Model 1 Summation

"In mathematics, ***summation*** (capital Greek sigma symbol: Σ) is the addition of a sequence of numbers; the result is their sum or total."

$$\sum_{i=1}^{100} i = 1 + 2 + 3 + \ldots + 100 = 5050$$

Source: https://en.wikipedia.org/wiki/Summation

Questions (20 min) Start time:

1. Consider how to calculate $\sum_{i=1}^{4} i = 10$.

a) Write out all the numbers that need to be added:

$\sum_{i=1}^{4} i =$ $4 + 3 + 2 + 1$

b) Show how this sum can be calculated in terms of a smaller summation.

$\sum_{i=1}^{4} i =$ $4 + \sum_{i=1}^{3} i$

2. Write an expression similar to #1b showing how any summation of n integers can be calculated in terms of a smaller summation.

$$\sum_{i=1}^{n} i = \text{n} + \sum_{i=1}^{n-1} i$$

3. What is the base case of the summation? (Write the complete formula, not just the value.)

$$\sum_{i=1}^{1} i = 1$$

4. Implement a recursive method that takes a single parameter `n` and returns the sum $1 + 2 + \ldots + n$. It should only have an `if` statement and two `return` statements.

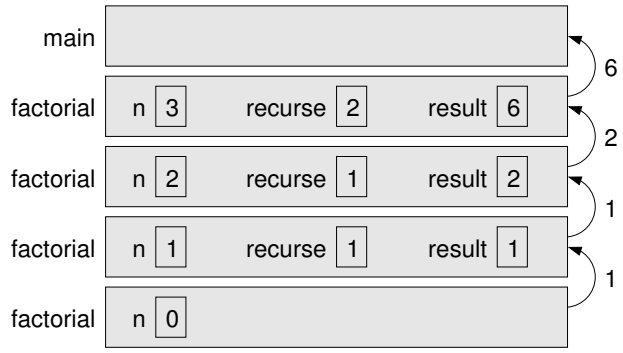
```
public static int summation(int n) {
    if (n == 1) {
        return 1;
    } else {
        return n + summation(n - 1);
    }
}
```

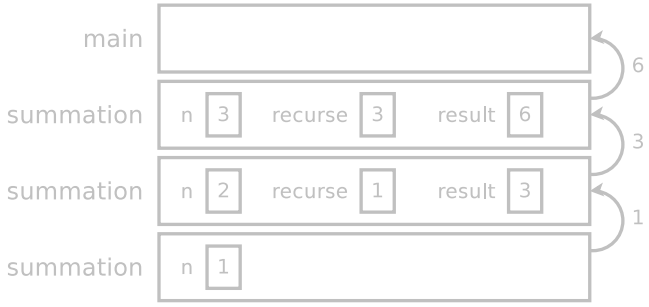
5. Discuss how the `factorial` method below uses temporary variables. What lines would you have to change to implement the `summation` method instead?

```
public static int factorial(int n) {
    if (n == 0) {
        return 1;  // base case
    }
    int recurse = factorial(n - 1);
    int result = n * recurse;
    return result;
}
```

1) Rename the method to `summation`.
2) Change the base case to be `if (n == 1)`.
3) The recursive step must invoke `summation`.
4) The result must add instead of multiply.

6. Here is a stack diagram of `factorial(3)` when invoked from `main`. Draw a similar diagram for `summation(3)` as described in the previous question.





7. Why are there no values for `recurse` and `result` in the stack diagram for the last call to `factorial` (when `n == 0`)?

The method returns without declaring and using those variables.

8. Looking at the stack diagram, how is it possible that the parameter `n` can have multiple values in memory at the same time?

Each distinct method call has its own memory for parameters and local variables. The value of `n - 1` in the first method call becomes the value of `n` in the next.