CSCI 1321

Administrivia

• "Design" phase of Homework 1 due today (at 11:59pm). If you have

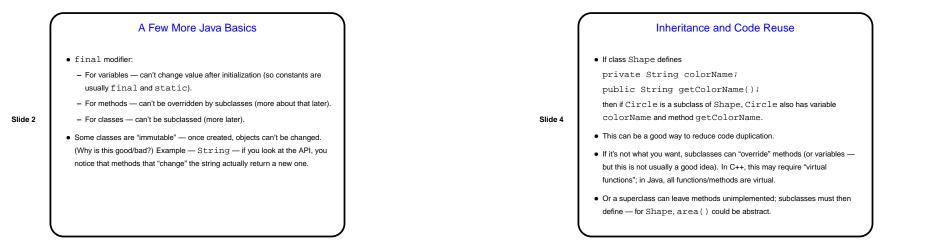
questions Tuesday (if not earlier via e-mail or office hours).

for this assignment.

CSCI 1321

September 11, 2003

Inheritance, Revisited • Recall two roles from "short version" earlier - code reuse, subtypes. last-minute questions, try e-mail. I'm likely to be lenient about late penalties • Recall that classes form a hierarchy/tree (with Object at root). • "Code" phase due Tuesday. There should be time to sort out last-minute Slide 3



Slide 1

CSCI 1321

Slide 6

CSCI 1321

Inheritance and Subtypes Interfaces and Types • In the "shapes" example, class Shape defines a type, and Circle and • Interfaces also define types. So if Shape implements interface Rectangle are subtypes. Anywhere we need a Shape, we can use a HasColorName, we can use a Shape anywhere a HasColorName is Circle. required. HasColorName o = new Shape(); Shape s = new Circle(); (but not Circle = new Shape()) Slide 5 Slide 7 • This is "inclusion polymorphism" - and is what will allow your project code to plug neatly into Dr. Lewis's framework. (The framework is written in terms of interfaces such as ${\tt Block}$ and ${\tt Screen};$ your classes will implement those interfaces.)

Multiple Inheritance Versus Interfaces • What if you want a class to inherit from multiple classes? C++ allows this ("multiple inheritance"). To avoid possible confusion/ambiguity, Java doesn't. • Instead, define "interfaces" — classes in which all methods are abstract. • In Shape example, we could define a HasColorName interface with methods getColorName and setColorName. • A class can "implement" as many interfaces as you like.

CSCI 1321

September 11, 2003

