Министерство науки и высшего образования Российской Федерации

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)
Институт прикладной математики и компьютерных наук

ОТЧЕТ

по дисциплине «Внедрение и тестирование программного обеспечения»
на тему «Лаборатория 1»

Выполнил
студент группы №932101
_____А. А. Орликов

Проверил
преподаватель
_____Е. Е. Мокина

_____
оценка

Томск–2024

Введение

Прототип — порождающий паттерн, задает виды создаваемых объектов с помощью экземпляра-прототипа и создает новые объекты путем копирования этого прототипа.

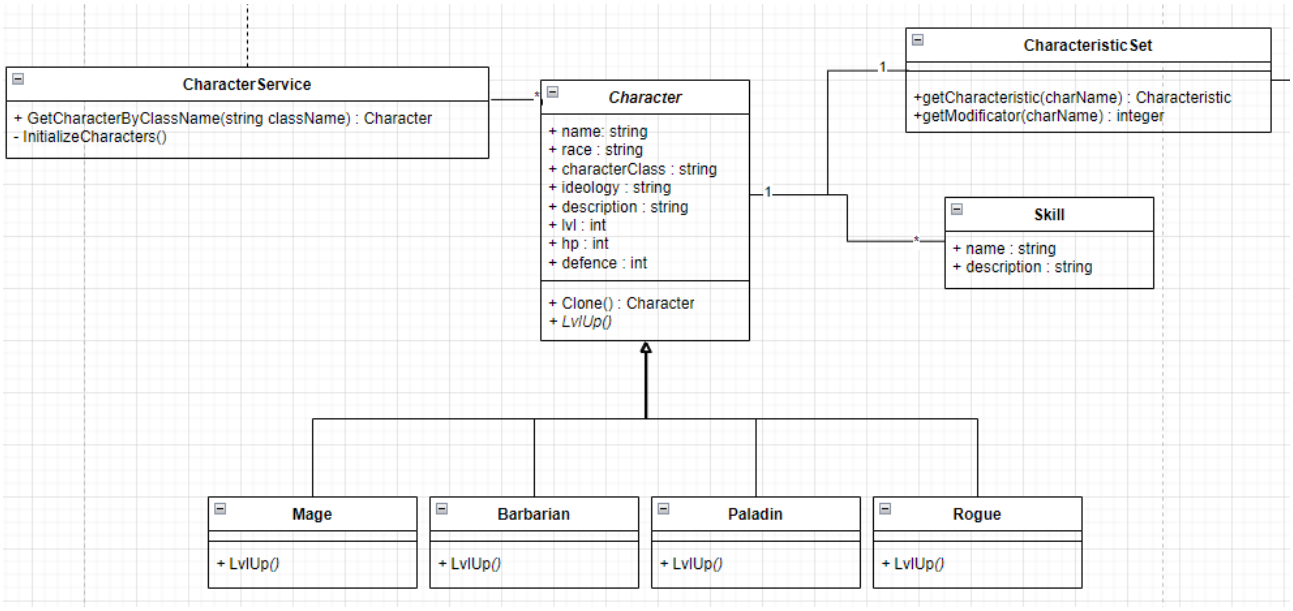
Паттерн используется чтобы:

- избежать дополнительных усилий по созданию объекта стандартным путём (имеется в виду использование конструктора, так как в этом случае также будут вызваны конструкторы всей иерархии предков объекта), когда это непозволительно дорого для приложения.
- избежать наследования создателя объекта (object creator) в клиентском приложении, как это делает паттерн abstract factory.

Признаки для использования паттерна:

- инстанцируемые классы определяются во время выполнения, например с помощью динамической загрузки;
- избежать построения иерархий классов или фабрик, параллельных иерархии классов продуктов;
- экземпляры класса могут находиться в одном из нескольких различных состояний. Может оказаться удобнее установить соответствующее число прототипов и клонировать их, а не инстанцировать каждый раз класс вручную в подходящем состоянии

Реализация



Ссылка на диаграмму:
https://drive.google.com/file/d/1gngctVkVXmJ4RLAnC3viyiMYUh8LcyS_/view?usp=sharing

В данной системе для персонажа имеется сложный конструктор, ибо в нём генерируется характеристики. Также при выборе имени персонажа мы обращаемся к api, требуется сократить создание персонажа.

У нас имеется множество классов, а в днд их еще больше, из-за чего у нас будет большая иерархия классов. Помимо этого у классов есть так называемые подклассы, которые

являются продолжениями своих классов и отличаются от них не слишком сильно, поэтому следует как-то сократить иерархию классов.

Для решения этой проблемы можно применить паттерн прототип, он решит проблему с вызовом конструктора, а также позволит сократить иерархию классов.

В коде же мы сможем не создавать на основе имени класса какой-либо объект, а просто клонировать

Было:

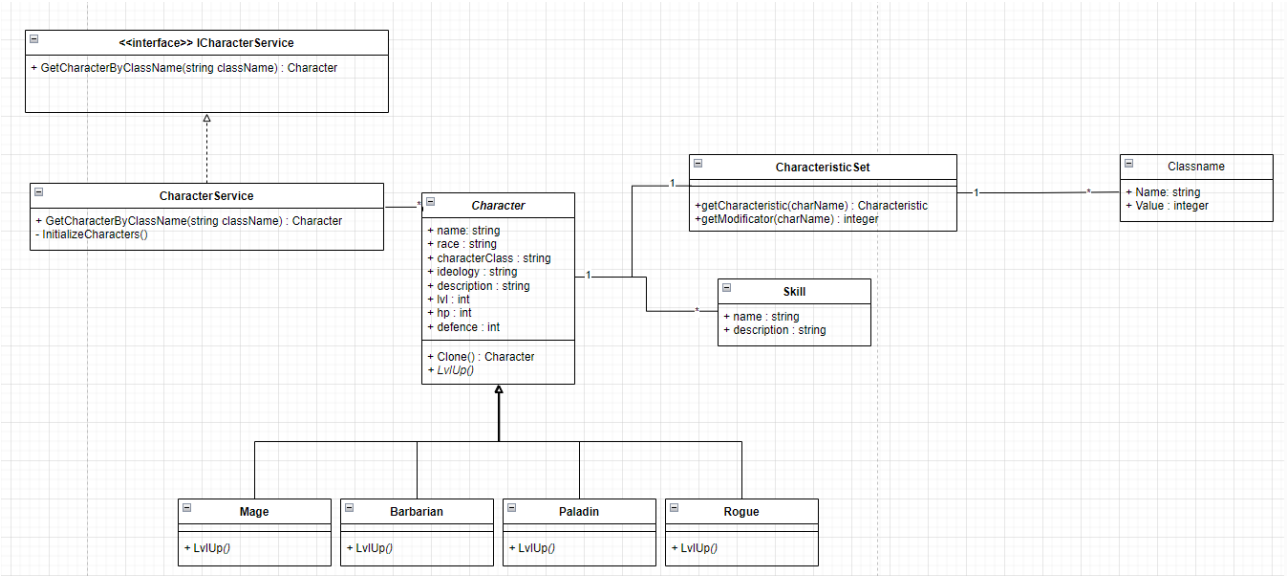
```
            Character character;
            switch(className)
            {
                case "mage":
                    character = new Mage();
                break;
                case "necromancer":
                    character = new Necromancer();
                break;
                case "paladin":
                    character = new Paladin();
                break;
                case "rogue":
                    character = new Rogue();
                break;
                default:
                    character = new Barbarian();
                break;
            }
```

Стало:

```
if (_characters.TryGetValue(className, out var characterPrototype))
            {
                var characterCopy = characterPrototype.Clone();
                return (Character)characterCopy;
            }
```

Вывод

Сокращено куча бесполезного кода, оптимизировано создание объектов, а также упрощено добавление новых классов в будующем


<<interface>> ICharacterService
+ GetCharacterByClassName(string className) : Character
CharacterService
+ GetCharacterByClassName(string className) : Character
- InitializeCharacters()
Character
+ name: string
+ race : string
+ characterClass : string
+ ideology : string
+ description : string
+ lvl : int
+ hp : int
+ defence : int
+ Clone() : Character
+ LvlUp()
CharacteristicSet
+getCharacteristic(charName) : Characteristic
+getModificator(charName) : integer
Classname
+ Name: string
+ Value : integer
Skill
+ name : string
+ description : string
Mage
+ LvlUp()
Barbarian
+ LvlUp()
Paladin
+ LvlUp()
Rogue
+ LvlUp()
1
*