

CSSE 220 Day 20

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

Check out *Inheritance2* from SVN

Questions?

Inheritance Review

»» A quick recap of last session

Inheritance

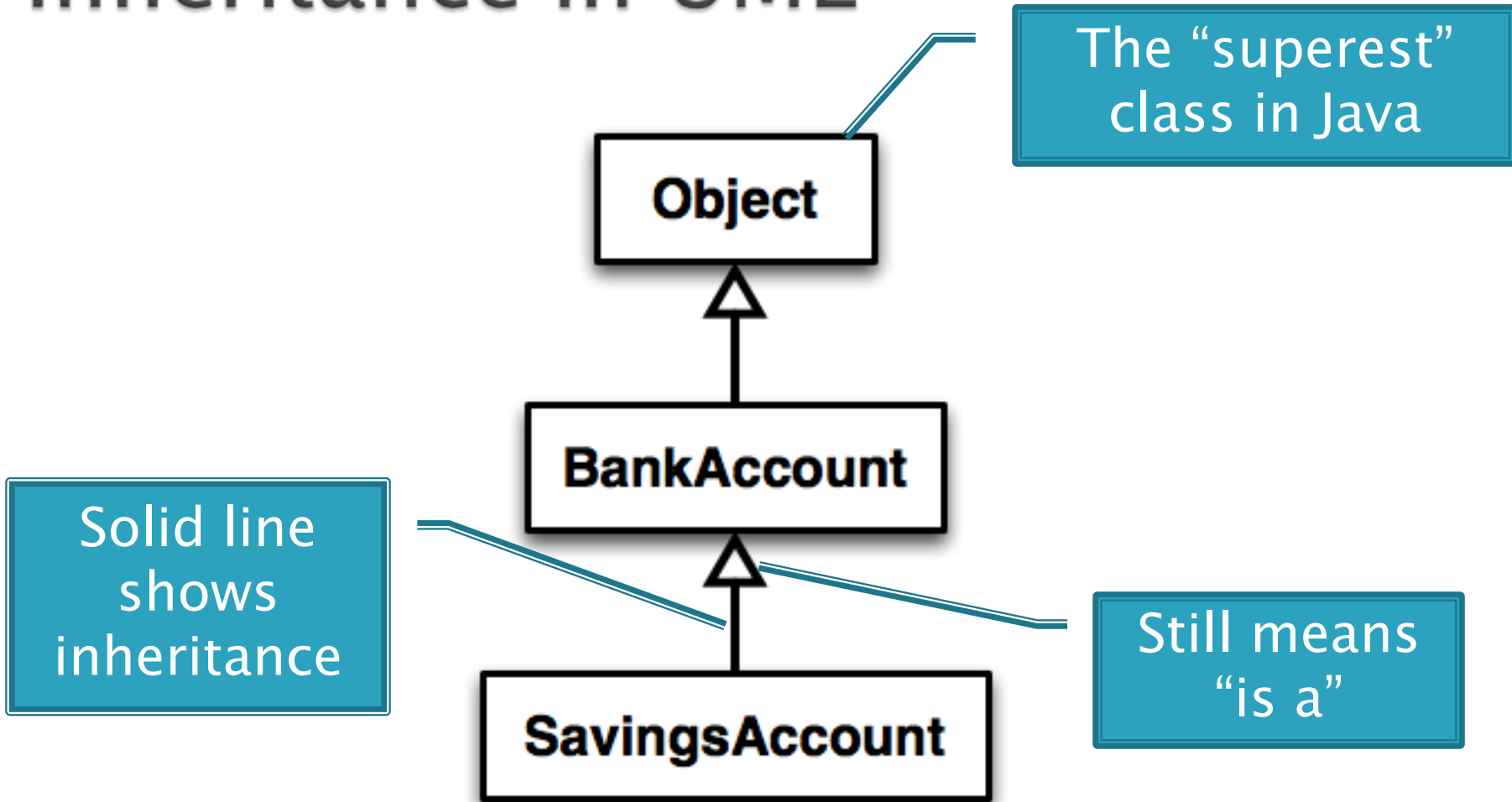
- ▶ Sometimes a new class is a **special case** of the concept represented by another
- ▶ Can “borrow” from an existing class, changing just what we need
- ▶ The new class **inherits** from the existing one:
 - all methods
 - all instance fields



Notation and Terminology

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

Inheritance in UML



The "superest" class in Java

Object


BankAccount

Solid line shows inheritance

SavingsAccount

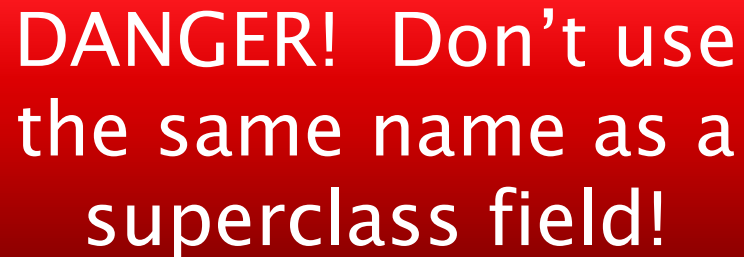
Still means "is a"

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

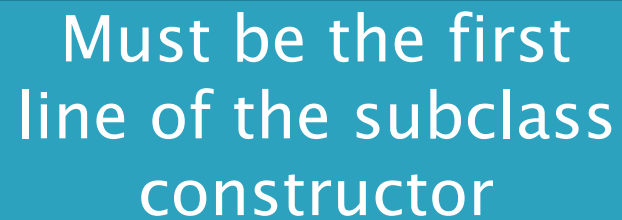


DANGER! Don't use
the same name as a
superclass field!

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
- 

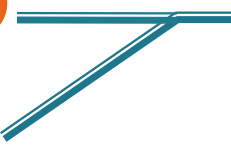


Object

»» 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 {...}**

Object Provides Several Methods

- ▶ **String toString()**  **Often overridden**
- ▶ **boolean equals(Object otherObject)**
- ▶ **Class getClass()**  **Sometimes useful**
- ▶ **Object clone()**  **Often dangerous!**
- ▶ ...

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...

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...

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){
 withdraw(amount);
 o.deposit(amount);
}`

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 exception)

BallWorlds

- »» Meet your partner, then we'll code Pulsar together

BallWorlds Teams – Clifton

n	Team
01	priceha,savrdada
02	agnerri,brooksma
03	mayhewrb,pohltm
04	maglioms,bristokb
05	goodca,schuenjr
06	mouldema,modenejm
07	bippuskw,tugayac,harrisse
08	westeras,veatchje
09	wagnerrj,ryanam
10	kleinnj,petitjam

n	Team
11	dohertjp,czaplikg
12	zellneaj,trederdj
13	
14	
15	
16	
17	

Check out *BallWorlds* from SVN

Team number used in repository name:

<http://svn.csse.rose-hulman.edu/repos/csse220-201120-ballworlds-teamXX>

BallWorlds Teams – Defoe

n	Team
21	handokkr,drakecb
22	hippstn,chenaurj
23	kaiserkp,carrila,redelmrw
24	oelschmm,meyerrd
25	deperarc,grovema
26	coblebj,lockeat
27	galvezdm,scolarrf
28	crouchjt,whiteaj
29	abdelroh,schepedw
30	jacobyam,zhangr1

n	Team
31	trammjn,moyessa
32	raonn,sheltotj
33	mccammjr,chappljd
34	
35	
36	
37	

Check out *BallWorlds* from SVN

Team number used in repository name:

<http://svn.csse.rose-hulman.edu/repos/csse220-201120-ballworlds-teamXX>

BallWorlds Worktime



Pulsar

Complete team pref. survey
before Wednesday 8 a.m.

Continue with Mover, etc.

Because this is a challenging assignment, we'll let you turn BallWorlds in before Friday at 5 p.m. for full credit. If you miss that deadline, you may turn it in by Sunday at 5 p.m. for 80% credit.