Return to Module Overview Page (https://onlinelearning.berkeley.edu/courses/665040/pages/module-three)

3.5: Constant Values

This is a simple concept, one that everyone understands but unfortunately not everyone applies. If in your program there appears a string or a number that is intended to be a fixed, constant quantity, you should always (a) give it a name, and (b) do so in such a way that the value cannot be inadvertently changed. In C, the mechanism is generally to use the preprocessor:

```
#define WEEKS_IN_A_YEAR 52
```

In Visual Basic, you use CONST:

```
Const WEEKS_IN_A_YEAR As Integer = 52
```

And in Java, the proper idiom is final:

```
final int WEEKS_IN_A_YEAR = 52;
```

final variables must always be initialized when they are declared. (This is a little white lie; actually they must be initialized once and only once, before their first use; but I can't quite explain the rules to you yet, because we haven't grown out of that early awkward stage!) The final keyword can be used in a number of other contexts as well. Local variables can be declared final (an initializer must be supplied). These act just like final variables outside of functions, but they are visible only inside the function in which they are declared. final may also be applied to function parameters, which prohibits assignment to those parameters:

```
void doSomething(final int x)
{
  x = 3; // syntax error!
}
```

The reasons for using final (or #define, or CONST) are manifold, but the most important one, again, is ease of maintenance. By knowing beyond a doubt that nowhere in your code have you inadvertently changed the value of WEEKS_IN_A_YEAR to 51, or 73, you've eliminated one possible source of bugs. And by using a named value instead of littering your code with magic numbers, you've (a) made it easier to read, and (b) made it possible to change all occurrences of that magic number everywhere it appears, effortlessly (perhaps WEEKS_IN_A_YEAR might need adjustment after a meteor impact).