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

2.2: The Basics of Java Syntax

Strong Typing

The most important feature of Java, and the one that distinguishes it from most other programming languages you may have used, is that it is a very *strongly typed language*. What this means is that

1. Every Java variable must be declared before use;
2. Every Java variable can hold a value of one specific data type only; and
3. The specific type is part of the declaration.

Java is more strongly typed than Visual Basic (of course), or C or C++. Java's type system is comparable to Pascal's. This means that although the following is perfectly valid in C, C++, or (conceptually) in Visual Basic,

```
int i = 3.2;
```

a Java compiler will produce

```
test.java:4: Incompatible type for =. Explicit cast needed to convert double to int.
i = 3.2;
  ^ 1 error
```

because the number 3.2 is *not* a valid integer. To make this code acceptable to the compiler (though in this case it's not clear why you'd want to!) you can use a *cast*, which looks like this:

```
int i = (int) 3.2;
```

The parenthetical type name tells the compiler to convert 3.2 to an integer before the assignment; it will do so by dropping the fractional part. There are limits to how much you can accomplish with a cast, however. In C or C++, this is perfectly valid:

```
int i = (int) "Hello, World!";
```

but there is absolutely no way to force a Java compiler to accept it. One very useful result of this is that anytime you see a variable of type "int" you know without a doubt that it contains a number, not a disguised pointer to a character string. Incidentally, the proper variable type to use to refer to a character string in Java is String:

```
String i = "Hello, World";
```

One interesting example of strong typing in Java is the existence of a basic type called boolean:

```
boolean fun = true;
```

Boolean variables can contain only the special literal values "true" and "false." Booleans are important because conditionals in Java are of type boolean. In the fragment

```
if (x < 3) { /* do something */ }
```

the expression (x < 3) is of type boolean, not integer, as it would be in C. As a result, this C programming favorite is an error in Java:

```
int x;
// more code
if (x) { /* do this if x is nonzero */ }
// syntax error!
```

You have to explicitly do the comparison:

```
if (x != 0) { /* do this if x is nonzero */ }
```