|  |
| --- |
| Пермский филиал федерального государственного автономного образовательного учреждения высшего образования  «Национальный исследовательский университет  «Высшая школа экономики»  *Факультет экономики, менеджмента и бизнес-информатики* |
|  |
| Никитин Андрей Владимирович  **Отчет по практической работе №11**  **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. Общая коллекция.**

* 1. Создать коллекцию, в которую добавить объекты созданной иерархии классов.
  2. Используя меню, реализовать в программе добавление и удаление объектов коллекции.
  3. Разработать и реализовать три запроса (количество элементов определенного вида, печать элементов определенного вида и т.п.).
  4. Выполнить перебор элементов коллекции с помощью метода foreach.
  5. Выполнить клонирование коллекции.
  6. Выполнить сортировку коллекции (если коллекция не отсортирована) и поиск заданного элемента в коллекции.

При работе с коллекцией использовать объекты из иерархии классов, разработанной в работе №10.

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

**Часть 2. Обобщенная коллекция.**

1. Создать обобщенную коллекцию, в которую добавить объекты созданной иерархии классов.

2. Используя меню, реализовать в программе добавление и удаление объектов коллекции.

3. Разработать и реализовать три запроса (количество элементов определенного вида, печать элементов определенного вида и т.п.).

4. Выполнить перебор элементов коллекции с помощью метода foreach.

5. Выполнить клонирование коллекции.

6. Выполнить сортировку коллекции (если коллекция не отсортирована) и поиск заданного элемента в коллекции.

При работе с коллекцией использовать объекты из иерархии классов, разработанной в работе №10.

**Часть 3. Собственная коллекция**

1. Создать иерархию классов (базовый – производный) в соответствии с вариантом (см. лаб. раб. №10).
2. В производном классе определить свойство, которое возвращает ссылку на объект базового класса (это свойство должно возвращать ссылку на объект базового класса, а не ссылку на вызывающий объект производного класса). Например, для иерархии классов Person-Student в классе производном классе Student можно определить свойство
3. Определить класс TestCollections, который содержит поля следующих типов

Коллекция\_1<TKey> ;

Коллекция\_1<string> ;

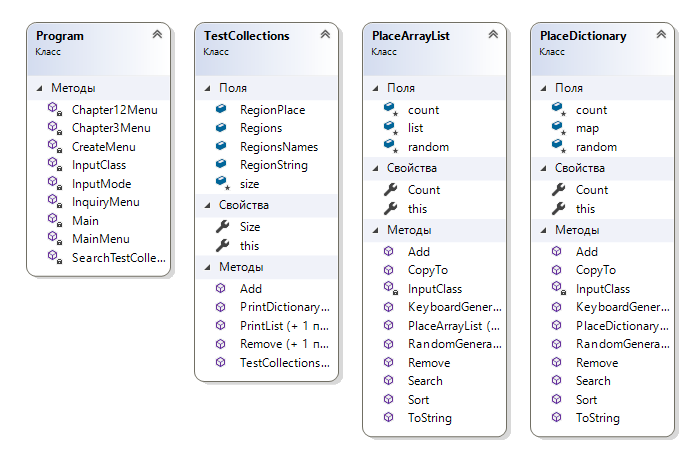
Коллекция\_2<TKey, TValue> ;

Коллекция\_2<string, TValue> .

где тип ключа TKey и тип значения TValue связаны отношением базовый-производный (см. задание 1), Коллекция\_1 и Коллекция\_2 – коллекции из пространства имен System.Collections.Generic.

1. Написать конструктор класса TestCollections, в котором создаются коллекции с заданным числом элементов.
2. Предусмотреть автоматическую генерацию элементов коллекции таким образом, что каждый объект (Student) содержит подобъект базового класса (Person).
3. Все четыре коллекции должны содержать одинаковое число элементов. Каждому элементу из коллекции Коллекция\_1<TKey> должен отвечать элемент в коллекции Коллекция\_2<TKey, TValue> с равным значением ключа. Список Коллекция\_1<string> состоит из строк, которые получены в результате вызова метода ToString() для объектов TKey из списка Коллекция\_1<TKey>. Каждому элементу списка Коллекция\_1<string> отвечает элемент в Коллекция\_2 <string, TValue> с равным значением ключа типа string.
4. Для четырех разных элементов – первого, центрального, последнего и элемента, не входящего в коллекцию – надо измерить время поиска элемента в коллекциях Коллекция\_1<TKey> и Коллекция\_1<string> с помощью метода Contains; элемента по ключу в коллекциях Коллекция\_2< TKey, TValue> и Коллекция\_2 <string, TValue > с помощью метода ContainsKey; значения элемента в коллекции Коллекция\_2< TKey, TValue > с помощью метода ContainsValue. Обратите внимание на то, что искать нужно сами элементы, а не ссылки на них!
5. Предусмотреть методы для работы с коллекциями (добавление и удаление элементов).

**Диаграмма классов:**

****

**Описание методов:**

1. Add(Place place) – добавляет в коллекцию элемент place;
2. Remove() – удалить элемент из коллекции;
3. KeyboardGeneration() – ручное создание коллекции;
4. RandomGeneration() – рандомная генерация коллекции;
5. Search() – поиск элемента коллекции;
6. Sort() – сортировка элементов коллекции;
7. ToString() – печать коллекции;
8. PrintDictionary() – печать словаря;
9. PrintList() – печать листа.

**Пример демонстрационной программы:**

static void Main(string[] args)

{

ArrayList placeList = new ArrayList();

Dictionary<string, Place> placeDictionary = new Dictionary<string, Place>();

TestCollections test = new TestCollections();

Random random = new Random();

int mode = InputMode("часть лабраторной работы", MainMenu(), 3);

while(mode != 0)

{

int type;

switch (mode)

{

case 1:

{

type = InputMode("номер задания", Chapter12Menu(), 7);

while (type != 0)

{

switch (type)

{

case 1:

{

int createType = InputMode("тип ввода", CreateMenu(), 2);

switch (createType)

{

case 1:

{

int placeType = InputClass();

switch (placeType)

{

case 1:

{

Region region = new Region();

region.InputCreate();

placeList.Add(region);

}

break;

case 2:

{

City city = new City();

city.InputCreate();

placeList.Add(city);

}

break;

case 3:

{

Megapolis megapolis = new Megapolis();

megapolis.InputCreate();

placeList.Add(megapolis);

}

break;

}

}

break;

case 2:

{

int placeType = random.Next(1, 3);

switch (placeType)

{

case 1:

{

Region region = new Region();

region.RandomCreate();

placeList.Add(region);

}

break;

case 2:

{

City city = new City();

city.RandomCreate();

placeList.Add(city);

}

break;

case 3:

{

Megapolis megapolis = new Megapolis();

megapolis.RandomCreate();

placeList.Add(megapolis);

}

break;

}

}

break;

}

}

break;

case 2:

{

Console.WriteLine("Введите имя объекта, которого хотите удалить");

string placeName = Console.ReadLine();

bool end = false;

for(int i=0; i < placeList.Count && !end; i++)

{

Place place = placeList[i] as Place;

if (place.PlaceName == placeName)

{

end = true;

placeList.Remove(place);

Console.WriteLine("Объект по вашему запросу был удален");

}

}

if (!end)

{

Console.WriteLine("Удаление не возможно. В коллекции нет объекта, удовлетворяющему вашему запросу");

}

}

break;

case 3:

{

int inquiry = InputMode("номер запроса", InquiryMenu(), 3);

switch (inquiry)

{

case 1:

{

int quantity = 0;

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

{

if (placeList[i] is Region)

{

quantity++;

}

}

Console.WriteLine("Кол-во объектов типа REgion: " + quantity);

}

break;

case 2:

{

List<City> cities = new List<City>();

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

{

if (placeList[i] is City)

{

cities.Add(placeList[i] as City);

}

}

if (cities.Count != 0)

{

foreach (City city in cities)

{

city.Show();

}

}

else

{

Console.WriteLine("В коллекции нет объектов типа City");

}

}

break;

case 3:

{

int quantity = 0;

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

{

if (placeList[i] is Megapolis)

{

quantity++;

}

}

Console.WriteLine("Кол-во объектов типа Megapolis: " + quantity);

}

break;

}

}

break;

case 4:

{

Console.WriteLine("Элементы коллекции");

Place[] placies = new Place[placeList.Count];

placeList.CopyTo(placies);

foreach (Place place in placies)

{

place.Show();

}

}

break;

case 5:

{

ArrayList placeCloneList = (ArrayList) placeList.Clone();

Console.WriteLine("Сколинируемая коллекция");

Place[] placies = null;

placeCloneList.CopyTo(placies);

foreach (Place place in placies)

{

place.Show();

}

}

break;

case 6:

{

placeList.Sort(new SortByName());

Console.WriteLine("Коллекция отсортирована по названию места в алфовитном порядке");

}

break;

case 7:

{

Console.WriteLine("Введите названия места, чтобы найти элемент с соответствующим названием");

string search = Console.ReadLine();

int pos = placeList.BinarySearch(search, new SearchByName());

if (pos != -1)

{

Console.WriteLine($"Объекст найден. Его позиция в коллекции: {pos + 1}");

}

else

{

Console.WriteLine($"Объекст не найден.");

}

}

break;

}

type = InputMode("номер задания", Chapter12Menu(), 7);

}

}

break;

case 2:

{

type = InputMode("номер задания", Chapter12Menu(), 7);

while (type != 0)

{

switch (type)

{

case 1:

{

int createType = InputMode("тип ввода", CreateMenu(), 2);

switch (createType)

{

case 1:

{

int placeType = InputClass();

switch (placeType)

{

case 1:

{

Region region = new Region();

region.InputCreate();

placeDictionary.Add(region.PlaceName, region);

}

break;

case 2:

{

City city = new City();

city.InputCreate();

placeDictionary.Add(city.PlaceName, city);

}

break;

case 3:

{

Megapolis megapolis = new Megapolis();

megapolis.InputCreate();

placeDictionary.Add(megapolis.PlaceName, megapolis);

}

break;

}

}

break;

case 2:

{

int placeType = random.Next(1, 3);

switch (placeType)

{

case 1:

{

Region region = new Region();

region.RandomCreate();

placeDictionary.Add(region.PlaceName, region);

}

break;

case 2:

{

City city = new City();

city.RandomCreate();

placeDictionary.Add(city.PlaceName, city);

}

break;

case 3:

{

Megapolis megapolis = new Megapolis();

megapolis.RandomCreate();

placeDictionary.Add(megapolis.PlaceName, megapolis);

}

break;

}

}

break;

}

Console.WriteLine("Объект добавлен");

}

break;

case 2:

{

Console.WriteLine("Введите имя объекта, которого хотите удалить");

string placeName = Console.ReadLine();

bool end = false;

if (placeDictionary.ContainsKey(placeName))

{

placeDictionary.Remove(placeName);

}

else

{

Console.WriteLine("Удаление не возможно. В коллекции нет объекта, удовлетворяющему вашему запросу");

}

}

break;

case 3:

{

int inquiry = InputMode("номер запроса", InquiryMenu(), 3);

switch (inquiry)

{

case 1:

{

int quantity = 0;

foreach (string key in placeDictionary.Keys)

{

if (placeDictionary[key] is Region)

{

quantity++;

}

}

Console.WriteLine("Кол-во объектов типа Region: " + quantity);

}

break;

case 2:

{

List<City> cities = new List<City>();

foreach (string key in placeDictionary.Keys)

{

if (placeDictionary[key] is City)

{

City city = placeDictionary[key] as City;

cities.Add(city);

}

}

if (cities.Count != 0)

{

foreach (City city in cities)

{

city.Show();

}

}

else

{

Console.WriteLine("В коллекции нет объектов типа City");

}

}

break;

case 3:

{

int quantity = 0;

foreach (string key in placeDictionary.Keys)

{

if (placeDictionary[key] is Megapolis)

{

quantity++;

}

}

Console.WriteLine("Кол-во объектов типа Megapolis: " + quantity);

}

break;

}

}

break;

case 4:

{

Console.WriteLine("Элементы коллекции");

foreach (string key in placeDictionary.Keys)

{

placeDictionary[key].Show();

}

}

break;

case 5:

{

Dictionary<string, Place> copy = new Dictionary<string, Place>(placeDictionary);

Console.WriteLine("Сколинируемая коллекция");

foreach (string key in copy.Keys)

{

copy[key].Show();

}

}

break;

case 6:

{

var list = placeDictionary.Keys.ToList();

list.Sort();

Dictionary<string, Place> tmp = new Dictionary<string, Place>();

foreach (var key in list)

{

tmp.Add(key, placeDictionary[key]);

}

placeDictionary = tmp;

Console.WriteLine("Коллекция отсортирована по названию места в алфовитном порядке");

}

break;

case 7:

{

Console.WriteLine("Введите названия места, чтобы найти элемент с соответствующим названием");

string search = Console.ReadLine();

var list = placeDictionary.Keys.ToList();

list.Sort();

if (list.IndexOf(search) != 0)

{

Console.WriteLine("Объект найден");

placeDictionary[search].Show();

}

else

{

Console.WriteLine("Объект не найден");

}

}

break;

}

type = InputMode("номер задания", Chapter12Menu(), 7);

}

}

break;

case 3:

{

type = InputMode("номер задания", Chapter3Menu(), 4);

while (type != 0)

{

switch (type)

{

case 1:

{

int createType = InputMode("тип ввода", CreateMenu(), 2);

switch (createType)

{

case 1:

{

Region region = new Region();

region.InputCreate();

test.Add(region);

}

break;

case 2:

{

Region region = new Region();

region.RandomCreate();

test.Add(region);

}

break;

}

}

break;

case 2:

{

Console.WriteLine("Введите имя объекта, которого хотите удалить");

string placeName = Console.ReadLine();

if (test.RegionString.ContainsKey(placeName))

{

test.Remove(test.RegionString[placeName]);

}

else

{

Console.WriteLine("Удаление не возможно. В коллекции нет объекта, удовлетворяющему вашему запросу");

}

}

break;

case 3:

{

if (test.Size >= 4)

{

SearchTestCollections(test);

}

else

{

test = new TestCollections();

for(int i = 0; i < random.Next(1,10); i++)

{

Region region = new Region();

region.RandomCreate();

test.Add(region);

}

SearchTestCollections(test);

}

}

break;

case 4:

{

Console.WriteLine("Выберете какую именно коллекцию хотите распечатать из объекта типа TestCollections");

Console.WriteLine("1) Regions\n2) Regions names\n3) Region Place\n4) Region String");

int print;

while(!(int.TryParse(Console.ReadLine(), out print)) || print < 1 || print > 4)

{

Console.WriteLine("Ошибка. Тип печати с таким номером не существует. Повторите ввод");

}

switch (print)

{

case 1:

{

test.PrintList(new List<Region>());

}

break;

case 2:

{

test.PrintList(new List<string>());

}

break;

case 3:

{

test.PrintDictionary(new SortedDictionary<Place, Region>());

}

break;

case 4:

{

test.PrintDictionary(new SortedDictionary<string, Region>());

}

break;

}

}

break;

}

type = InputMode("номер задания", Chapter3Menu(), 4);

}

}

break;

}

}

}

Ссылка на ГитХаб:

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