

	Ordering	Random Access	Key-Value Pairs	Allows Duplicates	Allows Null Values	Thread Safe	Blocking Operations	Upper Bounds	Usage Scenarios	
Most Commonly Known Collections										
ArrayList	YES	YES	NO	YES	YES	NO	NO	NO	<ul style="list-style-type: none"> * Default choice of List implementation * To store a bunch of things * Repetitions matters * Insertion order matters * Best implementation in case of huge lists which are read intensive (elements are accessed more frequently than inserted deleted) 	
HashMap	NO	YES	YES	NO	YES	NO	NO	NO	<ul style="list-style-type: none"> * Default choice of Map implementation * Majorly used for simple in-memory caching purpose. 	
Vector	YES	YES	NO	YES	YES	YES	NO	NO	<ul style="list-style-type: none"> * Historical implementation of List * A good choice for thread-safe implementation 	
Hashtable	NO	YES	YES	NO	NO	YES	NO	NO	<ul style="list-style-type: none"> * Similar to HashMap * Do not allow null values or keys * Entire map is locked for thread safety 	
Most Talked About Collections										
HashSet	NO	YES	NO	NO	YES	NO	NO	NO	<ul style="list-style-type: none"> * To store bunch of things * A very nice alternative for ArrayList if ** Do not want repetitions ** Ordering does not matter 	
TreeSet	YES	YES	NO	NO	NO	NO	NO	NO	<ul style="list-style-type: none"> * To store bunch of things in sorted order * A very nice alternative for ArrayList if ** Do not want repetitions ** Sorted order 	
LinkedList	YES	NO	NO	YES	YES	NO	NO	NO	<ul style="list-style-type: none"> * Sequential Access * Faster adding and deleting of elements * Slightly more memory than ArrayList * Add/Remove elements from both ends of the queue * Best alternative in case of huge lists which are more write intensive (elements added / deleted are more frequent than reading elements) 	
ArrayDeque	YES	YES	NO	YES	NO	NO	NO	NO	<ul style="list-style-type: none"> * Random Access * Faster searching and retrieval of elements * Add/Remove elements from both ends of the queue * Best alternative in case of huge lists which are more read intensive 	
Stack	YES	NO	NO	YES	YES	YES	NO	NO	<ul style="list-style-type: none"> * Similar to a Vector * Last-In-First-Out implementation 	
TreeMap	YES	YES	YES	NO	NO	NO	NO	NO	<ul style="list-style-type: none"> * A very nice alternative for HashMap if sorted keys are important 	
Special Purpose Collections										
WeakHashMap	NO	YES	YES	NO	YES	NO	NO	NO	<ul style="list-style-type: none"> * The keys that are not referenced will automatically become eligible for garbage collection * Usually used for advanced caching techniques to store huge data and want to conserve memory 	
Arrays	YES	YES	NO	YES	YES	NO	NO	YES	<ul style="list-style-type: none"> * A Utility class provided to manipulate arrays ** Searching ** Sorting ** Converting to other Collection types such as a List 	
Properties	NO	YES	YES	NO	NO	YES	NO	NO	<ul style="list-style-type: none"> * Properties are exactly same as the Hashtable * Keys and Values are String * Can be loaded from an input stream * Usually used to store application properties and configurations 	
Thread Safe Collections										
CopyOnWriteArrayList	YES	YES	NO	YES	YES	YES	NO	NO	<ul style="list-style-type: none"> * A thread safe variant of ArrayList * Best use for ** Small lists which are read intensive ** requires thread-safety 	
ConcurrentHashMap	NO	YES	YES	NO	NO	YES	NO	NO	<ul style="list-style-type: none"> * A thread safe variant of Hashtable * Best use for ** requires thread-safety ** Better performance at high load due to a better locking mechanism 	
ConcurrentSkipListMap	YES	YES	YES	NO	NO	YES	NO	NO	<ul style="list-style-type: none"> * A thread safe variant of TreeMap * Best use for ** requires thread-safety 	
ConcurrentSkipListSet	YES	NO	NO	NO	NO	YES	NO	NO	<ul style="list-style-type: none"> * A thread safe variant of TreeSet * Best use for ** Do not want repetitions ** Sorted order ** Requires thread-safety 	
CopyOnWriteArraySet	YES	YES	NO	NO	YES	YES	NO	NO	<ul style="list-style-type: none"> * A thread-safe implementation of a Set * Best use for ** Small lists which are read intensive ** requires thread-safety ** Do not want repetitions 	
ConcurrentLinkedQueue	YES	NO	NO	YES	NO	YES	NO	NO	<ul style="list-style-type: none"> * A thread-safe variant of PriorityQueue * Best use for ** Small lists ** No random access ** requires thread-safety 	
ConcurrentLinkedDeque	YES	NO	NO	YES	NO	YES	NO	NO	<ul style="list-style-type: none"> * A thread-safe variant of LinkedList * Best use for ** Small lists ** No random access ** Insertions, retrieval on both sides of the queue ** requires thread-safety" 	
Blocking Collections										
ArrayBlockingQueue	YES	NO	NO	YES	NO	YES	YES	YES	<ul style="list-style-type: none"> * Best use for Producer - Consumer type of scenarios with ** Lower capacity bound ** Predictable capacity * Has a bounded buffer. Space would be allocated during object creation 	
LinkedBlockingQueue	YES	NO	NO	YES	NO	YES	YES	YES	<ul style="list-style-type: none"> * Best use for Producer - Consumer type of scenarios with ** Large capacity bound ** Unpredictable capacity * Upper bound is optional 	
LinkedTransferQueue	YES	NO	NO	YES	NO	YES	YES	YES	<ul style="list-style-type: none"> * Can be used in situations where the producers should wait for consumer to receive elements. e.g. Message Passing 	
PriorityBlockingQueue	YES	NO	NO	YES	NO	YES	YES	NO	<ul style="list-style-type: none"> ** Best use for Producer - Consumer type of scenarios with ** Large capacity bound ** Unpredictable capacity ** Consumer needs elements in sorted order 	
LinkedBlockingDeque	YES	NO	NO	YES	NO	YES	YES	YES	<ul style="list-style-type: none"> * A Deque implementation of LinkedBlockingQueue ** Can add elements at both head and tail 	
SynchronousQueue	YES	NO	NO	YES	NO	YES	YES	NO	<ul style="list-style-type: none"> * Both producer and consumer threads will have to wait for a handoff to occur. * If there is no consumer waiting. The element is not added to the collection. 	
DelayQueue	YES	NO	NO	YES	NO	YES	YES	NO	<ul style="list-style-type: none"> * Similar to a normal LinkedBlockingQueue * Elements are implementations of Delayed interface * Consumer will be able to get the element only when it's delay has expired 	
						Source:	http://www.janeve.me/articles/which-java-collection-to-use			