Content of this lecture

Go over 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; another use of super. Sec. 4.1.3, pp. 147–148.

Purpose of super and this

This refers to the name of the object in which it appears. Sec. 4.1, pages 144-145

super is similar but refers only to components in the partitions above.

Employee c = new Employee(“Gries”, 1969, 50000); 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.

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

A second constructor in Employee

Provide flexibility, ease of use, to user

- Call Cindy 255-8240 for an appointment with David Gries.
- Email Lillian Lee to make an appointment: lle@cs.cornell.edu
- See a consultant in the ACCEL Lab: Sun, Mon, Tues, Wed, Thurs during office hours.
- See a TA.
- Peer tutoring (free). Ask in Olin 167 or visit On http://www.engineering.cornell.edu, click on "student services". On the page that comes up, click on "Engineering Learning Initiatives (ELI) " in the left column, upper part. Then, click on "peer tutoring" in the left column.
**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”

```java
/** = the anglicization of n. */
public static String anglicize(int n) {
    // Implementation...
}
```

---

### 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.