Общая постановка задачи: Реализовать классы по индивидуальному

заданию. Создать параметризованную коллекцию(List&lt;object&gt;) для хранения описанного

класса. Написать программу для работы с коллекцией, позволяющее:

 добавление элемента в коллекцию;

 считывание данных из файла;

 запись данных в тот же или указанный файл;

 сортировку данных;

 поиск элемента по заданному критерию;

 вывод всех элементов, удовлетворяющих условию;

 удаление элемента из коллекции.

Класс “Ярлык”. Поля – имя ярлыка, дата создания, путь к файлу

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab19\_\_2\_

{

public class Label: IComparable<Label>

{

public string \_Name { get; set; }

public DateTime \_DateOfCreation { get; set; }

public string \_Path { get; set; }

public Label() { }

public int CompareTo(Label other)

{

return \_Name.CompareTo(other.\_Name);

}

public override string ToString()

{

return $"ярлык с названием{\_Name}, путем {\_Path}. Дата создания {\_DateOfCreation}";

}

}

}

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab19\_\_2\_

{

public class Program

{

static List<object> labelList = new List<object>();

static void Main(string[] args)

{

for (; ; )

{

int num = 0;

do

{

Console.WriteLine("выберите пункцт меню");

Console.WriteLine("1 - добавить элементв коллекцию");

Console.WriteLine("2 - считать данные из файла");

Console.WriteLine("3 - записать данные в файл");

Console.WriteLine("4 - сортировать данные ");

Console.WriteLine("5 - найти по выбарнному критерию");

Console.WriteLine("6 - вывод всех элементов");

Console.WriteLine("7 - удалить элемент из коллекции");

num = int.Parse(Console.ReadLine());

} while (num < 0);

Label label1 = new Label();

switch (num)

{

case 1:

{

label1.\_Path = "File.txt";

Console.WriteLine("введите навзание для ярдыка");

label1.\_Name = Console.ReadLine();

label1.\_DateOfCreation = DateTime.Now;

labelList.Add(label1);

break;

}

case 2:

{

string DataAboutFiles = File.ReadAllText("File.txt");

Console.WriteLine(DataAboutFiles);

break;

}

case 3:

{

WriteDataToFile();

Console.WriteLine("Даныые о вашем листе записаны");

break;

}

case 4:

{

labelList.Sort();

Console.WriteLine("сортировка прошла успешно");

break;

}

case 5:

{

Console.WriteLine("Выберите критерий для поиска:");

Console.WriteLine("1 - По названию");

Console.WriteLine("2 - По дате создания");

int criteria = int.Parse(Console.ReadLine());

switch (criteria)

{

case 1:

Console.WriteLine("Введите название для поиска:");

string searchName = Console.ReadLine();

List<Label> searchResultsByName = labelList.OfType<Label>().Where(label => label.\_Name == searchName).ToList();

if (searchResultsByName.Any())

{

Console.WriteLine("Результаты поиска по названию:");

foreach (Label result in searchResultsByName)

{

Console.WriteLine(result.ToString());

}

}

else

{

Console.WriteLine("Нет результатов для введенного названия.");

}

break;

case 2:

Console.WriteLine("Введите дату для поиска (в формате ГГГГ-ММ-ДД):");

if (DateTime.TryParse(Console.ReadLine(), out DateTime searchDate))

{

List<Label> searchResultsByDate = labelList.OfType<Label>().Where(label => label.\_DateOfCreation.Date == searchDate.Date).ToList();

if (searchResultsByDate.Any())

{

Console.WriteLine("Результаты поиска по дате создания:");

foreach (Label result in searchResultsByDate)

{

Console.WriteLine(result.ToString());

}

}

else

{

Console.WriteLine("Нет результатов для введенной даты.");

}

}

else

{

Console.WriteLine("Некорректный формат даты.");

}

break;

default:

Console.WriteLine("Некорректный критерий поиска.");

break;

}

break;

}

case 6:

{

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

{

Console.WriteLine(labelList[i].ToString());

}

break;

}

case 7:

{

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

{

Console.WriteLine(i + " " + labelList[i].ToString());

}

int k = 0;

do

{

Console.WriteLine("выберите номер ярлыка для удаления ");

k = int.Parse(Console.ReadLine());

} while (k < 0 || k > labelList.Count);

RemoveLabel(k);

break;

}

}

}

}

static void WriteDataToFile()

{

try

{

using (StreamWriter writer = new StreamWriter("File.txt"))

{

foreach (object item in labelList)

{

writer.WriteLine(item.ToString());

}

}

}

catch (IOException)

{

Console.WriteLine("Ошибка при записи данных в файл");

}

}

static void RemoveLabel(int i)

{

labelList.Remove(labelList[i]);

}

}

}

Задание 2. Решить следующие задачи с использованием класса Queue.

Дан файл, содержащий информацию о сотрудниках фирмы: фамилия, имя, отчество, пол,

возраст, размер зарплаты. За один просмотр файла напечатать элементы файла в

следующем порядке: сначала все данные о мужчинах, потом все данные о женщинах,

сохраняя исходный порядок в каждой группе сотрудников.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab19\_1\_2

{

internal class Employer

{

public string \_name { get; set; }

public string \_surname { get; set; }

public string \_sex { get; set; }

public int \_salary { get; set; }

public Employer() { }

public Employer (string name, string surname, bool sex, int salary)

{

\_name = name;

\_surname = surname;

\_salary = salary;

\_sex = (sex == true) ? "Мужской" : "Женский";

}

public Employer(string name, string surname, string sex, string salary)

{

\_name = name;

\_surname = surname;

\_salary = int.Parse(salary);

\_sex = sex;

}

public override string ToString()

{

return $"{\_surname} {\_name} {\_sex} {\_salary}";

}

}

}

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

namespace Lab19\_1\_2

{

internal class Program

{

static void Main(string[] args)

{

Queue <Employer> MQueaue = new Queue<Employer>();

Queue <Employer> WQueaue = new Queue<Employer>();

Employer[] EmpAray = new Employer[10]

{

new Employer("A", "a", true, 1),

new Employer("B", "b", false, 1),

new Employer("C", "c", true, 1),

new Employer("D", "d", false, 1),

new Employer("E", "E", true, 1),

new Employer("F", "f", false, 1),

new Employer("G", "g", true, 1),

new Employer("H", "h", true, 1),

new Employer("I", "i", false, 1),

new Employer("J", "j", false, 1)

};

StreamWriter writer = new StreamWriter("Employers.txt");

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

{

writer.WriteLine(EmpAray[i]);

}

writer.Close();

string[] lines = File.ReadAllLines("Employers.txt");

Regex ManPattern = new Regex("Мужской");

Regex WomamanPattern = new Regex("Женский");

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

{

if (ManPattern.IsMatch(lines[i]))

{

string[] ch = lines[i].Split(' ');

Employer man = new Employer(ch[0], ch[1], ch[2], ch[3]);

MQueaue.Enqueue(man);

}

else if (WomamanPattern.IsMatch(lines[i]))

{

string[] ch = lines[i].Split(' ');

Employer woman = new Employer(ch[0], ch[1], ch[2], ch[3]);

WQueaue.Enqueue(woman);

}

else

{

Console.WriteLine($"ошибка с сотрудником {i}");

}

}

var e = MQueaue.Concat( WQueaue );

foreach (var a in e )

{

Console.WriteLine(a);

}

Console.ReadKey();

}

}

}

Решить задачи, используя класс ArrayList.

Во всех задачах данного пункта подразумевается, что исходная информация хранится в

текстовом файле input.txt, каждая строка которого содержит полную информацию о

некотором объекте, результирующая информация должна быть записана в файл output.txt.

Для хранения данных внутри программы использовать класс ArrayList. Непосредственно для

хранения информации об одном объекте использовать структуру. Реализовать метод

CompareTo интерфейса IComparable, перегрузить метод ToString базового класса object и

необходимые операции отношения, поля данных и дополнительные методы продумать

самостоятельно.

На основе данных входного файла составить инвентарную ведомость склада, включив

следующие данные: вид продукции, стоимость, сорт, количество. Вывести в новый файл

информацию о той продукции, количество которой менее заданной величины, отсортировав

ее по количеству продукции на складе.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab19\_1\_3

{

public struct Item: IComparable

{

public string \_type;

public double \_price;

public int \_amount;

public Item(string type, string price, string amount)

{

\_type = type;

\_price = Math.Round(double.Parse(price), 2);

\_amount = int.Parse(amount);

}

public int CompareTo(object obj)

{

Item a = (Item)obj;

if (this.\_amount > a.\_amount) return 1;

else if (this.\_amount < a.\_amount) return -1;

else return 0;

}

public override string ToString()

{

return $"{\_type} {\_price} {\_amount}";

}

}

}

using System;

using System.Collections;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab19\_1\_3

{

internal class Program

{

static void Main(string[] args)

{

ArrayList ItemList = new ArrayList();

string[] lines = File.ReadAllLines("items1.txt");

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

{

string[] c = lines[i].Split(' ');

Item item = new Item(c[0], c[1], c[2]);

ItemList.Add(item);

Console.WriteLine(item);

}

ItemList.Sort();

Console.WriteLine();

StreamWriter writer = new StreamWriter("items2.txt");

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

{

writer.WriteLine((ItemList[i]).ToString());

Console.WriteLine(ItemList[i].ToString());

}

writer.Close();

Console.ReadKey();

}

}

}

Задание 4 (общее). Решить задачу, используя класс HashTable.

Реализовать простейший каталог музыкальных компакт-дисков, который позволяет:

o Добавлять и удалять диски.

o Добавлять и удалять песни.

o Просматривать содержимое целого каталога и каждого диска в отдельности.

o Осуществлять поиск всех записей заданного исполнителя по всему каталогу

using System;

using System.Collections;

namespace Lab19\_1\_4

{

public class Disk

{

public string \_name { get; set; }

Hashtable hashtable = new Hashtable();

public Disk (string name)

{

\_name = name;

}

public override string ToString()

{

return $"{\_name}";

}

public void PrtntSongsWithSinger(string SingerName)

{

if (hashtable.Count == 0)

{

Console.WriteLine("на диске нету музыки ");

}

else

{

foreach (var value in hashtable.Keys)

{

if (hashtable[value].ToString() == SingerName)

{

Console.WriteLine($"песня: {value} Исполнитель: {hashtable[value]}");

}

}

}

}

public void PrtntSongs()

{

if (hashtable.Count == 0)

{

Console.WriteLine("на диске нету музыки ");

}

else

{

foreach (var value in hashtable.Keys)

{

Console.WriteLine($"песня: {value} Исполнитель: {hashtable.Values}");

}

}

}

public void AddSong(string song, object singer)

{

if (!hashtable.ContainsKey(song))

{

hashtable.Add(song, singer);

Console.WriteLine("Элемент успешно добавлен.");

}

else

{

Console.WriteLine("Элемент с таким ключом уже существует.");

}

}

public void DeleteSong(string song)

{

if (hashtable.ContainsKey(song))

{

hashtable.Remove(song);

Console.WriteLine("Элемент успешно удален.");

}

else

{

Console.WriteLine("Такой песни нету на данном диске");

}

}

}

}