

CSSE 220 Day 20

Inheritance recap
Object: the superest class of all
Inheritance and text in GUIs

Check out *Inheritance2* from SVN

Questions?

Project Team Preference Survey

- ▶ On ANGEL, under Lessons → Assignments
- ▶ Preferences help me to choose teams; I also consider your performance so far in the course
- ▶ Complete the survey by Wednesday noon
- ▶ Most teams will have 3 students
- ▶ Are you willing to be on a team of 2
- ▶ List up to 5 students you'd like to work with, highest preference first.
 - You may not get your first choices, so it's a good idea to list more than two
 - Best to choose partners whose commitment level and current Java coding/debugging ability is similar to yours
- ▶ List up to 2 students you'd prefer NOT to work with
 - I'll do my best to honor this, but I must find a team for everyone. (What if you don't complete the survey?)

Inheritance Review

»» A quick recap of last session

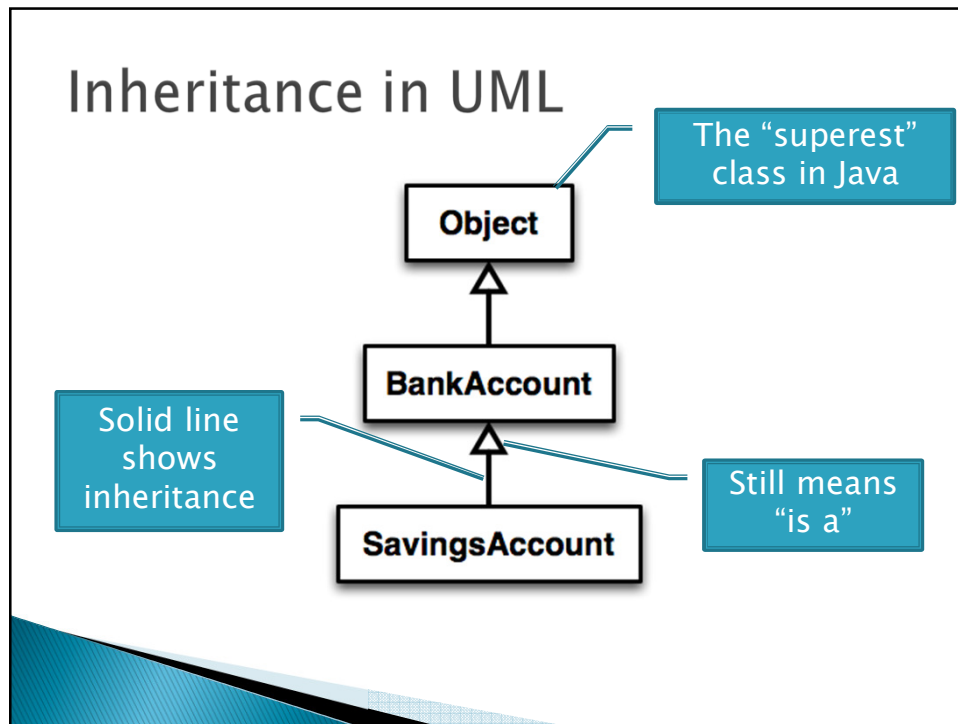
Inheritance

- ▶ Sometimes a new class is a **special case** of the concept represented by another
- ▶ Can “borrow” from an existing class, changing just what we need
- ▶ The new class **inherits** from the existing one:
 - all methods
 - all instance fields



Notation and Terminology

- ▶ **class SavingsAccount extends BankAccount** {
 // added fields
 // added methods
 }
- ▶ Say “SavingsAccount **is a** BankAccount”
- ▶ **Superclass**: BankAccount
- ▶ **Subclass**: SavingsAccount



With Methods, Subclasses can:

- ▶ **Inherit** methods **unchanged**
- ▶ **Override** methods
 - Declare a new method **with same signature** to use **instead of superclass method**
- ▶ **Add** entirely new methods not in superclass

With Fields, Subclasses:

- ▶ **ALWAYS inherit** all fields **unchanged**
- ▶ **Can add** entirely new fields not in superclass

**DANGER! Don't use
the same name as a
superclass field!**

Super Calls

- ▶ Calling superclass **method**:
 - **super.methodName(args);**
- ▶ Calling superclass **constructor**:
 - **super(args);**

Must be the first
line of the subclass
constructor

Access Modifiers

- ▶ **public**—any code can see it
- ▶ **private**—only the class itself can see it
- ▶ **default** (i.e., no modifier)—only code in the same **package** can see it
- ▶ **protected**—like default, but subclasses also have access

I, Object




»» The superest class in Java

Object

- ▶ **Every** class in Java inherits from **Object**
 - Directly and **explicitly**:
 - **public class String extends Object {...}**
 - Directly and **implicitly**:
 - **class BankAccount {...}**
 - **Indirectly**:
 - **class SavingsAccount extends BankAccount {...}**

Q1

Object Provides Several Methods

- ▶ **String toString()** 
- ▶ **boolean equals(Object otherObject)**
- ▶ **Class getClass()** 
- ▶ **Object clone()** 
- ▶ ...

Q2

Overriding toString()

- ▶ Return a concise, human-readable summary of the object state
- ▶ Very useful because it's called automatically:
 - During string concatenation
 - For printing
 - In the debugger
- ▶ **getClass().getName()** comes in handy here...

Q3

Overriding equals(Object o)

- ▶ Should return true when comparing two objects of same type with same “meaning”
- ▶ How?
 - Must check types—use **instanceof**
 - Must compare state—use **cast**
- ▶ Example...

Q4

Polymorphism

»» Review and Practice

Polymorphism and Subclasses

- ▶ A subclass instance **is** a superclass instance
 - Polymorphism still works!
 - **BankAccount ba = new SavingsAccount();**
ba. deposit(100);
- ▶ But not the other way around!
 - **SavingsAccount sa = new BankAccount();**
sa. addInterest();
- ▶ Why not?



BOOM!

Another Example

- ▶ Can use:
 - `public void transfer(double amt, BankAccount o){`
 `this.withdraw(amt);`
 `o.deposit(amt);`
 `}`
 in BankAccount
- ▶ To transfer between different accounts:
 - `SavingsAccount sa = ..;`
 - `CheckingAccount ca = ..;`
 - `sa.transfer(100, ca);`

Summary

- ▶ If B extends or implements A, we can write

`A x = new B();`

Declared type tells which methods x can access.
Compile-time error if try to use method not in A.

The actual type tells which class' version of the method to use.

- ▶ Can cast to recover methods from B:

`((B)x).foo()`

Now we can access all of B's methods too.

If x isn't an instance of B, it gives a run-time error (class cast)

Q5-7, hand in when done, then start reading BallWorlds spec

BallWorlds

- »» • Meet your partner
- Carefully read the requirements and provided code
- Ask questions (instructor and TAs).
- In a few minutes, we'll code Pulsar together

BallWorlds Teams - Section 1

csse220-201220-BW10,ameslc,smithgb
csse220-201220-BW11,koestedj,watterlm
csse220-201220-BW12,harrissa,rujirasl
csse220-201220-BW13,mcculfpe,toorha
csse220-201220-BW14,campbeeg,murphysw
csse220-201220-BW15,conwaygt,swenseen
csse220-201220-BW16,postcn,satchwsm
csse220-201220-BW17,dingx,gartzkds,harbisjs
csse220-201220-BW18,janeiraj,wangl2
csse220-201220-BW19,jacksoam,weirjm

Check out *BallWorlds* from SVN

BallWorlds Teams - Section 2

csse220-201220-BW20, morrista,olsonmc
csse220-201220-BW21,dialkc,piliseal
csse220-201220-BW22,lockarbm,minardar
csse220-201220-BW23,robinsdp,suttonjj
csse220-201220-BW24,riehelp,sanderej
csse220-201220-BW25,mccullwc,kodamach
csse220-201220-BW26,naylorbl,huangz,
csse220-201220-BW27,tuckerme,pearsojw
csse220-201220-BW28,modivr,sternetj
csse220-201220-BW29,faulknks,yuhasem

Check out *BallWorlds* from SVN

BallWorlds Teams - Section 3

csse220-201220-BW31,ruthat,smithnf
csse220-201220-BW32,lucekm,sturgedl
csse220-201220-BW33,coxap,oharace
csse220-201220-BW34,glenngs,timaeudg
csse220-201220-BW35,freemal,mengx
csse220-201220-BW36,wuj,whiteer
csse220-201220-BW37,belkat,oakesja
csse220-201220-BW38,moorejw,bollivbd,cookmj
csse220-201220-BW39,maxwellh,qinz

Check out *BallWorlds* from SVN

BallWorlds Worktime

» Pulsar
» Continue with Mover, etc.

Because this is a challenging assignment, we'll let you turn BallWorlds in before Friday at noon for full credit. If you miss that deadline, you may turn it in by Saturday at 11:59 p.m. for 90% credit.