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
  }
  ```

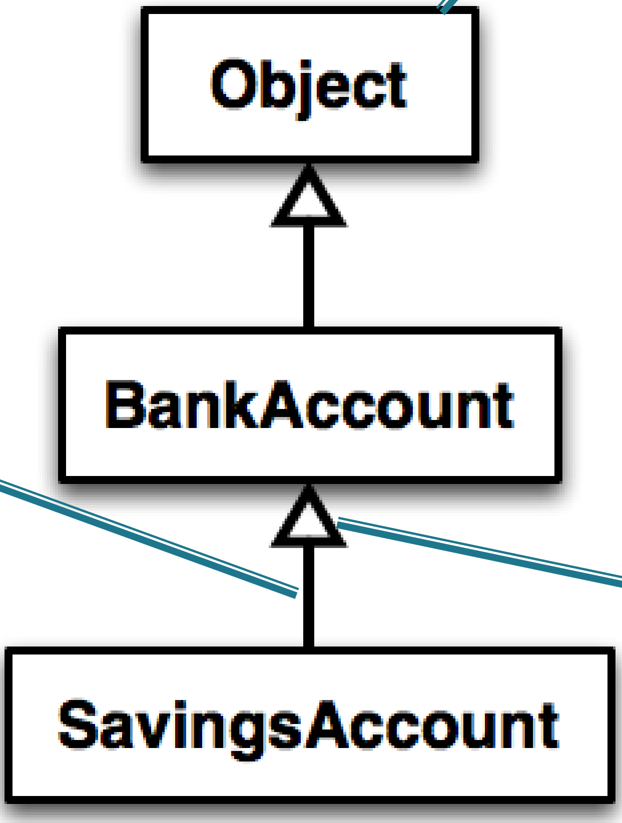
- Say “SavingsAccount **is a** BankAccount”

- **Superclass**: BankAccount

- **Subclass**: SavingsAccount

Inheritance in UML

The "superest" class in Java



Solid line shows inheritance

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

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)**

Often overridden

- **Class getClass()**

Sometimes useful

- **Object clone()**

Often dangerous!

- **…**

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){
        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)

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

BallWorlds

- Meet your partner, then we’ll code Pulsar together

BallWorlds Teams – Clifton

Team number used in repository name:
http://svn.csse.rose-hulman.edu/repos/csse220-201130-ballworlds-teamXX

11,chena1,hirtjd
12,spurrme,nelsonca
13,taos,lyonska
14,filhobc,wilsonam
15,addantnb,shanx
16,cornetcl,caijy
17,luok,eckertzs
18,elswicwj,hopwoocp

Check out *BallWorlds* from SVN

BallWorlds Teams – Defoe

n	Team
21	
22	
23	
24	
25	
26	
27	
28	
29	
30	

n	Team
31	
32	
33	
34	
35	
36	
37	

Check out *BallWorlds* from SVN

Team number used in repository name:
http://svn.csse.rose-hulman.edu/repos/csse220-201130-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.