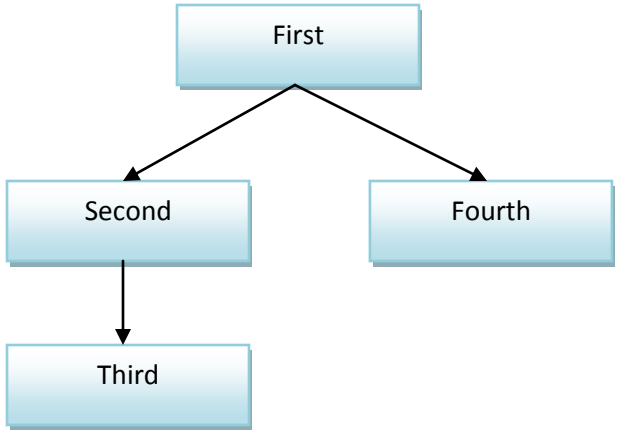
```
public class First {
    public void method2() {
        System.out.println("First2");
    }

    public void method3() {
        method2();
    }
}

public class Second extends First {
    public void method2() {
        System.out.println("Second2");
    }
}

public class Third extends Second {
    public void method1() {
        System.out.println("Third1");
        super.method2();
    }

    public void method2() {
        System.out.println("Third2");
    }
}

public class Fourth extends First {
    public void method1() {
        System.out.println("Fourth1");
    }

    public void method2() {
        System.out.println("Fourth2");
    }
}
```

Given the previous, draw the inheritance hierarchy starting at the top (the thing which doesn't extend anything) and then placing anything which extends it directly below it and so on. So for what is on the left, you have…



Then you build a table of methods from the top of the tree on down. If you call "super.<somemethod>()", then look at the inheritance tree for the super class and copy whatever it's <somemethod>() does into your cell of the table. If your cell says <method>() (note the lack of super.), then just write "<method>()" in the cell so if it's inherited, you do the correct thing.

Class	method1	method2	method3
First	-----	"First2"	method2()
Second	-----	"Second2"	method2()
Third	"Third1" "Second2"	"Third2"	method2()
Fourth	"Fourth1"	"Fourth2"	method2()

Remember that when executing a method for ObjectType, execute only the methods of ObjectType. For example, if the ObjectType Third, then a call to its method3() will call Third's method3.

First we define a few things with a color code

```
DeclaredType name = new ObjectType();    //declare variable
name.method();                           //call method
((CastToType)name).method();             //cast object, then call a method
```

When we try to execute one of the latter two, we follow this progression: