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

**Необобщенные коллекции** Реализуют ряд основных структур данных, включая динамический массив, стек, очередь, а также словари, в которых можно хранить пары "ключ-значение".

**(**позволяют хранить элементы без указания их типа.)

**Специальные коллекции** Оперируют данными конкретного типа или же делают это каким-то особым образом.

LinkedList StringCollection Например, имеются специальные коллекции для символьных строк, а также специальные коллекции, в которых используется однонаправленный список. ( рабаотаем с определ данными)

**Поразрядная коллекция** В прикладном интерфейсе Collections API определена одна коллекция с поразрядной организацией — это BitArray. Коллекция типа BitArray поддерживает поразрядные операции, т.е. операции над отдельными двоичными разрядами, например И, ИЛИ, исключающее ИЛИ.

**Обобщенные коллекции** Обеспечивают обобщенную реализацию нескольких стандартных структур данных, включая связные списки, стеки, очереди и словари. позволяя хранить элементы определенного типа, заданного при создании коллекции.

List<T>, Dictionary<TKey, TValue>, Queue<T>, Stack<T>.

**Параллельные коллекции** Поддерживают многопоточный доступ к коллекции. Это обобщенные коллекции, определенные в пространстве имен System.Collections.Concurrent.

2. Что такое generic-коллекции? Назовите примеры известных вам generic-коллекций.

Generic-коллекции (обобщенные коллекции) в программировании представляют собой контейнеры, которые позволяют хранить и управлять набором объектов определенного типа, независимо от конкретного типа данных.

**List<T>**: класс, представляющий последовательный список.

**LinkedList<T>**: класс двухсвязанного списка.

**SortedList<TKey, TValue>**: класс коллекции, хранящей наборы пар "ключ-значение", отсортированных по ключу.

**Dictionary<TKey, TValue>**: класс коллекции, хранящей наборы пар "ключ-значение".

**SortedDictionary<TKey, TValue>**: класс коллекции, хранящей наборы пар "ключ-значение", отсортированных по ключу.

**Queue<T>**: класс очереди объектов, работающей по алгоритму FIFO("первый вошел -первый вышел").

**SortedSet<T>**: класс отсортированной коллекции однотипных объектов.

**Stack<T>**: класс стека однотипных объектов.

3. В чем разница между ArrayList и Array?

Класс **Array** в языке программирования C# представляет структуру данных, которая позволяет хранить фиксированный набор элементов одного типа.

Класс **ArrayList** в языке программирования C# представляет динамическую коллекцию объектов переменного размера. Он позволяет хранить элементы различных типов, так как они обрабатываются как объекты.

4. Охарактеризуйте коллекции, которые вы использовали в своем варианте.

**Описание**: HashSet<T> — это неупорядоченная коллекция, которая хранит уникальные элементы. Использует хеширование для быстрого доступа к элементам.

**Описание**: LinkedList<T> — это двусвязный список, который позволяет эффективно добавлять и удалять элементы в любом месте списка.

**ObservableCollection<T>** - это обобщенная коллекция, которая представляет наблюдаемую коллекцию объектов, оповещающую об изменениях своего состояния.

5. Чем отличаются коллекции, расположенные в пространстве имен

System.Collections.Concurrent?

предназначены для использования в многопоточных приложениях. Они обеспечивают безопасный доступ из нескольких потоков и оптимизированы для работы в условиях конкурентного выполнения.

**ConcurrentBag<T>** Представляет потокобезопасную неупорядоченную коллекцию объектов.

**ConcurrentDictionary<TKey,TValue>** Представляет потокобезопасную коллекцию пар "ключ-значение", доступ к которой могут одновременно получать несколько потоков.

**ConcurrentQueue<T>** Предоставляет потокобезопасную коллекцию, обслуживаемую по принципу «первым поступил — первым обслужен» (FIFO).

**ConcurrentStack<T>** Предоставляет потокобезопасную коллекцию, обслуживаемую по принципу «последним поступил — первым обслужен» (LIFO).

**OrderablePartitioner<TSource>** Представляет определенный способ разделения упорядочиваемого источника данных на несколько разделов.

**Partitioner** Предоставляет общие стратегии создания разделов в массивах, списках и перечисляемых коллекциях.

**Partitioner<TSource>** Представляет определенный способ разделения источника данных на несколько разделов.

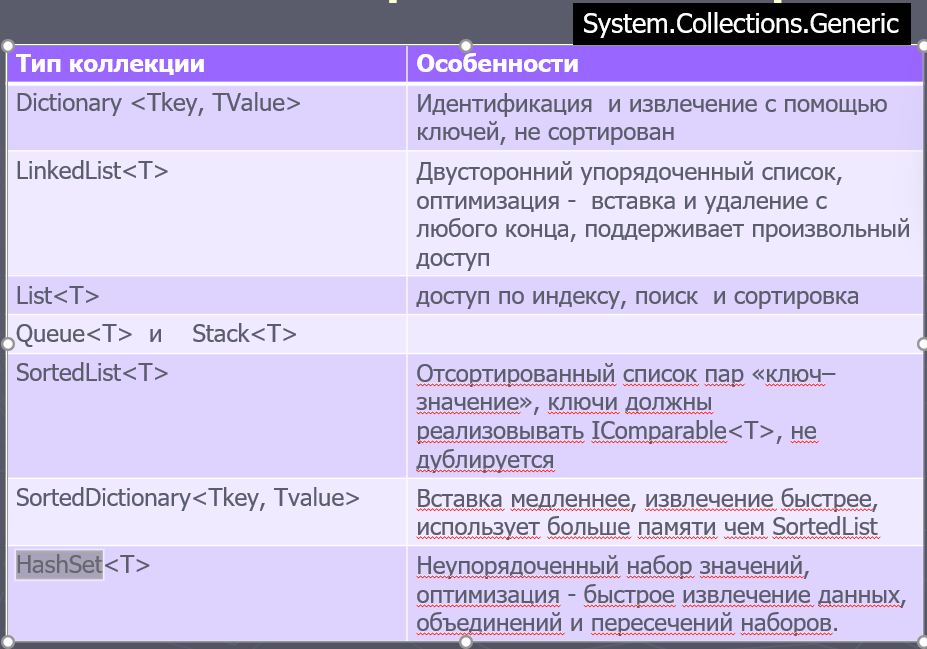
6. Какое пространство имен необходимо подключить в проект, чтобы иметь возможность использовать generic-коллекции?

 **Dictionary<TKey, TValue>**: Словарь, который хранит пары "ключ-значение" и обеспечивает быстрый доступ по ключу.

 **HashSet<T>**: Коллекция, которая хранит уникальные элементы без определенного порядка.

 **Queue<T>**: Очередь, которая поддерживает порядок FIFO (первый пришел — первый вышел).

 **Stack<T>**: Стек, который поддерживает порядок LIFO (последний пришел — первый вышел).



7. Что такое наблюдаемая коллекция? Как ее можно использовать?

Наблюдаемая коллекция (Observable Collection) - это коллекция, которая предоставляет механизм оповещения о изменениях своего состояния. Когда элементы добавляются, удаляются или изменяются в наблюдаемой коллекции, она автоматически генерирует события, чтобы другие объекты могли подписаться на эти события и реагировать на изменения.

* Подключить пространство имен System.Collections.ObjectModel.
* Создать экземпляр наблюдаемой коллекции, например, ObservableCollection<T>, где T - это тип элементов коллекции.
* Подписаться на события, которые генерирует наблюдаемая коллекция, чтобы реагировать на изменения. Обычно это делается путем добавления обработчиков событий CollectionChanged и PropertyChanged.
* Использовать методы и свойства наблюдаемой коллекции для добавления, удаления и изменения элементов.

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

IEnumerable<T>: определяет метод GetEnumerator, с помощью которого можно получать элементы любой коллекции. Реализация данного интерфейса позволяет перебирать элементы коллекции с помощью цикла foreach

IEnumerator<T>: определяет методы, с помощью которых потом можно получить содержимое коллекции по очереди

IEnumerable определяет метод GetEnumerator(), который возвращает объект типа IEnumerator для возможности итерации по коллекции.

IEnumerator позволяет перебирать элементы коллекции и предоставляет методы для перемещения по коллекции и доступа к текущему элементу.

**Основные отличия**

* **Назначение**:
  + IEnumerable используется для определения коллекции, которая может быть перечислена.
  + IEnumerator используется для перемещения по элементам коллекции и получения доступа к текущему элементу.
* **Методы**:
  + IEnumerable имеет метод GetEnumerator(), который возвращает итератор.
  + IEnumerator имеет методы MoveNext(), Reset() и свойство Current для управления итерацией.
* **Уровень абстракции**:
  + IEnumerable более высокоуровневый и определяет коллекцию как целое.
  + IEnumerator низкоуровневый и управляет состоянием итерации по элементам коллекции.

9. Поясните принцип работы коллекций:

a. LinkedList <T>

b. HashSet <T>

c. Dictionary <Tkey, TValue>

d. ConcurrentBag <Tkey, TValue>

e. Stack<t>, Queue<T>

f. SortedList, SortedList

LinkedList<T>: класс двухсвязанного списка.

HashSet <T>: представляет набор уникальных значений.

Dictionary<TKey, TValue>: класс коллекции, хранящей наборы пар "ключ-значение".

ConcurrentBag <Tkey, TValue>: Представляет потокобезопасную неупорядоченную коллекцию объектов.

Queue<T>: класс очереди объектов, работающей по алгоритму FIFO("первый вошел -первый вышел").

Stack<T>: класс стека однотипных объектов.

SortedList<TKey, TValue>: класс коллекции, хранящей наборы пар "ключ-значение", отсортированных по ключу.

SortedList<TKey, TValue> реализован в виде массива, а SortedDictionary<TKey, TValue> использует внутреннее двоичное дерево поиска.