**Class.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

abstract class Developer//разработчик

{

public Developer(string namedev)

{

this.Namedev = namedev;

}

private string namedev;

public string Namedev

{

get

{

return namedev;

}

set

{

namedev = value;

}

}

public virtual void infodev()

{

Console.WriteLine("Имя" + namedev);

}

public override string ToString()

{

return base.ToString() + " Имя автора " + namedev;

}

}

class PO : Developer,IComparable<PO>

{

private string name;

public PO(string name, string namedev) : base(namedev)

{

this.Name = name;

}

public string Name

{

get

{

return name;

}

set

{

if (value == "")

{

Console.WriteLine("Вы ввели пустое значение");

}

else

{

name = value;

}

}

}

public int CompareTo(PO obj)

{

if(obj!=null)

{

return this.Name.CompareTo(obj.Name);

}

else

{

throw new Exception("Параметры заданы не верно");

}

}

public override string ToString()

{

return base.ToString() + " Название продукта " + name;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Collections;

using System.Collections.ObjectModel;

using System.Collections.Specialized;

namespace ConsoleApp1

{

class Program

{

static void Main(string[] args)

{

ArrayList obj = new ArrayList();

obj.Add(5);

obj.Add(7);

obj.Add(2);

obj.Add(120);

obj.Add(17);

obj.Add("Hello World");

obj.Add(new Student("Katya"));

obj.RemoveAt(2);

Console.WriteLine(obj.Count);

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

{

if(obj[i] is Student)

{

Console.WriteLine(((Student)obj[i]).name);

}

else if (obj[i] is int )

{

Console.WriteLine((int)obj[i]);

}

else

{

Console.WriteLine((string)obj[i]);

}

}

Console.WriteLine(obj.Contains(70));

Stack<char> stack1 = new Stack<char>();

List<char> list = new List<char>();

for(char i='c';i<'f';i++)

{

stack1.Push(i);

}

for(char i='c';i<'f';i++)

{

list.Add(i);

}

foreach(char a in stack1)

{

Console.Write(a);

}

Console.WriteLine();

foreach(char a in list)

{

Console.Write(a);

}

Console.WriteLine();

list.RemoveRange(0, 2);

Console.WriteLine("После удаления : ");

foreach (char a in list)

{

Console.Write(a);

}

//найти во 2 коллеции заданное значение

Console.WriteLine(list.Contains('e'));

//3

Stack<PO> po = new Stack<PO>();

po.Push(new PO("laba1", "GIT"));

po.Push(new PO("laba2", "C#"));

po.Push(new PO("laba3", "OVERRIDE"));

po.Push(new PO("laba4", "NEWMETHOD"));

foreach(PO a in po)

{

Console.Write(a.Name+" ");

Console.WriteLine(a.Namedev);

}

po.Pop();

List<PO> pol = new List<PO>();

pol.Add(new PO("laba5", "OOP"));

pol.Add(new PO("laba6", "Exception"));

pol.Add(new PO("laba7", "Delegate"));

foreach(PO a in pol)

{

Console.Write(a.Name + " ");

Console.WriteLine(a.Namedev);

}

Console.WriteLine(pol.Contains(new PO("laba5", "OOP")));

ObservableCollection < PO > myPo= new ObservableCollection<PO>();

myPo.CollectionChanged += PO\_CollectionChanged;

myPo.Add(new PO("Iphone", "laba9"));

}

private static void PO\_CollectionChanged(object sender, NotifyCollectionChangedEventArgs e)

{

switch (e.Action)

{

case NotifyCollectionChangedAction.Add: // если добавление

PO newUser = e.NewItems[0] as PO;

Console.WriteLine("Добавлен новый объект: {0}", newUser.Name);

break;

case NotifyCollectionChangedAction.Remove: // если удаление

PO oldUser = e.OldItems[0] as PO;

Console.WriteLine("Удален объект: {0}", oldUser.Name);

break;

case NotifyCollectionChangedAction.Replace: // если замена

PO replacedUser = e.OldItems[0] as PO;

PO replacingUser = e.NewItems[0] as PO;

Console.WriteLine("Объект {0} заменен объектом {1}",

replacedUser.Name, replacingUser.Name);

break;

}

}

}

}