## Overview: abstract classes and interfaces

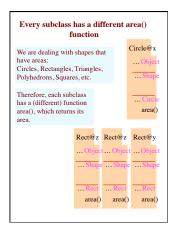
Make a class abstract so instances of it cannot be created. Make a method abstract so it must be overridden.

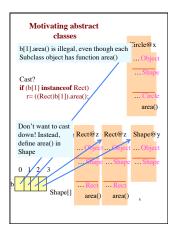
An interface is like an abstract class whose methods are all abstract and whose fields are all public constants. This allows multiple inheritance without ambiguity. An interface has a different syntax and a different way of using it.

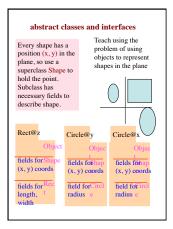
References to text and to JavaSummary.pptx Abstract class: C.27, slides 42-44 Abstract method: C.27, slide 44 Interface declaration: D.11-D.13, D.28, slide 60

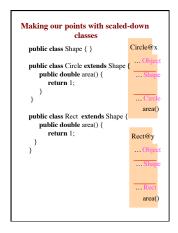
Implementing interfaces: D.14-D.15, slide

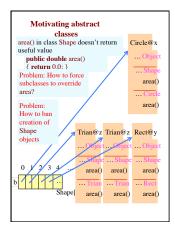
Casting with interfaces: none, slide 61 Interface Comparable: D.20, slide 62



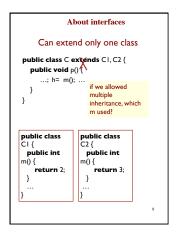


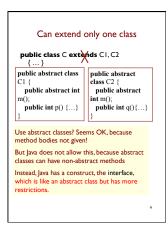


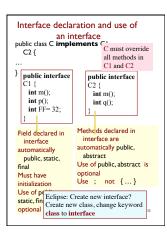










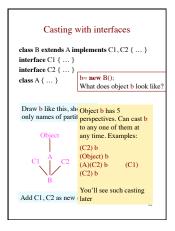


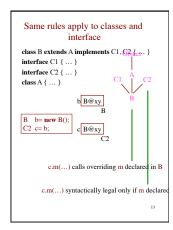


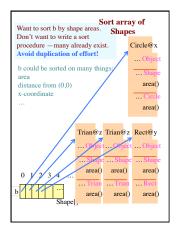
An interface is a fully abstract class, with a slightly different syntax.

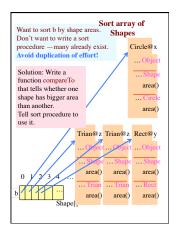
An interface can contain type signatures for methods, just like abstract methods in abstract classes, but they have to be public.

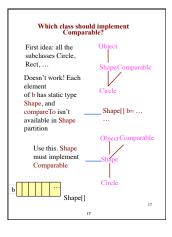
An interface can contain fields, but they have to be public, static, and final and they have to contain an initializer. So they are really just constants

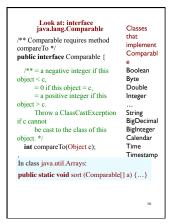


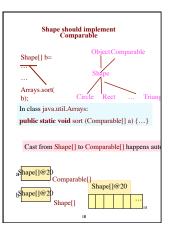


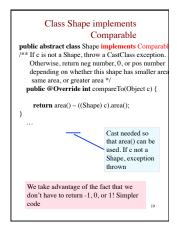












## Java Library static methods:

Arrays.sort(Comparable[] a) Collections.sort(List<Comparable> list)

Class arrays has many other useful static methods

## Beauty of interfaces:

Arrays.sort sorts an array or list C[] for any class C, as long as C implements interface Comparable —and thus implements compareTo to say which of two elements is bigger.

20