1.     ArrayList.  Продемонструвати мінімум 4 методи, сортування виконати обов‘язкового. Членом колекції повинні бути вбудовані типи та об‘єкти Вашого класу.

2.     Неузагальненої колекції, яка не розглядалась на лекції.

**Примітка: Членами колекцій повинні бути члени класу Вашого варіанту. А сама колекція повинна міститися в іншому класі, який за змістом може включати таку колекцію. Або наданий Вам клас повинен включати** колекцію, членами якої будуть об‘єкту класу підходящого за змістом.

Завдання2: Використовуючи один із класів свого варіанту, створіть мінімум **два** застосунки, які демонструю роботу  узагальнених колекцій:

1.     List або інша колекція за Вашим вибором – продемонструвати мінімум 4 методи. Членом колекції повинні об‘єкти Вашого класу.

2.     Продемонструйте використання обмежень для узагальнених класів.

3.     Реалізувати HashSet. В якості членів використовуйте об‘єкти Вашого класу.

using System;  
using System.Collections;  
using System.Collections.Generic;  
using System.Collections.ObjectModel;  
using System.Collections.Specialized;  
namespace ConsoleApp1  
{  
 public class MyReverserClass : IComparer  
 {  
 int IComparer.Compare(Object x, Object y)  
 {  
 return ((new CaseInsensitiveComparer()).Compare(y?.ToString(), x?.ToString()));  
 }  
 }  
 class Test: IEquatable<Test>, IComparable<Test>  
 {  
 private static int *\_id*;  
  
 public readonly int id;  
 public int Mark { get; set; }  
 public string Name { get; set; }  
 static Test() => *\_id* = 0;  
  
 public Test(int mark, string name)  
 {  
 id = ++*\_id*;  
 Name = name;  
 Mark = mark;  
 }  
 public override string ToString() => $"Name of test - \"{Name}\". The highest mark you can get is {Mark}";  
  
 public int CompareTo(Test other) => Equals(null, other) ? 0 : Mark.CompareTo(other.Mark);  
 public bool Equals(Test other) => !Equals(null, other) && this.Name.Equals(other.Name);  
  
 public override bool Equals(object obj) => !Equals(null, obj) && Equals(obj as Test);  
   
 public override int GetHashCode() => id;  
 }  
  
 internal class Exam<T> : IList<T> where T:Test  
 {  
 private List<T> \_tests;  
  
 public Exam(List<T> tests)  
 {  
 \_tests = tests;  
 }  
  
 public T this[int index] { get => ((IList<T>)\_tests)[index]; set => ((IList<T>)\_tests)[index] = value; }  
  
 public int Count => ((ICollection<T>)\_tests).Count;  
  
 public bool IsReadOnly => ((ICollection<T>)\_tests).IsReadOnly;  
  
 public void Add(T item)  
 {  
 ((ICollection<T>)\_tests).Add(item);  
 }  
  
 public void Clear()  
 {  
 ((ICollection<T>)\_tests).Clear();  
 }  
  
 public bool Contains(T item)  
 {  
 return ((ICollection<T>)\_tests).Contains(item);  
 }  
  
 public void CopyTo(T[] array, int arrayIndex)  
 {  
 ((ICollection<T>)\_tests).CopyTo(array, arrayIndex);  
 }  
  
 public IEnumerator<T> GetEnumerator()  
 {  
 return ((IEnumerable<T>)\_tests).GetEnumerator();  
 }  
  
 public int IndexOf(T item)  
 {  
 return ((IList<T>)\_tests).IndexOf(item);  
 }  
  
 public void Insert(int index, T item)  
 {  
 ((IList<T>)\_tests).Insert(index, item);  
 }  
  
 public bool Remove(T item)  
 {  
 return ((ICollection<T>)\_tests).Remove(item);  
 }  
  
 public void RemoveAt(int index)  
 {  
 ((IList<T>)\_tests).RemoveAt(index);  
 }  
  
 IEnumerator IEnumerable.GetEnumerator()  
 {  
 return ((IEnumerable)\_tests).GetEnumerator();  
 }  
 }  
  
   
 class Program  
 {  
 static void Main(string[] args)  
 {  
 Console.WriteLine("ArrayList");  
  
 ArrayList arrayLis = new ArrayList();  
   
 arrayLis.AddRange(new object[] {new Test(1,"Test")});  
 arrayLis.Add("The");  
 arrayLis.Add("quick");  
 arrayLis.Add("brown");  
 arrayLis.Add("fox");  
 arrayLis.AddRange(new object[] { "objValue2", 3, 4.0f });  
  
 Console.WriteLine("\tBefore");  
   
 PrintIndexAndValues(arrayLis);  
  
 arrayLis.Sort(new MyReverserClass());  
  
 Console.WriteLine("\tAfter");  
   
 PrintIndexAndValues(arrayLis);  
  
 arrayLis.RemoveAt(arrayLis.IndexOf("fox"));  
  
 Console.WriteLine("\tReverse");  
  
 arrayLis.Reverse();  
  
 PrintIndexAndValues(arrayLis);  
  
 arrayLis.Clear();  
   
 Console.WriteLine("List");  
  
 List<Test> tests = new List<Test>()  
 {  
 new Test(1, "First"),  
 new Test(2, "Second"),  
 new Test(3, "Third")  
 };  
 tests.Add(new Test(4, "Fourth"));  
 Console.WriteLine(tests.Contains(new Test(3, "Third")));  
 tests.Sort();  
   
 PrintIndexAndValues(tests);  
   
 Console.WriteLine("Queue");  
   
 Queue testQueue = new Queue();  
   
 testQueue.Enqueue(new Test(1, "First"));  
 testQueue.Enqueue(obj: 2);  
 testQueue.Enqueue(obj: "Third");  
  
 PrintIndexAndValues(testQueue);  
  
 Console.WriteLine("HashSet");  
   
 HashSet<Test> testSet = new HashSet<Test>();  
   
 testSet.Add(new Test(1, "First"));  
 testSet.Add(new Test(2, "Second"));  
 testSet.Add(new Test(3, "Third"));  
  
 PrintIndexAndValues(testSet);  
   
 ObservableCollection<Test> users = new ObservableCollection<Test>  
 {  
 new Test(1, "First"),  
 new Test(2, "Second"),  
 new Test(3, "Third")  
 };  
 users.CollectionChanged += TestCollectionChanged;  
 users.Add(new Test(4, "Fourth"));  
 users[1] = new Test(5, "Fight");  
 Console.ReadKey();  
 }  
 public static void PrintIndexAndValues(IEnumerable myList)  
 {  
 int i = 0;  
 foreach (Object obj in myList)  
 Console.WriteLine("\t[{0}]:\t{1}", i++, obj);  
 Console.WriteLine();  
 }  
 private static void TestCollectionChanged(object sender, NotifyCollectionChangedEventArgs e)  
 {  
 switch (e.Action)  
 {  
 case NotifyCollectionChangedAction.*Add*:  
 Test newTest = e.NewItems?[0] as Test;  
 Console.WriteLine($"Добавлен новый объект: {newTest?.Name}");  
 break;  
 case NotifyCollectionChangedAction.*Remove*:  
 Test oldTest = e.OldItems?[0] as Test;  
 Console.WriteLine($"Удален объект: {oldTest?.Name}");  
 break;  
 case NotifyCollectionChangedAction.*Replace*:  
 Test replacedTest = e.OldItems?[0] as Test;  
 Test replacingTest = e.NewItems?[0] as Test;  
   
 Console.WriteLine($"Объект {replacedTest?.Name} заменен объектом {replacingTest?.Name}");  
 break;  
 }  
 }  
 }  
}