Proxy objects

1. Proxy objects

A proxy class is a class that implements a list of interfaces specified at compile time. The proxy object typically holds a reference to an internal object that implements the same interfaces (or parts of them). The proxy object is implemented by delegating method calls to the internal object. This is the same principle as implemented by the class `java.lang.reflect.Proxy`, which created proxy objects dynamically at runtime using Java reflection.

Compared to the standard Proxy class, the generator has the obvious disadvantage, that you have to specify the implemented interfaces at runtime. On the other hand, it allows both to select the proxy objects super class and the derivation of subclasses. In fact the derivation of a subclass is much more obvious, simple and faster than the use of an InvocationHandler.

The proxy generator is implemented by the class ProxyGenerator. Use of the ProxyGenerator is demonstrated in the Ant target "generate.proxy".

2. Multiple Inheritance

Multiple inheritance is a design pattern which is not so easy to implement in Java - unless you use the ProxyGenerator. This is demonstrated by the JUnit test MultipleInheritanceTest, which creates a subclass of `java.util.Observable`, that also inherits from `java.util.ArrayList`. The example implements a list, which notifies its observers whenever the `add()` method is invoked.