How NOT to Win Friends and Influence Objects

Okay, but how do you keep from doing this? The principle provides some guidelines: take any object, and from any method in that object, invoke only methods that belong to:

- The object itself
- Objects passed in as a parameter to the method
- Any object the method creates or instantiates
- Any components of the object

Notice that these guidelines tell us not to call methods on objects that were returned from calling other methods!!

Think of a "component" as any object that is referenced by an instance variable. In other words, think of this as a HAS-A relationship.

This sounds kind of stringent, doesn't it? What's the harm in calling the method of an object we get back from another call? Well, if we were to do that, then we'd be making a request of another object's subpart (and increasing the number of objects we directly know). In such cases, the principle forces us to ask the object to make the request for us; that way, we don't have to know about its component objects (and we keep our circle of friends small). For example:

Without the Principle

```
public float getTemp() {
    Thermometer thermometer = station.getThermometer();
    return thermometer.getTemperature();
}
```

Here we get the thermometer object from the station and then call the getTemperature() method ourselves.

With the Principle

```
public float getTemp() {
    return station.getTemperature();
}
```

When we apply the principle, we add a method to the Station class that makes the request to the thermometer for us. This reduces the number of classes we're dependent on.