

# Recursion and Tail Recursion

## Functional Programming

Jens Egholm Pedersen and Anders Kalhauge



Spring 2018

Stacks in CPUs

Recursion

What do we need to know? What does the frame contain?

What do we need to know? What does the frame contain?

What do we need to know? What does the frame contain?

- Space for return value and arguments

What do we need to know? What does the frame contain?

- Space for return value and arguments
- Return address

What do we need to know? What does the frame contain?

- Space for return value and arguments
- Return address: Where do we go after we're done?

What do we need to know? What does the frame contain?

- Space for return value and arguments
- Return address: Where do we go after we're done?
- Space for local variables



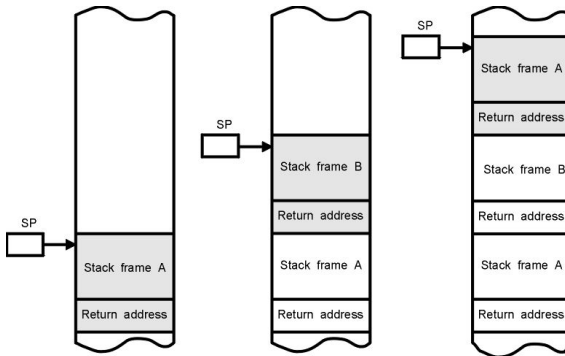
What do we need to know? What does the frame contain?

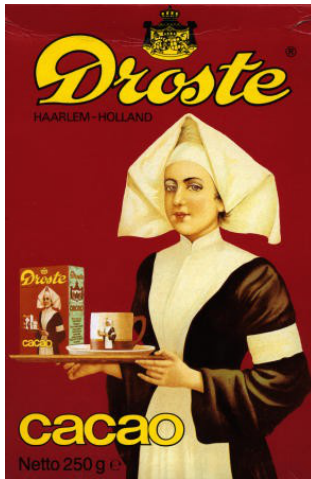
- Space for return value and arguments
- Return address: Where do we go after we're done?
- Space for local variables

**Disclaimer:** Implementation specific

What happens every time we call a function?

We create a new stack frame!





```
public void fact(int n) {  
    if (n == 0) return 1;  
    return n*fact(n - 1);  
}
```

fact(5)

5\*fact(4)

5\*4\*fact(3)

5\*4\*3\*fact(2)

5\*4\*3\*2\*fact(1)

5\*4\*3\*2\*1\*fact(0)

5\*4\*3\*2\*1\*1

5\*4\*3\*2\*1

5\*4\*3\*2

5\*4\*6

5\*24

120

```
public void fact(int n) { return fact(1, n); }  
  
private void fact(int acc, int n) {  
    if (n == 0) return acc;  
    return fact(n*acc, n - 1);  
}
```

```
fact(5)  
fact(1, 5)  
fact(5, 4)  
fact(20, 3)  
fact(60, 2)  
fact(120, 1)  
fact(120, 0)  
120
```

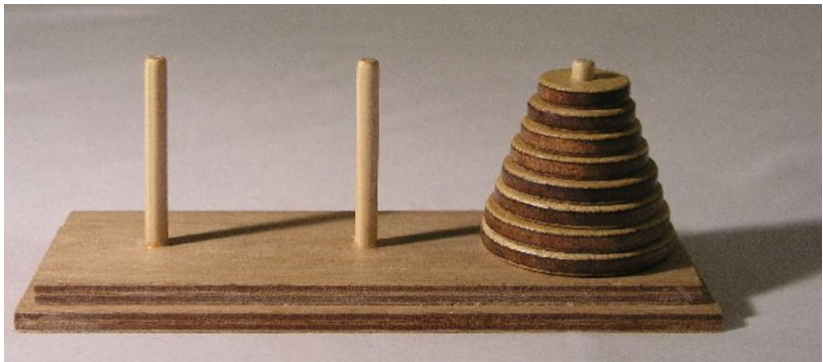
```
public long fibonacci(int n) {  
    if (n <= 1) return n;  
    else return fibonacci(n-1) + fibonacci(n-2);  
}
```

1. How many stack frames do we create with `fibonacci(2)`?
2. How many stack frames do we create with `fibonacci(5)`?
3. How many stack frames do we create with `fibonacci(10)`?
4. What is the general formula for how many stack frames the `fibonacci` function creates?

In a new Java project:

1. Implement a function for factorial using BigInteger  
`public static BigInteger factorial(BigInteger i);`
2. Run factorial with `100_000` as input. What happens?
3. Try to run it with `10`. Better now?
4. In Run -> Set Project Configuration -> Customize...  
-> VM Options write `'-Xss20m'`
5. Run it with `100_000` as input. What happened?

1. Only one disk can be moved at a time.
2. Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack i.e. a disk can only be moved if it is the uppermost disk on a stack.
3. No disk may be placed on top of a smaller disk.





With a list of the `Path` type we made last Tuesday:

Make a recursive function that reverses the order of list elements.