Object Hierarchies
is-a, has-a
Lecture 32

Readings
This Week: Ch 9.4-9.5 and into Ch 10.1-10.8
(Ch 11.4-11.5 and into Ch 13 in old 2nd ed)
Next Week: No new readings. Consolidate!
(Reminder: Readings are absolutely vital for
learning this stuff!)

Labs and Tutorials
This Week: Lab #9
Next Week: Lab #10

Survey #2
Dr. Ben Yu’s second survey for you all is up
on WebCT now.
Completely optional…
But, there’s a bit of extra credit for doing all
three surveys by their respective deadlines!
The deadline for this one is March 29.
Because of the server crash, if you did
this survey before the crash, you’ll have
do it again.

Final Exam
Wednesday, April 15, 7pm in SRC A
This wasn’t a good room last year, but we’re
stuck with the date, time, and room UBC
assigns.

Programming Assignment 3
Assignment 3 is up on WebCT!
Click on the “Assignments” icon.
Assigned Sunday evening – sorry for delay!
Due at NOON, April 6 (Monday), via
electronic hand in.
Tips:
There is some Eclipse setup. Set-up ASAP!
Work in pairs. Some conceptual stuff.
Think carefully before coding. If concepts
right, the coding is much much easier.
Programming Assignment 3

- I found out after class on Monday that some versions of Vista and Java don’t quite get along for some graphics programs.
- I’ve spent the last couple days pinning this down. My initial guesses weren’t correct.
- I have a fix now that works for several people. See the assignment webpage or WebCT for details.

Learning Goals
By the end of class today you will be able to...

- Decide when it’s good to inherit from another class and when it’s not so good.
- Explain the is-a and has-a relationships.

Review: Inheritance in Java

- You can declare a new class as an extension of an existing one.
- The new class automatically inherits all the instance fields and methods of the old class.
- The new class can add/change fields and methods.

```java
public class ChildClass extends ParentClass {
    ...
    put any additional fields and methods here
    ...
}
```

Review: Overriding

- You can declare a new class as an extension of an existing one.
- The new class automatically inherits all the instance fields and methods of the old class.
- The new class can add/change fields and methods.

```java
public class ChildClass extends ParentClass {
    anything new here is added to the ChildClass
}

anything with same signature as in ParentClass overrides the ParentClass
```

Review: Shadowing and Overriding

- These are basically the same concept:
  - If you have two declarations of the same signature (name and parameters), the closer declaration wins.

Special Case: Superclass Constructor

- Recall the special use of this in constructors?
  - this as first line of constructor calls a different constructor for the same object
  - E.g., UBCStudent class
Special Case: Superclass Constructor
- Recall the special use of `this` in constructors?
  - `this` as first line of constructor calls a different constructor for the same object
  - E.g., UBCStudent class
- Similar trick to call a superclass constructor using the `super` keyword:
  - `super` call must be first line of constructor

How Child Objects Are Constructed
- The child object contains everything the superclass object does, so the superclass constructor needs to get called.
- Java automatically calls the superclass’s constructor, before the child’s constructor starts working.
  - What parameters does it pass? None!
- The special case of calling `super` on the first line tells Java to call the superclass constructor in the way you specify, instead of without parameters.

Questions?

Silly Inheritance Example Revisited
- Let’s make UBCStudent extend TalkingDoll…
- We did this just to have a simple example to practice inheritance and overriding…
- But was this a good use of inheritance?

Silly Inheritance Example Revisited
- Let’s make UBCStudent extend TalkingDoll…
- We did this just to have a simple example to practice inheritance and overriding…
- But was this a good use of inheritance?
- Good: We got to re-use the name field and the getName and setName methods.
Silly Inheritance Example Revisited
- Let's make UBCStudent extend TalkingDoll…
- We did this just to have a simple example to practice inheritance and overriding…
- But was this a good use of inheritance?
- Good: We got to re-use the name field and the getName and setName methods.
- Bad: UBCStudents gained the undesirable property of saying random phrases.

Based on True Story
- Australian DSTO (Defence Science and Technology Organisation) military simulation.
- Not in Java, but in an object-oriented simulation system with the same concepts.
- Designers added kangaroos to the model…

The Real Story
"For a bit of extra fun (and not for any strategic reason like kangaroos betraying your cover!) our programmers decided to put in a bit of animated wildlife… It is true that the first time this was tried in the lab, we discovered that we had forgotten to remove the weapons and the 'fire' behaviour. It is NOT true that this happened in front of a bunch of visitors (American or any other flavour). We don't normally try things for the first time in front of an audience! What I didn't relate in the talk is that since we were not at that stage interested in weapons, we had not set any weapon or projectile types, so what the kangaroos fired at us was in fact the default object for the simulation, which happened to be large multicoloured beachballs."

Roughly What Happened
- setSpeed()
- setStrategy()
- setAAWeapon()
- display()

LightInfantry
Kangaroo
roos[i] = new Kangaroo(...);
roos[i].setSpeed(something fast);
roos[i].setStrategy(RETREAT);

Roughly What Happened

Last Time: Inheritance in Java
- You can declare a new class as an extension of an existing one.
- The new class automatically inherits all the instance fields and methods of the old class.
- The new class can add/change fields and methods.
- This means the subclass objects has everything and can do everything the superclass object can.
- So, the subclass object is a superclass object!
The is-a Relation

- You can declare a new class as an extension of an existing one.
- The new class automatically inherits all the instance fields and methods of the old class.
- The new class can add/change fields and methods.
- This means the subclass objects has everything and can do everything the superclass object can.
- So, the subclass object is a superclass object!

The is-a Relation

- This is a good test for whether one class should be a subclass of another.
- Do you think an object of one class “is a” (special kind or extension of) object of another class?
- If so, then this is likely a good time to use inheritance.
- If not, then you probably don’t want to use inheritance.

The is-a Relation

- This is a good test for whether one class should be a subclass of another.
- Are UBC students (possibly special versions of) talking dolls?

The is-a Relation

- This is a good test for whether one class should be a subclass of another.
- Are kangaroos a specialized form of infantry soldier?

The is-a Relation

- What would be a good use of inheritance if we wanted to model UBCStudent, TalkingDoll, and Celebrity in a program?
- What’s a general category that UBC students, talking dolls, and celebrities have in common?
- They are all characters/personas/agents with names!
Object Hierarchy

- So, we could create a class Person
  - with an instance field: String name;
  - with methods getName, setName, sayHello,…
- A UBC student is a person, so
  - class UBCStudent extends Person {...}
  - adds methods/fields for handling library fines, etc.
- A celebrity is a person
  - class Celebrity extends Person {...}
  - adds methods/fields for handling big salaries
- A talking doll is kind of a person
  - class TalkingDoll extends Person {...}
  - adds methods/fields for saying random things

Object Hierarchies

- Suppose we also want to model cats and dogs in this program. What would be a good object hierarchy?

Object Hierarchies

- In fact, in Java, ALL objects are actually subclasses, all descended from the class Object!
- (This is why every class you’ve declared automatically has a toString and clone method, although you’ll usually want to override these.)

Object Hierarchies

- Suppose we also want to model cats and dogs in this program. What would be a good object hierarchy?
  - OK, now suppose this is a virtual world simulation, so we want a location for all these Actors
  - Hey, we have a Point class! If we make Actor extend Point, then all of these classes will automatically have a location…
  - Good idea?
  - No… these other objects are NOT points!

Object Hierarchies

- OK, now suppose this is a virtual world simulation, so we want a location for all these Actors
  - Hey, we have a Point class! If we make Actor extend Point, then all of these classes will automatically have a location…
  - Good idea?

The has-a Relation

- If you think an object “has a” characteristic or some information, make that an instance field in the object. Don’t inherit the properties the object has.