Inheritance in Java

CSC207 Fall 2015
Inheritance hierarchy

All classes form a tree called the inheritance hierarchy, with `Object` at the root.

Class `Object` does not have a parent. All other Java classes have one parent.

If a class has no parent declared, it is a child of class `Object`.

A parent class can have multiple child classes.

Class `Object` guarantees that every class inherits methods `toString`, `equals`, and others.
Inheritance

Inheritance allows one class to inherit the data and methods of another class.

In a subclass, `super` refers to the part of the object defined by the parent class.

- Use `super.«attribute»` to refer to an attribute (data member or method) in the parent class.
- Use `super(«arguments»)` to call a constructor defined in the parent class.
Constructors and inheritance

If the first step of a constructor is `super(«arguments»)`, the appropriate constructor in the parent class is called.

- Otherwise, the no-argument constructor in the parent is called.

Net effect on order if, say, A is parent of B is parent of C?

Which constructor should do what? Good practise:

- Initialize your own variables.
- Count on ancestors to take care of theirs.
Multi-part objects

Suppose class Child extends class Parent.

An instance of Child has

- a Child part, with all the data members and methods of Child
- a Parent part, with all the data members and methods of Parent
- a Grandparent part, ... etc., all the way up to Object.

An instance of Child can be used anywhere that a Parent is legal.

- But not the other way around.
Name lookup

A subclass can reuse a name already used for an inherited data member or method.

Example: class Person could have a data member motto and so could class Student. Or they could both have a method with the signature sing().

When we construct
   x = new Student();
the object has a Student part and a Person part.

If we say x.motto or x.sing(), we need to know which one we’ll get!

In other words, we need to know how Java will look up the name motto or sing inside a Student object.
Name lookup rules

For a method call: expression.method(arguments)

- Java looks for method in the most specific, or bottom-most part of the object referred to by expression.
- If it’s not defined there, Java looks “upward” until it’s found (else it’s an error).

For a reference to an instance variable: expression.variable

- Java determines the type of expression, and looks in that box.
- If it’s not defined there, Java looks “upward” until it’s found (else it’s an error).
Shadowing and Overriding

Suppose class A and its subclass AChild each have an instance variable x and an instance method m.

A’s m is **overridden** by AChild’s m.

- This is often a good idea. We often want to specialize behaviour in a subclass.

A’s x is **shadowed** by AChild’s x.

- This is confusing and rarely a good idea.

If a method must not be overridden in a descendant, declare it **final**.