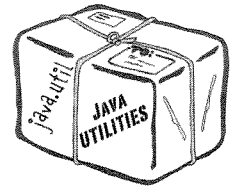
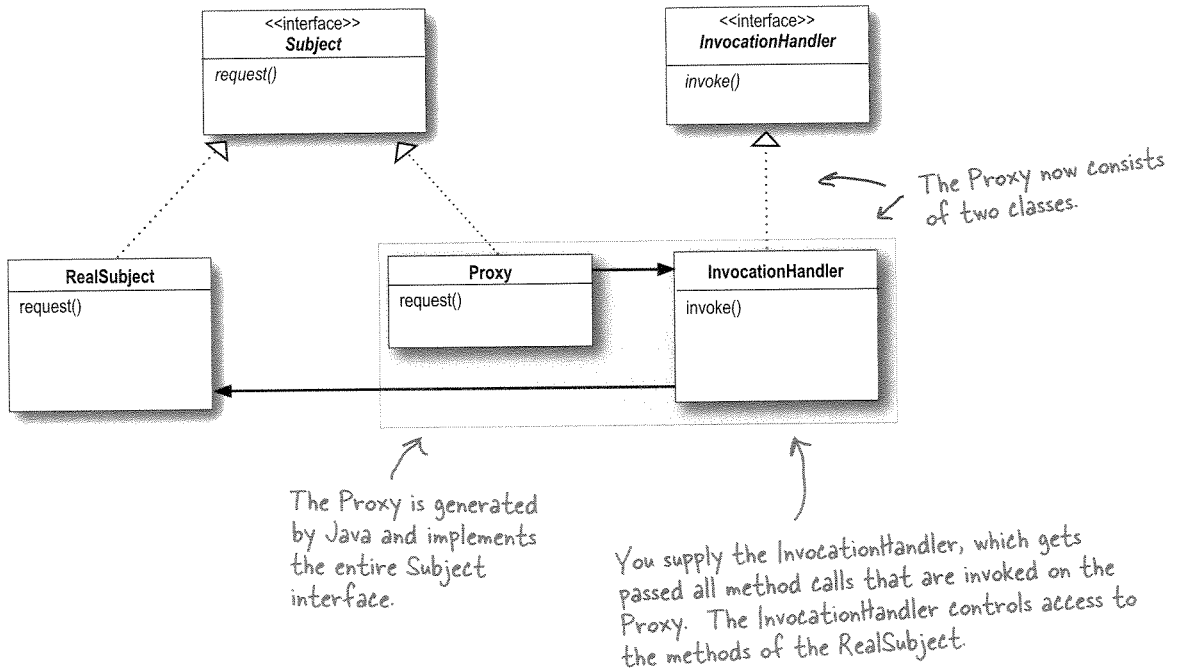


# Using the Java API's Proxy to create a protection proxy



Java's got its own proxy support right in the `java.lang.reflect` package. With this package, Java lets you create a proxy class *on the fly* that implements one or more interfaces and forwards method invocations to a class that you specify. Because the actual proxy class is created at runtime, we refer to this Java technology as a *dynamic proxy*.

We're going to use Java's dynamic proxy to create our next proxy implementation (a protection proxy), but before we do that, let's quickly look at a class diagram that shows how dynamic proxies are put together. Like most things in the real world, it differs slightly from the classic definition of the pattern:



Because Java creates the Proxy class *for you*, you need a way to tell the Proxy class what to do. You can't put that code into the Proxy class like we did before, because you're not implementing one directly. So, if you can't put this code in the Proxy class, where do you put it? In an `InvocationHandler`. The job of the `InvocationHandler` is to respond to any method calls on the proxy. Think of the `InvocationHandler` as the object the Proxy asks to do all the real work after it's received the method calls.

Okay, let's step through how to use the dynamic proxy...