Inheritance

March 27, 2012

CMPSCI 121, Spring 2012

Introduction to Problem Solving with Computers

Prof. Learned-Miller

Logistics

- Lots of new assignments

Inheritance

- Cats inherit properties from Animal class.
- Sailboats inherit properties from Boat class.
- Bananas inherit properties from Fruit class.
- All classes inherit properties from Object class.

- Properties include:
  - Constructors
  - Other methods
  - Attributes

Inheritance

- Only goes in one direction:
  - A Cat is an Animal, but
    an Animal is not a Cat.

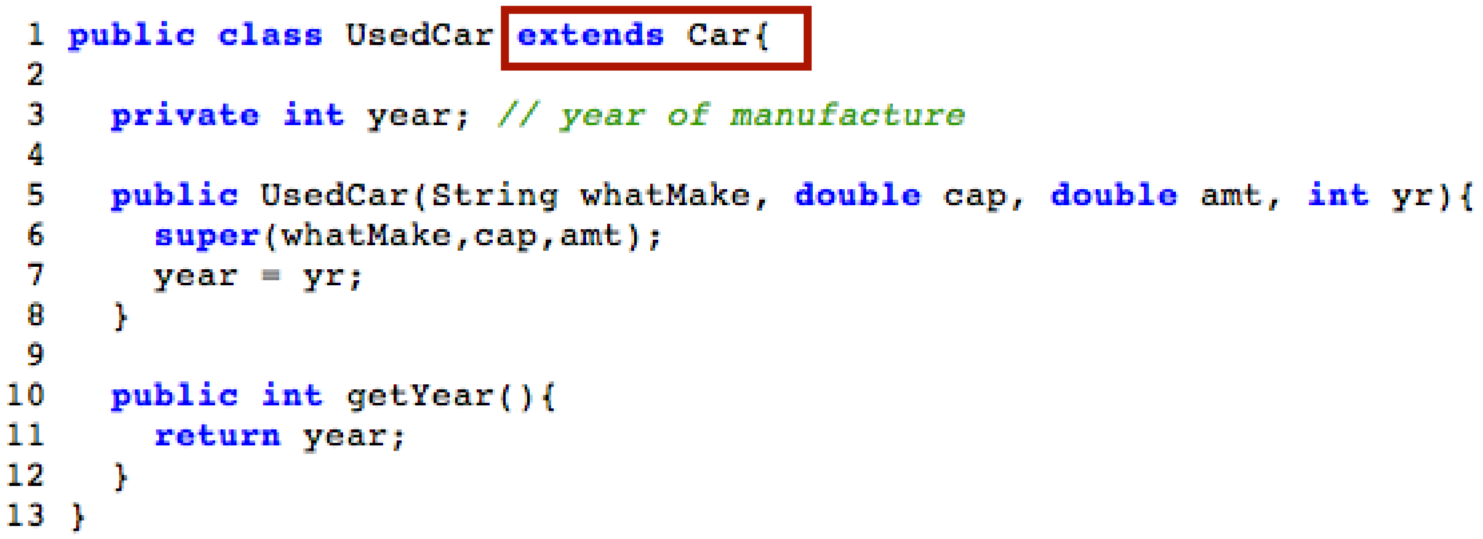
The Car Class

```
public class Car{
  // the Car attributes
  private String make; // manufacturer
  private  double fuelCapacity;
  private  double fuelAmount;
  // the Car constructor
  public Car(String what, double cap, double amt){
    make = what;
    fuelCapacity = cap;
    fuelAmount = amt;
  }
  // the Car methods
  public String getMake(){
    return make;}
  public double getCapacity(){
    return fuelCapacity;}
  public double getFuel(){
    return fuelAmount;}

  public void setFuel(double amt){
    fuelAmount = amt;
  }

  public double unusedCap(){
    return (fuelCapacity - fuelAmount);}

}
```

The UsedCar Class

```java
public class UsedCar extends Car{

  private int year; // year of manufacture

  public UsedCar(String whatMake, double cap, double amt, int yr){
    super(whatMake,cap,amt);
    year = yr;
  }

  public int getYear(){
    return year;
  }
}
```

The UsedCar Class



```java
public class UsedCar extends Car{

  private int year; // year of manufacture

  public UsedCar(String whatMake, double cap, double amt, int yr){
    super(whatMake,cap,amt);
    year = yr;
  }

  public int getYear(){
    return year;
  }
}
```

Some Terminology

- Car is the superclass.
- UsedCar is the subclass.
  - Also called the derived class.
- UsedCar is derived from Car.
- UsedCar extends Car.
- UsedCar inherits methods, attributes, and constructors from Car.

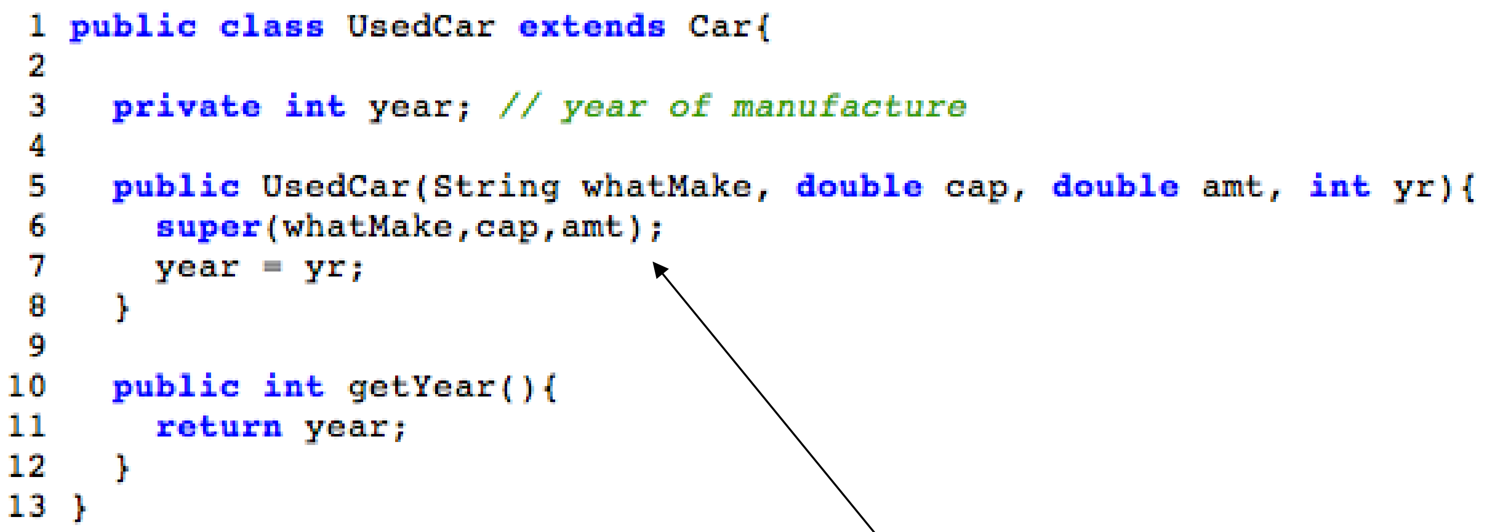
The UsedCar Class

```java
public class UsedCar extends Car{

  private int year; // year of manufacture

  public UsedCar(String whatMake, double cap, double amt, int yr){
    $\boxed{\texttt{super(whatMake,cap,amt);}}$
    year = yr;
  }

  public int getYear(){
    return year;
  }
}
```

“super” keyword

- Kind of like “this” keyword.
- Two different uses:
  - To call superclass constructor.
  - To call a superclass method.

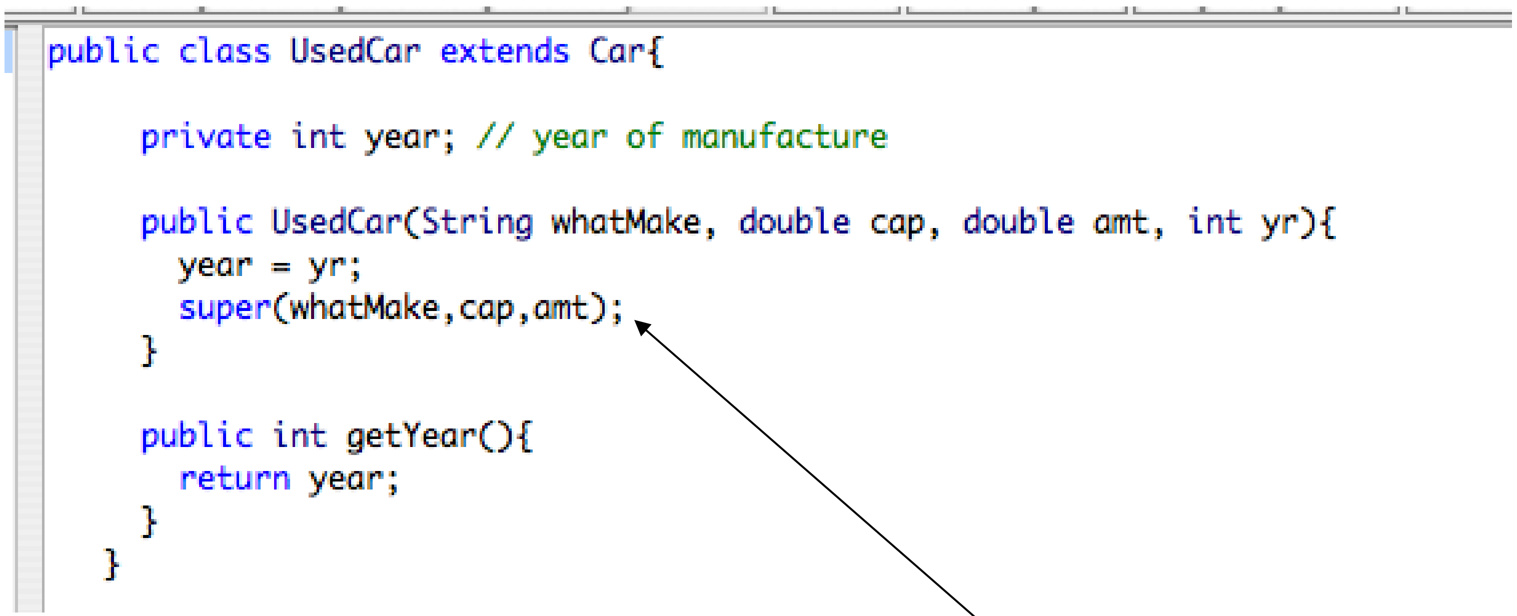
Calling a superclass constructor



```java
public class UsedCar extends Car{

  private int year; // year of manufacture

  public UsedCar(String whatMake, double cap, double amt, int yr){
    super(whatMake,cap,amt);
    year = yr;
  }

  public int getYear(){
    return year;
  }
}
```

Must be first line of constructor method if it is used.

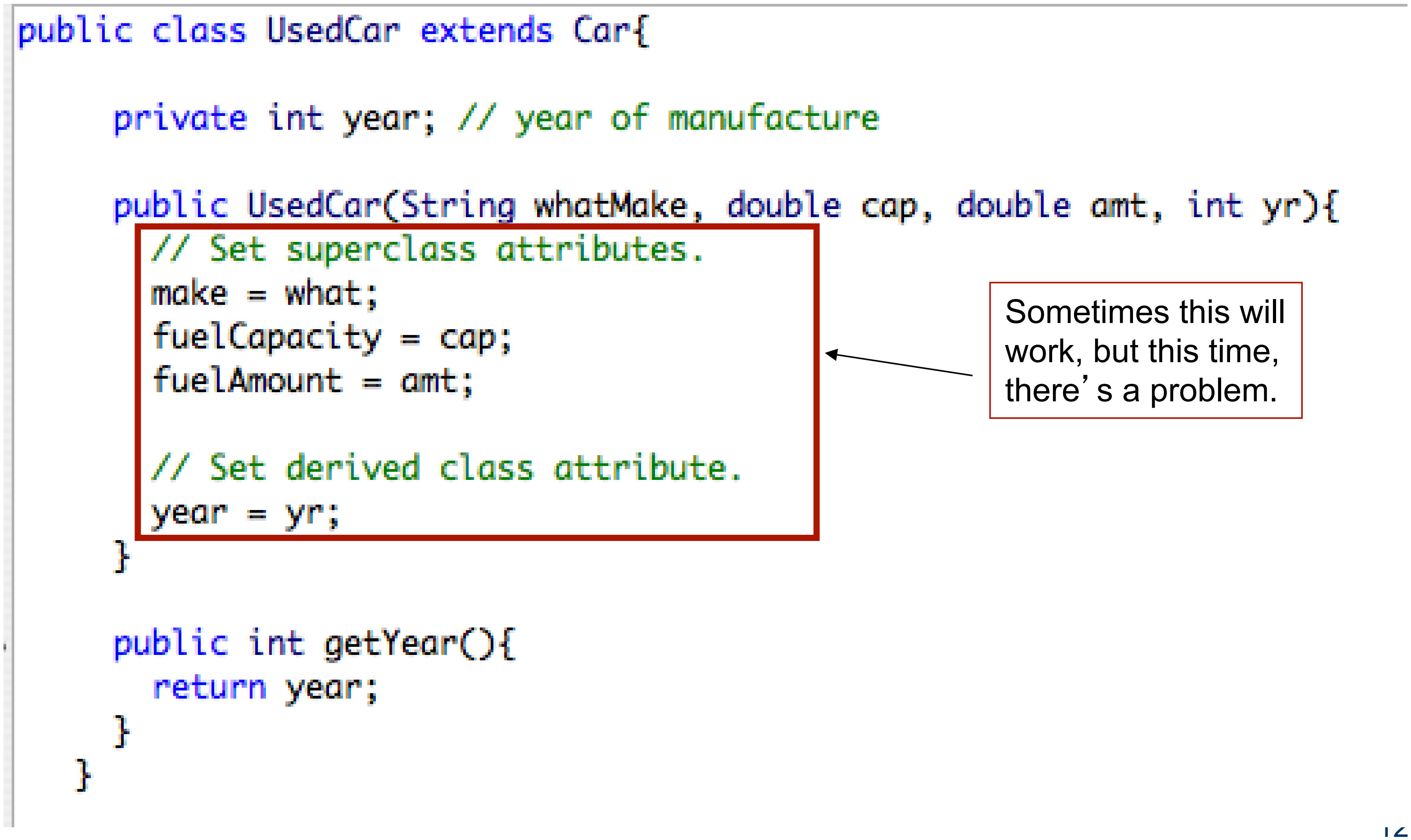
Calling a superclass constructor



```
public class UsedCar extends Car{

    private int year; // year of manufacture

    public UsedCar(String whatMake, double cap, double amt, int yr){
      year = yr;
      super(whatMake,cap,amt);
    }

    public int getYear(){
      return year;
    }
  }
```

Illegal use of super!

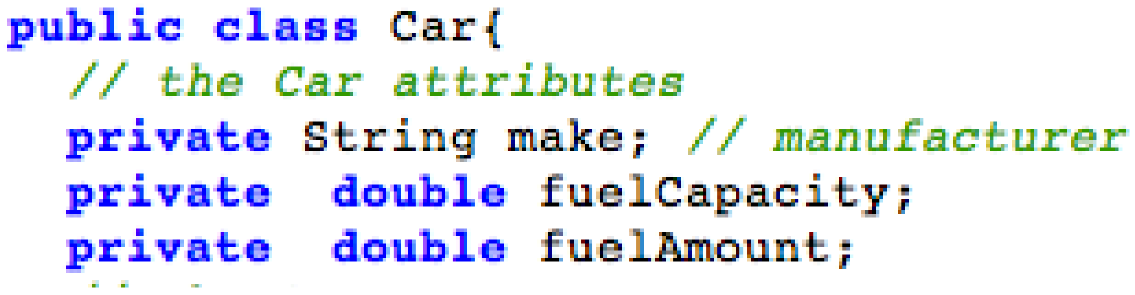
Alternative to using "super"



```
public class UsedCar extends Car{

    private int year; // year of manufacture

    public UsedCar(String whatMake, double cap, double amt, int yr){
      // Set superclass attributes.
      make = what;
      fuelCapacity = cap;
      fuelAmount = amt;

      // Set derived class attribute.
      year = yr;
    }

    public int getYear(){
      return year;
    }
  }
```

Sometimes this will work, but this time, there' s a problem.

Private keyword

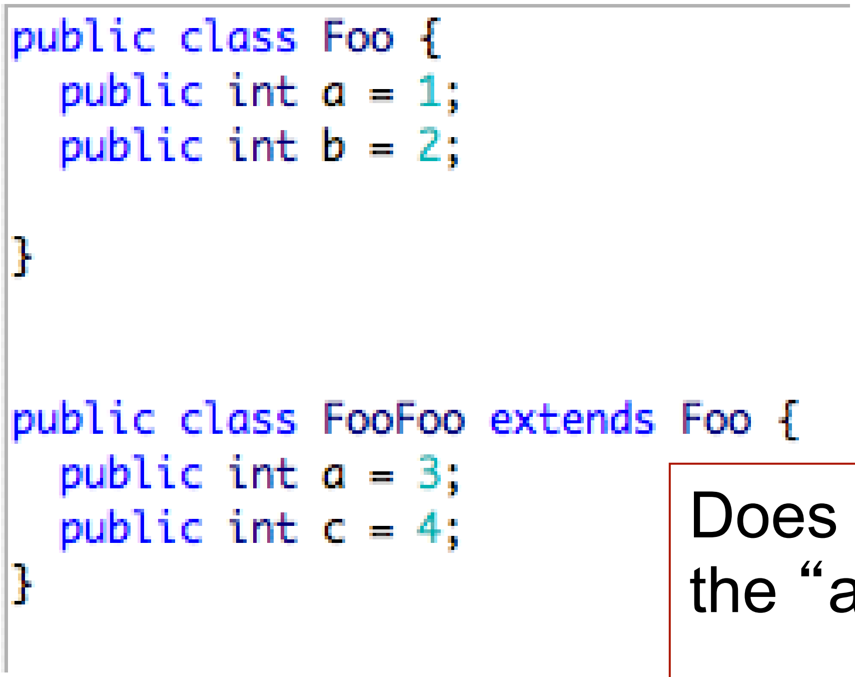


```
public class Car{
   // the Car attributes
   private String make; // manufacturer
   private  double fuelCapacity;
   private  double fuelAmount;
```

Private members of superclass are members of the derived class, BUT...

they cannot be directly accessed by the derived class if they are private.

Hidden attributes or methods



```
public class Foo {
  public int a = 1;
  public int b = 2;

}


public class FooFoo extends Foo {
  public int a = 3;
  public int c = 4;
}
```

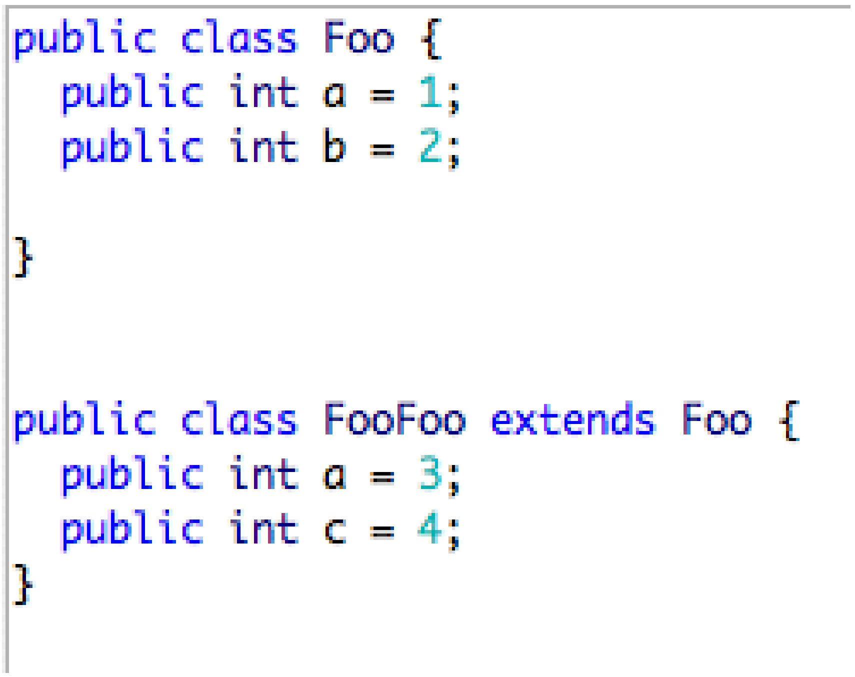
Does the “a” in FooFoo destroy the “a” in Foo?

NO!

But it makes it hard to access.

It hides the attribute.

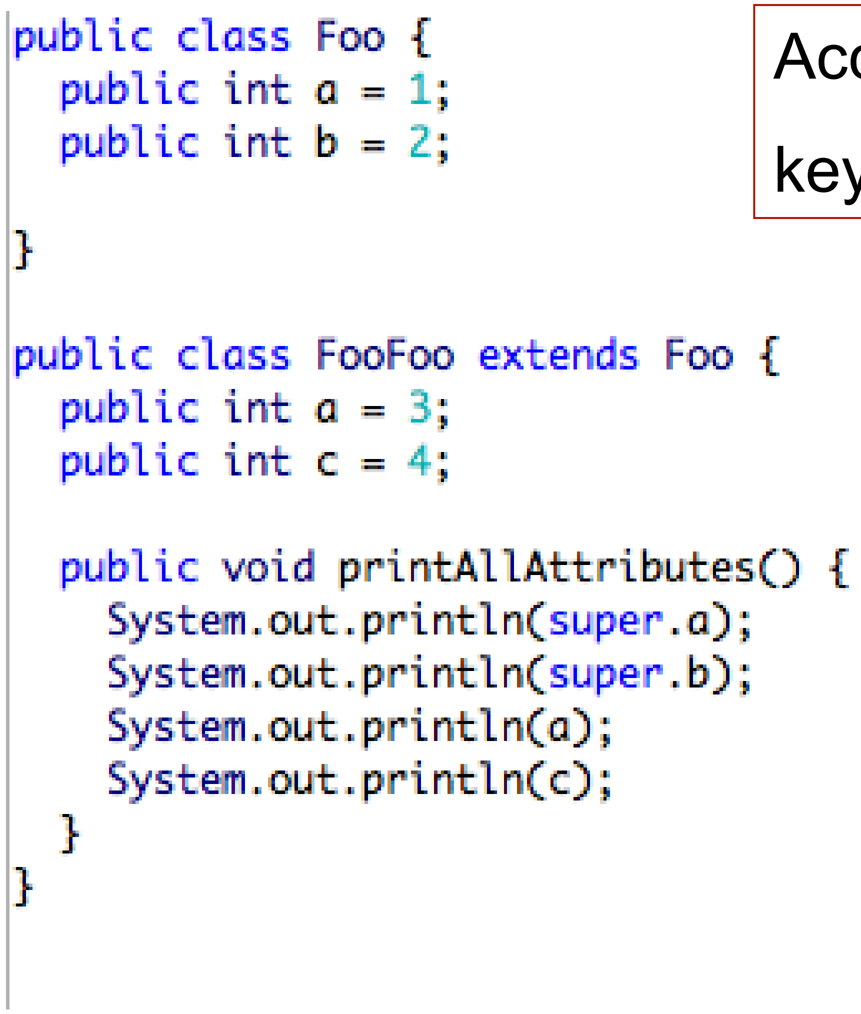
Hidden attributes or methods



```
public class Foo {
  public int a = 1;
  public int b = 2;

}


public class FooFoo extends Foo {
  public int a = 3;
  public int c = 4;
}
```

How to access “a” in superclass from derived class?

Hidden attributes or methods



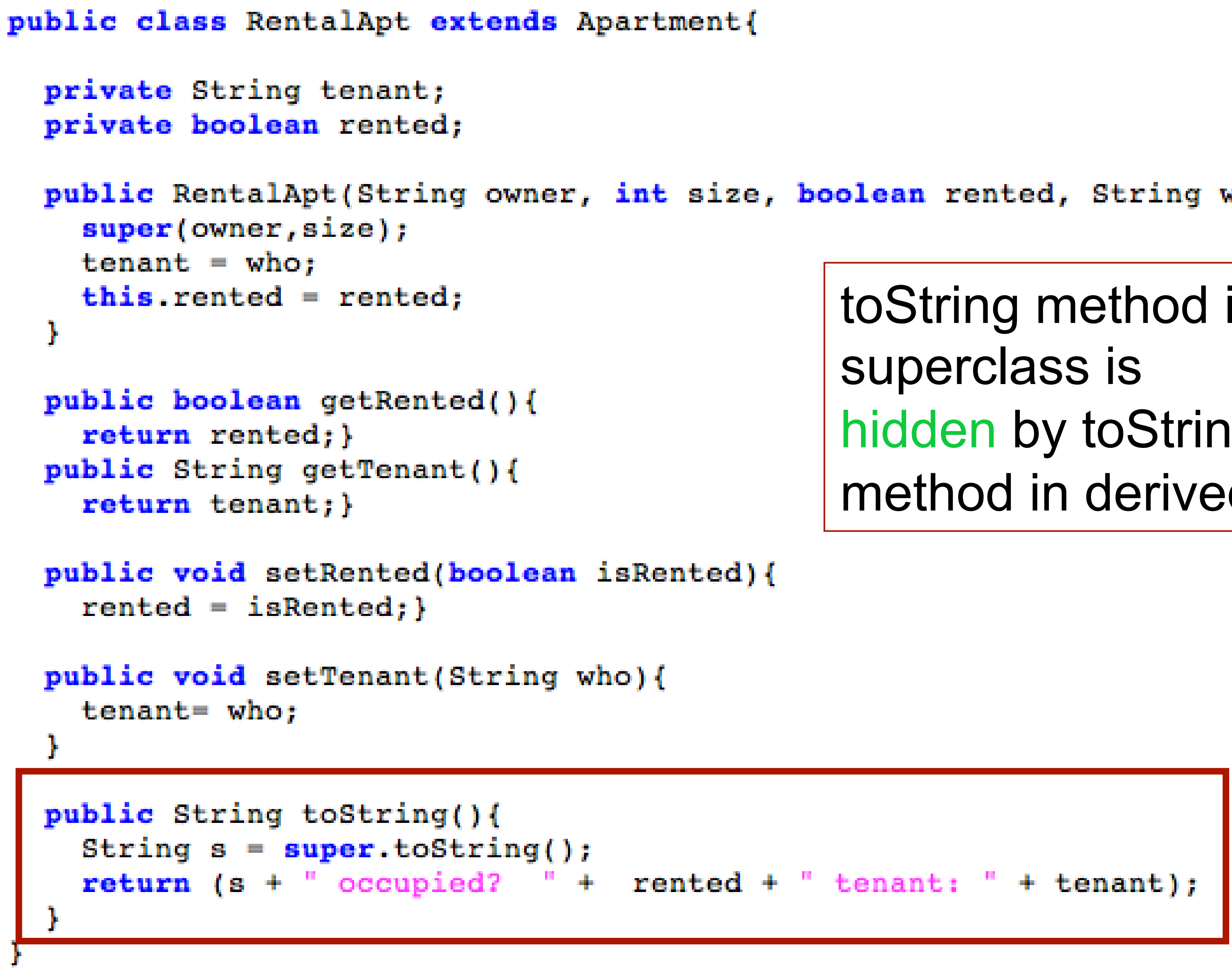
```java
public class Foo {
  public int a = 1;
  public int b = 2;

}

public class FooFoo extends Foo {
  public int a = 3;
  public int c = 4;

  public void printAllAttributes() {
    System.out.println(super.a);
    System.out.println(super.b);
    System.out.println(a);
    System.out.println(c);
  }
}
```

Accessing “a” in superclass with keyword super.

Using Super with Methods

```
public class Apartment{

  private String owner;
  private int size; // square feet

  public Apartment(String owner,int size){
    this.owner = owner;
    this.size = size;
  }

  public int getSize(){
    return size;}
  public String getOwner(){
    return owner;}

  public void setOwner(String newOwner){
    owner = newOwner;
  }

  public String toString(){
    return(owner + " size: " + size);
  }
}
```

Using Super with Methods



```
public class RentalApt extends Apartment{

  private String tenant;
  private boolean rented;

  public RentalApt(String owner, int size, boolean rented, String who){
    super(owner,size);
    tenant = who;
    this.rented = rented;
  }

  public boolean getRented(){
    return rented;}
  public String getTenant(){
    return tenant;}

  public void setRented(boolean isRented){
    rented = isRented;}

  public void setTenant(String who){
    tenant= who;
  }

  public String toString(){
    String s = super.toString();
    return (s + " occupied?  " +  rented + " tenant: " + tenant);
  }
}
```

toString method in superclass is hidden by toString method in derived class.

Java class hierarchy

- All classes derived from one "GrandParent" class. This grandparent class is called the *Object* class.
- This is useful for a bunch of reasons
  - Think of how you use the word "thing" in English. Allows you to talk about many different types of objects at the same time:
    - "Move all those things over here"
    - "Print out all of those things"

Accessing Super-Super class

At the end of class, somebody asked an excellent question:
How do you access the “grandparent” class, i.e. the
superclass of the superclass of a class?

While “super.a” will access the attribute “a” of a superclass (the parent class),
“super.super.a” will NOT access the attribute “a” of the
grandparent class.

So far, the only way I know how to do this is to use a method
in the parent class like this:

super.getParentValueOfa();

where getParentValueOfa() returns the value of “a” in
a class’s superclass.

Please come to office hours if you have questions about this
material, but it is not required for the class.

-Erik