Inheritance II

More

Inheritance:special cases

Security

CryptoClass - with method **encryptIt**

```
public class Sneaky extends CryptoClass
  public String encrypIt (String s) {
      return s;  //Not encrypted!!! gotcha!
```

How to prevent overriding methods

keyword that prevents things from being changed…

In CryptoClass, declare method

```
public final String encrypIt (String s) {…
```

Inheritance:special cases

final classes

a class that cannot be extended

cannot be used for inheritance

final public class Foo {…

Methods can also be final

final methods cannot be overridden

Inheritance:special cases

Abstract Ideas

"Vehicle" is an idea, but no thing is a pure vehicle

What is the weight of a "vehicle"?   getWeight( ) …???

How does a "vehicle" move?

What does it mean to create an instance of a Vehicle?

If the base class can't properly answer these, then perhaps the base class is an "Abstract class"

Abstract class -vs- concrete class

Abstract class cannot be ***instantiated***

i.e. cannot be created   **AbClass obj = ~~new AbClass( );~~**

Means that you **MUST** subclass it

Inheritance:special cases

Abstract Classes and methods

public abstract class Vehicle {…

abstract void move ( );

Note: no implementation! None! just   **;**

Subclass must implement all abstract methods

Subclass may override non-abstract methods

Although subclass can also be abstract

Inheritance

"Is-A" relationship holds

Child class is more specific than parent class

Commonalities belong higher in hierarchy

Beware of shadowing variables in child class

Each class manages its own data (let super do its job)

Abstract classes as appropriate

If you use inheritance, subclass should (just do it):
override: **equals** (Object obj)
override: **toString ( )**

BTW: Java uses single-inheritance only

Recent Keywords

static

extends

protected

super

final

abstract