Design of Sample Implementation-Object Code Example

The following design example reflects the associated code example. This is a good example of separation by interface, delegation, the use of implementation objects, and the Single Responsibility Principle (SRP).

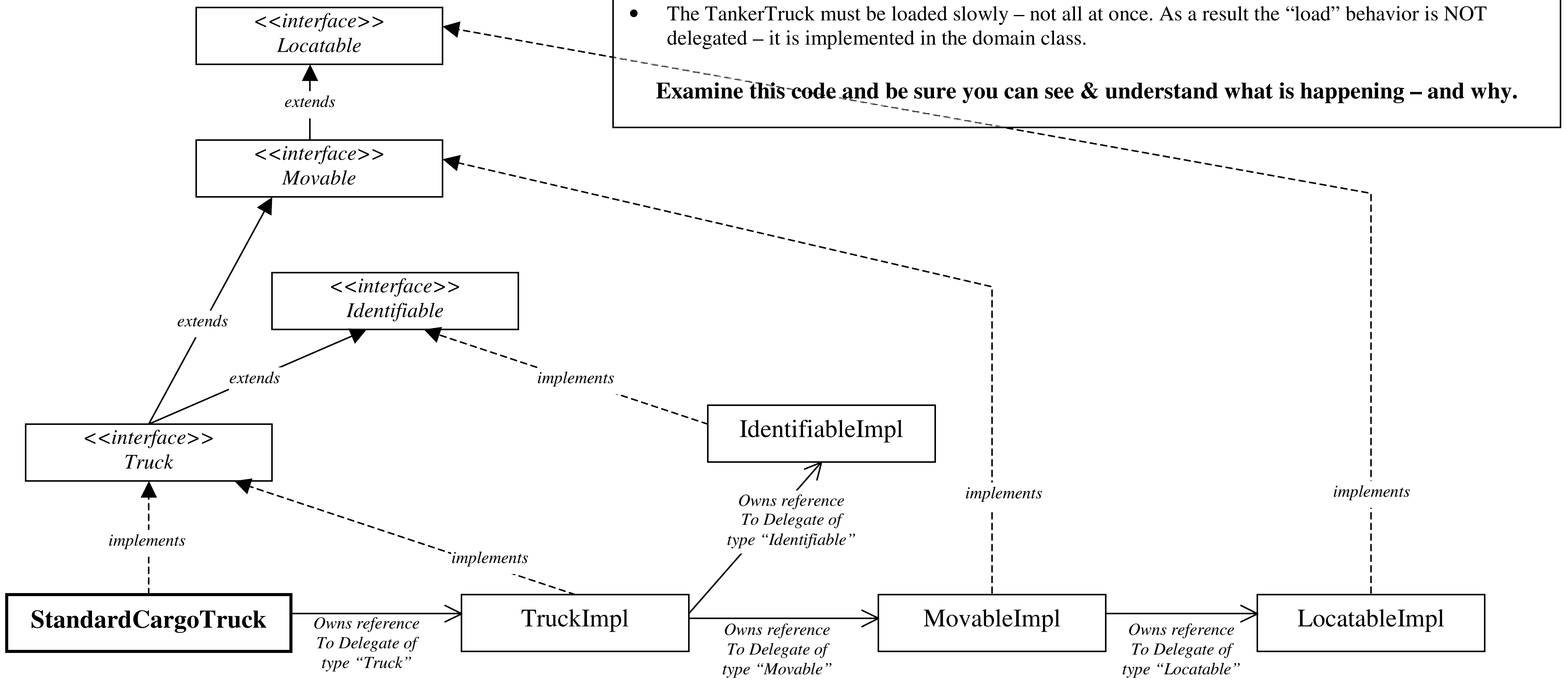
Single Responsibility Principle

There Should Never Be More Than One Reason For A Class To Change. If a class has more then one responsibility, then the responsibilities become coupled. Changes to one responsibility may impair or inhibit the class' ability to meet the others. This kind of coupling leads to fragile designs that break in unexpected ways when changed.
In the context of the Single Responsibility Principle (SRP) we define a responsibility to be "a reason for change." If you can think of more than one motive for changing a class, then that class has more than one responsibility. This is sometimes hard to see. We are accustomed to thinking of responsibility in groups
The SRP is one of the simplest of the principle, and one of the hardest to get right. Conjoining responsibilities is something that we do naturally. Finding and separating those responsibilities from one another is much of what software design is really about.

Domain Classes – There are 3 "domain" classes in this example – **StandardCargoTruck**, **ContainerTruck** & **TankerTruck**.

- The StandardCargoTruck does everything in a "standard" way, including "load" and "unload" behavior – no special implementations – so everything is delegated.
- The ContainerTruck can ONLY be fully loaded or unloaded – no partial loading or unloading. As a result the "load" and "unload" behavior is NOT delegated – it is implemented in the domain class.
- The TankerTruck must be loaded slowly – not all at once. As a result the "load" behavior is NOT delegated – it is implemented in the domain class.

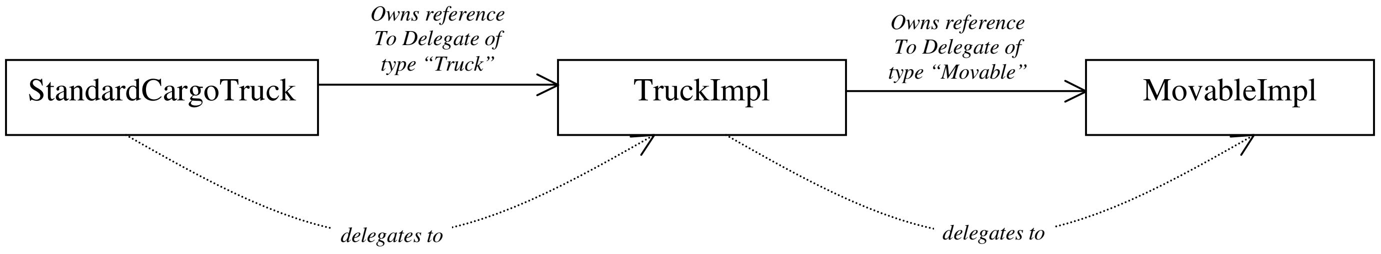
Examine this code and be sure you can see & understand what is happening – and why.



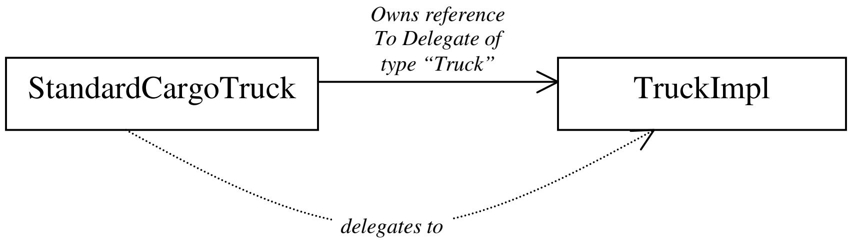
<<interface>> Locatable
extends
<<interface>> Movable
extends
<<interface>> Identifiable
extends
implements
<<interface>> Truck
IdentifiableImpl
implements
Owns reference To Delegate of type "Identifiable"
implements
implements
implements
StandardCargoTruck
TruckImpl
MovableImpl
LocatableImpl
Owns reference To Delegate of type "Truck"
Owns reference To Delegate of type "Movable"
Owns reference To Delegate of type "Locatable"

How the Delegation/Implementation Objects Code Example Works:

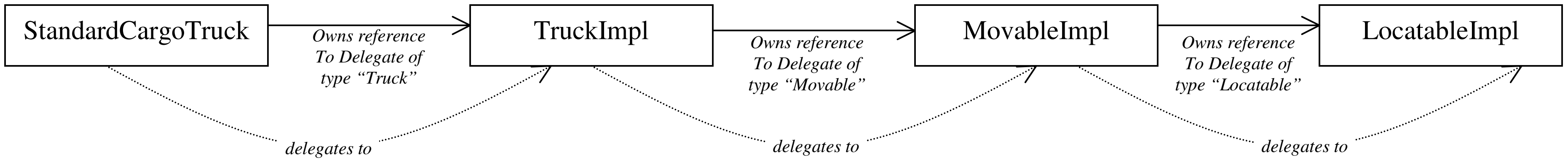
myStdCargoTruck.setDestination(aPoint3D) “setDestination(…)” is *Movable* behavior. *Movable* behavior is a subset of *Truck* behavior. So, the call is delegated from StandardCargoTruck to it’s *Truck* delegate (TruckImpl) and from there it is delegated to the *Movable* delegate (MovableImpl) where it is actually implemented.


StandardCargoTruck
Owns reference To Delegate of type “Truck”
TruckImpl
Owns reference To Delegate of type “Movable”
MovableImpl
delegates to
delegates to

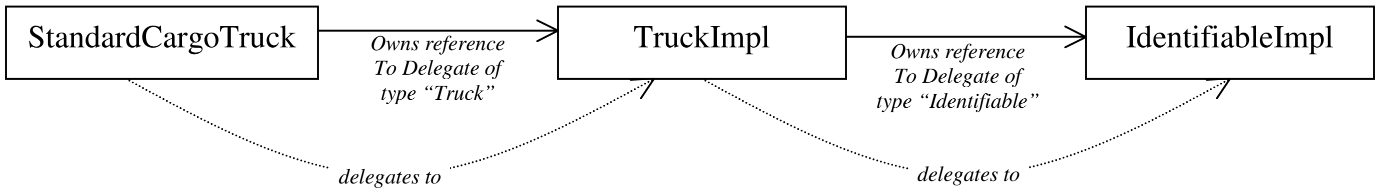
myStdCargoTruck.load(2250.0) “load(…)” is Truck behavior, so the call is delegated from StandardCargoTruck to it’s *Truck* delegate (TruckImpl) where it is actually implemented.



myStdCargoTruck.getLocation() “getLocation ()” is *Locatable* behavior. *Locatable* behavior is a subset of *Movable* behavior, which is a subset of *Truck* behavior. So, the call is delegated from StandardCargoTruck to it’s *Truck* delegate (TruckImpl) and from there it is delegated to the *Movable* delegate (MovableImpl) and from there it is delegated to the *Locatable* delegate (LocatableImpl)where it is actually implemented.


StandardCargoTruck
Owns reference To Delegate of type “Truck”
TruckImpl
Owns reference To Delegate of type “Movable”
MovableImpl
Owns reference To Delegate of type “Locatable”
LocatableImpl
delegates to
delegates to
delegates to

myStdCargoTruck.getIdentifier() “getIdentifier()” is *Identifiable* behavior. *Identifiable* behavior is a subset of *Truck* behavior. So, the call is delegated from StandardCargoTruck to it’s *Truck* delegate (TruckImpl) and from there it is delegated to the *Identifiable* delegate (Identifiable Impl) where it is actually implemented.


StandardCargoTruck
Owns reference To Delegate of type “Truck”
TruckImpl
Owns reference To Delegate of type “Identifiable”
IdentifiableImpl
delegates to
delegates to