Recursion
(Graphics in Java)

Lecture 35
News

- Homework 3 is due this Friday (December 4)

- Final exam: Dec 11, 8:30 am, OSBO A
  - Length: 2:30

- Extra office hour: Wednesday, Dec 9
  - 11:00 – noon
  - ICICS/CS x639
Reading Assignments

- Reading for this week: inheritance
  - Edition 3: Ch 10
  - Edition 2: Ch 13

- Optional reading: graphics, UI
  - Edition 3: Ch 2.11, 2.12, 9.5-9.8, 10.9, 10.10
  - Edition 2: Ch 5.1, 5.2, 11.5, 12.1-12.3
  - (NOT part of the final exam…)
Recap: Java Graphics

- Java systems provide lots of classes for user interfaces and graphics
  - These are not strictly part of the language, but they are available in almost all Java implementations
  - We will learn how to use some of these classes, and in the process practice inheritance
Recap: Creating a Window in Java

- Java JFrame class:
  - Part of the Swing package for user interfaces
  - Can create a window, name it, size it, etc.
Recap: Drawing Stuff

- How do we actually show something in our window?
  - Have to add objects of type JComponent to our frame
  - JComponent is an *abstract* base class for all kinds of drawable objects
  - Have to create subclasses that extend JComponent, and implement method void paintComponent( Graphics g )
Recap: Drawing Stuff

- The parameter to the paintComponent method is an object of type Graphics
  - Actually, in our case it will be providing a reference to a Graphics2D object
  - Graphics is an abstract base class
  - Graphics2D is a subclass of Graphics
  - The classes describe a graphics context, i.e. a collection of state that is required to draw stuff (i.e. window area, color, etc...)
Recap: Drawing Stuff

- Using the Graphics2D object, we can now draw geometric primitives
  - E.g: Line2D.Double
  - Line2D.Double is a *nested* class (class within a class)…
Objectives

- Understand simple recursive function calls
- Learn about Java graphics and event loops
How do we get from our hexagon to a snowflake?

Basic concept:

- We replace each line in the hexagon with four shorter lines:
Snowflakes

From this:
Snowflakes

- Get this:
Snowflakes

- Then do the same substitution again
Snowflakes

- And again...
How do we Program This?

- This is called a recursive structure
  - A line is replaced by four lines, which are in turn replaced by four other lines and so forth
  - At some point, we stop, and just draw a single line segment for each remaining edge
Recursion

- Let’s first consider a simpler example for recursion in programming.
- The *factorial* of an integer n (written as “n!”) is given as the product of all positive integers <=n.
  - i.e. 5! = 5*4*3*2*1, 4! = 4*3*2*1

- How can we implement a method that computes the factorial of a positive number?
Factorial

- (Solution 1: use for loops)
- Solution 2: recursively break down the problem into smaller chunks:
  - $5! = 5 \times (4 \times 3 \times 2 \times 1) = 5 \times (4!)$
  - $4! = 4 \times (3 \times 2 \times 1) = 4 \times (3!)$
  - $3! = 3 \times (2 \times 1) = 3 \times (2!)$
  - $2! = 2 \times (1) = 2 \times (1!)$
  - $1! = 1$
Recursive Implementation of Factorial

- In general we can calculate $n!$ as follows:
  - If $n = 1$, then $n! = 1$
  - If $n > 1$, then $n! = n \times ((n-1)!)$

- Can we translate this into a Java program?
Back to Snowflakes…

- We can describe the snowflake recursively in a very similar fashion:
  - A snowflake has six sides of a certain level
  - A side of level 0 is just a straight line
  - A side of level n (with n>0) consists of 4 sides of level (n-1)
Recursion

- Recursion / recursive method calls are a standard technique in programming
- Everything that can be done with recursion can also be done with loops, and vice versa

BUT:
- In some applications recursion is conceptually easier than loops (e.g. Snowflake)
- Sometimes loops are easier (e.g. Factorial?)
- Sometimes it is a wash…
Recursion

- Remember loop invariants?
  - Used to prove that loops terminate and are producing the correct computation

- Correctness of recursion
  - Similar techniques can be used to prove correctness of recursive programs
  - In our case: have numbers counting down from some positive value to 0 or 1