Отчет по лабораторной работе № 2

“Исследование механизма простого наследования”

по дисциплине

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Выполнил студент гр. ИВТ/б-22о

Горбенко К.Н.

Проверил:

Тимофеев И.С.

1. **Цель работы:** исследование основных средств создания базового и порожденного классов. Исследование особенностей вызова методов порожденного класса при простом наследовании.
2. **Задание на работу**: описать заданный по варианту базовый класс минимум с одним информационным полем. Для базового класса определить два конструктора, а также два метода: метод ввода данных и метод, указанный в варианте задания.  
    Описать класс-наследник, расширив его функциональность относительно базового класса в соответствии с вариантом задания – добавив поля и методы, свойственные этому классу. Определить в классе конструкторы, функции ввода данных и вывода данных на экран. Переопределить один из методов базового класса в классе-наследнике.  
    Создать объекты базового и производного классов, ввести необходимые данные. Для порожденного объекта осуществить вызов переопределенного метода и вызов этого же метода из базового класса.  
    Условие для **варианта № 8**:

Базовый класс Юнит с методом «Печать информации» и такими полями: Название, Единиц жизни, Урон.  
 Производный класс Лучник с дополнительным полем Радиус атаки. Переопределить для него функцию Печать так, чтобы она выводила данные о Лучнике на экран. Описать в классе дополнительную функцию Выживание, проверяющую количество жизни и печатающую сообщение, жив ли Лучник или уже нет.

1. **Код программы на языке C#:** Класс Unit: public class Unit  
    {  
    public string Name { get; }  
    public int Health { get; private set; }  
    public int Damage { get; private set; }  
     
    public Unit(string name)  
    {  
    if (String.IsNullOrWhiteSpace(name)) throw new ArgumentNullException(nameof(name));  
     
    Name = name;  
    Health = 100;  
    Damage = 10;  
    }  
     
    public Unit(string name, int health, int damage)  
    {  
    if (String.IsNullOrWhiteSpace(name)) throw new ArgumentNullException(nameof(name));  
     
    Name = name;  
    Health = health;  
    Damage = damage;  
    }  
     
    public virtual string Status()  
    {  
    return $"Unit name:{Name};health:{Health};damage:{Damage}";  
    }  
    }

Класс Archer:

|  |
| --- |
| public class Archer : Unit  {  public int Range { get; private set; }   public Archer(string name) : base(name)  {  Range = 10;  }   public Archer(string name, int health, int damage, int range) : base(name, health, damage)   {  Range = range;  }   public override string Status()  {  return $"Archer name:{Name};health:{Health};damage:{Damage};range:{Range}";  }   public bool IsAlive()  {  return Health > 0;  }  } |

Класс Program:

|  |
| --- |
| class Program  {  static void Main()  {  Console.WriteLine("Enter unit's name");  var unit = new Unit(Console.ReadLine());   Console.WriteLine("Enter archer's name");  var archer = new Archer(Console.ReadLine());   Console.WriteLine($"Unit's properties:\n" +  $"Name = {unit.Name}\n" +   $"Health = {unit.Health}\n" +  $"Damage = {unit.Damage}\n" +   $"Status = {unit.Status()}\n");   Console.WriteLine($"Archer's properties:\n" +  $"Name = {archer.Name}\n" +  $"Health = {archer.Health}\n" +  $"Damage = {archer.Damage}\n" +   $"Range = {archer.Range}\n" +   $"Status = {archer.Status()}\n" +   $"Is alive = {archer.IsAlive()}");   Console.ReadLine();  }  } |

1. **Тестовые примеры:**

Программа считывает имена юнита и лучника с клавиатуры. Затем Конструкторы соответствующих классов создают объекты классов со стандартными параметрами. Свойства Name, Health, Damage наследуются классом лучника, а метод Status переопределяется им. Кроме того, в производном классе был описан метод IsAlive(), не переопределяющий метод базового класса. Работа программы:

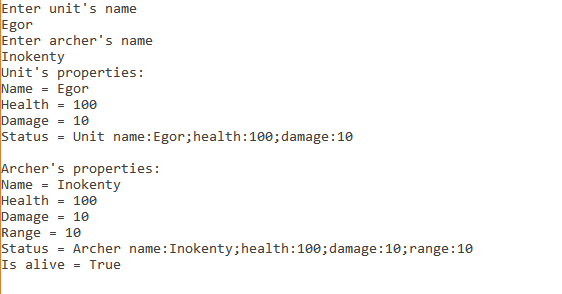


Рис. 1 - Результат работы программы

1. **Вывод:** в ходе лабораторной работы были изучены базовые механизмы наследования языков C# и C++.