|  |
| --- |
| Пермский филиал федерального государственного автономного образовательного учреждения высшего образования  «Национальный исследовательский университет  «Высшая школа экономики»  *Факультет экономики, менеджмента и бизнес-информатики* |
|  |
| \*ИМЯ\*  **Отчет по практической работе №10**  **Вариант 6**  студента образовательной программы «Программная инженерия»  по направлению подготовки *09.03.04 Программная инженерия*   |  |  | | --- | --- | |  | Руководитель  О. Л. Викентьева | |

Пермь, 2020 год

**Задание**

**Часть 1. Создание иерархии классов.**

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.  Без виртуальных функций программа будет работать не правильно! Объяснить почему. Объяснить **необходимость виртуальных функций**

**Классы: место, область, город, мкгаполис;**

**Часть 2. Динамическая идентификация типов.**  
  
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).

**Запросы: имена рабочих заданного цеха;** **количество рабочих в заданном цехе; наименование всех цехов на данном заводе.**

**Часть 3. Абстрактные классы и интерфейсы**

1. Составить иерархию классов в соответствии с вариантом. Во главе иерархии классов одолжен находиться интерфейс, которые определяет поведение объектов из иерархии классов (IExecutable).
2. Создать массив элементов  типа IExecutable и поместить в него экземпляры различных классов иерархии. Выполнить просмотр массива.
3. Реализовать сортировку элементов массива, используя стандартный интерфейс IComparable  и метод Sort класса Array.
4. Реализовать поиск элемента в массиве, используя стандартный интерфейс ICompare  и метод Sort класса Array.
5. Реализовать метод клонирования объектов из интерфейса IClonable. Показать разницу между клонированием и поверхностным копированием объектов

**Листинг программы**

#### **Класс Place**

public class Place : IComparer, IComparable, IPlace

{

// Список существующих континентов

private protected List<string> ContinentName = new List<string>()

{

"Евразия", "Северная Америка",

"Южная Америка", "Африка", "Австралия", "Антарктида"

};

// Список стран для рандомной генерации

private protected List<string> CountriesList = new List<string>()

{

"Россия", "Китай",

"США", "Канада",

"Бразилия", "Аргентина",

"Египет", "ЮАР",

"Австралия", "Новая Зеландия"

};

// Харкатеристики Place

private protected double square;

private protected string continent;

private protected string country;

private protected string placeName;

// Свойство: Занимаемая площадь Place

public double Square

{

get

{

return square;

}

set

{

if(value >= 0)

{

square = value;

}

else

{

Console.WriteLine("Error. The sqaure can't take negative values.");

}

}

}

// Свойство: На каком континенте распложен Place

public string Continent

{

get

{

return continent;

}

set

{

if(ContinentName.IndexOf(value) != -1)

{

continent = value;

}

else

{

Console.WriteLine("Error. There's no such continent.");

}

}

}

// Свойство: В какой стране распложен Place

public string Country

{

get

{

return country;

}

set

{

country = value;

}

}

// Свойство: Название Place

public string PlaceName

{

get

{

return placeName;

}

set

{

placeName = value;

}

}

// Конструктор класса без параметров

public Place()

{

square = 0;

continent = "";

country = "";

placeName = "";

}

// Конструктор класса с параметрами

public Place(string cont, string coun, string name, double sq)

{

continent = cont;

country = coun;

placeName = name;

square = sq;

}

// Вирутальный метод класса, показывающий информацию о Place

public virtual void Show()

{

Console.WriteLine(placeName);

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

Console.WriteLine($"Континент: {continent}");

Console.WriteLine($"Страна: {country}");

Console.WriteLine($"Площадь: {square}\n");

}

// Виртуальный метод класса, создающую информацию о Place с помощью вводимой информации

public virtual void InputCreate()

{

Console.WriteLine("Введите континент: ");

string cont = Console.ReadLine();

while(ContinentName.IndexOf(cont) == -1)

{

Console.WriteLine("Error. There's no such continent. Please repeat the input.");

}

continent = cont;

Console.WriteLine("Введите страну: ");

country = Console.ReadLine();

Console.WriteLine("Введите наименование места: ");

placeName = Console.ReadLine();

Console.WriteLine("Введите площадь: ");

double number;

while(!(double.TryParse(Console.ReadLine(), out number)) || number <= 0)

{

Console.WriteLine("Error. The sqaure can't take negative values. Please repeat the input.");

}

square = number;

}

// Виртуальный метод класса, создающую информацию о Place с помощью рандомайзера

public virtual void RandomCreate()

{

Random random = new Random();

int contPos = random.Next(0, ContinentName.Count);

switch (contPos)

{

case 0:

country = CountriesList[random.Next(0, 1)];

if (country == "Россия")

{

placeName = "Байкал";

square = 560000;

}

else

{

PlaceName = "Желтое море";

square = 380000;

}

break;

case 1:

country = CountriesList[random.Next(2, 3)];

if (country == "США")

{

placeName = "Мичиган";

square = 58016;

}

else

{

placeName = "Морейн";

square = 0.5;

}

break;

case 2:

country = CountriesList[random.Next(4, 5)];

if (country == "Бразилия")

{

placeName = "Пантанал";

square = 195000;

}

else

{

placeName = "Анды";

square = 3371000;

}

break;

case 3:

country = CountriesList[random.Next(6, 7)];

if (country == "Египет")

{

placeName = "Аравийская пустыня";

square = 2330000;

}

else

{

placeName = "Национальный парк Крюгер";

square = 19485;

}

break;

case 4:

country = CountriesList[random.Next(8, 9)];

if (country == "Австралия")

{

placeName = "АБольшой Барьерный риф";

square = 3487000;

}

else

{

placeName = "Хоббитон";

square = 5;

}

break;

case 5:

country = "";

PlaceName = "земля королевы мод";

Square = 27000000;

break;

}

continent = ContinentName[contPos];

}

// Виртуальный метод класса, который проверяет, принадлежит ли Place заданному запросу

public virtual bool Search(object search)

{

return search is string contry ? country == search : false;

}

// Реализация интрефейса IComparable для возможности сравнения двух Place по площади

public int CompareTo(object o)

{

if (o is Place place)

return Square.CompareTo(place.Square);

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

}

// Реализация интрефейса IComparable для возможности сравнения двух Place по всем параметрам

public int CompareTo(Place other)

{

if (ReferenceEquals(this, other)) return 0;

if (ReferenceEquals(null, other)) return 1;

var continentComparison = string.Compare(Continent, other.Continent, StringComparison.Ordinal);

if (continentComparison != 0) return continentComparison;

var countryComparison = string.Compare(Country, other.Country, StringComparison.Ordinal);

if (countryComparison != 0) return countryComparison;

var placeNameComparison = string.Compare(PlaceName, other.PlaceName, StringComparison.Ordinal);

if (placeNameComparison != 0) return placeNameComparison;

return Square.CompareTo(other.Square);

}

// Реализация интрефейса IComparer для возможности сравнения двух Place по площади с возвратом значения, указывающего, является ли один объект меньшим, равным или большим другого.

public int Compare(object o1, object o2)

{

Place p1 = o1 as Place;

Place p2 = o2 as Place;

if (p1.Square > p2.Square)

return 1;

if (p1.Square < p2.Square)

return -1;

else

return 0;

}

}

#### **Класс Region**

public class Region : Place, IComparer, IComparable, IPlace, ICloneable

{

// Характеристика Region

private protected int populationSize;

// Свйоство: Численность населения

public int PopulationSize

{

get

{

return populationSize;

}

set

{

if (value >= 0)

{

populationSize = value;

}

else

{

Console.WriteLine("Error. The sqaure can't take negative values.");

}

}

}

// Констурктор клааса без параметров

public Region()

: base()

{

populationSize = 0;

}

// Конструктор класса с параметрами

public Region(string cont, string coun, string name, double sq, int pSize)

:base(cont, coun, name, sq)

{

populationSize = pSize;

}

// Мметод класса, показывающий плотность населения в Region

public double PopulationDensity()

{

return populationSize / square;

}

// Вирутальный метод класса, показывающий информацию о Region

public override void Show()

{

Console.WriteLine(placeName);

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

Console.WriteLine("Характеристика");

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

Console.WriteLine($"Тип: область;");

Console.WriteLine($"Континент: {continent};");

Console.WriteLine($"Страна: {country};");

Console.WriteLine($"Площадь: {square};");

Console.WriteLine($"Население: {populationSize}.\n");

}

// Виртуальный метод класса, создающую информацию о Place с помощью вводимой информации

public override void InputCreate()

{

Console.WriteLine("Введите континент: ");

string cont = Console.ReadLine();

while (ContinentName.IndexOf(cont) == -1)

{

Console.WriteLine("Error. There's no such continent. Please repeat the input.");

}

continent = cont;

Console.WriteLine("Введите страну: ");

country = Console.ReadLine();

Console.WriteLine("Введите название области: ");

placeName = Console.ReadLine();

Console.WriteLine("Введите площадь: ");

double number;

while (!(double.TryParse(Console.ReadLine(), out number)) || number <= 0)

{

Console.WriteLine("Error. The sqaure can't take negative values. Please repeat the input.");

}

square = number;

Console.WriteLine("Введите численность населения: ");

int size;

while (!(int.TryParse(Console.ReadLine(), out size)) || number <= 0)

{

Console.WriteLine("Error. The sqaure can't take negative values. Please repeat the input.");

}

populationSize = size;

}

// Виртуальный метод класса, создающую информацию о Place с помощью рандомайзера

public override void RandomCreate()

{

Random random = new Random();

int contPos = random.Next(0, ContinentName.Count);

switch (contPos)

{

case 0:

country = CountriesList[random.Next(0, 1)];

if (country == "Россия")

{

placeName = "Пермский край";

square = 160600;

populationSize = 2637032;

}

else

{

placeName = "Провинция Аньхой";

square = 140200;

populationSize = 59500000;

}

break;

case 1:

country = CountriesList[random.Next(2, 3)];

if (country == "США")

{

placeName = "Штат Аляска";

square = 1718000;

populationSize = 731545;

}

else

{

placeName = "Провинция Квебек";

square = 1667441;

populationSize = 8294656;

}

break;

case 2:

country = CountriesList[random.Next(4, 5)];

if (country == "Бразилия")

{

placeName = "Штат Сан-Паулу";

square = 248197;

populationSize = 41262000;

}

else

{

placeName = "Провинция Буэнос-Айрес";

square = 307571;

populationSize = 16660000;

}

break;

case 3:

country = CountriesList[random.Next(6, 7)];

if (country == "Египет")

{

placeName = "Провинция Александрия";

square = 2300;

populationSize = 4110015;

}

else

{

placeName = "Восточно-Капская провинция";

square = 168966;

populationSize = 6562053;

}

break;

case 4:

country = CountriesList[random.Next(8, 9)];

if (country == "Австралия")

{

placeName = "Штат Виктория";

square = 277416;

populationSize = 5354042;

}

else

{

placeName = "Район Нортленд";

square = 13941;

populationSize = 1551692;

}

break;

case 5:

country = "";

placeName = "земля королевы мод";

square = 27000000;

populationSize = 0;

break;

}

continent = ContinentName[contPos];

}

// Виртуальный метод класса, который проверяет, принадлежит ли Region заданному запросу (прворяет название области по запросу)

public override bool Search(object search)

{

return search is string contry ? placeName == search : false;

}

// Реализация интрефейса IComparable для возможности сравнения двух Region по численности населения

public new int CompareTo(object o)

{

if (o is Region place)

return PopulationSize.CompareTo(place.PopulationSize);

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

}

// Реализация интрефейса IComparable для возможности сравнения двух Region по всем параметрам

public int CompareTo(Region other)

{

if (ReferenceEquals(this, other)) return 0;

if (ReferenceEquals(null, other)) return 1;

var continentComparison = string.Compare(Continent, other.Continent, StringComparison.Ordinal);

if (continentComparison != 0) return continentComparison;

var countryComparison = string.Compare(Country, other.Country, StringComparison.Ordinal);

if (countryComparison != 0) return countryComparison;

var placeNameComparison = string.Compare(PlaceName, other.PlaceName, StringComparison.Ordinal);

if (placeNameComparison != 0) return placeNameComparison;

var PopulationSizeComparison = this.Compare(PopulationSize, other.PopulationSize);

if (PopulationSizeComparison != 0) return PopulationSizeComparison;

return Square.CompareTo(other.Square);

}

// Реализация интрефейса IComparer для возможности сравнения двух Region по численности населения с возвратом значения, указывающего, является ли один объект меньшим, равным или большим другого.

public new int Compare(object o1, object o2)

{

Region p1 = o1 as Region;

Region p2 = o2 as Region;

if (p1.PopulationSize > p2.PopulationSize)

return 1;

if (p1.PopulationSize < p2.PopulationSize)

return -1;

else

return 0;

}

// Метод класса, позволяющий поверхостное копирование

public Region ShallowCopy()

{

return (Region)MemberwiseClone();

}

public object Clone()

{

throw new NotImplementedException();

}

}

#### **Класс City**

class City : Region, IComparer, IComparable, IPlace, ICloneable

{

// Характеристика City

private protected string regionName;

// Переопределение Свойства: Численность населения

public new int PopulationSize

{

get

{

return populationSize;

}

set

{

if (value >= 0 && value < 1000000)

{

populationSize = value;

}

else if (value >= 0 && value >= 1000000)

{

Console.WriteLine("Error. The population does not correspond to a City.");

}

else

{

Console.WriteLine("Error. The sqaure can't take negative values.");

}

}

}

// Название области

public string RegionName

{

get

{

return regionName;

}

set

{

regionName = value;

}

}

// Конструктор класса без параметров

public City()

: base()

{

populationSize = 0;

regionName = "";

}

// Конструктор класса с параметрами

public City(string cont, string coun, string name, string Rname, double sq, int pSize)

: base(cont, coun, name, sq, pSize)

{

regionName = Rname;

}

// Вирутальный метод класса, показывающий информацию о Region

public override void Show()

{

Console.WriteLine(placeName);

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

Console.WriteLine("Характеристика");

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

Console.WriteLine($"Тип: город;");

Console.WriteLine($"Континент: {continent};");

Console.WriteLine($"Страна: {country};");

Console.WriteLine($"Область: {regionName};");

Console.WriteLine($"Площадь: {square};");

Console.WriteLine($"Население: {populationSize}.\n");

}

// Виртуальный метод класса, создающую информацию о City с помощью вводимой информации

public override void InputCreate()

{

Console.WriteLine("Введите континент: ");

string cont = Console.ReadLine();

while (ContinentName.IndexOf(cont) == -1)

{

Console.WriteLine("Error. There's no such continent. Please repeat the input.");

}

continent = cont;

Console.WriteLine("Введите страну: ");

country = Console.ReadLine();

Console.WriteLine("Введите название области: ");

regionName = Console.ReadLine();

Console.WriteLine("Введите название города: ");

placeName = Console.ReadLine();

Console.WriteLine("Введите площадь: ");

double number;

while (!(double.TryParse(Console.ReadLine(), out number)) || number <= 0)

{

Console.WriteLine("Error. The sqaure can't take negative values. Please repeat the input.");

}

square = number;

Console.WriteLine("Введите численность населения: ");

int size;

while (!(int.TryParse(Console.ReadLine(), out size)) || number <= 0)

{

Console.WriteLine("Error. The sqaure can't take negative values. Please repeat the input.");

}

populationSize = size;

}

// Виртуальный метод класса, создающую информацию о Place с помощью рандомайзера

public override void RandomCreate()

{

Random random = new Random();

int contPos = random.Next(0, ContinentName.Count);

switch (contPos)

{

case 0:

country = CountriesList[random.Next(0, 1)];

if (country == "Россия")

{

placeName = "Пермь";

regionName = "Пермский край";

square = 801.44;

populationSize = 1055397;

}

else

{

placeName = "Пекин";

regionName = "Города центрального подчинения";

square = 16410.54;

populationSize = 21710000;

}

break;

case 1:

country = CountriesList[random.Next(2, 3)];

if (country == "США")

{

placeName = "Нью-Йорк";

regionName = "Штат Нью-Йорк";

square = 1214.9;

populationSize = 8405837;

}

else

{

placeName = "Монреаль";

regionName = "Провинция Квебек";

square = 363.13;

populationSize = 1942694;

}

break;

case 2:

country = CountriesList[random.Next(4, 5)];

if (country == "Бразилия")

{

placeName = "Бразилиа";

regionName = "Федеральный округ";

square = 5801.937;

populationSize = 2609997;

}

else

{

placeName = "Буэнос-Айрес";

regionName = "Провинция Буэнос-Айрес";

square = 202;

populationSize = 3063728;

}

break;

case 3:

country = CountriesList[random.Next(6, 7)];

if (country == "Египет")

{

placeName = "Каир";

regionName = "Провинция Каир";

square = 606;

populationSize = 9840591;

}

else

{

placeName = "Кейптаун";

regionName = "Западно-Капская провинция";

square = 2444.97;

populationSize = 3740026;

}

break;

case 4:

country = CountriesList[random.Next(8, 9)];

if (country == "Австралия")

{

placeName = "Штат Виктория";

regionName = "Штат Новый южный Уэльс";

square = 12144;

populationSize = 5131326;

}

else

{

placeName = "";

regionName = "";

square = 0;

populationSize = 0;

}

break;

case 5:

country = "";

placeName = "";

regionName = "земля королевы мод";

square = 27000000;

populationSize = 0;

break;

}

continent = ContinentName[contPos];

}

// Виртуальный метод класса, который проверяет, принадлежит ли City заданному запросу (проверяет: принадлежит ли город заданной области)

public override bool Search(object search)

{

return search is string contry ? regionName == search : false;

}

// Реализация интрефейса IComparable для возможности сравнения двух City по численности населения

public new int CompareTo(object o)

{

if (o is City place)

return PopulationSize.CompareTo(place.PopulationSize);

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

}

// Реализация интрефейса IComparable для возможности сравнения двух City по всем параметрам

public int CompareTo(City other)

{

if (ReferenceEquals(this, other)) return 0;

if (ReferenceEquals(null, other)) return 1;

var continentComparison = string.Compare(Continent, other.Continent, StringComparison.Ordinal);

if (continentComparison != 0) return continentComparison;

var countryComparison = string.Compare(Country, other.Country, StringComparison.Ordinal);

if (countryComparison != 0) return countryComparison;

var placeNameComparison = string.Compare(PlaceName, other.PlaceName, StringComparison.Ordinal);

if (placeNameComparison != 0) return placeNameComparison;

var RegionNameComparison = string.Compare(RegionName, other.RegionName, StringComparison.Ordinal);

if (RegionNameComparison != 0) return RegionNameComparison;

var PopulationSizeComparison = this.Compare(PopulationSize, other.PopulationSize);

if (PopulationSizeComparison != 0) return PopulationSizeComparison;

return Square.CompareTo(other.Square);

}

// Реализация интрефейса IComparer для возможности сравнения двух City по численности населения с возвратом значения, указывающего, является ли один объект меньшим, равным или большим другого.

public new int Compare(object o1, object o2)

{

City p1 = o1 as City;

City p2 = o2 as City;

if (p1.PopulationSize > p2.PopulationSize)

return 1;

if (p1.PopulationSize < p2.PopulationSize)

return -1;

else

return 0;

}

// Метод класса, позволяющий поверхостное копирование

public new City ShallowCopy()

{

return (City)MemberwiseClone();

}

}

#### **Класс Megapolis**

class Megapolis : Region, IComparer, IComparable, IPlace, ICloneable

{

// Характеристика Megapolis

private protected string regionName;

// Переопределение численности населения

public new int PopulationSize

{

get

{

return populationSize;

}

set

{

if (value >= 0 && value >= 1000000)

{

populationSize = value;

}

else if (value >= 0 && value < 1000000)

{

Console.WriteLine("Error. The population does not correspond to a megapolis.");

}

else

{

Console.WriteLine("Error. The sqaure can't take negative values.");

}

}

}

// Название области

public string RegionName

{

get

{

return regionName;

}

set

{

regionName = value;

}

}

// Конструктор класса без параметров

public Megapolis()

: base()

{

populationSize = 0;

regionName = "";

}

// Конструктор класса с параметрами

public Megapolis(string cont, string coun, string name, string Rname, double sq, int pSize)

: base(cont, coun, name, sq, pSize)

{

regionName = Rname;

}

// Вирутальный метод класса, показывающий информацию о Region

public override void Show()

{

Console.WriteLine(placeName);

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

Console.WriteLine("Характеристика");

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

Console.WriteLine($"Тип: мегаполис;");

Console.WriteLine($"Континент: {continent};");

Console.WriteLine($"Страна: {country};");

Console.WriteLine($"Область: {regionName};");

Console.WriteLine($"Площадь: {square};");

Console.WriteLine($"Население: {populationSize}.\n");

}

// Виртуальный метод класса, создающую информацию о Megapolis с помощью вводимой информации

public override void InputCreate()

{

Console.WriteLine("Введите континент: ");

string cont = Console.ReadLine();

while (ContinentName.IndexOf(cont) == -1)

{

Console.WriteLine("Error. There's no such continent. Please repeat the input.");

}

continent = cont;

Console.WriteLine("Введите страну: ");

country = Console.ReadLine();

Console.WriteLine("Введите название области: ");

regionName = Console.ReadLine();

Console.WriteLine("Введите название мегаполиса: ");

placeName = Console.ReadLine();

Console.WriteLine("Введите площадь: ");

double number;

while (!(double.TryParse(Console.ReadLine(), out number)) || number <= 0)

{

Console.WriteLine("Error. The sqaure can't take negative values. Please repeat the input.");

}

square = number;

Console.WriteLine("Введите численность населения: ");

int size;

while (!(int.TryParse(Console.ReadLine(), out size)) || number <= 0)

{

Console.WriteLine("Error. The sqaure can't take negative values. Please repeat the input.");

}

populationSize = size;

}

// Виртуальный метод класса, создающую информацию о Place с помощью рандомайзера

public override void RandomCreate()

{

Random random = new Random();

int contPos = random.Next(0, ContinentName.Count);

switch (contPos)

{

case 0:

country = CountriesList[random.Next(0, 1)];

if (country == "Россия")

{

PlaceName = "Пермь";

regionName = "Пермский край";

square = 801.44;

populationSize = 900000;

}

else

{

placeName = "Пекин";

regionName = "Города центрального подчинения";

square = 16410.54;

populationSize = 217100;

}

break;

case 1:

country = CountriesList[random.Next(2, 3)];

if (country == "США")

{

placeName = "Нью-Йорк";

regionName = "Штат Нью-Йорк";

square = 1214.9;

populationSize = 405837;

}

else

{

placeName = "Монреаль";

regionName = "Провинция Квебек";

square = 363.13;

populationSize = 942694;

}

break;

case 2:

country = CountriesList[random.Next(4, 5)];

if (country == "Бразилия")

{

placeName = "Бразилиа";

regionName = "Федеральный округ";

square = 5801.937;

populationSize = 609997;

}

else

{

placeName = "Буэнос-Айрес";

regionName = "Провинция Буэнос-Айрес";

square = 202;

populationSize = 363728;

}

break;

case 3:

country = CountriesList[random.Next(6, 7)];

if (country == "Египет")

{

placeName = "Каир";

regionName = "Провинция Каир";

square = 606;

populationSize = 840591;

}

else

{

placeName = "Кейптаун";

regionName = "Западно-Капская провинция";

square = 2444.97;

populationSize = 740026;

}

break;

case 4:

country = CountriesList[random.Next(8, 9)];

if (country == "Австралия")

{

placeName = "Штат Виктория";

regionName = "Штат Новый южный Уэльс";

square = 12144;

populationSize = 131326;

}

else

{

placeName = "Велингтон";

regionName = "Регион Велингтон";

square = 119.78;

populationSize = 412500;

}

break;

case 5:

country = "";

placeName = "";

regionName = "земля королевы мод";

square = 27000000;

populationSize = 0;

break;

}

continent = ContinentName[contPos];

}

// Виртуальный метод класса, который проверяет, принадлежит ли Megapolis заданному запросу (проверяет: принадлежит ли мегаполис заданной области)

public override bool Search(object search)

{

return search is string contry ? regionName == search : false;

}

// Реализация интрефейса IComparable для возможности сравнения двух Megapolis по численности населения

public new int CompareTo(object o)

{

if (o is Megapolis place)

return PopulationSize.CompareTo(place.PopulationSize);

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

}

// Реализация интрефейса IComparable для возможности сравнения двух Megapolis по всем параметрам

public int CompareTo(Megapolis other)

{

if (ReferenceEquals(this, other)) return 0;

if (ReferenceEquals(null, other)) return 1;

var continentComparison = string.Compare(Continent, other.Continent, StringComparison.Ordinal);

if (continentComparison != 0) return continentComparison;

var countryComparison = string.Compare(Country, other.Country, StringComparison.Ordinal);

if (countryComparison != 0) return countryComparison;

var placeNameComparison = string.Compare(PlaceName, other.PlaceName, StringComparison.Ordinal);

if (placeNameComparison != 0) return placeNameComparison;

var RegionNameComparison = string.Compare(RegionName, other.RegionName, StringComparison.Ordinal);

if (RegionNameComparison != 0) return RegionNameComparison;

var PopulationSizeComparison = this.Compare(PopulationSize, other.PopulationSize);

if (PopulationSizeComparison != 0) return PopulationSizeComparison;

return Square.CompareTo(other.Square);

}

// Реализация интрефейса IComparer для возможности сравнения двух Megapolis по численности населения с возвратом значения, указывающего, является ли один объект меньшим, равным или большим другого.

public new int Compare(object o1, object o2)

{

Megapolis p1 = o1 as Megapolis;

Megapolis p2 = o2 as Megapolis;

if (p1.PopulationSize > p2.PopulationSize)

return 1;

if (p1.PopulationSize < p2.PopulationSize)

return -1;

else

return 0;

}

// Метод класса, позволяющий поверхостное копирование

public new Megapolis ShallowCopy()

{

return (Megapolis)MemberwiseClone();

}

}

#### **Классы сортировки**

// Сортировка массива PlaceArray по названию объекта

public class SortByName : IComparer

{

int IComparer.Compare(object obj1, object obj2)

{

Place p1 = (Place)obj1;

Place p2 = (Place)obj2;

return String.Compare(p1.PlaceName, p2.PlaceName);

}

}

// Сортировка массива PlaceArray по континенту

public class SortByContinent : IComparer

{

int IComparer.Compare(object obj1, object obj2)

{

Place p1 = (Place)obj1;

Place p2 = (Place)obj2;

return String.Compare(p1.Continent, p2.Continent);

}

}

// Сортировка массива PlaceArray по стране

public class SortByContry : IComparer

{

int IComparer.Compare(object obj1, object obj2)

{

Place p1 = (Place)obj1;

Place p2 = (Place)obj2;

return String.CompareOrdinal(p1.Country, p2.Country);

}

}

// Сортировка массива PlaceArray по площади

public class SortBySquare : IComparer

{

int IComparer.Compare(object obj1, object obj2)

{

Place p1 = (Place)obj1;

Place p2 = (Place)obj2;

if (p1.Square > p2.Square)

return 1;

if (p1.Square > p2.Square)

return -1;

return 0;

}

}

#### **Класс PlaceArray**

class PlaceArray

{

public Place[] Arr;

public int Count;

// Конструктор класса без параметров

public PlaceArray()

{

Arr = null;

Count = 0;

}

// Конструктор класса с параметрами

public PlaceArray(int size)

{

Arr = new Place[size];

Count = size;

}

// Индексация к элементу массива

public Place this[int index]

{

get => Arr[index];

set => Arr[index] = value;

}

// Ввод размера массива

public static int InputSize()

{

int number;

Console.WriteLine("Введите рамер массива:");

while (!int.TryParse(Console.ReadLine(), out number) || number < 0 || number > 100)

{

Console.WriteLine("Ошибка. Вы неверно ввели размер массива. Повторите ввод.");

}

return number;

}

// Вывод всей инфорамции о персонах, информация о которых хранится в PersonArray

public void Show()

{

Console.WriteLine("Места:\n");

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

{

this[i].Show();

Console.WriteLine();

}

}

// Рандомная генерация массива

public void RandomGeneration(int size)

{

Random random = new Random();

Place [] place = new Place[size];

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

{

place[i] = random.Next(1, 3) switch

{

1 => new Region(),

2 => new City(),

3 => new Megapolis(),

\_ => place[i]

};

place[i].RandomCreate();

}

Arr = place;

Count = size;

}

// Сортировка массива по площади

public void Sort()

{

Array.Sort(Arr, new SortBySquare());

}

// Поиск элемента массива по заданной площади

public int Search(double square)

{

try

{

return Array.BinarySearch(Arr, square, new SortBySquare());

}

catch (Exception)

{

return -1;

}

}

// Поиск элемента массива по заданному названию оюъекта

public int Search(string name)

{

try

{

return Array.BinarySearch(Arr, name, new SortByName());

}

catch (Exception)

{

return -1;

}

}

}

#### **Интрефейс IPlace**

public interface IPlace

{

public void InputCreate();

public void RandomCreate();

public void Show();

public bool Search(object search);

}

#### **Класс Program**

public class Program

{

// Функция ввода номера части лабораторной работы

private static int InputMode()

{

int number;

Console.WriteLine("Введите часть (1-3) лабораторной работы (0 - выход из программы):");

while (!int.TryParse(Console.ReadLine(), out number) || number < 0 || number > 3)

{

Console.WriteLine("Ошибка. Вы неверно ввели часть лабораторной работы");

}

return number;

}

// Функция ввода номера типа Place

private static int InputClass()

{

int number;

Console.WriteLine("Выберите тип места, который хотите добавить:");

Console.WriteLine("1) Регион\n2) Город\n3) Мегаполис");

while (!int.TryParse(Console.ReadLine(), out number) || number < 1 || number > 3)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Ошибка. Вы неверно выбрали тип места. Повторите ввод");

Console.ForegroundColor = ConsoleColor.White;

}

return number;

}

static void Main()

{

PlaceArray placeArray;

int mode = InputMode();

while (mode != 0)

{

switch (mode)

{

case 1:

{

placeArray = new PlaceArray(PlaceArray.InputSize());

for (int i = 0; i < placeArray.Count; i++)

{

int type = InputClass();

switch (type)

{

case 1:

{

Region region = new Region();

placeArray[i] = region;

placeArray[i].InputCreate();

}

break;

case 2:

{

City city = new City();

placeArray[i] = city;

placeArray[i].RandomCreate();

}

break;

case 3:

{

Megapolis megapolis = new Megapolis();

placeArray[i] = megapolis;

placeArray[i].RandomCreate();

}

break;

}

}

placeArray.Show();

break;

}

case 2:

{

placeArray = new PlaceArray(6);

placeArray.RandomGeneration(placeArray.Count);

placeArray.Show();

break;

}

case 3:

{

Random random = new Random();

int size = random.Next(3, 9);

// 1.Составить иерархию классов в соответствии с вариантом. Иерархия должна содержать хотя бы один интерфейс и хотя бы один абстрактный класс.

Console.WriteLine("Иерархия классов:");

Console.WriteLine("Place - абстрактный класс, является 'родителем' остальных классов.");

Console.WriteLine("City, Megapolis, Region - классы, наследующие методы от Place.");

//2.Создать массив интерфейсных элементов и поместить в него экземпляры различных классов иерархии.

placeArray = new PlaceArray(size);

placeArray.RandomGeneration(size);

placeArray.Show();

//3.Реализовать сортировку элементов массива, используя стандартные интерфейсы и методы класса Array.

placeArray.Sort();

placeArray.Show();

//4.Реализовать поиск элемента в массиве, используя стандартные интерфейсы и методы класса Array.

int index = placeArray.Search(50);

Console.WriteLine(index > -1

? $"Искомая место находится на позиции {index}"

: "Место с заданными параметрами найдена не была!");

index = placeArray.Search("Андрей");

Console.WriteLine(index > -1

? $"Искомая место находится на позиции {index}"

: "Место с заданными параметрами найдена не была!");

//5.Реализовать в одном из классов метод клонирования объектов.Показать клонирование объектов.

Region Region = new Region();

Region newRegion = Region.ShallowCopy();

Region.Show();

newRegion.Show();

break;

}

}

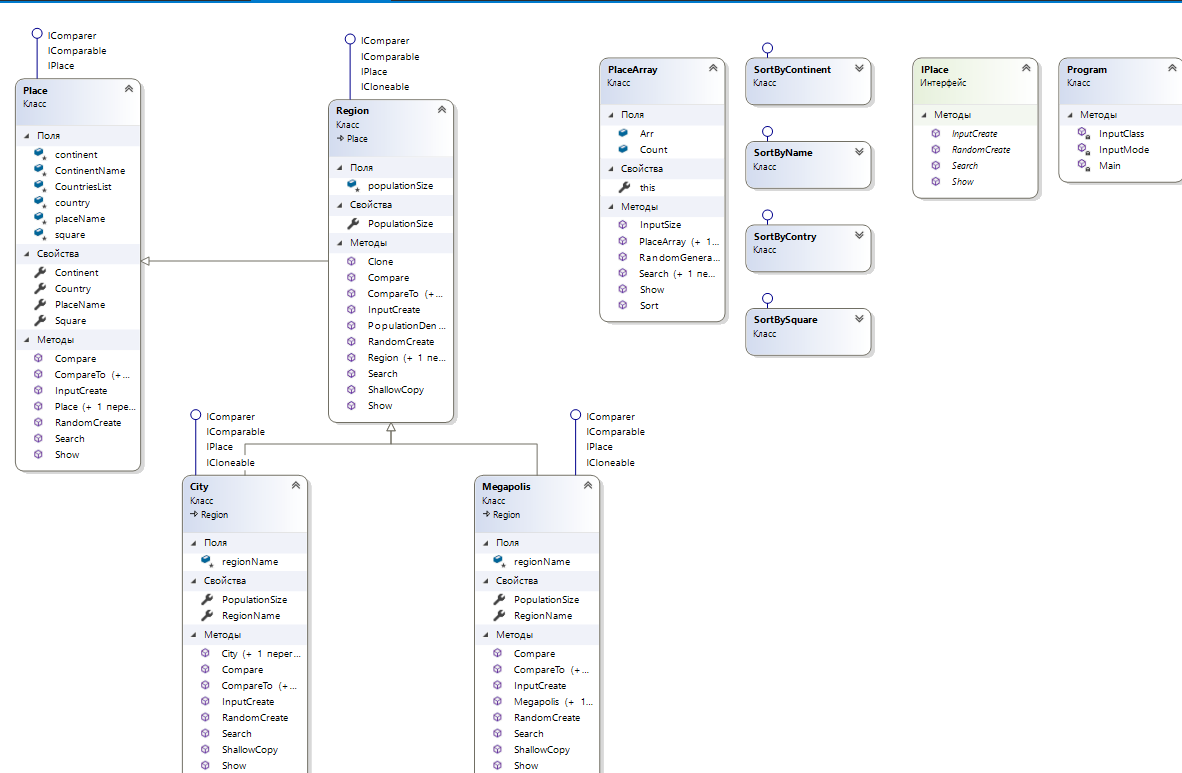
mode = InputMode();

}

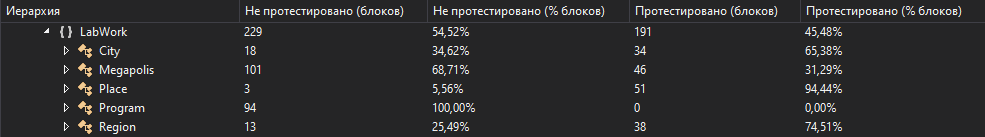
}

}

**Диаграмма классов**



**Тестирование – покрытие тестов**



Ссылка на ГитХаб:

<https://github.com/Anni-HSE/Place-and-Collections>