|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Пермский филиал федерального государственного автономного образовательного учреждения высшего образования “Национальный исследовательский университет “Высшая школа экономики” | | | | |
|  | |  | |  |
|  | | *Факультет экономики, менеджмента и бизнес-информатики* | |  |
|  | |  | |  |
|  | | Дмитриев Арсений Алексеевич | |  |
|  | |  | |  |
| **Лабораторная работа №11. «Коллекции»**  *Отчет по практической работе* | | | | |
|  | |  | |  |
|  | студента образовательной программы «Программная инженерия»  по направлению подготовки 09.03.04 Программная инженерия | | |  |
|  | |  | |  |
|  | |  | руководитель  к. тех. н., доцент каф.  инф. техн. в биз.  Викентьева О.Л. | |
|  | |  | |  |
|  | | Пермь, 2020 г. | |  |

Постановка задачи

Общая

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

Задание 1.

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.

Задание 2.

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.

Задание 3.

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
   1. Коллекция\_1<TKey> ;
   2. Коллекция\_1<string> ;
   3. Коллекция\_2<TKey, TValue> ;
   4. 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.

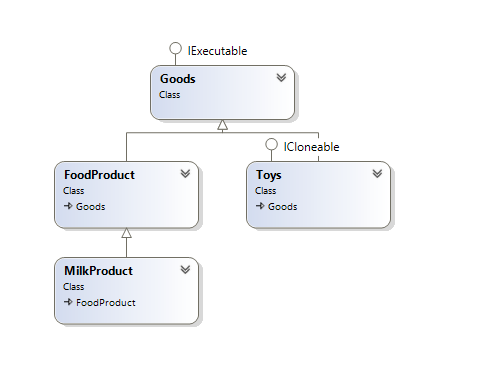
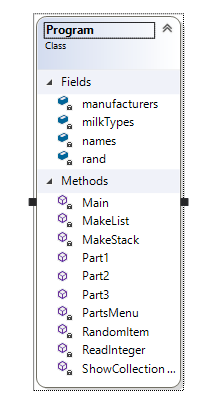
1. Написать конструктор класса TestCollections, в котором создаются коллекции с заданным числом элементов.
2. Предусмотреть автоматическую генерацию элементов коллекции таким образом, что каждый объект (Student) содержит подобъект базового класса (Person).
3. Все четыре коллекции должны содержать одинаковое 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).

Вариант №7

|  |  |  |  |
| --- | --- | --- | --- |
| Коллекции для заданий: | | |  |
| Задание 1 | Задание 2 | Задание 3 (1) | Задание 3 (2) |
| Stack | List<T> | LinkedList<T> | Dictionary <K,T> |

Задания 1 и 2

Диаграмма классов



Описание методов

Stack

Push() – Вставка нового элемента в верхушку стека

Pop() – Извлечение (возврат и удаление) верхнего элемента стека

Peek() – Возвращает (без удаления) верхний элемент стека

ToArray() – Возвращает все элементы стека в виде массива объектов

List<T>

Add() – Добавление элемента в конец листа

RemoveAt() – Удаление элемента по позиции (индексу) из листа

ToArray() – Возвращает все элементы листа в виде массива

Демонстрационная программа

Код: <https://github.com/AresDm0005/LabsTasks/tree/master/Lab11>

Задание 1

public static void Part1()

{

Console.WriteLine("Создание коллекции:");

int size = ReadInteger("Введите размер коллекции: ", 1, 25);

Stack<Goods> goods = MakeStack(size);

ShowCollection("Созданная коллекция:", ref goods);

bool finish = false;

do

{

PartsMenu();

int userChoise = ReadInteger("Опция: ", 0, 7);

switch (userChoise)

{

case 1:

{

goods.Push(RandomItem());

Console.WriteLine("Новый элемент:");

goods.Peek().Show();

break;

}

case 2:

{

Goods item = goods.Pop();

Console.WriteLine($"Элемент {item.Title} удален\n");

break;

}

case 3:

{

Console.WriteLine("Возможные производители:");

foreach (string txt in manufacturers) Console.Write(txt + " ");

Console.Write("\nВведите название интересующего производителя: ");

string manuf = Console.ReadLine();

int sum = 0, count = 0;

foreach (Goods good in goods)

{

if (good.Manufacturer == manuf)

{

sum += good.TotalRevenue();

count++;

}

}

if (count == 0) Console.WriteLine($"Товаров производителя {manuf} не найдено");

else Console.WriteLine($"Общая стоимость товаров, произведенных {manuf} = {sum} за {count} товаров");

break;

}

case 4:

{

int age = ReadInteger("Введите нижнюю возрастную границу: ", 0, 18);

int count = 0;

foreach (Goods good in goods.ToArray())

{

if (good is Toys)

{

Toys item = good as Toys;

if (item.AgeRestriction >= age)

{

count++;

Console.WriteLine($"Игрушка {item.Title}, производителя {item.Manufacturer}, имеет ограничение {item.AgeRestriction}+");

}

}

}

if (count == 0) Console.WriteLine("Игрушек с таким ограничением не найдено");

break;

}

case 5:

{

if (milkTypes == null) Console.WriteLine("Молочных продуктов не найдено");

else

{

Console.WriteLine("Возможные молочные продукты:");

foreach (string txt in milkTypes) Console.Write(txt + " ");

Console.Write("\nВведите интересующий молочный тип товара: ");

string type = Console.ReadLine();

Console.WriteLine();

int count = 0;

foreach (Goods good in goods)

{

if (good is MilkProduct)

{

MilkProduct item = (MilkProduct)good;

if (item.Type == type)

{

count++;

Console.WriteLine($"На складе есть {item.Quantity} штук {item.Type} {item.Title}");

}

}

}

if (count == 0) Console.WriteLine($"Молочных продуктов типа {type} не найдено");

}

break;

}

case 6:

{

Goods[] tmpGoods = goods.ToArray();

Array.Sort(tmpGoods, new DescendingSortByRevenue());

goods = new Stack<Goods>();

foreach (Goods good in tmpGoods) goods.Push(good);

ShowCollection("Отсортированная коллекция:", ref goods);

break;

}

case 7:

{

Stack<Goods> items = new Stack<Goods>();

foreach (Goods good in goods.ToArray()) items.Push((Goods)good.Clone());

int quantity = goods.Peek().Quantity;

goods.Peek().DeliverMade(100000);

Console.WriteLine("Goods: " + goods.Peek().Quantity);

Console.WriteLine("Cloned: " + items.Peek().Quantity);

goods.Peek().DeliverMade(quantity);

break;

}

default:

{

finish = true;

break;

}

}

Console.WriteLine();

} while (!finish);

}

Задание 2

public static void Part2()

{

Console.WriteLine("Создание коллекции:");

int size = ReadInteger("Введите размер коллекции: ", 1, 25);

List<Goods> goods = MakeList(size);

ShowCollection("Созданная коллекция:", ref goods);

bool finish = false;

do

{

PartsMenu();

int userChoise = ReadInteger("Опция: ", 0, 7);

switch (userChoise)

{

case 1:

{

goods.Add(RandomItem());

Console.WriteLine("Новый элемент:");

goods[goods.Count - 1].Show();

break;

}

case 2:

{

Console.WriteLine($"Элемент {goods[goods.Count - 1].Title} удален\n");

goods.RemoveAt(goods.Count - 1);

break;

}

case 3:

{

Console.WriteLine("Возможные производители:");

foreach (string txt in manufacturers) Console.Write(txt + " ");

Console.Write("\nВведите название интересующего производителя: ");

string manuf = Console.ReadLine();

int sum = 0, count = 0;

foreach (Goods good in goods)

{

if (good.Manufacturer == manuf)

{

sum += good.TotalRevenue();

count++;

}

}

if (count == 0) Console.WriteLine($"Товаров производителя {manuf} не найдено");

else Console.WriteLine($"Общая стоимость товаров, произведенных {manuf} = {sum} за {count} товаров");

break;

}

case 4:

{

int age = ReadInteger("Введите нижнюю возрастную границу: ", 0, 18);

int count = 0;

foreach (Goods good in goods.ToArray())

{

if (good is Toys)

{

Toys item = good as Toys;

if (item.AgeRestriction >= age)

{

count++;

Console.WriteLine($"Игрушка {item.Title}, производителя {item.Manufacturer}, имеет ограничение {item.AgeRestriction}+");

}

}

}

if (count == 0) Console.WriteLine("Игрушек с таким ограничением не найдено");

break;

}

case 5:

{

if (milkTypes == null) Console.WriteLine("Молочных продуктов не найдено");

else

{

Console.WriteLine("Возможные молочные продукты:");

foreach (string txt in milkTypes) Console.Write(txt + " ");

Console.Write("\nВведите интересующий молочный тип товара: ");

string type = Console.ReadLine();

Console.WriteLine();

int count = 0;

foreach (Goods good in goods)

{

if (good is MilkProduct)

{

MilkProduct item = (MilkProduct)good;

if (item.Type == type)

{

count++;

Console.WriteLine($"На складе есть {item.Quantity} штук {item.Type} {item.Title}");

}

}

}

if (count == 0) Console.WriteLine($"Молочных продуктов типа {type} не найдено");

}

break;

}

case 6:

{

Goods[] tmpGoods = goods.ToArray();

Array.Sort(tmpGoods, new DescendingSortByRevenue());

goods = new List<Goods>();

foreach (Goods good in tmpGoods) goods.Add(good);

ShowCollection("Отсортированная коллекция:", ref goods);

break;

}

case 7:

{

List<Goods> items = new List<Goods>();

int i = 0;

foreach (Goods good in goods) items.Add((Goods)good.Clone());

int quantity = goods[goods.Count - 1].Quantity;

goods[goods.Count - 1].DeliverMade(100000);

Console.WriteLine("Goods: " + goods[goods.Count - 1].Quantity);

Console.WriteLine("Cloned: " + items[goods.Count - 1].Quantity);

goods[goods.Count - 1].DeliverMade(quantity);

break;

}

default:

{

finish = true;

break;

}

}

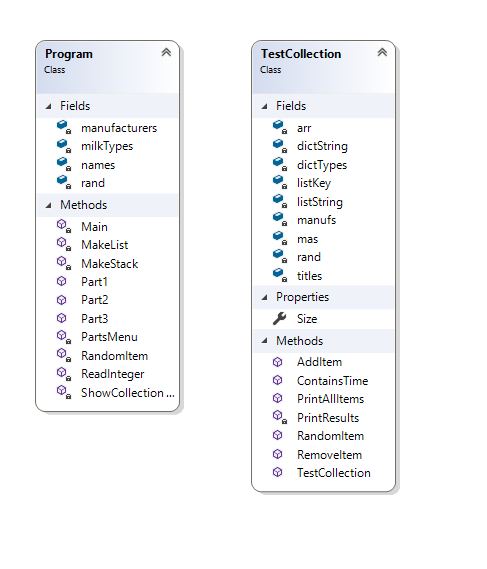
Console.WriteLine();

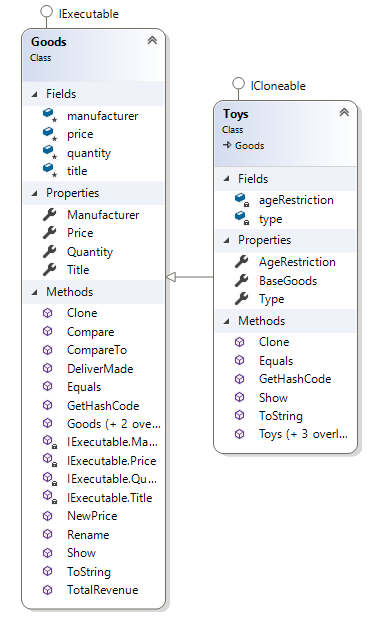
} while (!finish);

}

Заданиe 3

Диаграмма классов





Демонстрационная программа

Код: <https://github.com/AresDm0005/LabsTasks/tree/master/Lab11>

Class TestCollection

using System;

using System.Collections.Generic;

using System.Diagnostics;

using GoodsTypes;

namespace Lab11

{

class TestCollection

{

private LinkedList<Goods> listKey;

private LinkedList<string> listString;

private Dictionary<Goods, Toys> dictTypes;

private Dictionary<string, Toys> dictString;

private Goods[] arr;

private Toys[] mas;

private string[] titles = { "Монополия", "Домик", "Простоквашино", "Ряженка", "Маска", "Матроскин", "Яблоки", "Куб", "Треугольник", "Пакет", "Калька", "Чудо", "Языки", "Пластик" };

private string[] manufs = { "Nestle", "BMW", "Pepsi Co.", "Coca Cola Co.", "Нытвенский молзавод", "Hasbro", "Lego", "Asus" };

private static Random rand = new Random();

public int Size { get; private set; }

public Toys RandomItem()

{

string title = titles[rand.Next(titles.Length)];

string manuf = manufs[rand.Next(manufs.Length)];

int price = rand.Next(49, 2000);

int quantity = rand.Next(4, 51) \* 200;

string[] types = { "Мягкая игрушка", "Настольная игра", "Конструктор" };

string type = types[rand.Next(types.Length)];

int age = rand.Next(19);

Toys toys = new Toys(title, manuf, price, quantity, age, type);

Goods tmp = toys.BaseGoods;

return toys;

}

public TestCollection(int size)

{

listKey = new LinkedList<Goods>();

listString = new LinkedList<string>();

dictTypes = new Dictionary<Goods, Toys>(size);

dictString = new Dictionary<string, Toys>(size);

arr = new Goods[size];

mas = new Toys[size];

Size = size;

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

{

Toys toy = RandomItem();

arr[i] = toy.BaseGoods;

mas[i] = toy;

listKey.AddLast(toy.BaseGoods);

listString.AddLast(toy.BaseGoods.ToString());

dictTypes.Add(toy.BaseGoods, toy);

dictString.Add(toy.BaseGoods.ToString(), toy);

}

}

public void AddItem(Goods good, Toys toy)

{

if(dictTypes.ContainsKey((Goods)good.Clone()) || dictTypes.ContainsValue((Toys)toy.Clone()))

{

Console.WriteLine("Такой элемент уже существуют");

return;

}

listKey.AddLast((Goods)good.Clone());

listString.AddLast(good.ToString());

dictTypes.Add((Goods)good.Clone(), (Toys)toy.Clone());

dictString.Add(good.ToString(), (Toys)toy.Clone());

Size = listString.Count;

}

public void RemoveItem(Goods good)

{

if (!dictTypes.ContainsKey((Goods)good.Clone()))

{

Console.WriteLine("Такого ключа изначально не существует");

return;

}

listKey.Remove((Goods)good.Clone());

listString.Remove(good.ToString());

dictTypes.Remove((Goods)good.Clone());

dictString.Remove(good.ToString());

Size = listString.Count;

}

public void ContainsTime()

{

// 0 - Constains по listKey

// 1 - Contains по listString

// 2 - ContainsKey по dictTypes

// 3 - ContainsKey по dictString

// 4 - ContainsValue по dictTypes

Stopwatch[,] times = new Stopwatch[5, 4];

bool[,] found = new bool[5, 4];

Goods[] search = { (Goods)arr[0].Clone() , (Goods)arr[Size / 2].Clone(), (Goods)arr[Size - 1].Clone(), new Goods("Маски", "Медицинские",10,1000) };

Toys[] toys = { (Toys)mas[0].Clone(), (Toys)mas[Size/2].Clone(), (Toys)mas[Size-1].Clone(), new Toys("Маски", "Медицинские", 10, 1000, 3, "Обучающие") };

// 0

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

{

times[0, i] = new Stopwatch();

times[0, i].Start();

found[0, i] = listKey.Contains(search[i]);

times[0, i].Stop();

}

// 1

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

{

times[1, i] = new Stopwatch();

times[1, i].Start();

found[1, i] = listString.Contains(search[i].ToString());

times[1, i].Stop();

}

// 2

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

{

times[2, i] = new Stopwatch();

times[2, i].Start();

found[2, i] = dictTypes.ContainsKey(search[i]);

times[2, i].Stop();

}

// 3

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

{

times[3, i] = new Stopwatch();

times[3, i].Start();

found[3, i] = dictString.ContainsKey(search[i].ToString());

times[3, i].Stop();

}

// 4

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

{

times[4, i] = new Stopwatch();

times[4, i].Start();

found[4, i] = dictTypes.ContainsValue(toys[i]);

times[4, i].Stop();

}

PrintResults(ref times, ref found);

}

private void PrintResults(ref Stopwatch[,] times, ref bool[,] found)

{

Console.WriteLine("Коллекция1: LinkedList; Коллекция2: Dictionary\nTKey: Goods; TValue: Toys\n");

string[] collNames = { "Коллекция1<TKey>", "Коллекция1<string>", "Коллекция2<TKey, TValue>", "Коллекция2<string, TValue>", "Коллекция2<TKey, TValue>" };

string[] funcName = { "Contains", "Contains", "ContainsKey", "ContainsKey", "ContainsValue" };

Console.WriteLine($"Размер массива: {Size}\n");

for(int i = 0; i<times.GetLength(0); i++)

{

Console.WriteLine($"{collNames[i]}, метод: {funcName[i]}");

for (int j = 0; j < times.GetLength(1); j++)

{

string ans = found[i, j] ? "Да" : "Нет";

Console.WriteLine($"\t{times[i,j].Elapsed} мс. Найдено: {ans}");

}

Console.WriteLine();

}

}

public void PrintAllItems()

{

Console.WriteLine("Сгенерированные объекты:");

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

{

Console.WriteLine($"{i}:\nБазовый:\t{arr[i]}\nПроизводный:\t{mas[i]}");

}

Console.WriteLine("\n");

}

}

}

Демонстрационная программа

public static void Part3()

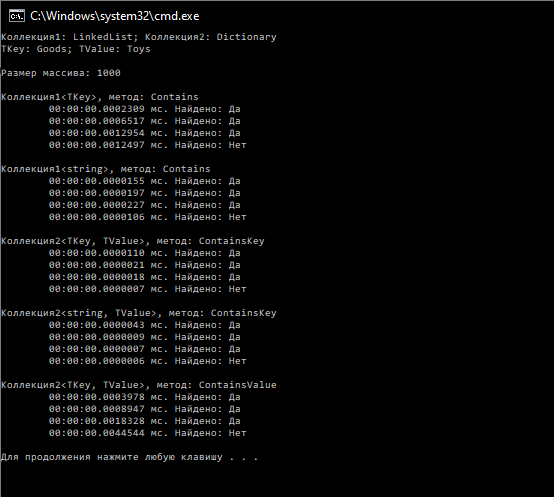
{

TestCollection test = new TestCollection(1000);

test.ContainsTime();

}

Результаты работы программы



Для каждой коллекции проверялось 4 объекта – 3 объекта, содержащиеся в коллекции (первый, средний, последний), и 1 случайный объект, не содержащийся в коллекции.

Быстрее всего операция поиска сработала на поиске по ключу в Dictionary<K, V>, так как операция поиска ключа в словаре требует 1-2 операции (вычисление хеш-значения ключа, проверка 1-2 элементов в корзине, с номером равном хешу). Следующая по скорости – коллекция LinkedList, содержащая строковые представления объектов Goods. Операция поиска в списке в среднем требует Length/2 операций (O(N)), из-за чего поиск в списке происходит медленнее, чем поиск по ключу в словаре. Далее по скорости – LinkedList, содержащий объекты класса Goods. Разница между списками со строками и объектами Goods объясняется разной скоростью сравнения данных. Худший по скорости – поиск в словаре по значению, так как в таком случае необходимо делать полный обход (как в листе), который происходит медленней.