This reference is for Processing 3.0+. If you have a previous version, use the reference included with your software in the Help menu. If you see any errors or have suggestions, please let us know. If you prefer a more technical reference, visit the Processing Core Javadoc and Libraries Javadoc.

Name

ArrayList

Examples

```
// These are code fragments that show how to use an ArrayList.
// They won't compile because they assume the existence of a Particle class.

// Declaring the ArrayList, note the use of the syntax "<Particle>" to indicate
// our intention to fill this ArrayList with Particle objects
ArrayList<Particle> particles = new ArrayList<Particle>();

// Objects can be added to an ArrayList with add()
particles.add(new Particle());

// Particles can be pulled out of an ArrayList with get()
Particle part = particles.get(0);
part.display();

// The size() method returns the current number of items in the list
int total = particles.size();
println("The total number of particles is: " + total);

// You can iterate over an ArrayList in two ways.
// The first is by counting through the elements:
for (int i = 0; i < particles.size(); i++) {
  Particle part = particles.get(i);
  part.display();
}

// The second is using an enhanced loop:
for (Particle part : particles) {
  part.display();
}

// You can delete particles from an ArrayList with remove()
particles.remove(0);
println(particles.size()); // Now one less!

// If you are modifying an ArrayList during the loop,
// then you cannot use the enhanced loop syntax.
// In addition, when deleting in order to hit all elements,
// you should loop through it backwards, as shown here:
for (int i = particles.size() - 1; i >= 0; i--) {
  Particle part = particles.get(i);
  if (part.finished()) {
    particles.remove(i);
  }
}
```

Description

An `ArrayList` stores a variable number of objects. This is similar to making an array of objects, but with an `ArrayList`, items can be easily added and removed from the ArrayList and it is resized dynamically. This can be very convenient, but it's slower than making an array of objects when using many elements. Note that for resizable lists of integers, floats, and Strings, you can use the Processing classes IntList, FloatList, and StringList.

An ArrayList is a resizable-array implementation of the Java List interface. It has many methods used to control and search its contents. For example, the length of the `ArrayList` is returned by its `size()` method, which is an integer value for the total number of elements in the list. An element is added to an `ArrayList` with the `add()` method and is deleted with the `remove()` method. The `get()` method returns the element at the specified position in the list. (See the above example for context.)

For a list of the numerous `ArrayList` features, please read the Java reference description.

Constructor

```
ArrayList<Type>()
ArrayList<Type>(initialCapacity)
```

Parameters

`Type`	Class Name: the data type for the objects to be placed in the ArrayList.
`initialCapacity`	int: defines the initial capacity of the list; it's empty by default

Related

`IntList`
`FloatList`
`StringList`

Updated on April 30, 2017 02:33:23pm EDT