ENUMS AND FOR-EACH LOOPS

Java 5.0 introduced a new "enhanced" form of the `for` loop that is designed to be convenient for processing data structures. A data structure is a collection of data items, considered as a unit. For example, a listis a data structure that consists simply of a sequence of items. The enhanced `for` loop makes it easy to apply the same processing to every element of a list or other data structure. Data structures are a major topic in computer science, but we won't encounter them in any serious way until Chapter 7. However, one of the applications of the enhanced `for` loop is to `enum` types, and so we consider it briefly here. (Enums were introduced in Subsection 2.3.3.)

The enhanced for loop can be used to perform the same processing on each of the enum constants that are the possible values of an enumerated type. The syntax for doing this is:

```
for ( enum-type-name variable-name : enum-type-name.values() )
   statement
```

or

```
for ( enum-type-name variable-name : enum-type-name.values() ) {
   statements
}
```

If *MyEnum* is the name of any enumerated type, then `MyEnum.values()` is a function call that returns a list containing all of the values of the enum.

(`values()` is a static member function in *MyEnum* and of any other `enum`.) For this enumerated type, the `for` loop would have the form:

```
for ( MyEnum  variable-name  :  MyEnum.values() )
   statement
```

The intent of this is to execute the **statement** once for each of the possible values of the *MyEnum* type. The **variable-name** is the loop control variable. In the **statement**, it represents the enumerated type value that is currently being processed. This variable should **not** be declared before the `for` loop; it is essentially being declared in the loop itself.

To give a concrete example, suppose that the following enumerated type has been defined to represent the days of the week:

```
enum Day { MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY,
SATURDAY, SUNDAY }
```

Then we could write:

```
for ( Day d : Day.values() ) {
   System.out.print( d );
   System.out.print(" is day number ");
   System.out.println( d.ordinal() );
}
```

`Day.values()` represents the list containing the seven constants that make up the enumerated type. The first time through this loop, the value of `d` would be the first enumerated type value `Day.MONDAY`, which has ordinal number 0, so the output would be "MONDAY is day number 0". The second time through the loop, the value of `d` would be `Day.TUESDAY`, and so on through `Day.SUNDAY`. The

body of the loop is executed once for each item in the list `Day.values()`, with `d` taking on each of those values in turn. The full output from this loop would be:

```
MONDAY is day number 0
TUESDAY is day number 1
WEDNESDAY is day number 2
THURSDAY is day number 3
FRIDAY is day number 4
SATURDAY is day number 5
SUNDAY is day number 6
```

Since the intent of the enhanced for loop is to do something "for each" item in a data structure, it is often called a for-each loop. The syntax for this type of loop is unfortunate. It would be better if it were written something like "`foreach Day d in Day.values()`", which conveys the meaning much better and is similar to the syntax used in other programming languages for similar types of loops. It's helpful to think of the colon (:) in the loop as meaning "in."

Source : http://math.hws.edu/javanotes/c3/s4.html