***For-each* Loop**

**Purpose**

The basic *for* loop was extended in Java 5 to make iteration over arrays and other collections more convenient. This newer *for* statement is called the *enhanced for* or *for-each* (because it is called this in other programming languages). I've also heard it called the *for-in* loop.

**Use it** in preference to the standard for loop if applicable (see last section below) because it's much more readable.

**Series of values**. The *for-each* loop is used to access each successive value in a collection of values.

**Arrays and Collections**. It's commonly used to iterate over an array or a Collections class (eg, ArrayList).

***Iterable<E>***. It can also iterate over anything that implements the *Iterable<E>* interface (must define iterator() method). Many of the Collections classes (eg, ArrayList) implement *Iterable<E>*, which makes the *for-each* loop very useful. You can also implement *Iterable<E>* for your own data structures.

**General Form**

The *for-each* and equivalent *for* statements have these forms. The two basic equivalent forms are given, depending one whether it is an array or an *Iterable* that is being traversed. In both cases an extra variable is required, an index for the array and an iterator for the collection.

|  |  |
| --- | --- |
| **For-each loop** | **Equivalent for loop** |
| for (*type* *var* : *arr*) {  *body-of-loop*  } | for (int *i* = 0; i < *arr*.length; *i*++) {  *type* *var* = *arr*[*i*];  *body-of-loop*  } |
| for (*type* *var* : *coll*) {  *body-of-loop*  } | for (Iterator<*type*> *iter* = *coll*.iterator(); *iter*.hasNext(); ) {  *type* *var* = *iter*.next();  *body-of-loop*  } |

**Where the *for-each* is appropriate**

Altho the enhanced *for* loop can make code much clearer, it can't be used in some common situations.

* **Only access**. Elements cannot be assigned to, eg, not to increment each element in a collection.
* **Only single structure**. It's not possible to traverse two structures at once, eg, to compare two arrays.
* **Only single element**. Use only for single element access, eg, not to compare successive elements.
* **Only forward**. It's possible to iterate only forward by single steps.
* **At least Java 5**. Don't use it if you need compatibility with versions before Java 5.