CS100J 20 September 2005

More on Methods. Developing methods

Also: About the use of **this** and **super**

Read section 2.5 on stepwise refinement
Listen to PLive activities 2.5.1 -- 2.5.4!!

Prelim Monday, 26 September
7:30 to 9:00. Ives 305

Quotes that relate to specifying a method before writing it.

A verbal contract isn't worth the paper it's written on.

What is not on paper has not been said.

If you don't know where you are going, any road will take you there.

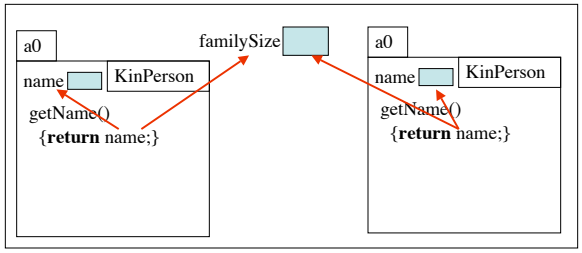
If you fail to plan you are planning to fail.

Don't try to solve a problem until you know what the problem is.

About using **this** and **super**

Inside-out rule in most programming languages (see p. 83):

Code in a construct can reference any of the names declared or defined in that construct, as well as names that appear in enclosing constructs (unless a name is declared twice, in which case the closer one prevails).

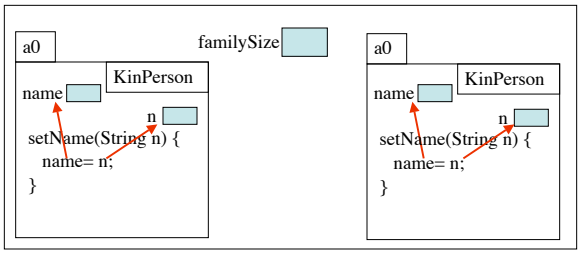


File drawer for class KinPerson

About using **this** and **super**

Inside-out rule in most programming languages (see p. 83):

Code in a construct can reference any of the names declared or defined in that construct, as well as names that appear in enclosing constructs (unless a name is declared twice, in which case the closer one prevails).

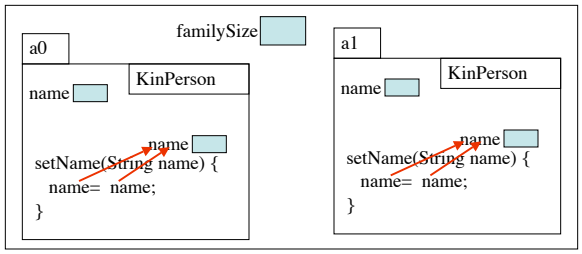


File drawer for class KinPerson

About using **this** and **super**

Inside-out rule in most programming languages (see p. 83):

Code in a construct can reference any of the names declared or defined in that construct, as well as names that appear in enclosing constructs (unless a name is declared twice, in which case the closer one prevails).



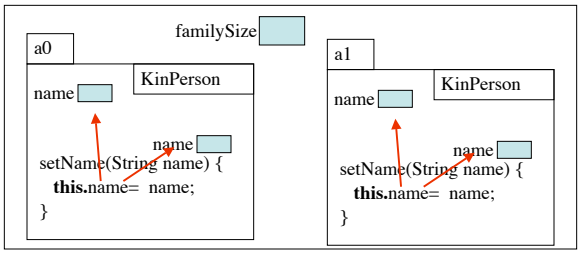
File drawer for class KinPerson

About using **this** and **super**

Within a folder, this refers to the folder itself,

In folder a0, **this** refers to folder a0.

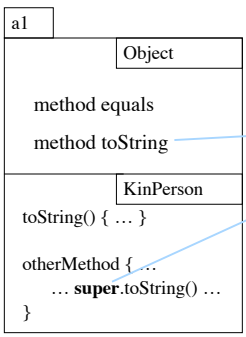
In folder a1, **this** refers to folder a1.



File drawer for class KinPerson

About using **this** and **super**

Within a subclass folder, super refers to the folder —except the lowest partition.



Because of the keyword **super**, this calls toString in the Object partition.

Anglicizing an integer

```
/** = the English equivalent of n, for 1 <= n < 1,000
    e.g. ang(3) is "three"
         ang(412) is "four hundred twelve"
         ang(762) = "seven hundred sixty two" */
public static String ang(int n)
```

Hint at how to do it. When we add 5 + 3 + 2 + 8, we start with x = 0 and add one value to x at a time, step by step:

x= x + 5; x= x + 3; x= x + 2; x= x + 8;

Definition of x: *x is the sum of the values added so far*.

Can we start with String variable s = "" and step by step catenate pieces on to the end of it?

What's the definition of s?

Anglicizing an integer

```
/** = the English equivalent of n, for 1 <= n < 1,000
    e.g. ang(3) is "three"
         ang(641) is "six hundred forty one" */
public static String ang(int n)
```

Start with String variable s = "" and step by step catenate pieces on to the end of it? Use two local variables, s and k.

	s	k
start:	""	641
	"six hundred"	41
	"six hundred forty"	1
end	"six hundred forty one"	0

Definition of s and k:

ang(n) is s + ang(k)

To find ang(n), anglicize k and append the result to s.

Anglicizing an integer

```
/** = the English equivalent of n, for 1 <= n < 1,000
    e.g. ang(3) is "three"
         ang(641) is "six hundred forty one" */
public static String ang(int n) {
  // ang(n) is  s + ang(k)
  String s= "";
  int k= n;

}
```

You are expected to study section 13.4!

This definition of s and k is *very important*. This definition will drive the development. Whenever we append something to s, we have to change k to keep the definition true. The definition helps us develop the method body and helps the reader understand it.

Whenever you declare a local variable whose value will change often over execution of the method, **write a comment near its declaration to define it**!!

You will use the definition often as you develop the method. Without the definition, you will forget what the variable means and will make mistakes.

Anglicizing an integer

```
/** = the English equivalent of n, for 1 <= n < 1,000
    e.g. ang(3) is "three"
         ang(641) is "six hundred forty one" */
public static String ang(int n) {
  // ang(n) is  s + ang(k)
  String s= "";
  int k= n;

}
```

The rest of this lecture is devoted to the development of this algorithm, using DrJava. The final program will be put on the course website.