Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Новосибирский государственный технический университет»

Кафедра Автоматизированных систем управления

**Практическая работа №3**

по дисциплине «Программирование»

|  |  |
| --- | --- |
| Выполнили:  Кашин Андрей Вячеславович Мосина Виктория Владимировна  Группа: АВТ-312  Факультет: АВТ.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_2024 г. | Проверил:  Достовалов Дмитрий Николаевич  Балл: \_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г. |

Новосибирск 2024

Оглавление

[Постановка задачи 3](#_Toc180964353)

[Тестирование программы 4](#_Toc180964354)

[Исходный код 5](#_Toc180964355)

[Class.cs 5](#_Toc180964356)

[1task.cs 6](#_Toc180964357)

[2task.cs 7](#_Toc180964358)

[3task.cs 8](#_Toc180964359)

[4task.cs 9](#_Toc180964360)

Постановка задачи

Вариант 9

*Первая коллекция*: Queue<T> *Первый тип*: int *Вторая коллекция*: List<T>

1. Создайте необобщенную коллекцию **ArrayList**.

a) Заполните ее 5-ю случайными целыми числами

b) Добавьте к ней строку

c) Удалите заданный элемент

d) Выведите количество элементов и коллекцию на консоль.

e) Выполните поиск в коллекции заданного значения

2. Создайте **обобщенную коллекцию** в соответствии с вариантом задания и заполните ее данными, тип которых определяется вариантом задания (колонка – первый тип).

a) Выведите коллекцию на консоль

b) Удалите из коллекции n элементов

c) Добавьте другие элементы (используйте все возможные методы добавления для вашей коллекции).

d) Создайте вторую коллекцию (см. таблицу) и заполните ее данными из первой коллекции. В случае не совпадения количества параметров (например, **LinkedList<T>** и **Dictionary<Tkey, TValue>)**, при нехватке - генерируйте ключи, в случае избыточности – оставляйте **TValue**.

e) Выведите вторую коллекцию на консоль.

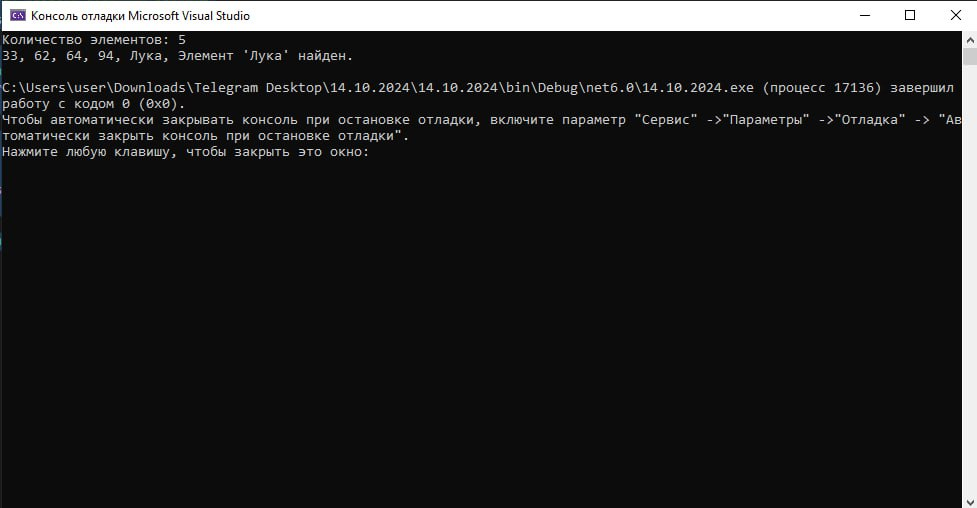
f) Найдите во второй коллекции заданное значение.

3. Повторите задание п.2 для **пользовательского типа данных** (в качестве типа **T** возьмите любой свой класс из лабораторной работы № 1). Реализуйте интерфейсы **IComparable** и **IComparer** для сравнения объектов вашего класса и интерфейс **ICloneable** для глубокого копирования объектов. Продемонстрируйте их использование. При выводе коллекций используйте цикл **foreach**.

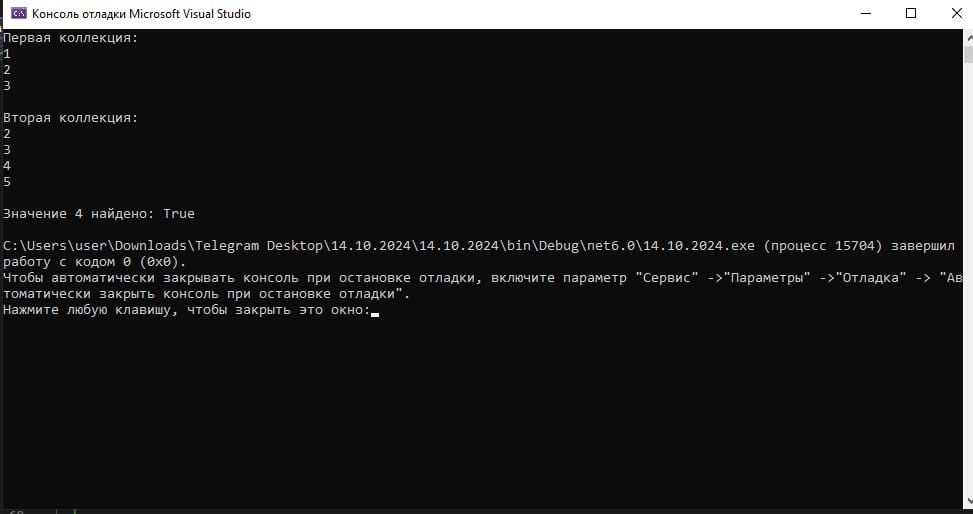
4. Создайте объект наблюдаемой коллекции **ObservableCollection<T>**. Создайте произвольный метод и зарегистрируйте его на событие **CollectionChange**. Напишите демонстрацию с добавлением и удалением элементов. В качестве типа T используйте свой класс из лабораторной работы № 1.

Тестирование программы

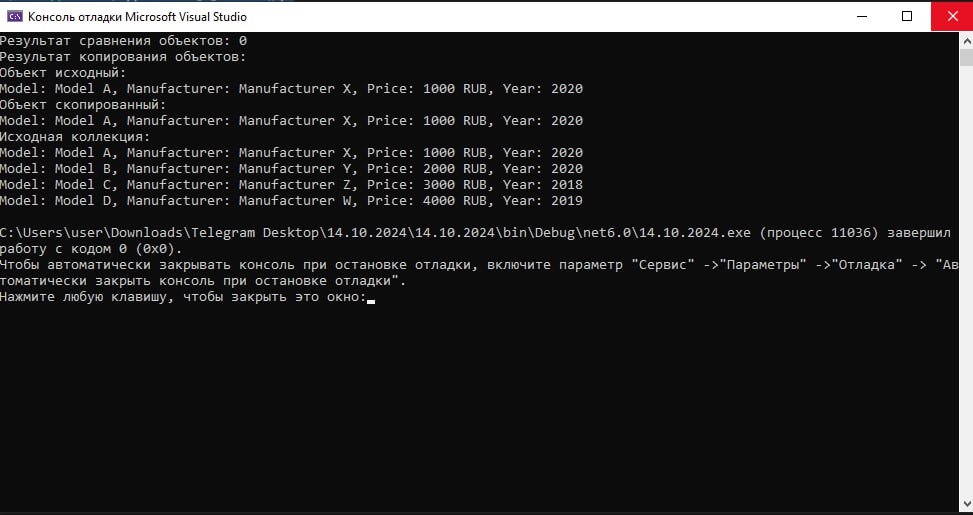
1 задание



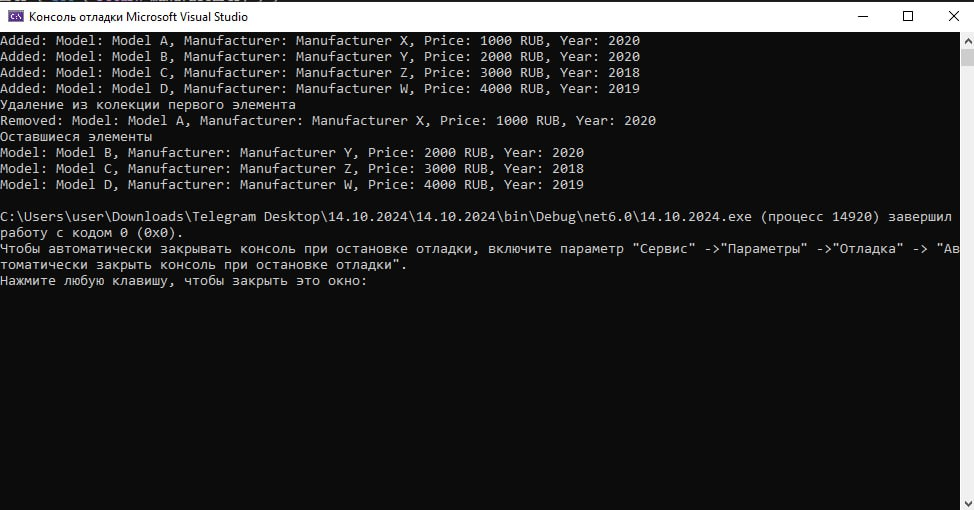
2 задание



3 задание



4 задание



Исходный код

## Class.cs

|  |
| --- |
| using System;  using System.Collections;  using System.Collections.Generic;  using System.Diagnostics;  using System.Runtime.ConstrainedExecution;  using System.Text;  using System.Threading.Tasks;  using System.Xml.Linq;  //Базовый класс Бытовая техника  public class HomeAppliance  {  protected string model;  protected string manufacturer;  protected double price;  protected int year;  public HomeAppliance(string m = "Unknown", string manu = "Unknown", double p = 0.0, int y = 2000)  {  model = m;  manufacturer = manu;  price = p;  year = y;  }  public override string ToString()  {  return $"Model: {model}, Manufacturer: {manufacturer}, Price: {price} RUB, Year: {year}";  }  public interface ICloneable  {  object Clone();  }  public object Clone()  {  return new HomeAppliance(model,manufacturer,price,year);  }  public virtual void Print()  {  Console.WriteLine($"Model: {model}, Manufacturer: {manufacturer}, Price: {price} RUB, Year: {year}");  }  // Геттеры  public string GetModel { get { return model; } }  public string GetManufacturer { get { return manufacturer; } }  public double GetPrice { get { return price; } }  public int GetYear { get { return year; } }  // Сеттеры  public double SetPrice  {  set  {  if (value < 0)  throw new ArgumentException("Price cannot be negative.");  else  price = value;  }  }  public int SetYear  {  set  {  int currentYear = 2024;  if (value < 1980 || value > currentYear)  throw new ArgumentException($"Year must be between 1980 and {currentYear}.");  else  year = value;  }  }  }  public class ApplianceComparer : IComparer<HomeAppliance>  {  public int Compare(HomeAppliance x, HomeAppliance y)  {  if (x.GetYear != y.GetYear)  return 0;  else if (x.GetPrice != y.GetPrice)  return 0;  else if (x.GetModel != y.GetModel)  return 0;  else return 1;  }  } |

## 1task.cs

|  |
| --- |
| using System;  using System.Collections;  using System.Collections.ObjectModel;  using System.Collections.Specialized;  using System.Diagnostics;  using System.Linq;  //Задание 1  namespace otter  {  class Program  {  static void Main(string[] args)  {  string searchValue = "Лука";  int indexToRemove = 4;  // a) Создаем пустую коллекцию ArrayList  ArrayList list = new ArrayList();  // b) Заполняем список 5-ю случайными целыми числами  Random rnd = new Random();  for (int i = 0; i < 5; i++)  list.Add(rnd.Next(100));  // c) Добавляем строку  list.Add("Лука");  // d) Удаляем заданный элемент  list.RemoveAt(indexToRemove);  // e) Выводим количество элементов и коллекцию на консоль  Console.WriteLine($"Количество элементов: {list.Count}");  foreach (var item in list)  Console.Write($"{item}, ");  // f) Выполняем поиск в коллекции заданного значения  if (list.Contains(searchValue))  Console.WriteLine($"Элемент '{searchValue}' найден.");  else  Console.WriteLine($"Элемент '{searchValue}' не найден.");  Console.ReadKey();  }  }  } |

## 2task.cs

|  |
| --- |
| using System;  using System.Collections;  using System.Collections.ObjectModel;  using System.Collections.Specialized;  using System.Diagnostics;  using System.Linq;  class Program  {  static void Main()  {  int n = 1;  int searchValue = 4;  // a) Первая коллекция  Queue<int> queue = new Queue<int>();  queue.Enqueue(1);  queue.Enqueue(2);  queue.Enqueue(3);  // b) Вывод первой коллекции на консоль  Console.WriteLine("Первая коллекция:");  foreach (var item in queue)  {  Console.WriteLine(item);  }  // c) Удалить n элементов  while (n > 0)  {  queue.Dequeue();  n--;  }  // d) Добавить другие элементы  queue.Enqueue(4);  queue.Enqueue(5);  // e) Создание второй коллекции  List<int> list = new List<int>(queue);  // f) Вывести вторую коллекцию на консоль  Console.WriteLine("\nВторая коллекция:");  foreach (var item in list)  {  Console.WriteLine(item);  }  // g) Найти заданное значение  bool isFound = list.Exists(x => x == searchValue);  Console.WriteLine($"\nЗначение {searchValue} найдено: {isFound}");  Console.ReadKey();  }  } |

## 3task.cs

|  |
| --- |
| using System;  using System.Collections;  using System.Collections.ObjectModel;  using System.Collections.Specialized;  using System.Diagnostics;  using System.Linq;  class Program  {  static void Main()  {  var appliances = new List<HomeAppliance>();  appliances.Add(new HomeAppliance("Model A", "Manufacturer X", 1000.0, 2020));  appliances.Add(new HomeAppliance("Model B", "Manufacturer Y", 2000.0, 2020));  appliances.Add(new HomeAppliance("Model C", "Manufacturer Z", 3000.0, 2018));  appliances.Add(new HomeAppliance("Model D", "Manufacturer W", 4000.0, 2019));  List<HomeAppliance> appliancess = new List<HomeAppliance>()  {  new HomeAppliance("Washing Machine", "Samsung", 20000, 2020),  new HomeAppliance("Refrigerator", "LG", 60000, 2020),  new HomeAppliance("Air Conditioner", "Panasonic", 70000, 2022),  new HomeAppliance("Vacuum", "Xiaomi", 10000, 2023),  new HomeAppliance("Microwave Oven", "Whirlpool", 30000, 2020)  };  ApplianceComparer comparer = new ApplianceComparer();  appliances.Sort(comparer);  // Сравнение двух объектов  var appA = appliancess[0];  var appB = appliancess[1];  int comparisonResult = comparer.Compare(appA, appB);  Console.WriteLine("Результат сравнения объектов: " + comparisonResult);  // Глубокое копирование объекта  var clonedAppliance = (HomeAppliance)appliances[0].Clone();  Console.WriteLine("Результат копирования объектов: ");  Console.WriteLine("Объект исходный: ");  appliances[0].Print();  Console.WriteLine("Объект скопированный: ");  clonedAppliance.Print();  // Вывод коллекции  Console.WriteLine("Исходная коллекция:");  foreach (var appliance in appliances)  {  appliance.Print();  }  Console.ReadKey();  }  } |

## 4task.cs

|  |
| --- |
| using System;  using System.Collections;  using System.Collections.ObjectModel;  using System.Collections.Specialized;  using System.Diagnostics;  using System.Linq;  class Program  {  static void Main(string[] args)  {  // Создаем коллекцию и регистрируем обработчик события  ObservableCollection<HomeAppliance> appliances = new ObservableCollection<HomeAppliance>();  appliances.CollectionChanged += Appliances\_CollectionChanged;  // Добавляем элементы в коллекцию  appliances.Add(new HomeAppliance("Model A", "Manufacturer X", 1000.0, 2020));  appliances.Add(new HomeAppliance("Model B", "Manufacturer Y", 2000.0, 2020));  appliances.Add(new HomeAppliance("Model C", "Manufacturer Z", 3000.0, 2018));  appliances.Add(new HomeAppliance("Model D", "Manufacturer W", 4000.0, 2019));  // Удаляем элемент из коллекции  Console.WriteLine("Удаление из колекции первого элемента");  appliances.RemoveAt(0); // Удаляем первый элемент  // Выводим оставшиеся элементы  Console.WriteLine("Оставшиеся элементы");  foreach (var appliance in appliances)  {  appliance.Print();  }  }  private static void Appliances\_CollectionChanged(object sender, NotifyCollectionChangedEventArgs e)  {  if (e.Action == NotifyCollectionChangedAction.Add)  {  foreach (HomeAppliance newAppliance in e.NewItems)  {  Console.WriteLine($"Added: {newAppliance}");  }  }  else if (e.Action == NotifyCollectionChangedAction.Remove)  {  foreach (HomeAppliance oldAppliance in e.OldItems)  {  Console.WriteLine($"Removed: {oldAppliance}");  }  }  }  } |