# **Using the Foreach Loop**

This guide details the foreach loop, its benefits, when and how to use it.

## ***What is a foreach loop?***

Java, before version 5.0, had only three kinds of loops: while, do while, and for. Version 5.0 added the foreach loop – a loop that has no index, no loop condition, and no increment. How can a loop like that be useful? Simple: the foreach loop lets you iterate through all of the elements of a Collection (array, ArrayList, etc.) easily. That is, instead of producing the position, it produces the actual element of the array.

## ***When can I use a foreach loop?***

To use a foreach loop, you need to have a Collection. However, a foreach loop is only useful if you need to access **all** of the elements of an array. If you only need some, use a traditional for loop.

## ***What’s the syntax?***

Here’s an example for a method that prints the value of each element in an array of double:

**public** **void** print(**double**[] array) {

**for** (**double** a: array) {

System.*out*.println(a);

}

}

To use this loop in your own code, just change “**double**“ to whatever your array’s type is.

## ***How does it work?***

Here’s a line-by-line breakdown

* **public** **void** print(**double**[] array) {
  + Just the function declaration. It passes the array you want to use.
* **for** (**double** a: array) {
  + This is the actual foreach loop. It sets the variable a (of type double) to an element in the array and then passes it to the inside of the loop. Note that the variable a is the same as if you accessed it in a standard for loop with array[index].
* System.*out*.println(a);
  + Prints the actual element. Notices that it accesses a directly, not array[a].

## ***When can’t I use a foreach loop?***

A foreach loop is a read loop, not a write loop. It cannot be used to write elements to the array. Since foreach loops are new, not all Java Collection support them. However, regular arrays and ArrayLists do.