НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КПІ ім. Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра програмного забезпечення комп’ютерних систем

ЛАБОРАТОРНА РОБОТА № N

з дисципліни

««C# .Net. Реалізація основних принципів ООП мовою C#»»

ТЕМА:

Підготував: студент групи КП-52

Крисюк Андрій Михайлович

Перевірила:

Заболотня Тетяна Миколаївна

Київ 2016

Завдання на лабораторну роботу №1.

«C# .Net. Реалізація основних принципів ООП мовою C#»

Мета роботи: Ознайомитися з основами об’єктного підходу у мові С#, створенням класів, об’єктів, механізмами інкапсуляції, наслідування та поліморфізму.

Постановка задачі:

Побудувати ієрархію класів що відтворюватимуть відношення наслідування між об’єктами реального світу. (Кількість класів >= 5). При цьому забезпечити реалізацію таких механізмів ООП:

1. Наявність у класах статичних даних. Забезпечити коректну ініціалізацію цих даних.  
2. Наявність у класах змінних та методів з різними модифікаторами доступу.  
3. Наявність у класах властивостей.   
4. Конструктори. Створити для розроблюваних класів такі конструктори:  
- конструктор за замовчанням;   
- перевизначений конструктор;   
- приватний конструктор;   
- статичний конструктор.   
Продемонструвати, яким чином викликаються конструктори базового та дочірнього класів.   
5. Використати віртуальні та перевизначені методи.   
6. Продемонструвати поліморфізм методів.

Протокол має містити діаграму класів + код програми має містити коментарі.

Код програми

class Program

{

public abstract class Hero

{

protected string name;

protected int health;

protected int shield;

protected int damage;

protected Hero() { }

protected Hero(string name, int health, int shield)

{

this.name = name;

this.health = health;

this.shield = shield;

}

public int Damage

{

get { return damage; }

set { damage = value; }

}

public int Health

{

get { return health; }

set { health = value; }

}

public int Shield

{

get { return shield; }

set { shield = value; }

}

public string Name

{

get { return name; }

}

public abstract void attack(Hero enemy);

protected bool isAlive()

{

if (health > 0)

return true;

else

return false;

}

protected void kill(Hero enemy)

{

enemy.Health = 0;

Console.WriteLine(this.Name + " killed " + enemy.Name);

return;

}

}

public class Boss : Hero

{

private static Boss reaper;

private Boss()

{

name = "The Reaper";

health = 10000;

shield = 10000;

damage = 10000;

}

public static Boss createBoss()

{

reaper = new Boss();

return reaper;

}

public override void attack(Hero enemy)

{

this.kill(enemy);

}

}

public class Soldier : Hero

{

private static double accuracy; //parameter that affects the damage

static Soldier()

{

accuracy = 0.89;

}

public Soldier()

{

name = "Jhon Shepard";

health = 100;

damage = (int)(50 \* accuracy);

shield = 50;

}

public Soldier(string name, int health, int shield) : base(name, health, shield)

{

this.damage = (int)(50 \* accuracy);

}

/\* Soldier can't break enemies shield, if him damage is less then points of shield.

To break shield he need to use another skill \*/

public override void attack(Hero enemy)

{

if(!this.isAlive())

{

Console.WriteLine(this.Name + " is dead. He can't attack");

return;

}

if (enemy.isAlive())

{

if (enemy.Shield > this.damage)

{

Console.WriteLine(this.Name + " can't break through the shields of " + enemy.Name);

return;

}

else if (enemy.Shield <= this.damage && enemy.Shield > 0)

{

//if damage more then shield, some part of damage passes to health

int curr\_damage = damage - enemy.Shield;

enemy.Shield = 0;

enemy.Health -= curr\_damage;

}

else

{

if (enemy.Health > damage)

enemy.Health -= damage;

else

{

this.kill(enemy);

return;

}

}

Console.WriteLine(enemy.Name + " got " + damage + " damage from " + this.Name); //log damage

}

else

{

Console.WriteLine(enemy.Name + " has been already dead");

return;

}

}

/\* This skill allow soldier kill every enemy, but i must limit it in next lab\*/

public void aimedShot(Hero enemy)

{

if (!this.isAlive())

{

Console.WriteLine(this.Name + " is dead. He can't attack");

return;

}

if (enemy.isAlive())

{

enemy.Health = 0;

Console.WriteLine(enemy.Name + " was killed by aimed shot from " + this.Name);

}

else

{

Console.WriteLine(enemy.Name + " has been already dead");

return;

}

}

/\* Inspiration - special skill of Soldier, that increase accuracy of all Soldiers, that will be create,

\* and doubles damage of Soldier, whom used this skill\*/

public void inspiration()

{

accuracy = 1;

this.damage \*= 2;

}

}

public class Adept : Hero

{

protected int energy = 100; //energy allows Adept and Medic to use their skills

public int Energy

{

get { return energy; }

set { energy = value; }

}

public Adept()

{

name = "Kaiden Alenko";

health = 75;

damage = 30;

shield = 75;

}

public Adept(string name, int health, int shield) : base(name, health, shield)

{

this.damage = 30;

}

/\*Adept attacks directly health, avoiding the shields\*/

public override void attack(Hero enemy)

{

if (!this.isAlive())

{

Console.WriteLine(this.Name + " is dead. He can't attack");

return;

}

if (enemy.isAlive())

{

if (enemy.Health > damage)

enemy.Health -= damage;

else

{

this.kill(enemy);

return;

}

Console.WriteLine(enemy.Name + " got " + damage + " damage from " + this.Name); //log damage

}

else

{

Console.WriteLine(enemy.Name + " has been already dead");

return;

}

}

/\* singularity attacks every enemy, that specified in params and dicrease health of each enemy by half.

If health less than 20, killed enemy. Consuming energy \*/

public virtual void singularity(params Hero[] enemies)

{

int consuming = 30;

if (this.isAlive() && this.Energy > consuming)

{

this.energy -= consuming;

for (int i = 0; i < enemies.Length; i++)

{

if (enemies[i].isAlive())

{

if (enemies[i].Health <= 20)

this.kill(enemies[i]);

else

enemies[i].Health /= 2;

}

}

}

}

}

public class Engineer : Hero

{

public Engineer()

{

name = "Garus Vakarian";

health = 75;

damage = 35;

shield = 150;

}

public Engineer(string name, int health, int shield) : base(name, health, shield)

{

this.damage = 35;

}

/\* engineer have double damage against shields, and standart damage against health\*/

public override void attack(Hero enemy)

{

if (!this.isAlive())

{

Console.WriteLine(this.Name + " is dead. He can't attack");

return;

}

if (enemy.isAlive())

{

if (enemy.Shield > 2 \* this.damage)

{

enemy.Shield -= 2 \* this.damage;

Console.WriteLine(enemy.Name + " got " + 2 \* this.damage + " damage from " + this.Name); //log double damage

return;

}

else if (enemy.Shield <= 2 \* this.damage && enemy.Shield > 0)

{

int curr\_damage = (2 \* damage - enemy.Shield)/2; //but standart damage for health

if(enemy.Health < curr\_damage)

{

this.kill(enemy);

return;

}

else

enemy.Health -= curr\_damage;

}

else

{

if (enemy.Health > damage)

enemy.Health -= damage;

else

{

this.kill(enemy);

return;

}

}

Console.WriteLine(enemy.Name + " got " + damage + " damage from " + this.Name); //log standart damage

}

else

{

Console.WriteLine(enemy.Name + " has been already dead");

return;

}

}

/\*Engineers skill is overload. Overload breaks shields of enemies, if they quantity less or equal 3,

\* and reduces shields by 3/4 if quantity of enemies more then 3 \*/

public void overload(params Hero[] enemies)

{

if (this.isAlive())

{

if (enemies.Length > 3)

{

for (int i = 0; i < enemies.Length; i++)

{

if(enemies[i].isAlive())

enemies[i].Shield /= 4;

}

}

else

{

for (int i = 0; i < enemies.Length; i++)

{

if (enemies[i].isAlive())

enemies[i].Shield = 0;

}

}

}

}

}

public class Medic : Adept

{

public Medic()

{

name = "Doctor Chaves";

health = 50;

shield = 50;

damage = 15;

energy = 200;

}

public Medic(string name) : this()

{

this.name = name;

}

public Medic(string name, int health, int shield) : base(name, health, shield)

{

damage = 15;

}

/\*Medic can heal himself, but spend energy for this\*/

public void heal()

{

int consuming = 40;

if (energy > consuming)

{

if (this.isAlive())

{

energy -= consuming;

this.Health += 100;

}

}

}

/\*Also medic can heal another Hero or resurrect him, but heal less health\*. Spend energy too\*/

public void heal(Hero ally)

{

int consuming = 40;

if (this.isAlive() && energy > consuming)

{

if (!ally.isAlive())

ally.Health = 50;

else

ally.Health += 75;

energy -= consuming;

}

}

public override void singularity(params Hero[] enemies)

{

int consuming = 70;

if (this.isAlive() && this.Energy >= consuming)

{

this.energy -= consuming;

for (int i = 0; i < enemies.Length; i++)

{

if (enemies[i].isAlive())

this.kill(enemies[i]);

}

}

}

}

