**Необобщенные коллекции**

Плюсы:

- позволяет манипулировать объектами с разными типами данных (например, это нужно, когда мы заранее не знаем какой тип данных попадет в коллекцию)

Минусы:

- нарушает типовую безопасность

**Обобщенные коллекции**

**Класс List<T>**

- в классе реализуется обобщенный динамический массив

- ничем не отличается от класса необобщенной коллекции ArrayList

- обеспечивает типовую безопасность, за счет работы с конкретным типом данных

- емкость массива может увеличиваться с автоматически

**Класс HashSet<T> и SortedSet<T>**

- содержат в себе только уникальные элементы

- прирост производительности на больших коллекциях, поскольку некоторые операции выполняются быстрее (например, Contain, Add, Remove выполняются за О(1))

**Класс ObservableCollection<T>**

- по функциональности похож на класс List<T> за исключением того что сообщает внешним объектам об изменении в коллекции

**Класс LinkedList<T>**

- представляет двух связный список, в котором каждый элемент содержит ссылку одновременно на следующий и предыдущий элемент

- с помощью методов класса, можно обращаться к различным элементам, как в конце, так и в начале списка

**Класс Dictionary<TKey, TValue>**

- позволяет хранить в себе пары «ключ-значение» в коллекции, как в словаре

- значения доступны в словаре по соответствующим ключам

**Класс Queue<T >**

- представляет собой обычную очередь, работающую по алгоритму FIFO (первый вошел – первый вышел), как обычная очередь в магазин.

**Класс Stack<T >**

- представляет коллекцию, работающую по алгоритму LIFO (последний вошел – первый вышел)

- при такой организации следующий элемент записывается поверх предыдущего

- извлечение из коллекции происходит в обратном порядке: извлекается тот элемент который находится выше всех в стеке.