**Лабораторная работа №11.2**

**Задание:**

Реализовать иерархию классов.

1. К классу из лабораторной работы №10 прописать его базовый (класс-родитель) **абстрактный класс** (должно быть слово **abstract**), в котором поля и методы будут использоваться для последующих наследников.

Пример: *Класс Покемон с полями имя, пол, вес, рост и др.*

(Соответственно в классе Псайдак указанные поля для прошлой лабораторной, что будут едины для всех наследников класса Покемон, определять не нужно)

1. От абстрактного класса породить еще два новых класса (помимо уже существующего). Определить для них методы абстрактного класса, добавить конструкторы, использующие конструктор абстрактного класса-родителя (через ключевое слово **base**), еще два-три новых поля и метода, один из которых **виртуальный**. Продемонстрировать работу каждого метода в консоли. Можно с использованием массивов персонажей.

Пример: *Класс Пикачу с дополнительными полями удар молнии, прыжок и др.*

*Класс Слоупок с дополнительными полями замедление, вызов дождя (это оказывается канон :с) и др.*

*Класс Псайдак (из прошлого задания с соответствующими полями).*

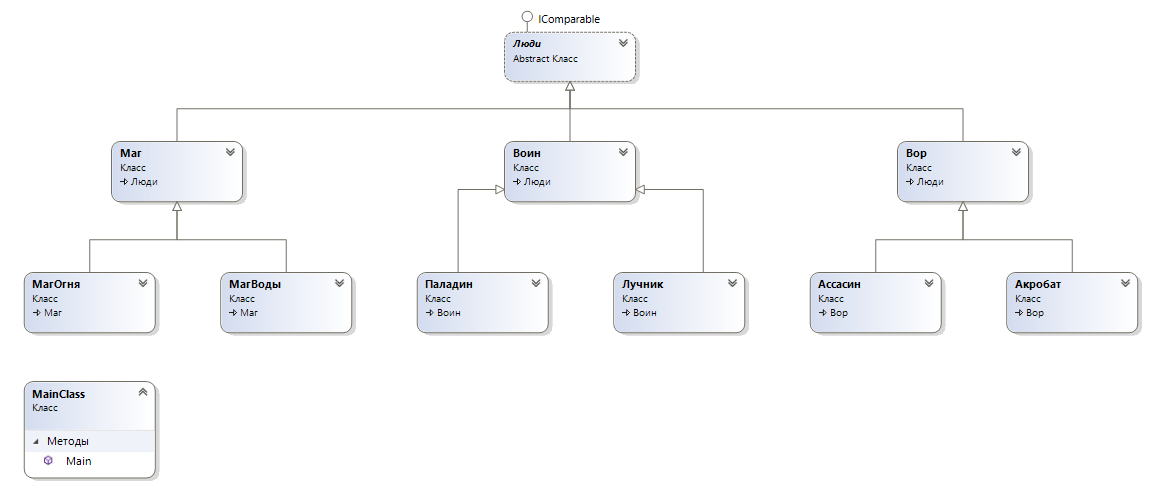
1. От каждого созданного производного класса построить еще как минимум по два наследника, расширяющие возможности этих классов. Определить конструкторы, использующие конструктор класса-родителя (через ключевое слово **base**), добавить новые поля и реализовать виртуальные методы класса-родителя (через ключевое слово **override**). Продемонстрировать работу каждого метода в консоли. Можно с использованием массивов персонажей.

Пример: *Наследники класса Пикачу: Пичу (с дополнительным полем умиление) и Райчу (с дополнительным полем передвижение на хвосте).*

*Наследники класса Слоупок: Слоубро (с дополнительным полем… ну… командная медлительность?) и Слоукинг (с дополнительным полем медлительности на всех участников боя (включая и союзников :с)).*

*+ наследники класса Псайдак.*

**Диаграма:**



**Код:**

using System;

using System.Collections;

namespace Test\_2

{

abstract class Люди : IComparable

{

public string Name { get; set; }

public string RaceClass { get; set; }

public string Type { get; set; }

public uint Age { get; set; }

public int Hp { get; set; }

public int Stamina { get; set; }

public int Magic { get; set; }

public int Force { get; set; }

public int Heal { get; set; }

public int Level { get; set; }

public int Iq { get; set; }

public bool IsAlive { get; set; }

public Люди(string name, uint age, string raceClass)

{

Name = name; Age = age; RaceClass = raceClass;

Hp = 0; Stamina = 0; Magic = 0; Force = 0; Heal = 0; Level = 0; Iq = 0; IsAlive = true;

}

int IComparable.CompareTo(object obj)

{

Люди p = (Люди)obj;

return String.Compare(Age.ToString(), p.Age.ToString());

}

public void CheckingAge()

{

if (Age < 30)

{

Hp += 25; Stamina += 35; Magic += 20; Force += 25; Heal += 10; Iq += 80;

}

else

{

Hp += 15; Stamina += 20; Magic += 40; Force += 15; Heal += 5; Iq += 100;

}

}

abstract public void BuffType();

public void CheckingDeath()

{

if ((Hp <= 0) || (Stamina <= 0) || (Magic <= 0))

{

Console.WriteLine("Смэрть персонажа по имени " + Name + "!");

IsAlive = false;

}

}

abstract public void Battle();

abstract public void Waiting();

public virtual void DisplayParams()

{

Console.WriteLine($"\nИмя: {Name} \nВозраст: {Age} \nКласс: {RaceClass} \nТип: {Type} \nHP: {Hp} \nВыносливость: {Stamina} \nМагия: {Magic} \nСила: {Force} \nУровень: {Level} \nИнтеллект: {Iq} \n");

}

public void DisplaySortChoice()

{

Console.Clear();

Console.WriteLine("Выберите параметр для сортировки.");

Console.WriteLine($"\n1. Возраст\n2. HP\n3. Выносливость\n4. Магия\n5. Сила\n6. Уровень\n7. Интеллект\n");

}

public virtual void DisplayCharacters(int i)

{

Console.Write(i + 1 + ".\n");

DisplayParams();

}

public void HandlingInstruction()

{

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

Console.WriteLine("Инструкция по управлению! \n0. Показать инструкцию повторно. \n1. Выбрать созданного персонажа. \n2. Ожидание. \n3. В бой! \n4. Статистика всех персонажей. \n5. Статистика текущего персонажа. \n6. Cортировка по возрастанию. \n7. Сортировка по убыванию\n10. Выход.");

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

}

//Куча вспомогательных классов для сортировки массива по конкретному полю с использованием интерфейса IComparable

private class SortAgeAscendingHelper : IComparer

{

int IComparer.Compare(object a, object b)

{

Люди p1 = (Люди)a;

Люди p2 = (Люди)b;

if (p1.Age > p2.Age)

return 1;

if (p1.Age < p2.Age)

return -1;

else

return 0;

}

}

private class SortAgeDescendingHelper : IComparer

{

int IComparer.Compare(object a, object b)

{

Люди p1 = (Люди)a;

Люди p2 = (Люди)b;

if (p1.Age < p2.Age)

return 1;

if (p1.Age > p2.Age)

return -1;

else

return 0;

}

}

private class SortHPAscendingHelper : IComparer

{

int IComparer.Compare(object a, object b)

{

Люди p1 = (Люди)a;

Люди p2 = (Люди)b;

if (p1.Hp > p2.Hp)

return 1;

if (p1.Hp < p2.Hp)

return -1;

else

return 0;

}

}

private class SortHPDescendingHelper : IComparer

{

int IComparer.Compare(object a, object b)

{

Люди p1 = (Люди)a;

Люди p2 = (Люди)b;

if (p1.Hp < p2.Hp)

return 1;

if (p1.Hp > p2.Hp)

return -1;

else

return 0;

}

}

private class SortStaminaAscendingHelper : IComparer

{

int IComparer.Compare(object a, object b)

{

Люди p1 = (Люди)a;

Люди p2 = (Люди)b;

if (p1.Stamina > p2.Stamina)

return 1;

if (p1.Stamina < p2.Stamina)

return -1;

else

return 0;

}

}

private class SortStaminaDescendingHelper : IComparer

{

int IComparer.Compare(object a, object b)

{

Люди p1 = (Люди)a;

Люди p2 = (Люди)b;

if (p1.Stamina < p2.Stamina)

return 1;

if (p1.Stamina > p2.Stamina)

return -1;

else

return 0;

}

}

private class SortMagicAscendingHelper : IComparer

{

int IComparer.Compare(object a, object b)

{

Люди p1 = (Люди)a;

Люди p2 = (Люди)b;

if (p1.Magic > p2.Magic)

return 1;

if (p1.Magic < p2.Magic)

return -1;

else

return 0;

}

}

private class SortMagicDescendingHelper : IComparer

{

int IComparer.Compare(object a, object b)

{

Люди p1 = (Люди)a;

Люди p2 = (Люди)b;

if (p1.Magic < p2.Magic)

return 1;

if (p1.Magic > p2.Magic)

return -1;

else

return 0;

}

}

private class SortForceAscendingHelper : IComparer

{

int IComparer.Compare(object a, object b)

{

Люди p1 = (Люди)a;

Люди p2 = (Люди)b;

if (p1.Force > p2.Force)

return 1;

if (p1.Force < p2.Force)

return -1;

else

return 0;

}

}

private class SortForceDescendingHelper : IComparer

{

int IComparer.Compare(object a, object b)

{

Люди p1 = (Люди)a;

Люди p2 = (Люди)b;

if (p1.Force < p2.Force)

return 1;

if (p1.Force > p2.Force)

return -1;

else

return 0;

}

}

private class SortLevelAscendingHelper : IComparer

{

int IComparer.Compare(object a, object b)

{

Люди p1 = (Люди)a;

Люди p2 = (Люди)b;

if (p1.Level > p2.Level)

return 1;

if (p1.Level < p2.Level)

return -1;

else

return 0;

}

}

private class SortLevelDescendingHelper : IComparer

{

int IComparer.Compare(object a, object b)

{

Люди p1 = (Люди)a;

Люди p2 = (Люди)b;

if (p1.Level < p2.Level)

return 1;

if (p1.Level > p2.Level

)

return -1;

else

return 0;

}

}

private class SortIQAscendingHelper : IComparer

{

int IComparer.Compare(object a, object b)

{

Люди p1 = (Люди)a;

Люди p2 = (Люди)b;

if (p1.Iq > p2.Iq)

return 1;

if (p1.Iq < p2.Iq)

return -1;

else

return 0;

}

}

private class SortIQDescendingHelper : IComparer

{

int IComparer.Compare(object a, object b)

{

Люди p1 = (Люди)a;

Люди p2 = (Люди)b;

if (p1.Iq < p2.Iq)

return 1;

if (p1.Iq > p2.Iq)

return -1;

else

return 0;

}

}

public static IComparer sortAgeAscending()

{

return new SortAgeAscendingHelper();

}

public static IComparer sortAgeDescending()

{

return new SortAgeDescendingHelper();

}

public static IComparer sortHPAscending()

{

return new SortHPAscendingHelper();

}

public static IComparer sortHPDescending()

{

return new SortHPDescendingHelper();

}

public static IComparer sortStaminaAscending()

{

return new SortStaminaAscendingHelper();

}

public static IComparer sortStaminaDescending()

{

return new SortStaminaDescendingHelper();

}

public static IComparer sortMagicAscending()

{

return new SortMagicAscendingHelper();

}

public static IComparer sortMagicDescending()

{

return new SortMagicDescendingHelper();

}

public static IComparer sortForceAscending()

{

return new SortForceAscendingHelper();

}

public static IComparer sortForceDescending()

{

return new SortForceDescendingHelper();

}

public static IComparer sortLevelAscending()

{

return new SortLevelAscendingHelper();

}

public static IComparer sortLevelDescending()

{

return new SortLevelDescendingHelper();

}

public static IComparer sortIQAscending()

{

return new SortIQAscendingHelper();

}

public static IComparer sortIQDescending()

{

return new SortIQDescendingHelper();

}

}

class Воин : Люди

{

public int Defence { get; set; }

public int Parry { get; set; }

public Воин(string name, uint age, string raceClass) : base(name, age, raceClass)

{

Defence = 0;

Parry = 0;

}

public override void BuffType()

{

Hp += 50;

Stamina += 20;

Magic += 5;

Force += 30;

Heal += 10;

Iq += 10;

Defence += 10;

Parry += 10;

}

public override void Battle()

{

Hp -= 25 + Heal;

Stamina -= 20;

Magic -= 10;

Force += 5;

Level++;

Iq++;

Defence += 5;

Parry += 5;

CheckingDeath();

}

public override void Waiting()

{

Hp += 10;

Stamina += 10;

Magic += 5;

Force += 5;

Iq++;

Defence += 2;

Parry++;

}

public override void DisplayParams()

{

base.DisplayParams();

Console.WriteLine($"Защита: {Defence} \nПарирование: {Parry} \n");

}

}

class Паладин : Воин

{

public Паладин(string name, uint age, string raceClass) : base(name, age, raceClass)

{

Type = "Паладин";

}

public override void BuffType()

{

Hp += 70;

Stamina += 50;

Magic += 20;

Force += 40;

Heal += 10;

Iq += 10;

Defence += 20;

Parry += 15;

}

public override void Battle()

{

Hp -= 25 + Heal;

Stamina -= 25;

Magic -= 10;

Force += 5;

Level++;

Iq++;

Defence += 5;

Parry += 5;

CheckingDeath();

}

}

class Лучник : Воин

{

public Лучник(string name, uint age, string raceClass) : base(name, age, raceClass)

{

Type = "Лучник";

}

public override void BuffType()

{

Hp += 40;

Stamina += 15;

Magic += 10;

Force += 20;

Heal += 10;

Iq += 15;

Defence += 10;

Parry += 10;

}

public override void Battle()

{

Hp -= 25 + Heal;

Stamina -= 15;

Magic -= 5;

Force += 5;

Level++;

Iq++;

Defence += 5;

Parry += 5;

CheckingDeath();

}

}

class Маг : Люди

{

public int AbilityRange { get; set; }

public int Fortitude { get; set; }

public Маг(string name, uint age, string raceClass) : base(name, age, raceClass)

{

AbilityRange = 0;

Fortitude = 0;

}

public override void BuffType()

{

Hp += 35;

Stamina += 20;

Magic += 50;

Force += 10;

Heal += 5;

Iq += 30;

AbilityRange += 50;

Fortitude += 20;

}

public override void Battle()

{

Hp -= 25 + Heal;

Stamina -= 10;

Magic -= 15;

Force += 5;

Level++;

Iq++;

AbilityRange += 10;

Fortitude += 5;

CheckingDeath();

}

public override void Waiting()

{

Hp += 10;

Stamina += 5;

Magic += 10;

Force += 5;

Iq++;

AbilityRange += 5;

Fortitude++;

}

public override void DisplayParams()

{

base.DisplayParams();

Console.WriteLine($"Дальность способностей: {AbilityRange} \nСтойкость духа: {Fortitude} \n");

}

}

class МагОгня : Маг

{

public МагОгня(string name, uint age, string raceClass) : base(name, age, raceClass)

{

Type = "Маг огня";

}

public override void BuffType()

{

Hp += 70;

Stamina += 35;

Magic += 60;

Force += 10;

Heal += 5;

Iq += 20;

AbilityRange += 40;

Fortitude += 20;

}

public override void Battle()

{

Hp -= 25 + Heal;

Stamina -= 15;

Magic -= 30;

Force += 5;

Level++;

Iq++;

AbilityRange += 5;

Fortitude += 5;

CheckingDeath();

}

}

class МагВоды : Маг

{

public МагВоды(string name, uint age, string raceClass) : base(name, age, raceClass)

{

Type = "Маг воды";

}

public override void BuffType()

{

Hp += 80;

Stamina += 20;

Magic += 60;

Force += 5;

Heal += 5;

Iq += 30;

AbilityRange += 50;

Fortitude += 30;

}

public override void Battle()

{

Hp -= 25 + Heal;

Stamina -= 10;

Magic -= 25;

Force += 5;

Level++;

Iq++;

AbilityRange += 10;

Fortitude += 5;

CheckingDeath();

}

}

class Вор : Люди

{

public int Stealth { get; set; }

public int BreakIn { get; set; }

public Вор(string name, uint age, string raceClass) : base(name, age, raceClass)

{

Stealth = 0;

BreakIn = 0;

}

public override void BuffType()

{

Hp += 30;

Stamina += 40;

Magic += 5;

Force += 20;

Heal += 10;

Iq += 20;

Stealth += 30;

BreakIn += 10;

}

public override void Battle()

{

Hp -= 25 + Heal;

Stamina -= 20;

Magic -= 5;

Force += 5;

Level++;

Iq++;

Stealth += 5;

BreakIn += 5;

CheckingDeath();

}

public override void Waiting()

{

Hp += 10;

Stamina += 15;

Magic += 5;

Force += 5;

Iq++;

Stealth++;

BreakIn++;

}

public override void DisplayParams()

{

base.DisplayParams();

Console.WriteLine($"Скрытность: {Stealth} \nВзлом: {BreakIn} \n");

}

}

class Ассасин : Вор

{

public Ассасин(string name, uint age, string raceClass) : base(name, age, raceClass)

{

Type = "Ассасин";

}

public override void BuffType()

{

Hp += 50;

Stamina += 30;

Magic += 10;

Force += 25;

Heal += 10;

Iq += 30;

Stealth += 30;

BreakIn += 10;

}

public override void Battle()

{

Hp -= 25 + Heal;

Stamina -= 20;

Magic -= 5;

Force += 5;

Level++;

Iq++;

Stealth += 5;

BreakIn += 5;

CheckingDeath();

}

}

class Акробат : Вор

{

public Акробат(string name, uint age, string raceClass) : base(name, age, raceClass)

{

Type = "Акробат";

}

public override void BuffType()

{

Hp += 70;

Stamina += 50;

Magic += 5;

Force += 10;

Heal += 5;

Iq += 20;

Stealth += 25;

BreakIn += 15;

}

public override void Battle()

{

Hp -= 25 + Heal;

Stamina -= 25;

Magic -= 5;

Force += 5;

Level++;

Iq++;

Stealth += 5;

BreakIn += 5;

CheckingDeath();

}

}

class MainClass

{

public static void Main(string[] args)

{

try

{

string name;

string raceClass = "";

string type = "";

uint age;

string choice = "1";

int quantity = 0;

int characterChoice = 0;

Console.WriteLine("Привет! Сколько персонажей ты хочешь сделать?"); quantity = int.Parse(Console.ReadLine());

Люди[] people = new Люди[quantity];

Люди mainCharacter;

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

{

Console.WriteLine($"Дай {i + 1}-му несколько параметров:");

Console.Write($"{i + 1}-е имя: "); name = Console.ReadLine();

while (name == string.Empty)

{

Console.WriteLine("Введи всё же имя персонажу.");

Console.Write($"{i + 1}-е имя: "); name = Console.ReadLine();

}

Console.Write($"{i + 1}-й возраст: "); age = Convert.ToUInt32(Console.ReadLine());

while (raceClass != "воин" && raceClass != "маг" && raceClass != "вор")

{

Console.Write($"{i + 1}-й класс(воин, маг, вор): ");

raceClass = Console.ReadLine();

}

while (type != "паладин" && type != "лучник" && type != "маг огня" && type != "маг воды" && type != "ассасин" && type != "акробат")

{

switch (raceClass)

{

case "воин": Console.Write($"{i + 1}-й тип(паладин или лучник): "); type = Console.ReadLine(); break;

case "маг": Console.Write($"{i + 1}-й тип(маг огня или маг воды): "); type = Console.ReadLine(); break;

case "вор": Console.Write($"{i + 1}-й тип(ассасин или акробат): "); type = Console.ReadLine(); break;

}

}

switch (type)

{

case "паладин": people[i] = new Паладин(name, age, raceClass); break;

case "лучник": people[i] = new Лучник(name, age, raceClass); break;

case "маг огня": people[i] = new МагОгня(name, age, raceClass); break;

case "маг воды": people[i] = new МагВоды(name, age, raceClass); break;

case "ассасин": people[i] = new Ассасин(name, age, raceClass); break;

case "акробат": people[i] = new Акробат(name, age, raceClass); break;

}

people[i].CheckingAge();

people[i].BuffType();

name = "";

raceClass = "";

type = "";

}

Console.Clear();

mainCharacter = people[0];

Console.WriteLine("Пока ты управляешь персонажем с именем " + mainCharacter.Name);

Console.WriteLine("Ну что ж, действуй! Выбери, что ты хочешь сделать.");

while ((choice != "10") && mainCharacter.IsAlive)

{

Console.WriteLine("\nЧто дальше?\n");

people[0].HandlingInstruction();

choice = Console.ReadLine();

if (choice == "0")

{

Console.Clear();

mainCharacter.HandlingInstruction();

}

else if (choice == "1")

{

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

{

people[i].DisplayCharacters(i);

}

Console.WriteLine("И кого же ты хочешь выбрать?"); characterChoice = int.Parse(Console.ReadLine());

mainCharacter = people[characterChoice - 1];

}

else if (choice == "2")

{

mainCharacter.Waiting();

}

else if (choice == "3")

{

mainCharacter.Battle();

}

else if (choice == "4")

{

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

{

people[i].DisplayCharacters(i);

}

}

else if (choice == "5")

{

mainCharacter.DisplayParams();

}

else if (choice == "6")

{

mainCharacter.DisplaySortChoice();

switch (int.Parse(Console.ReadLine()))

{

case 1: Array.Sort(people, Люди.sortAgeAscending()); break;

case 2: Array.Sort(people, Люди.sortHPAscending()); break;

case 3: Array.Sort(people, Люди.sortStaminaAscending()); break;

case 4: Array.Sort(people, Люди.sortMagicAscending()); break;

case 5: Array.Sort(people, Люди.sortForceAscending()); break;

case 6: Array.Sort(people, Люди.sortLevelAscending()); break;

case 7: Array.Sort(people, Люди.sortIQAscending()); break;

default:

Console.WriteLine("Попробуй выбрать предложенное!");

break;

}

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

{

people[i].DisplayCharacters(i);

}

}

else if (choice == "7")

{

mainCharacter.DisplaySortChoice();

switch (int.Parse(Console.ReadLine()))

{

case 1: Array.Sort(people, Люди.sortAgeDescending()); break;

case 2: Array.Sort(people, Люди.sortHPDescending()); break;

case 3: Array.Sort(people, Люди.sortStaminaDescending()); break;

case 4: Array.Sort(people, Люди.sortMagicDescending()); break;

case 5: Array.Sort(people, Люди.sortForceDescending()); break;

case 6: Array.Sort(people, Люди.sortLevelDescending()); break;

case 7: Array.Sort(people, Люди.sortIQDescending()); break;

default:

Console.WriteLine("Попробуй выбрать предложенное!");

break;

}

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

{

people[i].DisplayCharacters(i);

}

}

else if (choice == "10")

{

break;

}

else

{

Console.WriteLine("Попробуй выбрать предложенное!");

}

Console.ReadKey();

Console.Clear();

}

Console.WriteLine("До встречи!");

Console.ReadLine();

}

catch (Exception)

{

Console.WriteLine("Ты что-то ввёл не так, попробуй ещё раз!");

}

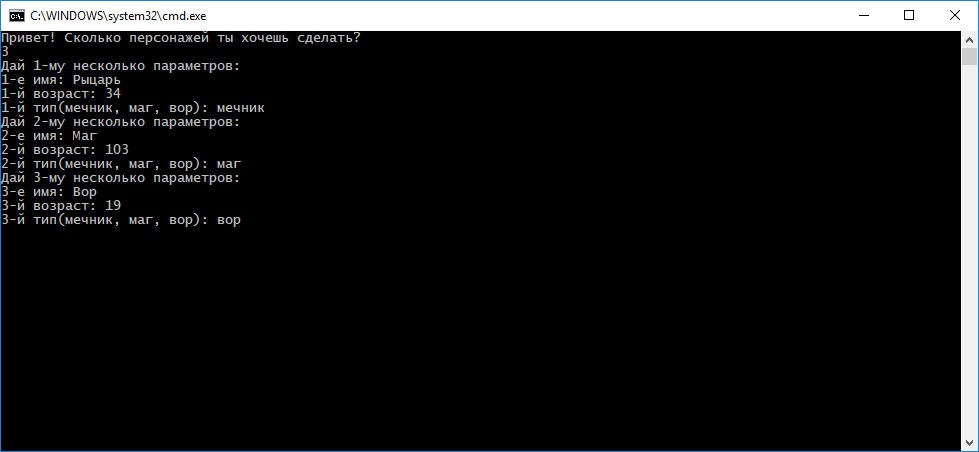
}

}

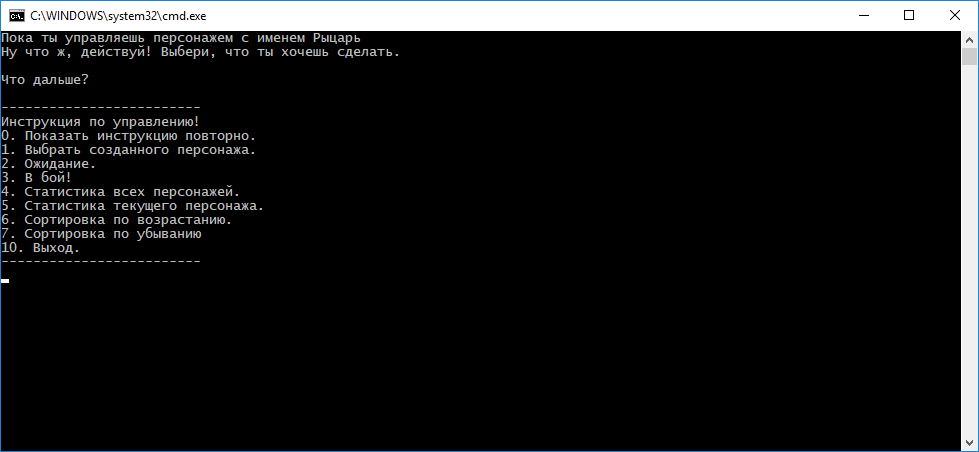
}

**Скриншоты:**

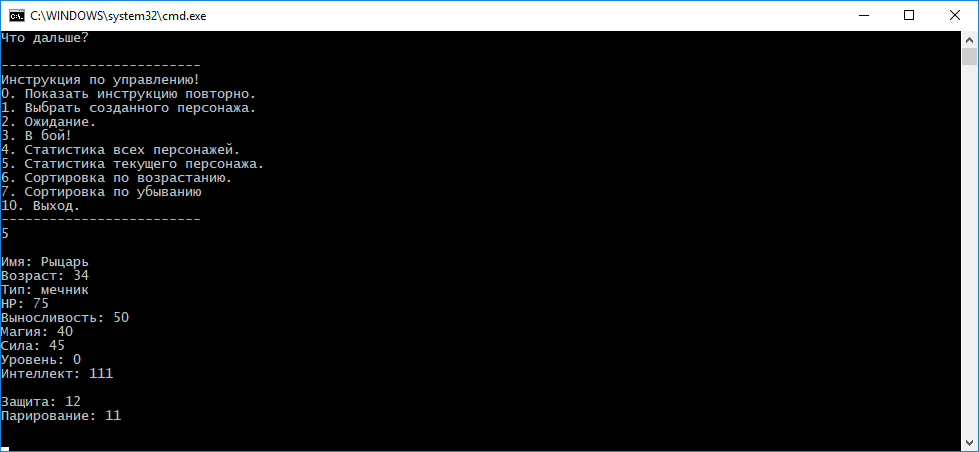
**//**у каждого типа есть свой буст к характеристикам, у мечника буст к хп, у мага к магии и у вора к выносливости соответсвенно, также буст зависит от возраста, чем старше, тем лучше ментальные навыки, чем моложе – физические



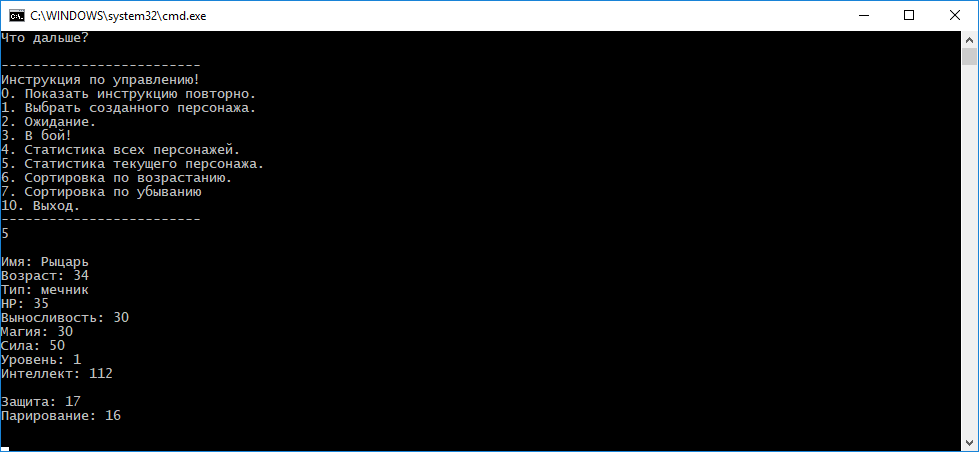
//тыкаем Enter, чтобы продолжить дальше



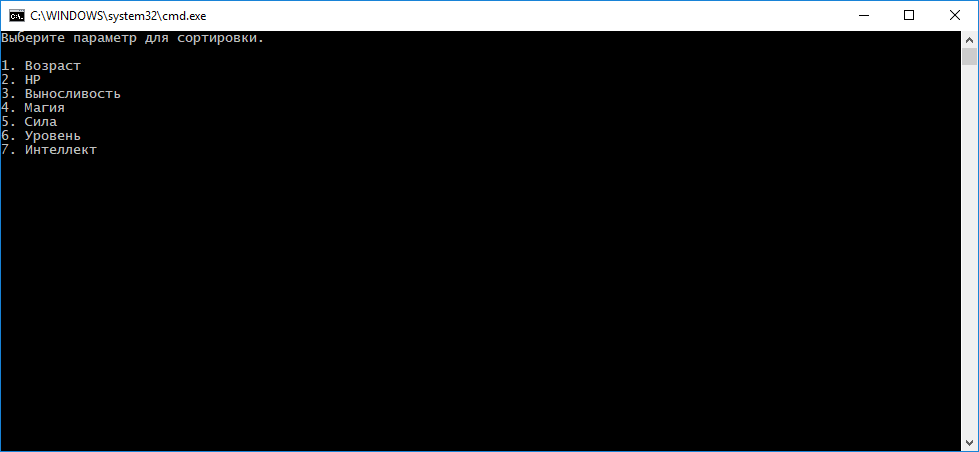
//после Ожидания(2) у персонажа меняются характеристики, в том числе индивидуальные, для каждого Типа



//после В бой!(3) у персонажа отнимается ХП, выносливость и магия, остальные хар-ки повышаются



//при выборе сортировки(6 или 7) появляется данное меню с выбором параметра



//например, по возрастанию возраста

