МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Санкт-Петербургский национальный исследовательский университет

информационных технологий, механики и оптики»

Факультет информационных технологий и программирования

Кафедра информационных систем

Лабораторная работа № 2

Создание базовых сущностей движка

Выполнил студент группы М3203:

Грохотов Владислав Геннадиевич

САНКТ-ПЕТЕРБУРГ

2019

**Задание:**

Необходимо определить базовые классы для функционирования движка. Во всех классах должна быть защита от прямого доступа к полям! Модификация полей только через методы или свойства.

Определить класс **Unit** со следующими полями:

* Type - уникальный идентификатор типа юнита, пока можно использовать строки или числа (впоследствии переделаем на перечисление)
* HitPoints - количество жизней юнита (здоровье)
* Attack - показатель атаки юнита (будет использоваться при определении наносимого урона)
* Defence - показатель защиты юнита (будет использоваться при определении полученного юнитом урона)
* Damage - диапазон значений (от-до), определяющий разброс базового урона юнита.
* Initiative - дробное значение, будет использоваться при определении приоритета хода юнита в каждом раунде

Значения всех полей заполняются через конструктор. Этот класс является описателем, то есть не должен модифицироваться в ходе работы программы, вся информация доступна только на чтение.

Определить класс **UnitsStack**, который содержит следующее:

* класс юнита
* количество юнитов

Значения всех полей заполняются через конструктор. Этот класс является описателем, то есть не должен модифицироваться в ходе работы программы, вся информация доступна только на чтение.

Определить класс **Army**. Данный класс содержит стеки юнитов. Максимум в одной армии может быть 6 стеков.

Значения всех полей (одно поле - стеки) заполняются через конструктор. Этот класс является описателем, то есть не должен модифицироваться в ходе работы программы, вся информация доступна только на чтение.

Создать классы-наследники от Unit - конкретные юниты с определенными параметрами.

**Ход рассуждений:**

Для описательных классов UnitsStack, Army, Unit для каждого поля были созданы поля с геттерами ( без сеттеров, обеспечение только чтения).

Для поля Damage было решено использовать Tuple из двух uint из-за легкости обращения.

Для проверки количества стаков и юнитов в стаке был создан статический класс Config с указанием нужных констант. В конструкторе классов Army и UnitsStack были вставлены соответствующие проверки.

Для сохранения целостности информации в классе Army было принято решение возращать клон списка стеков, соответственно необходимо было создавать и клоны информации классов UnitsStack и Unit. Методы Clone в каждом классе выполняют эту задачу.

**Код:**

1. **Unit.cs**

namespace lab2

{

public class Unit

{

public uint Type { get; }

public uint HitPoints { get; }

public uint Attack { get; }

public uint Defence { get; }

public (uint, uint) Damage { get; }

public double Initiative { get; }

public Unit(uint type, uint hitPoints, uint attack, uint defence, (uint, uint) damage,

double initiative)

{

this.Type = type;

this.HitPoints = hitPoints;

this.Attack = attack;

this.Defence = defence;

this.Damage = damage;

this.Initiative = initiative;

}

public Unit Clone()

{

return new Unit(this.Type, this.HitPoints, this.Attack, this.Defence, this.Damage, this.Initiative);

}

}

}

1. **UnitsStack.cs**

using System;

namespace lab2

{

public class UnitsStack

{

public Unit UnitType { get; }

public int Amount { get;}

public UnitsStack(Unit unitType, int amount)

{

if (amount > Config.MAX\_STACK\_NUMBER)

{

throw new ArgumentException("Too much Units in one stack");

}

this.Amount = amount;

this.UnitType = unitType;

}

public override string ToString()

{

return ($"Type: {this.UnitType.Type}, Amount: {this.Amount}\n");

}

public UnitsStack Clone()

{

return new UnitsStack(this.UnitType.Clone(), this.Amount);

}

}

}

1. **Army.cs**

using System;

using System.Collections.Generic;

namespace lab2

{

public class Army

{

private readonly List<UnitsStack> \_stacksList;

public List<UnitsStack> StacksList

{

get

{

var newStacksList = new List<UnitsStack>();

\_stacksList.ForEach((stack) => newStacksList.Add(stack.Clone()));

return newStacksList;

}

}

//public int Amount => \_stacksList.Count;

public Army(List<UnitsStack> stacksList)

{

if (stacksList.Count > Config.MAX\_ARMY\_NUMBER)

{

throw new ArgumentException("To much Stacks");

}

var newStacksList = new List<UnitsStack>();

stacksList.ForEach((stack) => newStacksList.Add(stack.Clone()));

this.\_stacksList = newStacksList;

}

public override string ToString()

{

string result = "Army:\n";

foreach (var stack in \_stacksList)

{

result += stack.ToString();

}

return result;

}

public Army Clone()

{

return new Army(this.StacksList);

}

}

}

1. **Config.cs**

namespace lab2

{

public static class Config

{

public const uint MAX\_ARMY\_NUMBER = 6;

public const uint MAX\_STACK\_NUMBER = 999\_999;

}

}

1. **Units – все классы находятся в разных файлах, название файла соответствует названию класса**

namespace lab2.Units

{

public class Angel : Unit

{

public Angel() : base(1, 180, 27, 27, (45, 45), 11) { }

}

}

namespace lab2.Units

{

public class Arbalester : Unit

{

public Arbalester() : base(8, 10, 4, 4, (2, 8), 8) { }

}

}

namespace lab2.Units

{

public class BoneDragon : Unit

{

public BoneDragon() : base(3, 150, 27, 28, (15, 30), 11) { }

}

}

namespace lab2.Units

{

public class Cyclops: Unit

{

public Cyclops() : base(4, 85, 20, 15, (18, 26), 10) { }

}

}

namespace lab2.Units

{

public class Devil : Unit

{

public Devil() : base(2, 166, 27, 25, (36, 66), 11) { }

}

}

namespace lab2.Units

{

public class Fury : Unit

{

public Fury() : base(10, 16, 5, 3, (5,7), 16) { }

}

}

namespace lab2.Units

{

public class Griffin: Unit

{

public Griffin(): base(5, 30, 7, 5, (7, 12), 15) { }

}

}

namespace lab2.Units

{

public class Hydra : Unit

{

public Hydra() : base(11, 80, 15, 12, (7, 14), 7) { }

}

}

namespace lab2.Units

{

public class Lich : Unit

{

public Lich() : base(7, 50, 15, 15, (12, 17), 10) { }

}

}

namespace lab2.Units

{

public class Shaman : Unit

{

public Shaman() : base(6, 40, 12, 10, (7,12), 10.5) { }

}

}

namespace lab2.Units

{

public class Skeleton : Unit

{

public Skeleton() : base(9, 5, 1, 2, (1, 1), 10) { }

}

}