

CS2110: Hiding static methods

Static methods can be redeclared in a subclass, but the effect is not overriding but *hiding*. In the program shown below, placing the annotation `@Override` on method `m` in subclass `S` will result in a syntactic error and the program will not compile.

We illustrate hiding of static methods with an example. Calling method `Test.main`, given below, results in this output, which we discuss.

```
C.m called
C.m called
S.m called
C.m called
```

In method `Test.main`, first an object of class `C` is created and (a pointer to it is) stored in `c`. The result is shown below, with variable `c` containing a pointer to the new object.

Then, method `m` is called *in the preferred way*, using class name `C`: `C.m()`. Then, `m` is called using `c.m()`. By the inside-out rule, one looks in object `C@4` for method `m()`, then in the enclosing scope, where method `m()` is found. You see in the output shown above that method `m` in class `C` was called twice.

Next, an object of subclass `S` is created and stored in variable `s` whose type is `C`. The result is shown below, with variable `s` pointing to object `S@60`.

Then, method `S.m` is called in the preferred way, using class name `S`: `S.m()`.

The next call, using `s.m()`, illustrates that overriding does *not* happen. According to the overriding rule, method `m` declared in subclass `S` should be called, but it is not! Instead, since the type of variable `s` is `C`, the static method declared in class `C` is called, resulting in “C.m called” being printed.

So, in this special case of hiding a static method (or variable), the type of the pointer to the object dictates which static method is to be called.

```
public class Test {
    public static void main(String args[]) {
        C c= new C();
        C.m();
        c.m();

        C s = new S();
        S.m();
        s.m();
    }
}

class C {
    public static void m() {
        System.out.println("C.m called");
    }
}

class S extends C {
    public static void m() {
        System.out.println("S.m called");
    }
}
```

