

# CSSE 220

## Day 20

Recursion

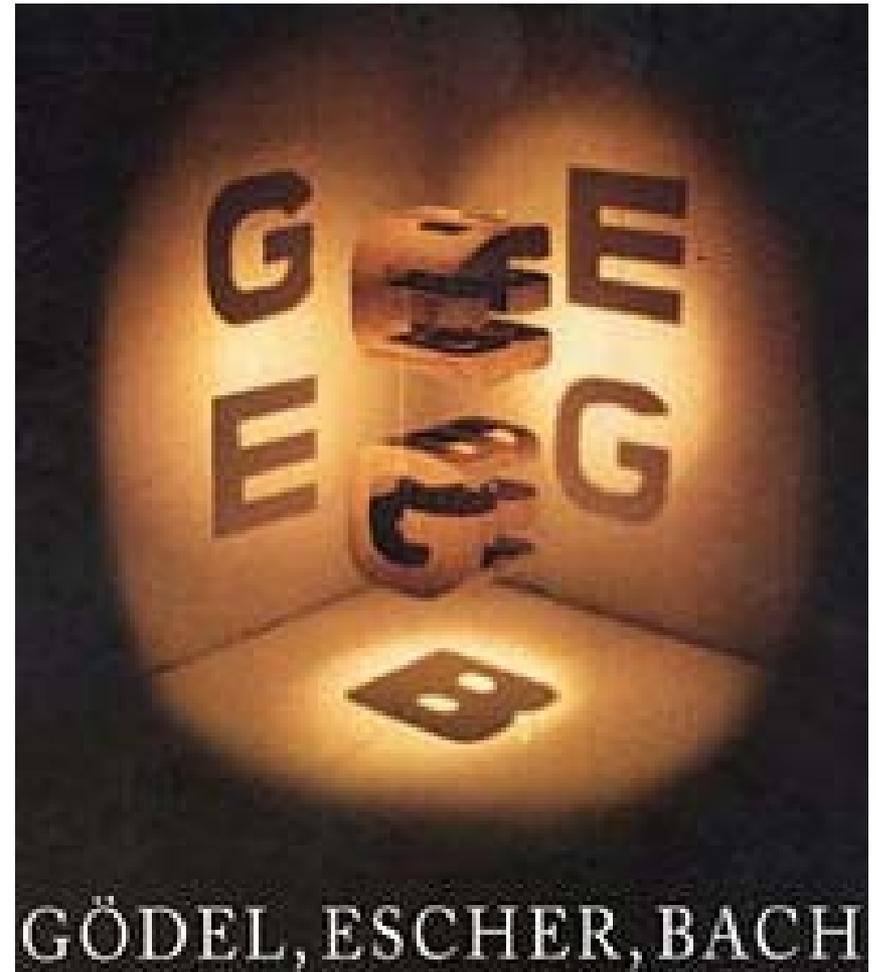
Checkout *Recursion* project from SVN

# Questions?

Don't forget:  
Exam 2  
Thursday, Feb 5

# Gödel, Escher, Bach

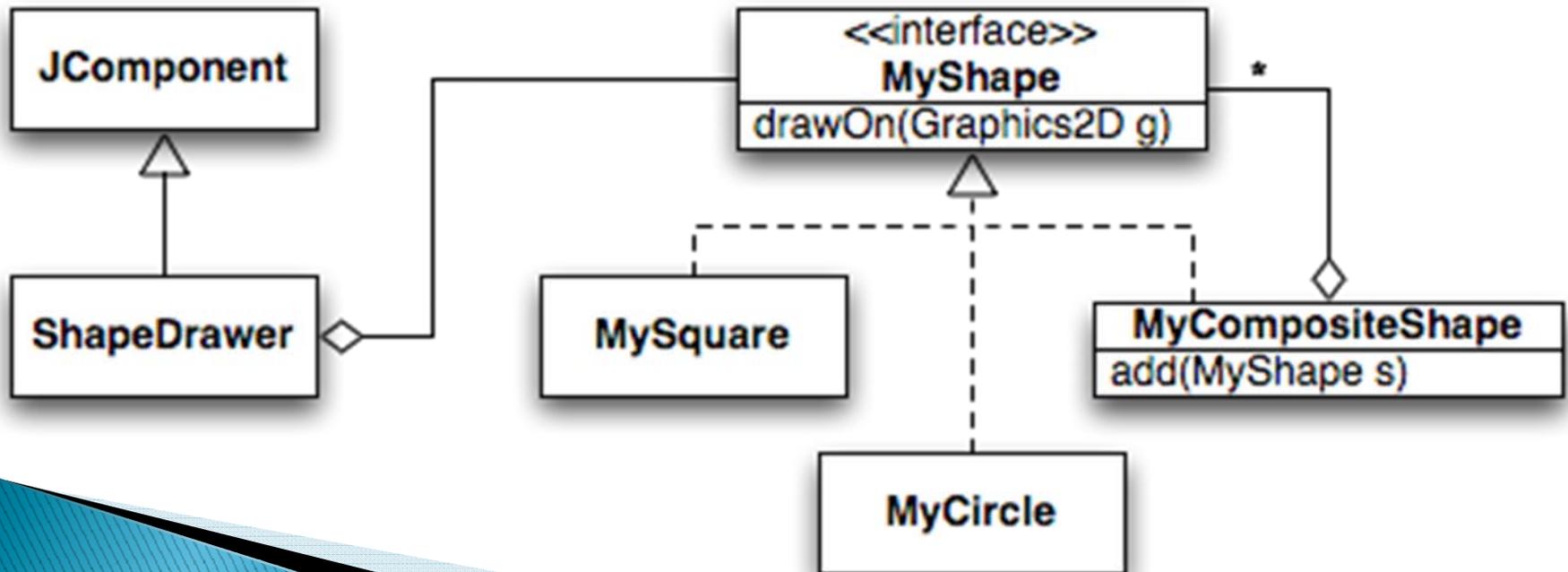
- ▶ By Douglas Hofstadter
- ▶ Argues that intelligence arises (in part) because of our ability to think about thinking



# Recursion

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

recurs



# 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

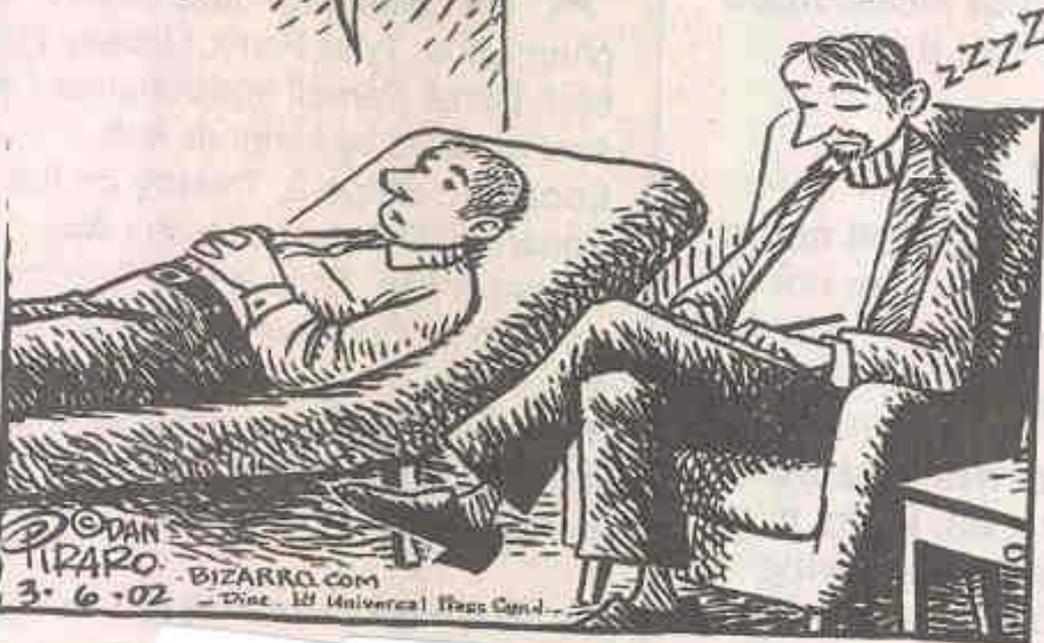
7. "Erase" the frame when the method is done.

Thanks for David Gries for this technique

# Too much Recursion!

Bizarro

I have this recurring dream that I'm lying here telling you about a recurring dream about lying here telling you about a recurring dream about...



© DAN TIRARO  
3.6.02 BIZARRO.COM  
- Div. of Universal Press Corp. -

# Programming Problem

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

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

# Recursive Functions

- ▶ Factorial:

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

Base Case

Recursive step

- ▶ Ackermann function:

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

# Recursive Helpers

- ▶ Our `isPalindrome()` makes lots of new Sentence objects
- ▶ We can make it better with a “recursive helper method”
- ▶ 

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

# 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

# Homework 20

- ▶ Reading: 13.4–13.5, ANGEL quiz
- ▶ Two small individual programming problems
  - Sentence reverse()
  - Sierpinski Triangle
- ▶ Team work on Vector Graphics

# 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

# Vector Graphics Work Time

- »» Should have completed Status Report for Cycle 1 and listed User Stories for Cycle 2