №1***

**Завдання**

Створити класи, що знаходяться у зв’язку наслідування. Показати реалізацію:

- перевантаження методів;

- приховання батьківської властивості

- перевизначення батьківського методу;

- приховання батьківського методу.

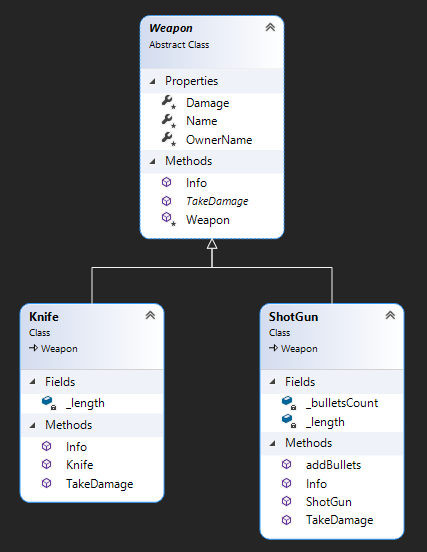
- створення абстрактного класу;

- створення абстрактного та віртуального методу.

При неможливості відтворити класи з курсової роботи з дисципліни «Об’єктно-орієнтоване програмування» необхідно обрати будь-яку ієрархію класів із визначеної предметної області.

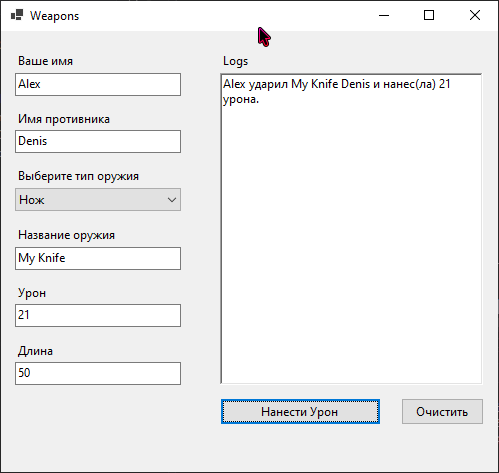
***Результати виконання завдання 1:***

*Діаграма класів додатку (рис.1):*

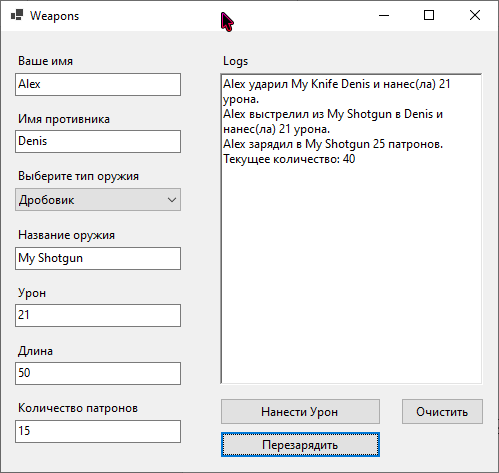


**Рис. 1 Діаграма класів додатку**

*Результати роботи додатку (рис.2, рис.3)*



**Рис. 2 Результат виконаня додатку**



**Рис. 3 Результат виконаня додатку**

*Лістинг програмного коду:*

using System;

using System.Windows.Forms;

namespace Lab\_1

{

abstract class Weapon

{

protected string Name { get; set; }

protected string OwnerName { get; set; }

protected double Damage { get; set; }

protected Weapon(string name, string ownerName, double damage)

{

Name = name;

OwnerName = ownerName;

Damage = damage;

}

public virtual string Info()

{

return $"Название: {Name}, Владелец: {OwnerName}, Урон: {Damage}";

}

public abstract string TakeDamage(string To);

}

class Knife : Weapon

{

private readonly double \_length;

public Knife(string name, string ownerName, double damage, double length) : base(name, ownerName, damage)

{

\_length = length;

}

public override string Info()

{

return $"Название: {Name}, Владелец: {OwnerName}, Урон: {Damage}, Длина: {\_length}";

}

public override string TakeDamage(string to)

{

return $"{OwnerName} ударил {Name} {to} и нанес(ла) {Damage} урона.";

}

}

class ShotGun : Weapon

{

private readonly double \_length;

private int \_bulletsCount;

public ShotGun(string name, string ownerName, double damage, double length, int bulletsCount) : base(name,

ownerName, damage)

{

\_length = length;

\_bulletsCount = bulletsCount;

}

public override string Info()

{

return $"Название: {Name}, Владелец: {OwnerName}, Урон: {Damage}, Длина: {\_length}";

}

public override string TakeDamage(string to)

{

if (\_bulletsCount > 0)

{

\_bulletsCount--;

return $"{OwnerName} выстрелил из {Name} в {to} и нанес(ла) {Damage} урона.";

}

return $"{OwnerName} не смог(ла) выстрелить из {Name} в {to}. Нет патронов!";

}

public string addBullets(int count)

{

\_bulletsCount += count;

return $"{OwnerName} зарядил в {Name} {count} патронов. Текущее количество: {\_bulletsCount}";

}

}

public partial class MainForm : Form

{

private Weapon \_weapon;

private string damageTextBoxLastText = "0";

private string lengthTextBoxLastText = "0";

private string bulletsCountTextBoxLastText = "15";

public MainForm()

{

InitializeComponent();

}

private void MainForm\_Load(object sender, EventArgs e)

{

weaponTypeBox.Items.Add("Нож");

weaponTypeBox.Items.Add("Дробовик");

weaponTypeBox.SelectedIndex = 0;

damageTextBox.Text = damageTextBoxLastText;

lengthTextBox.Text = lengthTextBoxLastText;

bulletsCountTextBox.Text = bulletsCountTextBoxLastText;

}

private void PrintToLogs(string msg)

{

logsTextBox.Text += msg;

logsTextBox.Text += '\n';

}

private void reloadWeaponBtn\_Click(object sender, EventArgs e)

{

if (weaponTypeBox.SelectedIndex == 1)

{

ShotGun \_\_weapon = new ShotGun(

weaponNameTextBox.Text,

nameTextBox.Text,

double.Parse(damageTextBox.Text),

double.Parse(lengthTextBox.Text),

int.Parse(bulletsCountTextBox.Text));

PrintToLogs(\_\_weapon.addBullets(25));

}

}

private void takeDamageBtn\_Click(object sender, EventArgs e)

{

#region Checks

if (nameTextBox.Text.Length <= 0)

{

MessageBox.Show("Вы не ввели ваше имя.");

return;

}

if (enemyNameTextBox.Text.Length <= 0)

{

MessageBox.Show("Вы не ввели имя противника.");

return;

}

if (weaponNameTextBox.Text.Length <= 0)

{

MessageBox.Show("Вы не ввели название оружия.");

return;

}

#endregion

switch (weaponTypeBox.SelectedIndex)

{

case 0:

\_weapon = new Knife(

weaponNameTextBox.Text,

nameTextBox.Text,

double.Parse(damageTextBox.Text),

double.Parse(lengthTextBox.Text));

break;

case 1:

\_weapon = new ShotGun(

weaponNameTextBox.Text,

nameTextBox.Text,

double.Parse(damageTextBox.Text),

double.Parse(lengthTextBox.Text),

int.Parse(bulletsCountTextBox.Text));

break;

}

PrintToLogs(\_weapon.TakeDamage(enemyNameTextBox.Text));

}

private void clearLogsBtn\_Click(object sender, EventArgs e)

{

logsTextBox.Text = "";

}

private void weaponTypeBox\_SelectedIndexChanged(object sender, EventArgs e)

{

switch (weaponTypeBox.SelectedIndex)

{

case 0:

label6.Hide();

bulletsCountTextBox.Hide();

reloadWeaponBtn.Hide();

break;

case 1:

label6.Show();

bulletsCountTextBox.Show();

reloadWeaponBtn.Show();

break;

}

}

private void danageTextBox\_TextChanged(object sender, EventArgs e)

{

if (double.TryParse(damageTextBox.Text, out \_))

{

damageTextBoxLastText = damageTextBox.Text;

}

else

{

damageTextBox.Text = damageTextBoxLastText;

}

}

private void lenghtTextBox\_TextChanged(object sender, EventArgs e)

{

if (double.TryParse(lengthTextBox.Text, out \_))

{

lengthTextBoxLastText = lengthTextBox.Text;

}

else

{

lengthTextBox.Text = lengthTextBoxLastText;

}

}

private void bulletsCountTextBox\_TextChanged(object sender, EventArgs e)

{

if (int.TryParse(bulletsCountTextBox.Text, out \_))

{

bulletsCountTextBoxLastText = bulletsCountTextBox.Text;

}

else

{

bulletsCountTextBox.Text = bulletsCountTextBoxLastText;

}

}

private void weaponNameTextBox\_TextChanged(object sender, EventArgs e)

{

if (weaponNameTextBox.Text.Length > 0)

{

damageTextBox.Enabled = true;

lengthTextBox.Enabled = true;

bulletsCountTextBox.Enabled = true;

takeDamageBtn.Enabled = true;

reloadWeaponBtn.Enabled = true;

}

else

{

damageTextBox.Enabled = false;

lengthTextBox.Enabled = false;

bulletsCountTextBox.Enabled = false;

takeDamageBtn.Enabled = false;

reloadWeaponBtn.Enabled = false;

}

}

}

}