Class ArrayList

An instance of class java.util.ArrayList contains a list of objects. It may be a list of the courses you have taken, a list of high temperatures for each day in Ithaca in 2006, or a list of objects of totally different classes. There are methods for adding elements to the list, deleting elements, searching for an element, and so on. We illustrate these now, using DrJava as well as the powerpoint window so that you can see what is happening.

Creating an ArrayList and appending elements to it

First, let's import class `java.util.ArrayList` and then create a new `ArrayList` —storing it in `b`. By calling function `size` of the new `ArrayList` object, you can see that it is initially empty.

Let's add a `String` object. As you can see, `add` is a function, which always returns **`true`**. The reason for this is beyond the scope of this course. Of course, you can always put a semicolon after the call to `add` and make a statement out of it, as we do here. If you give function `add` an integer as argument, autoboxing will automatically create a new object of class `Integer` that wraps the integer, and this `Integer` object will be added to the list. By calling function `size`, we see that the size of the list is now 3.

Function toString

The description returned by function `toString` is a list of the objects, separated by commas and delimited by square brackets. Or, we can abbreviate and simply use b, and b.toString will be called. Use this function whenever you want to see what is in an `ArrayList`.

Inserting elements at a specified position

As you can see, the objects in the list are numbered 0, 1, 2, and so on. You can insert an object at any position you want, using the two-parameter function `add`. To show this, let's add an integer 7 at position 1 —note how elements 1 and 2 are shifted to positions 2 and 3. Let's call function `toString` to make sure that this works as we said —yes, it does.

Inserting an object anywhere but at the end takes more time than appending an object to the list, because a number of elements have to be shifted down. The time used is proportional to the number of elements shifted.

Retrieving objects from an ArrayList

Function `get` retrieves the name of the object at the specified position. To show this, let's get the first element, which should be the string "abc".

The elements of the `ArrayList` have apparent class `Object`, and if we assign it to a `String` variable, it will be cast appropriately. If we try to assign it to some other variable, say of class `Integer`, we get a class cast exception. So be careful when retrieving elements from an `ArrayList`.

Changing the element that is in a list

As you can see, list `b` contains four elements: the string "abc" and three wrapped integers. You can change the element at position 2 to the wrapped boolean value **`true`** by calling function `set(2, true)`. As an added benefit, this function call returns the element that was previously at position 2. Let's print `b` again so that you can see that the change actually happened.

Removing an element

Finally, use function call `b.remove(1)` to remove the object at position 1 from the list and return it. This call requires elements numbered 2, 3, …, to be shifted up to positions 1, 2, …, so it takes time proportional to the number of elements moved. Let's make sure it was removed by typing b in the interactions pane.

Summary

The purpose of this lecture was to show you how a list of objects could be maintained in an instance of class `ArrayList`. We have showed you the basic methods. There are others —for example, for searching the list for a value. Peruse the API spec of class `ArrayList` to see what other methods are available.

We have not, and will not, shown applications that use `ArrayList`.

Class ArrayList