LESSON 1:

Review of Recursion

What is Recursion?

Recursion is the idea of a method running itself inside of itself

```
public int f(x) {
    if (x==1)
        return 1;
    else
        return f(x-1);
}
```

f(5) = ~~f(4)~~ 1

f(4) = ~~f(3)~~ 1

f(3) = ~~f(2)~~ 1

f(2) = ~~f(1)~~ 1

f(1) = 1

```
public int f(int x) {
    return f(x-1);
}
```

Infinite Recursion

f(5) = f(4)

f(4) = f(3)

f(3) = f(2)

. . . .

f(-2000) = f(-2001)

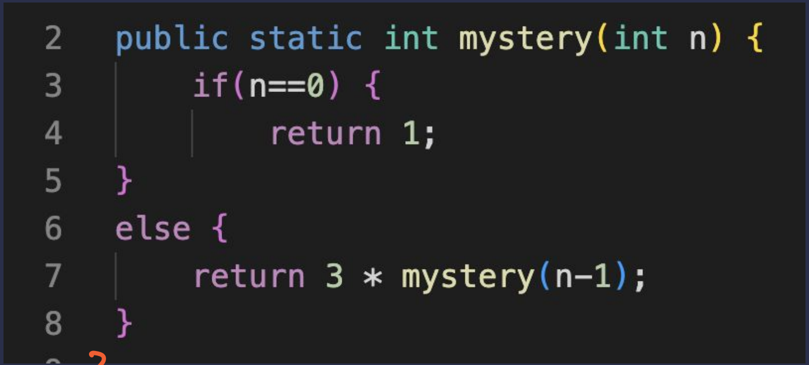
Prevent Inf recur:

① Inputs of the function need to more towards the base case

② Base case needs to not contain a recur call.

Analyzing Recursion

mystery = m



```
public static int mystery(int n) {
    if(n==0) {
        return 1;
}
else {
    return 3 * mystery(n-1);
}
```

3

$m(5) = 3 * \cancel{m(4)}^{81} = 243$

$m(4) = 3 * \cancel{m(3)}^{27} = 81$

$m(3) = 3 * \cancel{m(2)}^{9} = 27$

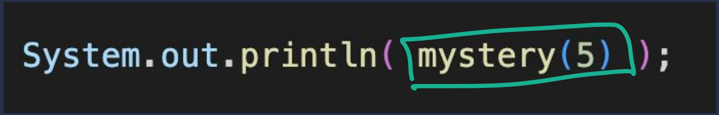
$m(2) = 3 * \cancel{m(1)}^{3} = 9$

$m(1) = 3 * \cancel{m(0)}^{1} = 3$

$m(0) = 1$

$$\begin{array}{r} 81 \\ \times\ 3 \\ \hline 243 \end{array}$$

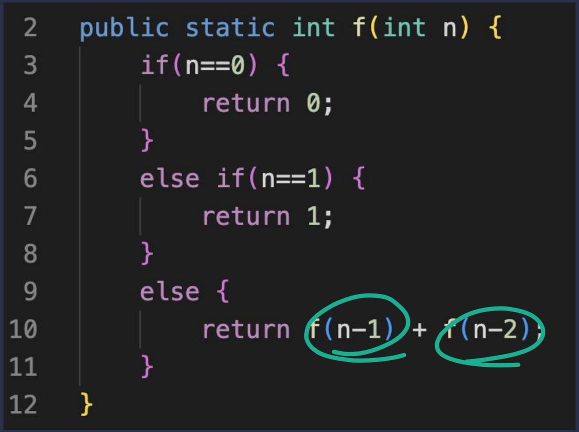
What is the result of:



```
System.out.println(mystery(5));
```

$= 243$

Analyzing Recursion, Part 2



```java
public static int f(int n) {
    if(n==0) {
        return 0;
    }
    else if(n==1) {
        return 1;
    }
    else {
        return f(n-1) + f(n-2);
    }
}
```

$$f(5) = \overset{3}{\cancel{f(4)}} + \overset{2}{\cancel{f(3)}} = 5$$

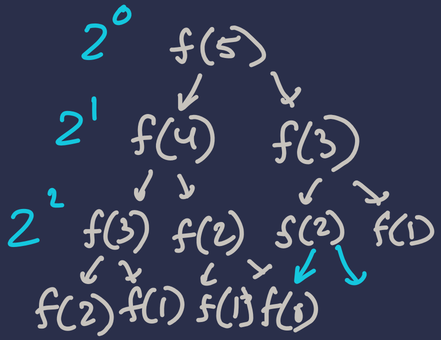
$$f(4) = \overset{2}{\cancel{f(3)}} + \overset{1}{\cancel{f(2)}} = 3$$

$$f(3) = \cancel{f(2)}^{1} + \cancel{f(1)}^{1} = 2$$

$$f(2) = \cancel{f(1)}^{1} + \cancel{f(0)}^{0} = 1$$

$$f(1) = 1$$

$$f(0) = 0$$



What is the result of:

```java
System.out.println( f(5) );
```

= 5

Why Do We Care?

- Break down large problems into smaller ones

① Sorting

② Data Structures