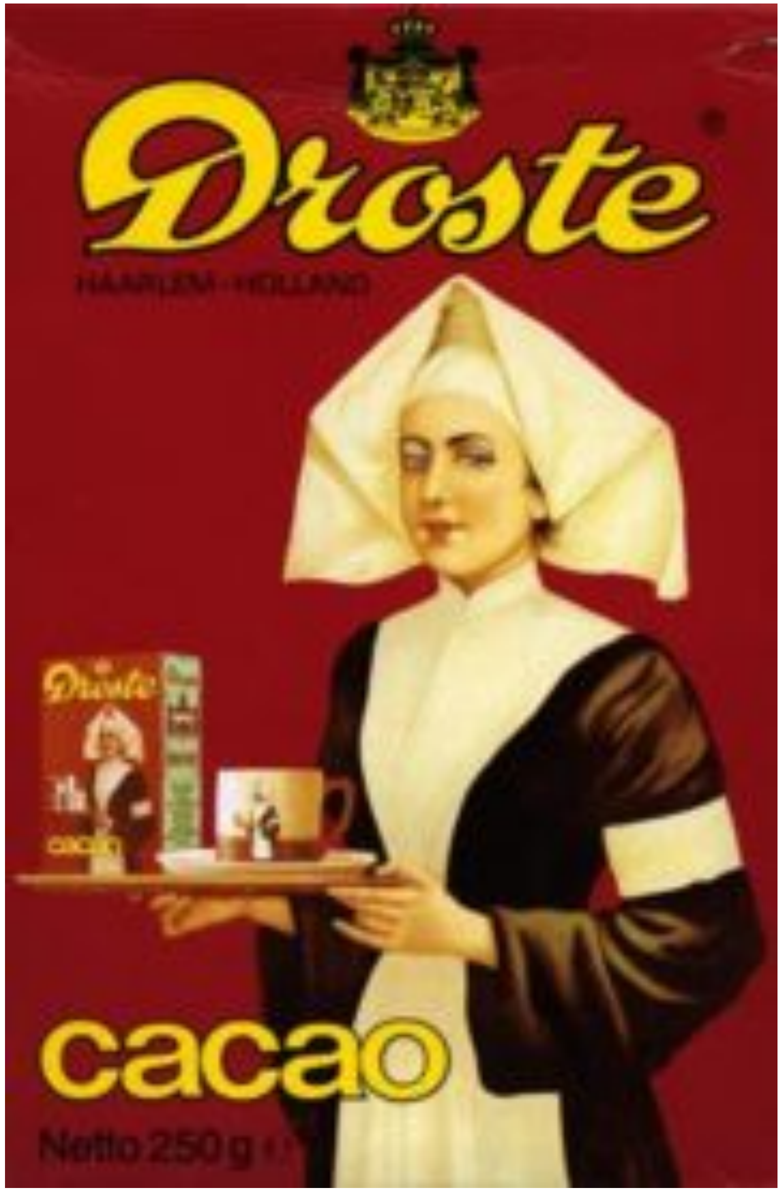


CSSE 220

Recursion

Import *Recursion* project from the repo

Announcements

- The next 4 class days:
  - A new way to think: **Recursion**
  - A new way to break up and re-use code: **Interfaces**
    - Making interactive apps requires this

Recursion

- A solution technique where the same computation **occurs repeatedly** as the problem is solved

recurs

- Examples:
  - Sierpinski Triangle: https://en.wikipedia.org/wiki/Sierpinski_triangle
  - Towers of Hanoi: http://www.mathsisfun.com/games/towerofhanoi.html or search for Towers of Hanoi

An example – Triangle Numbers

If each red block has area 1, what is the ***area*** **A(n)** of the Triangle whose *width* is n?

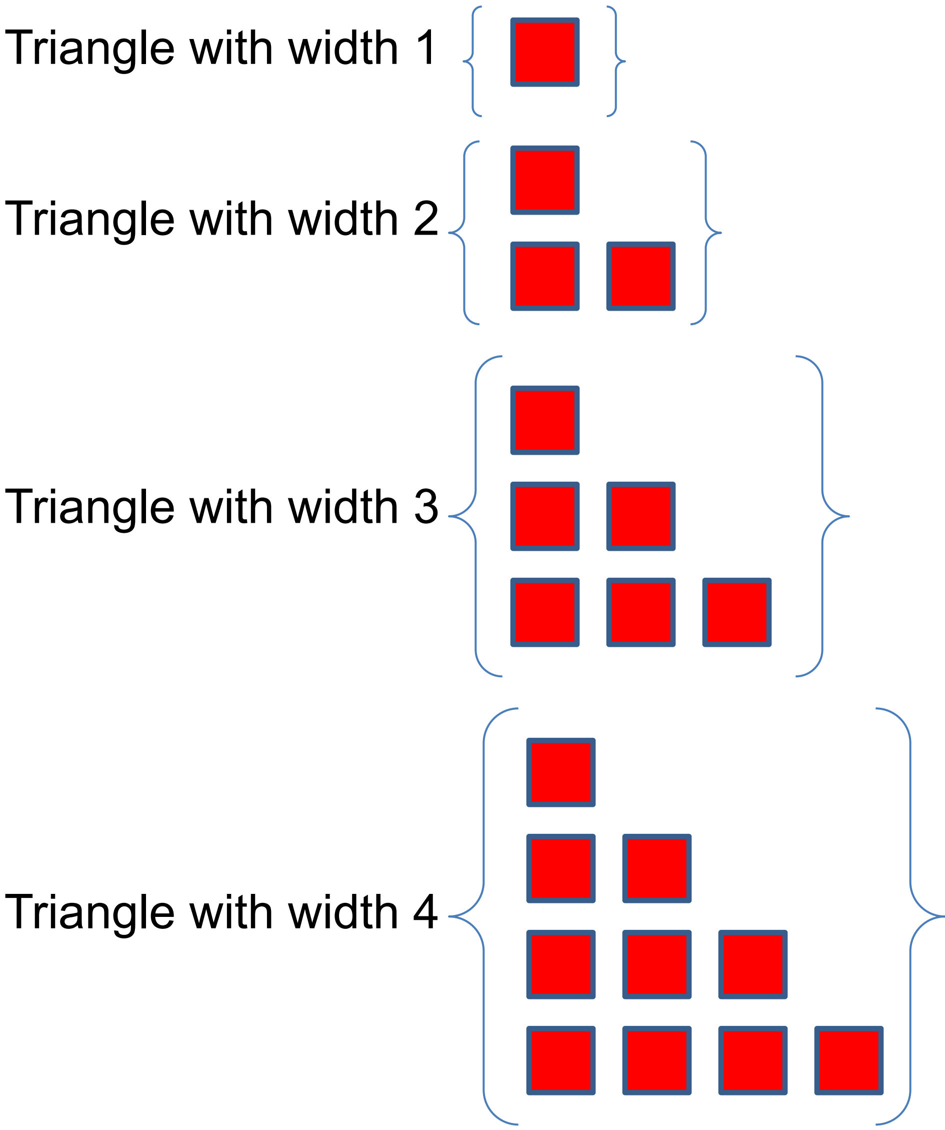
- Answer:

  A(n) = n + A(n-1)

The above holds for which *n* ?

What is the answer for other *n* ?

- Answer: The recursive equation holds for
  n >= 0
  When n = 0, the area is 0.
- $A(n) = \sum_{k=0}^{n} k$



Key Rules to Using Recursion

- Always have a **base case**
  - *At the base case*: the problem is simple (or small) enough that we can just return a precomputed answer
  - So we do not need to make another recursive call

- When *not at the base case*:
  - We must guarantee that we make the recursive call with a smaller version of the problem
  - This is called our progress metric

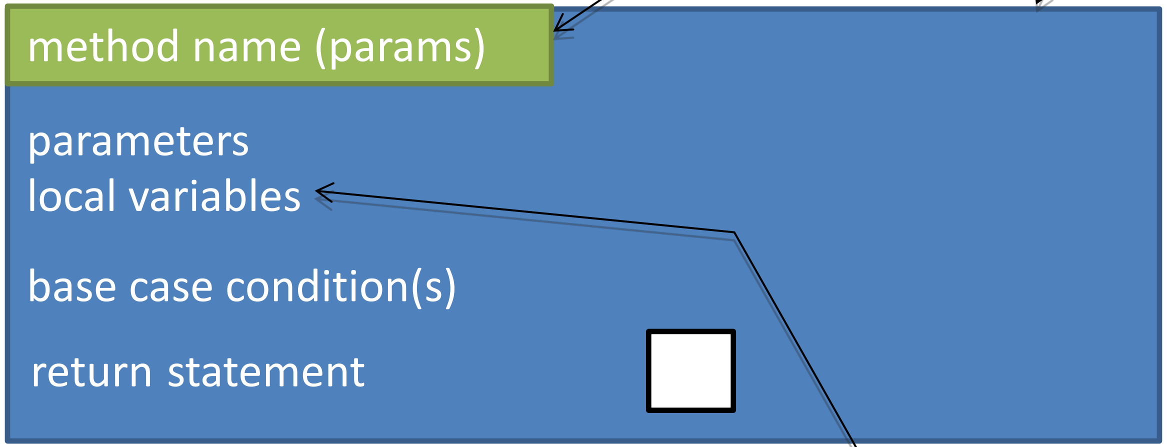
- **You gotta believe**
  - Trust in the recursive solution
  - Just consider one step at a time

Diagramming a Recursive Operation
The Setup Steps

Always do the following 4 steps each time a recursive operation is invoked

1. Draw this *activation box* when method starts

2. Fill in method's name and formal parameters



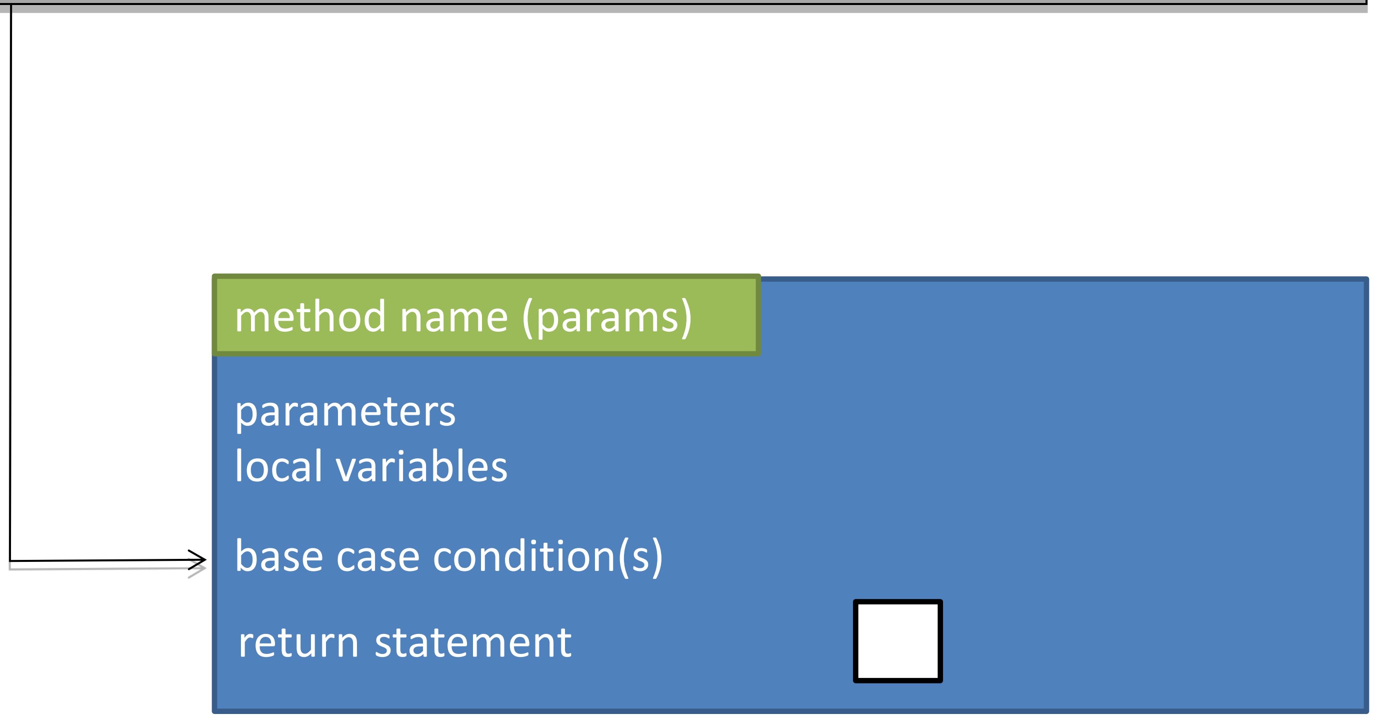
3. List every parameter and its incoming value.

4. List every locally declared variable **but do no write their values yet**

Q1-Q2

Diagramming a Recursive Operation
The Call Steps – Eval Base Case

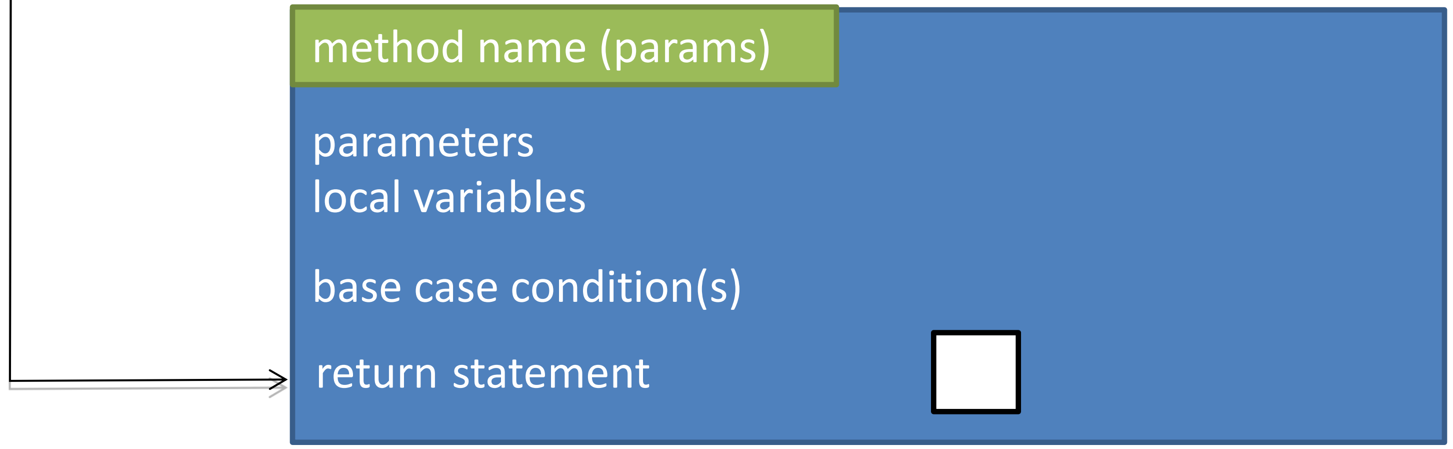
5. Evaluate base case expression:
    5.1 Write down the code that is evaluated
    5.2 Write “yes” if at base case, write “no” if not at base case



Q1-Q2

Diagramming a Recursive Operation
The Call Steps – Eval Base Case

6. Handle base case that evaluates to “yes” in 5.2
    - 6.1 Look at base case code and write the value to be returned in the white square
    - 6.2 Draw an arrow from white square back to *return statement* of previous activation, or back to initial client call if this was the first activation

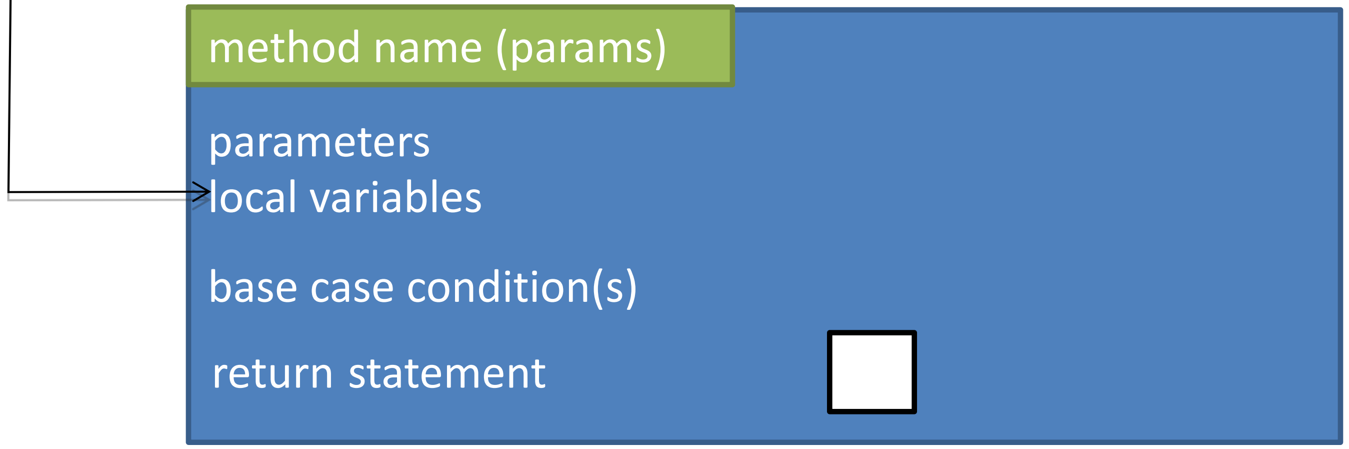


Q1-Q2

Diagramming a Recursive Operation
The Call Steps – Non-Base Case

7. Handle non-base case ("no" in 5.2) by stepping through non-base case code:

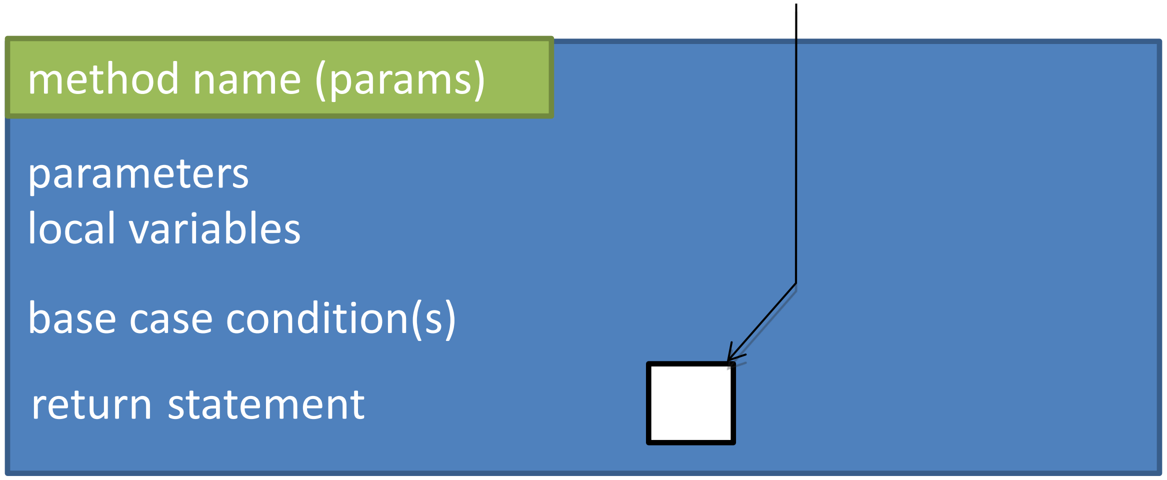
    7.1 Give local variables their current values based on code



Q1-Q2

Diagramming a Recursive Operation *The Call Steps – Non-Base Case*

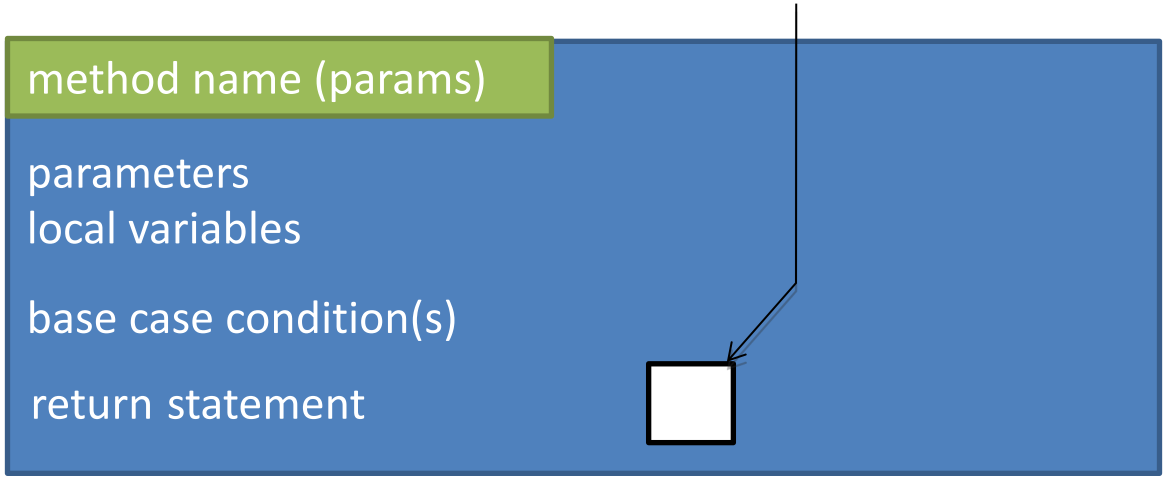
8. Do setup for next recursive call
    - 8.1 Draw the next *activation box* for the next activation (Steps 1 through 4)
    - 8.2 Draw an arrow from small white square to this new activation box
    - 8.3 Next to this arrow write "yes" if progress metric is true.
    It is true if problem size has been reduced prior to the call.



Q1-Q2

Diagramming a Recursive Operation
The Call Steps – Non-Base Case

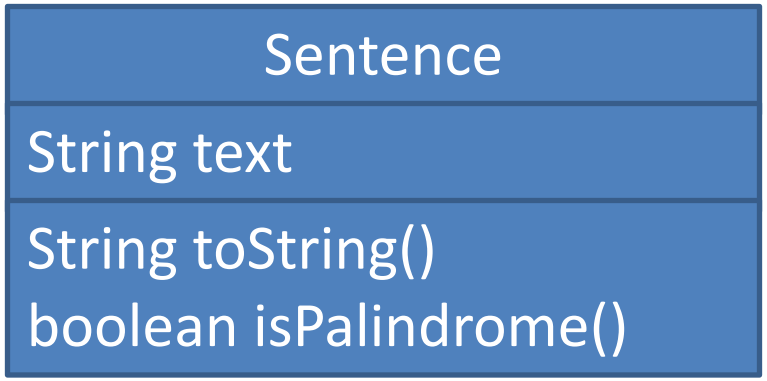
9. Make recursive call
    - 9.1 Go to next *activation box* and do Call Steps 5 through 9
    - 9.2 Take value returned by the *next activation* box and use it to determine value to return for this current activation, write this return value in the white square
    - 9.3 Draw an arrow from white square back to return statement of previous activation, or back to initial client call if this was the first activation



Q1-Q2

Programming Problem

- Add a recursive method to Sentence for computing whether Sentence is a palindrome



Practice Practice Practice

- Head to http://codingbat.com/java/Recursion-1 and solve 5 problems. I personally like bunnyEars, bunnyEars2, count7, fibonacci, and noX
- Get help from me if you get stuck
- Then take a look at the recursion homework