1. Перечислите стандартные коллекции NET Framework.

В среде .NET Framework поддерживаются пять типов коллекций:

Необобщенные, обобщенные, параллельные, специальные и с поразрядной организацией.

2. Поясните принцип работы коллекции:

a) Stack<T>

Создает стек. Обеспечивает такие же функциональные возможности, как и

необобщенный класс Stack

b) Queue<T>

Создает очередь. Обеспечивает такие же функциональные возможности, как и

необобщенный класс Queue

c) HashSet<T>

Сохраняет ряд уникальных значений, используя хеш таблицу

d) List<T>

Создает динамический массив. Обеспечивает такие же функциональные

возможности, как и необобщенный класс ArrayList

e) Dictionary<Tkey, TValue>

Сохраняет пары "ключ-значение". Обеспечивает такие же функциональные

возможности, как и необобщенный класс Hashtable

f) LinkedList<T>

Сохраняет элементы в двунаправленном списке

g) SortedDictionary<TKey, TValue>

Создает отсортированный список из пар "ключ-значение"

h) SortedList<TKey, TValue>

Создает отсортированный список из пар "ключ-значение". Обеспечивает такие

же функциональные возможности, как и необобщенный класс SortedList

i) SortedSet<T>

Создает отсортированное множество

3. Охарактеризуйте необобщенные, специальные, с поразрядной организацией, обобщенные и параллельные коллекции.

*Необобщенные коллекции*

Реализуют ряд основных структур данных, включая динамический массив, стек, очередь, а также словари, в которых можно хранить пары "ключ-значение". Необобщенные коллекции оперируют данными типа object., а значит они могут служить для хранения данных любого типа, причем в одной коллекции допускается наличие разнотипных данных. Такие коллекции не типизированы, поскольку в них хранятся ссылки на данные типа object. Классы и интерфейсы необобщенных коллекций находятся в пространстве имен **System.Collections**.

*Специальные коллекции*

Оперируют данными конкретного типа или же делают это каким-то особым образом. Например, имеются специальные коллекции для символьных строк, а также специальные коллекции, в которых используется однонаправленный список. Специальные коллекции объявляются в пространстве

имен **System.Collections.Specialized**.

*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, а следовательно, она существенно отличается своими возможностями от остальных типов коллекций. Коллекция типа BitArray объявляется в пространстве имен System.Collections.

*Обобщенные коллекции*

Обеспечивают обобщенную реализацию нескольких стандартных структур данных, включая связные списки, стеки, очереди и словари. Эти коллекции - типизированы в силу их обобщенного характера, значит, что в ней могут храниться только такие элементы данных, которые совместимы по типу с данной коллекцией. Благодаря этому исключается случайное несовпадение типов. Они объявляются в пространстве имен **System.Collections.Generic**.

*Параллельные коллекции*

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

В пространстве имен System.Collections.ObjectModel находится также ряд классов, поддерживающих создание пользователями собственных обобщенных коллекций.

4. Какие интерфейсы используются в коллекциях C#?

**ICollection<T>** Определяет основополагающие свойства обобщенных коллекций

**IComparer<T>** Определяет обобщенный метод Compare() для сравнения объектов, хранящихся

в коллекции

**IDictionary<Tkey, TValue>** Определяет обобщенную коллекцию, состоящую из пар "ключ-значение"

**IEnumerable<T>** Определяет обобщенный метод GetEnumerator(), предоставляющий перечислитель для любого класса коллекции

**Enumerator<T>** Предоставляет методы, позволяющие получать содержимое коллекции по

очереди

**IEqualityComparer<T>** Сравнивает два объекта на предмет равенства

**IList<T>** Определяет обобщенную коллекцию, доступ к которой можно получить с

помощью индексатора

5. Для чего используется интерфейс IComparable?

Определяет обобщённый метод сравнения для типа, который реализуется типом значения или классом, чтобы упорядочить или отсортировать его экземпляры.

Содержит метод CompareTo(Object) сравнивает текущий экземпляр с другим объектом того же типа и возвращает целое число, которое показывает, расположен ли текущий экземпляр перед, после или на той же позиции в порядке сортировки что и другой объект.

6. Что содержит интерфейс IEnumerator или обобщенный интерфейс IEnumerator<T>? Где и как его можно использовать?

Определяет обобщенный метод GetEnumerator(), предоставляющий перечислитель для любого класса коллекции.

7. Что такое наблюдаемая коллекция? Где и каким образом ее можно использовать?

**ObservableCollection<T>**

* пользовательский интерфейс получает информацию об изменениях коллекции
* унаследован от Collection<T>, использует внутри себя List<T>