Proxy

An interface for accessing a particular resource

By Alexander Tilkin

Motivation

- You are calling *foo.bar()*
- This assumes that *foo* is in the same process as *bar()*
- What if, later on, you want to pull all *Foo*-related operations into a separate process
  - Can you avoid changing your code?
- Proxy to the rescue!
  - Same interface, entirely different behavior
- This is called a *communication proxy*
  - Other types: logging, virtual, guarding, ...

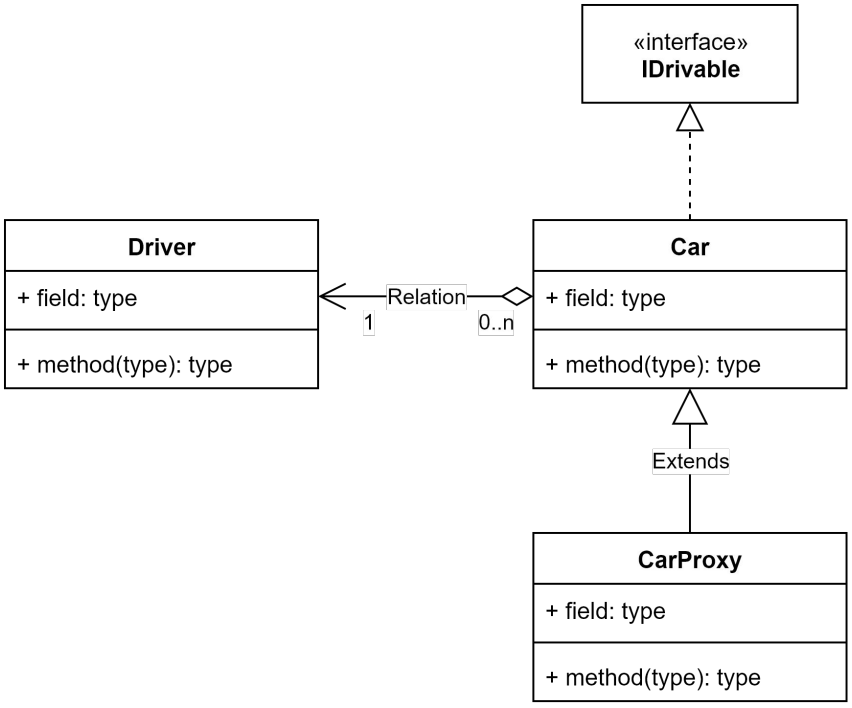
Proxy

A class that functions as an interface to a particular resource. That resource may be remote, expensive to construct, or may require logging or some other added functionality.

units.unit_12_proxy.ProtectionProxyExample.java

Protection Proxy

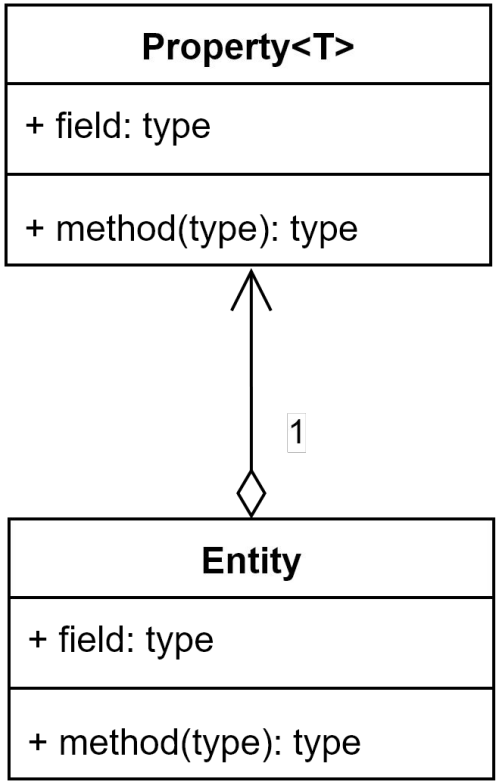
- When we have a class that has a functionality we want to protect we can create a proxy class that extends the original class
- The proxy protection class basically applies data validation prior to invoking the methods of the protected class and decides whether the methods can be invoked



units.unit_12_proxy.LoggingProxyExample.java

Property Proxy

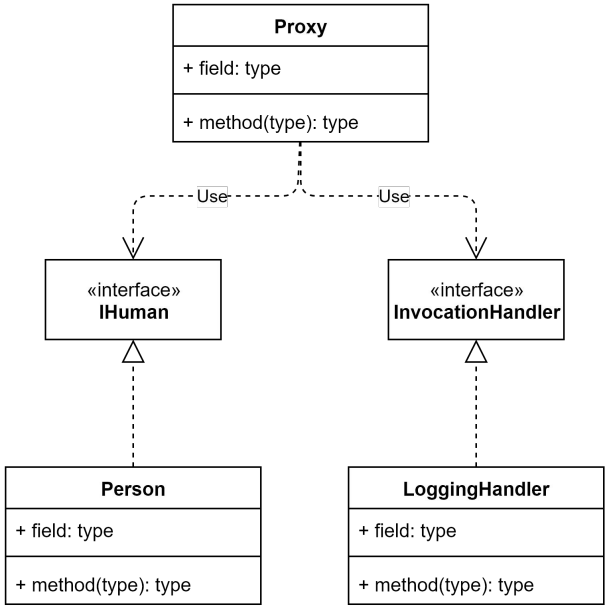
- The idea of property proxy is storing a generic property in the Entity class. The type of property can changed dynamically using the entity proxy



units.unit_12_proxy.DynamicLoggingProxyExample.java

Dynamic Logging

- In this example we demonstrate how one can write an application with an abstract logging proxy
- Using the abstract logging proxy, we can log any class we want with a full agnosticism
- InnvocationHandler - is the interface implemented by the invocation handler of a proxy instance. Each proxy instance has an associated invocation handler. When a method is invoked on a proxy instance, the method invocation is encoded and dispatched to the invoke method of its invocation handler.
- Proxy - provides static methods for creating dynamic proxy classes and instances, and it is also the superclass of all dynamic proxy classes created by those methods



Side by Side: Proxy and Decorator

Proxy	**Decorator**
Provides an identical interface	Provides an enhanced interface
Doesn’t have to aggregate	Typically aggregates (or has reference to) what it is decorating
Might even be working with materialized object	

units.unit_12_proxy.ProxyCodingExample.java

Proxy Coding Exercise

You are given the **Person** class and asked to write a **ResponsiblePerson** proxy that does the following:

- Allows person to drink unless they are younger than 18 (in that case return “too young”)
- Allows person to drive unless they are younger than 16 (in that case return “too young”)
- In case of driving and drinking, return “do not do that!!!! :/”

Summary

- A proxy has the same interface as the underlying object
- To create a proxy, simply replicate the existing interface of an object
- Add relevant functionality to the redefined member functions
- Different proxies (communication, logging, caching etc.) have completely different behaviours