LESSON: Variables

```
public static boolean frontIsDark()
{
    Robot.move();
    if (Robot.onDark())
    {
        backUp();
        return true;
    }
    else
    {
        backUp();
        return false;
    }
}
```

Duplicate code!

What should happen when we copy/paste?

Alarms should go off in our head,
signaling that there must be a better way!

```
public static boolean frontIsDark()
{
    Robot.move();

    <Somehow test Robot.onDark()
     and remember the answer>

    backUp();

    return <that answer from before>;
}
```

How do we tell Java to remember a value?

Introducing variables ...

WHAT IS A VARIABLE?

A variable is a name for a place in memory, where we can store a value.

WHEN SHOULD I USE A VARIABLE?

Use a variable whenever you want Java to remember a value.

```
Example:


boolean x;
x = true;
System.out.println(x);   true

boolean y;
y = x;
System.out.println(x);   true
System.out.println(y);   true

x = !x;
System.out.println(x);   false
System.out.println(y);   true
```

```
DECLARATION STATEMENTS


For Example:    boolean x ;


What It Does

    Declares that x will someday store a boolean value.

    You cannot store a value in a variable until that
    variable has been declared.

    This helps the compiler catch typos in your code.


In General:     ____ ________ ;
                type variable
```

```
ASSIGNMENT STATEMENTS

For Example:    x = !y;

Pronounced:     x "gets the value of" not y.

In general:     ________ = __________ ;
                variable   expression

What It Does

    Finds the value of the expression on the right.

    Assigns that value to the variable on the left.

    This is not = from math class!
    We are not testing if the left and right are equal.
    We are not setting one side equal to the other.
    = is not symmetrical.
```

```
PRINT STATEMENTS


For Example:    System.out.println(x && !y);


In general:     System.out.println( __________ );
                                    expression

What It Does

    Prints the value of the expression to the console.

    This is a statement.

    It does not return a value.

    A value is printed as a side effect of executing
    a print statement.
```

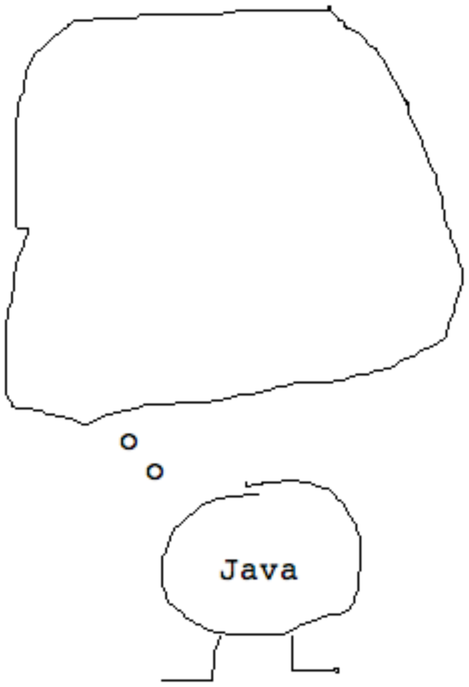
7 Kinds Of Statements

```
FILENAME . METHODNAME ();

if ( BOOLEXP ) { STMTS } else { STMTS }

return EXP ;

while ( BOOLEXP ) { STMTS }

TYPE VARIABLE ;

VARIABLE = EXP ;

System.out.println( EXP )
```

So far,
everything Java remembers
has been visible to us:
where the robot is,
which way its facing,
which squares are dark.

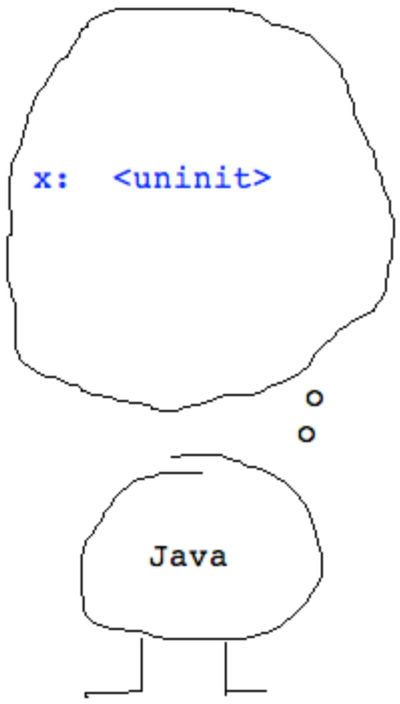
But when you execute an
assignment statement,
Java remembers something
that isn't visible to us.

Therefore, when we trace
our code, we'll need to
write down the values of
variables.



THE LIFE OF A VARIABLE

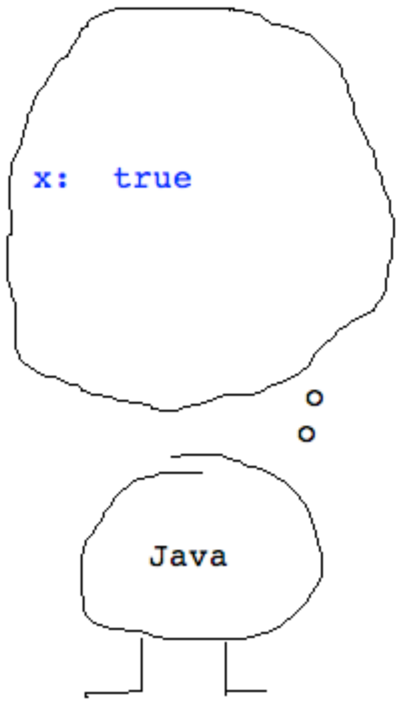
```
    boolean x;

    x = true;

    System.out.println(x);

    x = !x;
 }
```



We DECLARE x.

THE LIFE OF A VARIABLE

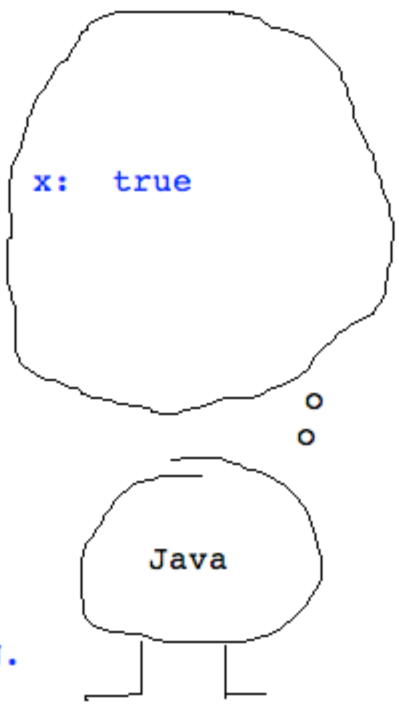
```
    boolean x;

    x = true;

    System.out.println(x);

    x = !x;
 }
```



We INITIALIZE x.

THE LIFE OF A VARIABLE

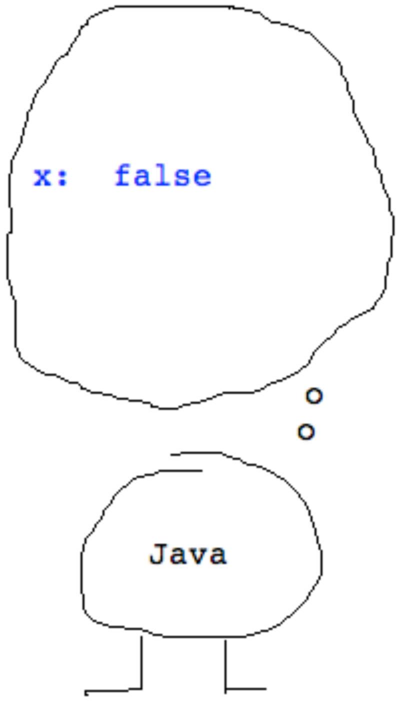
```
     boolean x;

     x = true;

     System.out.println(x);

     x = !x;
 }
```



We can now use x as an EXPRESSION.

THE LIFE OF A VARIABLE

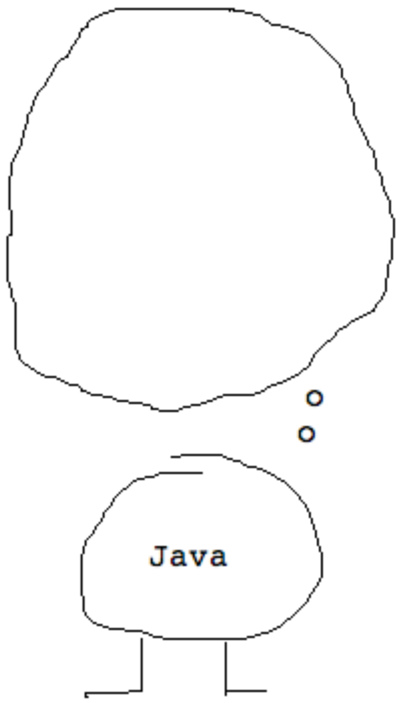
```
     boolean x;

     x = true;

     System.out.println(x);

     x = !x;
 }
```



And even change x.

THE LIFE OF A VARIABLE

```
     boolean x;

     x = true;

     System.out.println(x);

     x = !x;
 }
```



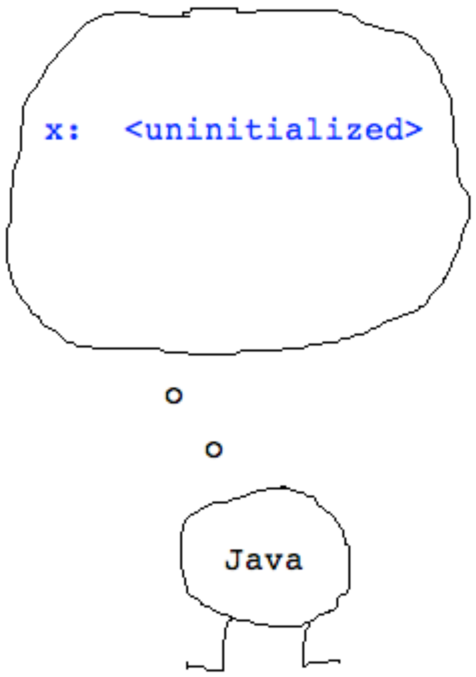
x DIES at the closing brace.

```
public static boolean frontIsDark()
{
    Robot.move();
    boolean dark;
    dark = Robot.onDark();
    backUp();
    return dark;
}
```

```
//after:  robot has moved to the wall and has returned
//        to its original location/direction
public static void goToWallAndBack()
{
}
```

What does Java need to remember
as the robot performs this task?

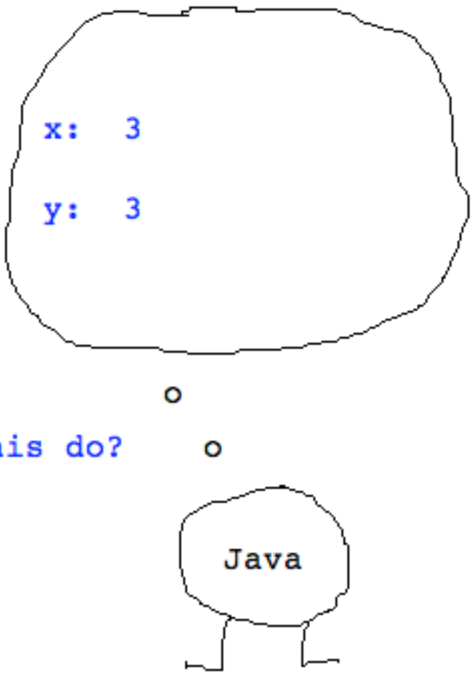
```
int x;
```



This is how you declare
a variable when you want
Java to remember an
integer.

```
int x;

x = 3;

int y;

y = x;

x = x + 1;
```

What will this do?

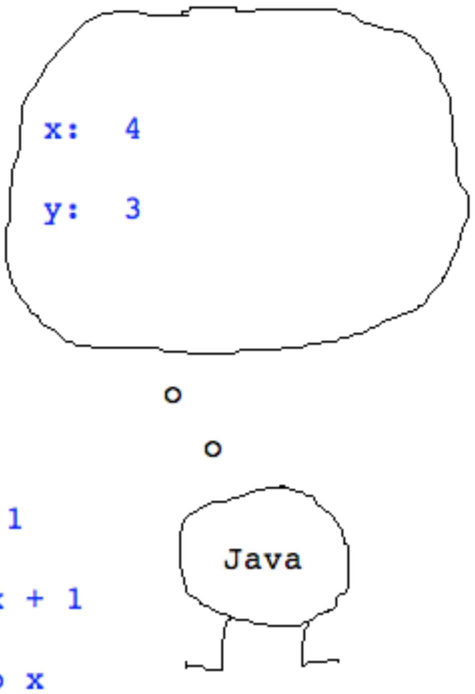


```
int x;

x = 3;

int y;

y = x;

x = x + 1;
```

x "gets the value of" x + 1

First, find the value of x + 1

Then, assign that value to x



```
//after:  robot has moved to the wall and has returned
//        to its original location/direction
public static void goToWallAndBack()
{
    int distFromStart;
    distFromStart = 0;
    while (Robot.frontIsClear())
    {
        Robot.move();
        distFromStart = distFromStart + 1;
    }
    while (distFromStart > 0)
    {
        backUp();
        distFromStart = distFromStart - 1;
    }
}
```

```
//returns: distance to nearest wall
//after:   robot has moved to the wall and has returned
//         to its original location/direction
public static int distanceToWall()
{
    ...
}
```

This method returns an integer.

What will this method need to remember?

2 integers. how many steps we are from the start, and the answer to return

```
public static int distanceToWall()
{
    int distFromStart;
    distFromStart = 0;
    while (Robot.frontIsClear())
    {
        Robot.move();
        distFromStart = distFromStart + 1;
    }
    int distToWall;
    distToWall = distFromStart; ← save this distance
    while (distFromStart > 0)     to return it later
    {
        backUp();
        distFromStart = distFromStart - 1;
    }
    return distToWall;
}
```

```
Operations on Integers


      BOOLEXP:   ...
                 EXP == EXP      equals
                 EXP != EXP      not equals
                 INTEXP < INTEXP
                 INTEXP > INTEXP
                 INTEXP <= INTEXP
                 INTEXP >= INTEXP


      INTEXP:    ...
                 INTEXP + INTEXP
                 INTEXP - INTEXP
                 INTEXP * INTEXP  multiply
                 INTEXP / INTEXP  divide
                 INTEXP % INTEXP  remainder ("mod")
                 - INTEXP
```

What's wrong with this code?

```
distanceToWall();
```

I've asked the computer to
find the distance to the wall,
but I'm not doing anything with the result.

I call void methods for their side effects.
Calls to void methods are used as statements.

```
Robot.move();

turnRight();
```

I call non-void methods for their return value.
Calls to non-void methods are used as expressions.

```
dark = Robot.onDark();

if (Robot.frontIsClear()) ...

distance = distanceToWall();

if (distanceToWall() < 5) ...
```

```
//asks user for 2 numbers and prints out whichever
//number is higher
public static void max()
{
    System.out.println("Enter first number");
    int x;
    x = Integer.parseInt(Util.input());
    System.out.println("Enter second number");
    int y;
    y = Integer.parseInt(Util.input());
    if (x > y)
    {
        System.out.println(x);
    }
    else
    {
        System.out.println(y);
    }
}
```

See Animation.java for our animation code ...