Assigning Variables

Variables and Types

What we know so far

Computers take inputs, follow instructions, and calculate outputs.

We want to write programs so that we can make computers do useful things for us.

Programs can be written in many languages, including Java, and then compiled so they can be run by a computer.

Programs written in Java use parenthesis, curly braces, semicolons, and quote marks to help express their structure.

Statements

In mathematics, we write stuff like:
5 + x = 8

This statement can then be manipulated according to the rules of algebra so that we can find the value of x.

We do this in order to start from a description of the world and use the power of algebraic reasoning to reach a conclusion.

In programming, we can write stuff like:

```
x = 3;
```

Java is not algebra

When we write a statement like 5 + x = 8 in mathematics, the = or "equals sign" represents a claim that both sides have the same value. We are saying that 5 + x and 8 are exactly equal.

When we write the statement `x = 3;` in Java, we are saying something different. The = here is an operator.

An operator is a command. So our statement in Java is a command to the computer, not a claim about the truth.

The = in Java is called the assignment operator. It is not an indication of equality between values, but rather an instruction that we as programmers tell Java to follow.

=, the assignment operator

```
[thing on left] = [thing on right];
```

When the computer interprets the =, it does this:

"Take the thing on the right and store it in the thing on the left."

So `x = 3;` means: take 3 and store somewhere called x.

Note therefore that the following statement would be meaningless (and incorrect) in Java:

```
5 = x;
```

Variables

When the computer executes our code and stores 3 in x, then we can use x as a symbol in future statements.

```
x = 3;
System.out.println( x );
```

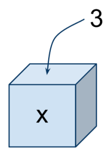
This will print 3, not the letter x.

When Java encounters a symbol (that is, the name of a variable) it substitutes the value of that symbol before it continues with the rest of the specified operations.

The box metaphor

Sometimes it's helpful to think of a variable as a box.

If we say **`x = 3;`** to Java, we can think of Java creating a box with the label x on it and then placing the value 3 inside it.



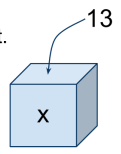
More Operators

We can also say something like:
`x = 3 + 10;`

The assignment operator says: take the thing on its right and store it into the thing on its left. And what is on its right?
```
    3 + 10
```

But we don't store that mathematical statement. The **+** character is another operator: the addition operator.

The addition operator says: take the thing on its left and add that to the thing on its right.



Symbol Resolution

We can use the variables we define in later statements:

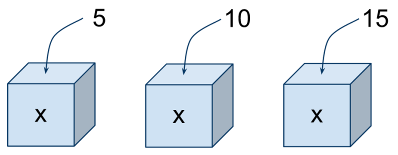
```
x = 3 + 10;
y = x - 4;    // the value of y becomes 9
```

When the program execution reaches the second line, the assignment operator takes what's on the left and stores it into what's on the right. And what's on the right?
```
    x - 4
```

In order for the subtraction operator to do its work here, we must first find the value of x. So Java goes and gets that value from the box, and uses it in the computation.

Variables store one value at a time

```
x = 5;
x = 5 + 5;
x = 20 - 5;
```



When we assign a new value to the variable, the previous value is gone completely. The box can only hold one thing.

Types and Declarations

We can make statements such as:
`x = 40;`

We can also make statements such as:
`y = "Hello there";`

How does Java know what kind of thing **`x`** and **`y`** represent? How does it know what kind of thing can go into each box?

We, the programmers, have to tell it. We do this by declaring the variable with a type:
```
int x;
String y;
```

Types and Declarations

Java recognizes many data types, but we'll focus on these four:

```
int     // for integer, whole number values
double  // for floating-point, decimal values
boolean // for boolean, true/false values
String  // for strings of character values
```

A variable must be declared before it can be used:
```
  int x;
  x = 32;
```
The declaration can also be combined with an assignment:
```
  int x = 32;
```