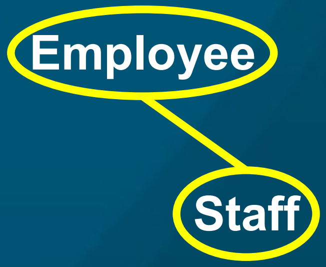
ch8: Polymorphism

30 Jan 2008
CMPT166
Dr. Sean Ho
Trinity Western University

Review last time

- Inheritance for software reusability
  - “has a” vs. “is a kind of”
- Subclass/superclass constructors
  - super()
- Subclass/superclass references
  - Downcasting



Quiz 1 (10 min)

- Explain what the JDK and JRE are and contrast them.
- Explain what an applet is.
- How are comments done in Java? (both ways)
- Each box in a UML class diagram has three sections. What are they?
- What is method overloading?
- Write a complete command-line Java program that prints “Hello World!”.
  - Doc-comments not necessary

Quiz 1: answers #1-3

- Explain what the JDK and JRE are and contrast them. [4]
  - Java Development Kit: compiler and runtime
  - Java Runtime Environment: just the VM
- Explain what an applet is. [2]
  - Small program to be run within a webpage
- How are comments done in Java? [2]
  - /* C-style */ and // double-slashes

Quiz 1: answers #4-5

- Each box in a UML class diagram has three sections. What are they? [3]
  - Class name, attributes (variables), methods
- What is method overloading? [3]
  - Different copies/versions of a method, depending on the type of the arguments

Quiz 1: answers #6

- Write a complete command-line Java program that prints "Hello World!". [6]

```
public class HelloWorld {
    public static void main( String args[] ) {
        System.out.println( "Hello, World!" );
    }
}
```

What's on for today

- Copy constructors
- Type-wrapper classes for the primitive types
- Polymorphism
  - Dynamic method binding
  - final keyword for classes and methods
  - Abstract and concrete classes
    - abstract keyword for classes and methods
- Interfaces
  - vs. abstract superclasses

Copy constructors

- Just like in Python, names referring to object instances are references (aliases) to the object
  - Student bob = new Student("Bob", 1234);
  - Student sally = bob;   // alias
- It is good habit to add a copy constructor that copies the contents of an existing instance of the same class:
  - public class Student {

```
    public Student( Student orig ) {
        this.name = orig.name;          // what if orig is null?
        this.ID = orig.ID;
        ....
```

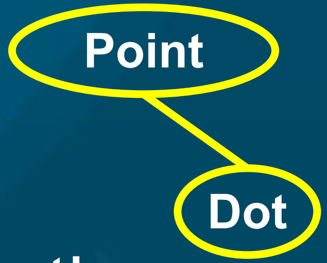
  - Student sally = new Student(bob) ;

Primitive type-wrapper classes

- Eight primitive types in Java
  - Primitives are not really objects
- Type-wrapper classes for each of the eight:
  - Character, Byte, Integer, Boolean, etc.
  - Enable us to represent primitives as Object
  - Can then process them polymorphically
- Type-wrapper classes declared final
  - Many methods declared static
    - e.g., Integer.parseInt(String)

Polymorphism

- Think carefully about class hierarchy in program design
- Write programs/algorithms to operate on superclass objects
  - As generic as possible
- Instances of subclasses can be operated on by the algorithms without need for modification
- Dynamic method binding:
  - Java chooses correct method (e.g., toString()) from subclass



final: methods/classes

- We've seen final on variables: set as constant
- final on a method prevents subclasses from overriding
- final on a class means it cannot be extended
  - (Other classes cannot inherit from it)