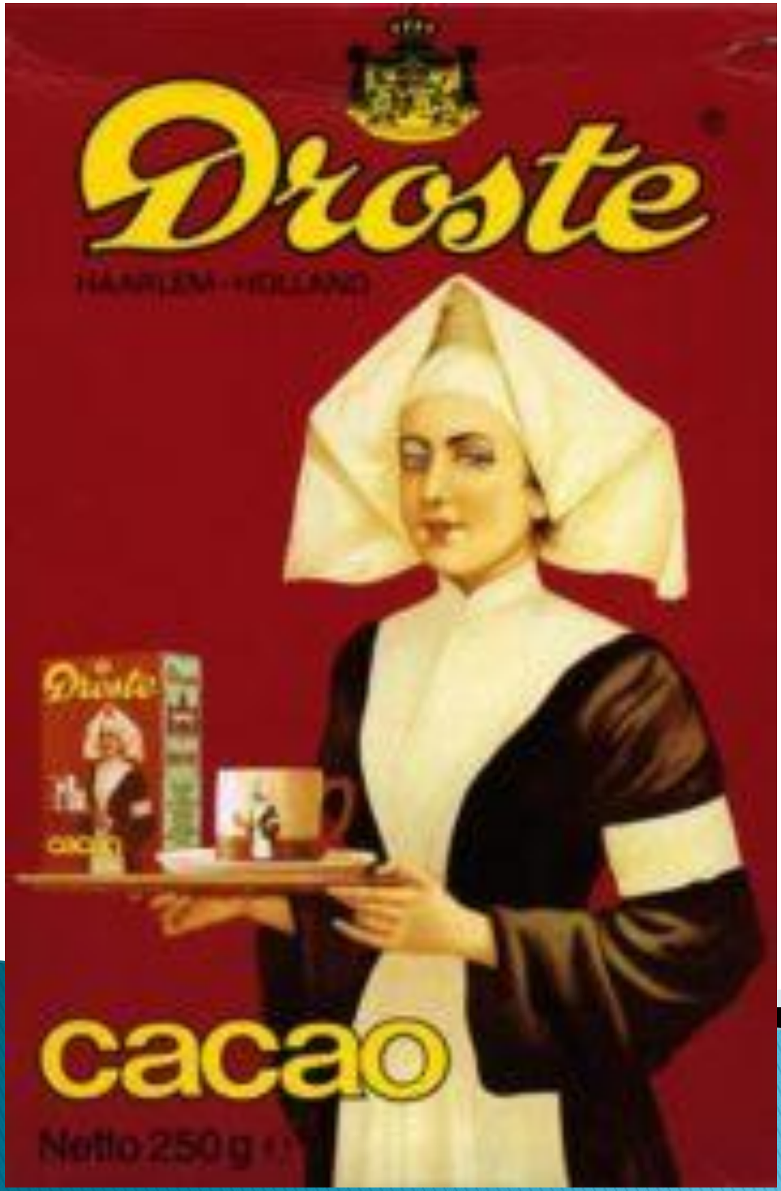


CSSE 220
Day 21

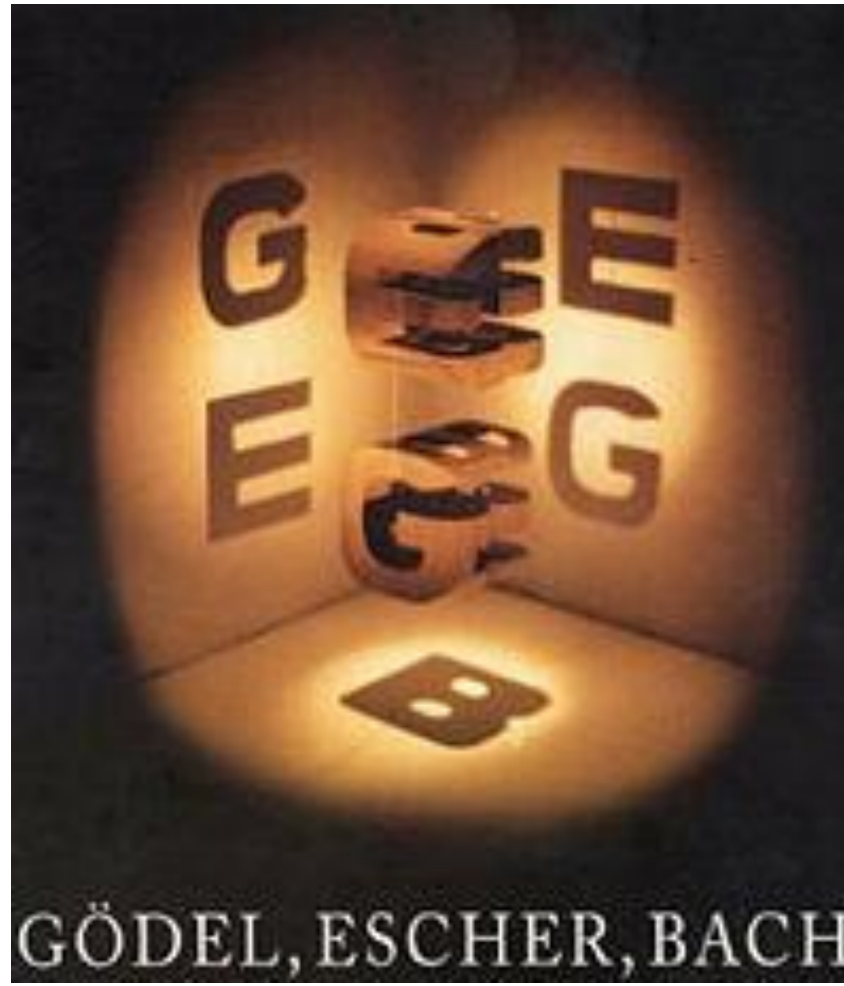
Recursion

Checkout *Recursion* project from SVN

Questions?

Gödel, Escher, Bach

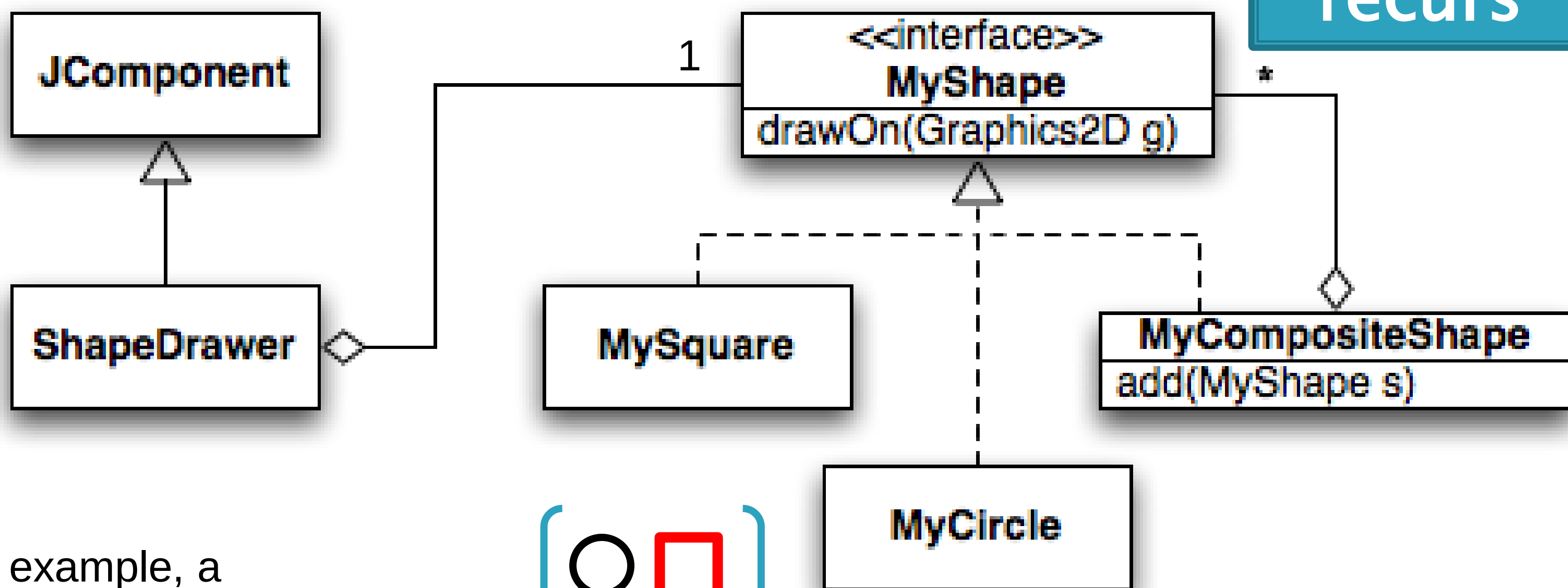
- By Douglas Hofstadter
- Argues that intelligence arises (in part) because of **our ability to think about thinking**



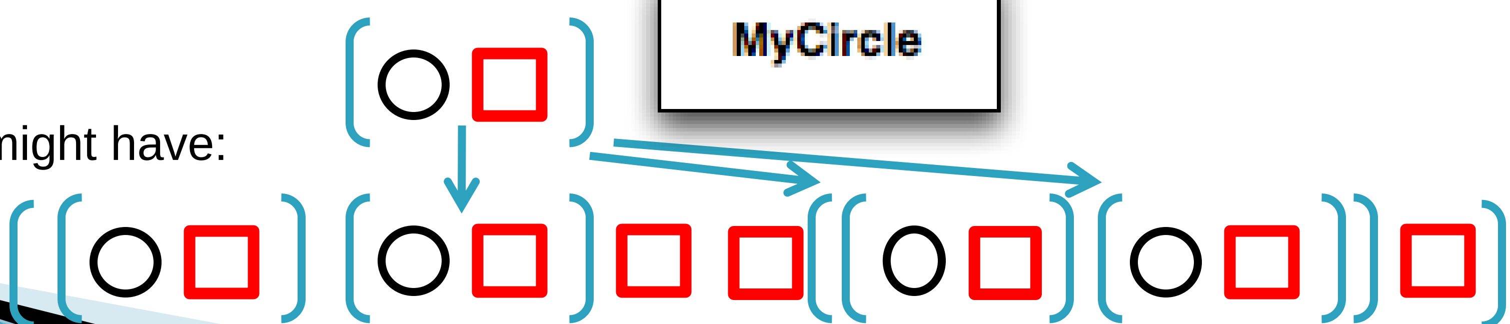
Recursion

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

recurs



For example, a ShapeDrawer might have:

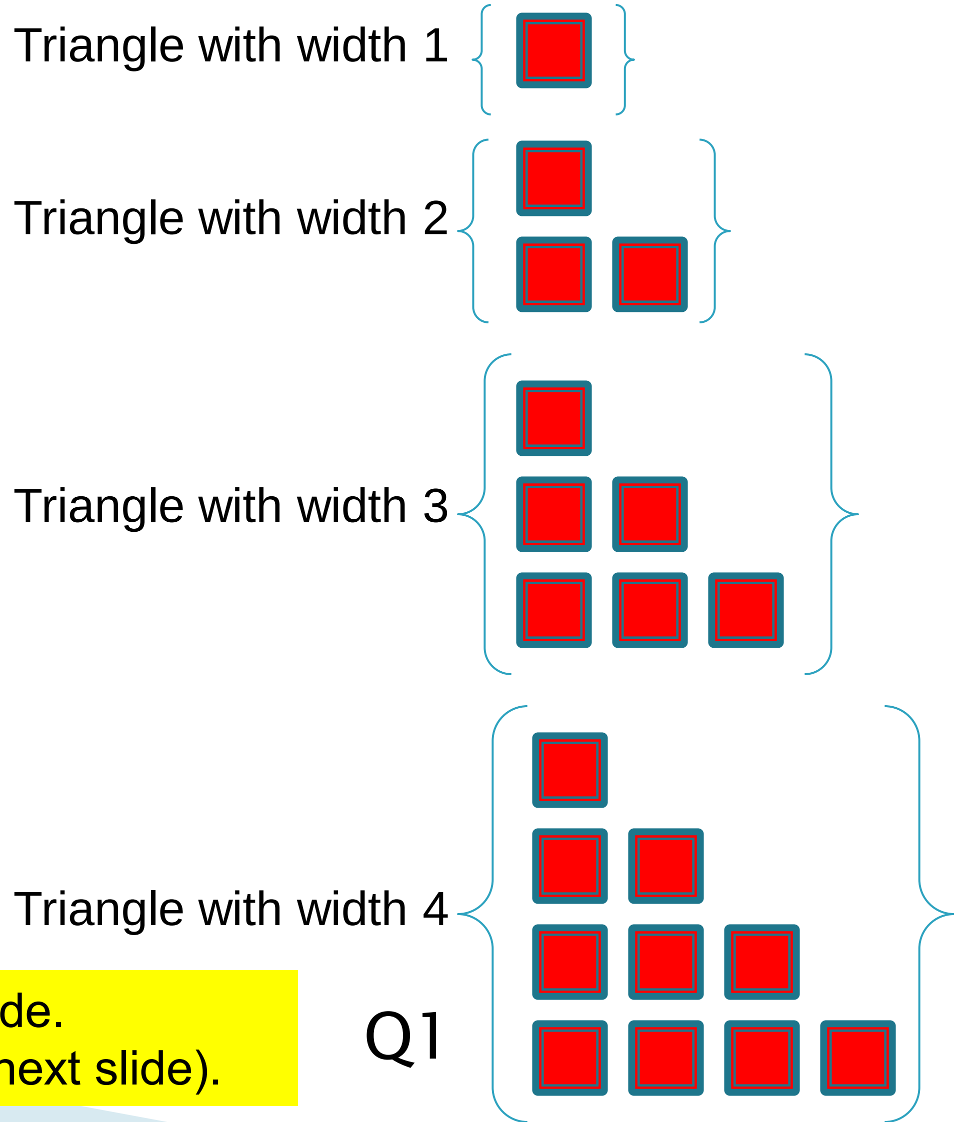


An example – Triangle Numbers

- 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 what *n* ? What is the answer for other *n* ?
  - Answer: The recursive equation holds for $n > 1$.

    For $n = 1$, the area is 1.



Let's see how this translates naturally to code.
Then let's trace the execution of the code (next slide).

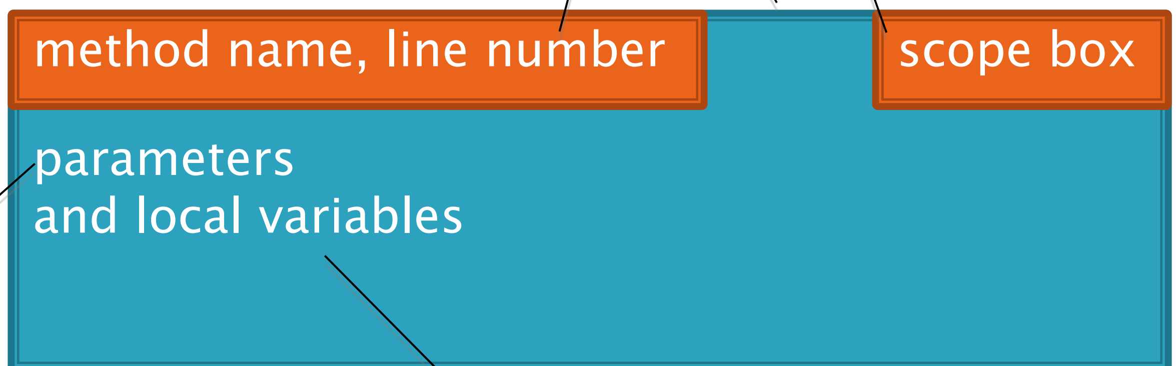
Q1

Frames for Tracing Recursive Code

1. Draw box when method starts

2. Fill in name and first line no.

3. Write class name (for static method) or draw reference to object (for non-static method)



4. List every parameter and its argument value.

5. List every local variable declared in the method, **but no values yet**

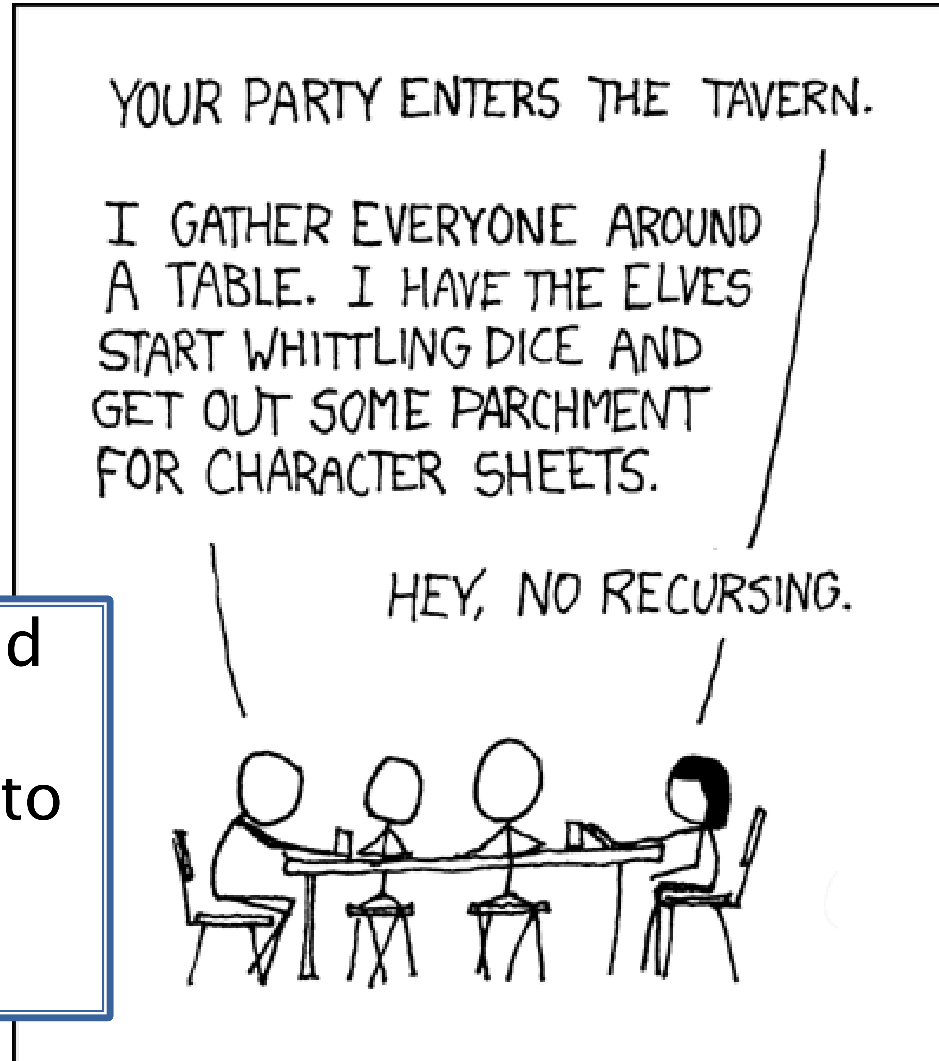
6. Step through the method, update the line number and variable values, draw new frame for new calls

7. “Erase” the frame when the method is done.

Thanks for David Gries for this technique

Q2-9

Tabletop Roleplaying



I may have also tossed one of a pair of teleportation rings into the ocean with interesting results.

Key Rules to Using Recursion

- Always have a **base case** that **doesn't recurse**
- Make sure recursive case always makes **progress**, by **solving a smaller problem**
- **You gotta believe**
  - Trust in the recursive solution
  - Just consider one step at a time

Programming Problem

- Add a *recursive* method to Sentence for computing whether Sentence is a palindrome
  - A *palindrome* is a String that is the same backwards as forwards
    - We will ignore punctuation, spaces, and case.
  - Key idea: use the definition of `isPalindrome()` in defining `isPalindrome()`. How can we make progress to a smaller problem?
  - Here, **`x.isPalindrome()`** iff **`___.isPalindrome()`** ________________?
  - **`x.isPalindrome()`** iff **`xMinusFirstAndLastLetter.isPalindrome()`** and ________________?

Examples of palindromes from http://www.fun-with-words.com/palin_example.html

Never odd or even
Murder for a jar of red rum
May a moody baby doom a yam?
Go hang a salami; I'm a lasagna hog!
Oozy rat in a sanitary zoo
Do geese see God?

Sentence
String text
String toString() boolean equals() boolean isPalindrome

Don't worry about punctuation, spaces and case at this point of your thinking.

`x.isPalindrome()` iff **`xMinusFirstAndLastLetter.isPalindrome()`** **and** first letter equals last letter

Q10

Recursive Helpers

- Our isPalindrome() makes lots of new Sentence objects

- We can make it better with a “recursive helper method”
  - Many recursive problems require a helper method

```
public boolean isPalindrome() {
    return isPalindrome(0,  this.text.length() - 1);
}
```

Position of first letter of the remaining String to check

Position of last letter of the remaining String to check

Homework part 1

- Reverse a string…recursively!
- A recursive helper can make this really short!

Another Definition of Recursion

- "If you already know what recursion is, just remember the answer. Otherwise, find someone who is standing closer to Douglas Hofstadter than you are; then ask him or her what recursion is."

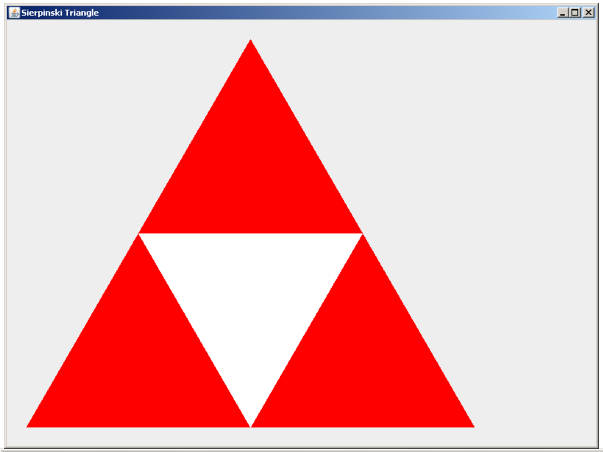
  —Andrew Plotkin

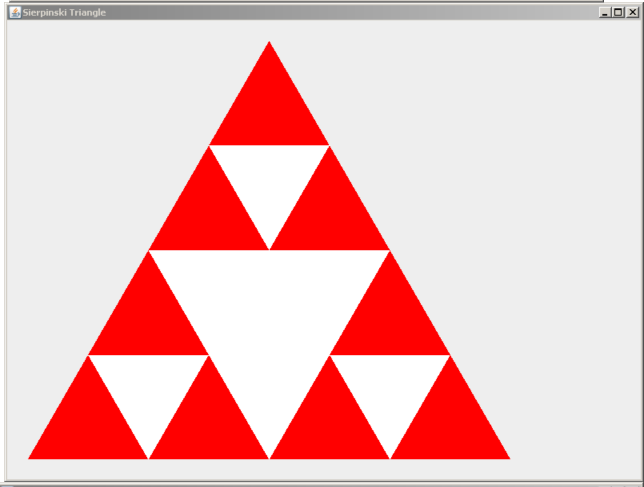
HW, part 2: Sierpinski

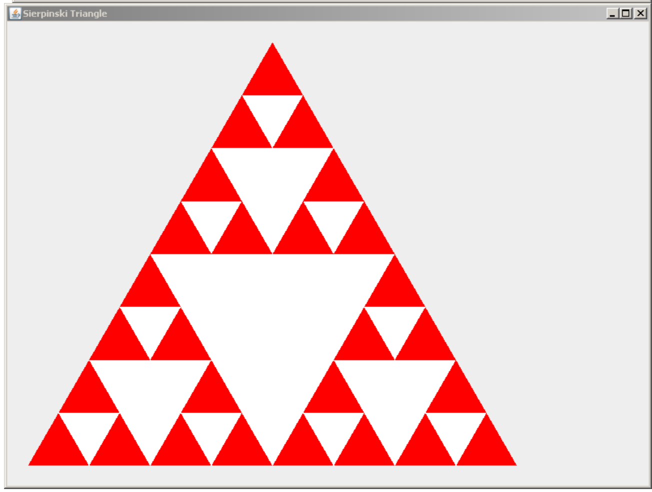
```java
private void drawSierpinski(Graphics2D g,
                double left, double bottom,
                double base) {
  // TODO Don't forget your base case

  // Draws the first equilateral triangle
  // called for by the algorithm.
  Point2D p1 = new Point2D.Double(
                      left, bottom);
  Point2D p2 = new Point2D.Double(
                     left + base, bottom);
  Point2D p3 = new Point2D.Double(
                      left + base / 2.0,
   bottom - base * HEIGHT_TO_BASE_RATIO);
  Shape triangle = makeTriangle(
                              p1, p2, p3);
  g.setColor(Color.RED);
  g.fill(triangle);

  // TODO Implement rest of this method.
}
```







Recursive Functions

- Factorial:

Base Case

$$n! = \begin{cases} 1 & \text{if } n \leq 1 \\ n * (n-1)! & \text{otherwise} \end{cases}$$

Recursive step

- Ackermann function:

$$A(m,n) = \begin{cases} n+1 & \text{if } m = 0 \\ A(m-1,1) & \text{if } m > 0 \text{ and } n = 0 \\ A(m-1, A(m,n-1)) & \text{otherwise} \end{cases}$$

Q11–14

Exam 2 is next Friday morning. Major topics are:

- UML class diagrams and how to implement them
- event-driven programming
- GUI programming
- polymorphism
- interfaces
- inheritance
- recursion

Work Time

Work on VectorGraphics with your team

- Cycle 1 code and status report and Cycle 2 user stories are due Tuesday.

- Or work on recursion problems, due Thursday