Отчет по лабораторной работе № 3

“Исследование механизма виртуальных функций”

по дисциплине

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Выполнил студент гр. ИВТ/б-22о

Горбенко К.Н.

Проверил:

Тимофеев И.С.

1. **Цель работы:** исследование основных средств описания виртуальных функций и использование их при написании объектно-ориентированных программ.
2. **Задание на работу**: для варианта задания, полученного в лабораторной работе №2, необходимо выполнить следующее:  
    Переопределить базовый класс как абстрактный - для этого определить заданную вариантом функцию как чисто виртуальную. В обязательном порядке переопределить эту функцию в производном классе.  
    Дополнительно определить еще один производный класс со своими уникальными полями, переопределить в нем чисто виртуальный метод.  
    Создать указатель на базовый класс. Под указатель поместить объект производного класса. Проиллюстрировать работу программы с объектом через указатель - ввод/вывод данных, вызов функций класса.   
    Проиллюстрировать корректную работу механизма виртуальных функций - вызвать виртуальную функцию и убедиться, что тип выполняемой функции соответствует типу объекта под указателем. Поменять объект под указателем на объект второго производного класса. Проверить корректную работу механизма.  
    Описать деструкторы классов, содержащие тестовые фразы для вывода на экран. Отследить во время выполнения программы работу деструкторов по тестовым фразам — место вызова, порядок выполнения. Определить работу деструкторов с использованием виртуальных методов. Отследить изменения в работе программы.  
    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 С#** Класс Unit:

|  |
| --- |
| public abstract class Unit  {  public string Name { get; private set; }   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 bool IsAlive()  {  return Health > 0;   }   public virtual string Status()  {  return $"name:{Name};health:{Health};damage:{Damage};alive:{IsAlive()};";  }  } |

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 base.Status() + $"range:{Range};";  }  } |

Класс Swordsman:

|  |
| --- |
| public class Swordsman : Unit  {  public int Armor { get; private set; }    public Swordsman(string name) : base(name)  {  Armor = 0;  }   public Swordsman(string name, int health, int damage, int armor) : base(name, health, damage)  {  Armor = armor;  }   public override string Status()  {  return base.Status() + $"armor:{Armor};";  }  } |

Класс ShieldBearer

|  |
| --- |
| class ShieldBearer : Swordsman  {  public int Defence { get; set; }   public ShieldBearer(string name) : base(name)  {  Defence = 5;  }   public ShieldBearer(string name, int health, int damage, int armor, int defence) : base(name, health, damage, armor)  {  Defence = defence;  }   public override string Status()  {  return base.Status() + $"defence:{Defence};";  }  } |

Класс Program:

|  |
| --- |
| class Program  {  static void Main()  {  WriteLine("Enter archer's name:");  Unit person1 = new Archer(ReadLine());  WriteLine(person1.Status());   WriteLine("Enter swordsman's name:");  Unit person2 = new Swordsman(ReadLine());  WriteLine(person2.Status());   WriteLine("Enter shield bearer's name:");  Unit person3 = new ShieldBearer(ReadLine());  WriteLine(person3.Status());   WriteLine("Enter another shield bearer's name");  Swordsman person4 = new ShieldBearer(ReadLine());  WriteLine(person4.Status());   WriteLine("Press enter to exit");   ReadLine();  }  } |

1. **Описание логики программы** Базовый класс Unit был объявлен абстрактным (создавать объект этого класса запрещается), метод Status() - виртуальным. Таким образом, каждому классу, наследуемому от класса Unit, необходимо переопределять метод Status().   
    Метод IsAlive() наследуется каждым классом-наследником от класса Unit.  
    Поля Name, Health, Damage объявлены в базовом классе и наследуются каждым из классов-наследников.  
    Каждый из классов Archer и Swordsman содержит по дополнительному полю поверх полей, наследуемых от класса Unit.  
    Конструкторы классов-наследников не инициализируют базовые поля, а вызывают для этого базовые конструкторы. Таким образом, каждому классу-наследнику необходимо в конструкторе инициализировать только поля, объявленные внутри этого класса.
2. **Описание программы** Программа создает объекты классов наследников и помещает ссылки на них в переменные базового класса (обратная операция недопустима). При вызове метода Status() для каждой переменной (всего их 4), вызывается метод, переопределенный в соответствующем классе-наследнике.
3. **Вывод:** в ходе лабораторной работы № 3 был изучен механизм виртуальных функций. Для этого был создан абстрактный класс с виртуальным методом, который переопределялся в каждом из классов-наследников. В результате для каждого класса вызывался соответствующий переопределенный метод.