A use of "this."

The expression **this** in a method evaluates to a pointer (or the name of) the object in which it occurs. Therefore, the following expression refers to field *f* of the object:

this.*f*

Generally, we suggest using "**this**." *only* when it necessary. Don't clutter up a program with useless stuff.

We now show a case where it is necessary. Consider the class and constructor shown below. The constructor is supposed to store its parameter *name* in field *name*. However, by the inside-out rule, both occurrences of *name* in the assignment statement refer to parameter *name*. This assignment statement simply copies the value in parameter *name* and stores it back in parameter *name*.

Here's some terminology: Parameter *name shadows* field *name*; it prevents referencing field *name* directly.

```
public class Person {
    String name;    // name of the person

    /** Constructor: an instance with name name. */
    public Person(String name) {
        name= name;   // this does not work!
    }
}
```

The class given below solves the problem, using "**this**.".

```
public class Person {
    String name;    // name of the person

    /** Constructor: an instance with name name. */
    public Person(String name) {
        this.name= name;   // this works!
    }
}
```

Thus, use "**this**.*f*" when field *f* has been shadowed by a declaration of *f* in some method, either as a parameter or as a local variable.

Note that there is *never* a need in Java to use "**this**." in a method call, as in

```
this.m(5).      // Please don't do this!
```

because method names cannot be shadowed —one cannot declare a method within a method in Java.