# Expressions and Assignment Statements

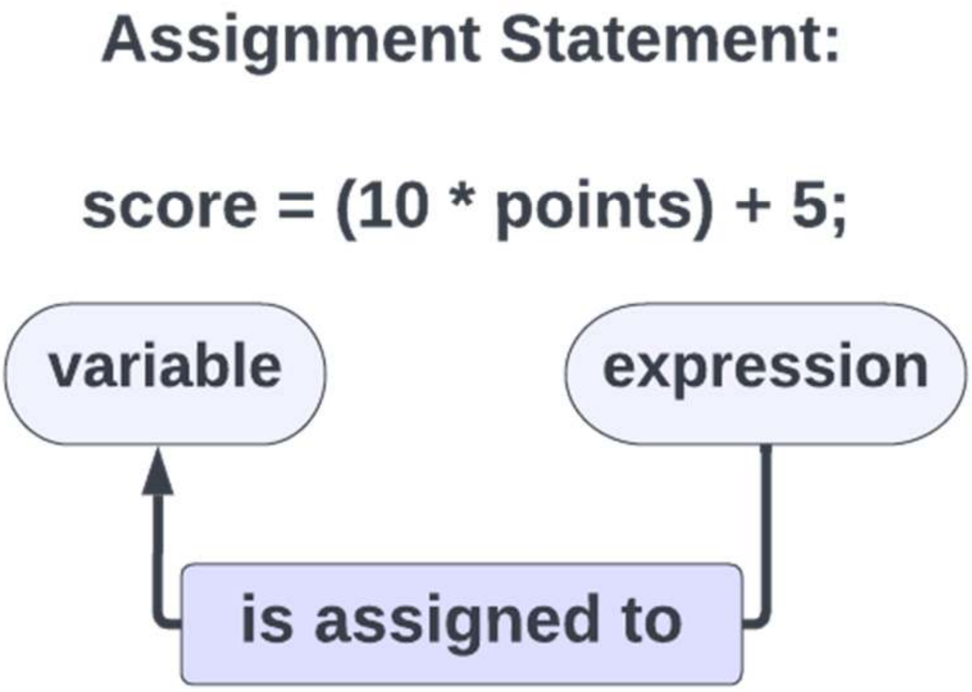
Expressions and assignment statements are important building blocks for any Java program. They allow you to manipulate and work with variable data in your programs. Let's talk about exactly what we can do with them.

#Expressions and Assignment Statements

An expression is a combination of one or more **values**, **variables**, and/or **operators** that results in a **single value**. For example, the expression `2 + 3` results in the value `5`, and the expression `x + y` adds the values of the variables `x` and `y`. These are examples of expressions, which can then be used as is, or stored as variables via an **assignment statement**.

Assignment statements initialize or change the value stored in a variable using the **assignment operator** `=`. An assignment statement always has a single variable on the left hand side of the `=` sign. In Java, this variable will be accompanied by its type as well, as in `int number`. On the right side of the equals sign, we have the **value of the expression.** This can be a primitive value, such as `5`, or whatever matches the declared type, including a String like `"John"`. The value could also contain math operators and other variables, as needed, such as setting a variable `z` equal to the value of `x + y`. Whatever the case may be, the value of the expression will be copied into the memory location of the variable on the left hand side.

InfoWarningTip
Remember that a variable holds a value that can change. Anytime you see the assignment operator `=`, the variable is being **reassigned** to that new value.



Remember: variable = expression.

One way to remember that the variable is on the left is to say “gets” or “is assigned”, rather than saying "equals" when reading your code. You won't often have to read your code aloud, but reading it in your head is an important step in understanding what's going on. Affecting your language in this way, to interpret the symbols on the screen in ways that match what's going on in computer lingo, is a good way to keep your brain working in sync with your computer. Reading the equals sign as "gets" or "is assigned" will help you remember that the variable on the left hand side is getting/being assigned the value of what happens on the right. In the figure above, score is assigned the value of 10 times points (which is itself another variable) plus 5.

The following video by Dr. Colleen Lewis shows how variables can change values in memory using assignment statements. We'll discuss stack frames and `public static void main` at a later date. For now, just pay attention to how the variables are assigned (and reassigned) as the program progresses:

Open in new tab

As we saw in the video, we can set one variable to a copy of the value of another variable like `y = x;`. This won’t change the value of the variable that you are copying from, and changing the original value `x` won't change something that had been previously copied from that variable `y`. To change `y` after the video's explanation, we'd have to reassign `y` using a statement such as `y = x;` again, which would then access `x`'s new value. After all, it's a copy of what's there at the time, not an inalterable link between the two variables.

#**Coding Exercise:**

Step through the code below and see how the values of the variables change.

Run

```
  public class VariableAssignment {

   public static void main(String[] args) {
     int x = 3;
     int y = 2;
     System.out.println(x);
     System.out.println(y);
     x = y;
     System.out.println(x);
     System.out.println(y);
     y = 5;
     System.out.println(x);
     System.out.println(y);
   }
}
```

CONSOLESHELL

#**Coding Exercise:**

The following program is supposed to figure out the total money value given the number of dimes, quarters and nickels. There is an error in the calculation of the total.

Fix the error in the following code to compute the correct amount in cents:

Run

```
public class CalculateMoney {
   public static void main(String[] args) {
        int numDimes = 7;
        int numQuarters = 3;
        int numNickels = 8;
        // Corrected calculation for total amount in cents
        int total = numDimes * 10 + numQuarters * 25 + numNickels * 5;
        System.out.println("Total = " + total);
    }
}
```

CONSOLESHELL

#**Coding Exercise:**

Calculate and print the total pay given the weekly salary and the number of weeks worked. Use string concatenation (see hint below IDE if you need it) with the totalPay variable to produce the output `Total Pay = 3000`. Don’t hardcode the number 3000 in your print statement.

Run

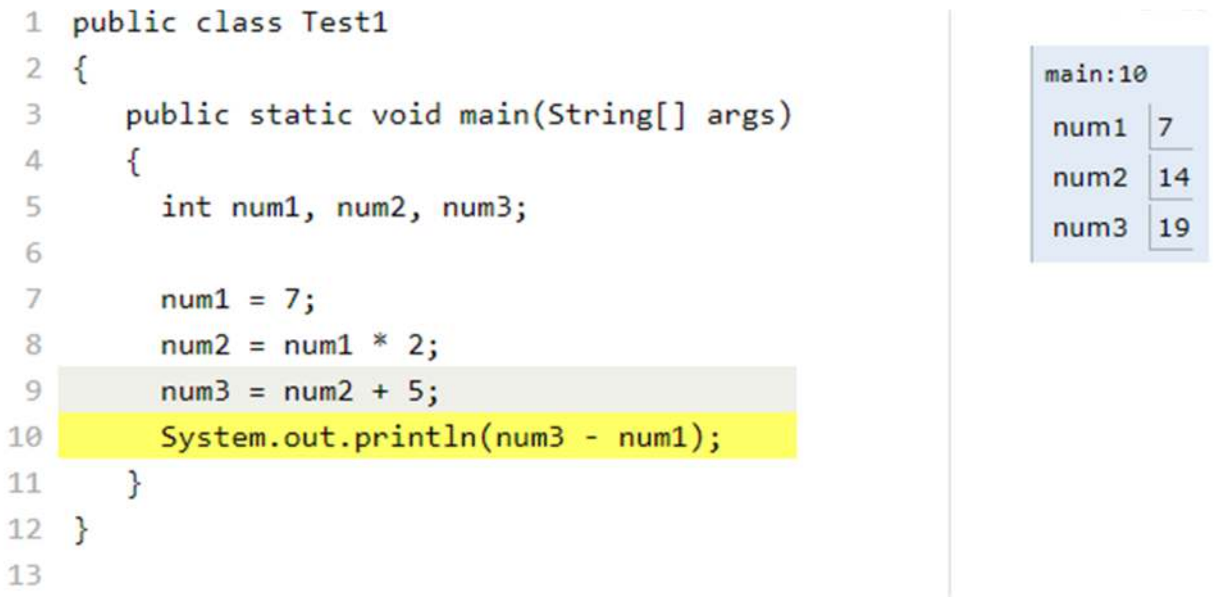
```
public class SalaryCalculation {
    public static void main(String[] args) {
        int weeklySalary = 500;
        int numWeeks = 6;
        int totalPay;
        totalPay = weeklySalary * numWeeks;
        // Print the total pay using string concatenation
        System.out.println("Total Pay = " + totalPay);
    }
}
```

CONSOLESHELL

InfoWarningTip

If you're unsure what string concatenation is, the previous question's IDE used it to print "Total = " and the total amount of money. Copy the structure and you'll get the job done in no time.

#**Check Your Understanding**



```
1  public class Test1
2  {
3     public static void main(String[] args)
4     {
5       int num1, num2, num3;
6
7       num1 = 7;
8       num2 = num1 * 2;
9       num3 = num2 + 5;
10      System.out.println(num3 - num1);
11    }
12 }
13
```

```
main:10
 num1  7
 num2  14
 num3  19
```

A code snippet from a program

The code above shows the variable state in memory after line 9 is executed. What is printed when line 10 is executed?

#Debugging Exercise:

Assume you have a package with a given height of 3 inches and a width of 5 inches. If the package is rotated 90 degrees, you should swap the values for the height and width. The code below makes an attempt to swap the values stored in two variables **h** and **w**, which represent height and width. Variable **h** should end up with w's initial value of 5 and w should get h's initial value of 3. Unfortunately this code has an error and does not work. Step through the code to understand why it fails to swap the values in h and w. Note: we will fix the code further down the page here, so just try to understand it and then move on for now.

Run

```
public class ErrorSwap {
   public static void main(String[] args) {
     int h = 3;
     int w = 5;
     System.out.println(h);  //3
     System.out.println(w);  //5
     h = w;
     w = h;
     System.out.println(h);  //expected 5
     System.out.println(w);  //expected 3
   }
```

}
CONSOLESHELL

In the text box below, explain in your own words why the `ErrorSwap` program code does not swap the values stored in `h` and `w`:
The ErrorSwap program code does not swap the values stored in `h` and `w` because of the way the variables are being reassigned.

#**Check your understanding**
Swapping two variables requires a third variable. Before assigning `h = w`, you need to store the original value of `h` in the temporary variable. In the mixed-up program below, drag the blocks to the right to put them in the right order. There may be unused blocks.

The following has the correct code that uses a third variable named "temp" to swap the values in **h** and **w**. The code is mixed up and contains extra block(s) which is/are not needed in a correct solution. Drag the needed blocks from the left into the correct order on the right, then check your solution.

Options

```
w = h;
```

Solution

```
int h = 3;
int w = 5;
int temp = 0;


temp = h;


h = w;


w = temp;
```

#Incrementing the value of a variable

If you wanted to use a variable to keep track of a score, you would probably increment it (add one to the current value) whenever the score went up. You can do this by setting the variable to the current value of the variable equal to that current value plus one (`score = score + 1`) as shown below. This formula might look a little crazy in a math class, but it makes sense in coding because the variable on the left is set to the value of the arithmetic expression on the right. Reassignment of a variable, as shown in the code below, is a completely valid coding strategy.

#Coding Exercise:

Step through the code and see how the score value changes.

```
public class UpdateScore {
    public static void main(String[] args) {
        int score = 0;
        System.out.println(score);
        score = score + 1;
        System.out.println(score);
    }
}
```

CONSOLESHELL

InfoWarningTip

Pro-tip: another way to add one to a given variable is to simply write the name of the variable followed by `++`. In the above, you could increment the score with `score++;`, completely omitting the equals sign assignment operator. This also works with `--` if you need to decrement something by one.

#Check Your Understanding

What is the value of b after the following code executes?

```
int b = 5;
b = b * 2;
```

b = 5

b = 2

b = 7

b = 10

What are the values of x, y, and z after the following code executes?

```
int x = 0;
int y = 1;
int z = 2;
x = y;
y = y * 2;
z = 3;
```

x = 0, y = 1, z = 2

x = 1, y = 2, z = 3

x = 2, y = 2, z = 3

x = 1, y = 0, z = 3

#Extra Resources

You can read more about Expressions and Statements here:

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/expressions.html

Back
Unsubmit
Next