List

The word *list* is used in several different but related ways in computer science. We describe them here.

General meaning of list

The most general meaning of *list* is a bunch of values given in a particular order, with duplicates allowed. Here is a list of five coins in a pile (nickel, penny, penny, quarter, penny). Here is a list of the coin values: (5, 1, 1, 25, 1).

There are many ways of writing a list. We have used a standard way: "(" and ")" delimit the list and ", " separates items of the list.

As another example, consider the alphabetical list of major automakers in the U.S. as of July 2018 (taken from Wikipedia.org):

Fiat Chrysler Automobiles
Ford
General Motors
Tesla

A finite list is often called a *sequence*. If the list is potentially infinite in length, it could be called a *stream*.

Numbering items in (or elements of) a list

Generally, the first item in a list is number 0, the second number 1, etc. That may seem strange, but most programming languages use this numbering scheme. So, for the list b = (5, 1, 1, 25, 1), we could write: b[0] is 5, b[1] is 1, b[2] is 1, b[3] is 25, and b[4] is 1.

Data structures

The word *list* is often used in the names of several data structures, such as the singly linked list, doubly linked list, and circular linked list.

Abstract data type or ADT (look up in JavaHyperText)

An abstract data type is basically the definition of a set of values together with operations on them. We might define the List ADT as a list of values as shown above together with operations for:

Creating an empty list,
Returning the length or size of the list (i.e. number of items in it),
Prepending an item to the list,
Appending an item to the list
Returning an item of the list (e.g. get(b, i) to return item number i of list b).

The Java interface List

Java provides an interface `List`, in package `java.util`, which actually defines an ADT. It defines almost 30 operations on lists, including those listed above. Here is the definition of `List` for version 8 of Java:

https://docs.oracle.com/javase/8/docs/api/java/util/List.html

Since `List` is an interface, objects of it cannot be created. Instead, one constructs objects of a class that implements the interface. Execution of the following assignment statement constructs an object of class `ArrayList` and stores (a pointer to) it in a variable of type `List`. `List ll` initially contains no elements.

```
List<Integer> ll= new ArrayList<>();
```

Storing the object in a variable of type `List` instead of `ArrayList` has an advantage. It restricts use of `ll` to only the operations available in ADT `List`.