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

1. Необобщенные коллекции

Оперируют данными типа object.

Могут хранить данные любого типа в одной коллекции.

Не типизированы, так как хранят ссылки на данные типа object.

Примеры таких коллекций находятся в пространстве имен System.Collections.

2. Специальные коллекции

Оперируют данными конкретного типа или предоставляют специальные возможности.

Например, есть специальные коллекции для строк и однонаправленных списков.

Объявляются в пространстве имен System.Collections.Specialized.

3. Поразрядная коллекция (BitArray):

Предоставляет поразрядные операции (И, ИЛИ, исключающее ИЛИ) и отличается от других коллекций.

Находится в пространстве имен System.Collections.

4. Обобщенные коллекции (Generic Collections):

Предоставляют обобщенную реализацию структур данных, таких как связные списки, стеки, очереди и словари.

Типизированы, и в них могут храниться только элементы, совместимые по типу с данной коллекцией.

Объявляются в пространстве имен System.Collections.Generic.

5. Параллельные коллекции (Concurrent Collections):

Поддерживают многопоточный доступ к коллекции.

Обобщенные коллекции, определенные в пространстве имен System.Collections.Concurrent.

6.Коллекции в пространстве имен System.Collections.ObjectModel:

Содержат классы, поддерживающие создание пользовательских обобщенных коллекций.

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

Generic-коллекции это обобщённые коллекции.

* List<T>: Представляет динамический массив, который можно легко изменять.
* Dictionary<TKey, TValue>: Представляет коллекцию ключ/значение.
* Queue<T>: Реализует очередь для элементов типа T.
* Stack<T>: Реализует стек для элементов типа T.
* HashSet<T>: Представляет набор уникальных элементов.
* LinkedList<T>: Представляет двусвязный список.
* ObservableCollection<T>: Представляет коллекцию, которая предоставляет уведомления об изменениях.
* SortedSet<T>: Представляет набор уникальных элементов, отсортированных по порядку.

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

**Array** является базовым массивом в C# и имеет фиксированный размер при создании. Элементы массива должны иметь одинаковый тип данных. Размер массива фиксирован и не может быть изменен после создания.

**ArrayList** является динамическим массивом объектов, который может изменять свой размер во время выполнения. ArrayList может хранить элементы разных типов (т.е., объекты произвольных классов). Размер ArrayList динамически увеличивается при добавлении элементов и уменьшается при удалении.

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

1. List<T>: Представляет динамический массив, который можно легко изменять.
2. Queue<T>: Реализует очередь для элементов типа T.
3. ConcurrentDictionary<TKey, TValue>: Безопасный для потоков словарь, который поддерживает параллельное чтение и запись.

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

Пространство имен System.Collections.Concurrent в C# предоставляет коллекции, предназначенные для поддержки параллельного программирования и работы в многопоточной среде. Они обеспечивают безопасное изменение данных из разных потоков и предоставляют высокоэффективные алгоритмы для таких сценариев.

* ConcurrentBag<T>: Позволяет параллельно добавлять и извлекать элементы из коллекции. Это беспорядочная коллекция, и каждый поток имеет свой стек элементов.
* ConcurrentQueue<T>: Представляет безопасную для потоков FIFO (First-In-First-Out) очередь.
* ConcurrentStack<T>: Представляет безопасный для потоков стек, где элементы добавляются и удаляются с вершины стека.
* BlockingCollection<T>: Обеспечивает блокировку в случае пустой или полной коллекции. Предназначен для использования в сценариях производителя-потребителя.
* ConcurrentDictionary<TKey, TValue>: Безопасный для потоков словарь, который поддерживает параллельное чтение и запись.
* Partitioner: Предоставляет методы для упрощения итерации и обработки данных в параллельных циклах.
* OrderablePartitioner: Расширяет Partitioner для обеспечения возможности управления порядком обработки элементов.

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

System.Collections.Generic

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

Наблюдаемая коллекция (Observable Collection) - это коллекция в языке программирования, которая предоставляет механизм уведомлений об изменениях в своей структуре или содержимом. В C# наблюдаемая коллекция обычно реализуется с использованием класса ObservableCollection<T>, который предоставляет событие CollectionChanged. Это событие возникает при любом изменении коллекции.

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

**IEnumerator:**

Назначение: предоставляет методы для перебора элементов в коллекции.

Члены:

MoveNext(): Перемещает указатель на следующий элемент коллекции.

Reset(): Сбрасывает указатель перед первым элементом коллекции.

Current: Получает текущий элемент коллекции (свойство).

**IEnumerable:**

Назначение: определяет метод GetEnumerator(), который возвращает объект, реализующий интерфейс IEnumerator. Этот интерфейс позволяет объекту быть перечисляемым.

Члены: GetEnumerator(): Возвращает объект, реализующий IEnumerator.

**Отличие назначений интерфейсов:**

**IEnumerable**: используется для того, чтобы объект можно было использовать в операторе foreach для перебора элементов.

**IEnumerator**: необходим для реализации пользовательского перечислителя, который будет использоваться для перебора элементов коллекции.

***9. Поясните принцип работы коллекций:***

* **LinkedList** (Двусвязный список): Каждый элемент связан с предыдущим и следующим элементами. Это обеспечивает эффективную вставку и удаление элементов в середине списка.
* **HashSet** (Множество): Хранит уникальные элементы без упорядочения. Использует хэш-таблицу для быстрого поиска элементов. Операции добавления, удаления и поиска выполняются за постоянное время в среднем случае.
* **Dictionary** (Словарь): Представляет коллекцию пар "ключ-значение". Операции поиска, добавления и удаления выполняются по ключу. Также использует хэш-таблицу для обеспечения эффективности.
* **ConcurrentBag** (Параллельный мешок):: Коллекция, предназначенная для параллельного программирования. Позволяет безопасно добавлять и извлекать элементы из разных потоков. Не гарантирует порядок элементов при извлечении.
* **Stack и Queue** (Стек и Очередь): Стек - это коллекция, где элементы добавляются и извлекаются с конца (последним пришел, первым ушел). Очередь - это коллекция, где элементы добавляются в конец, а извлекаются из начала (первым пришел, первым ушел).
* **SortedList и SortedDictionary** (Отсортированный список и отсортированный словарь): Эти коллекции хранят элементы в отсортированном порядке. SortedList представляет коллекцию пар "ключ-значение" и поддерживает доступ по индексу. SortedDictionary также представляет коллекцию пар "ключ-значение", но не поддерживает доступ по индексу.