```
//父类
class Foo {
    int i = 1;

    Foo() {
        System.out.println(i);              ----------(1)
        int x = getValue();
        System.out.println(x);              ----------(2)
    }

    {
        i = 2;
    }

    protected int getValue() {
        return i;
    }
}

//子类
class Bar extends Foo {
    int j = 1;

    Bar() {
        j = 2;
    }

    {
        j = 3;
    }

    @Override
    protected int getValue() {
        return j;
    }
```

```
}

public class ConstructorExample {
    public static void main(String... args) {
        Bar bar = new Bar();
        System.out.println(bar.getValue());              ----------(3)
    }
}/* Output:
            2
            0
            2
 *///:~



 //Foo类构造函数的等价变换：
    Foo() {
        i = 1;
        i = 2;
        System.out.println(i);
        int x = getValue();
        System.out.println(x);
    }

    //Bar类构造函数的等价变换
    Bar() {
        Foo();
        j = 1;
        j = 3;
        j = 2
    }
```

(2)处输出是0，为什么呢？因为在执行Foo的构造函数的过程中，由于Bar重载了Foo中的getValue方法，所以根据Java的多态特性可以知道，其调用的getValue方法是被Bar重载的那个getValue方法。但由于这时Bar的构造函数还没有被执行，因此此时j的值还是默认值0，因此(2)处输出是0。