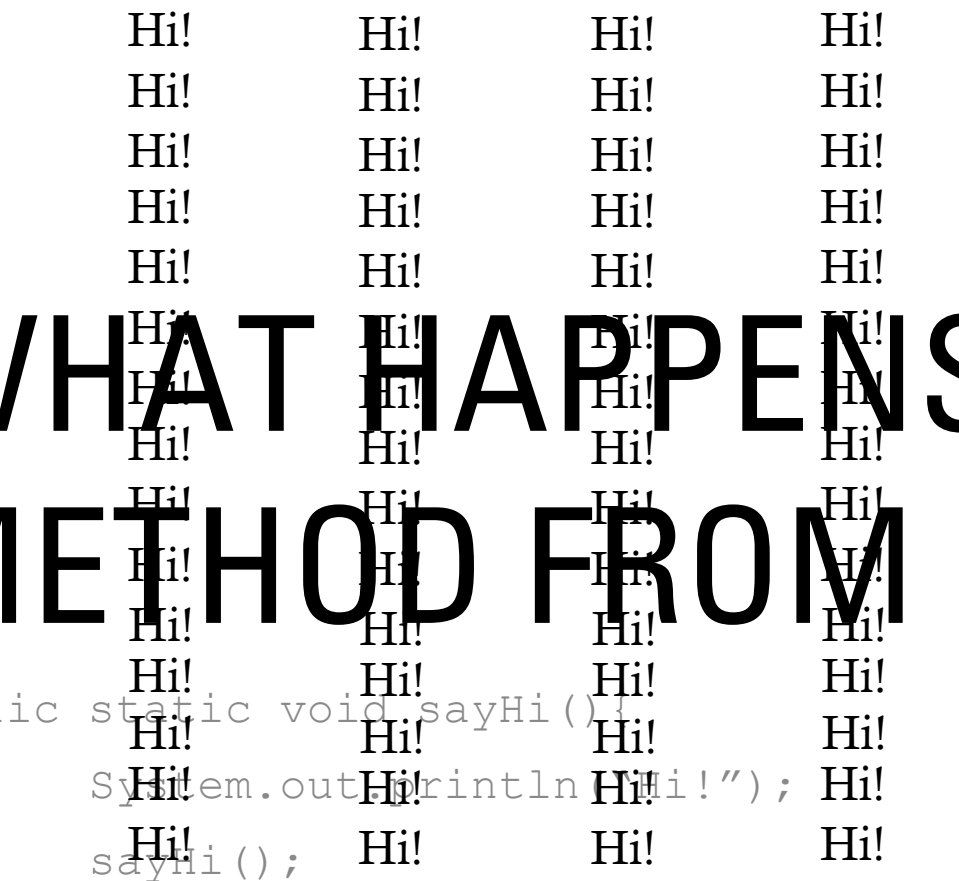

WEEK TEN

Acknowledgements: Slides created based off material provided by Dr. Michael Raymer and Dr. Travis Doom



WHAT HAPPENS IF
METHOD FROM IN

```
public static void sayHi() {
    System.out.println("Hi!");
    sayHi();
}
```

Exc

Exception in thread "main" java.lang.StackOverflowError

WHAT JUST HAPPENED?

Recursion!

RECURSION

- **Recursive method:** any method that calls itself
- Starts with a complicated problem and breaks it down into smaller and smaller problems
 - aka divide and conquer decomposition
- Why use recursion?
 - Sometimes simpler/easier to understand and maintain
 - Sometimes more flexible
 - Sometimes faster
 - **Short answer:** it's another tool in your toolbox

OKAY, BUT HOW DO WE AVOID A STACK OVERFLOW ERROR?

An endpoint

Termination condition

A base case!

THE BASE CASE

- A condition where we no longer make a recursive call
- Returns a basic and simple value
- Designed to reverse the recursion process to return to the original recursive call

```
public static int sum(String num){  
    if (num.length() == 1) {  
        return Integer.parseInt(num);  
    } else {  
        return Integer.parseInt(num.substring(0, 1)) + sum(num.substring(1));  
    }  
}
```

HOW TO UNDERSTAND RECURSION

- A few very simple base cases (often just one)
- Rules to break down complex cases into simpler cases of the same general problem
- Are we done yet? If so, return the results.
- If not, simplify the problem (move towards the base case), by constructing a solution from smaller similar problems
- If you have an already solved similar but simpler problem, how can that help you solve a more complex problem?

PRACTICE TOGETHER

- Reverse a word
- **Input:** hello **output:** olleh
- What is our base case?
- What small step can we take before making the recursive call?

YOUR TURN

- Write a recursive method that counts how many even numbers are in a String
- You can assume the String will only include numbers
- **Input:** “2395832” **Output:** 3