

## Java - Data Structures

The data structures provided by the Java utility package are very powerful and perform a wide range of functions. These data structures consist of the following interface and classes –

- Enumeration
- BitSet
- Vector
- Stack
- Dictionary
- Hashtable
- Properties

All these classes are now legacy and Java-2 has introduced a new framework called Collections Framework, which is discussed in the next chapter. –

### The Enumeration

The Enumeration interface isn't itself a data structure, but it is very important within the context of other data structures. The Enumeration interface defines a means to retrieve successive elements from a data structure.

For example, Enumeration defines a method called `nextElement` that is used to get the next element in a data structure that contains multiple elements.

To have more detail about this interface, check [The Enumeration](#) .

### The BitSet

The BitSet class implements a group of bits or flags that can be set and cleared individually.

This class is very useful in cases where you need to keep up with a set of Boolean values; you just assign a bit to each value and set or clear it as appropriate.

For more details about this class, check [The BitSet](#) .

### The Vector

The Vector class is similar to a traditional Java array, except that it can grow as necessary to accommodate new elements.

Like an array, elements of a Vector object can be accessed via an index into the vector.

The nice thing about using the Vector class is that you don't have to worry about setting it to a specific size upon creation; it shrinks and grows automatically when necessary.

For more details about this class, check [The Vector](#) .

## The Stack

The Stack class implements a last-in-first-out (LIFO) stack of elements.

You can think of a stack literally as a vertical stack of objects; when you add a new element, it gets stacked on top of the others.

When you pull an element off the stack, it comes off the top. In other words, the last element you added to the stack is the first one to come back off.

For more details about this class, check [The Stack](#) .

## The Dictionary

The Dictionary class is an abstract class that defines a data structure for mapping keys to values.

This is useful in cases where you want to be able to access data via a particular key rather than an integer index.

Since the Dictionary class is abstract, it provides only the framework for a key-mapped data structure rather than a specific implementation.

For more details about this class, check [The Dictionary](#) .

## The Hashtable

The Hashtable class provides a means of organizing data based on some user-defined key structure.

For example, in an address list hash table you could store and sort data based on a key such as ZIP code rather than on a person's name.

The specific meaning of keys with regard to hash tables is totally dependent on the usage of the hash table and the data it contains.

For more detail about this class, check [The Hashtable](#) .

## The Properties

Properties is a subclass of Hashtable. It is used to maintain lists of values in which the key is a String and the value is also a String.

The Properties class is used by many other Java classes. For example, it is the type of object returned by `System.getProperties()` when obtaining environmental values.

For more detail about this class, check [The Properties](#) .