Recursion

Mr. Poole
Java

Recursion takes time to understand.

It comes with practice.

Mathematical Recursive Formula

Recursive formulas give us two pieces of information:

1. The first term of the sequence
2. The pattern rule to get any term from the term that comes before it

Here is a recursive formula of the sequence $3, 5, 7, \ldots$ along with the interpretation for each part.

$$\begin{cases} a(1) = 3 & \leftarrow \text{the first term is } 3 \\ a(n) = a(n-1) + 2 & \leftarrow \text{add } 2 \text{ to the previous term} \end{cases}$$

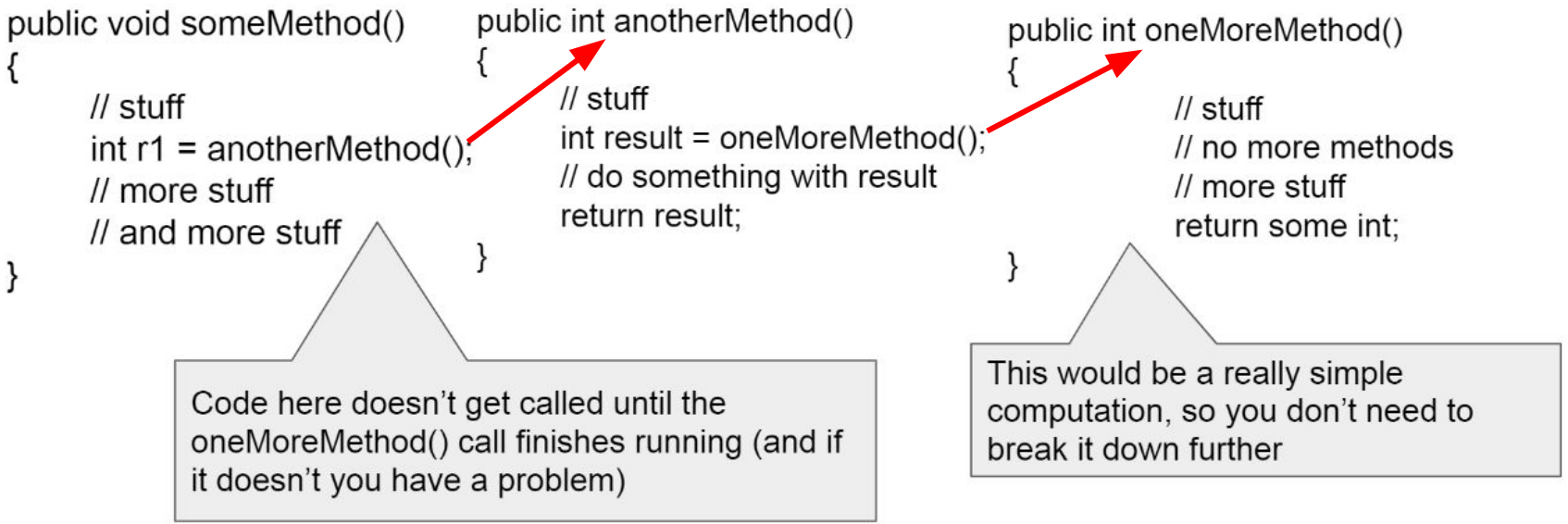
Recursion in work

$$\begin{cases} a(1) = 3 & \leftarrow \text{the first term is } 3 \\ a(n) = a(n-1) + 2 & \leftarrow \text{add } 2 \text{ to the previous term} \end{cases}$$

$a(n) = a(n-1) + 2$			
$a(1)$			$= 3$
$a(2)$	$= a(1) + 2$	$= 3 + 2$	$= 5$
$a(3)$	$= a(2) + 2$	$= 5 + 2$	$= 7$
$a(4)$	$= a(3) + 2$	$= 7 + 2$	$= 9$
$a(5)$	$= a(4) + 2$	$= 9 + 2$	$= 11$

We can apply this same concept to computer science

How Calling a Method Works



Recursion Steps - MUST HAVE

1. **Base Case (i.e. when to stop)**
2. **Work towards Base Case**
3. **Recursive Call (i.e. calling itself)**

For example, we can define the operation "find your way home" as:

1. If you are at home, stop moving.
2. Take one step toward home.
3. "find your way home".

Recursion Example - Factorial

Factorial (n!) : product of all the numbers 1...n. So

4! = 4 * 3 * 2 * 1

Notice:

4! = 4 * 3!

3! = 3 * 2!

2! = 2 * 1!

1! = 1 (by definition) - this is so simple you don't have anything else to do

1. Base Case - Factorial

When input == 1.

```
public static void factorial(int n){
    if(n == 1)
        return 1;
}
```

2. Work towards Base Case

Call n-1 to move closer to 1

```
public static void factorial(int n){
    if(n == 1)
        return 1;
    else
        return n * factorial(n-1);
}
```

3. Recursive Call

Call factorial on n-1

```
public static void factorial(int n){
    if(n == 1)
        return 1;
    else
        return n * factorial(n-1);
}
```

4. Do something

Here we are multiplying by n for each call

```
public static void factorial(int n){
    if(n == 1)
        return 1;
    else
        return n * factorial(n-1);
}
```

Let’s try recursion!

Given one number, print out that number all the way to 1 and back up to the number.

Ex: function(3)

Output: 3 2 1 1 2 3

Example

Lab: Recursion

1. Implement factorial as recursive
2. Implement a recursive Power function
    a. **public static int recurPower(int base, int n)**
    b. This method should recursively determine the base raised to the nth power.
    c. Assume 0 <= n