|  |
| --- |
| Пермский филиал федерального государственного автономного образовательного учреждения высшего образования  «Национальный исследовательский университет  «Высшая школа экономики»  *Факультет экономики, менеджмента и бизнес-информатики* |
|  |
| Меньшиков Олег Владимирович  по направлению подготовки *09.03.04 Программная инженерия*  образовательная программа «Программная инженерия»   |  |  | | --- | --- | |  | Доцент к[афедры информационных технологий в бизнесе](https://perm.hse.ru/bi/infobis/), кандидат технических наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  О.Л. Викентьева | |

**Оглавление**

[Постановка задач 3](#_Toc52393374)

[Задача 1 3](#_Toc52393375)

[Задача 2 3](#_Toc52393376)

[Задача 3 3](#_Toc52393377)

[Анализ 3](#_Toc52393378)

[Задача 1 3](#_Toc52393379)

[Задача 2 3](#_Toc52393380)

[Задача 3 4](#_Toc52393381)

[Блок-схемы 5](#_Toc52393382)

[Задача 1 5](#_Toc52393383)

[Задача 2 7](#_Toc52393384)

[Задача 3 8](#_Toc52393385)

[Текст программы 9](#_Toc52393386)

[Задача 1 9](#_Toc52393387)

[Задача 2 10](#_Toc52393388)

[Задача 3 11](#_Toc52393389)

**Вариант 10**

# **Постановка задач**

Часть 1

**Основное содержание работы.**

Написать программу, в которой создается иерархия классов. Записать объекты классов в массив, выполнить просмотр элементов массива. Показать использование виртуальных функций.

**Порядок выполнения работы.**

1. Определить иерархию классов (в соответствии с вариантом).
2. Реализовать классы.
3. Написать демонстрационную программу, в которой создаются объекты различных классов и помещаются в массив, после чего массив просматривается.
4. Реализовать 2 варианта программы: с помощью виртуальных и не виртуальных методов.
5. Без виртуальных функций программа будет работать неправильно! Объяснить почему. Объяснить необходимость виртуальных функций

Вариант 10:

**персона**, студент, школьник, студент-заочник;

Часть 2

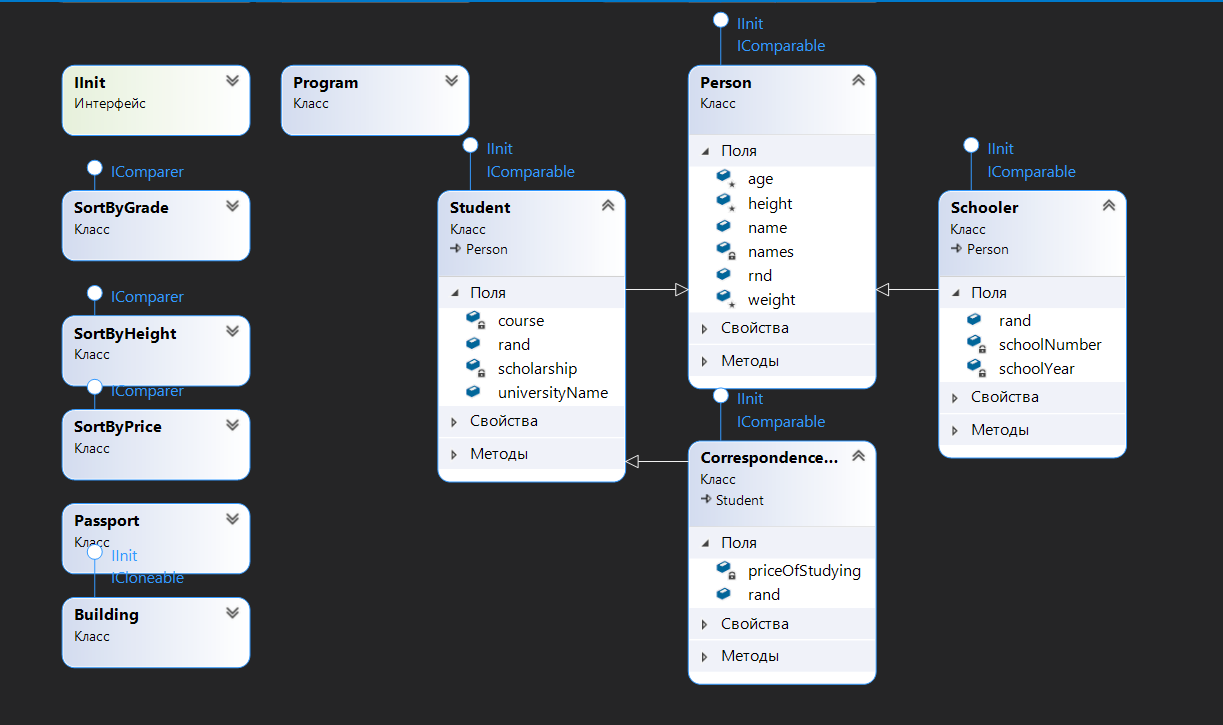
**Основное содержание работы.**

1. Реализовать метод для выполнения заданных запросов. При необходимости (для выполнения запроса) в класс могут быть добавлены новые поля (по сравнению с частью 1). В программе должно быть минимум ввода с клавиатуры. Поля объектов задаются в тексте программы. С клавиатуры вводятся только параметры запроса.
2. Реализовать не менее трех запросов, соответствующих иерархии классов (можно реализовать свои запросы).

Часть 3

1. Составить иерархию классов в соответствии с вариантом. Иерархия должна содержать хотя бы один интерфейс и хотя бы один абстрактный класс.
2. Создать массив интерфейсных элементов и поместить в него экземпляры различных классов иерархии.
3. Реализовать сортировку элементов массива, используя стандартные интерфейсы и методы класса Array.
4. Реализовать поиск элемента в массиве, используя стандартные интерфейсы и методы класса Array.
5. Реализовать в одном из классов метод клонирования объектов. Показать клонирование объектов.

# **Анализ**



# **Листинг**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace labor10

{

public class Building : IInit, ICloneable

{

private int year;

public int Year

{

get

{

return year;

}

set

{

if (value > 2021)

year = 2021;

else if (value < 0)

year = 0;

else

year = value;

}

}

public string name;

public Passport pas;

static Random rnd = new Random();

private int countFloors;

public int CountFloors

{

get

{

return countFloors;

}

set

{

if (value <= 0)

countFloors = 1;

else

countFloors = value;

}

}

//конструкторы

public Building()

{

this.year = 1;

this.name = "NoName";

this.countFloors = 1;

this.pas = new Passport(rnd.Next(1000, 10000));

}

public Building(int year, string name, int floors, int num)

{

this.year = year;

this.name = name;

this.countFloors = floors;

this.pas = new Passport(num);

}

public object Init()

{

string[] names = new string[] { "Tower", "Castle", "House", "Residence", "Apartment", "Shack", "Bungalow" };

string[] lastNames = new string[] { "A.", "J.", "P.", "M.", "B.", "R.", "G.", "E.", "N.", "O." };

Building b = new Building(rnd.Next(1, 2022), names[rnd.Next(0, names.Length)] + " " + lastNames[rnd.Next(0, names.Length)],

rnd.Next(1, 101), rnd.Next(1000, 10000));

return b;

}

public override string ToString()

{

return this.name + ", год постройки: " + this.year + ", количество этажей: " + this.countFloors + " Паспорт: " + this.pas.sery;

}

public override bool Equals(object obj)

{

if (obj is Building)

{

Building s = (Building)obj;

return (this.year == s.Year) && (this.name == s.name);

}

return false;

}

//клонирование

public object Clone()

{

return new Building(this.year, this.name, this.countFloors, this.pas.sery);

}

//поверхностное копирование

public object SwallowCopy()

{

return this.MemberwiseClone();

}

}

}

namespace labor10

{

public class CorrespondenceStudent : Student, IInit, IComparable

{

private int priceOfStudying;

public int PriceOfStudying

{

get { return priceOfStudying; }

set

{

if (value < 0)

Console.WriteLine("Цена обучения не может быть отрицательной!");

else

priceOfStudying = value;

}

}

public static Random rand = new Random();

public CorrespondenceStudent() : base()

{

this.priceOfStudying = 0;

}

public CorrespondenceStudent(int age, int height, int weight, int course, string name,

string universityName, int price, int scholarship)

: base(age, height, weight, course, name, universityName, scholarship)

{

this.priceOfStudying = price;

}

//методы

public new void ShowSimple()

{

Console.WriteLine("Информация об объекте класса CorrespondenceStudent:");

Console.WriteLine($"Имя: {this.name}; Курс: {this.Course}; Университет: {this.universityName}; " +

$"Цена обучения: {this.priceOfStudying}; Возраст: {this.age}; Рост: {this.height}; Вес: {this.weight};");

}

override public void ShowVirt()

{

Console.WriteLine("Информация об объекте класса CorrespondenceStudent:");

Console.WriteLine($"Имя: {this.name}; Курс: {this.Course}; Университет: {this.universityName}; " +

$"Цена обучения: {this.priceOfStudying}; Возраст: {this.age}; Рост: {this.height}; Вес: {this.weight};");

}

//от интерфейса

public override object Init()

{

Person p = (Person)base.Init();

string[] names = new string[] { "Smith", "John", "Peterson", "McCourty", "Brady", "Brown", "Gronkowski", "Evans", "Arians", "Godwin" };

string[] lastNames = new string[] { "A.", "J.", "P.", "M.", "B.", "R.", "G.", "E.", "N.", "O." };

string[] universities = new string[] { "HSE", "PSU" };

CorrespondenceStudent corrStud = new CorrespondenceStudent(p.Age, p.Height, p.Weight, rand.Next(1, 5),

names[rand.Next(0, names.Length)] + " " + lastNames[rand.Next(0, names.Length)],

universities[rand.Next(0, universities.Length)], rand.Next(80000, 120001), 0);

return corrStud;

}

//перегрузки

public override string ToString()

{

return name + " из " + universityName + " весом " + weight + "кг с " + Course + " курса платит за учебу " + priceOfStudying;

}

public override bool Equals(object obj)

{

if (obj is CorrespondenceStudent)

{

CorrespondenceStudent s = (CorrespondenceStudent)obj;

return (this.age == s.Age) && (this.height == s.Height) && (this.weight == s.Weight) && (this.Course == s.Course)

&& (this.name == s.name) && (this.universityName == s.universityName) && (this.priceOfStudying == s.PriceOfStudying)

&& (this.Scholarship == s.Scholarship);

}

return false;

}

public override int CompareTo(object obj)

{

if (obj is CorrespondenceStudent)

{

CorrespondenceStudent s = (CorrespondenceStudent)obj;

return this.name.CompareTo(s.name);

}

else

throw new Exception("Невозможно сравнить два объекта");

}

public override int GetHashCode()

{

return base.GetHashCode();

}

}

}

namespace labor10

{

interface IInit

{

object Init();

}

}

namespace labor10

{

public class Passport

{

private int s;

public int sery

{

get { return s; }

set

{

if ((value < 0) || (value > 9999))

s = 0;

else

s = value;

}

}

public Passport(int s)

{

this.s = s;

}

}

}

namespace labor10

{

public class Person: IInit, IComparable

{

//поля

protected int age;

public int Age

{

get { return age; }

set

{

if (value < 0)

Console.WriteLine("Возраст не может быть меньше нуля!");

else

age = value;

}

}

protected int height;

public int Height

{

get { return height; }

set

{

if (value < 0)

Console.WriteLine("Рост не может быть меньше нуля!");

else

height = value;

}

}

protected int weight;

public int Weight

{

get { return weight; }

set

{

if (value < 0)

Console.WriteLine("Возраст не может быть меньше нуля!");

else

weight = value;

}

}

public string name;

public static Random rnd = new Random();

//конструкторы

public Person()

{

this.name = "NoName";

this.age = 0;

this.height = 0;

this.height = 0;

}

public Person(string name, int age, int height, int weight)

{

this.name = name;

this.age = age;

this.height = height;

this.weight = weight;

}

//методы

public void ShowSimple()

{

Console.WriteLine("Информация об объекте класса Person:");

Console.WriteLine($"Имя: {this.name}; Возраст: {this.age}; Рост: {this.height}; Вес: {this.weight};");

}

virtual public void ShowVirt()

{

Console.WriteLine("Информация об объекте класса Person:");

Console.WriteLine($"Имя: {this.name}; Возраст: {this.age}; Рост: {this.height}; Вес: {this.weight};");

}

string[] names = new string[] { "Ivanov", "Smirnov", "Petrov", "Vasyechkin", "Fhilimonov", "Belov", "Anyechkin", "Kornev" };

//методы интерфейса

public virtual object Init()

{

Person p = new Person(names[rnd.Next(0, names.Length)], rnd.Next(10, 25), rnd.Next(150, 190), rnd.Next(50, 100));

return p;

}

//перегрузки

public override string ToString()

{

return "Имя: " + name + " Возраст: " + age + " Вес: " + weight + " Рост: " + height;

}

public override bool Equals(object obj)

{

if (obj is Person)

{

Person s = (Person)obj;

return (this.name == s.name) && (this.age == s.Age) && (this.height == s.Height) && (this.weight == s.Weight);

}

return false;

}

public virtual int CompareTo(object obj)

{

if (obj is Person)

{

Person s = (Person)obj;

return this.age.CompareTo(s.Age);

}

else

throw new Exception();

}

}

}

namespace labor10

{

class Program

{

static int ParseInt(string input)

{

int n;

bool ok;

Console.Write(input);

do

{

string str = Console.ReadLine();

ok = int.TryParse(str, out n);

if (!ok)

Console.WriteLine("Число должно быть целым!");

}

while (!ok);

return n;

}

static double ParseDouble(string input)

{

double n;

bool ok;

Console.WriteLine(input);

do

{

string str = Console.ReadLine();

ok = double.TryParse(str, out n);

if (!ok)

Console.WriteLine("Число должно быть целым!");

}

while (!ok);

return n;

}

//часть 2

static void Menu()

{

Console.WriteLine(" Меню запросов");

Console.WriteLine("1. Количество учащихся в 133 школе.");

Console.WriteLine("2. Суммарная стипендия всех студентов в университете HSE.");

Console.WriteLine("3. Количество студентов-заочников на 3 курсе в университете PSU.");

Console.WriteLine("4. Средний рост всех учащихся в 135 школе.");

Console.WriteLine("5. Максимальная цена обучения в университете PSU.");

Console.WriteLine("6. Минимальный возраст студента университета HSE.");

Console.WriteLine("7. Средний вес студентов-заочников в университете PSU.");

Console.WriteLine("8. Переход к части №3.");

}

static void Output1()

{

Console.WriteLine("Этот раздел посвящен демонстрации работы виртуальных методов в наследовании классов.\n" +

"(В дальнейшем, чтобы перейти к следующему шагу, нажмите пробел).");

Console.ReadKey();

Console.WriteLine();

Console.WriteLine("<> Создадим 4 объекта: базовый(Person) и 3 производных(Student, Schooler, CorrespondenceStudent).");

}

static void Output2()

{

Console.ReadKey();

Console.WriteLine();

Console.WriteLine("<> Также создадим массив объектов типа Person и добавим в него все наши 4 объекта.");

Console.ReadKey();

Console.WriteLine();

Console.WriteLine("<> Через foreach вызовем НЕПЕРЕОПРЕДЕЛЕННЫЙ(невиртуальный) метод ShowSimple() для каждого объекта:");

Console.ReadKey();

Console.WriteLine();

}

static void Output3()

{

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

Console.ReadKey();

Console.WriteLine();

Console.WriteLine("Вывод: когда вызывается невиртуальный метод, то для каждого объекта нашего массива типа Person[]\n" +

"вызывается метод ShowSimple() базового класса Person.");

Console.ReadKey();

Console.WriteLine();

Console.WriteLine("<> Через foreach вызовем ПЕРЕОПРЕДЕЛЕННЫЙ(виртуальный) метод ShowVirt():");

Console.ReadKey();

Console.WriteLine();

}

static void Output4()

{

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

Console.ReadKey();

Console.WriteLine();

Console.WriteLine("Вывод: когда вызывается виртуальный метод, то для каждого объекта нашего массива типа Person[]\n" +

"вызывается метод ShowVirt() соответствующего класса определенного объекта нашего массива.\n\n");

Console.Write("Нажмите любую клавишу, чтобы перейти к части №2... ");

Console.ReadKey();

}

static int CountSchoolesIn133School(IInit[] arr)

{

int numberOfSchool = 133;

int count = 0;

foreach(IInit p in arr)

{

if (p is Schooler)

{

Schooler sch = (Schooler)p;

if (sch.SchoolNumber == numberOfSchool)

count++;

}

}

Console.WriteLine($"В 133 школе учатся {count} детей.");

return count;

}

static int AmountScholarshipInHSE(IInit[] arr)

{

string university = "HSE";

int count = 0;

foreach (IInit p in arr)

{

if (p is Student)

{

Student stud = (Student)p;

if (Equals(university, stud.universityName))

count += stud.Scholarship;

}

}

Console.WriteLine($"В университете HSE студенты суммарно получают стипендию, равную {count}.");

return count;

}

static int CorrespStudOf3CourseInPSU(IInit[] arr)

{

string university = "PSU";

int count = 0;

foreach (IInit p in arr)

{

if (p is CorrespondenceStudent)

{

CorrespondenceStudent corrStud = (CorrespondenceStudent)p;

if ((corrStud.universityName == university) && (corrStud.Course == 3))

count++;

}

}

Console.WriteLine($"В университете PSU количество 3-курсников равно {count}.");

return count;

}

static int AverageHeightIn135School(IInit[] arr)

{

int numberOfSchool = 135;

int count = 0;

int commonHeight = 0;

foreach (IInit p in arr)

{

if (p is Schooler)

{

Schooler sch = (Schooler)p;

if (sch.SchoolNumber == numberOfSchool)

{

commonHeight += sch.Height;

count++;

}

}

}

if (count == 0)

{

Console.WriteLine($"В 135 школе не нашлось учеников.");

return 0;

}

else

{

Console.WriteLine($"В 135 школе средний рост учеников = {commonHeight / count}.");

return commonHeight / count;

}

}

static int MaxPriceOfEducatInPSU(IInit[] arr)

{

string university = "PSU";

int max = -1;

foreach (IInit p in arr)

{

if (p is CorrespondenceStudent)

{

CorrespondenceStudent stud = (CorrespondenceStudent)p;

if (Equals(university, stud.universityName))

{

if (stud.PriceOfStudying > max)

max = stud.PriceOfStudying;

}

}

}

if (max == -1)

{

Console.WriteLine("В университете PSU не нашлось студентов-заочников.");

return 0;

}

else

{

Console.WriteLine($"В университете PSU максимальная цена обучения = {max}.");

return max;

}

}

static int MinStudAgeInHSE(IInit[] arr)

{

string university = "HSE";

int min = int.MaxValue;

foreach (IInit p in arr)

{

if (p is Student)

{

Student stud = (Student)p;

if (Equals(university, stud.universityName))

if (stud.Age < min)

min = stud.Age;

}

}

if (min == int.MaxValue)

{

Console.WriteLine("В HSE не нашлось студентов.");

return 0;

}

else

{

Console.WriteLine($"Возраст самого молодого студента в HSE {min}.");

return min;

}

}

static int AverageWeightStudInPSU(IInit[] arr)

{

string university = "PSU";

int count = 0;

int commonWeight = 0;

foreach (IInit p in arr)

{

if (p is CorrespondenceStudent)

{

CorrespondenceStudent corrStud = (CorrespondenceStudent)p;

if (corrStud.universityName == university)

{

commonWeight += corrStud.Weight;

count++;

}

}

}

if (count == 0)

{

Console.WriteLine("В PSU не нашлось студентов.");

return 0;

}

else

{

Console.WriteLine($"Средний вес студента в PSU {commonWeight / count}.");

return commonWeight / count;

};

}

static void DemonstrCloneAndCopy()

{

Building s1 = new Building();

s1 = (Building)s1.Init();

Console.WriteLine("Попробуем разобраться в разнице между копированием и клонированием!");

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

Console.WriteLine("1.1 Создадим объект типа Building: ");

Console.WriteLine("<> Оригинал: " + s1);

Console.WriteLine("1.2 После клонирования(в клоне мы изменили имя и паспорт): ");

Building clone = (Building)s1.Clone();

clone.name = "Clone" + clone.name;

clone.pas.sery = 5555;

Console.WriteLine("<> Оригинал: " + s1);

Console.WriteLine("<> Клон: " + clone);

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

Console.WriteLine("Вывод: Когда мы что-то меняем у клона, то в оригинале это не меняется(даже если это ссылочный тип)," +

"\n - в этом суть глубокого клонирования.");

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

Building b1 = new Building();

b1 = (Building)b1.Init();

Console.WriteLine("2.1 Создадим объект типа Building: ");

Console.WriteLine("<> Оригинал: " + b1);

Console.WriteLine("2.2 После копирования(в копии мы изменили имя и паспорт): ");

Building copy = (Building)b1.SwallowCopy();

copy.name = "Clone" + copy.name;

copy.pas.sery = 7777;

Console.WriteLine("<> Оригинал: " + b1);

Console.WriteLine("<> Клон: " + copy);

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

Console.WriteLine("Вывод: Когда мы меняем у копии поле ссылочного типа, то и в оригинале это поле так же меняется\n" +

" - в этом суть поверхностного копирования.");

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

}

static IInit[] CreateRandomArray()

{

IInit[] arr2 = new IInit[30];

for (int i = 0; i < 30; i++)

{

if (i < 10)

{

Student stud = new Student();

stud = (Student)stud.Init();

arr2[i] = stud;

}

else if (i < 20)

{

CorrespondenceStudent corStud = new CorrespondenceStudent();

corStud = (CorrespondenceStudent)corStud.Init();

arr2[i] = corStud;

}

else if (i < 30)

{

Schooler sch = new Schooler();

sch = (Schooler)sch.Init();

arr2[i] = sch;

}

}

return arr2;

}

static void OutputIInitArray(IInit[] arr2)

{

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

for (int j = 0; j < arr2.Length; j++)

{

if (j == 10) Console.WriteLine("---------------------------------------------------------------------------------------------------------");

if (j == 20) Console.WriteLine("---------------------------------------------------------------------------------------------------------");

Console.WriteLine($"{j + 1}) {arr2[j]}");

}

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

}

static void BinarySearchDemonstr()

{

Schooler[] arrStud = new Schooler[7];

for (int i = 0; i < 7; i++)

{

Schooler s = new Schooler();

arrStud[i] = (Schooler)s.Init();

}

Console.WriteLine("\nМассив элементов типа Schooler:");

for(int i = 0; i < arrStud.Length; i++)

Console.WriteLine($"{i+1})" + arrStud[i]);

Array.Sort(arrStud);

Console.WriteLine("\nОтсортированный массив элементов типа Schooler:");

for (int i = 0; i < arrStud.Length; i++)

Console.WriteLine($"{i + 1})" + arrStud[i]);

Console.Write("Поиск по имени - напишите имя школьника, номер которого вы хотите найти:");

string name = Console.ReadLine();

Schooler temp = new Schooler(0, 0, 0, 5, name, 133);

int pos = Array.BinarySearch(arrStud, temp);

if (pos >= 0)

Console.WriteLine($"Индекс элемента в массиве: {pos + 1}\n");

else

Console.WriteLine("Школьника с таким именем нет в массиве\n");

}

//часть 3

static void Menu3()

{

Console.WriteLine(" Меню запросов");

Console.WriteLine("1. Массив типа IInit[], сожержащий новый класс Builder.");

Console.WriteLine("2. Метод Sort с интерфейсом IComparable(по имени).");

Console.WriteLine("3. Метод Sort с интерфейсом IComparer(по цене за обучение).");

Console.WriteLine("4. Метод Sort с интерфейсом IComparer(по классу в школе).");

Console.WriteLine("5. Поиск с помощью BinarySearch.");

Console.WriteLine("6. Работа с интерфейсом ICloneable.");

Console.WriteLine("7. Завершить работу программы.");

}

static IInit[] ArrayOfIInit()

{

Console.WriteLine("\nМассив содержит объекты разных классов(в том числе новый класс Builder), но их объединяет интефейс IInit, " +

"\nпоэтому мы можем записать их все в массив типа IInit[]:");

IInit[] arr3 = new IInit[20];

for (int i = 0; i < 20; i++)

{

if (i < 5)

{

Student stud = new Student();

stud = (Student)stud.Init();

arr3[i] = stud;

}

else if (i < 10)

{

CorrespondenceStudent corStud = new CorrespondenceStudent();

corStud = (CorrespondenceStudent)corStud.Init();

arr3[i] = corStud;

}

else if (i < 15)

{

Schooler sch = new Schooler();

sch = (Schooler)sch.Init();

arr3[i] = sch;

}

else if (i < 20)

{

Building b = new Building();

b = (Building)b.Init();

arr3[i] = b;

}

}

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

for (int j = 0; j < arr3.Length; j++)

{

if (j == 5) Console.WriteLine("---------------------------------------------------------------------------------------------------------");

if (j == 10) Console.WriteLine("---------------------------------------------------------------------------------------------------------");

if (j == 15) Console.WriteLine("---------------------------------------------------------------------------------------------------------");

Console.WriteLine($"{j + 1}) {arr3[j]}");

}

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

return arr3;

}

static Student[] DemonstrIComparable()

{

Student[] arrStud = new Student[7];

for (int i = 0; i < 7; i++)

{

Student s = new Student();

arrStud[i] = (Student)s.Init();

}

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

Console.WriteLine("До сортировки:");

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

foreach (var stud in arrStud)

Console.WriteLine(stud);

Array.Sort(arrStud);

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

Console.WriteLine("После сортировки по имени(сортировка использует реализацию стандартного интерфейса IComparable):");

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

foreach (var stud in arrStud)

Console.WriteLine(stud);

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

return arrStud;

}

static CorrespondenceStudent[] DemonstrSortByPrice()

{

CorrespondenceStudent[] arrCor = new CorrespondenceStudent[7];

for (int i = 0; i < 7; i++)

{

CorrespondenceStudent s = new CorrespondenceStudent();

arrCor[i] = (CorrespondenceStudent)s.Init();

}

Console.WriteLine("До сортировки:");

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

foreach (var stud in arrCor)

Console.WriteLine(stud);

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

Array.Sort(arrCor, new SortByPrice());

Console.WriteLine("После сортировки по цене обучения:");

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

foreach (var stud in arrCor)

Console.WriteLine(stud);

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

return arrCor;

}

static Schooler[] DemonstrSortByGrade()

{

Schooler[] arrStud = new Schooler[7];

for (int i = 0; i < 7; i++)

{

Schooler s = new Schooler();

arrStud[i] = (Schooler)s.Init();

}

Console.WriteLine("До сортировки:");

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

foreach (var stud in arrStud)

Console.WriteLine(stud);

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

Array.Sort(arrStud, new SortByGrade());

Console.WriteLine("После сортировки по классам в школе:");

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

foreach (var stud in arrStud)

Console.WriteLine(stud);

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

return arrStud;

}

static Person[] DemonstrSortByHeight()

{

Person[] arrStud = new Person[7];

for (int i = 0; i < 7; i++)

{

Person s = new Person();

arrStud[i] = (Person)s.Init();

}

Console.WriteLine("До сортировки:");

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

foreach (var stud in arrStud)

Console.WriteLine(stud);

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

Array.Sort(arrStud, new SortByHeight());

Console.WriteLine("После сортировки по росту:");

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

foreach (var stud in arrStud)

Console.WriteLine(stud);

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

return arrStud;

}

static IInit[] CreateArrPart3()

{

IInit[] arr2 = new IInit[20];

for (int i = 0; i < 20; i++)

{

if (i < 5)

{

Student stud = new Student();

stud = (Student)stud.Init();

arr2[i] = stud;

}

else if (i < 10)

{

CorrespondenceStudent corStud = new CorrespondenceStudent();

corStud = (CorrespondenceStudent)corStud.Init();

arr2[i] = corStud;

}

else if (i < 15)

{

Person sch = new Person();

sch = (Person)sch.Init();

arr2[i] = sch;

}

else if (i < 20)

{

Building b = new Building();

b = (Building)b.Init();

arr2[i] = b;

}

}

return arr2;

}

static void Main(string[] args)

{

Console.WriteLine("==========================================Лабораторная №10==============================================\n");

Console.WriteLine(" ЧАСТЬ №1");

Output1();

Person per = new Person();

per = (Person)per.Init();

Student student = new Student();

student = (Student)student.Init();

Schooler schooler = new Schooler();

schooler = (Schooler)schooler.Init();

CorrespondenceStudent correspStudent = new CorrespondenceStudent();

correspStudent = (CorrespondenceStudent)correspStudent.Init();

Output2();

Person[] arrSimple = new Person[] { per, student, schooler, correspStudent };

foreach (Person person in arrSimple)

{

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

person.ShowSimple();

}

Output3();

Person[] arrVirt = new Person[] { per, student, schooler, correspStudent };

foreach (Person person in arrVirt)

{

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

person.ShowVirt();

}

Output4();

Console.WriteLine("\n\n\n ЧАСТЬ №2\n");

//создание объектов для массива

IInit[] arr2 = CreateRandomArray();

Console.WriteLine("Элементы типа IInit в массиве:");

//меню

bool keepPart2 = true;

while (keepPart2)

{

OutputIInitArray(arr2);

Menu();

int i = ParseInt("Ваш выбор: ");

switch(i)

{

case 1:

int count = CountSchoolesIn133School(arr2);

break;

case 2:

int amount = AmountScholarshipInHSE(arr2);

break;

case 3:

int countStud = CorrespStudOf3CourseInPSU(arr2);

break;

case 4:

int middleHeight = AverageHeightIn135School(arr2);

break;

case 5:

int maxPrice = MaxPriceOfEducatInPSU(arr2);

break;

case 6:

int minAge = MinStudAgeInHSE(arr2);

break;

case 7:

int averageWeight = AverageWeightStudInPSU(arr2);

break;

case 8:

keepPart2 = false;

break;

default:

Console.WriteLine("Такого пункта нет в меню!");

break;

}

if (i != 8) Console.ReadKey();

}

Console.WriteLine("\n\n\n ЧАСТЬ №3\n");

IInit[] arr3 = CreateArrPart3();

Console.ReadKey();

bool keepPart3 = true;

while(keepPart3)

{

{

Menu3();

int i = ParseInt("Ваш выбор: ");

switch (i)

{

case 1:

ArrayOfIInit();

break;

case 2:

DemonstrIComparable();

break;

case 3:

DemonstrSortByPrice();

break;

case 4:

DemonstrSortByGrade();

break;

case 5:

BinarySearchDemonstr();

break;

case 6:

DemonstrCloneAndCopy();

break;

case 7:

keepPart3 = false;

break;

default:

Console.WriteLine("Такого пункта нет в меню!");

break;

}

if (i != 7) Console.ReadKey();

}

}

}

}

}

namespace labor10

{

public class Schooler : Person, IInit, IComparable

{

private int schoolYear;

public int SchoolYear

{

get { return schoolYear; }

set

{

if (value < 0)

Console.WriteLine("Класс не может быть отрицательным!");

else if (value > 11)

Console.WriteLine("класс не бывает больше 11!");

else

schoolYear = value;

}

}

private int schoolNumber;

public int SchoolNumber

{

get { return schoolNumber; }

set

{

if (value < 0)

Console.WriteLine("Номер школы не может быть отрицательным!");

else

schoolNumber = value;

}

}

public static Random rand = new Random();

public Schooler() : base()

{

this.schoolYear = 0;

this.name = "NoName";

this.schoolNumber = 0;

}

public Schooler(int age, int height, int weight, int year, string name, int schoolNumber)

: base(name, age, height, weight)

{

this.schoolYear = year;

this.schoolNumber = schoolNumber;

}

//методы

public new void ShowSimple()

{

Console.WriteLine("Информация об объекте класса Schooler:");

Console.WriteLine($"Имя: {this.name}; Класс: {this.schoolYear}; Школа: №{this.schoolNumber}; " +

$"Возраст: {this.age}; Рост: {this.height}; Вес: {this.weight};");

}

public override void ShowVirt()

{

Console.WriteLine("Информация об объекте класса Schooler:");

Console.WriteLine($"Имя: {this.name}; Класс: {this.schoolYear}; Школа: №{this.schoolNumber}; " +

$"Возраст: {this.age}; Рост: {this.height}; Вес: {this.weight};");

}

//от интерфейса

public override object Init()

{

Person p = (Person)base.Init();

string[] names = new string[] { "Ivanov", "Smirnov", "Petrov", "Vasyechkin", "Fhilimonov", "Belov", "Anyechkin", "Kornev" };

string[] lastNames = new string[] { "A.", "J.", "P.", "M.", "B.", "R.", "G.", "E.", "N.", "O." };

int[] schools = new int[] { 133, 135 };

Schooler sch = new Schooler(p.Age, p.Height, p.Weight, rand.Next(1, 12),

names[rand.Next(0, names.Length)] + " " + lastNames[rand.Next(0, names.Length)],

schools[rand.Next(0, 2)]);

return sch;

}

//перегрузки

public override string ToString()

{

return name + " из школы № " + schoolNumber + " класса " + schoolYear + " ростом " + height;

}

public override bool Equals(object obj)

{

if (obj is Schooler)

{

Schooler s = (Schooler)obj;

return (this.age == s.Age) && (this.height == s.Height) && (this.weight == s.Weight) && (this.schoolYear == s.SchoolYear)

&& (this.name == s.name) && (this.schoolNumber == s.SchoolNumber);

}

return false;

}

public override int CompareTo(object obj)

{

if (obj is Schooler)

{

Schooler s = (Schooler)obj;

return this.name.CompareTo(s.name);

}

else

throw new Exception("Невозможно сравнить два объекта");

}

}

}

namespace labor10

{

class SortByGrade : IComparer

{

public int Compare(object x, object y)

{

Schooler sch1 = (Schooler)x;

Schooler sch2 = (Schooler)y;

if (sch1.SchoolYear < sch2.SchoolYear) return -1;

else if (sch1.SchoolYear > sch2.SchoolYear) return 1;

else return 0;

}

}

}

namespace labor10

{

class SortByHeight : IComparer

{

public int Compare(object x, object y)

{

Person p1 = (Person)x;

Person p2 = (Person)y;

if (p1.Height < p2.Height) return -1;

else if (p1.Height > p2.Height) return 1;

else return 0;

}

}

}

namespace labor10

{

class SortByPrice : IComparer

{

public int Compare(object x, object y)

{

CorrespondenceStudent p1 = (CorrespondenceStudent)x;

CorrespondenceStudent p2 = (CorrespondenceStudent)y;

if (p1.PriceOfStudying < p2.PriceOfStudying) return -1;

else if (p1.PriceOfStudying > p2.PriceOfStudying) return 1;

else return 0;

}

}

}

namespace labor10

{

public class Student : Person, IInit, IComparable

{

private int course;

public int Course

{

get { return course; }

set

{

if (value < 0)

Console.WriteLine("Номер курса не может быть отрицательным!");

else if (value > 5)

Console.WriteLine("Номер курса не бывает больше 5!");

else

course = value;

}

}

private int scholarship;

public int Scholarship

{

get { return scholarship; }

set

{

if (value < 0)

Console.WriteLine("Размер стипендии не может быть отрицательным!");

else

scholarship = value;

}

}

public string universityName;

public static Random rand = new Random();

public Student() : base()

{

this.course = 0;

this.name = "NoName";

this.universityName = "nameless University";

this.scholarship = 0;

}

public Student(int age, int height, int weight, int course, string name, string universityName, int scholarship)

: base(name, age, height, weight)

{

this.course = course;

this.name = name;

this.universityName = universityName;

this.scholarship = scholarship;

}

//методы

public new void ShowSimple()

{

Console.WriteLine("Информация об объекте класса Student:");

Console.WriteLine($"Имя: {this.name}; Курс: {this.course}; Университет: {this.universityName}; " +

$"Стипендия: {scholarship}; Возраст: {this.age}; Рост: {this.height}; Вес: {this.weight};");

}

override public void ShowVirt()

{

Console.WriteLine("Информация об объекте класса Student:");

Console.WriteLine($"Имя: {this.name}; Курс: {this.course}; Университет: {this.universityName}; " +

$"Стипендия: {scholarship}; Возраст: {this.age}; Рост: {this.height}; Вес: {this.weight};");

}

//от интерфейса

public override object Init()

{

Person p = (Person)base.Init();

string[] names = new string[] { "Ivanov", "Smirnov", "Petrov", "Vasyechkin", "Fhilimonov", "Belov", "Anyechkin", "Kornev" };

string[] universities = new string[] { "HSE", "PSU" };

string[] lastNames = new string[] { "A.", "J.", "P.", "M.", "B.", "R.", "G.", "E.", "N.", "O." };

Student stud = new Student(p.Age, p.Height, p.Weight, rand.Next(1, 5),

names[rand.Next(0, names.Length)] + " " + lastNames[rand.Next(0, names.Length)],

universities[rand.Next(0, universities.Length)], rand.Next(1700, 45000));

return stud;

}

//перегрузки

public override string ToString()

{

return name + " " + age + " лет весом " + weight + "кг из " + universityName + " с " + course +" курса имеет стипендию равную " + scholarship;

}

public override bool Equals(object obj)

{

if (obj is Student)

{

Student s = (Student)obj;

return (this.age == s.Age) && (this.height == s.Height) && (this.weight == s.Weight) && (this.course == s.Course)

&& (this.name == s.name) && (this.universityName == s.universityName) && (this.scholarship == s.Scholarship);

}

return false;

}

public override int CompareTo(object obj)

{

if (obj is Student)

{

Student s = (Student)obj;

return this.name.CompareTo(s.name);

}

else

throw new Exception("Невозможно сравнить два объекта");

}

}

}

# **Тестирование**

