

Interface Inheritance

CSE 114 INTRODUCTION TO OBJECT-ORIENTED PROGRAMMING



Announcements

Midterm 3:

- Exam: 23-May
- Review: 21-May
- Covers: primarily material from after exam 2 (Arrays of Objects) up through lecture and lab on inheritance (next week)

Today: Subtyping, interface inheritance

Reading assignment for this slide set: the lecture notes

java.lang.Object class

Every Java class inherits `Object` as its parent class

`Object` is a *superclass* of any and every class that we create

- e.g., `Point` is a *subclass* of `Object`

We also say that

- `Object` is a *supertype* of `Point`
- `Point` is a subtype of `Object`

A variable of supertype can hold a subtype object

A variable of subtype **cannot** hold a supertype object. Why not?

Subtype polymorphism (subtyping)

In general, we can substitute a subtype object for a supertype variable: *substitution principle* (subtyping is the theoretical basis that makes this possible)

Declared type vs. actual type

```
Object o1;           // declared type of o1 is Object
o1 = new Point(1, 2); // actual type of o1 is Point
```

The assignment above is possible because of subtyping

See [Subtype.java](#) and [Point.java](#)

Inheritance

Interface inheritance

- Subtype inherits **only the interface** of its supertype

Implementation inheritance

- Subtype inherits **both interface and implementation** of its supertype

Interface inheritance

An interface is really a new type that we are defining, much like the type that gets created when we define a class.

We can use the interface name, Shape here, as the type of a variable. This also means that if a class '*implements*' this interface, the class **MUST implement ALL the methods** that are declared in the interface to be able to act as a complete class.

The methods declared in an interface are declared as abstract. These *abstract* methods become *concrete* when they are implemented in a class that inherits the interface.

See [Shape.java](#), [Circle.java](#), [Rectangle.java](#), [Box.java](#), [Point.java](#), [UseShape.java](#)

Sorting and searching (revisited)

Now that we understand interfaces, let us redo sorting an array of objects using the [Comparable](#) interface.

See [sorting_objects_2/Point.java](#)

[Selection.java](#)

[ArrayTools.java](#)