Міністерство освіти і науки України
Національний технічний університет України «КПІ ім. Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

З лабораторної роботи №4

з курсу

«Основи розробки програмного забезпечення на платформі Microsoft.NET»

Виконав студент *ІП-01 Галько Міла*
(шифр, ПІБ)

Перевірила *Ліщук К.І.*
(ПІБ)

Київ 2022

Комп‘ютерний практикум № 4. Шаблони проектування. Структурні шаблони

Мета:

- ознайомитися з основними шаблонами проектування, навчитися застосовувати їх при проектуванні і розробці ПЗ.

Постановка задачі комп‘ютерного практикуму № 4

При виконанні комп‘ютерного практикуму необхідно виконати наступні дії:
1) Вивчити структурні патерни. Знати загальну характеристику та призначення кожного з них, особливості реалізації кожного зі структурних патернів та випадки їх застосування.
2) Реалізувати задачу згідно варіанту №1. Розробити інтерфейси та класи з застосування одного або декількох патернів. Повністю реалізувати методи, пов‘язані з реалізацією обраного патерну.
3) Повністю описати архітектуру проекту (призначення методів та класів), особливості реалізації обраного патерну. Для кожного патерну необхідно вказати основні класи та їх призначення,
4) Навести UML-діаграму класів

Варіант №1

Розробити структуру організації армії в грі фентезі. Армія може складатися з загонів ельфів, орків, мінотаврів, кентаврів, циклопів, драконів, гідр, лицарів. Армія може містити як загони, так і одиночних воїнів, загін може складатися з інших загонів і одиночних воїнів.

Програмний код:

Посилання на github: *https://github.com/MilaHalko/C4_.NET/tree/Lab4*

Також код доданий в кінці документу.

Обирання патерну

Відповідно до завдання, розуміємо, що маємо конкретні об’єкти по типу Орків, Ельфів і т.д. Самі ж вони можуть утворювати загони, а загони – армію. Все це нагадує структуру дерева, де конкретні представники воїнів є листям. Отже можемо застосувати патерн Composite. Таким чином Клієнт отримає змогу як працювати з окремими частинами дерева, так і цілком з ним. У нашому випадку загін – це контейнер. Армія – це той самий контейнер, корінь дерева.

Архітектура проекту

Отже у загальному абстрактному класі Unit визначимо атрибут «ім’я» та абстрактні методи для додавання/видалення об’єктів. Додатково зазначимо абстрактні методи для визначення сили армії, її розмірів, кількості воїнів, та потреби у їжі.

Від Unit успадкуємо клас Troop – контейнер. Обов’язковим атрибутом буде список з Unit-ів для можливості додавання як нових контейнерів, так і воїнів. Щодо методів – усі вони реалізуються відповідно до задачі.

Також від Unit успадковуємо абстрактний клас Warrior. Даний клас реалізує декілька абстрактних методів, де логіка для усіх майбутніх спадкоємців буде однаковою. Наприклад методи додавання та видалення згідно патерну не повинні бути реалізовані у листях (повідомлення про помилку). Отже для кожного воїна не має потреби писати одне й те саме багато разів. Легше 1 раз зазначити це у базовому класі.

Останнім кроком є додання спадкоємців Warrior, конкретних воїнів. У них визначаємо усі інші нереалізовані методи.

Опис класів і методів

1) **Abstract class Unit** – Компонент, загальний інтерфейс для роботи з компонентами дерева.
2) **Class Troop** – Контейнер, містить дочірні елементи дерева.

Реалізовані методи у Troop:

a. ***Add/Remove*** – додавання та видалення об'єктів із вмісту контейнера.
b. ***GetForce/GetSize/GetFoodNeeds/GetQuantity/Display*** – методи що виконуються до усього вмісту контейнера переадресовуючи основну роботу своїм компонентам. Відповідно до переліку методи повертають загальну суму сили, розміру, потреб у їжі, кількості чи простий вивід воїнів.

3) **Abstract class Warrior** – загальний клас воїнів, реалізуючий декілька методів, де логіка для спадкоємців однакова.

Реалізовані методи у Warrior:

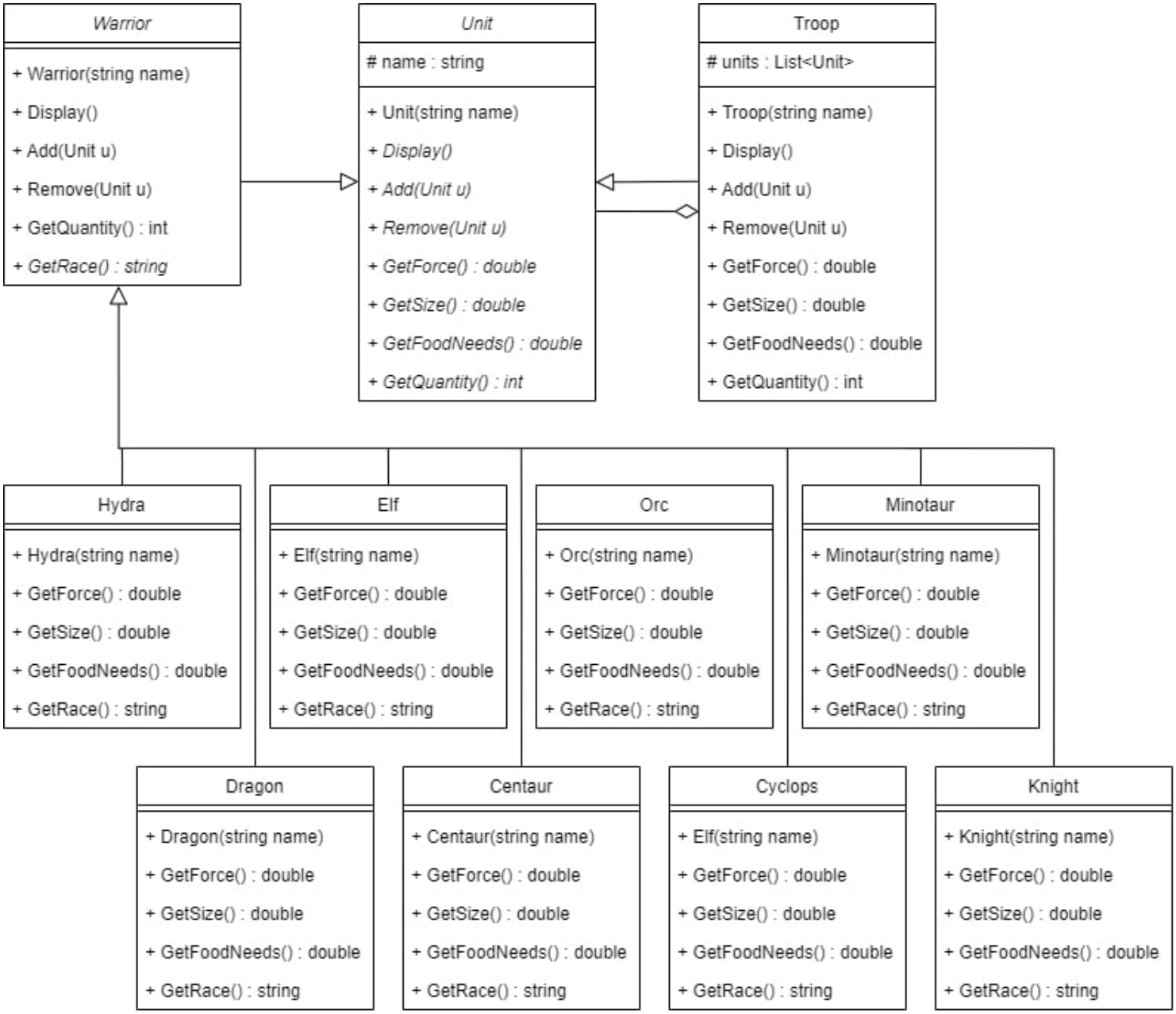
a. ***Add/Remove*** – повертає помилку, оскільки у воїн не може бути контейнером.
b. ***GetQuantity*** – повертає 1.
c. ***Display*** – вивід воїна та його характеристик.

4) **Класи: Elf, Orc, Hydra, Dragon, Knight, Cyclops, Centaur, Minotaur** – конкретна реалізація воїнів.

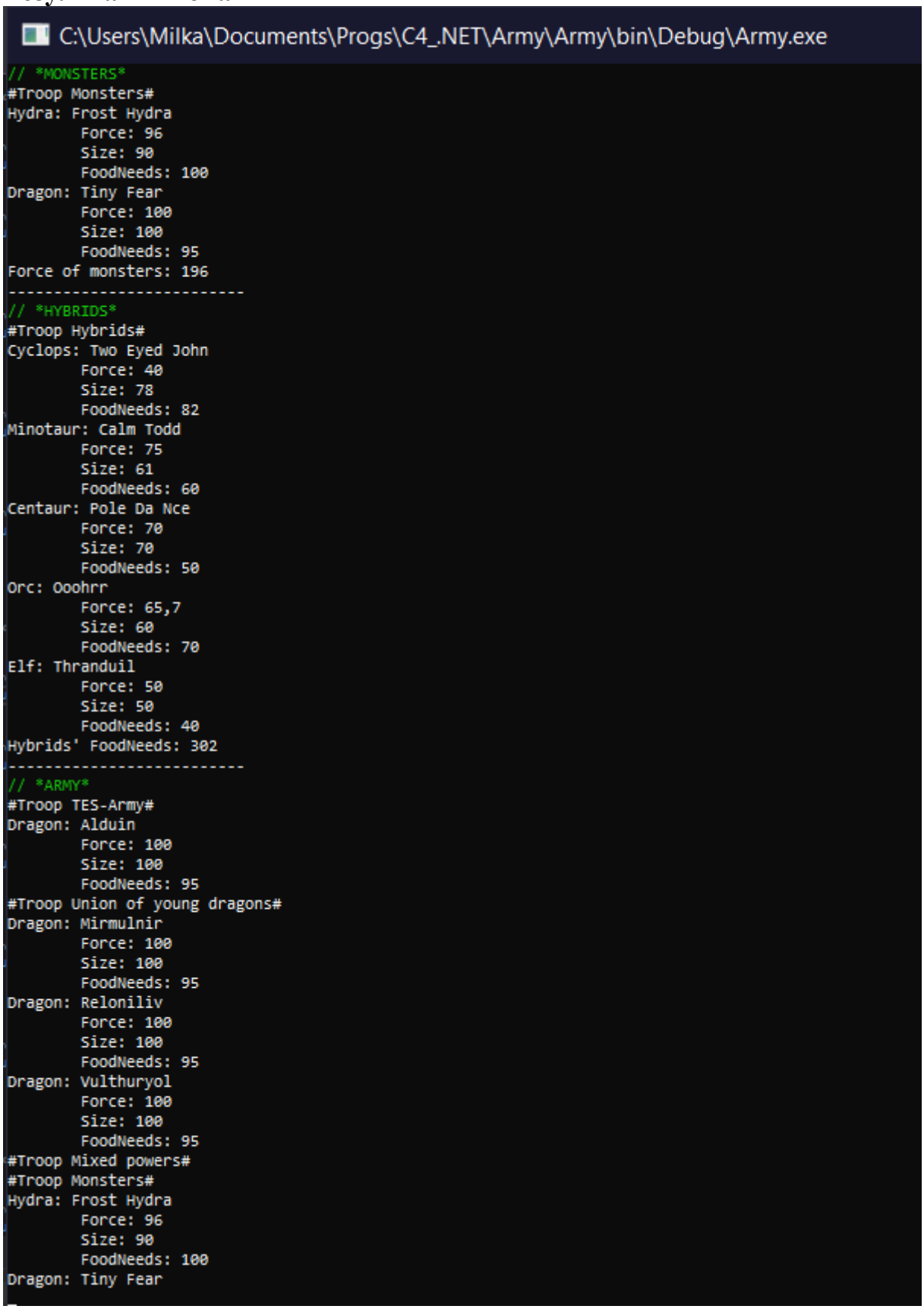
Реалізовані методи у спадкоємцях Warrior:

a. ***GetForce/GetSize/GetFoodNeeds/GetRace*** – в залежності від воїна метод повертає певне значення, що еквівалентно силі, розміру, потреби їжі чи расі.

UML діаграма



Результат виконання



Вміст Program.cs:

```
using System;
using System.Collections.Generic;
using Army.Files;

namespace Army
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Unit army = new Troop("TES-Army");
            army.Add(new Dragon("Alduin"));

            Unit dragonUnion = new Troop("Union of young dragons");
            dragonUnion.Add(new Dragon("Mirmulnir"));
            dragonUnion.Add(new Dragon("Reloniliv"));
            dragonUnion.Add(new Dragon("Vulthuryol"));

            var knight = new Knight("Sir Davion");
            dragonUnion.Add(knight);
            dragonUnion.Remove(knight);

            army.Add(dragonUnion);

            Unit mainForces = new Troop("Mixed powers");
            Unit monsters = new Troop("Monsters");
            Unit hybrids = new Troop("Hybrids");
            Unit people = new Troop("People");

            army.Add(mainForces);
            mainForces.Add(monsters);
            mainForces.Add(hybrids);
            mainForces.Add(people);

            monsters.Add(new Hydra("Frost Hydra"));
            monsters.Add(new Dragon("Tiny Fear"));

            hybrids.Add(new Cyclops("Two Eyed John"));
            hybrids.Add(new Minotaur("Calm Todd"));
            hybrids.Add(new Centaur("Pole Da Nce"));
            hybrids.Add(new Orc("Ooohrr"));
            hybrids.Add(new Elf("Thranduil"));

            people.Add(knight);

            Console.ForegroundColor = ConsoleColor.Green;
            Console.WriteLine("// *MONSTERS*");

            Console.ForegroundColor = ConsoleColor.White;
            monsters.Display();
            Console.WriteLine($"Force of monsters: {monsters.GetForce()}");
            Console.WriteLine("-------------------------");


            Console.ForegroundColor = ConsoleColor.Green;
            Console.WriteLine("// *HYBRIDS*");

            Console.ForegroundColor = ConsoleColor.White;
            hybrids.Display();
            Console.WriteLine($"Hybrids' FoodNeeds: {hybrids.GetFoodNeeds()}");
            Console.WriteLine("-------------------------");


            Console.ForegroundColor = ConsoleColor.Green;
            Console.WriteLine("// *ARMY*");

            Console.ForegroundColor = ConsoleColor.White;
            army.Display();
            Console.WriteLine($"Quantity of Army units: {army.GetQuantity()}");
            Console.WriteLine($"Space by Army: {army.GetSize()}");
            Console.WriteLine("-------------------------");

            Console.ReadLine();
        }
    }
}
```

Вміст Unit.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Army.Files
{
    internal abstract class Unit
    {
        protected string name;

        public Unit(string name)
        {
            this.name = name;
        }
        public abstract void Display();
        public abstract void Add(Unit u);
        public abstract void Remove(Unit u);
        public abstract double GetForce();
        public abstract double GetSize();
        public abstract double GetFoodNeeds();
        public abstract int GetQuantity();
    }
}
```

Вміст Troop.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Army.Files
{
    class Troop : Unit
    {
        protected List<Unit> units = new List<Unit>();

        public Troop(string name) : base(name)
        { }

        public override void Add(Unit u)
        {
            units.Add(u);
        }

        public override void Remove(Unit u)
        {
            units.Remove(u);
        }

        public override double GetForce()
        {
            double force = 0;
            foreach (Unit u in units)
            {
                force += u.GetForce();
            }
            return force;
        }

        public override double GetSize()
        {
            double size = 0;
            foreach (Unit u in units)
            {
                size += u.GetSize();
            }
            return size;
        }
        public override double GetFoodNeeds()
        {
            double fn = 0;
            foreach (Unit u in units)
```

```
            {
                fn += u.GetFoodNeeds();
            }
            return fn;
        }
        public override int GetQuantity()
        {
            int q = 0;
            foreach (var u in units)
            {
                q += u.GetQuantity();
            }
            return q;
        }

        public override void Display()
        {
            Console.WriteLine($"#Troop {name}#");
            foreach (var u in units)
            {
                u.Display();
            }
        }

    }
}
```

Вміст Warrior.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Army.Files
{
    internal abstract class Warrior : Unit
    {
        protected Warrior(string name) : base(name)
        {
        }

        public override void Display()
        {
            Console.WriteLine($"{GetRace()}: {name}\n" +
                $"\tForce: {GetForce()}\n" +
                $"\tSize: {GetSize()}\n" +
                $"\tFoodNeeds: {GetFoodNeeds()}");
        }
        public override void Add(Unit u)
        {
            throw new NotImplementedException();
        }

        public override void Remove(Unit u)
        {
            throw new NotImplementedException();
        }

        public override int GetQuantity()
        {
            return 1;
        }

        public abstract string GetRace();
    }

    class Elf : Warrior
    {
        public Elf(string name)
            : base(name)
        { }
        public override double GetForce()
        {
            return 50;
        }
```

```
        public override double GetSize()
        {
            return 50;
        }

        public override double GetFoodNeeds()
        {
            return 40;
        }

        public override string GetRace()
        {
            return "Elf";
        }
    }

    class Orc : Warrior
    {
        public Orc(string name)
            : base(name)
        { }
        public override double GetForce()
        {
            return 65.7;
        }

        public override double GetSize()
        {
            return 60;
        }

        public override double GetFoodNeeds()
        {
            return 70;
        }

        public override string GetRace()
        {
            return "Orc";
        }
    }

    class Minotaur : Warrior
    {
        public Minotaur(string name)
            : base(name)
        { }
        public override double GetForce()
        {
            return 75;
        }

        public override double GetSize()
        {
            return 61;
        }

        public override double GetFoodNeeds()
        {
            return 60;
        }

        public override string GetRace()
        {
            return "Minotaur";
        }
    }

    class Centaur : Warrior
    {
        public Centaur(string name)
            : base(name)
        { }
        public override double GetForce()
        {
            return 70;
        }
```

```
        public override double GetSize()
        {
            return 70;
        }

        public override double GetFoodNeeds()
        {
            return 50;
        }

        public override string GetRace()
        {
            return "Centaur";
        }
    }


    class Cyclops: Warrior
    {
        public Cyclops(string name)
            : base(name)
        { }
        public override double GetForce()
        {
            return 40;
        }

        public override double GetSize()
        {
            return 78;
        }

        public override double GetFoodNeeds()
        {
            return 82;
        }

        public override string GetRace()
        {
            return "Cyclops";
        }
    }

    class Dragon : Warrior
    {
        public Dragon(string name)
            : base(name)
        { }
        public override double GetForce()
        {
            return 100;
        }

        public override double GetSize()
        {
            return 100;
        }

        public override double GetFoodNeeds()
        {
            return 95;
        }

        public override string GetRace()
        {
            return "Dragon";
        }
    }

    class Hydra : Warrior
    {
        public Hydra(string name)
            : base(name)
        { }
        public override double GetForce()
        {
            return 96;
        }
```

```
        public override double GetSize()
        {
            return 90;
        }

        public override double GetFoodNeeds()
        {
            return 100;
        }

        public override string GetRace()
        {
            return "Hydra";
        }
    }

    class Knight : Warrior
    {
        public Knight(string name)
            : base(name)
        { }
        public override double GetForce()
        {
            return 30;
        }

        public override double GetSize()
        {
            return 30;
        }

        public override double GetFoodNeeds()
        {
            return 25;
        }

        public override string GetRace()
        {
            return "Knight";
        }
    }

}
```