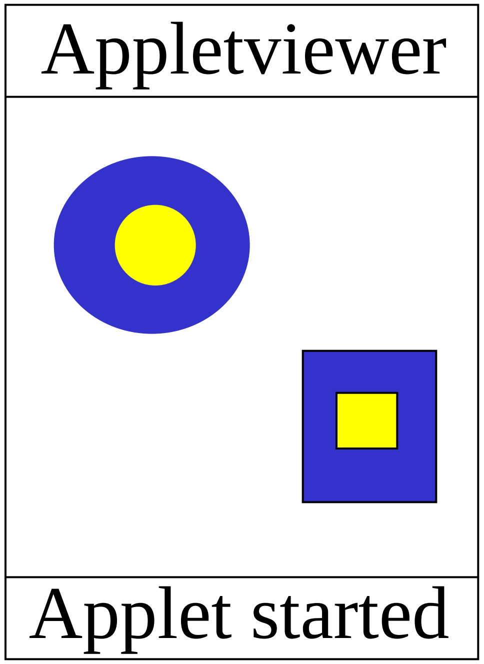
Step Into Java: Inheritance

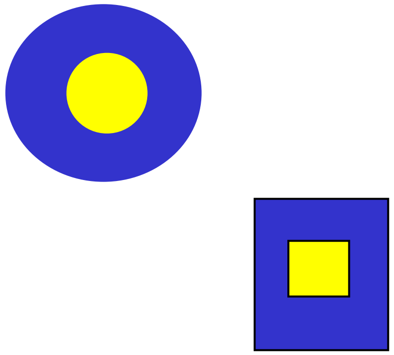
Mr. Neat

Java

What if I wanted to drag either of these objects around the window?



We want to have
a variable in our program
that can hold objects from
different classes.



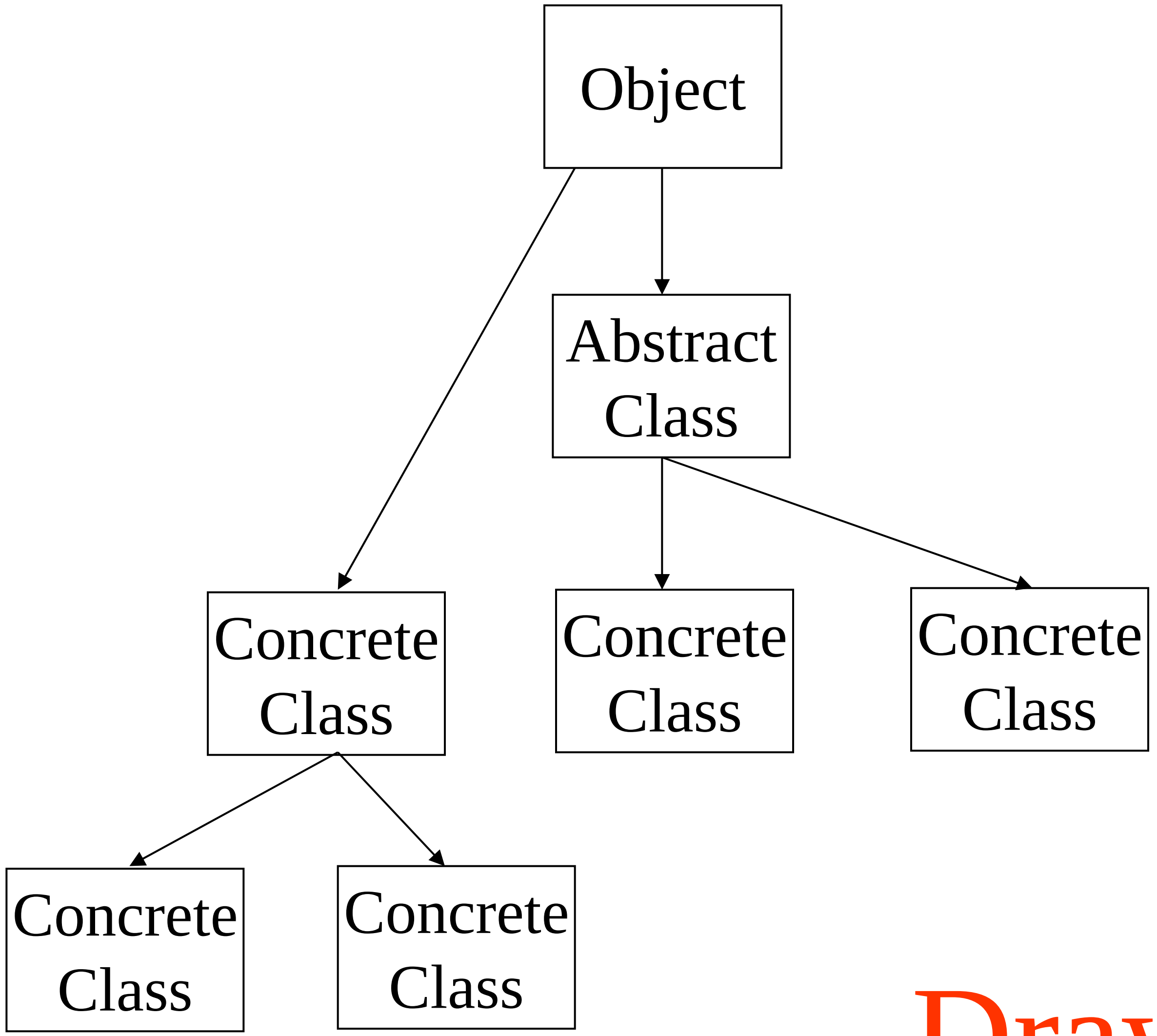
There are many ways to
do this in java. We have done
this already…

We are going to explore *inheritance* to do this next

Inheritance

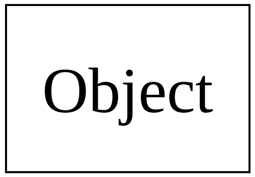
- Some objects in java are similar to others
- Inheritance enables the programmer to extend a class to make a more specialized class.
- The new class has all of the features of the original class, plus the new added features.

Inheritance Family Tree



Draw these!!

Inheritance Family Tree



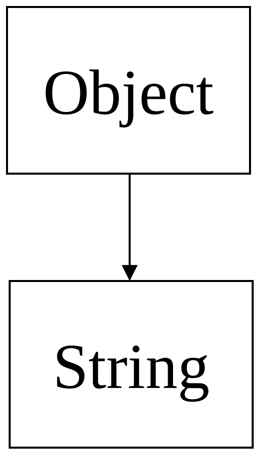
capital

Every object *is-an* Object.

Java phrase

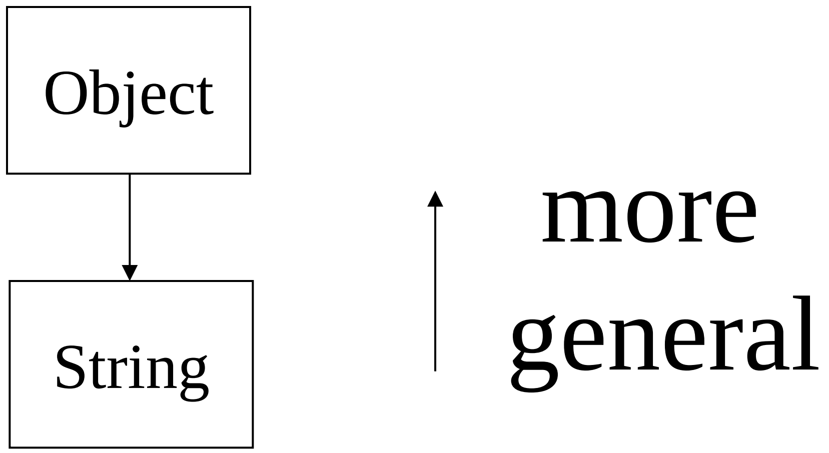
That means we can do this:

```
Object myString = new String("crazy");
```



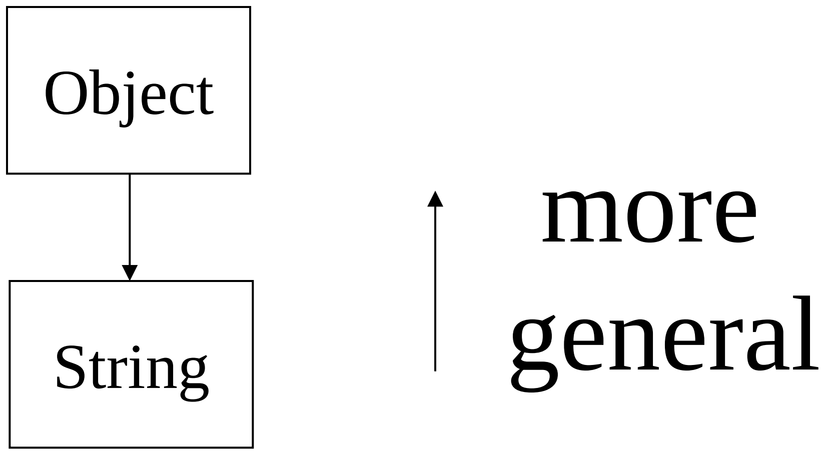
That means we can do this:

Object myString = new String(“crazy”);



```
Object myString = new String(“crazy”);
System.out.println(myString);
```

What would be the output?



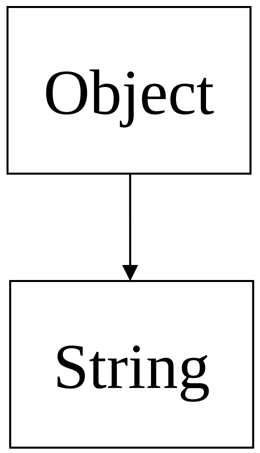
System.out.print calls a class’s toString() method.

If the class, does not have a toString() method, it calls the super class’s toString() method.

If no toString() method exists for any super class, the Object’s toString() method is called.

```
Object myString = new String(“crazy”);
System.out.println(myString);
```

What would be the output?



has a toString() method

has a toString() method

Think of class String extending class Object.

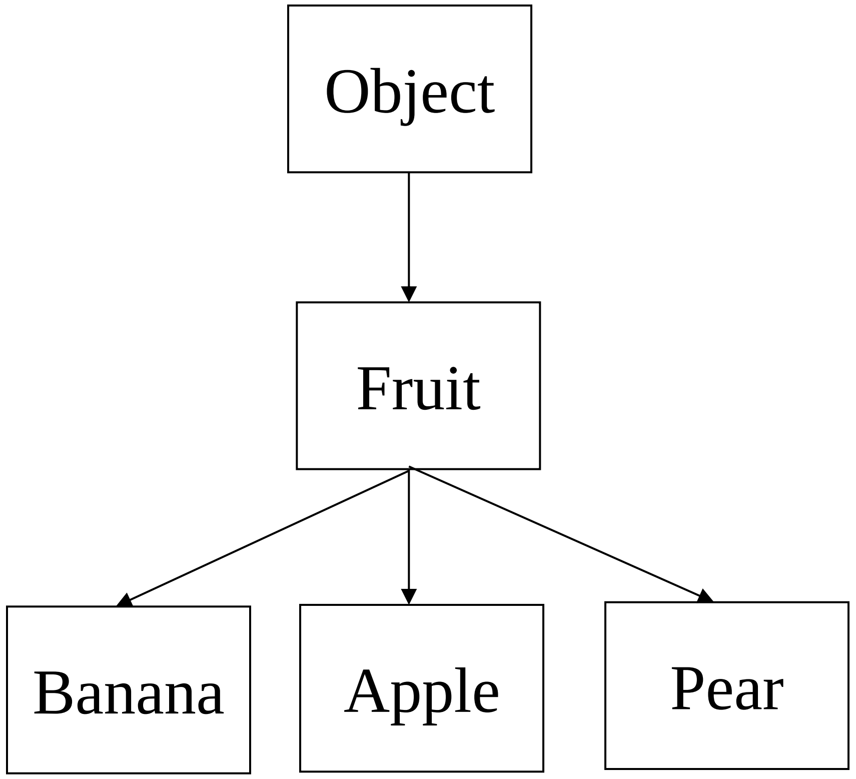
public class String extends Object

-Inherits all Objects private fields
-Inherits all Objects public methods

But

- The extended class does not have access to the super class’s private fields.
- Bummer!

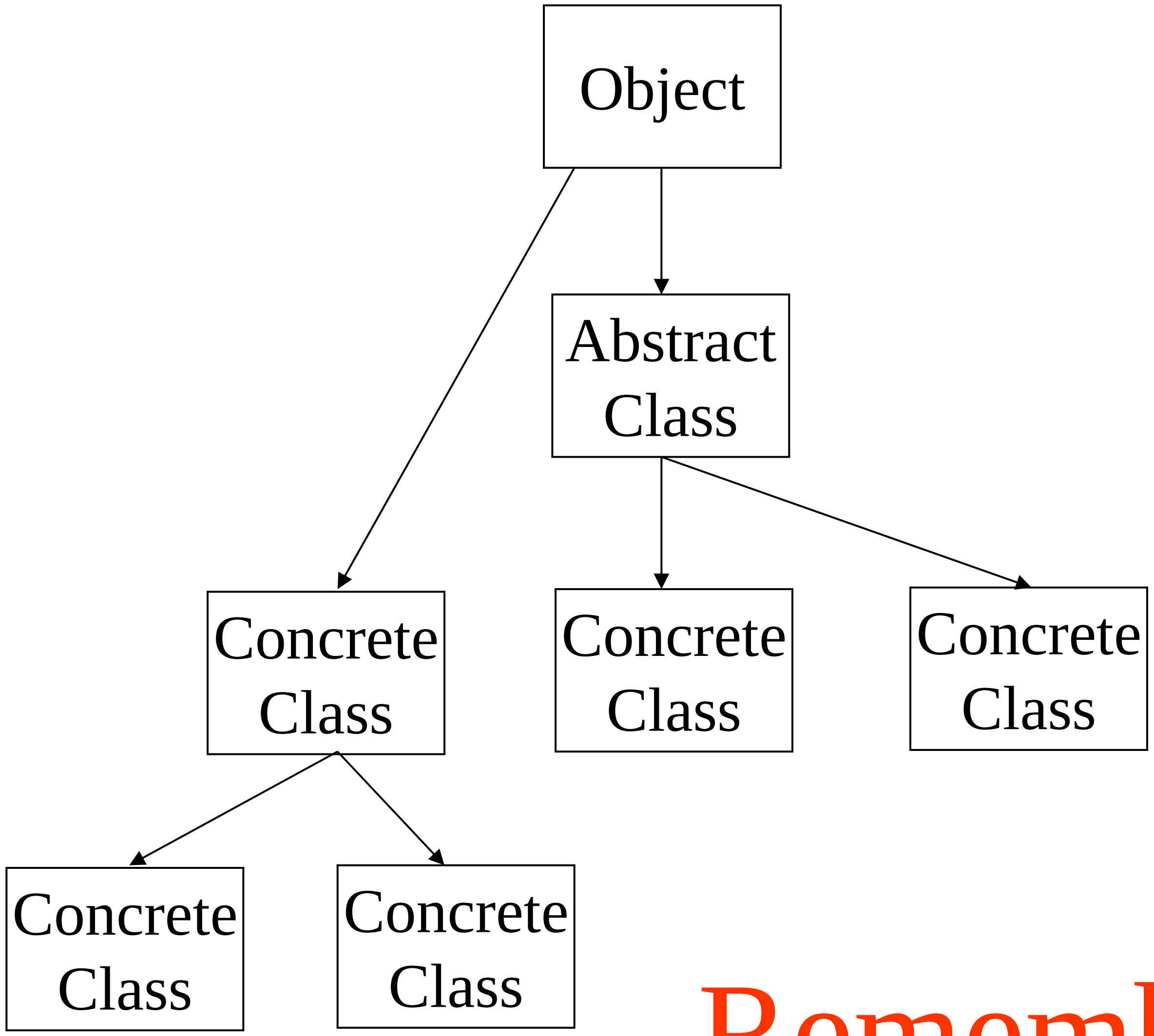
Inheritance Family Tree



Make a class Apple

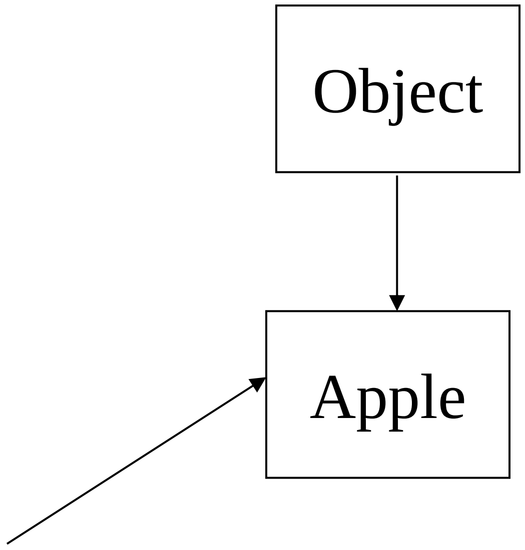
- Include one double field to store the weight of the Apple.
- Include a constructor that has one parameter to initialize the Apple’s private weight field.
- Write a Client program that constructs one Apple object. You choose the weight.

Inheritance Family Tree



Remember this!!

Inheritance Family Tree



- This is a *concrete* class

-Can construct objects of the class

- Sort of like

  public class Apple extends Object

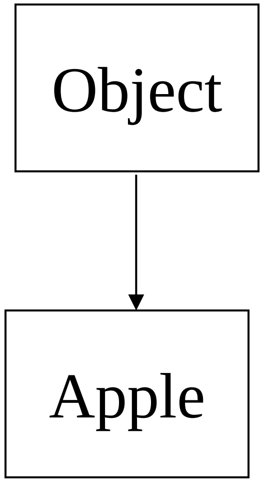
  {

Do this in your Client program:

```
Apple ripe = new Apple(.5);
System.out.println(ripe);
```

What happens?

Inheritance Family Tree

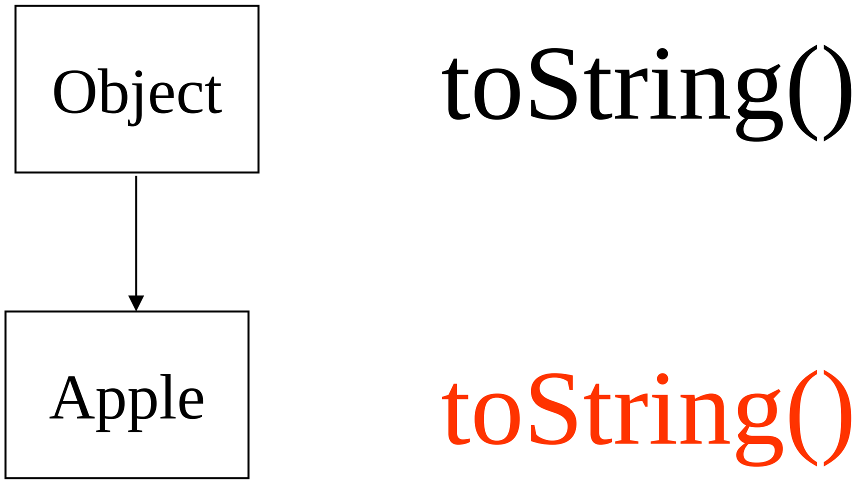


toString()

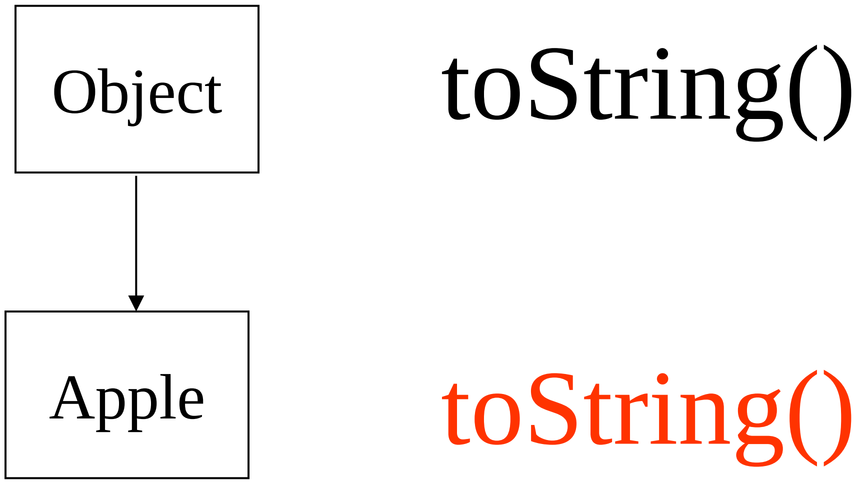
Since the Apple class does not have a toString() method, Java Inheritance calls the super class's toString() method. In this case it is Object's toString() method.

Add a toString() method to the Apple class that says “The weight is: *apple weight*”.

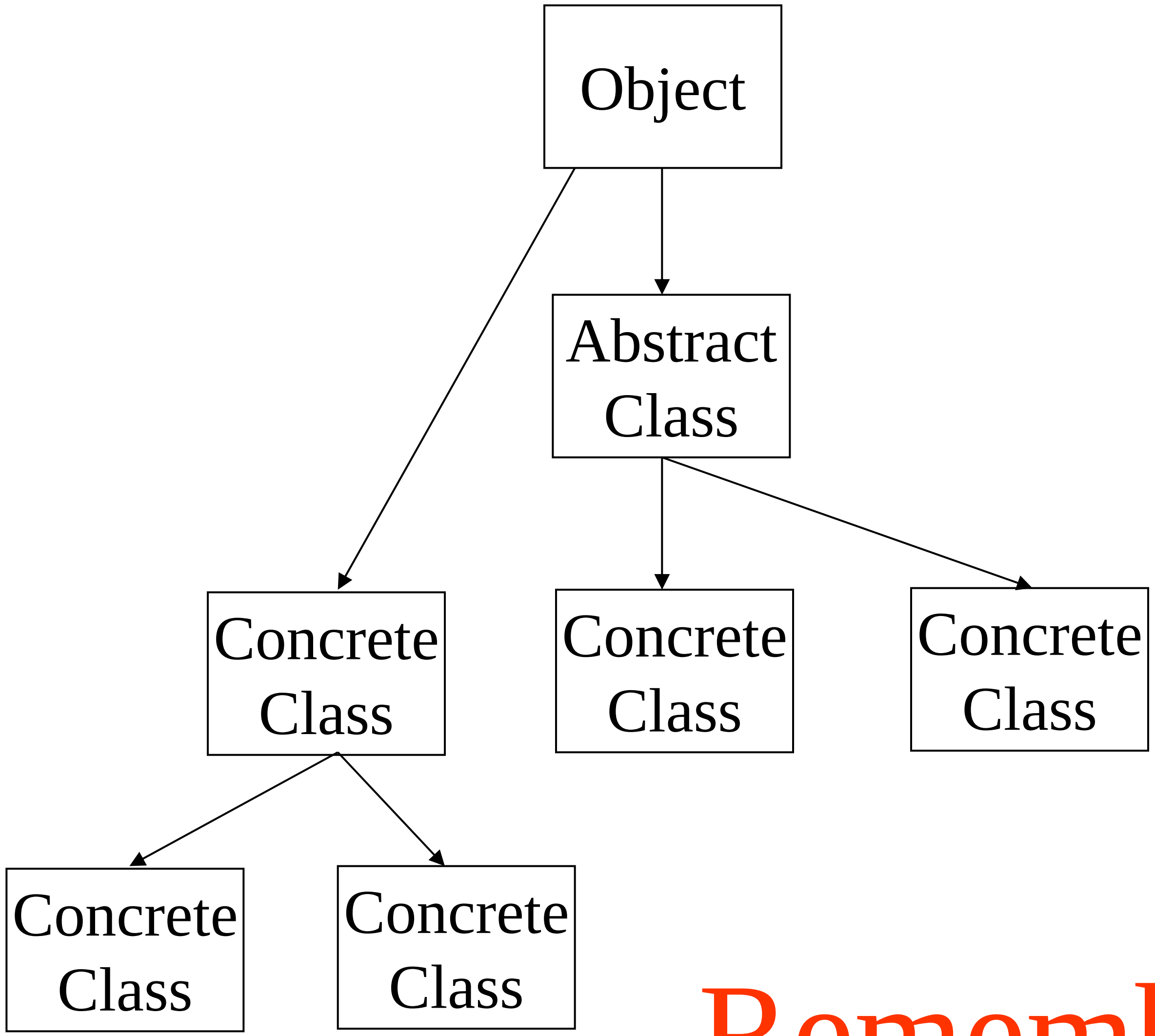
private
weight
field



Change the Apple toString() method to super.toString().

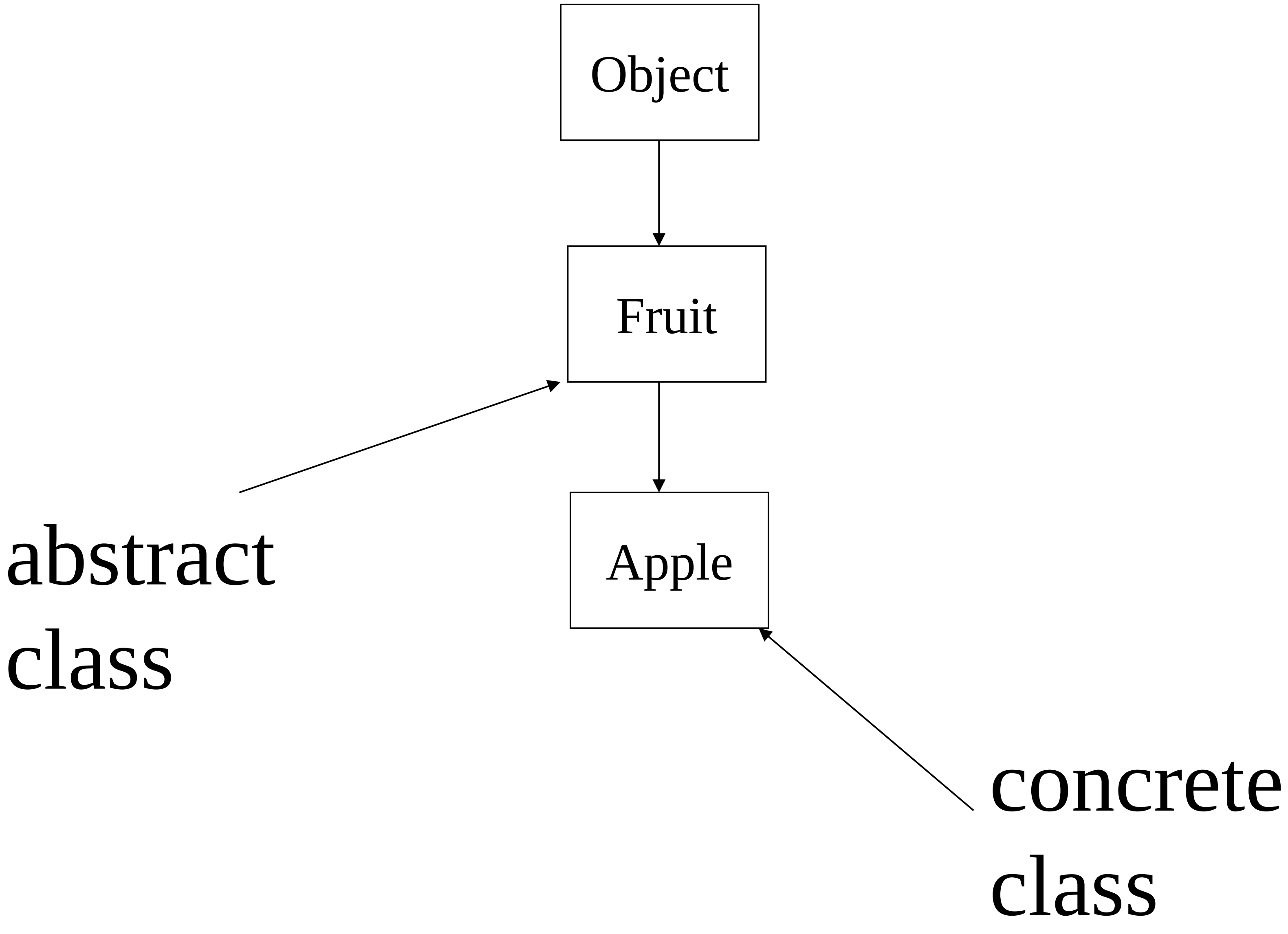


Inheritance Family Tree



Remember this!!

Inheritance Family Tree



Add an abstract class Fruit

- New file
- public abstract class Fruit{ }
- Redefine the Apple class as

  public class Apple extends Fruit{
- In the Client program, construct a Fruit object which is really an Apple object: Fruit foo = new Apple(.3);
- Compile and run

Add a new method getWeight() to Apple class

- Method should return the value of the Apple’s private weight field.
- This is called an accessor method (gives access, but not the ability to change it).

Add an abstract method getWeight() to the Fruit class

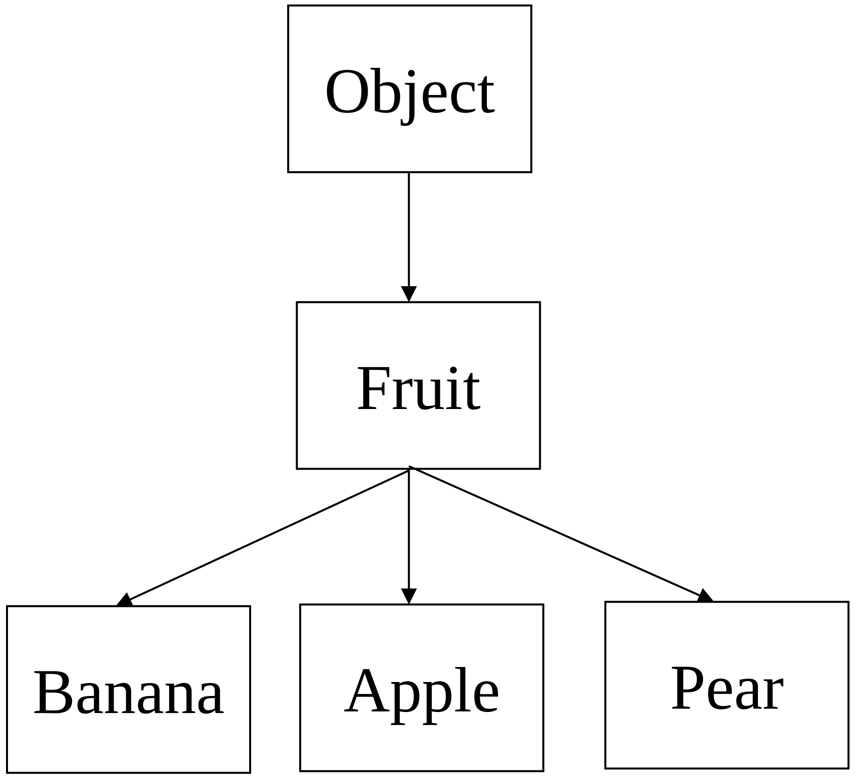
- public abstract double getWeight(); // that's it
- In the Client program, define an
  - Object that is an Apple
  - Fruit object that is an Apple
  - Apple object that is an Apple

System.out.print the weight of each object.

Inheritance Family Tree

$$\begin{tikzcd}
\boxed{\text{Object}} \arrow[d] & \text{no getWeight() method} \\
\boxed{\text{Fruit}} \arrow[d] & \text{abstract getWeight() method} \\
\boxed{\text{Apple}} & \text{A “real” getWeight() method} \\
 & \text{Concrete!} \arrow[u]
\end{tikzcd}$$

Inheritance Family Tree

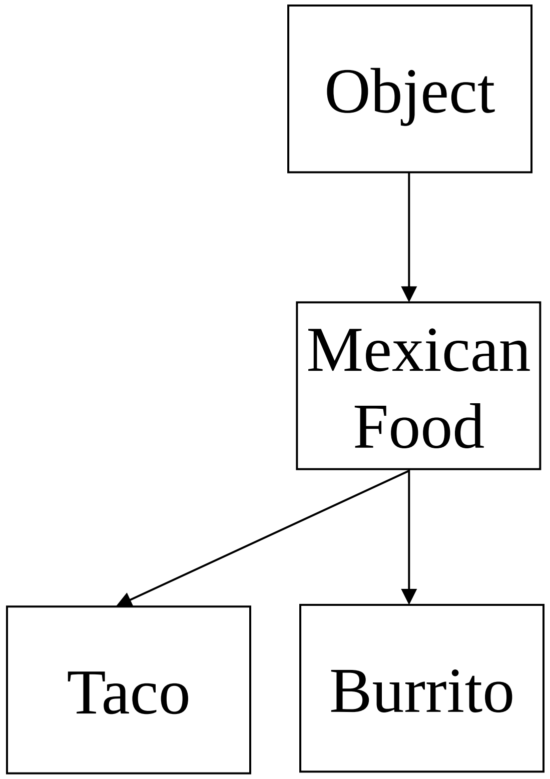


Now This is Possible…

```
Fruit piece1 = new Apple(.2);
Fruit piece2 = new Banana(.35);
Fruit piece3 = new Pear(1.0);

Fruit collection[] = {new Apple(.1), new Banana(.2),
                                new Pear(.3)}
```

Next Lab Drag 2 ______



abstract class

concrete classes

Next Lab Drag 2 ______

- drag your Taco and your Burrito objects
 around the window
- Use the Mexican Food classes defined in the previous labs
-Both the Taco object and the Burrito
object must extend the Mexican Food abstract class
-Define an abstract move method in the abstract Mexican Food class.