**Вопросы к лабораторной работе №9:**

1. **На какие основные виды/типы делятся все коллекции .NET? Охарактеризуйте каждый из них.**
   * **Необобщенные;**

- наличие разнотипных данных

- ссылки на данные типа object (не обеспечивают типовую безопасность)

- System.Collections

* + **Обобщенные;**

- обеспечивают обобщенную реализацию нескольких стандартных структур

данных, включая связные списки, стеки, очереди и словари

- типизированные в силу своего обобщенного характера (обеспечивают типовую безопасность)

- System.Collections.Generic

* + **Специальные;**

**-** оперируют данными конкретного типа или же делают это каким-то особым образом

- System.Collections.Specialized

* + **С поразрядной организацией (поразрядная коллекция);**

- BitArray

- System.Collections

* + **Параллельные;**

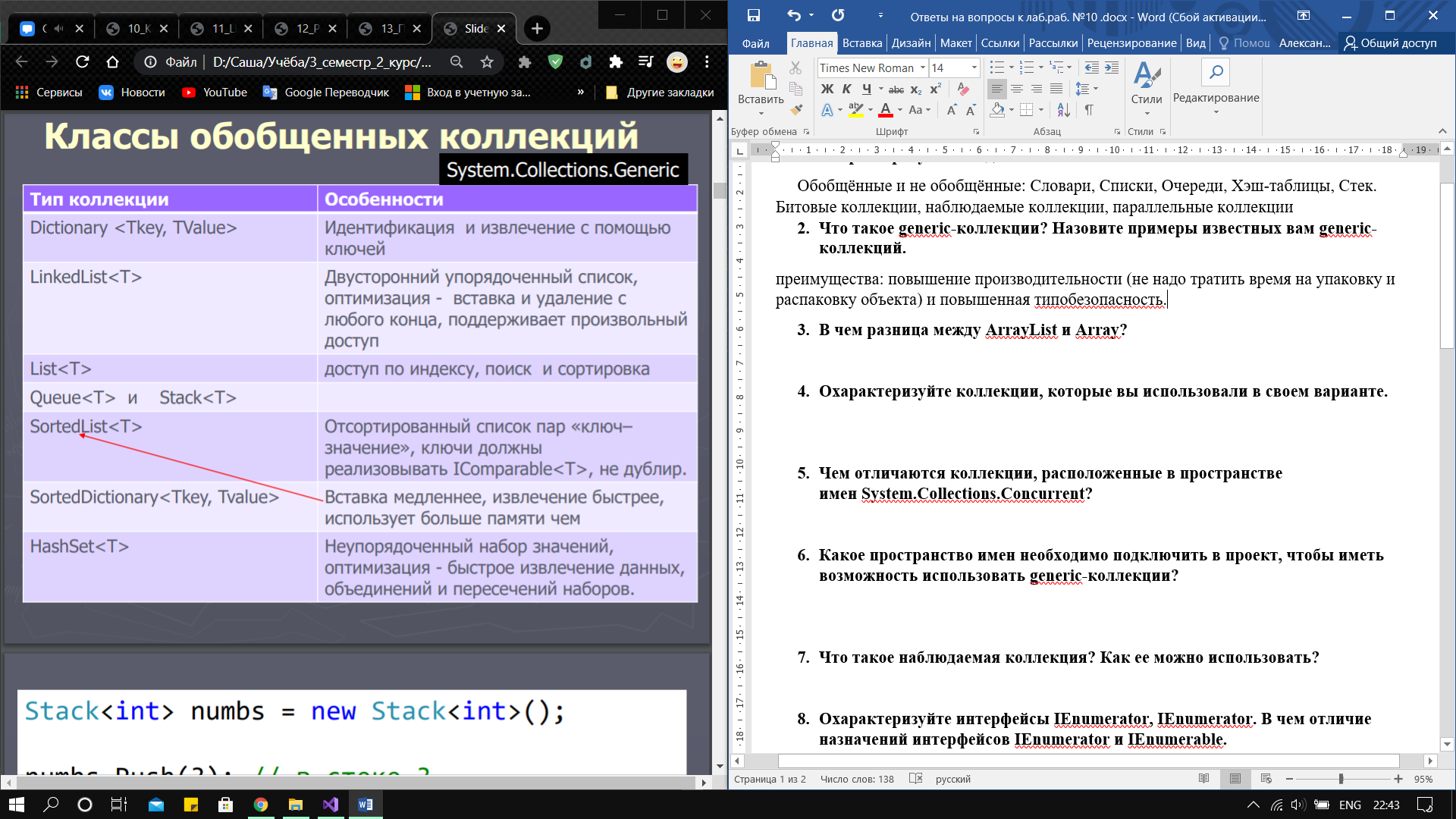
- многопоточный доступ к коллекции

- System.Collections.Concurrent

Обобщённые и необобщённые: Словари, Списки, Очереди, Хэш-таблицы, Стек.

1. **Что такое Generic-коллекции? Назовите примеры известных вам generic-коллекций.**

**Обобщённые коллекции** – это те же самые обобщённые классы (классы, которые изменяют своё поведение в зависимости от приписываемого им типа). Их использование перед необобщёнными коллекциями имеет те же преимущества: повышение производительности и повышенная типобезопасность.



1. **В чем разница между ArrayList и Array?**

Класс **ArrayList** определяется как массив переменной длины, который состоит из ссылок на объекты и может динамически увеличивать и уменьшать свой размер. Это список, каждый элемент связан друг с другом через указатели.

**Array** – это массив, представляющий из себя низкоуровневую структуру данных, которая по существу отображает область в памяти и имеет фиксированный размер.

1. **Охарактеризуйте коллекции, которые вы использовали в своем варианте.**

Еще один распространенный тип коллекции представляют словари. Словарь хранит объекты, которые представляют пару ключ-значение. Класс словаря **Dictionary<K, V>** типизируется двумя типами: параметр **K** представляет тип ключей, а параметр **V** предоставляет тип значений.

* **void Add(K key, V value)**: добавляет новый элемент в словарь
* **void Clear()**: очищает словарь
* **bool ContainsKey(K key)**: проверяет наличие элемента с определенным ключом и возвращает true при его наличии в словаре
* **bool ContainsValue(V value)**: проверяет наличие элемента с определенным значением и возвращает true при его наличии в словаре
* **bool Remove(K key)**: удаляет по ключу элемент из словаря

Другая версия этого метода позволяет получить удленный элемент в выходной параметр: **bool Remove(K key, out V value)**

* **bool TryGetValue(K key, out V value)**: получает из словаря элемент по ключу key. При успешном получении передает значение элемента в выходной параметр value и возвращает true
* **bool TryAdd(K key, V value)**: добавляет в словарь элемент с ключом key и значением value. При успешном добавлении возвращает true

Класс List<T> из пространства имен System.Collections.Generic представляет простейший список однотипных объектов. Класс List типизируется типом, объекты которого будут хранится в списке.

* **void Add(T item)**: добавление нового элемента в список
* **void AddRange(IEnumerable<T> collection)**: добавление в список коллекции или массива
* **int BinarySearch(T item)**: бинарный поиск элемента в списке. Если элемент найден, то метод возвращает индекс этого элемента в коллекции. При этом список должен быть отсортирован.
* **void CopyTo(T[] array)**: копирует список в массив array
* **void CopyTo(int index, T[] array, int arrayIndex, int count)**: копирует из списка начиная с индекса index элементы, количество которых равно count, и вставляет их в массив array начиная с индекса arrayIndex
* **bool Contains(T item)**: возвращает true, если элемент item есть в списке
* **void Clear()**: удаляет из списка все элементы

1. **Чем отличаются коллекции, расположенные в пространстве имен System.Collections.Concurrent?**

**Параллельные коллекции** - коллекции классов, предназначенные для безопасной работы в многопоточной среде, которыми можно воспользоваться при создании многопоточных приложений.

1. **Какое пространство имен необходимо подключить в проект, чтобы иметь возможность использовать generic-коллекции?**

System.Collections.Generic

1. **Что такое наблюдаемая коллекция? Как ее можно использовать?**

**ObservableCollection<T>**

В случае если нужна информация о том, когда элементы коллекции удаляются или добавляются, можно использовать класс ObservableCollection<T>. Этот класс предназначен для того, чтобы пользовательский интерфейс мог получать информацию об изменениях коллекции. Пространство имен этого класса — System.Collections.ObjectModel.

Класс ObservableCollection<T> унаследован от базового класса Collection<T>, который может применяться для создания специальных коллекций; он использует внутри себя List<T>. Методы базового класса SetItem() и RemoveItem() переопределены для инициации события CollectionChanged.

var data = new ObservableCollection<string>();

data.CollectionChanged += Data\_CollectionChanged;

data.Add("One");

data.Add("Two");

data.Insert(1, "Three");

data.Remove("One");

1. **Охарактеризуйте интерфейсы IEnumerator, IEnumerator. В чем отличие назначений интерфейсов IEnumerator и IEnumerable.**

Интерфейс ***IEnumerable*** имеет метод GetEnumerator(), возвращающий ссылку на другой интерфейс IEnumerator – перечислитель. Коллекция, которая реализует данный интерфейс, может использоваться с оператором foreach.

А интерфейс ***IEnumerator*** определяет функционал для перебора внутренних объектов в контейнере. Этот интерфейс может перечислять: имеет свойство Current и методы MoveNext() и Reset().

public interface IEnumerator

{

    bool MoveNext(); // перемещение на одну позицию вперед в контейнере элементов

    object Current {get;}  // текущий элемент в контейнере

    void Reset(); // перемещение в начало контейнера

}

1. **Поясните принцип работы коллекций:**
2. **LinkedList <T> -** Сохраняет элементы в двунаправленном списке.
3. **HashSet <T> -** Сохраняет ряд уникальных значений, используя хеш-таблицу.
4. **Dictionary <Tkey, TValue> -** Сохраняет пары "ключ-значение". Обеспечивает такие же функциональные возможности, как и необобщенный класс Hashtable.
5. **ConcurrentBag <Tkey, TValue> -** Этот класс не определяет никакого порядка для добавления или извлечения элементов. Он реализует концепцию отображения потоков на используемые внутренние массивы. Для доступа к элементам применяются методы Add(), TryPeek() и TryTake().
6. **Stack<t>, Queue<T> -** Создает стек; обеспечивает такие же функциональные возможности, как и необобщенный класс Stack.

- Создает очередь. Обеспечивает такие же функциональные возможности, как и необобщенный класс Queue.

1. **SortedList -** хранит наборы пар ключ-значение, отсортированные по ключу.