

CS1110 20 September 2010

Today: Pick up: A3
Today's slides

Inside-out rule; use of **this**, **super**

Developing methods (using Strings). You can do A3 in groups of 2.

Read sec. 2.5, stepwise refinement BUT GROUP EARLY ON

Listen to Plive, 2.5.1–2.5.4. CMS

Reading for next lecture:
the same

Office hours
are being held

Research on spelling

According to a research at Cambridge University, it doesn't matter in what order the letters in a word are, the only important thing is that the first and last letter be at the right place. The rest can be a total mess and you can still read it without problem. This is because the human mind does not read every letter by itself, but the word as a whole.

A3: Adding functionality to A1
Due Tuesday, 22 February

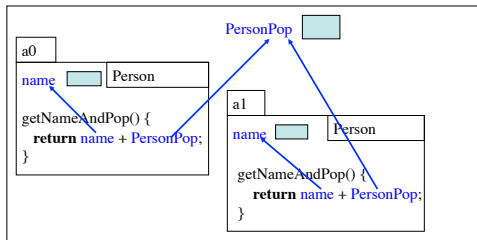
- Keeping class invariant true
- Use already-written functions
- Boolean expressions
- Use of null and testing for it
- Use of static variables

Form groups on the CMS early,
well before you submit.

A3: graded in conventional way.
Submit once and get a grade.

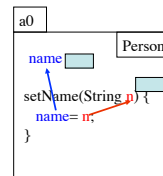
The inside-out rule (see p. 83)

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. (If a name is declared twice, the closer one prevails.)

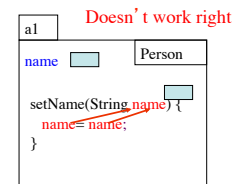


File drawer for class Person 3

Method parameters participate in the inside-out rule: remember the frame.



Parameter **n** would be found in the frame for the method call.



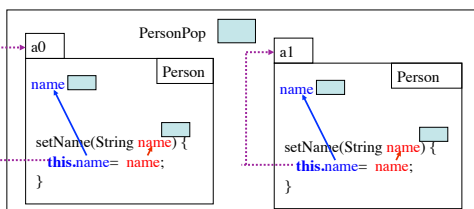
Parameter **name** "blocks" the reference to the field **name**.

A solution: **this** and **super**

Memorize: Within an object, **this** evaluates to the name of the object.

In folder a0,
this refers to a0

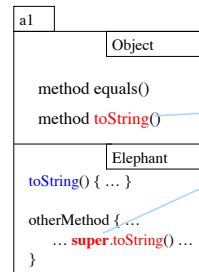
In folder a1,
this refers to a1



File drawer for class Person 5

About **super**

Within a subclass object, **super** refers to the partition above the one that contains **super**.



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

Strings are (important) objects that come with useful methods.

```
String s= "abc d";
```

Note the "index (number) from 0" scheme:

```
0 1 2 3 4
a b c d
```

| | |
|---------------------|--------|
| s | as |
| length() | String |
| charAt(int) | |
| substring(int) | |
| substring(int, int) | |
| indexOf(String) | |
| lastIndexOf(String) | |
| ... | |

s.length() is 5
s.charAt(2) is 'c'
s.substring(2) is "c d"
s.substring(1,3) is "bc"

To find specs of methods in String:
1. Visit course website
2. Click [Links](#)
3. Click [Specs for version 1.6](#)
4. Click String in lower left pane

Strings are (important) objects that come with useful methods.

```
String s= "abc d";
```

Note the "index (number) from 0" scheme:

```
0 1 2 3 4
a b c d
```

Text pp. 175–181 discusses Strings
Look in CD ProgramLive
Look at API specs for String

s.length() is 5 (number of chars)
s.charAt(2) is 'c' (char at index 2)
s.substring(2,4) is "c" (NOT "c d")
s.substring(2) is "c d"
"bcd".trim() is "bcd" (trim beginning and ending blanks)
s.indexOf(s1) –index or position of first occurrence of s1 in s (-1 if none)

8

Principles and strategies embodied in stepwise refinement

Develop algorithm step by step, using principles and strategies embodied in "stepwise refinement" or "top-down programming".
READ Sec. 2.5 and Plive p. 2-5.

- **Take small steps.** Do a little at a time
- **Refine.** Replace an English statement (**what to do**) by a sequence of statements to do it (**how to do it**).
- **Refine.** Introduce a local variable —but only with a reason
- **Compile often**
- **Intersperse programming and testing**
- **Write a method specification** —before writing its body
- **Separate concerns:** focus on one issue at a time
- **Mañana principle:** next slide

9

Reordering a name

```
lastNameFirst("David Gries") is "Gries, David"
```

```
/** = s copy of s but in the form <last-name>, <first-name>.
Precondition: s contains a name in the form
<first-name> <last-name>
with one blank between the two names*/
public static String lastNameFirst(String s) {
}
}
```

10

Reordering a name

```
lastNameFirst("David Gries") is "Gries, David"
```

```
/** = s copy of s but in the form <last-name>, <first-name>.
Precondition: s contains a name in the form
<first-name> <last-name>
with one or more blanks between the two names*/
public static String lastNameFirst(String s) {
}
}
```

11

Principles and strategies

- **Mañana Principle.**

During programming, you may see the need for a new method. A good way to proceed in many cases is to:

1. Write the specification of the method.
2. Write just enough of the body so that the program can be compiled and so that the method body does something reasonable, but no the complete task. So you *put off* completing this method until another time —**mañana (tomorrow)** —but you have a good spec for it.
3. Return to what you were doing and continue developing at that place, presumably writing a call on the method that was just "stubbed in", as we say.

12