For-each Loop

Purpose -iteration over arrays and other collections more convenient. This newer *for* statement is called the *enhanced for* or *foreach* (because it is called this in other programming languages.

Use it in preference to the standard for loop if applicable 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).

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.

```
//... Foreach loop over all elements in arr.
for (type var : arr) {
   body-of-loop
}
```

```
//... For loop using index.
for (int i = 0; i < arr.length; i++) {
   type var = arr[i];
   body-of-loop
}
```

```
double[] ar = {1.2, 3.0, 0.8};
int sum = 0;
for (double d : ar) {  // d gets successively each value in ar.
   sum += d;
}
```

```
double[] ar = {1.2, 3.0, 0.8};
int sum = 0;
for (int i = 0; i < ar.length; i++) {
   sum += ar[i];
}
```

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 can not 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.
- **Does not work with** versions before Java 5.