Charla de Tesis 1

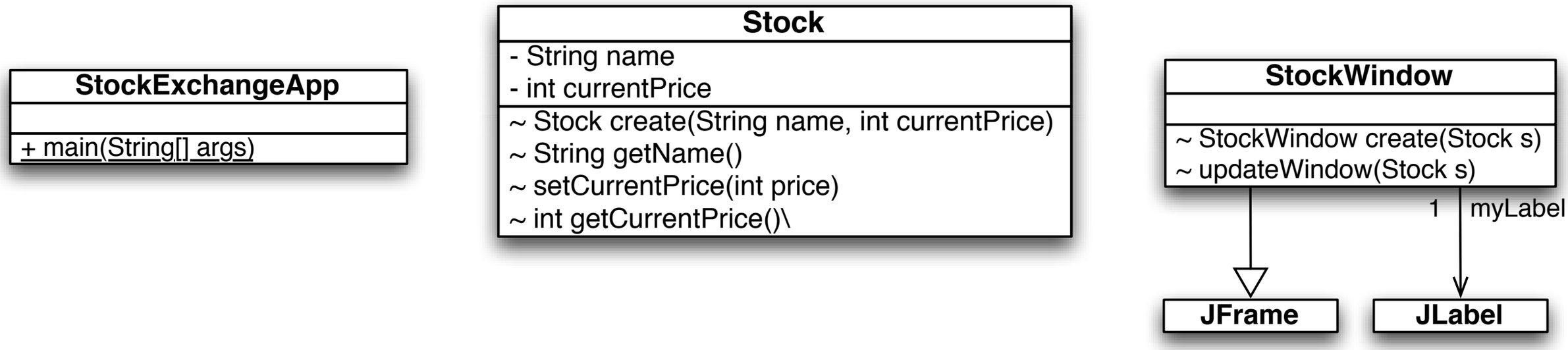
Magíster en TI

Marel Josué Oliva - moliva@dcc.uchile.cl
(material seleccionado de Joerg Kienzle)

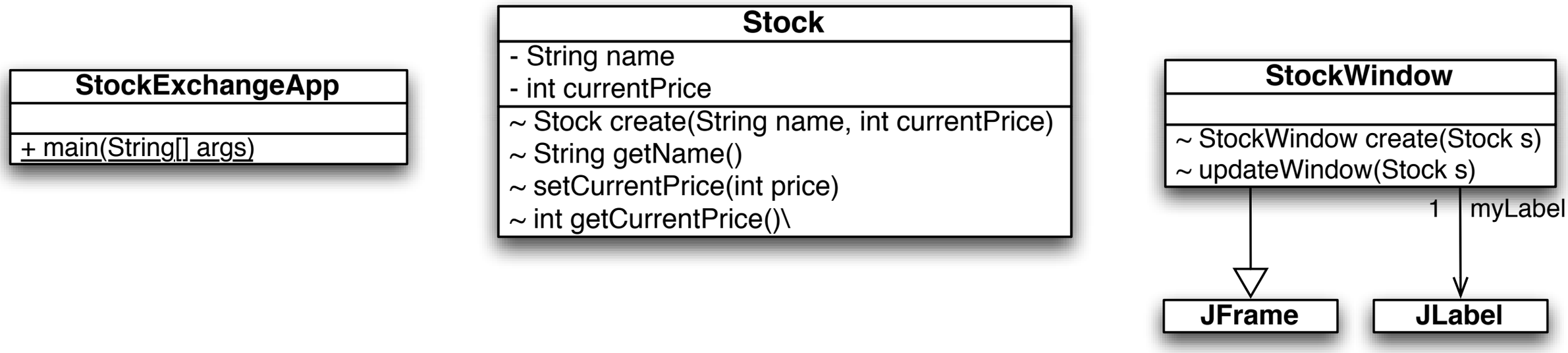
Stock Exchange Application (1)

- Un servidor con la información de los valores (nombre y precio)
- Varios GUI
- Acceso remoto

Stock Exchange Application (2)

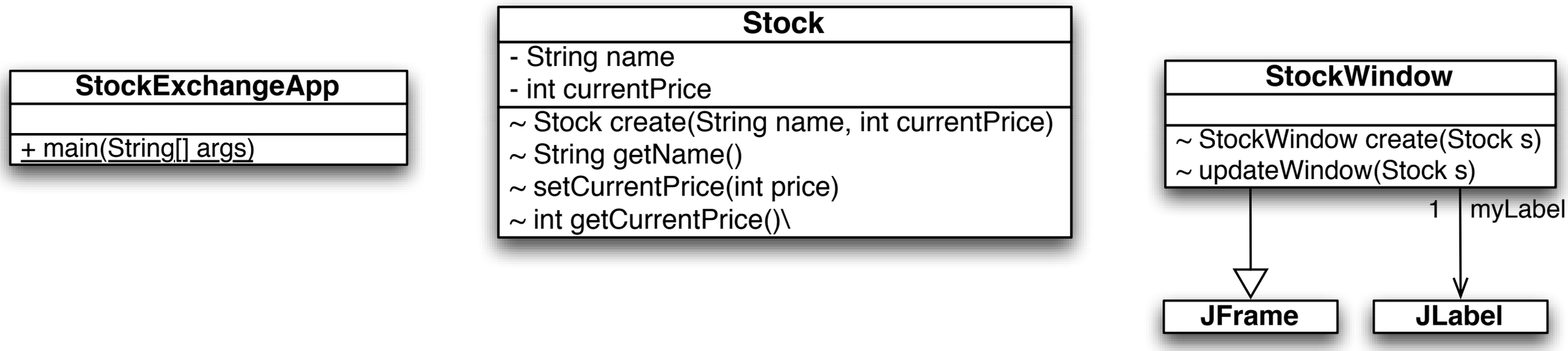


Stock Exchange Application (2)



¿Cómo desacoplar Stock de StockApplication?

Stock Exchange Application (2)



¿Cómo desacoplar Stock de StockApplication?

Patrón de diseño Observer

Consecuencias del patrón Observer

Consecuencias del patrón Observer

- Observer es cross-cutting

Consecuencias del patrón Observer

- Observer es cross-cutting
- Implementaciones de aspectos:
  - AspectJ, PHANtom, etc.

Consecuencias del patrón Observer

- Observer es cross-cutting
- Implementaciones de aspectos:
  - AspectJ, PHANtom, etc.
- ¿Cómo analizar/diseñar preocupaciones transversales?

Reusable Aspect Models (RAM)

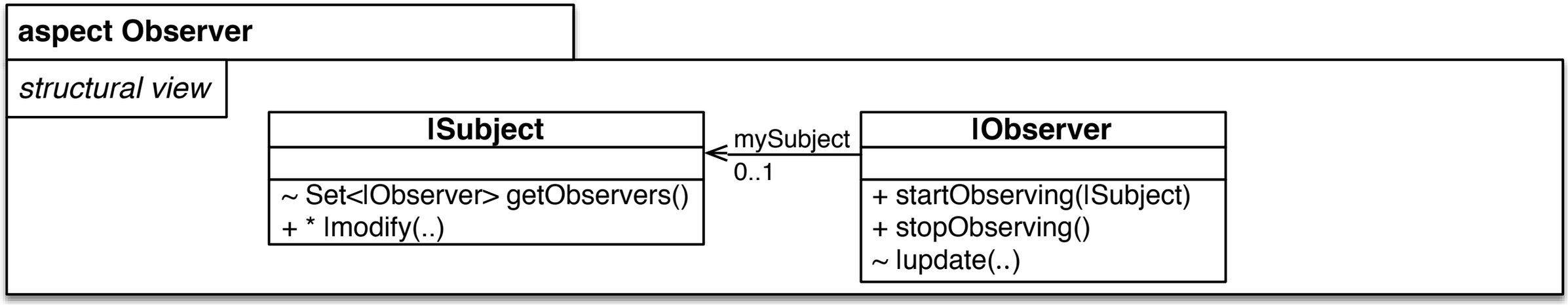
- Diseño de modelos de software reutilizables, gracias al modelamiento orientado a aspectos.

- Preocupación = aspecto

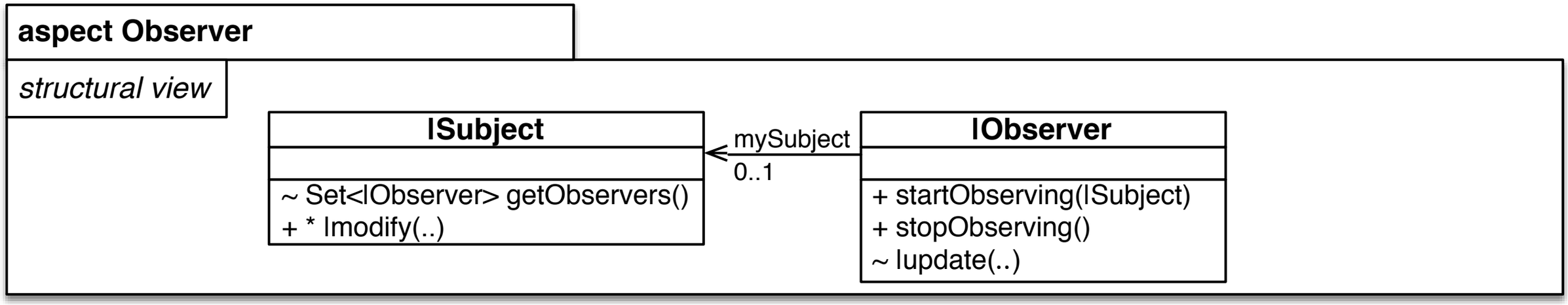
- "Aspect Model" => paquete UML => estructura y comportamiento de una preocupación

- Resuelve problemas de escalabilidad y consistencia del modelado orientado a aspectos

Vista estructural (Observer)

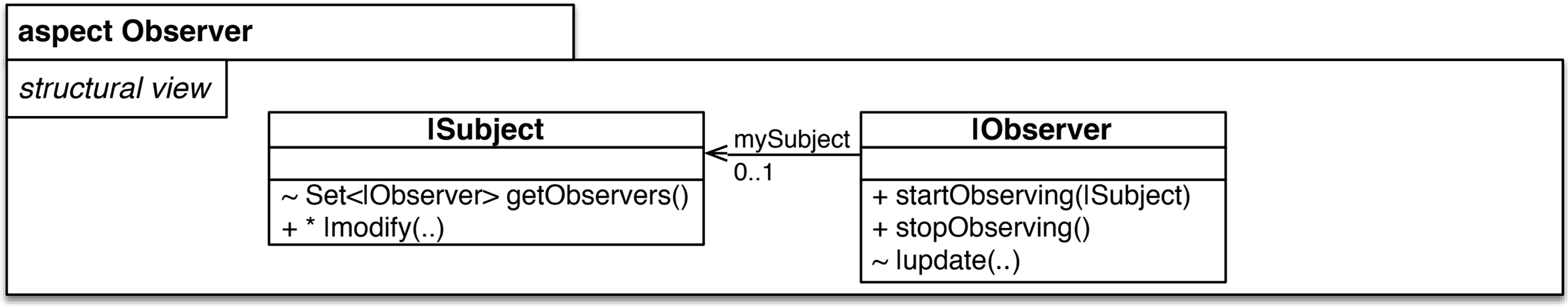


Vista estructural (Observer)



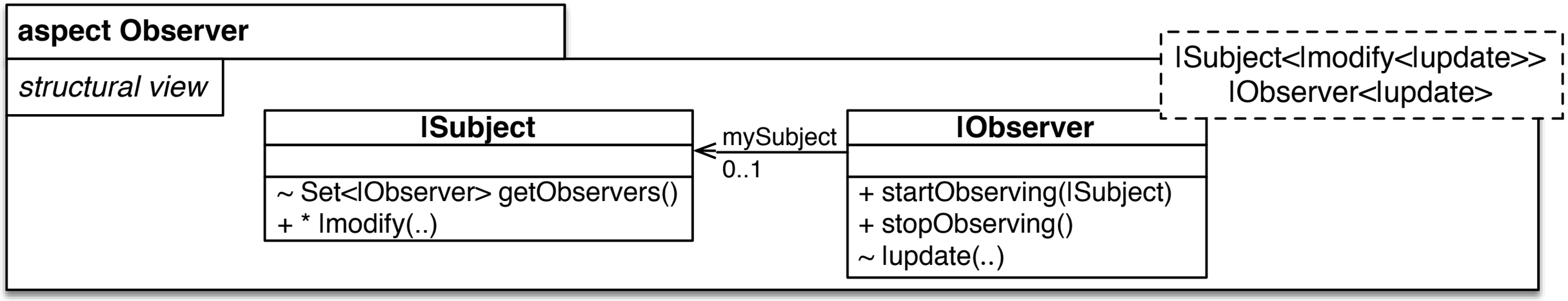
1. Conceptos relevantes al usuario (clases, operaciones)

Vista estructural (Observer)



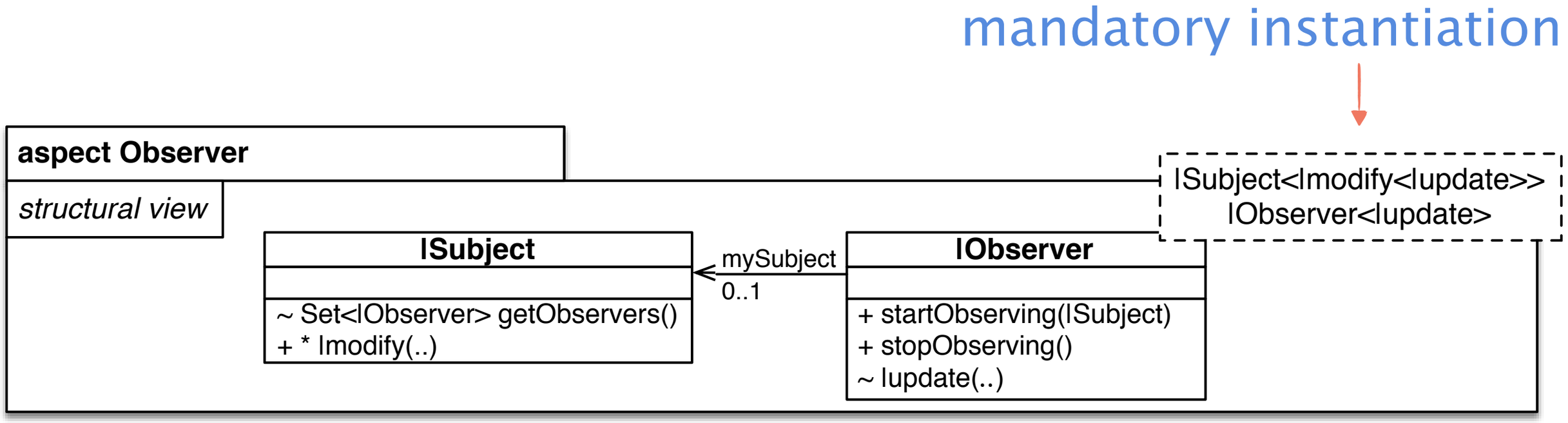
2. (**+**) interfaz aspecto
   (**~**) priv. aspecto
   (**-**) privado

Vista estructural (Observer)



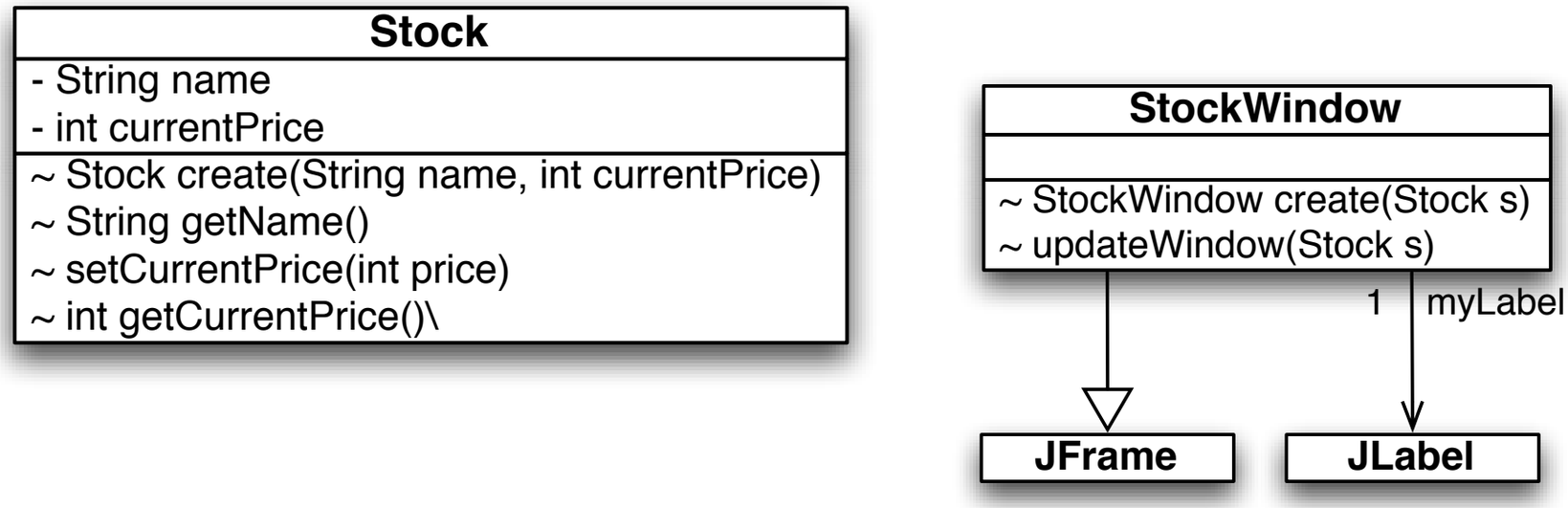
3. -Clases y operaciones incompletas

Vista estructural (Observer)

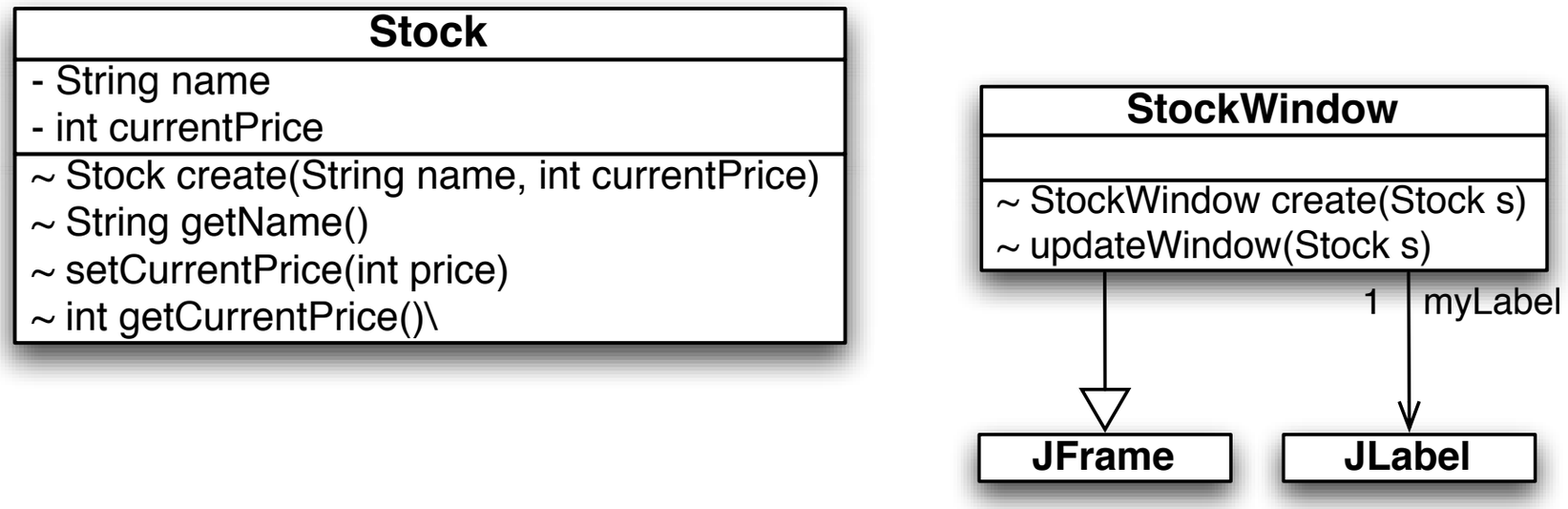


3. | -Clases y operaciones incompletas

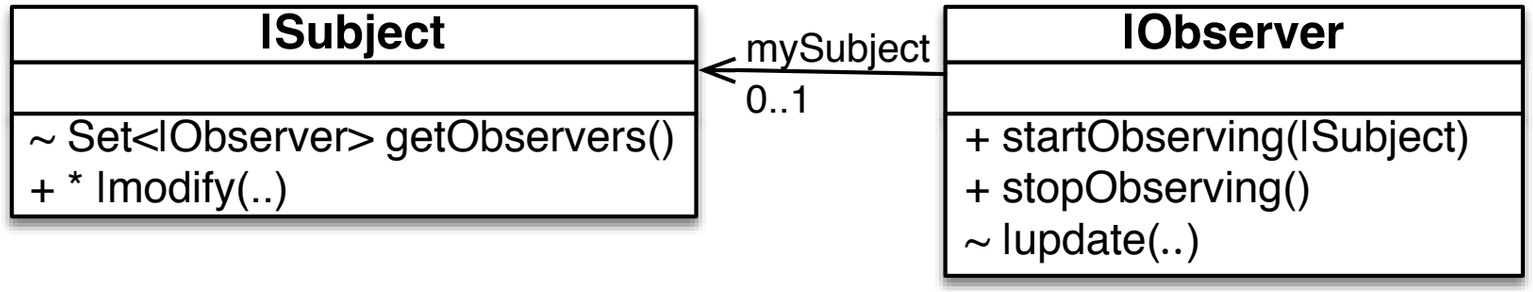
Reutilización por instanciación

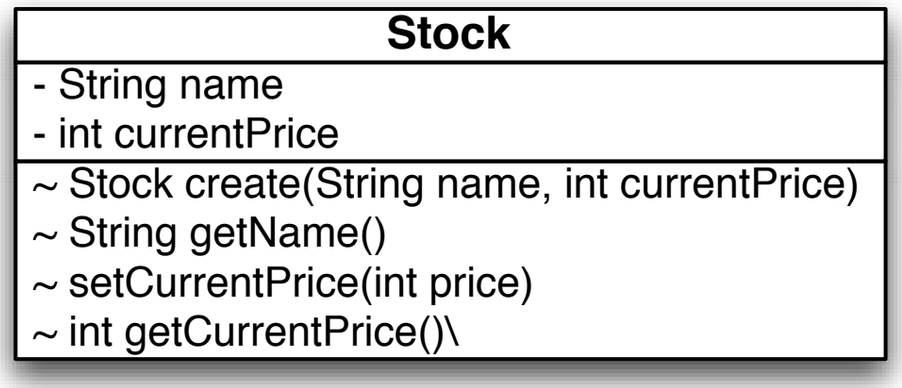


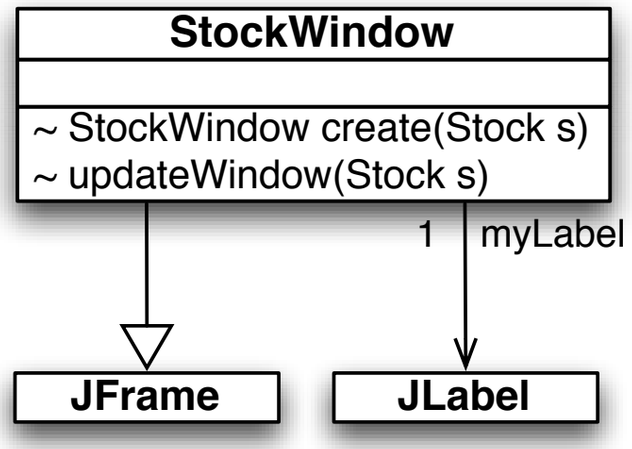
Reutilización por instanciación



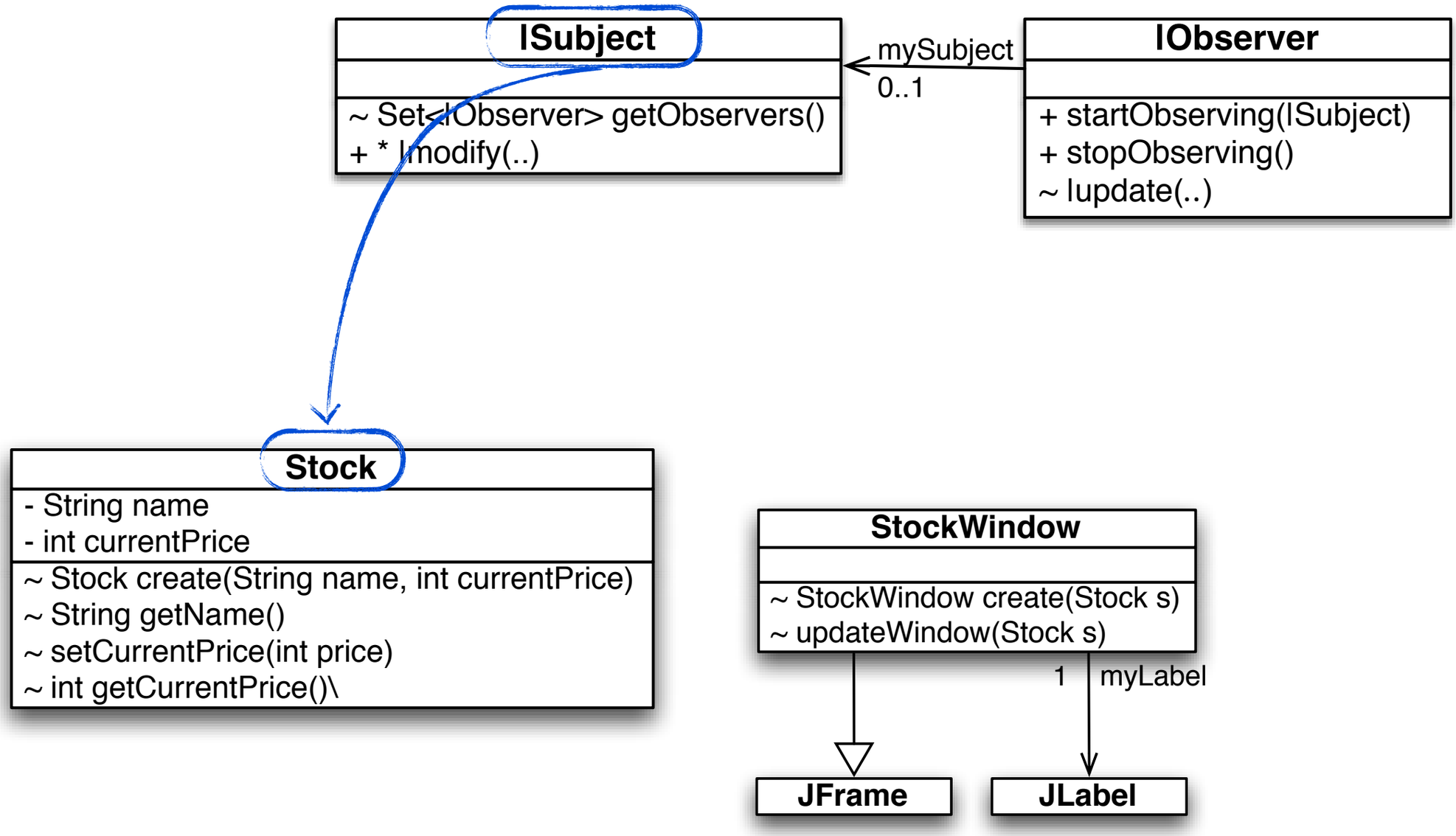
Reutilización por instanciación





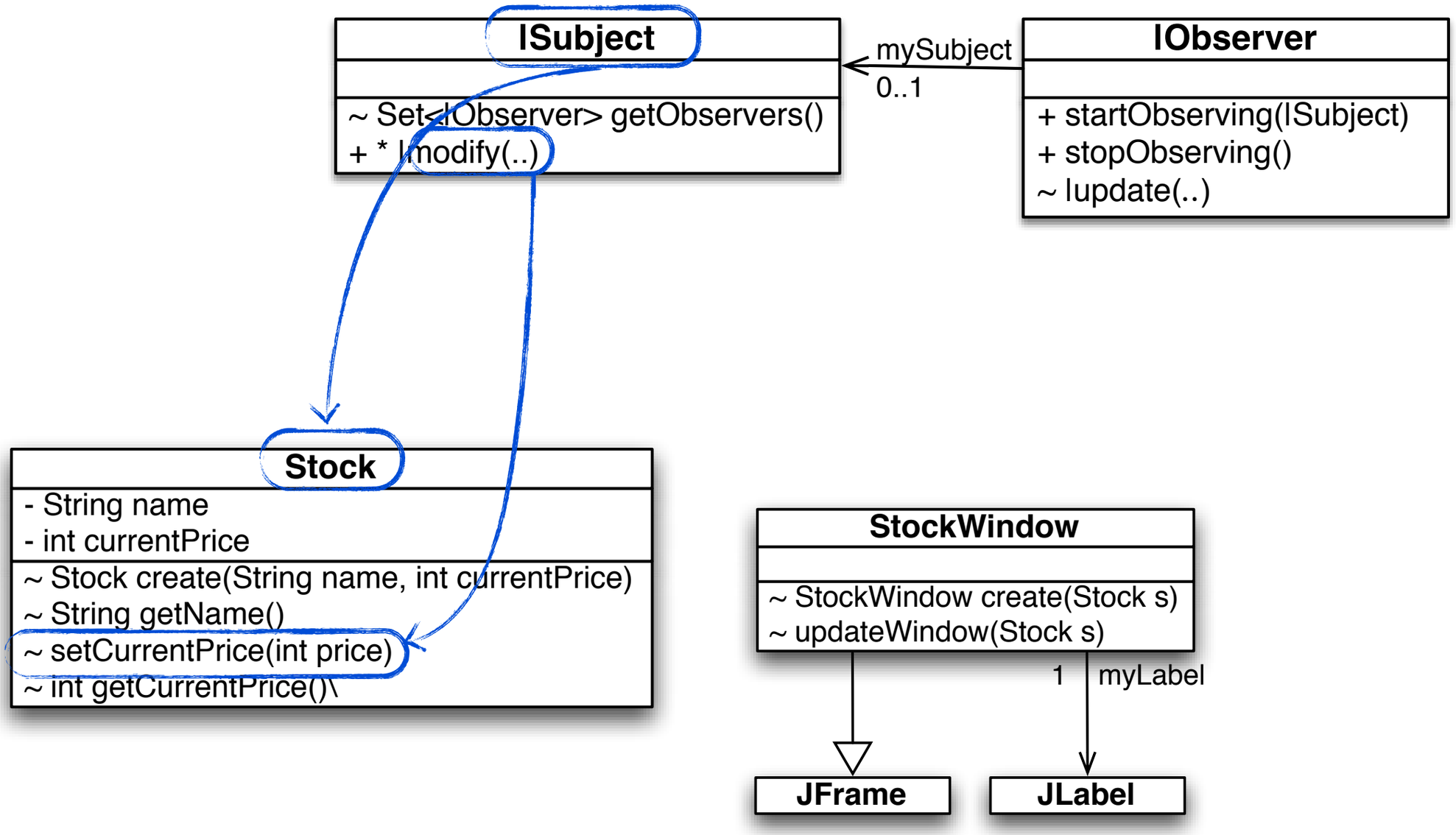


Reutilización por instanciación



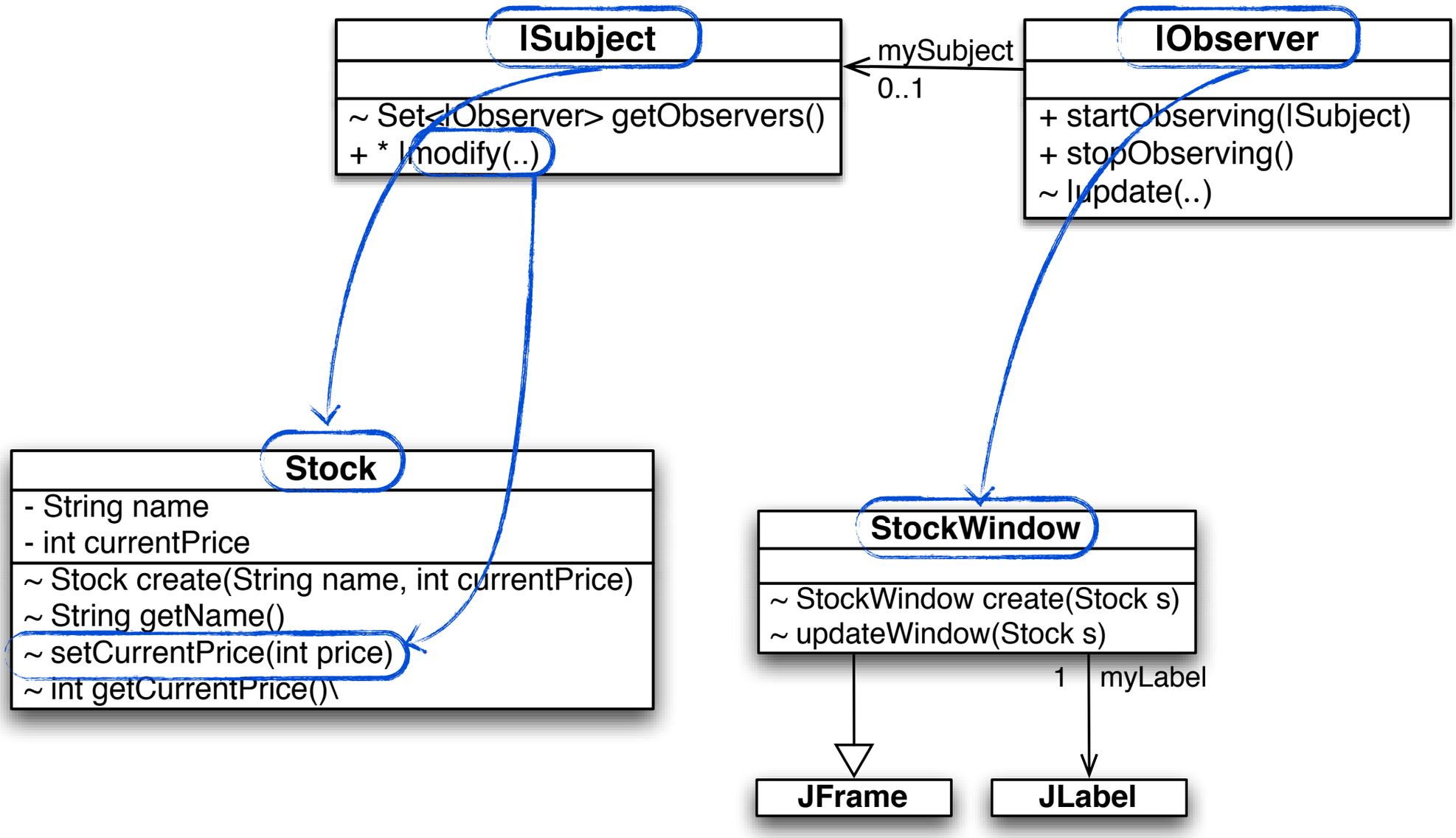
Stock -> |Subject

Reutilización por instanciación



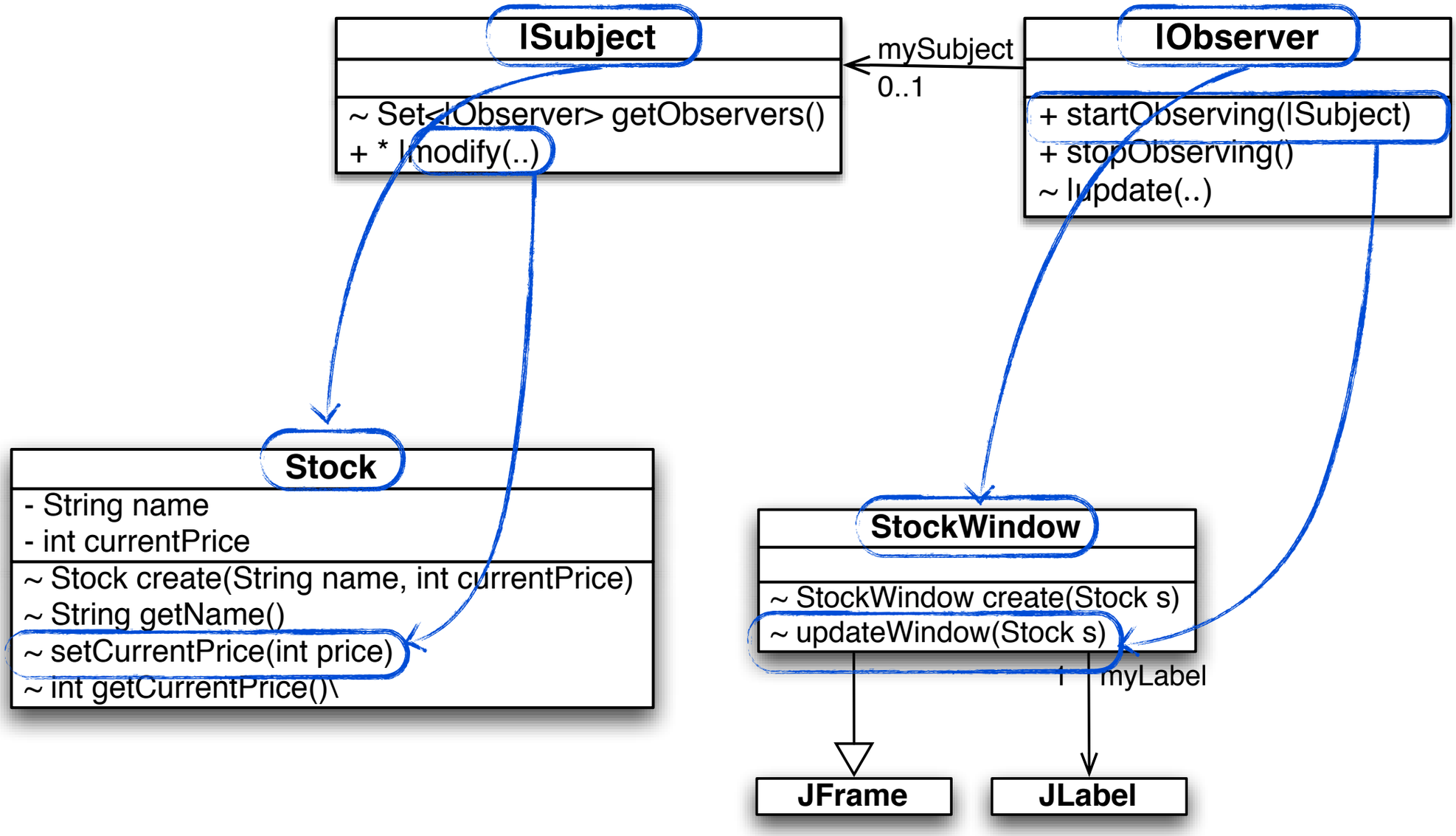
setCurrentPrice -> |modify

Reutilización por instanciación



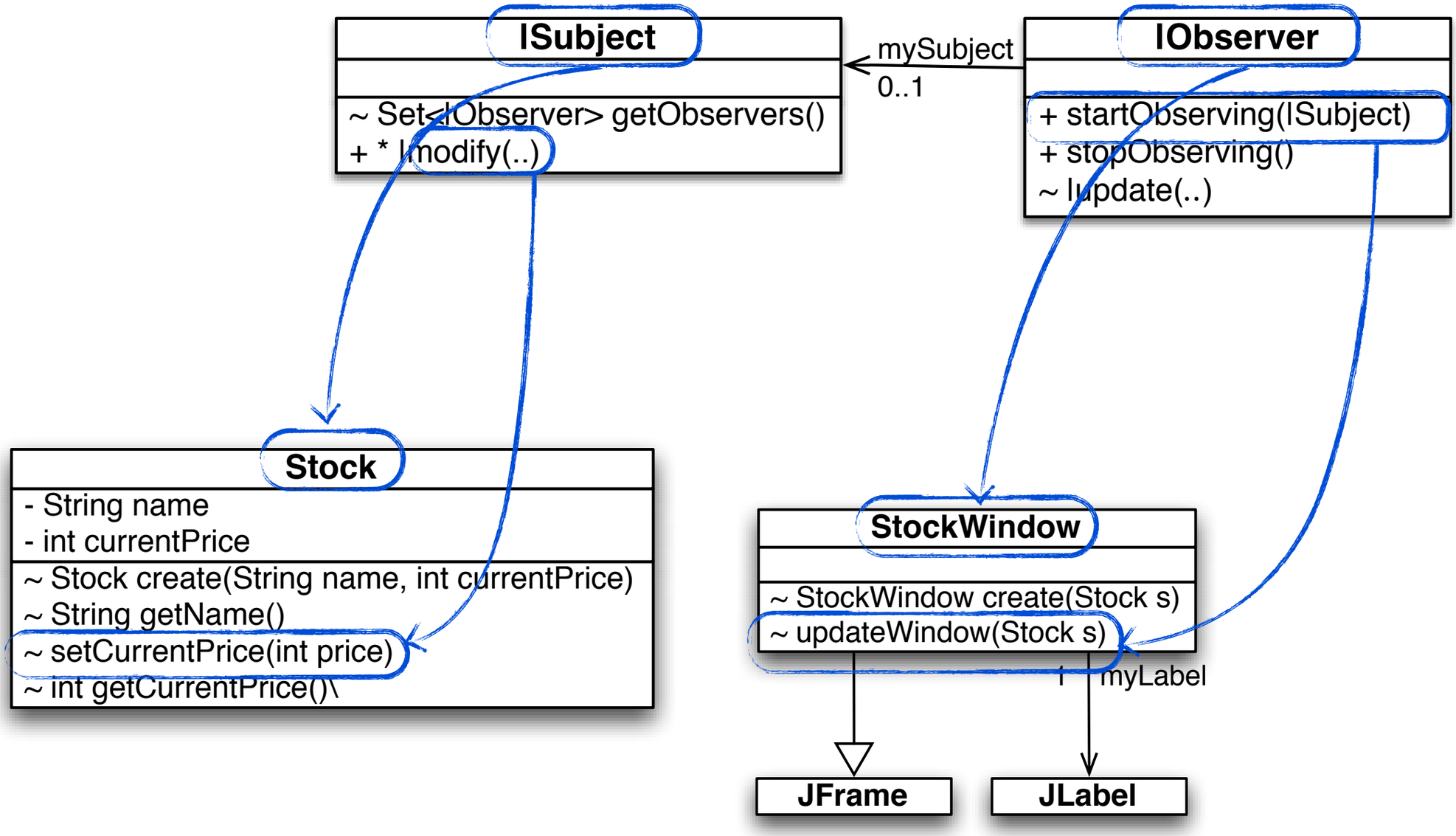
StockWindow -> |Obsever

Reutilización por instanciación



updateWindow -> |startObserving

Reutilización por instanciación

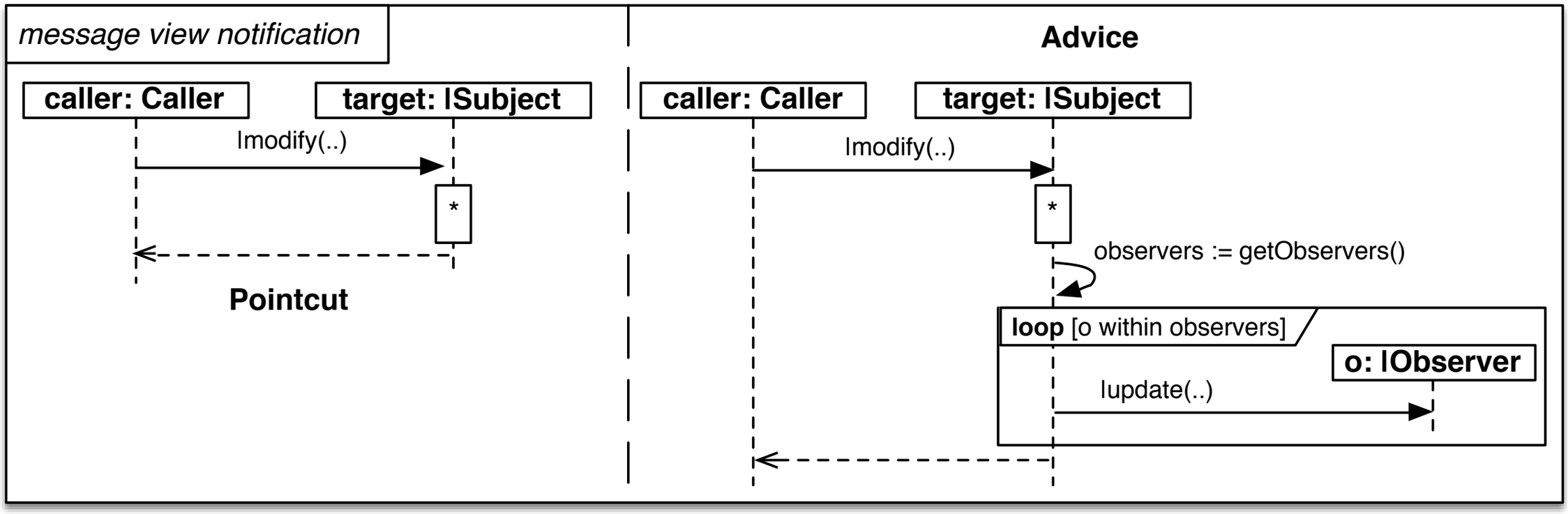


updateWindow -> |startObserving

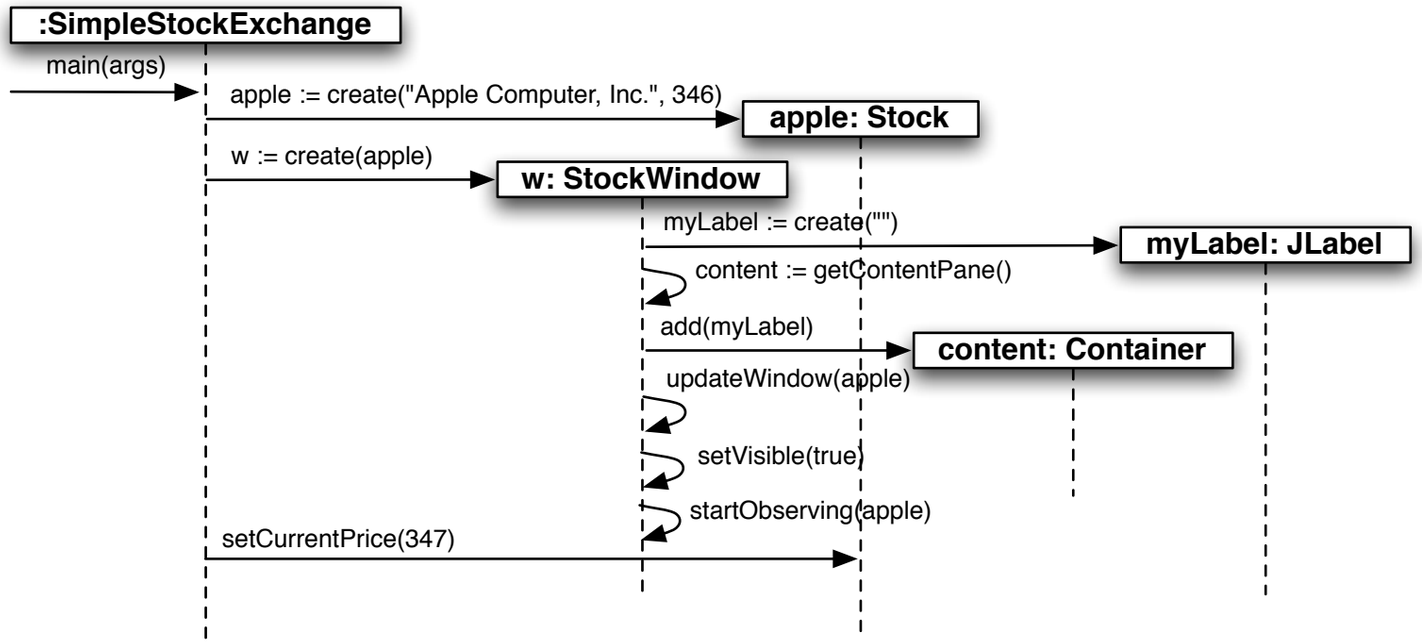
Weaving

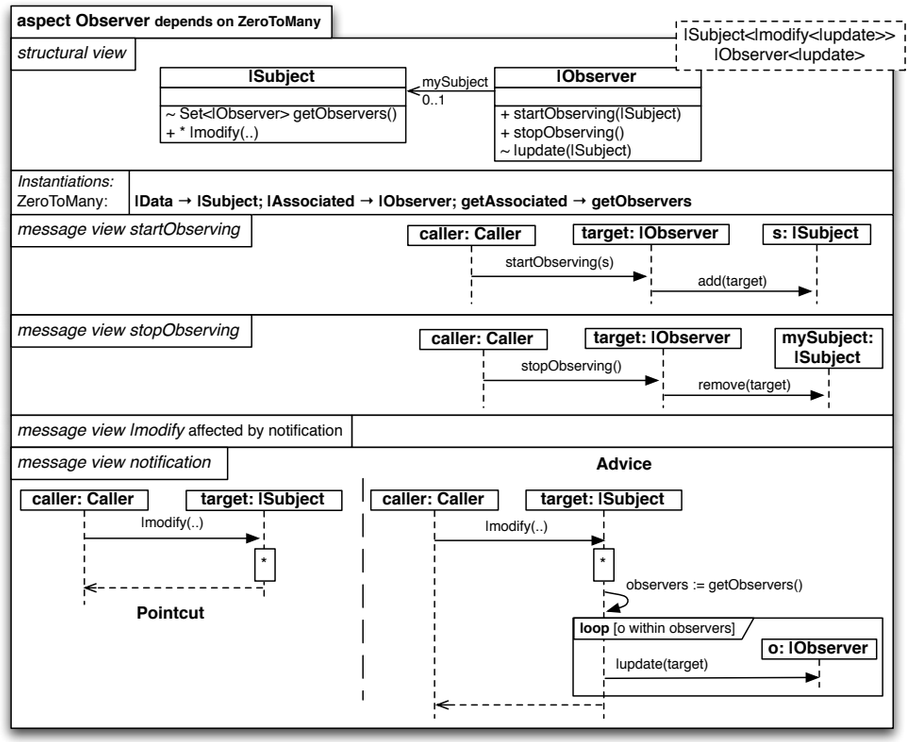

mySubject
0..1
Stock
- String name
- int currentPrice
~ Stock create(String name, int currentPrice)
~ String getName()
~ int getCurrentPrice()
~ setCurrentPrice(int price)
~ Set<StockWindow> getObservers()
~ add(StockWindow w)
~ remove(StockWindow w)
StockWindow
~ StockWindow create(Stock s)
~ updateWindow(Stock s)
~ startObserving(Stock)
~ stopObserving()

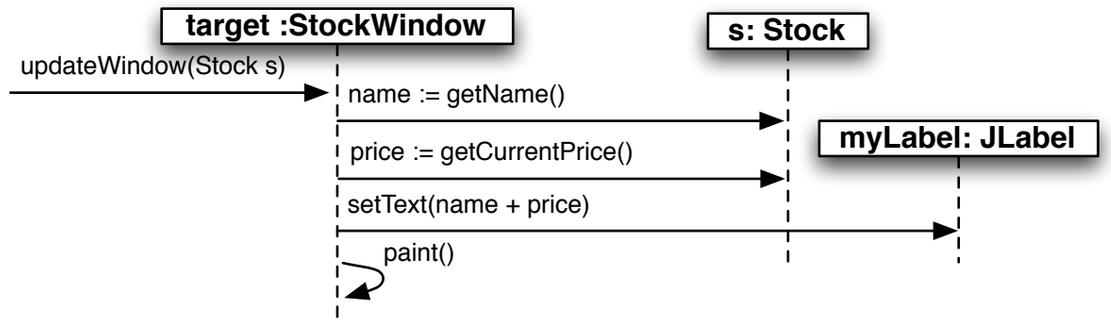
Vista Mensajes (Observer)



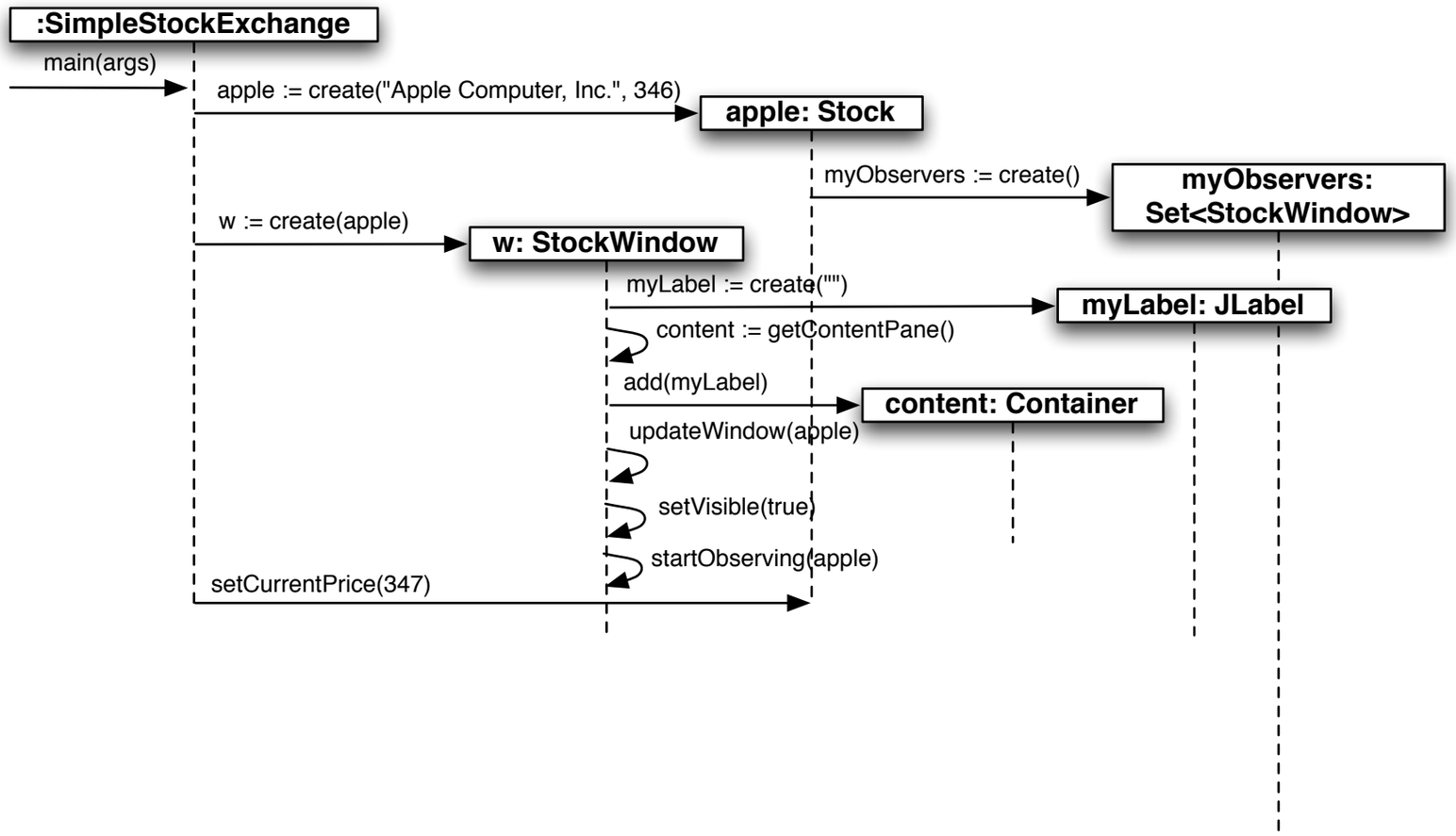
Weaving - diagramas de secuencia

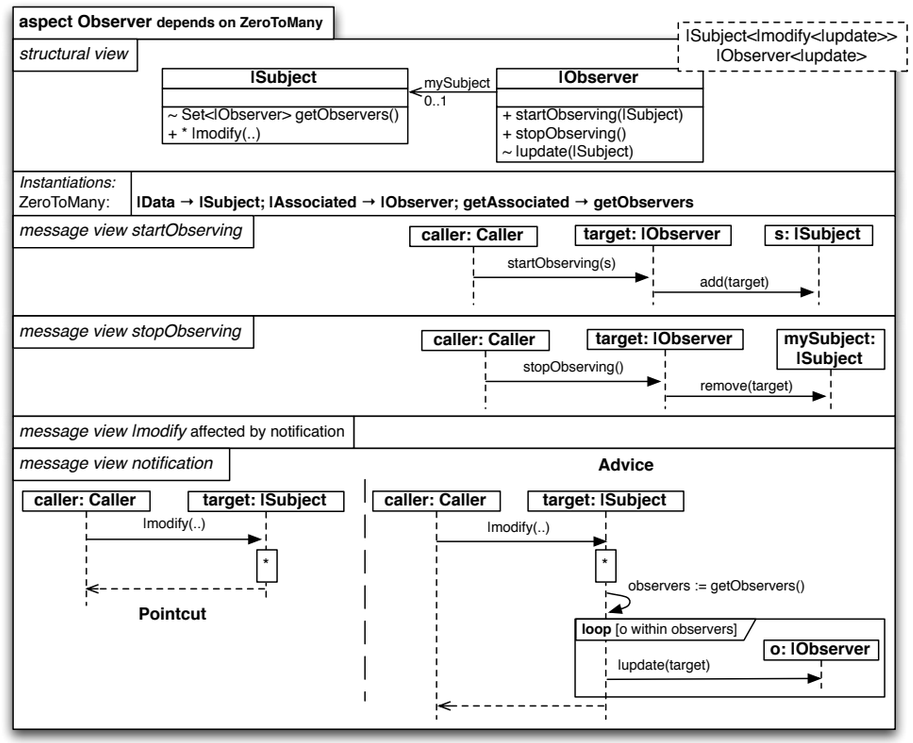


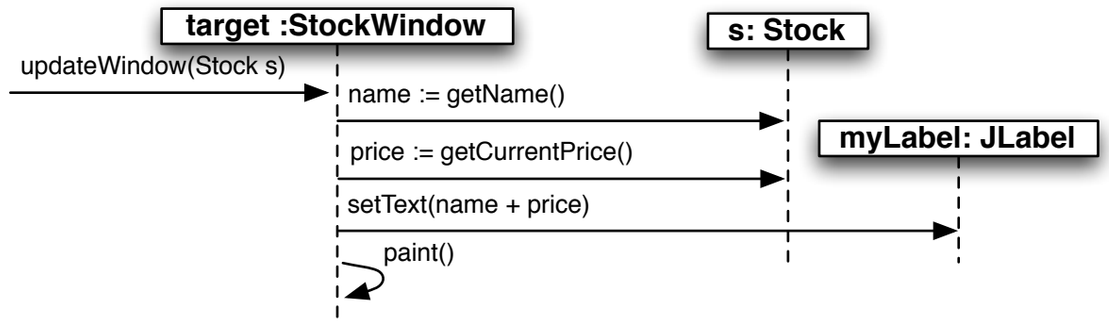




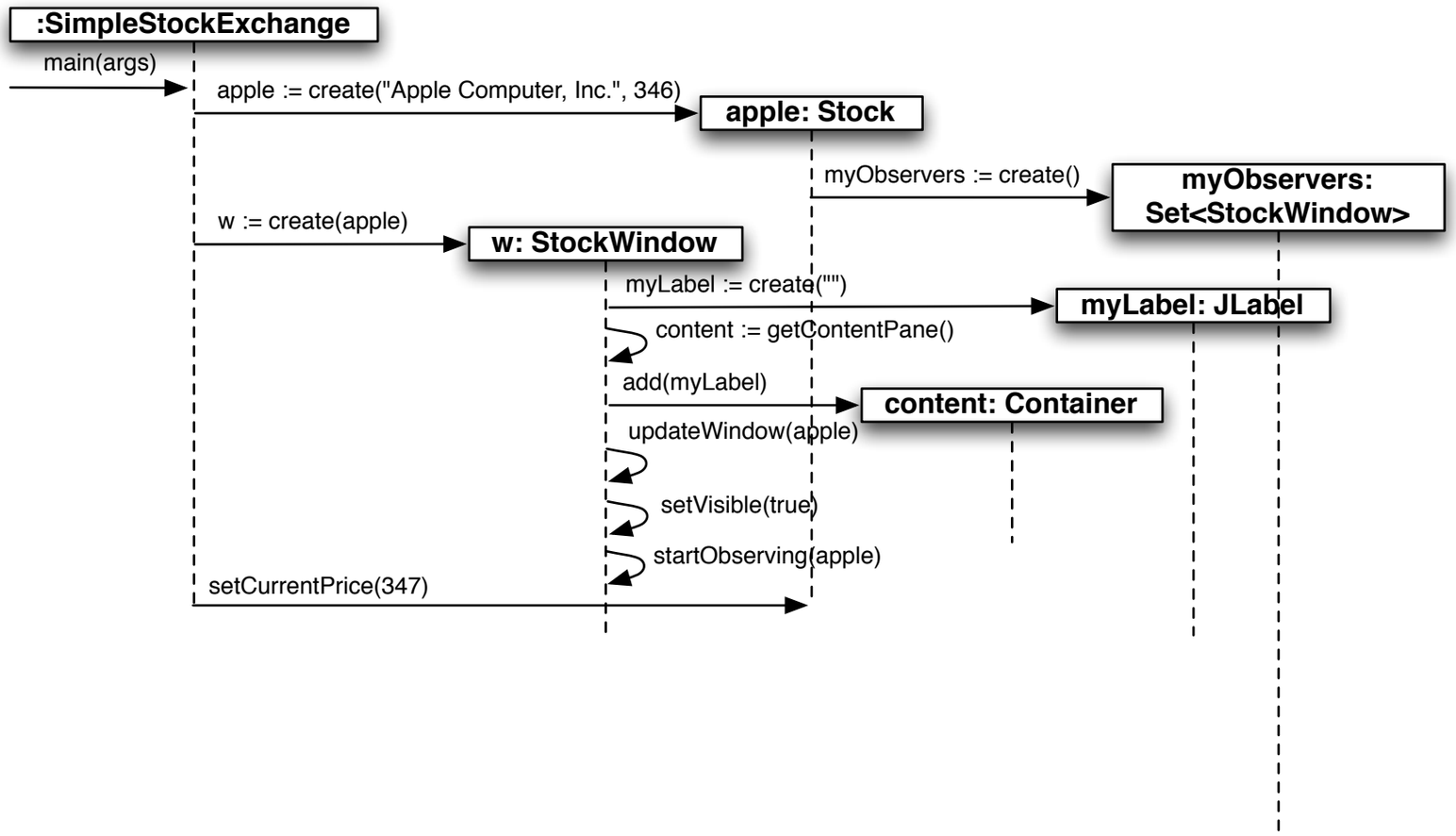
Weaving - diagramas de secuencia

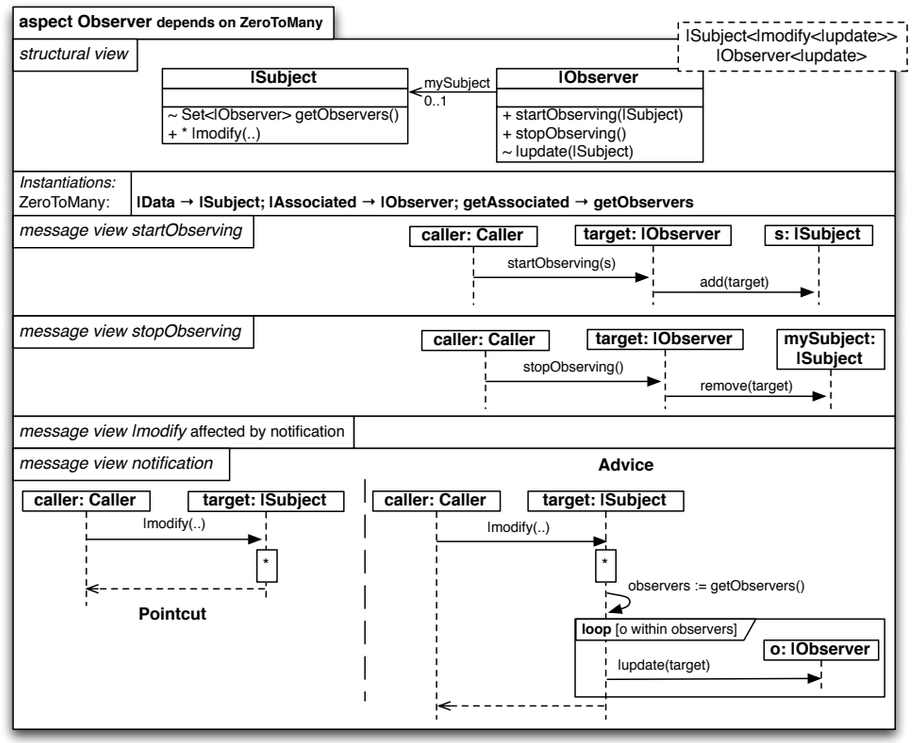


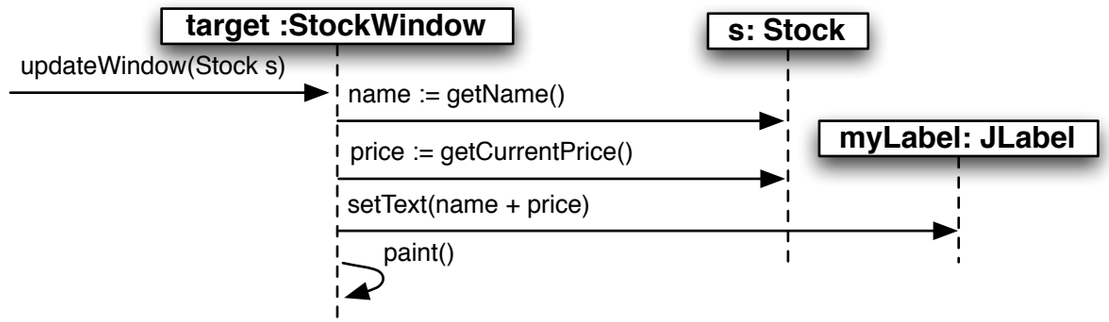




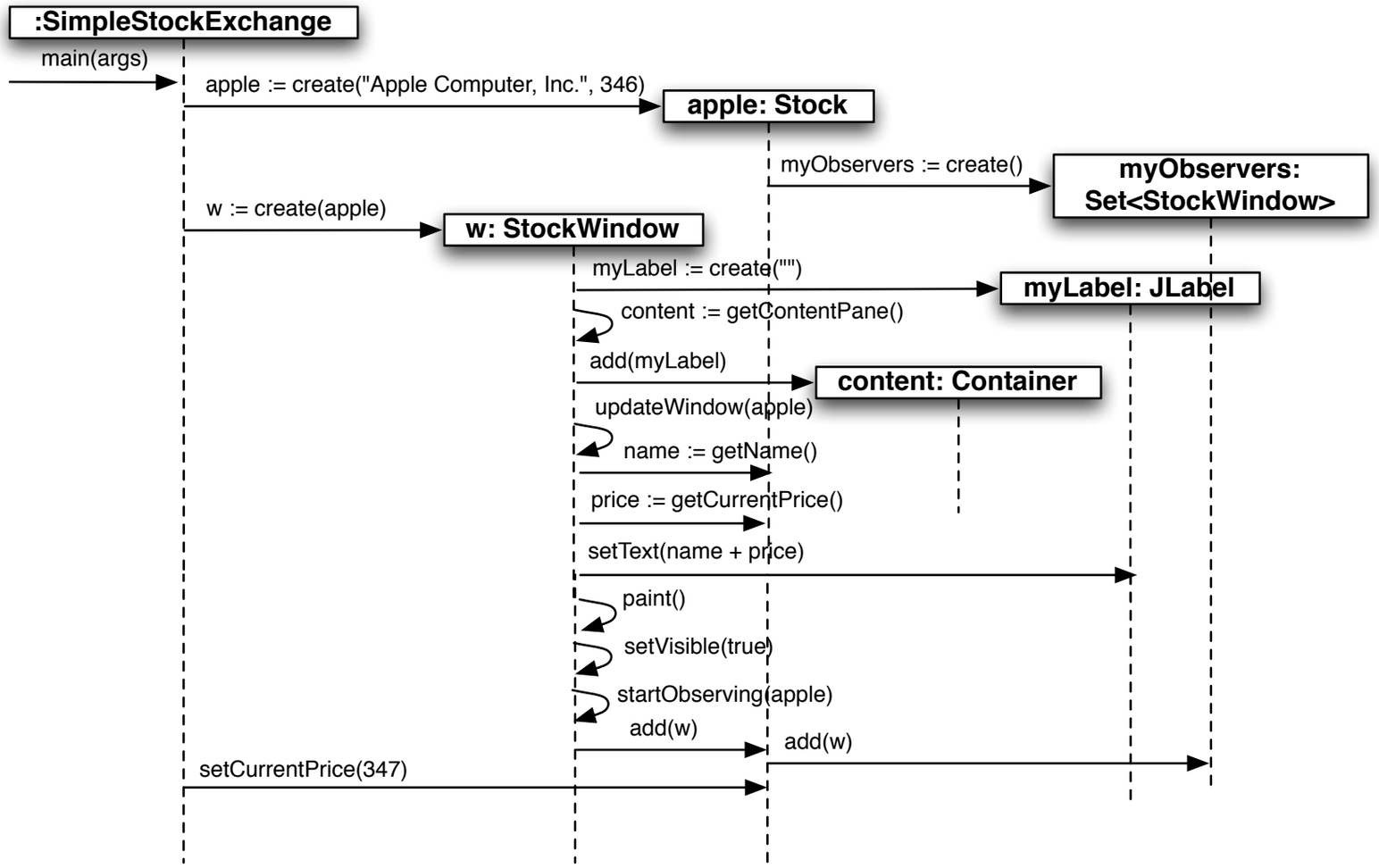
Weaving - diagramas de secuencia

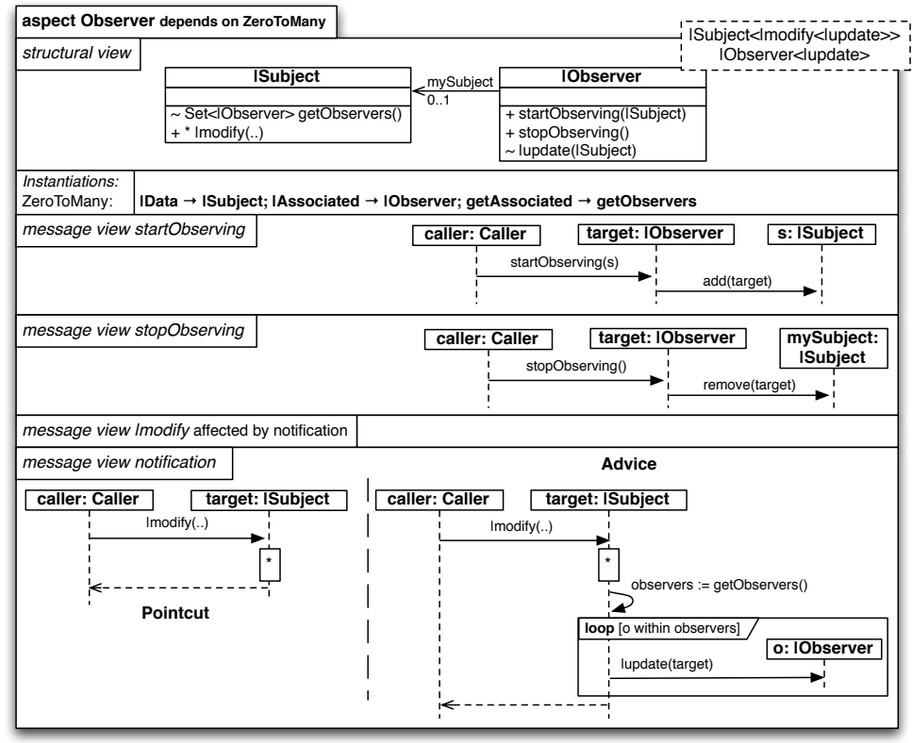


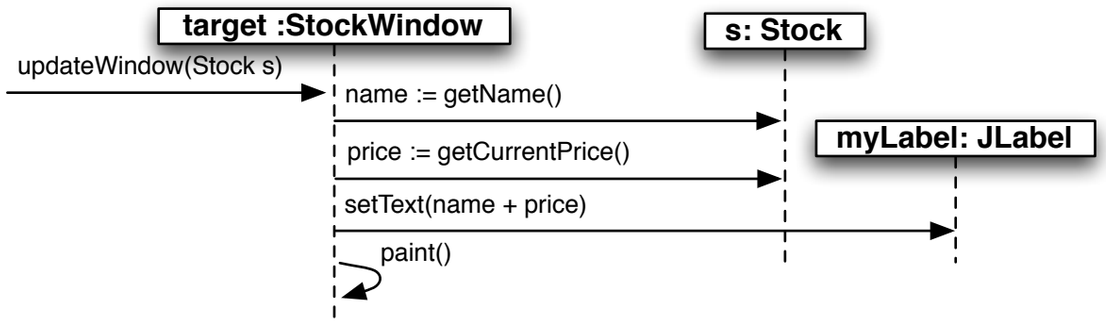




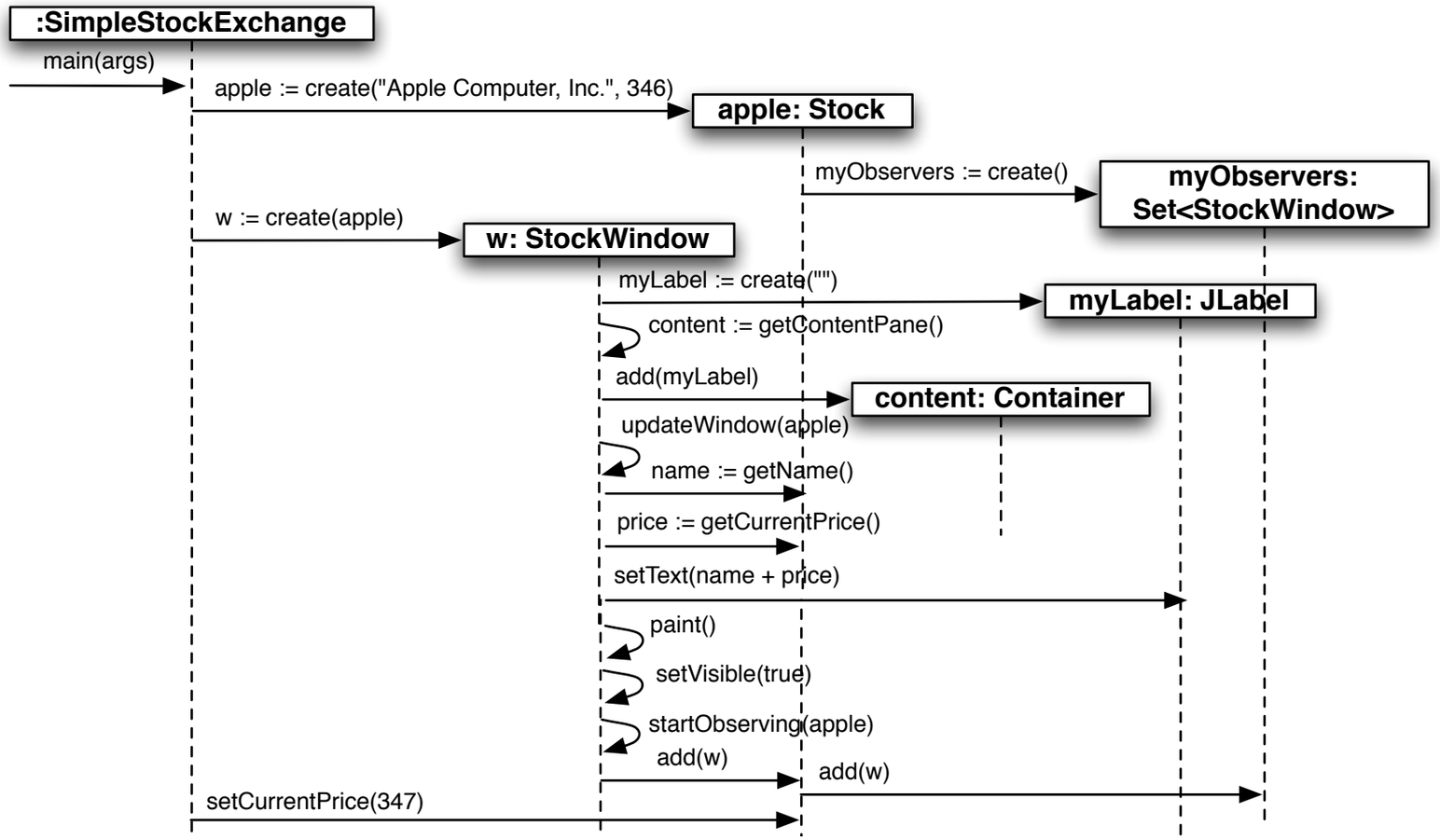
Weaving - diagramas de secuencia

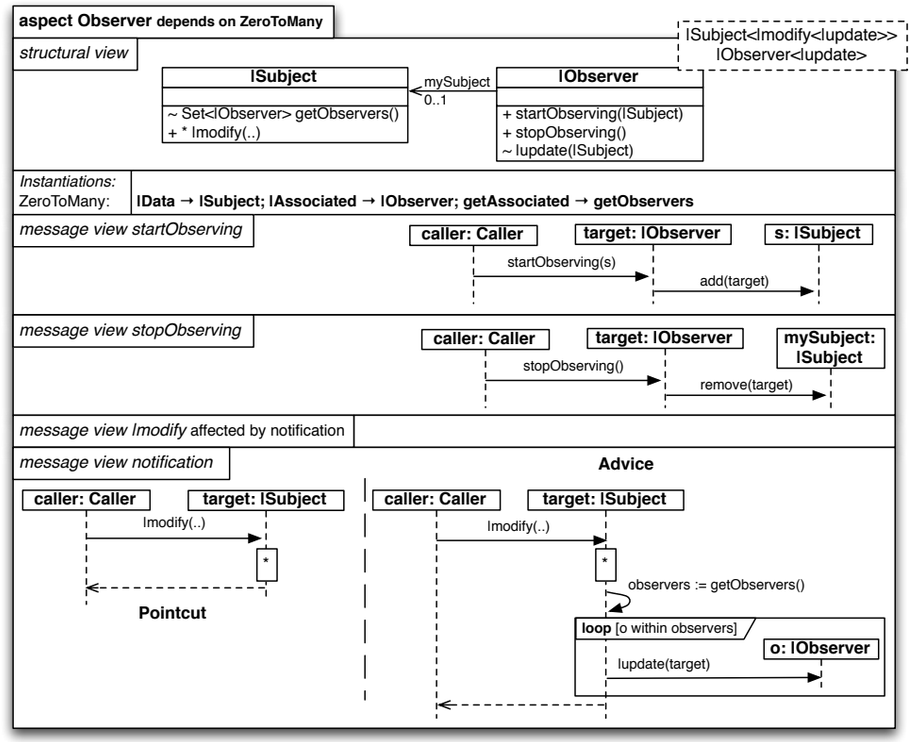


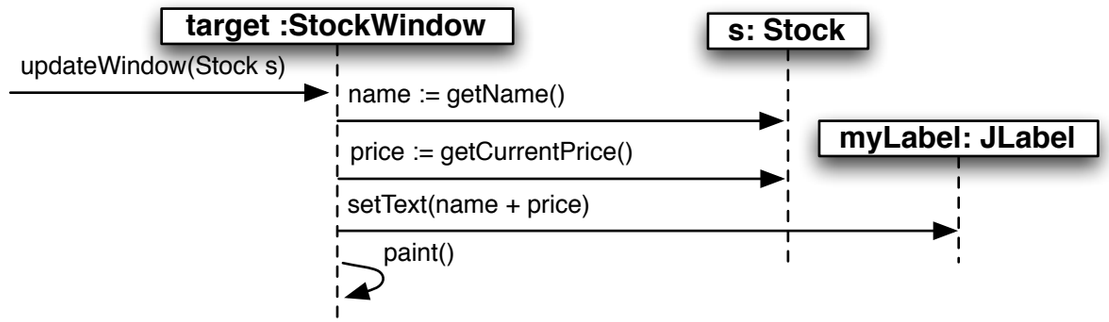




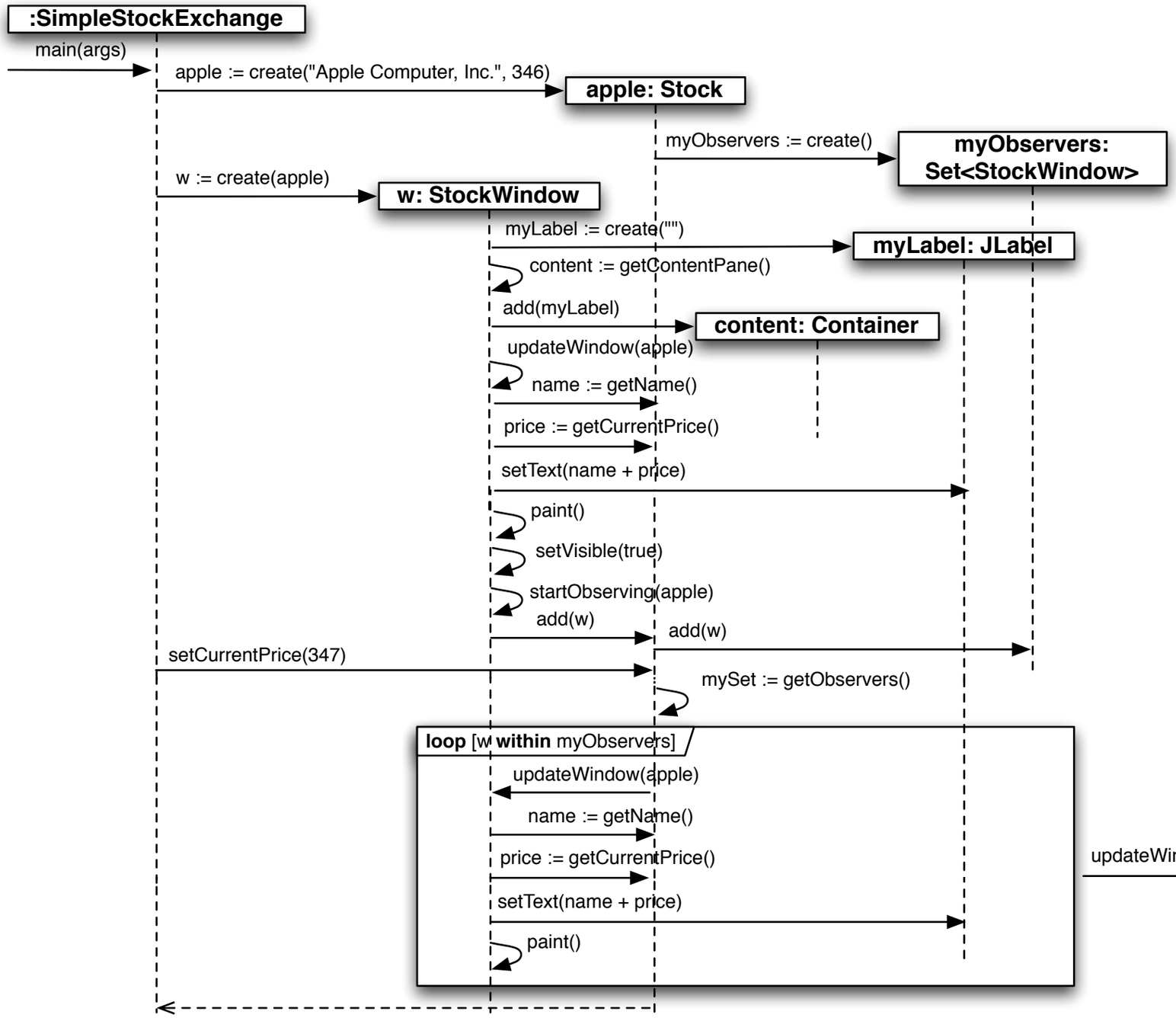
Weaving - diagramas de secuencia

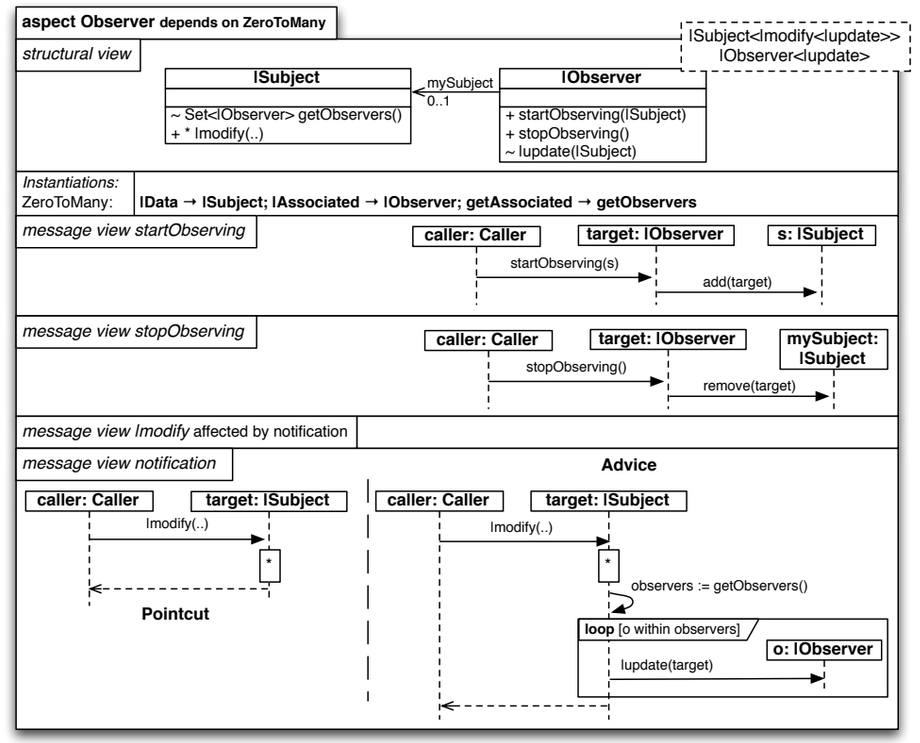


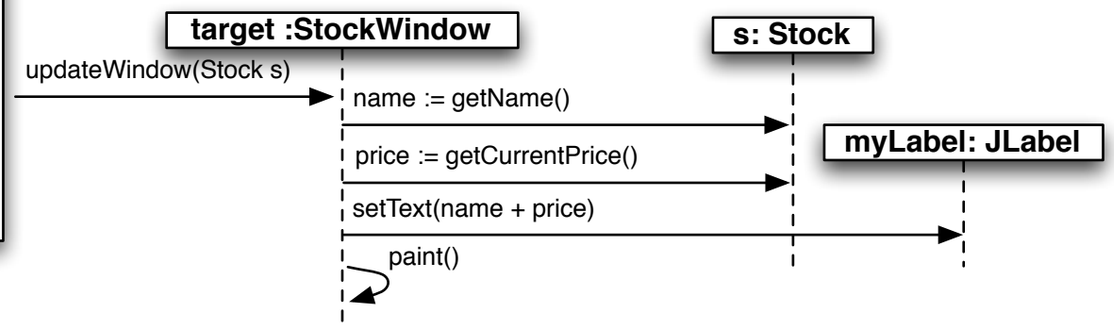




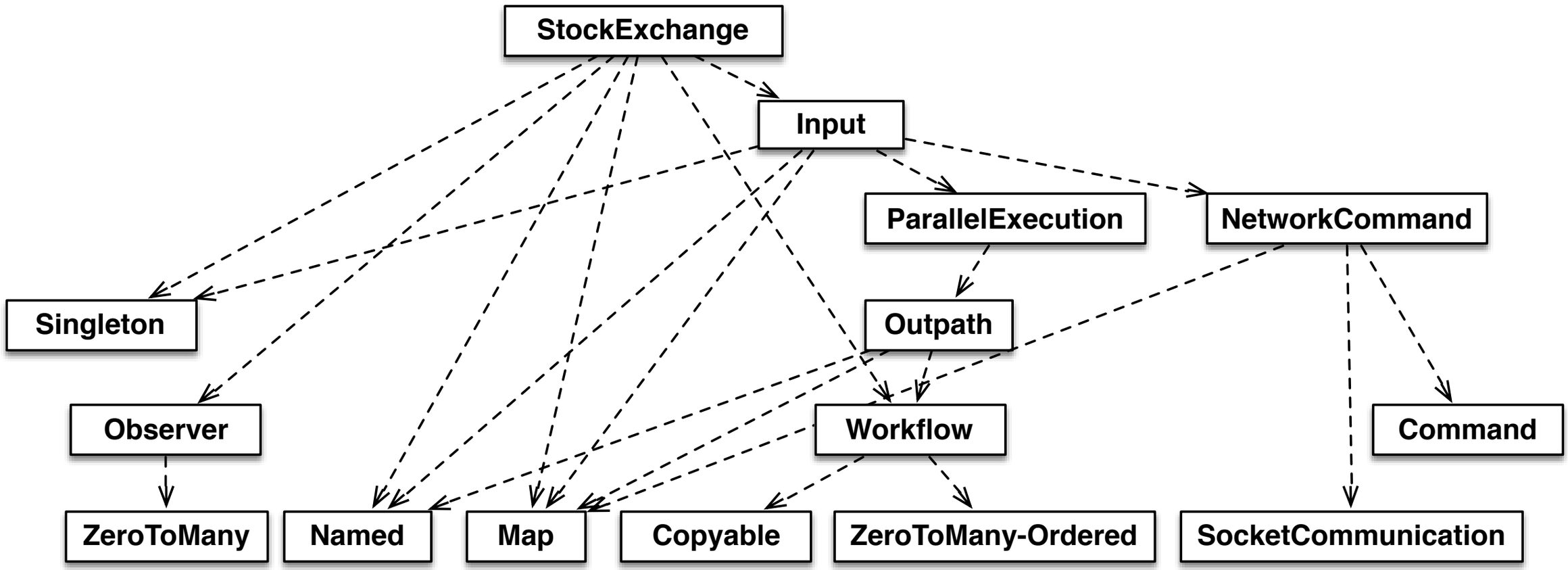
Weaving - diagramas de secuencia





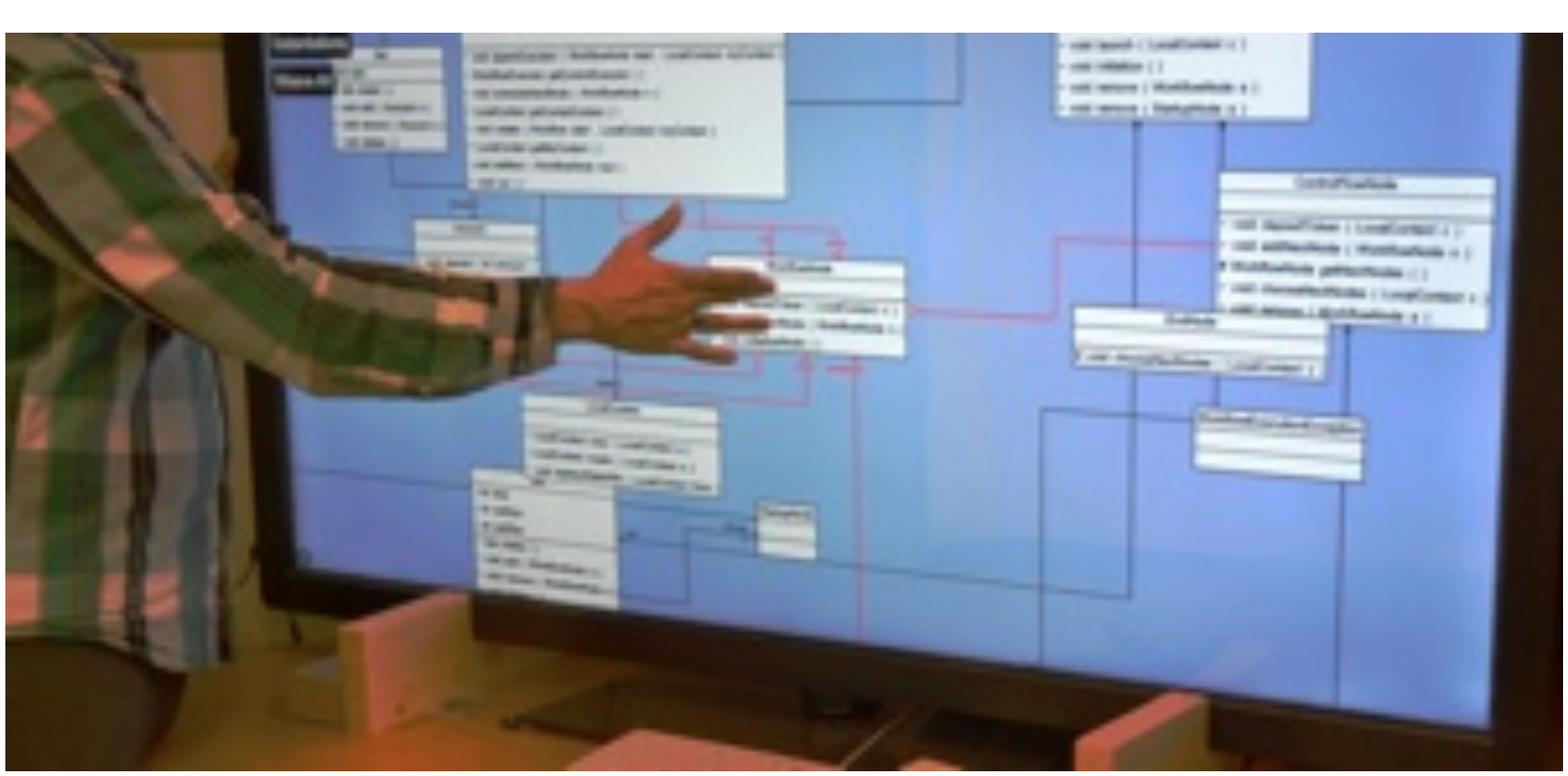


Jerarquía


StockExchange
Input
ParallelExecution
NetworkCommand
Singleton
Outpath
Observer
Workflow
Command
ZeroToMany
Named
Map
Copyable
ZeroToMany-Ordered
SocketCommunication

TouchRAM

- Modela aspectos reutilizables por medio de una notación gráfica y una interfaz táctil.



Oportunidad

"El formato **textual** es por lo general más usado, **escala mejor** y construirlo requiere mucho menos esfuerzo" [Voelter]

TextRAM

Un lenguaje de especificación de dominio textual, para el desarrollo y manejo de modelos RAM.

Estrategia

- Model Driven Software Engineering (MDSE)
- Domain-Specific-Languages (DSL)
- Language Workbench (Xtext, Xtend & Eclipse)
- Herramienta de visualización (Graphiz)

MDSE

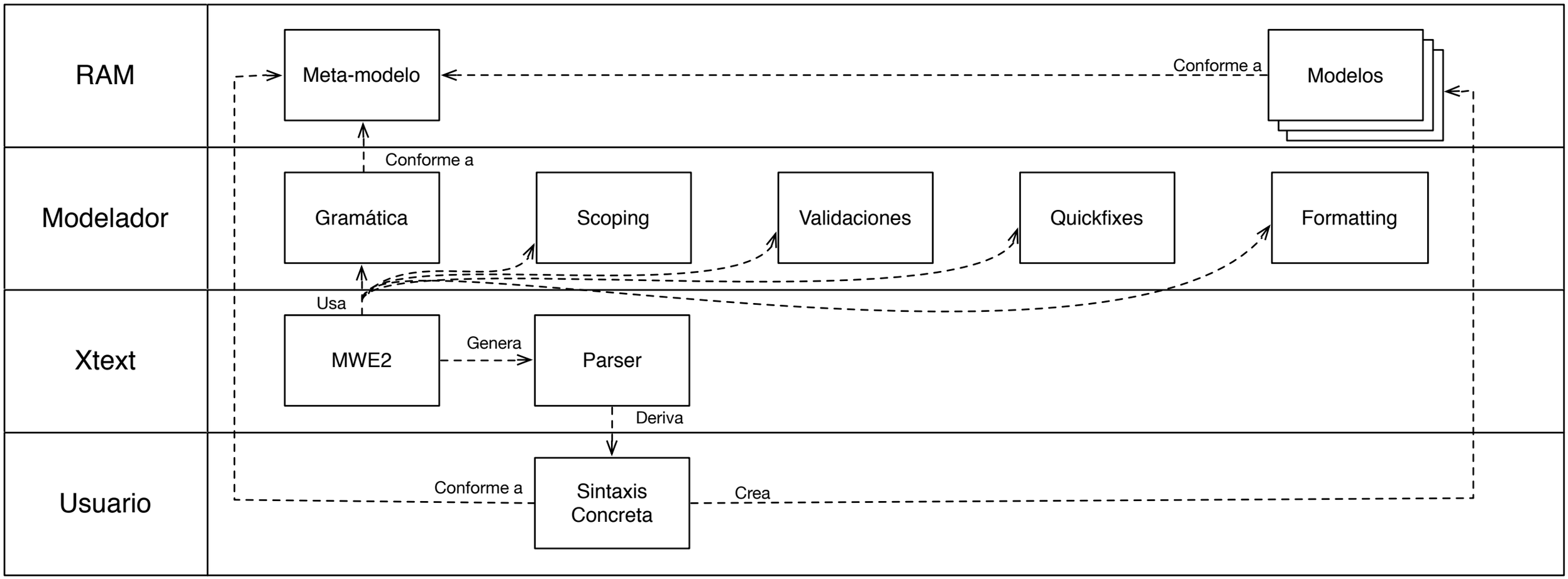
- Metodología para aplicar las ventajas del modelamiento a las actividades de ingeniería de software.
- Software = Modelos + Transformaciones
- Lenguaje de modelamiento (DSL)
- Meta-modelado
- Transformaciones (M2M, M2T)

General vs Expecífico

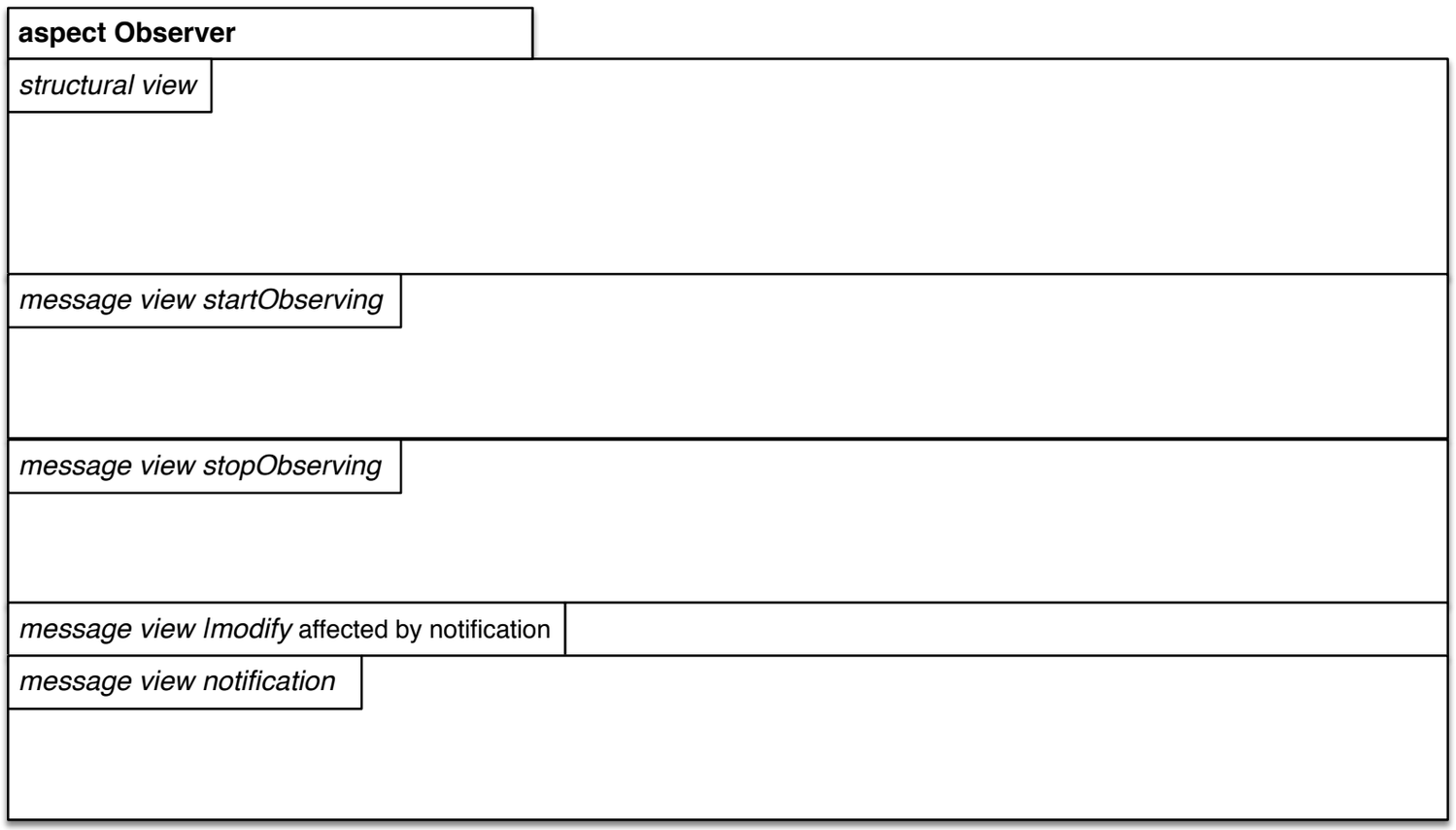
General vs Expecífico

DSL

Metodología

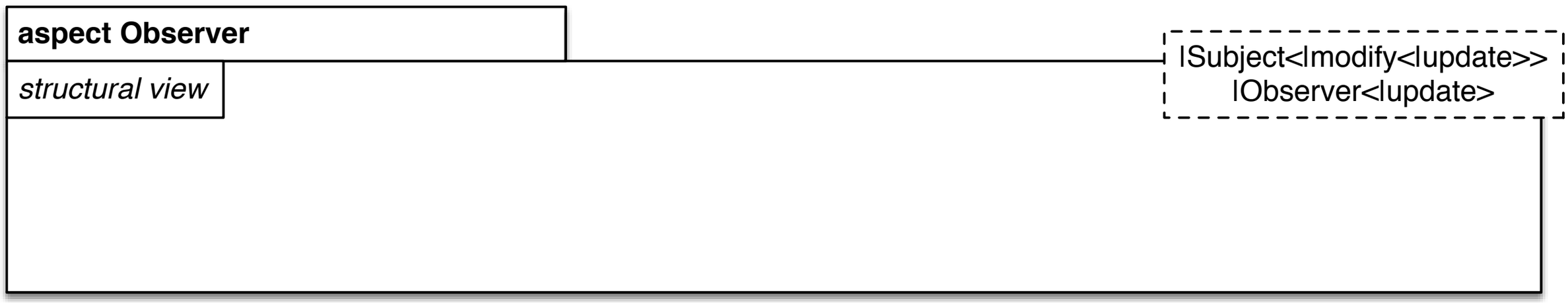


```
aspect Observer {
    structure { ... }

    message startObserving { ... }
    message stopObserving  { ... }
    message |modify affectedBy notification
    message notification   { ... }
}
```



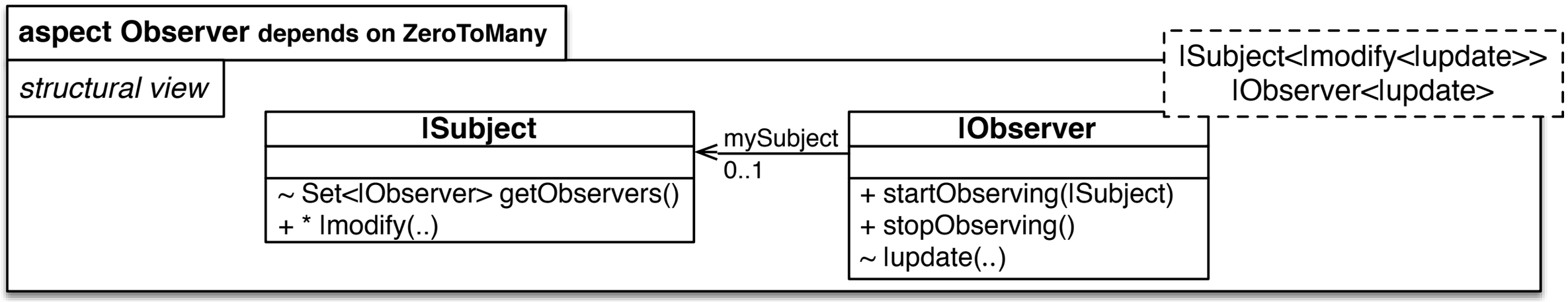
Aspect Model

```
aspect Observer {
    structure {
        mandatory {
            |Subject<|modify<|update>>
            |Observer<|update>
        }...
    }
}
```



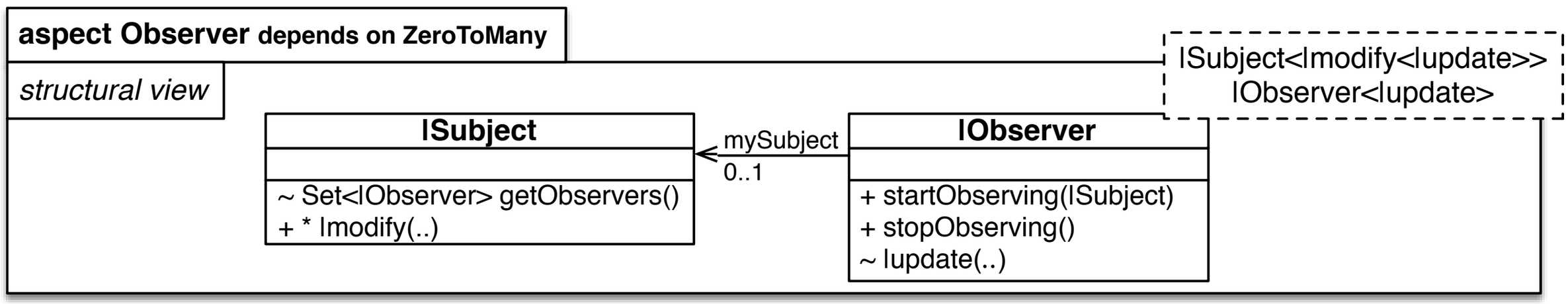
Mandatory instantiations

```
aspect Observer dependsOn ZeroToMany {
    structure {
        mandatory { ... }
        class |Subject {
            ~ Set<|Observer> getObservers()
            + *modify(..)
        }
        class |Observer {
            + startObserving(|Subject)
            + stopObserving ()
            ~ |update(..)
        }
    }
...
}
```



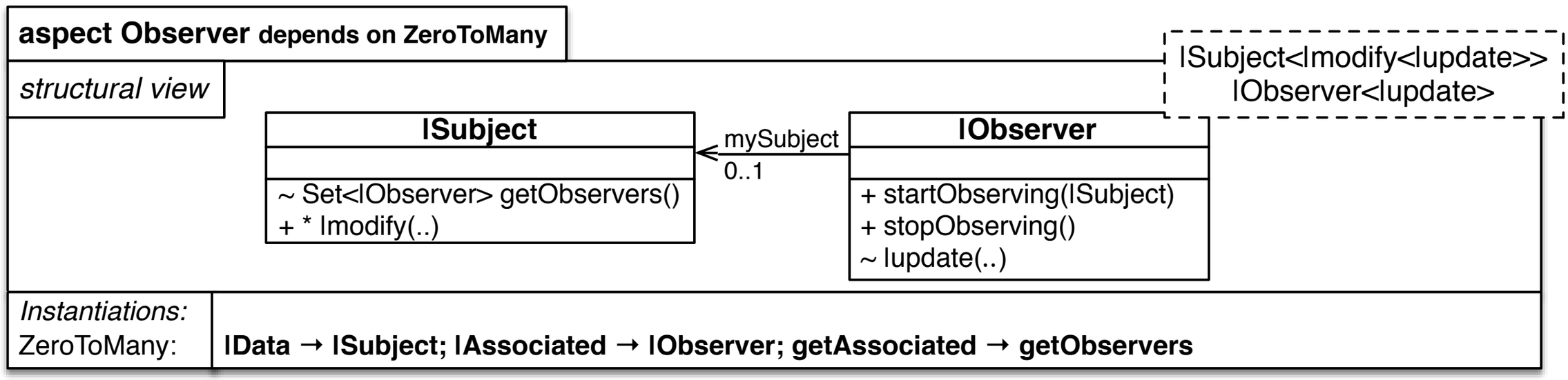
class definition

```
aspect Observer dependsOn ZeroToMany {
    structure {
        mandatory        { ... }
        class |Subject   { ... }
        class |Observer  { ... }

        associations {
            |Observer -> 0..1 |Subject { mySubject }
        }
    }
...
}
```



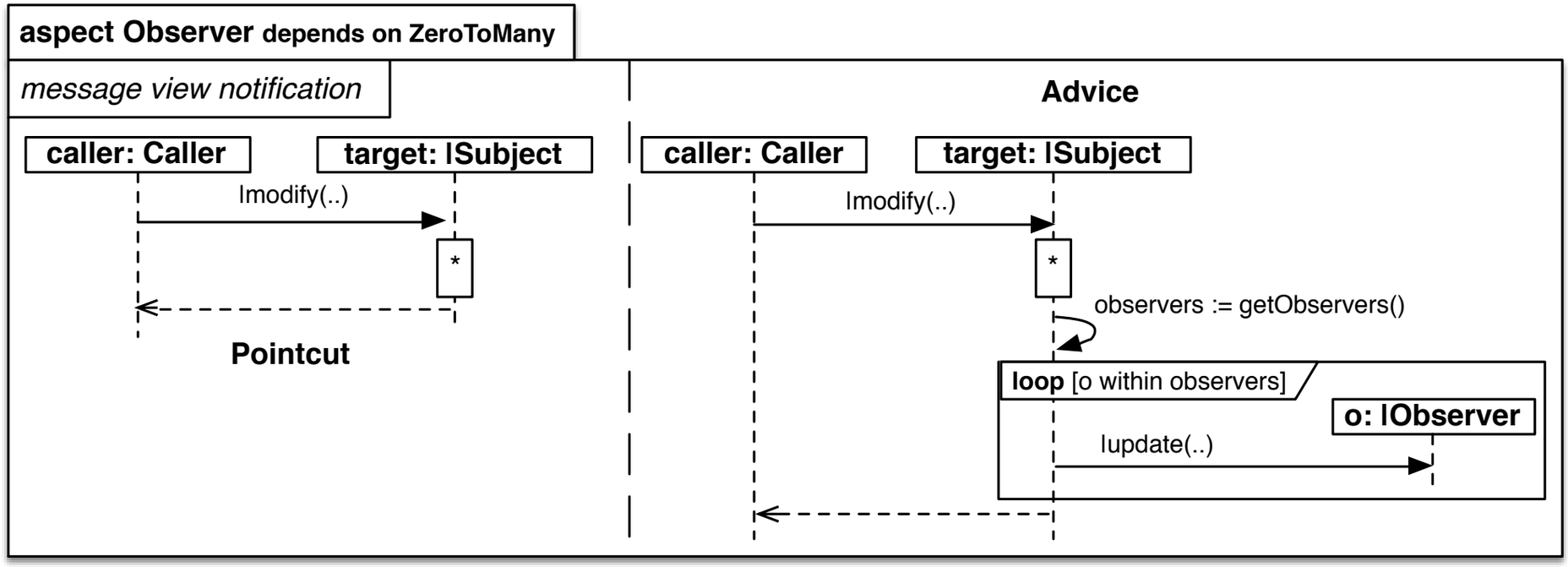
class association

```
aspect Observer dependsOn ZeroToMany {
    structure {
        mandatory        { ... }
        class |Subject   { ... }
        class |Observer  { ... }
        associations     { ... }

        instantiations  {
            |Data          -> Subject
            |Associated    -> |Observer
            getAssociated -> getObservers
        }

    }
...
}
```



instantiations

```
message notification {
     pointcut {
         caller:Caller   -> target:|Subject { |modify(..) } *
     }
     advice {
         caller:Caller   -> target:|Subject { |modify(..) } *
         target:|Subject -> target:|Subject { observers:= getObservers() }

         loop [o within observers] {
             target:|Subject -> o:|Observer { |update(..) }
         }
     }
 }
```



message view

Observer Aspect Model