CS 149

Professor: Kevin Molloy

(adapted from slides originally developed by Alvin Chao)

Clarity With If-Else

Imagine we are working on a game application that requires us to determine when the player has won. Players win when their score *exceeds* 100 points. Here are five possible implementations (assume that `win` is declared as a boolean variable.)

```
// A
if (points > 100) {
   win = true;
} else if (points < 100){
   win = false;
}
```

```
// B
if (points > 100) {
   win = true;
} else if (points <= 100){
   win = false;
}
```

```
// C
if (points > 100) {
   win = true;
} else {
   win = false;
}
```

```
// D
if (points > 100) {
   win = true;
}
if (points < 100){
   win = false;
}
```

```
// E
win = points > 100;
```

Calculating Factorials

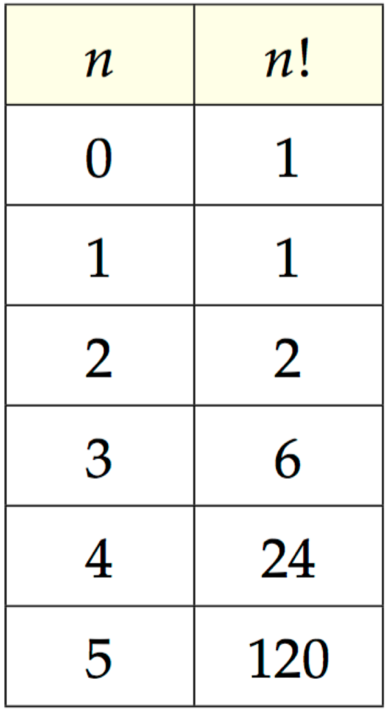
"In mathematics, the *factorial* of a non-negative integer *n*, denoted by *n*!, is the product of all positive integers less than or equal to *n*. For example, 5! = 5 × 4 × 3 × 2 × 1 = 120."

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

1. Consider how to calculate 4! = 24.

   a) Write out all the numbers that need to be multiplied:
   4! =

   b) Rewrite the expression using 3! instead of 3 × 2 × 1:
   4! =



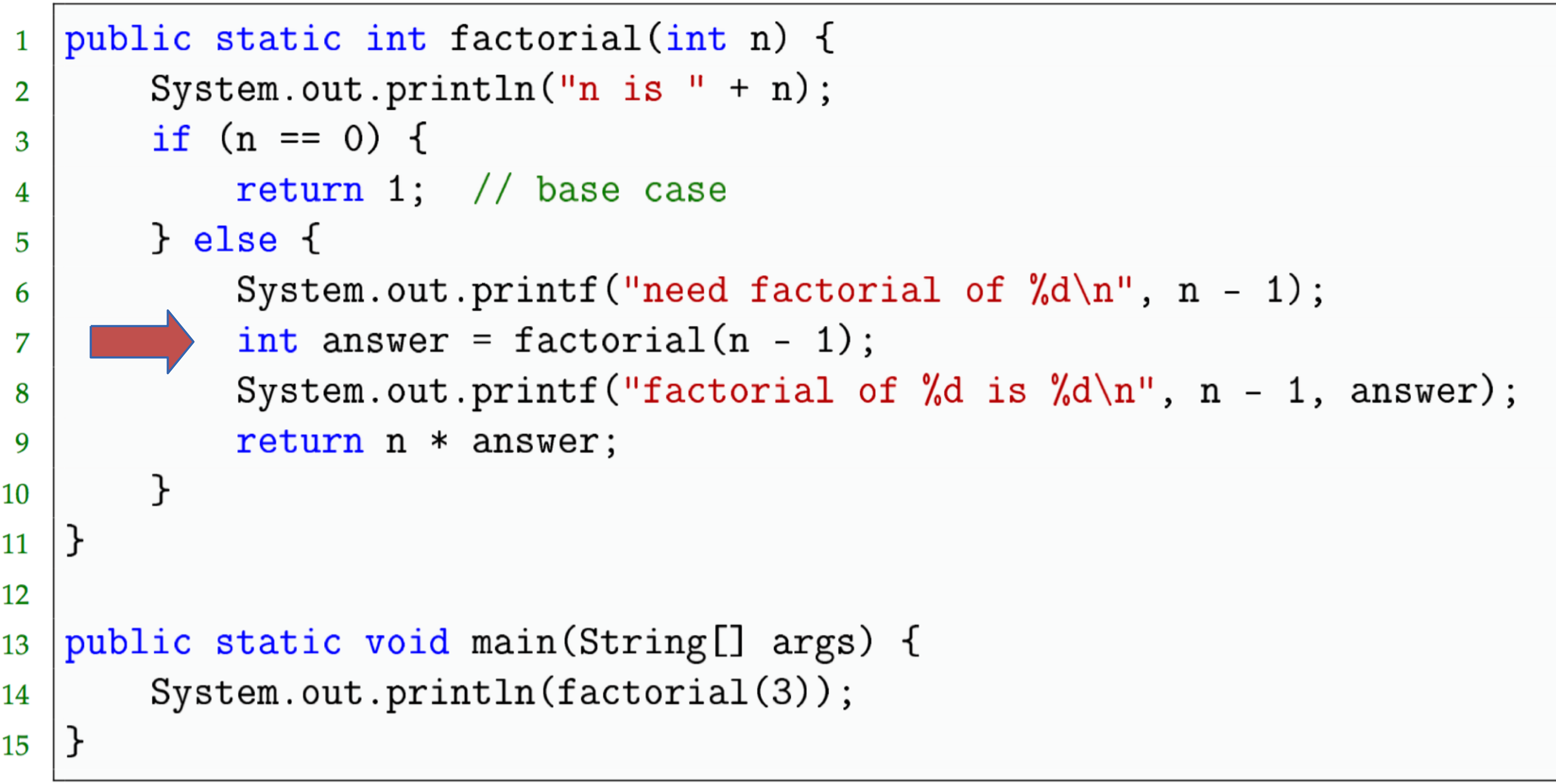
n	$n!$
0	1
1	1
2	2
3	6
4	24
5	120

Recursive Approach

1. Write an expression similar to before showing how each factorial can be calculated in terms of a simpler factorial.
   a) 3! =
   b) 2! =
   c) *n*! =
2. What is the value of 0! based on the model? Does it make sense to define 0! in terms of a simpler factorial? Why or why not?

If we repeatedly break down a problem into smaller versions of itself, we eventually reach a basic problem that can't be broken down any further. Such a problem, like 0!*, is referred to as the* ***base case****.*

Recursion Trace



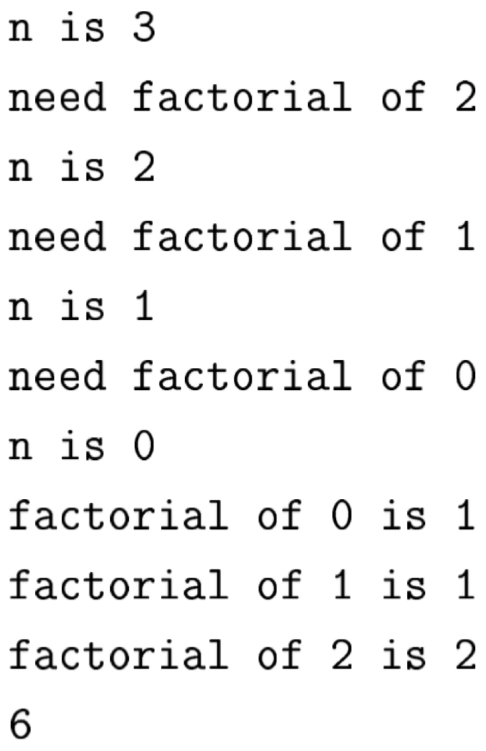
```java
public static int factorial(int n) {
    System.out.println("n is " + n);
    if (n == 0) {
        return 1;  // base case
    } else {
        System.out.printf("need factorial of %d\n", n - 1);
        int answer = factorial(n - 1);
        System.out.printf("factorial of %d is %d\n", n - 1, answer);
        return n * answer;
    }
}

public static void main(String[] args) {
    System.out.println(factorial(3));
}
```

Recursion - Factorials

A method that invokes itself is called **recursive**. What two steps were necessary to define factorial? How were they implemented in Java?

- How many distinct method calls would be made to factorial to compute the factorial of 3? Lets review the value of the parameter *n* for each of these separate calls.
- Here is the complete output from the program in #5. Identify which distinct method call printed each line. In other words, which lines were printed by factorial(3), which lines were printed by factorial(2), and so on.



```
n is 3
need factorial of 2
n is 2
need factorial of 1
n is 1
need factorial of 0
n is 0
factorial of 0 is 1
factorial of 1 is 1
factorial of 2 is 2
6
```

- **Acknowledgements**

Parts of this activity are based on materials developed by Chris Mayfield and Nathan Sprague.

</end>