CSE143 Inheritance Example

Assuming that the following classes have been defined:

```
public class One {
    public void method1() {
        System.out.println("One1");
    }
}

public class Two extends One {
    public void method3() {
        System.out.println("Two3");
    }
}

public class Three extends One {
    public void method2() {
        System.out.println("Three2");
        method1();
    }
}

public class Four extends Three {
    public void method1() {
        System.out.println("Four1");
        super.method1();
    }
    public void method3() {
        System.out.println("Four3");
    }
}
```

And assuming the following variables have been defined:

```
One var1 = new Two();
One var2 = new Three();
One var3 = new Four();
Three var4 = new Four();
Object var5 = new Three();
Object var6 = new One();
```

In the table below, indicate in the right-hand column the output produced by the statement in the left-hand column. If the statement produces more than one line of output, indicate the line breaks with slashes as in "a/b/c" to indicate three lines of output with "a" followed by "b" followed by "c". If the statement causes an error, fill in the right-hand column with either the phrase "compiler error" or "runtime error" to indicate when the error would be detected.

Statement	Output
`var1.method1();`	_____
`var2.method1();`	_____
`var3.method1();`	_____
`var4.method1();`	_____
`var5.method1();`	_____
`var6.method1();`	_____
`var4.method2();`	_____
`var4.method3();`	_____
`((Two)var1).method2();`	_____
`((Three)var1).method2();`	_____
`((Two)var1).method3();`	_____
`((Four)var2).method1();`	_____
`((Four)var3).method1();`	_____
`((Four)var4).method3();`	_____
`((One)var5).method1();`	_____
`((Four)var5).method2();`	_____
`((Three)var5).method2();`	_____
`((One)var6).method1();`	_____
`((One)var6).method2();`	_____
`((Two)var6).method3();`	_____

CSE143 Inheritance Example
Solution

Call	Output	Discussion
var1.method1();	One1	variable is of type One, One role includes method1, no cast, actual object is a Two which writes out "One1" when method1 is called
var2.method1();	One1	variable is of type One, One role includes method1, no cast, actual object is a Three which writes out "One1" when method1 is called
var3.method1();	Four1/One1	variable is of type One, One role includes method1, no cast, actual object is a Four which writes out "Four1/One1" when method1 is called
var4.method1();	Four1/One1	variable is of type Three, Three role includes method1, no cast, actual object is a Four which writes out "Four1/One1" when method is called
var5.method1();	compiler error	variable is of type Object, Object role does not include method1
var6.method1();	compiler error	variable is of type Object, Object role does not include method1
var4.method2();	Three2/Four1/One1	variable is of type Three, Three role includes method2, no cast, actual object is a Four which writes out "Three2/Four1/One1" when method2 is called (note that method2 calls its method1 polymorphically, which is why this output includes "Four1")
var4.method3();	compiler error	variable is of type Three, Three role does not include method3 (even though the object itself is a Four that is capable of performing this action)
((Two)var1).method2();	compiler error	because of cast we pay attention to it rather than the variable type (because we have renegotiated the contract), cast is to Two, Two role does not include method2
((Three)var1).method2();	runtime error	cast is to Three, Three role includes method2 so we pass the compiler, but actual object is a Two which can't fill the Three role (casting across the hierarchy, like asking someone to accept a bike when they were expecting a car), so we get a runtime error
((Two)var1).method3();	Two3	cast is to Two, Two role includes method3, actual object is a Two which cal fill the Two role, so the cast is okay, and a Two object writes "Two3" when its method3 is called
((Four)var2).method1();	runtime error	cast is to Four, Four role includes method1, actual object is a Three which can't fill the Four role; this was, in essence, a stupid cast to do because it isn't necessary, but if you tell this kind of lie, Java will complain
((Four)var3).method1();	Four1/One1	cast is to Four, Four role includes method1, actual object is a Four which can fill the Four role, so cast is okay and a Four object writes "Four1/One1" when method1 is called
((Four)var4).method3();	Four3	cast is to Four, Four role includes method3, actual Object is a Four, which can fill the Four role, so cast is okay and a Four object writes "Four3" when method3 is called
((One)var5).method1();	One1	cast is to One, One role includes method1, actual object is a Three, which can fill the One role, so cast is okay and a Three object writes "One1" when method1 is called
((Four)var5).method2();	runtime error	cast is to Four, Four role includes method2, actual object is a Three which can't fill the Four role
((Three)var5).method2();	Three2/One1	cast is to Three, Three role includes method2, actual object is a Three which can fill the Three role and a Three object writes "Three2/One1" when method2 is called
((One)var6).method1();	One1	cast is to One, One role includes method1, actual object is a One which can fill the One role, so cast is okay and a One object writes "One1" when method1 is called
((One)var6).method2();	compiler error	cast is to One, One role does not include method2
((Two)var6).method3();	runtime error	cast is to Two, Two role includes method3, actual object is a One, which can't fill the Two role