The Java API includes the **List** interface. The List interface is implemented by the **ArrayList** and **Vector**, therefore they both include **isEmpty**, **get**, **indexOf**, **remove** and **size** methods. The remove method is overloaded to accept either an **int** (index) as a parameter or an **Object**, therefore the one that takes the int works like our own API. The Vector predates the List interface, so it also has its own methods for **removeElementAt** (that takes and int) and **removeElement** (that takes an Object).

Note: In Java...

A **Set** is a **Collection** with **no duplicates**.
A **List** is an **ordered** collection that may **allow duplicates**.

Java provides both the List and Set interfaces (as well as others for other categories of Collections).

Author's **ArrayListClass**	Java's **ArrayList**	Java's **Vector** (List method/Classic method)
1. b isEmpty()	boolean isEmpty()	boolean isEmpty()
2. D retrieveAt(i)	E get(int)	E get(int) / E elementAt(int)
3. i seqSearch(D)	int indexOf(Object)	int indexOf(Object)
4. v removeAt(i)	E remove(int)	E remove(int) / void removeElementAt(i)
5. i listSize()	int size()	i size()

Note: Vector has lastIndexOf but does NOT work like our own API's near equivalent.
Note: The List interface does not provide a remove that returns void.

Types noted above:

b	boolean
D	DataElement (textbook)
i	int
v	void
E	<E> (List interface is implemented with **Generics** in place of **Object**)

While we have been building our own data wrapper-classes as subclasses of our own DataElement class, the preferred way to do this is with Generics. Generics are surveyed in the Advanced Java course and there are chapters devoted to Generics and Collections in the Introductory Java textbook.