

COS 126 Precept

Will Clarkson
February 26, 2008

Agenda

- ▶ Function/Method Review
- ▶ Recursion
- ▶ Recursive Graphics
- ▶ Sierpinski Assignment
 - ▶ Math Review
 - ▶ Tips
- ▶ Return NBody



Methods

Recall the syntax of a function/method:

```
public static <Return Type> FunctionName(Arguments..)  
{  
    <Function Body>  
}
```



Factorial: Non-Recursive

```
/*  
long factorial(n)  
returns n! provided n > 0  
*/  
public static long factorial(int n){  
    long result = 1;  
    for(int i = n; i > 1; i--)  
    {  
        result *= i;  
    }  
    return result;  
}
```



Recursion

- ▶ Think of as a function which calls itself
- ▶ Breaks up problems into smaller sub-problems
 - ▶ GCD(a,b), Factorial, Sierpinski, H-Tree
- ▶ **Required Parts:**
 - ▶ Base Case
 - ▶ Reduction Step
 - ▶ Break problem into sub-components
 - ▶ Calls itself on smaller problems 1 or more times
 - ▶ Compute and return final result



Recursive Example

- ▶ $5! = 5*4*3*2*1$
- ▶ $7! = 7*6*5*4*3*2*1$
- ▶ In General
 - ▶ $N! = N*(N-1)*(N-2)*...*2*1$
- ▶ Can break up into sub-problems
 - ▶ $N! = N (N-1)!$



Recursive Functions

- ▶ **A function which calls itself**
 - ▶ Processes smaller sub-problems
 - ▶ Must have Base Case
 - ▶ Otherwise we continue forever
- ▶ **Exercise: Factorial.java (Recursive)**



Trace of Factorial(5)

Factorial(5)

5*Factorial(4)

4*Factorial(3)

3* Factorial(2)

2* Factorial(1)

return 1

return 2 (1)

return 3 (2)

return 4 (6)

return 5*(24)

//Returns N! provided N > 0

```
public static long factorial(int N)
{
    if (N == 1)
        return 1;
    return N * factorial(N-1);
}
```



Must have Base Case

- ▶ **Otherwise we repeat forever!**
 - ▶ Get an Error: `java.lang.StackOverflowError`
- ▶ **Problem must get smaller**
 - ▶ There is a limit to what computer will allow
- ▶ **See Back of Sheet**



Recursive Graphics

- ▶ See StdDraw API (Section 1.5 on Booksite)
- ▶ `StdDraw.setPenColor(StdDraw.RED)`
 - ▶ See Booksite 1.5
- ▶ `StdDraw.fillPolygon(double [] x, double [] y)`
 - ▶ Draws a filled polygons with vertices at:
 - ▶ $(x[0],y[0]),(x[1],y[1]),\dots,(x[n-1],y[n-1])$
- ▶ `StdDraw.show(x)`
 - ▶ Where x is the # of milliseconds to wait



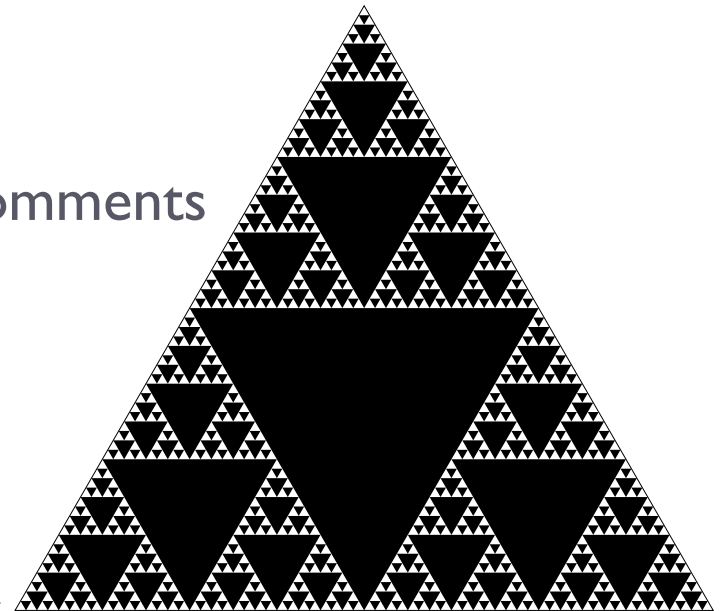
Htree

- ▶ Lets trace it
- ▶ This is style of modularity we want for Sierpinski
 - ▶ 1 Main Function
 - ▶ 1 Function to Draw H-Tree at a given position
 - ▶ 1 Function to do recursion



Sierpinski.java

- ▶ Draw triangles that are symmetric at all scales
- ▶ Equilateral Triangles
 - ▶ Tools
 - ▶ Pythagorean Theorem
 - ▶ <http://www.clarku.edu/~djoyce/trig/tangents.html>
- ▶ Hardwired Constants
 - ▶ For this assignment they are fine
 - ▶ Make sure you explain them in your comments



Art.java

- ▶ **Must be sufficiently different than H-Tree and Sierpinski**
 - ▶ See examples on website
- ▶ **Grading**
 - ▶ 12 points for Sierpinski
 - ▶ 6 points for Art.java
 - ▶ 2 points for readme.txt



Return NBody

