Inheritance

28 Jan 2008
CMPT166
Dr. Sean Ho
Trinity Western University

TRINITY WESTERN UNIVERSITY

Example: TimeTest

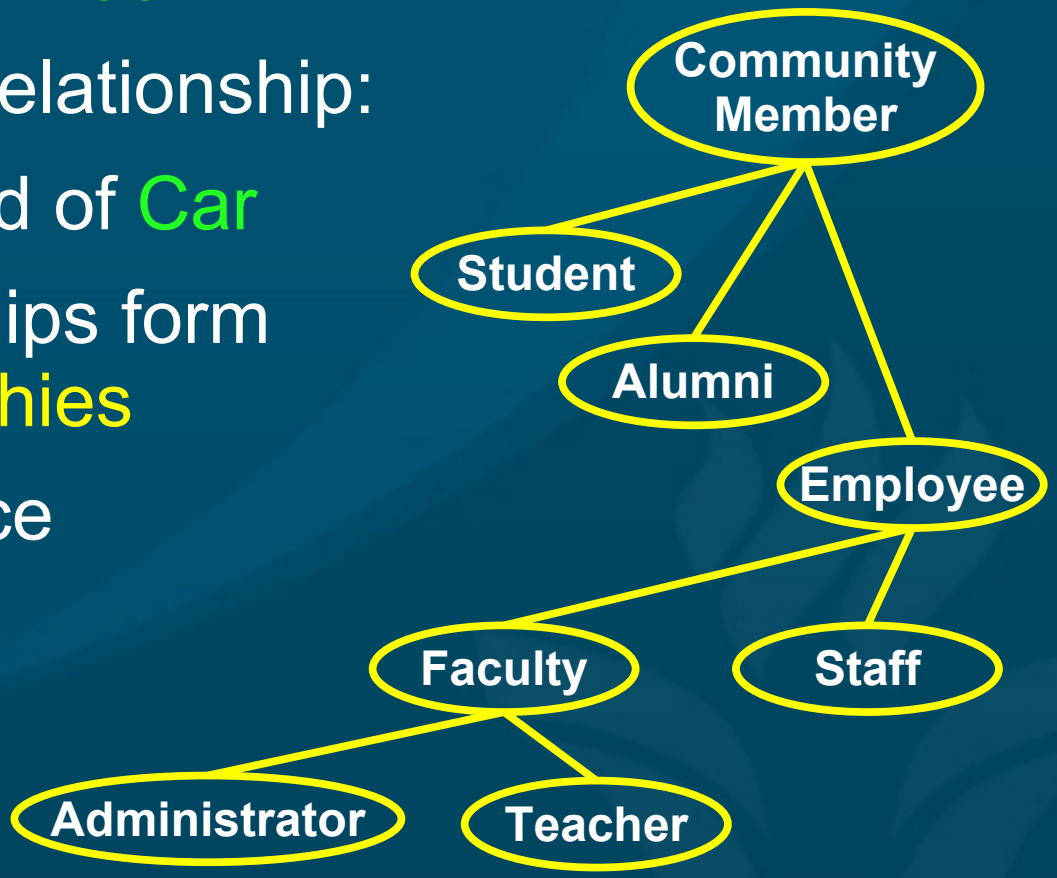
- (See java/ examples directory)
- Time1 is a subclass of Object
  - No main() function
  - Instantiated in TimeTest
- TimeTest is our main program
  - Testbed for the Time1 class

Why use inheritance?

- Reusability
  - Create new classes from existing ones
    - Absorb attributes and behaviours
    - Add new capabilities
- Polymorphism
    - Enable developers to write programs with a general design
    - A single program can handle a variety of existing and future classes
    - Aids in extending program, adding new capabilities

Superclasses and subclasses

- Attribute: "has a" relationship:
  - A Car has a steeringWheel
- Subclass: "is a kind of" relationship:
  - A Convertible is a kind of Car
  - Inheritance relationships form tree-like class hierarchies
- Polymorphism: write once
  - changeOil() method
  - works on all Cars, not just Convertibles



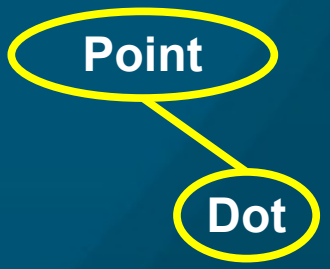
Constructors

- class Dot extends Point

- A subclass' constructor does not inherit/override the superclass constructor
- But it implicitly calls the superclass constructor:
  - public Dot() { /* implicitly calls Point() */ }
  - Can also explicitly call with super()
    - public Dot() {
      - super(); // explicitly call Point() first
      - ... // do Dot-specific stuff here
- See PointDot.java

Using subclass instances

- An instance of a subclass can be treated as an instance of the superclass:
  - Point p2 = new Dot();
  - Cannot do vice-versa:
    - Dot d1 = new Point(); // illegal!



- instanceof checks the class of an object:
  - if (p2 instanceof Dot)
- A superclass reference may be downcast back to the subclass if appropriate:
  - Dot d2 = (Dot) p2; // this is ok: p2 is really a Dot