	Основная функциональность	Примеры типичного использования
Set	Неупорядоченный набор неповторяющихся элементов. Расширяет интерфейс Collection. Если добавить в набор элемент, который уже находится в нем, то он будет не добавлен.	HashSet – хранит элементы в хеш - таблице => Имеет наиболее высокую производительность, но не гарантирует порядок элементов. LinkedHashSet – отличается от HashSet тем, что хранит элементы в порядке их вставки. Немного медленнее HashSet. TreeSet – хранит элементы в отсортированном порядке. Работает значительно медленнее HashSet'а.
List	Служит для работы с упорядоченными коллекциями. К каждому элементу можно обратиться по индексу. Расширяет интерфейс Collection.	ArrayList – наиболее широко используем. Обладает наибольшей производительностью в плане доступа к случайному элементу в массиве. LinkedList – В отличие от ArrayList, обладает большей скоростью вставки элемента в произвольное место в списке, однако доступ к элементу прямо пропорциональный его позиции относительно начала последовательности.
Queue	Очередь. Представляет структуру данных, работающую по принципу FIFO. Есть и реализации LIFO.	PriorityQueue – FIFO. Прямая реализация интерфейса. Поддерживает возможность управления порядком элементов с помощью компаратора. LinkedQueue – FIFO. Элементы имеют ссылки друг на друга. ArrayDeque – LIFO. Двунаправленная очередь, реализующая интерфейс Deque, расширяющий интерфейс Queue.
Map	Предназначен для работы со словарями, в которых есть ключи и соответствующие им значения.	HashMap – хранит ключе в хеш-таблице => Имеет наиболее высокую производительность, но не гарантирует порядок. Ключи и Значения могут быть null. LinkedHashMap – хранит ключи в порядке их вставки в Map. Чуть-чуть медленнее HashMap'а. Ключи и значения могут быть null. TreeMap – хранит ключи в отсортированном порядке => работает медленнее HashMap'а. Значения могут быть null.