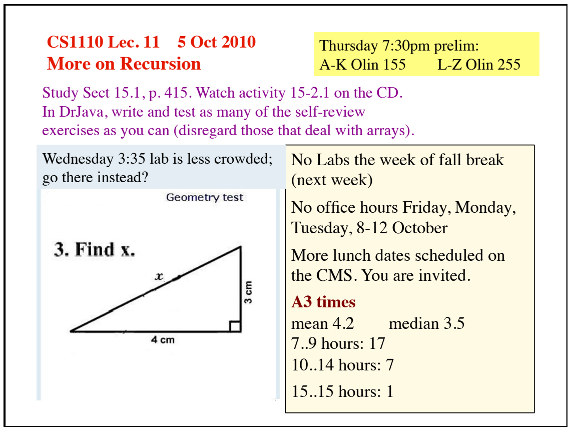


CS1110 Lec. 11 5 Oct 2010
More on Recursion

Thursday 7:30pm prelim:
A-K Olin 155 L-Z Olin 255

Study Sect 15.1, p. 415. Watch activity 15-2.1 on the CD.
In DrJava, write and test as many of the self-review exercises as you can (disregard those that deal with arrays).

Wednesday 3:35 lab is less crowded; go there instead?

Geometry test
3. Find x.
x
3 cm
4 cm

No Labs the week of fall break (next week)

No office hours Friday, Monday, Tuesday, 8-12 October

More lunch dates scheduled on the CMS. You are invited.

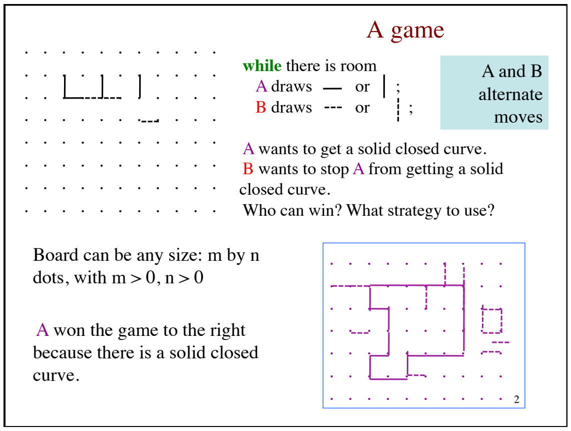
A3 times

mean 4.2 median 3.5

7..9 hours: 17

10..14 hours: 7

15..15 hours: 1



A game

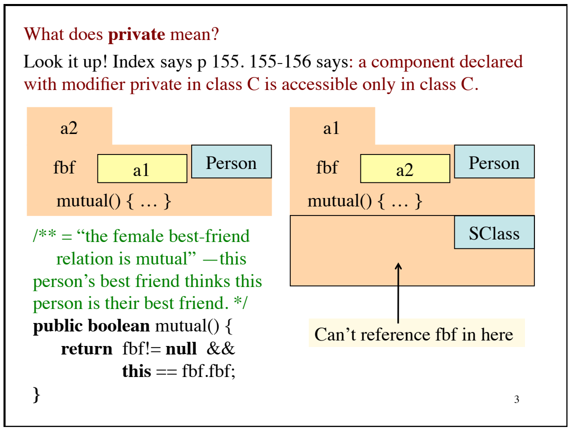
while there is room
A draws — or | ;
B draws --- or ¦ ;

A and B alternate moves

A wants to get a solid closed curve.
B wants to stop A from getting a solid closed curve.
Who can win? What strategy to use?

Board can be any size: m by n dots, with m > 0, n > 0

A won the game to the right because there is a solid closed curve.



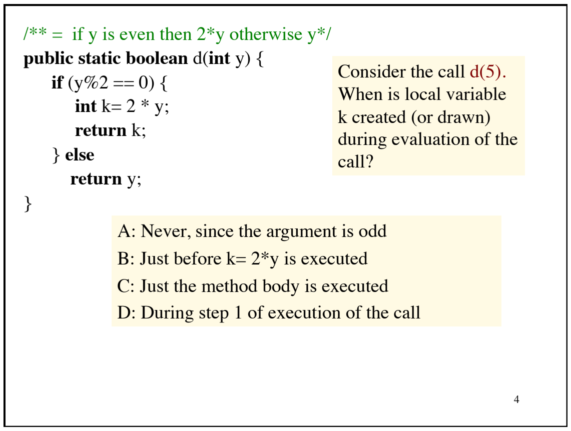
What does **private** mean?

Look it up! Index says p 155. 155-156 says: a component declared with modifier private in class C is accessible only in class C.

a2
fbf
a1
Person
mutual() { … }
a1
fbf
a2
Person
mutual() { … }
SClass

Can't reference fbf in here

```
/** = "the female best-friend
    relation is mutual" —this
person's best friend thinks this
person is their best friend. */
public boolean mutual() {
    return  fbf!= null  &&
            this == fbf.fbf;
}
```



```
/** =  if y is even then 2*y otherwise y*/
public static boolean d(int y) {
    if (y%2 == 0) {
        int k= 2 * y;
        return k;
    } else
        return y;
}
```

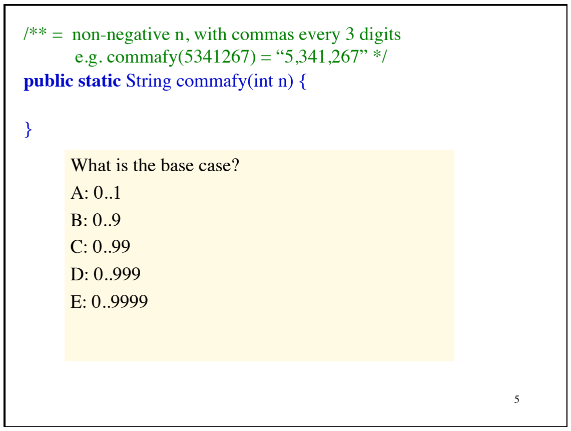
Consider the call d(5). When is local variable k created (or drawn) during evaluation of the call?

A: Never, since the argument is odd

B: Just before k= 2*y is executed

C: Just the method body is executed

D: During step 1 of execution of the call



```
/** =  non-negative n, with commas every 3 digits
        e.g. commafy(5341267) = "5,341,267" */
public static String commafy(int n) {

}
```

What is the base case?

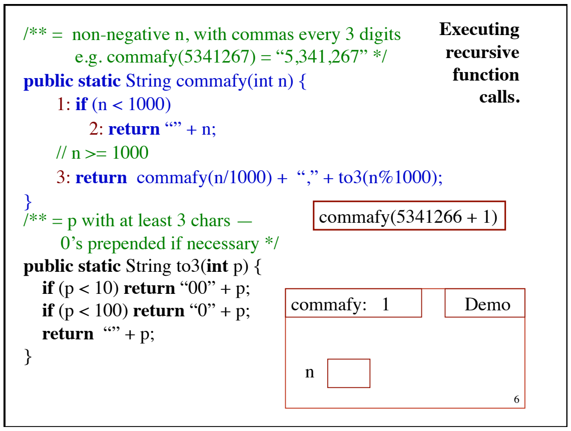
A: 0..1

B: 0..9

C: 0..99

D: 0..999

E: 0..9999



Executing recursive function calls.

```
/** =  non-negative n, with commas every 3 digits
        e.g. commafy(5341267) = "5,341,267" */
public static String commafy(int n) {
    1: if (n < 1000)
        2: return "" + n;
    // n >= 1000
    3: return  commafy(n/1000) +  "," + to3(n%1000);
}
/** = p with at least 3 chars —
    0's prepended if necessary */
public static String to3(int p) {
    if (p < 10) return "00" + p;
    if (p < 100) return "0" + p;
    return  "" + p;
}
```

commafy(5341266 + 1)

commafy: 1
Demo
n

Recursive functions

Properties:

(1) $b^{c} = b * b^{c-1}$

(2) For c even

$b^{c} = (b*b)^{c/2}$

e.g 3*3*3*3*3*3*3*3

= (3*3)*(3*3)*(3*3)*(3*3)

```
/** = b c. Precondition: c ≥ 0*/
public static int exp(double b, int c)
```

Recursive functions

```
/** = b c. Precondition: c ≥ 0*/
public static int exp(double b, int c) {
   if (c == 0)
      return 1.0;
   if (c is odd)
      return b * exp(b, c–1);
   // c is even and > 0
   return exp(b*b, c / 2);
}
```

c	number of calls
0	1
1	2
2	2
4	3
8	4
16	5
32	6
2^n	n + 1

32768 is 2^{15}

so b^{32768} needs only 16 calls!

Binary arithmetic

Decimal	Binary	Octal
00	00	00
01	01	01
02	10	02
03	11	03
04	100	04
05	101	05
06	110	06
07	111	07
08	1000	10
09	1001	11
10	1010	12

	Binary
$2^0 = 1$	1
$2^1 = 2$	10
$2^2 = 4$	100
$2^3 = 8$	1000
$2^4 = 16$	10000
$2^5 = 32$	100000
$2^6 = 64$	1000000
$2^{15} = 32768$	1000000000000000

Test c odd: Test last bit = 1

Divide c by 2: Delete the last bit

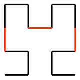
Subtract 1 when odd: Change last bit from 1 to 0.

Exponentiation algorithm processes binary rep. of the exponent.

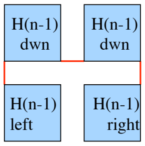
Hilbert's space-filling curve

Hilbert(1):

Hilbert(2):



Hilbert(n):



As the size of each line gets smaller and smaller, in the limit, this algorithm fills every point in space. Lines never overlap.

All methods used in today's lecture will be on course website

Hilbert's space-filling curve