CS100J Classes, stepwise refinement 14 Feb 2007

Miscellaneous points about classes.
More on stepwise refinement.

Prelim 7:30-9:00 Thursday, 21 Sept., Olin 155

Review session: 1:00-3:00, Sunday, 17 Sept., Philips 101

Rsrecah on spleilng

Aoccdrnig to a rscheearch at Cmabirgde Uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is that the frsit and lsat ltteer be at the rghit pclae. The rset can be a total mses and you can sitll raed it wouthit porbelm.Tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef, but the wrod as a wlohe.

Help: Get it now if you need it!!

- One-on-one help from TAs. For info, get on the course website and click "Staff-info".
- Call Cindy Pakkala 255-8240 for an appointment with Gries.
- See a consultant in the ACCEL Sun, Mon, Tues, Thurs 2:30pm to 11pm. Wed, 3:35-11:00pm..
- Peer tutoring (free). On http://www.engineering.cornell.edu, click on "student services". On the page that comes up, click on "Learning Initiatives (L.I.F.E.) in the left column, upper part. Then, click on "peer tutoring" in the left column.
- Take an AEW courses. Ask in Olin 167.

Content of this lecture

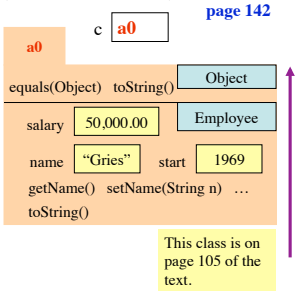
This lecture contains some final miscellaneous points to round out your knowledge of classes and subclasses. There are a few more things to learn after this, but we will handle them much later.

- Inheriting fields and methods and overriding methods. Sec. 4.1 and 4.1.1: pp. 142–145
- Purpose of **super** and **this**. Sec. 4.1.1, pp. 144–145.
- More than one constructor in a class; another use of **this**. Sec. 3.1.3, pp. 110–112.
- Constructors in a subclass —calling a constructor of the super-class. Sec. 4.1.3, pp. 147–148.

Employee c= **new** Employee("Gries", 1969, 50000); **Sec. 4.1, page 142**
c.toString()

Which method toString() is called?

Overriding rule or bottom-up rule:
To find out which is used, start at the bottom of the class and search upward until a matching one is found.



This class is on page 105 of the text.

Terminology. Employee **inherits** methods and fields from Object. Employee **overrides** function toString.

Purpose of super and this **Sec. 4.1, pages 144-145**

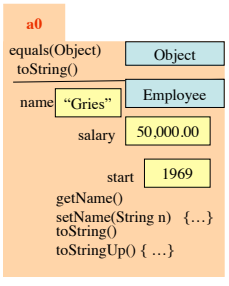
this refers to the name of the object in which it appears.
super is similar but refers only to components in the partitions above.

```
/** = String representation of this
Employee */
public String toString() {
    return this.getName() + ", year " +
        getStart() + ", salary " + salary;
}
```

ok, but unnecessary

```
/** = toString value from superclass */
public String toStringUp() {
    return super.toString();
}
```

necessary



A second constructor in Employee
Provide flexibility, ease of use, to user **Sec. 3.1.3, page 110**

```
/** Constructor: a person with name n, year hired d, salary s */
public Employee(String n, int d, double s) {
    name= n; start= d; salary= s;
}
```

First constructor

```
/** Constructor: a person with name n, year hired d, salary 50,000 */
public Employee(String n, int d) {
    name= n; start= d; salary= 50000;
}
```

Second constructor; salary is always 50,000

```
/** Constructor: a person with name n, year hired d, salary 50,000 */
public Employee(String n, int d) {
    this(n, d, 50000);
}
```

Another version of second constructor; calls first constructor

Here, **this** refers to the other constructor.
You HAVE to do it this way

```
public class Executive extends Employee {
  private double bonus;

  /** Constructor: name n, year hired
          d, salary 50,000, bonus b */
  public Executive(String n, int d, double b) {
      super(n, d);
      bonus= b;
  }
}
```

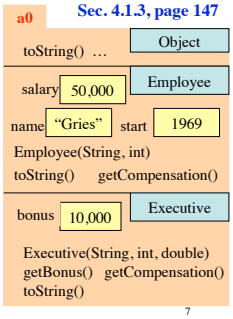
The first (and only the first) statement in a constructor has to be a call to a constructor of the superclass. If you don't put one in, then this one is automatically used:

```
super();
```

Principle: Fill in superclass fields first.

Calling a superclass constructor from the subclass constructor

Sec. 4.1.3, page 147

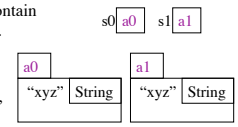


More about equals

To test whether two String values contain the same string, use function equals.

s1.equals(s2) is true.

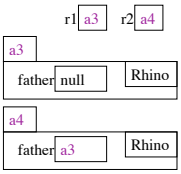
s1 == s2 asks whether a0 == a1, which is false



To test whether two Rhinos are the same Rhino, use ==.

To test whether r2's father is r1, use r2.father == r1

a3 == a3



Anglicizing an Integer

anglicize("1") is "one"
anglicize("15") is "fifteen"
anglicize("123") is "one hundred twenty three"
anglicize("10570") is "ten thousand five hundred seventy"

```
/** = the anglicization of n.
      Precondition: 0 <= n < 1,000,000 */
public static String anglicize(int n) {

}
```

We develop this 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.

Important is how we intersperse programming and testing.

Help: Get it now if you need it!!

- One-on-one help from TAs. For info, get on the course website and click "Staff-info".
- Call Cindy Pakkala 255-8240 for an appointment with Gries.
- See a consultant in the ACCEL Sun, Mon, Tues, Thurs 2:30pm to 11pm. Wed, 3:35-11:00pm..
- Peer tutoring (free). On http://www.engineering.cornell.edu, click on "student services". On the page that comes up, click on "Learning Initiatives (L.I.F.E.) in the left column, upper part. Then, click on "peer tutoring" in the left column.
- Take an AEW courses. Ask in Olin 167.