Содержание

[Задание 2](#_Toc24284265)

[Постановка задачи 2](#_Toc24284266)

[Ролевая компьютерная игра 3](#_Toc24284267)

[Диаграмма классов 6](#_Toc24284268)

[Исходный код 7](#_Toc24284269)

[Основы проектирования ПО 8](#_Toc24284270)

[Содержимое текстовых файлов 14](#_Toc24284271)

# **Задание**

## **Постановка задачи**

Выберите любую из предложенных предметных областей.

1. Создайте набор взаимосвязанных классов, представляющих предметную область, в котором должны присутствовать:
   * не менее 2 деревьев наследования,
   * не менее 3 уровней наследования,
   * вложенные классы,
   * интерфейсы и абстрактные классы,
   * множественное наследование,
   * обобщённые классы.
2. Утвердите ваш проект у преподавателя.
3. Применяя основные принципы ООП, создайте члены классов, среди которых должны присутствовать:
   * открытые, закрытые и защищённые члены,
   * статические члены,
   * константы,
   * перекрытые и переопределённые в потомках методы,
   * обобщённые методы.
4. Создайте программу, реализующую ваш проект. В программе продемонстрируйте использование созданных вами классов.

Примените в вашем проекте как минимум по одному шаблону проектирования структурного, порождающего и поведенческого типов.

## **Ролевая компьютерная игра**

В ролевой компьютерной игре игрок играет за персонажа, развивая его навыки, выполняя различные миссии и сражаясь с неигровыми персонажами.

Игроку должно даваться на выбор несколько разных персонажей, отличающихся набором способностей. Все персонажи должны иметь следующие характеристики:

* **очки опыта** - определяют уровень персонажа,
* **очки здоровья** - уменьшаются при нанесении урона (если они заканчиваются, то соответствующий персонаж уничтожается),
* **очки энергии** - уменьшаются при использовании специальных навыков,
* **показатель атаки** - определяет размер урона,
* **показатель защиты** - уменьшает наносимый урон.

При желании вы можете расширить набор характеристик.

Тематику игры выберите самостоятельно.

Ресурсы

Экономика игры должна состоять из нескольких ресурсов.

Особенности ресурсов:

* набор ресурсов одинаков для всех игроков вне зависимости от выбранного персонажа,
* игрок воспроизводит ресурсы, выполняя задания и продавая артефакты,
* игрок тратит ресурсы, покупая артефакты у неигровых персонажей.

Перечень ресурсов и их особенности определите самостоятельно.

Артефакты

Артефакты предназначены для изменения характеристик персонажа.

Особенности артефактов:

* могут восстанавливать очки здоровья и энергии,
* могут увеличивать количество очков опыта, здоровья, энергии, а также повышать показатели атаки и защиты,
* продаются и покупаются у неигровых персонажей за ресурсы.

Перечень артефактов, их характеристики и особенности определите самостоятельно.

Неигровые персонажи

Игрок взаимодействует с неигровыми персонажами.

Особенности неигровых персонажей:

* могут быть настроены агрессивно или нейтрально по отношению к персонажу игрока,
* агрессивные персонажи могут атаковать персонажа игрока,
* нейтральные персонажи могут торговать с персонажем игрока,
* нейтральные персонажи могут выдавать и принимать задания от игрока,
* могут быть атакованы игроком, в этом случае нейтральные персонажи становятся агрессивными.

Перечень неигровых персонажей, их характеристики и особенности определите самостоятельно.

Задания

Задания позволяют игроку накапливать опыт.

Возможные условия выполнения задания:

* уничтожение одного или нескольких неигровых персонажей,
* передача неигровому персонажу определённого количества ресурсов,
* передача неигровому персонажу одного или нескольких определённых артефактов.

Особенности заданий:

* выдаются нейтральными персонажами,
* при выполнении условий задания игроку даются очки опыта.

Перечень заданий и их особенности определите самостоятельно.

Персонаж игрока

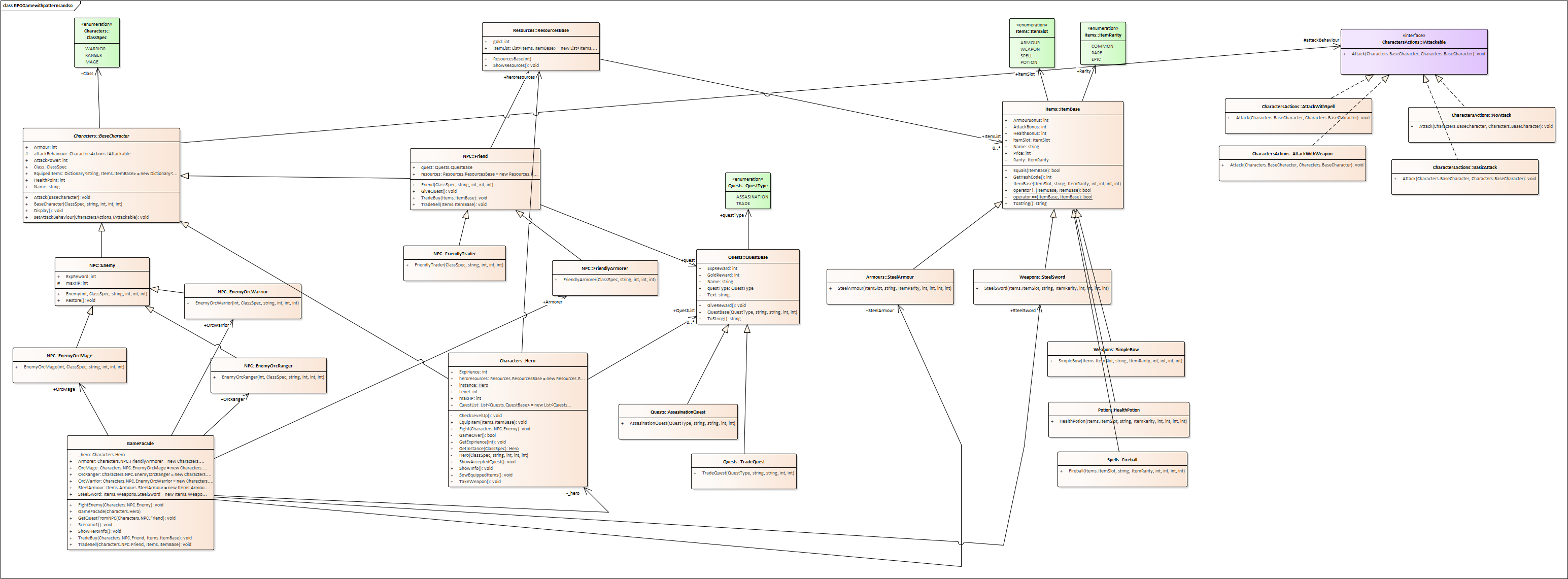
Игрок взаимодействует с игрой с помощью персонажа.

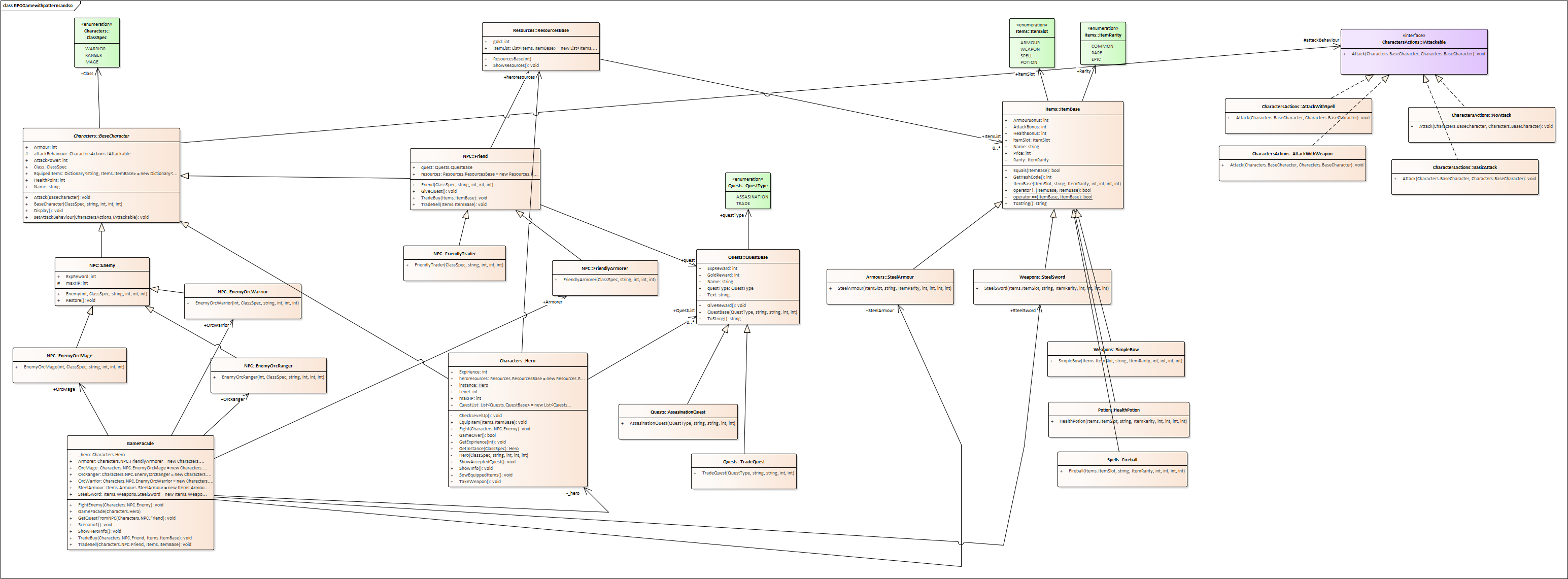
Особенности персонажей:

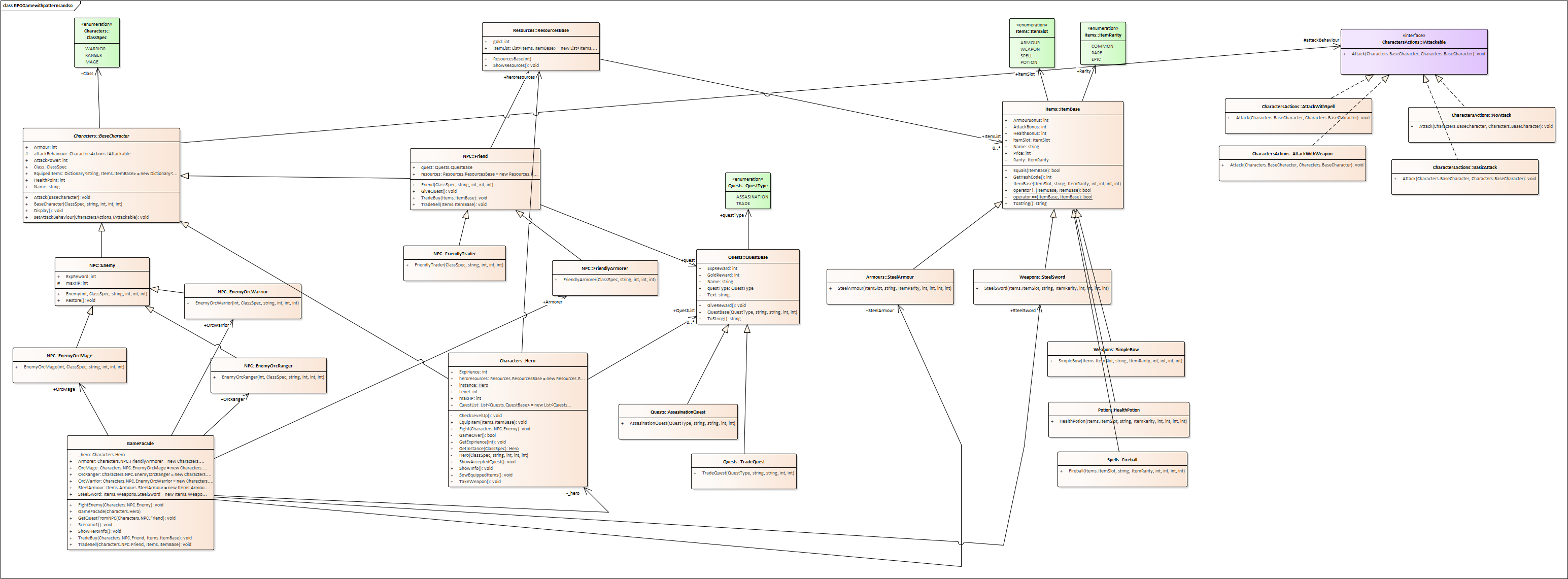
* могут атаковать любого другого персонажа,
* могут покупать или продавать артефакты у нейтральных персонажей,
* могут использовать артефакты,
* могут брать или сдавать задания у нейтральных персонажей.
* при выполнении задания получают очки опыта.

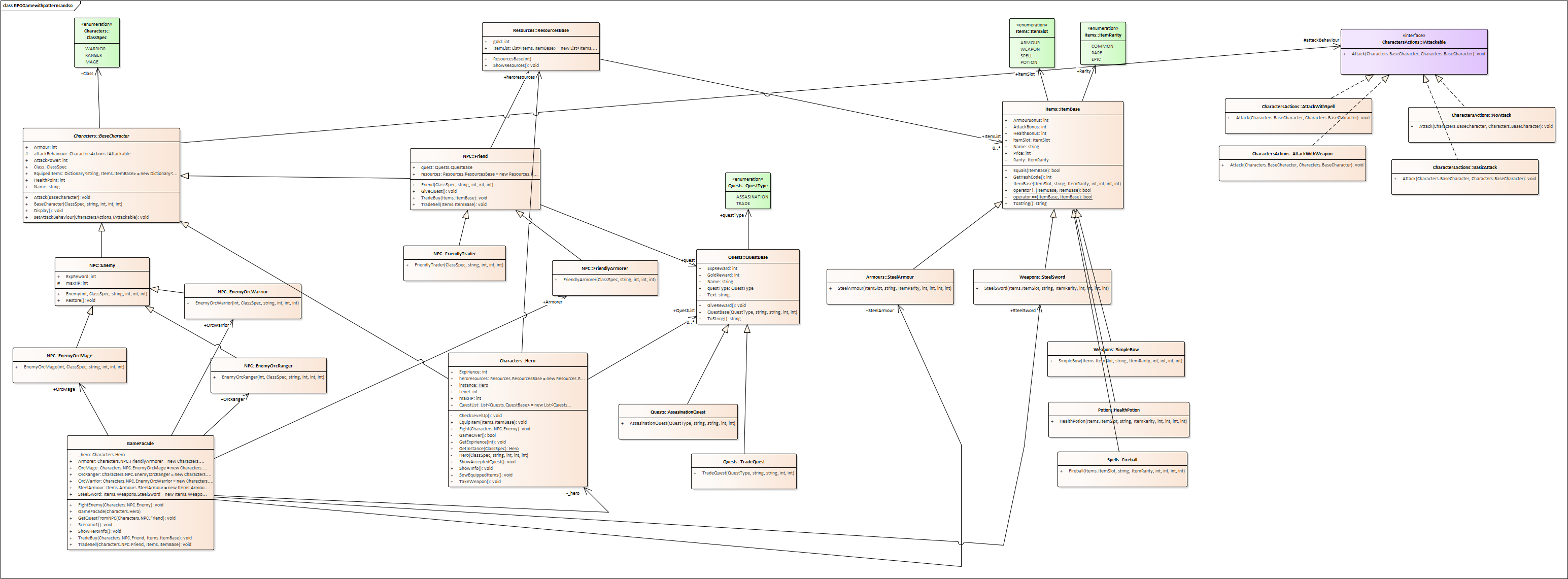
Перечень персонажей, их характеристики и особенности определите самостоятельно.

# **Диаграмма классов**









# **Исходный код**

Список всех каталогов и файлов приложения с описанием:

Characters – каталог с файлами классов персонажей

* + - NPC – каталог с файлами классов не игровых персонажей
      * Enemy.cs - файл, описывающий врагов
      * EnemyOrcMage.cs - файл, описывающий ‘OrcMage’
      * EnemyOrcRanger.cs - файл, описывающий ‘OrcRanger’
      * EnemyOrcWarrior.cs - файл, описывающий ‘OrcWarrior’
      * Friend.cs - файл, описывающий дружественных NPC
      * FriendlyArmorer.cs-файл,описывающий ‘FriendlyArmorer’
      * FriendlyTrader.cs - файл, описывающий ‘FriendlyTrader
    - BaseCharacter.cs - файл, описывающий абстрактный класс персонажей
    - Hero.cs - файл, описывающий класс главного героя
  + CharacterActions – каталог с файлами интерфейса атаки и его реализаций
    - AttackWithSpell.cs – класс, реализующий атаку персонажа
    - AttackWithWeapon.cs – класс, реализующий атаку персонажа
    - BasicAttack.cs – класс, реализующий атаку персонажа IAttackable.cs – интерфейс, реализующий паттерн ‘стратегия’
    - NoAttack.cs – класс, реализующий атаку персонажа
  + Items – каталог с файлами классов, описывающих предметы
    - Armours – каталог с файлами классов, описывающих броню
      * SteelArmour.cs – класс, описывающий предмет в игре
    - Potion – каталог с файлами, описывающих зелья
      * HealthPotion.cs – класс, описывающий предмет в игре
    - Spells – каталог с файлами, описывающих заклинания
      * Fireball.cs – класс, описывающий заклинание в игре
    - Weapons – каталог с файлами, описывающих оружие
      * SimpleBow.cs – класс, описывающий предмет в игре
      * SteelSword.cs – класс, описывающий предмет в игре
    - ItemBase.cs - абстрактный класс, содержащий базовый набор свойств и методов предметов
  + Quests – каталог с файлами, описывающих задания
    - AssasinationQuest.cs - класс, описывающий боевой квест
    - QuestBase.cs – абстрактный класс, содержащий базовый набор свойств и методов заданий
    - TradeQuest.cs - класс, описывающий торговый квест
  + Resources – каталог с файлами, описывающих ресурсы игры
    - ResourcesBase.cs – класс, описывающий ресурсы игры
  + GameFacade.cs – класс для объединения функциональности классов в сценарии
  + Program.cs – файл для заппуска тесовых сценариев

# **Основы проектирования ПО**

**Порождающий шаблон** проектирования «Синглтон». Синглтон — шаблон проектирования, гарантирующий, что в однопоточном приложении будет единственный экземпляр некоторого класса, и предоставляющий глобальную точку доступа к этому экземпляру.

В моем проекте я использовал синглтон для реализации главного персонажа, так как его уникальность обусловлена предметной областью. Исходный код класса «Hero» с реализацией этого шаблона:

**class** **Hero**:BaseCharacter

{

**private** **static** Hero instance;

**public** **static** Hero GetInstance(ClassSpec classSpec)

{

**if** (instance == **null**)

{

**switch** (classSpec)

{

**case** ClassSpec.WARRIOR:

instance = **new** Hero(ClassSpec.WARRIOR, *"HeroWarrior"*, 120, 0, 10);

**break**;

**case** ClassSpec.RANGER:

instance = **new** Hero(ClassSpec.RANGER, *"HeroRanger"*, 100, 2, 5);

**break**;

**case** ClassSpec.MAGE:

instance = **new** Hero(ClassSpec.RANGER, *"HeroMage"*, 80, 1, 2);

**break**;

}

}

**return** instance;

}

. . . .

}

Пример использования этого шаблона:

Дружественные NPC могут выдавать квесты. Для этого мы получаем объект игрока и добавляем ему в пул квест.

**public** **void** GiveQuest()

{

Console.WriteLine(quest.ToString());

Characters.Hero.GetInstance(ClassSpec.WARRIOR).QuestList.Add(quest);

}

Торговля осуществляется также через обращение к синглтону.

public void TradeSell(Items.ItemBase ItemToSell)

{

int tmp = 0;

Characters.Hero hero = Characters.Hero.GetInstance(Hero.ClassSpec.WARRIOR);

foreach (Items.ItemBase item in resources.ItemList)

{

if (item == ItemToSell && item.Price <= hero.heroresources.gold)

{

hero.heroresources.gold -= item.Price;

hero.heroresources.ItemList.Add(item);

resources.gold += item.Price;

resources.ItemList.Remove(item);

Console.WriteLine($"You have bought '{item.Name}' for {item.Price} gold. Current gold: {hero.heroresources.gold}");

tmp++;

break;

}

}

if (tmp==0) { Console.WriteLine("Something gone wrong. Mb you are cut on money))"); }

}

**Поведенческий шаблон** «Стратегия» — шаблон проектирования, предназначенный для определения семейства алгоритмов, инкапсуляции каждого из них и обеспечения их взаимозаменяемости. Это позволяет выбирать алгоритм путём определения соответствующего класса. Шаблон Strategy позволяет менять выбранный алгоритм независимо от объектов-клиентов, которые его используют.

Этот паттерн используется для реализации боевой системы. Исходный код интерфейса:

**interface** IAttackable

{

**void** Attack(Characters.BaseCharacter Attacker,Characters.BaseCharacter Target);

}

Пример его реализации:

**class** **AttackWithSpell**:IAttackable

{

**public** **void** Attack(Characters.BaseCharacter Attacker, Characters.BaseCharacter Target)

{

Target.HealthPoint = Target.HealthPoint - (Attacker.AttackPower \* Attacker.EquipedItems[*"Spell"*].AttackBonus - Target.Armour);

Console.WriteLine(*" {1} cast {0} on {3}. {0} <{2} HP>"*,

Target.Name, Attacker.Name, Target.HealthPoint, Attacker.EquipedItems[*"Spell"*].Name);

}}

**class** **NoAttack**:IAttackable

{

**public** **void** Attack(Characters.BaseCharacter Attacker, Characters.BaseCharacter Target)

{

Console.WriteLine(*"This character can not attack"*);

}

}

Данная реализация позволяет выбрать подходящий тип атаки для разных игровых персонажей. Например персонаж «OrcMage» или «Hero» могут использовать заклинания, следовательно будут использовать алгоритм «AttackWithSpell», в то время как торговцы не могут атаковать, они будут использовать алгоритм «NoAttack». Преимущество данной реализации заключается в простоте расширения видов атаки.

Пример использования:

**abstract** **class** **BaseCharacter**

{

**protected** CharactersActions.IAttackable attackBehaviour;

**public** BaseCharacter(ClassSpec @class = ClassSpec.WARRIOR, **string** name = *"NoName"*,

**int** healthPoint = 100, **int** armour = 0, **int** attackPower = 0)

{

EquipedItems.Add(*"Weapon"*, **new** Items.Weapons.SteelSword());

EquipedItems.Add(*"Armour"*, **new** Items.Armours.SteelArmour());

EquipedItems.Add(*"Spell"*, **new** Items.Spells.Fireball());

EquipedItems.Add(*"Potion"*, **new** Items.Potion.HealthPotion());

Class = @class;

Name = name;

HealthPoint = healthPoint;

Armour = armour;

AttackPower = attackPower;

attackBehaviour = **new** CharactersActions.BasicAttack();

}

**public** **void** setAttackBehaviour(CharactersActions.IAttackable newAttackBehaviour)

{

attackBehaviour = newAttackBehaviour;

}

}

Задаем классу-родителю игровых персонажей базовый тип атаки. Также определяем метод для изменения этого типа.

**class** **EnemyOrcMage** : Enemy

{

**public** EnemyOrcMage(**int** expReward = 10, ClassSpec @class = ClassSpec.MAGE, **string** name = *"OrcsmerMage"*,

**int** healthPoint = 25, **int** armour = 0, **int** attackPower = 2)

: **base**(expReward, @class, name, healthPoint, armour, attackPower)

{

EquipedItems[*"Spell"*] = **new** Items.Spells.Fireball();

attackBehaviour = **new** CharactersActions.AttackWithSpell();

}

}

Снаряжаем персонажа магическим оружием и присваиваем ему тип атаки «AttackWithSpell».

**Структурный шаблон** фасад — шаблон проектирования, позволяющий скрыть сложность системы путём сведения всех возможных внешних вызовов к одному объекту, делегирующему их соответствующим объектам системы.

Этот паттерн используется для реализации взаимодействия пользователя и игровой системы. С ее помощью он может вызывать игровые сценарии и действия без вникания в работу системы.

Исходный код шаблона:

**class** **GameFacade**

{

*//типа комплексные бои прописать*

**private** Characters.Hero \_hero;

**public** Characters.NPC.EnemyOrcWarrior OrcWarrior = **new** Characters.NPC.EnemyOrcWarrior();

**public** Characters.NPC.EnemyOrcRanger OrcRanger = **new** Characters.NPC.EnemyOrcRanger();

**public** Characters.NPC.EnemyOrcMage OrcMage = **new** Characters.NPC.EnemyOrcMage();

**public** Characters.NPC.FriendlyArmorer Armorer = **new** Characters.NPC.FriendlyArmorer();

**public** Items.Armours.SteelArmour SteelArmour = **new** Items.Armours.SteelArmour();

**public** Items.Weapons.SteelSword SteelSword = **new** Items.Weapons.SteelSword();

**public** GameFacade(Characters.Hero hero)

{

Console.WriteLine($*"Game is started you are <{hero.Name}>"*);

\_hero = hero;

}

**public** **void** FightEnemy(Characters.NPC.Enemy enemy)

{

\_hero.Fight(enemy);

}

**public** **void** ShowHeroInfo()

{

\_hero.ShowInfo();

}

**public** **void** TradeBuy(Characters.NPC.Friend Trader, Items.ItemBase item)

{

Trader.TradeSell(item);

}

**public** **void** TradeSell(Characters.NPC.Friend Trader, Items.ItemBase item)

{

Trader.TradeBuy(item);

}

**public** **void** GetQuestFromNPC(Characters.NPC.Friend friend)

{

friend.GiveQuest();

}

**public** **void** Scenario1()

{

Console.WriteLine(*"You are in the Armorer Shop"*);

Armorer.TradeSell(SteelSword);

Armorer.TradeSell(SteelArmour);

\_hero.EquipItem(SteelSword);

\_hero.EquipItem(SteelArmour);

Console.WriteLine(*"--------------------"*);

ShowHeroInfo();

Console.WriteLine(*"--------------------"*);

GetQuestFromNPC(Armorer);

Console.WriteLine(*"--------------------"*);

FightEnemy(OrcWarrior);

}

}

Пример использования в тестировании базовых функций программы:

**class** **Program**

{

**static** **void** Main(**string**[] args)

{

**var** Hero = Characters.Hero.GetInstance(Characters.BaseCharacter.ClassSpec.WARRIOR);

GameFacade game = **new** GameFacade(Hero);

game.Scenario1();

game.FightEnemy(game.OrcWarrior);

game.TradeBuy(game.Armorer, **new** Items.Weapons.SteelSword());

game.ShowHeroInfo();

Console.ReadKey();

}

}

# **Содержимое текстовых файлов**

Enemy.cs

namespace RPGGamewithpatternsandso.Characters.NPC

{

class Enemy:BaseCharacter

{

public int ExpReward;

protected int maxHP;

public Enemy(int expReward, ClassSpec @class = ClassSpec.WARRIOR, string name = "NoName", int healthPoint = 100, int armour = 0, int attackPower = 0) : base(@class, name, healthPoint, armour, attackPower)

{

ExpReward = expReward;

maxHP = healthPoint;

}

public void Restore()

{

HealthPoint = maxHP;

}

}

}

EnemyOrcMage.cs

namespace RPGGamewithpatternsandso.Characters.NPC

{

class EnemyOrcMage : Enemy

{

public EnemyOrcMage(int expReward = 10, ClassSpec @class = ClassSpec.MAGE, string name = "OrcsmerMage",

int healthPoint = 25, int armour = 0, int attackPower = 2)

: base(expReward, @class, name, healthPoint, armour, attackPower)

{

EquipedItems["Spell"] = new Items.Spells.Fireball();

attackBehaviour = new CharactersActions.AttackWithSpell();

}

}

}

EnemyOrcRanger.cs

namespace RPGGamewithpatternsandso.Characters.NPC

{

class EnemyOrcRanger:Enemy

{

public EnemyOrcRanger(int expReward = 10, ClassSpec @class = ClassSpec.RANGER, string name = "OrcsmerRanger",

int healthPoint = 20, int armour = 1, int attackPower = 3)

: base(expReward, @class, name, healthPoint, armour, attackPower)

{

EquipedItems["Weapon"] = new Items.Weapons.SimpleBow();

attackBehaviour = new CharactersActions.AttackWithWeapon();

}

}

}

EnemyOrcWarrior.cs

namespace RPGGamewithpatternsandso.Characters.NPC

{

class EnemyOrcWarrior : Enemy

{

public EnemyOrcWarrior(int expReward = 99, ClassSpec @class = ClassSpec.WARRIOR, string name = "OrcsmerWarrior",

int healthPoint = 30, int armour = 2, int attackPower = 5)

: base(expReward, @class, name, healthPoint, armour, attackPower)

{

}

}

}

Friend.cs

namespace RPGGamewithpatternsandso.Characters.NPC

{

class Friend:BaseCharacter

{

public Quests.QuestBase quest;

public Resources.ResourcesBase resources = new Resources.ResourcesBase(999);

public Friend(ClassSpec @class = ClassSpec.WARRIOR, string name = "NoName", int healthPoint = 100, int armour = 0,

int attackPower = 0) : base(@class, name, healthPoint, armour, attackPower)

{

}

public void GiveQuest()

{

Console.WriteLine(quest.ToString());

Characters.Hero.GetInstance(ClassSpec.WARRIOR).QuestList.Add(quest);

}

public void TradeSell(Items.ItemBase ItemToSell)

{

int tmp = 0;

Characters.Hero hero = Characters.Hero.GetInstance(Hero.ClassSpec.WARRIOR);

foreach (Items.ItemBase item in resources.ItemList)

{

if (item == ItemToSell && item.Price <= hero.heroresources.gold)

{

hero.heroresources.gold -= item.Price;

hero.heroresources.ItemList.Add(item);

resources.gold += item.Price;

resources.ItemList.Remove(item);

Console.WriteLine($"You have bought '{item.Name}' for {item.Price} gold. Current gold: {hero.heroresources.gold}");

tmp++;

break;

}

}

if (tmp==0) { Console.WriteLine("Something gone wrong. Mb you are cut on money))"); }

}

public void TradeBuy(Items.ItemBase ItemToBuy)

{

Characters.Hero hero = Characters.Hero.GetInstance(Hero.ClassSpec.WARRIOR);

foreach (Items.ItemBase item in resources.ItemList)

{

if (item == ItemToBuy && item.Price <= resources.gold)

{

hero.heroresources.gold += item.Price;

resources.ItemList.Add(item);

resources.gold -= item.Price;

hero.heroresources.ItemList.Remove(item);

Console.WriteLine($"You have sold '{item.Name}' for {item.Price} gold. Current gold: {hero.heroresources.gold}");

break;

}

else { Console.WriteLine("Something gone wrong"); }

}

}

}

}

FriendlyArmorer.cs

namespace RPGGamewithpatternsandso.Characters.NPC

{

class FriendlyArmorer : Friend

{

public FriendlyArmorer(ClassSpec @class = ClassSpec.WARRIOR, string name = "Armorer", int healthPoint = 100, int armour = 0, int attackPower = 0)

: base(@class, name, healthPoint, armour, attackPower)

{

attackBehaviour = new CharactersActions.NoAttack();

quest = new Quests.AssasinationQuest();

for (int i = 0; i<=10;i++)

{

resources.ItemList.Add(new Items.Armours.SteelArmour());

resources.ItemList.Add(new Items.Weapons.SimpleBow());

resources.ItemList.Add(new Items.Weapons.SteelSword());

}

}

}

}

FriendlyTrader.cs

namespace RPGGamewithpatternsandso.Characters.NPC

{

class FriendlyTrader : BaseCharacter

{

public FriendlyTrader(ClassSpec @class, string name, int healthPoint, int armour, int attackPower)

: base(@class, "GoodsTrader", healthPoint, armour, attackPower)

{

attackBehaviour = new CharactersActions.NoAttack();

}

}

}

BaseCharacter.cs

namespace RPGGamewithpatternsandso.Characters

{

abstract class BaseCharacter

{

public enum ClassSpec

{

WARRIOR,

RANGER,

MAGE

}

public ClassSpec Class;

public Dictionary<string, Items.ItemBase> EquipedItems = new Dictionary<string, Items.ItemBase>();

public string Name;

public int HealthPoint;

public int Armour;

public int AttackPower;

protected CharactersActions.IAttackable attackBehaviour;

public BaseCharacter(ClassSpec @class = ClassSpec.WARRIOR, string name = "NoName",

int healthPoint = 100, int armour = 0, int attackPower = 0)

{

EquipedItems.Add("Weapon", new Items.Weapons.SteelSword());

EquipedItems.Add("Armour", new Items.Armours.SteelArmour());

EquipedItems.Add("Spell", new Items.Spells.Fireball());

EquipedItems.Add("Potion", new Items.Potion.HealthPotion());

Class = @class;

Name = name;

HealthPoint = healthPoint;

Armour = armour;

AttackPower = attackPower;

attackBehaviour = new CharactersActions.BasicAttack();

}

public void setAttackBehaviour(CharactersActions.IAttackable newAttackBehaviour)

{

attackBehaviour = newAttackBehaviour;

}

public void Attack(BaseCharacter baseCharacter)

{

attackBehaviour.Attack(this, baseCharacter);

}

public void Display()

{

Console.WriteLine("This character is <{0} {1}> with <{2} HP>, <{3} Armour>, <{4} AP>",

Name,Class,HealthPoint,Armour,AttackPower);

}

}

}

Hero.cs

namespace RPGGamewithpatternsandso.Characters

{

//синглтон

class Hero:BaseCharacter

{

public List<Quests.QuestBase> QuestList = new List<Quests.QuestBase>();

public int Expirience;

public int Level;

public int maxHP;

public Resources.ResourcesBase heroresources = new Resources.ResourcesBase(100);

private static Hero instance;

private Hero(ClassSpec @class, string name, int healthPoint, int armour, int attackPower)

: base(@class, name, healthPoint, armour, attackPower)

{

//attackBehaviour = new CharactersActions.AttackWithWeapon();

maxHP = healthPoint;

Expirience = 0;

Level = 0;

}

public static Hero GetInstance(ClassSpec classSpec)

{

if (instance == null)

{

switch (classSpec)

{

case ClassSpec.WARRIOR:

instance = new Hero(ClassSpec.WARRIOR, "HeroWarrior", 120, 0, 10);

break;

case ClassSpec.RANGER:

instance = new Hero(ClassSpec.RANGER, "HeroRanger", 100, 2, 5);

break;

case ClassSpec.MAGE:

instance = new Hero(ClassSpec.RANGER, "HeroMage", 80, 1, 2);

break;

}

}

return instance;

}

public void Fight(Characters.NPC.Enemy target)

{

Console.WriteLine($"{Name} start fight to death with {target.Name}");

while (target.HealthPoint > 0)

{

Attack(target);

if (target.HealthPoint > 0)

target.Attack(this);

if (GameOver())

break;

}

if (!GameOver())

{

foreach (Quests.QuestBase q in QuestList)

{

if (q.questType == Quests.QuestBase.QuestType.ASSASINATION)

q.GiveReward();

}

target.Restore();

GetExpirience(target.ExpReward);

}

}

public void GetExpirience(int exp)

{

Expirience += exp;

CheckLevelUp();

}

public void TakeWeapon()

{

attackBehaviour = new CharactersActions.AttackWithWeapon();

}

public void ShowAcceptedQuest()

{

foreach (Quests.QuestBase q in QuestList)

{

q.ToString();

}

}

public void SowEquippedItems()

{

foreach (string key in EquipedItems.Keys)

{

Console.WriteLine(" In your {0} slot equipped {1}",

EquipedItems[key].itemSlot,EquipedItems[key].Name);

Console.WriteLine(EquipedItems[key].ToString());

}

}

public void ShowInfo()

{

Console.WriteLine($"{@Class}, {Name}, <hp {HealthPoint}>, <armor {Armour}>, <attackpower {AttackPower}> it");

heroresources.ShowResources();

SowEquippedItems();

}

public void EquipItem(Items.ItemBase item)

{

int tmp = 0;

foreach (Items.ItemBase i in heroresources.ItemList)

{

if (i == item)

{

Console.WriteLine("you have equiped a {0} in {1} slot", i.Name, i.itemSlot);

attackBehaviour = new CharactersActions.AttackWithWeapon();

heroresources.ItemList.Remove(i);

tmp++;

break;

}

}

if (tmp == 0) { Console.WriteLine("You dont have such an item: {0}", item.Name); }

}

private void CheckLevelUp()

{

if (Expirience >= 100)

{

Level = Level++;

maxHP += 10;

HealthPoint = maxHP;

Console.WriteLine("You get level up. HP restored");

}

}

private bool GameOver()

{

if (HealthPoint <= 0)

{

Console.WriteLine("Your Hero is dead. Game over!");

return true;

}

else

{

return false;

}

}

}

}

AttackWithSpell.cs

namespace RPGGamewithpatternsandso.CharactersActions

{

class AttackWithSpell:IAttackable

{

public void Attack(Characters.BaseCharacter Attacker, Characters.BaseCharacter Target)

{

Target.HealthPoint = Target.HealthPoint - (Attacker.AttackPower \* Attacker.EquipedItems["Spell"].AttackBonus - Target.Armour);

Console.WriteLine(" {1} cast {0} on {3}. {0} <{2} HP>",

Target.Name, Attacker.Name, Target.HealthPoint, Attacker.EquipedItems["Spell"].Name);

}

}

}

AttackWithWeapon.cs

namespace RPGGamewithpatternsandso.CharactersActions

{

class AttackWithWeapon:IAttackable

{

public void Attack(Characters.BaseCharacter Attacker, Characters.BaseCharacter Target)

{

Target.HealthPoint = Target.HealthPoint - (Attacker.AttackPower + Attacker.EquipedItems["Weapon"].AttackBonus - Target.Armour);

Console.WriteLine(" {1} attack {0} with {3}. {0} <{2} HP>",

Target.Name, Attacker.Name, Target.HealthPoint, Attacker.EquipedItems["Weapon"].Name);

}

}

}

BasicAttack.cs

namespace RPGGamewithpatternsandso.CharactersActions

{

class BasicAttack:IAttackable

{

public void Attack(Characters.BaseCharacter Attacker, Characters.BaseCharacter Target)

{

Target.HealthPoint = Target.HealthPoint - (Attacker.AttackPower - Target.Armour);

Console.WriteLine(" {1} attack {0} with basicAttack. {0} <{2} HP>"

,Target.Name,Attacker.Name,Target.HealthPoint);

}

}

}

IAttackable.cs

namespace RPGGamewithpatternsandso.CharactersActions

{

interface IAttackable

{

void Attack(Characters.BaseCharacter Attacker,Characters.BaseCharacter Target);

}

}

NoAttack.cs

namespace RPGGamewithpatternsandso.CharactersActions

{

class NoAttack:IAttackable

{

public void Attack(Characters.BaseCharacter Attacker, Characters.BaseCharacter Target)

{

Console.WriteLine("This character can not attack");

}

}

}

SteelArmour.cs

namespace RPGGamewithpatternsandso.Items.Armours

{

class SteelArmour : ItemBase

{

public SteelArmour(ItemSlot slot = ItemSlot.ARMOUR,string name = "SteelArmour", ItemRarity rarity = ItemRarity.COMMON,

int price = 100, int attackBonus = 0, int armourBonus = 3, int healthBonus = 10)

: base(slot,name, rarity, price, attackBonus, armourBonus, healthBonus)

{

}

}

}

HealthPotion.cs

namespace RPGGamewithpatternsandso.Items.Potion

{

class HealthPotion : ItemBase

{

public HealthPotion(Items.ItemSlot slot = ItemSlot.POTION ,string name = "HealthPotion", ItemRarity rarity = ItemRarity.COMMON,

int price = 5, int attackBonus = 0, int armourBonus = 0, int healthBonus = 50)

: base(slot,name, rarity, price, attackBonus, armourBonus, healthBonus)

{

}

}

}

Fireball.cs

namespace RPGGamewithpatternsandso.Items.Spells

{

class Fireball : ItemBase

{

public Fireball(Items.ItemSlot slot = ItemSlot.SPELL, string name = "Fireball", ItemRarity rarity = ItemRarity.COMMON,

int price = 4, int attackBonus = 6, int armourBonus = 0, int healthBonus = 0)

: base(slot, name, rarity, price, attackBonus, armourBonus, healthBonus)

{

}

}

}

SimpleBow.cs

namespace RPGGamewithpatternsandso.Items.Weapons

{

class SimpleBow : ItemBase

{

public SimpleBow(Items.ItemSlot slot = ItemSlot.WEAPON, string name = "SimpleBow", ItemRarity rarity = ItemRarity.COMMON, int price = 5, int attackBonus = 3,

int armourBonus = 0, int healthBonus = 0) : base(slot,name, rarity, price, attackBonus, armourBonus, healthBonus)

{

}

}

}

SteelSword.cs

namespace RPGGamewithpatternsandso.Items.Weapons

{

class SteelSword : ItemBase

{

public SteelSword(Items.ItemSlot slot = ItemSlot.WEAPON, string name = "SteelSword", ItemRarity rarity = ItemRarity.COMMON, int price = 10,

int attackBonus = 3, int armourBonus = 0, int healthBonus = 0)

: base(slot, name, rarity, price, attackBonus, armourBonus, healthBonus)

{

}

}

}

ItemBase.cs

namespace RPGGamewithpatternsandso.Items

{

public enum ItemSlot

{

ARMOUR,

WEAPON,

SPELL,

POTION

}

class ItemBase

{

public enum ItemRarity

{COMMON,

RARE,

EPIC}

public ItemRarity Rarity;

public ItemSlot itemSlot;

public string Name;

public int Price;

public int AttackBonus;

public int ArmourBonus;

public int HealthBonus;

public ItemBase(ItemSlot slot, string name = "NoName", ItemRarity rarity = ItemRarity.COMMON, int price = 0,

int attackBonus = 0, int armourBonus = 0, int healthBonus = 0)

{

itemSlot = slot;

Name = name;

Rarity = rarity;

Price = price;

AttackBonus = attackBonus;

ArmourBonus = armourBonus;

HealthBonus = healthBonus;

}

public override string ToString()

{

return $"{Name} {Rarity}, cost {Price} gold \nstats: <{AttackBonus} attackb> <{HealthBonus} hb> <{ArmourBonus} armourb>";

}

public static bool operator ==(ItemBase item1, ItemBase item2)

{

return item1.Equals(item2);

}

public static bool operator !=(ItemBase item1, ItemBase item2)

{

return !item1.Equals(item2);

}

public bool Equals(ItemBase item)

{

if (Name == item.Name && Rarity == item.Rarity && Price == item.Price)

{

return true;

}

else

{

return false;

}

}

public override int GetHashCode()

{

return Price.GetHashCode();

}

}

}

AssasinationQuest.cs

namespace RPGGamewithpatternsandso.Quests

{

class AssasinationQuest : QuestBase

{

public AssasinationQuest(QuestType qType = QuestType.ASSASINATION, string name = "Orc assasination",

string text = "Hello! Go kill any type of Enemy, I will pay", int goldReward = 10, int expReward = 10)

: base(qType, name, text, goldReward, expReward)

{

}

}

}

QuestBase.cs

namespace RPGGamewithpatternsandso.Quests

{

class QuestBase

{

public enum QuestType

{

ASSASINATION,

TRADE

}

public QuestType questType;

public string Name;

public string Text;

public int GoldReward;

public int ExpReward;

public QuestBase(QuestType qType, string name = "NoName", string text = "Text", int goldReward = 0, int expReward = 0)

{

questType = qType;

Name = name;

Text = text;

GoldReward = goldReward;

ExpReward = expReward;

}

public void GiveReward()

{

Characters.Hero.GetInstance(Characters.BaseCharacter.ClassSpec.WARRIOR).heroresources.gold += GoldReward;

Characters.Hero.GetInstance(Characters.BaseCharacter.ClassSpec.WARRIOR).Expirience += GoldReward;

Console.WriteLine($"Quest name '{Name}' completed. \nRewards: exp {ExpReward}, gold {GoldReward}");

}

public override string ToString()

{

return $"Quest name '{Name}'.\n{Text} \nRewards: exp {ExpReward}, gold {GoldReward}";

}

}

}

TradeQuest.cs

namespace RPGGamewithpatternsandso.Quests

{

class TradeQuest : QuestBase

{

public TradeQuest(QuestType qType = QuestType.TRADE, string name = "Trade",

string text = "Hello! Sell me some thing", int goldReward = 10, int expReward = 10)

: base(qType, name, text, goldReward, expReward)

{

}

}

}

ResourcesBase.cs

using RPGGamewithpatternsandso.Items;

namespace RPGGamewithpatternsandso.Resources

{

class ResourcesBase

{

public int gold;

public List<Items.ItemBase> ItemList = new List<Items.ItemBase>();

public ResourcesBase(int gold)

{

this.gold = gold;

ItemList.Add(new Items.Potion.HealthPotion());

}

public void ShowResources()

{

Console.WriteLine("You have {0} gold. Items in your stash:", gold);

foreach (Items.ItemBase item in ItemList)

{

Console.WriteLine(" "+item.ToString());

}

}

}

}

GameFacade.cs

namespace RPGGamewithpatternsandso

{

class GameFacade

{

//типа комплексные бои прописать

private Characters.Hero \_hero;

public Characters.NPC.EnemyOrcWarrior OrcWarrior = new Characters.NPC.EnemyOrcWarrior();

public Characters.NPC.EnemyOrcRanger OrcRanger = new Characters.NPC.EnemyOrcRanger();

public Characters.NPC.EnemyOrcMage OrcMage = new Characters.NPC.EnemyOrcMage();

public Characters.NPC.FriendlyArmorer Armorer = new Characters.NPC.FriendlyArmorer();

public Items.Armours.SteelArmour SteelArmour = new Items.Armours.SteelArmour();

public Items.Weapons.SteelSword SteelSword = new Items.Weapons.SteelSword();

public GameFacade(Characters.Hero hero)

{

Console.WriteLine($"Game is started you are <{hero.Name}>");

\_hero = hero;

}

public void FightEnemy(Characters.NPC.Enemy enemy)

{

\_hero.Fight(enemy);

}

public void ShowHeroInfo()

{

\_hero.ShowInfo();

}

public void TradeBuy(Characters.NPC.Friend Trader, Items.ItemBase item)

{

Trader.TradeSell(item);

}

public void TradeSell(Characters.NPC.Friend Trader, Items.ItemBase item)

{

Trader.TradeBuy(item);

}

public void GetQuestFromNPC(Characters.NPC.Friend friend)

{

friend.GiveQuest();

}

public void Scenario1()

{

Console.WriteLine("You are in the Armorer Shop");

Armorer.TradeSell(SteelSword);

Armorer.TradeSell(SteelArmour);

\_hero.EquipItem(SteelSword);

\_hero.EquipItem(SteelArmour);

Console.WriteLine("--------------------");

ShowHeroInfo();

Console.WriteLine("--------------------");

GetQuestFromNPC(Armorer);

Console.WriteLine("--------------------");

FightEnemy(OrcWarrior);

}

}

}

Program.cs

namespace RPGGamewithpatternsandso

{

class Program

{

static void Main(string[] args)

{

var Hero = Characters.Hero.GetInstance(Characters.BaseCharacter.ClassSpec.WARRIOR);

GameFacade game = new GameFacade(Hero);

game.Scenario1();

//game.FightEnemy(game.OrcWarrior);

//game.TradeBuy(game.Armorer, new Items.Weapons.SteelSword());

//game.ShowHeroInfo();

Console.ReadKey();

}

}

}