

Types

A *type* is a name for a set of properties that apply to a specific variable. For example, if we say something is of type *int* then it has integer values that can be represented in 32 bits (roughly plus or minus 2 billion), it has a specific set of arithmetic operators (including +, -, *, / and %, where the last two are for integer division), and so forth. If we say something is of type *char* then it takes up only 8 bits and has values that are single characters, such as 'a' or '7'.

In Java variables must be declared before they can be used. A *declaration* has the form

```
<type name> <variable name>;
```

or

```
<typename> <list of variable names>;
```

such as

```
int x;
```

or

```
int x, y, z;
```

In many situations you can include with the declaration the initial value of the variable:

```
int x = 10;
```

A *block* is a sequence of statements inside curly brackets, such as

```
if (x < 10) {  
    System.out.print( "One digit" );  
    String size = "small";  
}
```

The code in red forms a block.

A variable declared in a block is only visible within that block. For example:

```
if (x < 10) {  
    int digits = 1;  
}  
else if (x < 100) {  
    int digits = 2;  
}  
System.out.print( digits );
```

This is an error because the variable *digits* in the print statement is not declared. Each branch of the if-statement creates a variable called *digits* which is only visible within that branch.

Here is how that example should have been written:

```
int digits;
if (x < 10) {
    digits = 1;
}
else if (x < 100) {
    digits = 2;
}
System.out.print( digits );
```

Now there is only one variable called *digits* and it is visible through this entire block of code.

Java has 8 primitive types: boolean, char, byte, short, int, long, float, double. We will primarily use only 4 of these: boolean, char, int and double.

The *boolean* data type has values true and false (which must be written in lowercase).

The two common boolean operators are

&& for *and*

and

|| for *or*

For example $(x < 10) \&\& (x \geq 0)$ means that variable x has a value between 0 and 9.

The *int* data type has 32-bit integer values. The largest value this holds is $2^{31}-1$, which is roughly 2 billion:

2^{10} is 1024, which is roughly 10^3 .
So 2^{31} is roughly $2*(2^{10})^3$
or $2*(10^3)^3$,
which is $2*10^9$.

(There; don't you feel better knowing that?)

Powers of 2 come up a lot; it is useful to be able to estimate large powers of 2.

The *double* datatype consists of 64-bit floating point values. The system will automatically convert ints to floats or doubles, but not vice versa:

```
double x = 34; // this is fine  
int y = 3.14; // this is an error
```

Sometimes you need to change the type of an object. This is called *casting* the object into a new type. To do this, put the new type in parentheses in front of the value:

```
int y = (int) 3.14; // this sets y to 3
```

Note that when you cast a float into an int, it is truncated rather than rounded.

The *char* datatype represents single text characters. You may not have worked with char before; Python treats single characters as strings of length 1. In Java the char 'a' is a very different critter from the String "a".

Here are some typical char values:

```
char x = 'a';
```

```
char y = '3';
```

```
char z = '\n'; // the newline character;  
                useful for printing
```

```
char w = '\t'; // the tab character
```

```
char v = '\"'; // the single quote character
```

There is a class `Character` that serves as a *wrapper* for `char` values for times when you need a reference value that holds a single character. The `Character` class has a number of useful static methods.

```
String toString(char ch)
```

```
boolean Character.isLetter(char ch);
```

```
boolean Character.isDigit(char ch);
```

```
boolean Character.isWhitespace(char ch);
```

```
boolean Character.isUpperCase(char ch);
```

```
boolean Character.isLowerCase(char ch);
```

```
char Character.toUpperCase(char ch);
```

```
char Character.toLowerCase(char ch);
```