# Recursion

CS 1410 2023
Cody Henrichsen

1

## Vocab

- Recursion
- Base Case
- Recursive Case
- Infinite Case
- Stack Overflow
- Integer Overflow

2

## What is recursion?

- Defining a method with itself
- A problem where the solution for the large problem is the same as the solution for the smaller problem
- Every recursive solution can be done iteratively
- Uses an implicit data structure to maintain information
  - The stack of calls to the method
- Potential for algorithmic elegance but it can weighed down by efficiency
- Can use helper methods to make abstraction easier

3

## Base Case

- When recursion MUST stop
- Must be reachable
- Based on the parameter(s) of the method

4

## Recursive Case

- Call the method again with a modified parameter(s)
- Should shrink the problem
- Modification of the parameter makes it approach the base case
- We want the execution to stop eventually

5

## Recursive Structure: Return Type

- Method with a parameter(s)
- If else in the method based on the parameter
- Base case for the if returns the value
- Recursive case returns a value based on a call to the method with a changed parameter

6

## Recursive Structure: Void

- Method with a parameter(s)
- If/else in the method based on the contents of the parameter
- Base case does "something"
  - Print
  - Nothing
- Recursive case calls the method with a modified parameter

7

## Factorial

- The product of all numbers leading to this number
- Remember 0! is 1

8

## Factorial Logic

Code

- Iterative
  - Loop from 2 to n multiplying and saving back into the variable
  - `value *= index;`

Math

$$factorial(x) = \begin{cases} 1 & , x = 0 \\ \Pi_{k=1}^{n} k & , x > 0 \end{cases}$$

9

## Factorial Code

```
private int factorial(int number)
{
    if (number == 0 || number == 1)
    {
        return 1;
    }
    else
    {
        return number * factorial (number - 1);
    }
}
```

10

## Fibonacci

- Nifty sequence of numbers AKA a pattern
- The Fibonacci number n is the sum of the two previous Fibonacci values

| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Value | 1 | 1 | 2 | 3 | 5 | 8 | 13 | 21 | 34 | 55 | 89 |

11

## Fibonacci Logic

- Iterative
  - Loop from 2-n adding values to the sum and swapping values as you go

$$fibonacci(x) = \begin{cases} 1 & ,x = 0, 1 \\ fibonacci(x-1) + fibonacci(x-2), x > 1 \end{cases}$$

12

## Fibonacci Code

```
private int fib(int number)
{
    if (number == 0 || number == 1)
    {
        return 1;
    }
    else
    {
        return fib (number - 1) + fib(number - 2);
    }
}
```

13

## Type Overflow

- Int
  - Factorial breaks the int data type at 17
  - Overflow exceeded the bounds completely at 34
- Double
  - Factorial hits infinity at 160

14

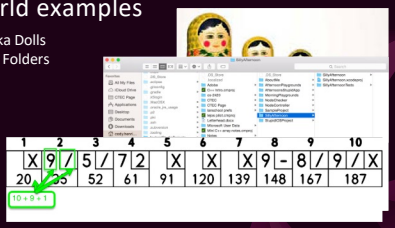
## Stack Overflow

- Incorrect base case
  - Infinite recursion
  - Don't do recursive constructor calls!

15

## Real world examples

- Matryoshka Dolls
- Computer Folders
- Bowling!




16

## Coming soon

- Sort and search
- Tree Traversal

17