Let’s have some fun together

Manisha Mahawar
twitter : @mmahawar
blog: http://braintwitter.blogspot.com/

Generics

- Why to use them
  - Runtime vs. compile time
- How to use them
  - Examples
- exercise:
  - implement a generic solution for stack

Enumerations

- What are enums
- Advantages and don’ts
  - readable, safer and more powerful
  - Strategy Type enum
  - interfaces can be used for enum
  - Switches on enum is not good idea
  - Try not to use ordinals
- exercise :
  - How to implement Singleton using enum

How to decouple client from implementations

- static factor method (Flyweight pattern)
  - to invoke, new object is not required ex. Boolean.valueOf
  - unlike constructor they have name ex. BigIneger. (int, int, Random) vs. BigInteger.probablePrime().
  - can return an object of any subtype of their return type ex. EnumSet (RegularEnumSet vs JumboEnumSet)
  - cannot be subclassed - composition instead of inheritance
  - does not stand out in API like constructors

Many constructor Parameters

- Lets code

Few don’ts

- Don’t forget to override toString - makes your class pleasant to use
- Don’t Abuse a class with static methods and fields - think in terms of object
- Watch
  - for objects while coding
  - for unintentional autoboxing
  - for Memory Leak in java
  - you must get it right the first time

Few do-essssss

- Always check parameter for validity
- When in doubt , leave it out
- Should return empty objects vs. nulls
- Declare variable before using
- for loops over while loop
- User StringBuffer vs. string concatenation.
- Refer to object by their interface

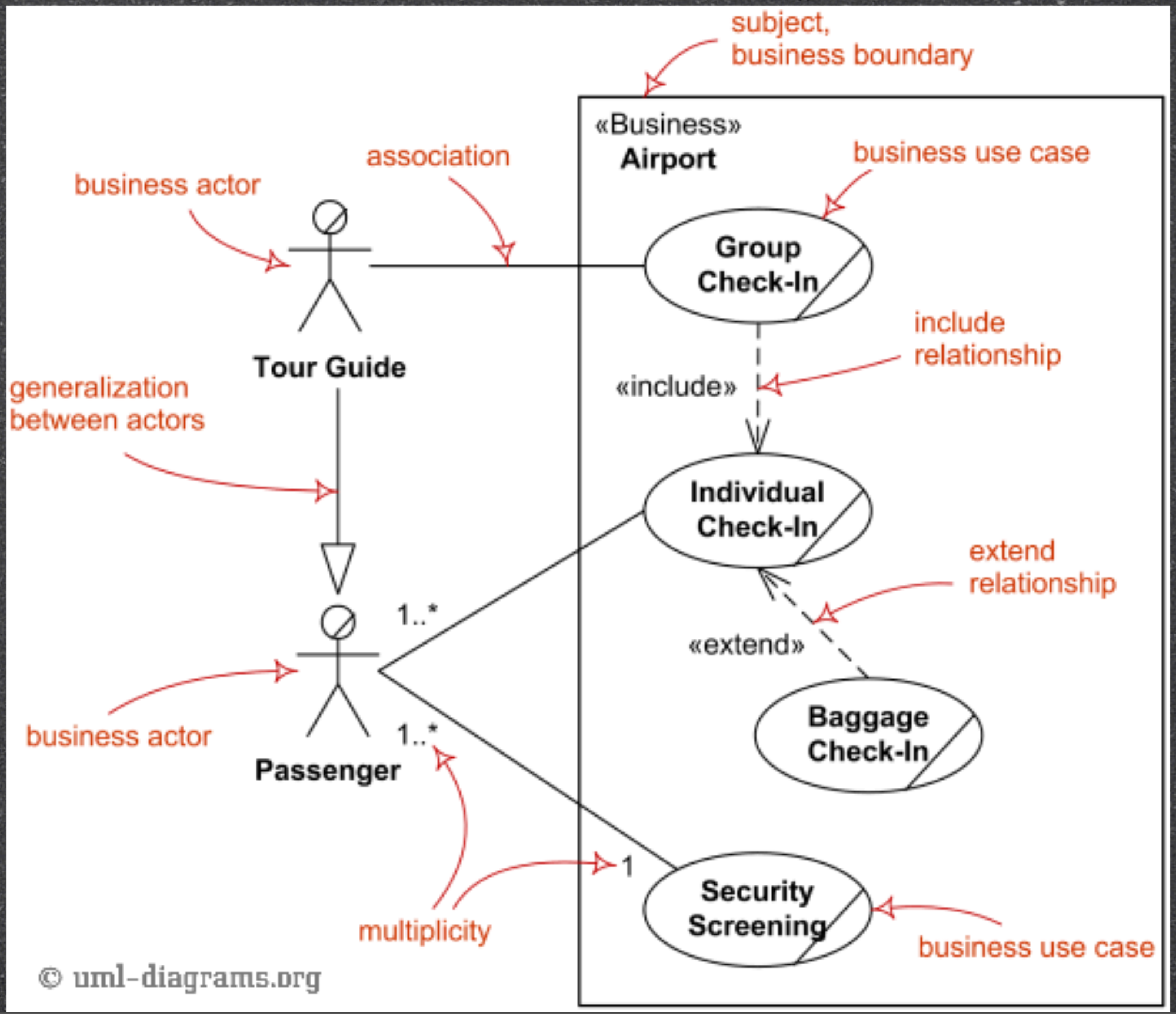
SOLIC Principle

- SRP: There should be only one reason for a class to change
- OCP: A module should be open for extension but closed for modification.
- LSP: In order to be substituted the contract of the base class must be honored by derived class.
- ISP: Many client specific interfaces are better than one general purpose interface
- DIP: Depend upon Abstraction. Do not depend upon concretions.

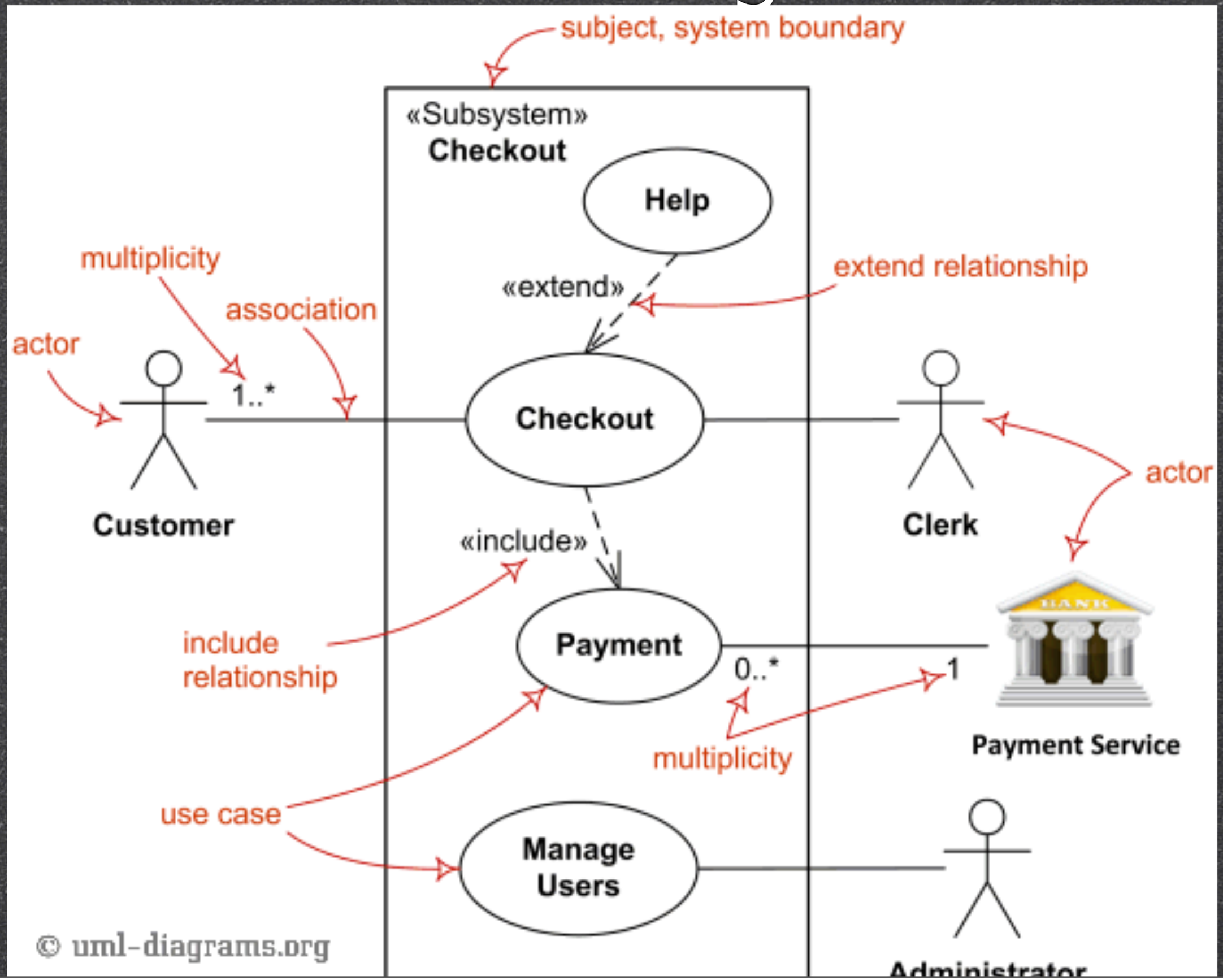
UML Diagram

- Use case Diagram
- Class Diagram
- Sequence Diagram
- Activity Diagram
- Deployment Diagram

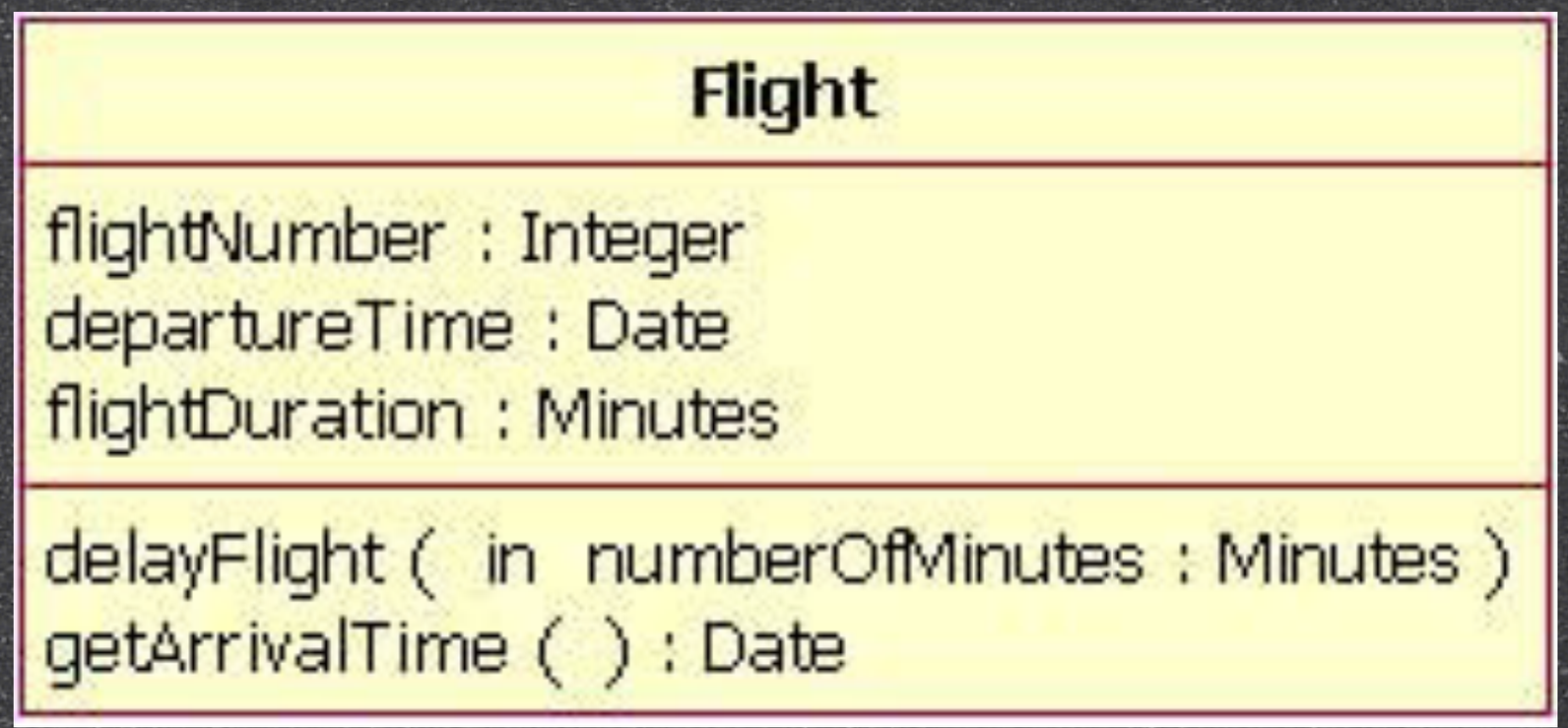
Use case diagram



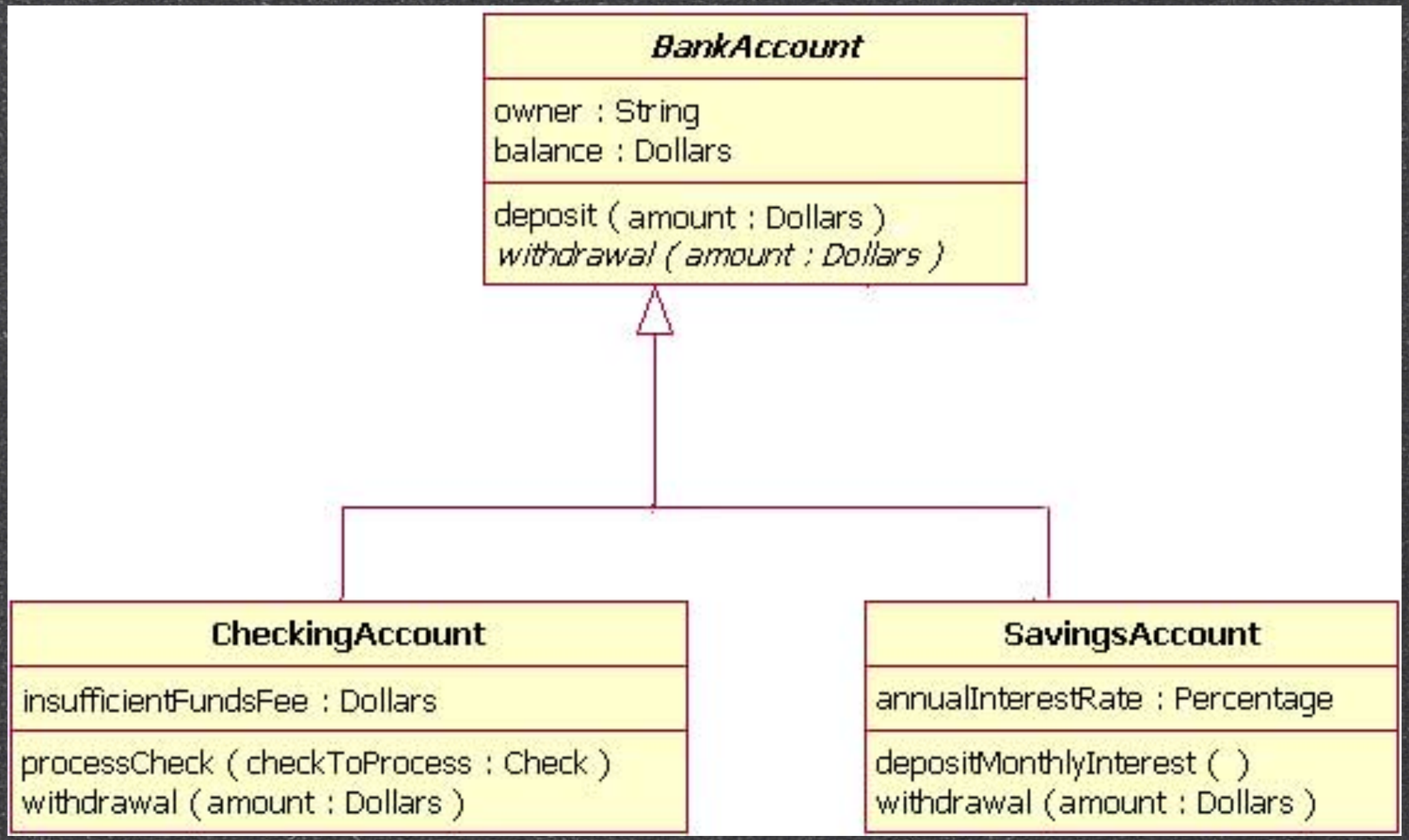
Use case diagram...



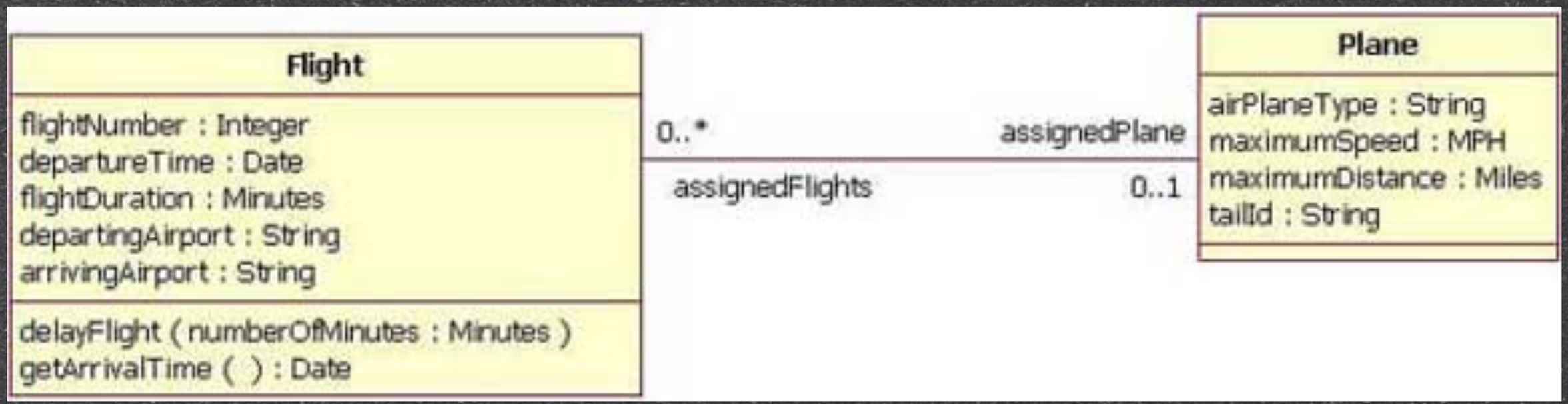
Class diagram

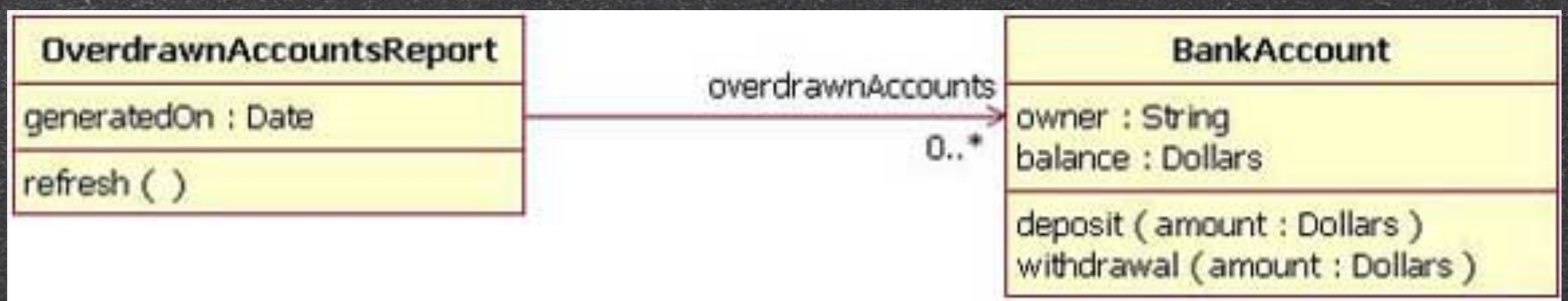


CD - > Inheritance

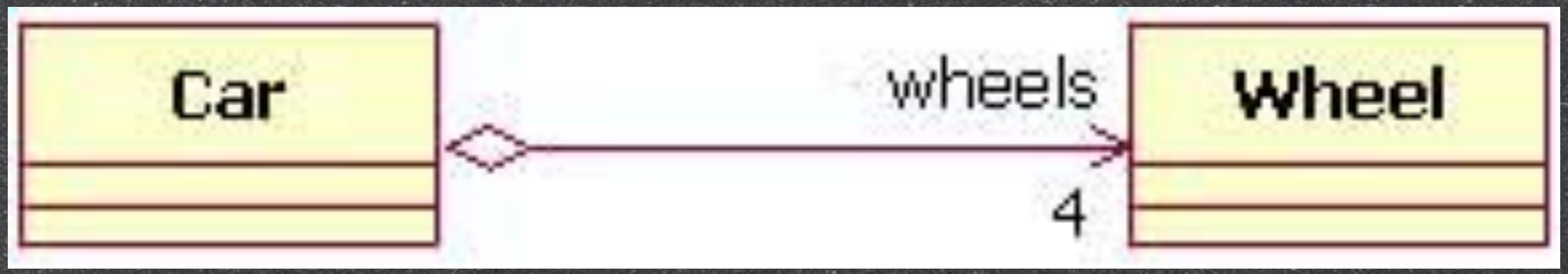


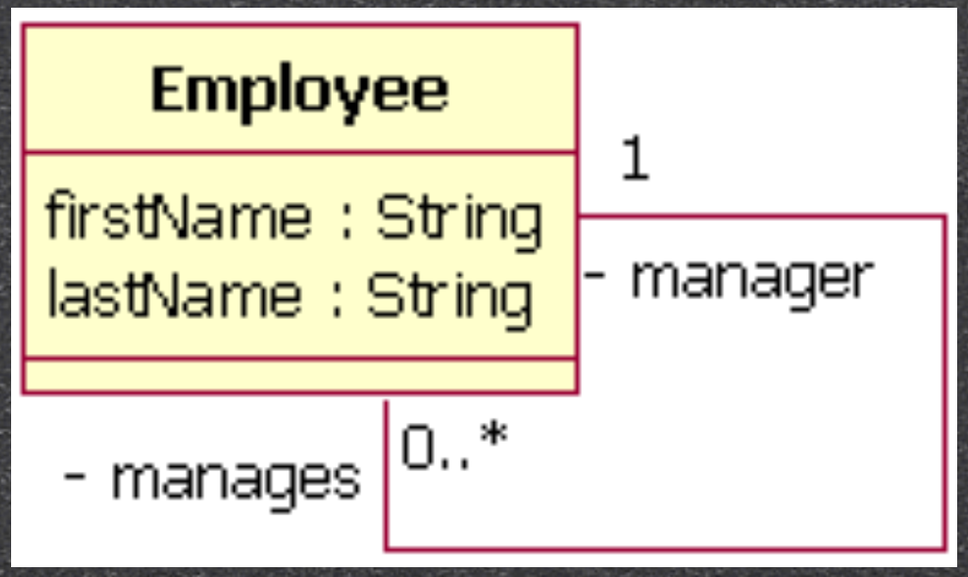
CD->Association





CD->Aggregation

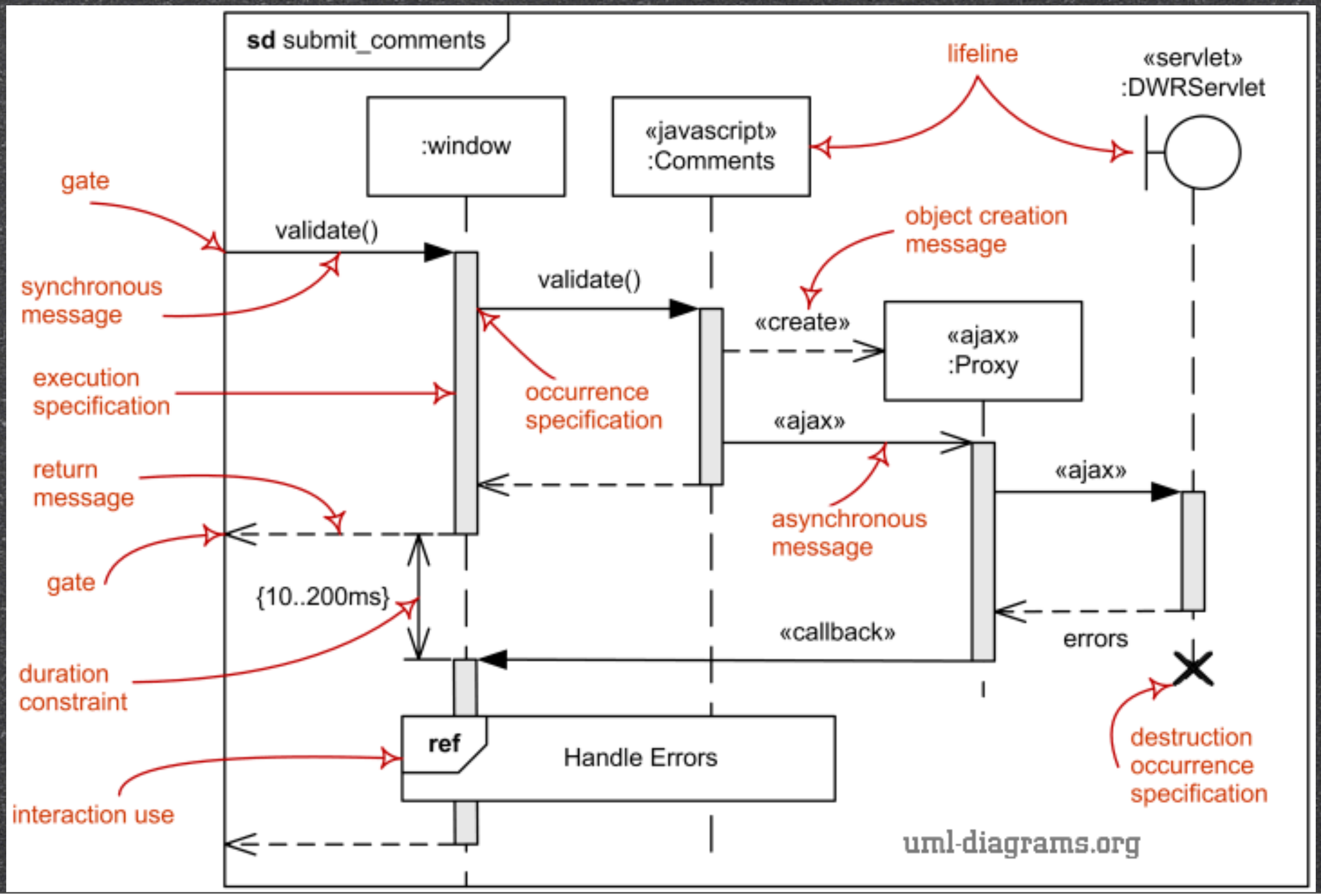




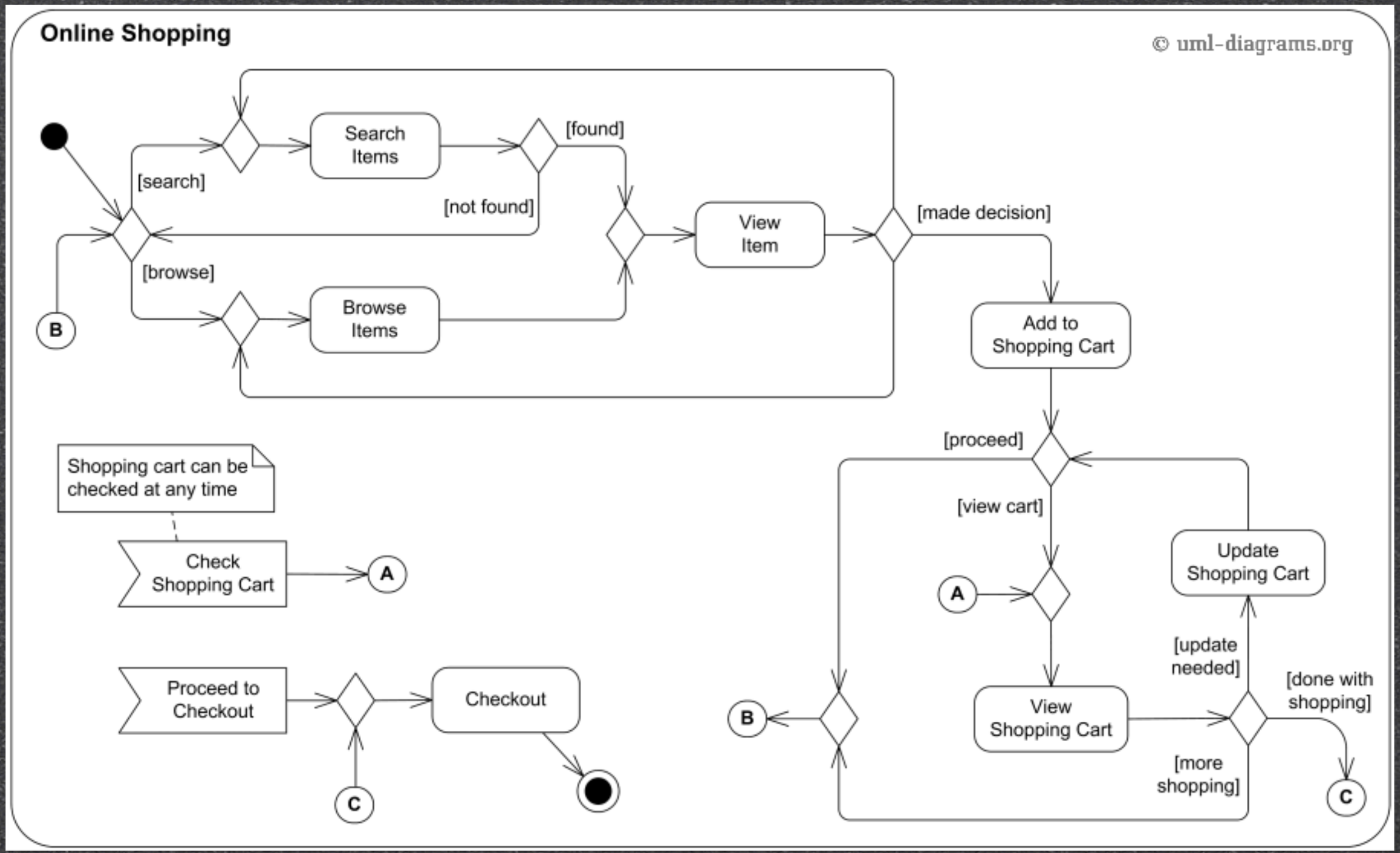
Visibility Type

+	Public
-	Private
#	Protected
~	Package

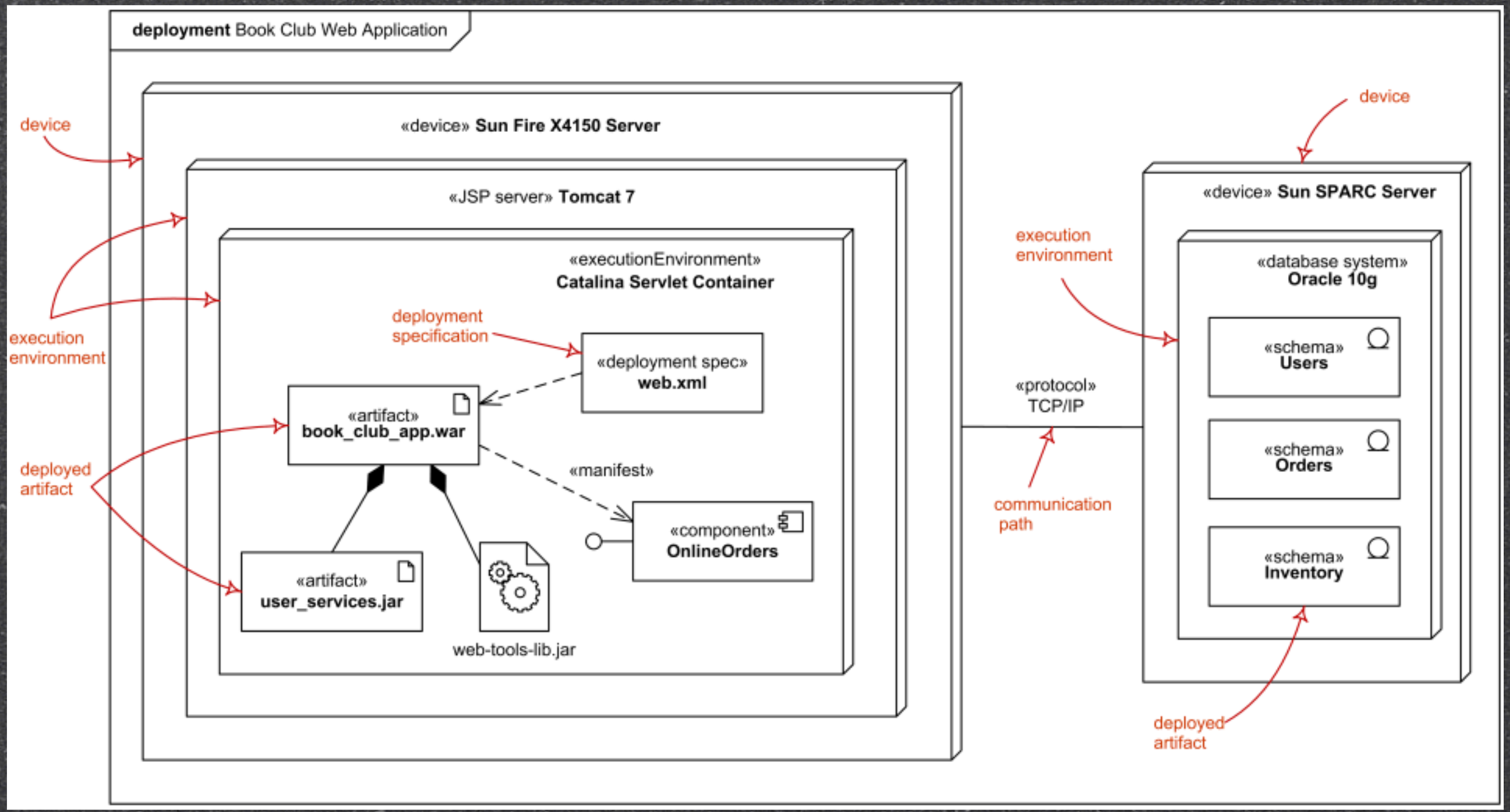
Sequence Diagram



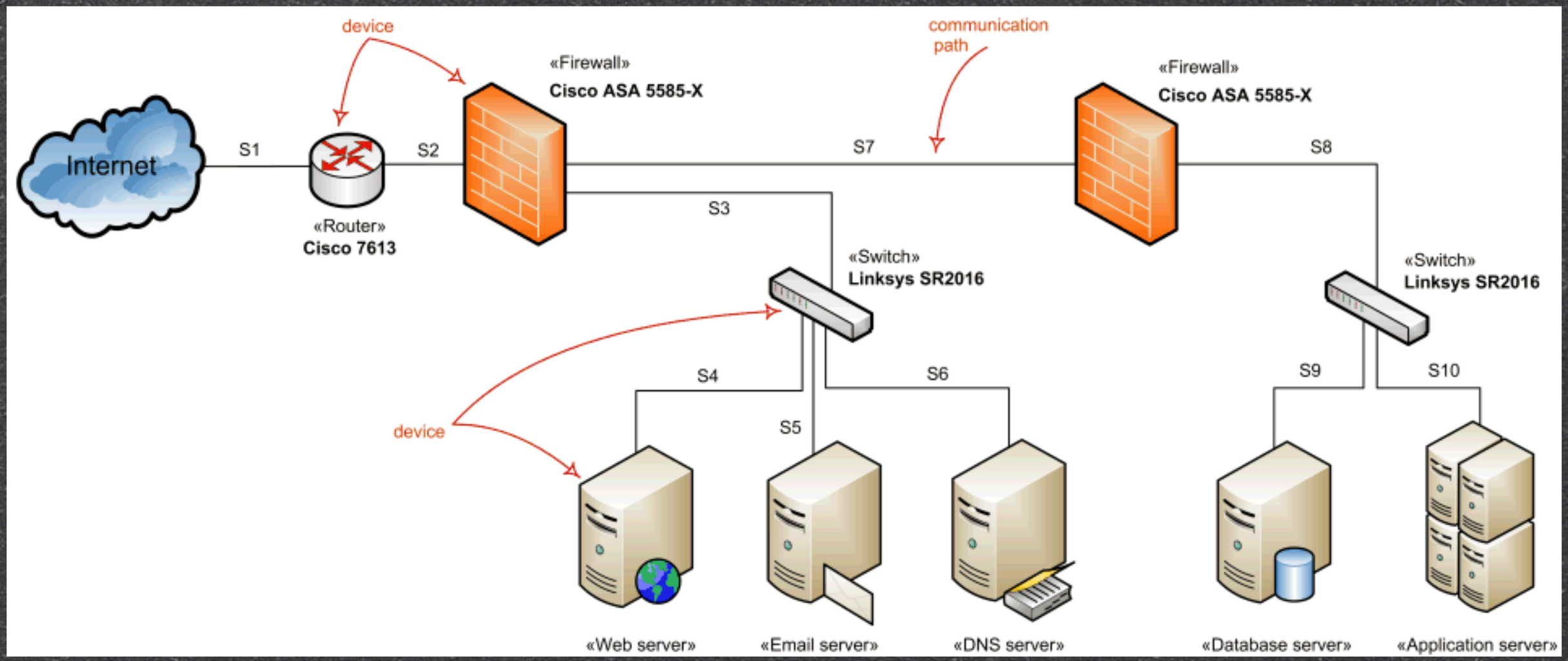
Activity Diagram



Deployment Diagram



Network Architecture Diagram



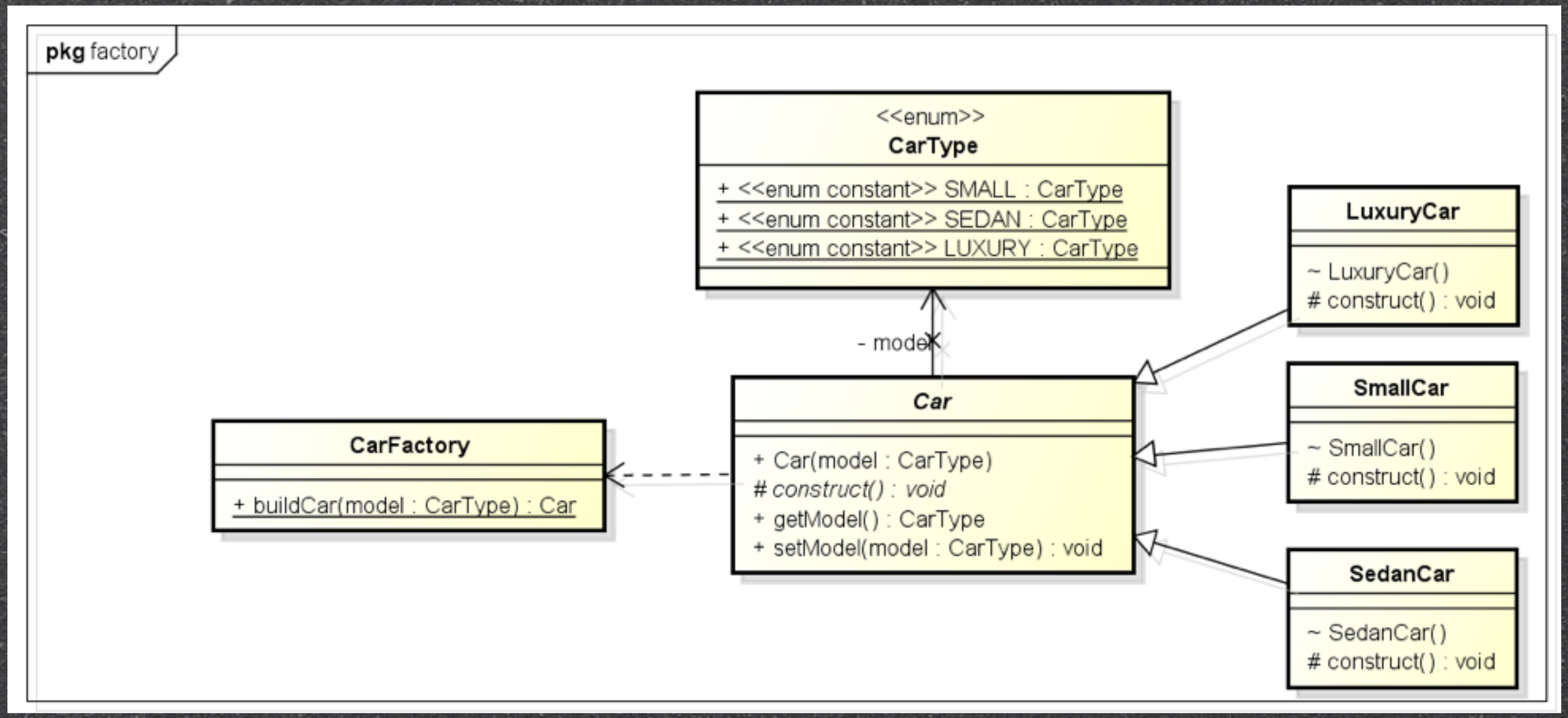
Design

- Continuous care vs. initial dump
- http://stackoverflow.com/questions/1673841/examples-of-gof-design-patterns

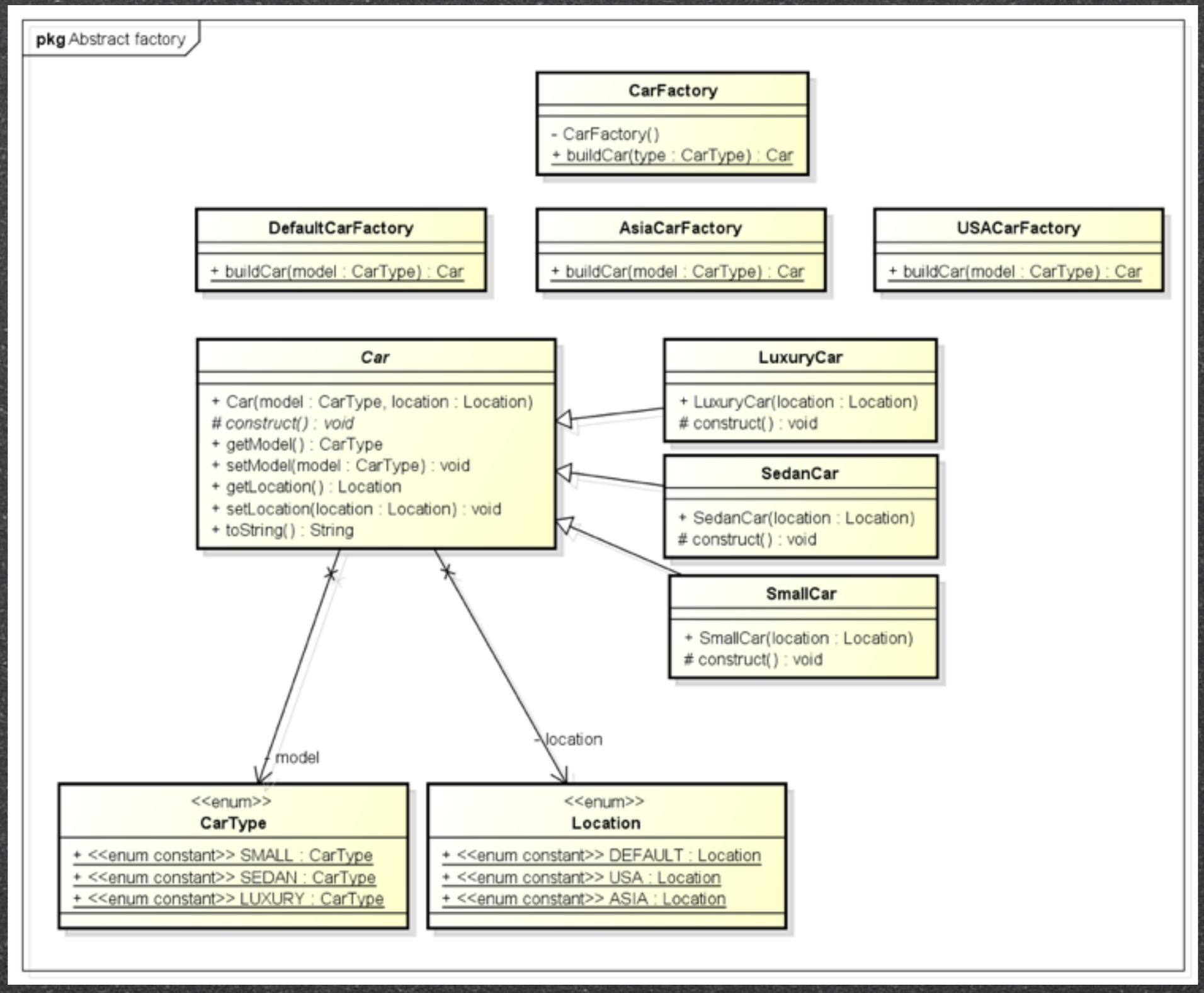
Singleton

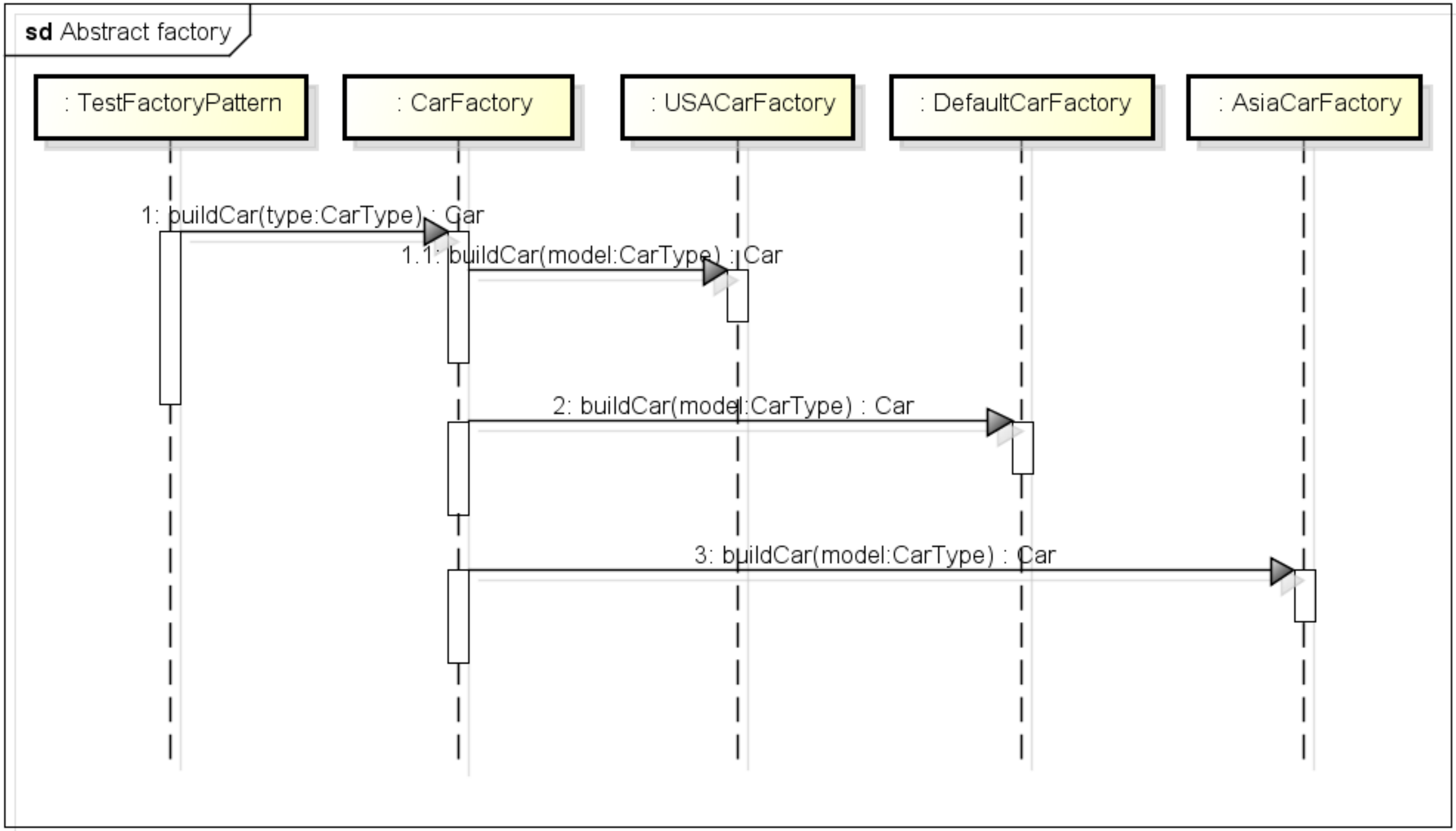
- A singleton is simply a class that is instantiated exactly once.
  - Singleton classes are very difficult to test....

Factory

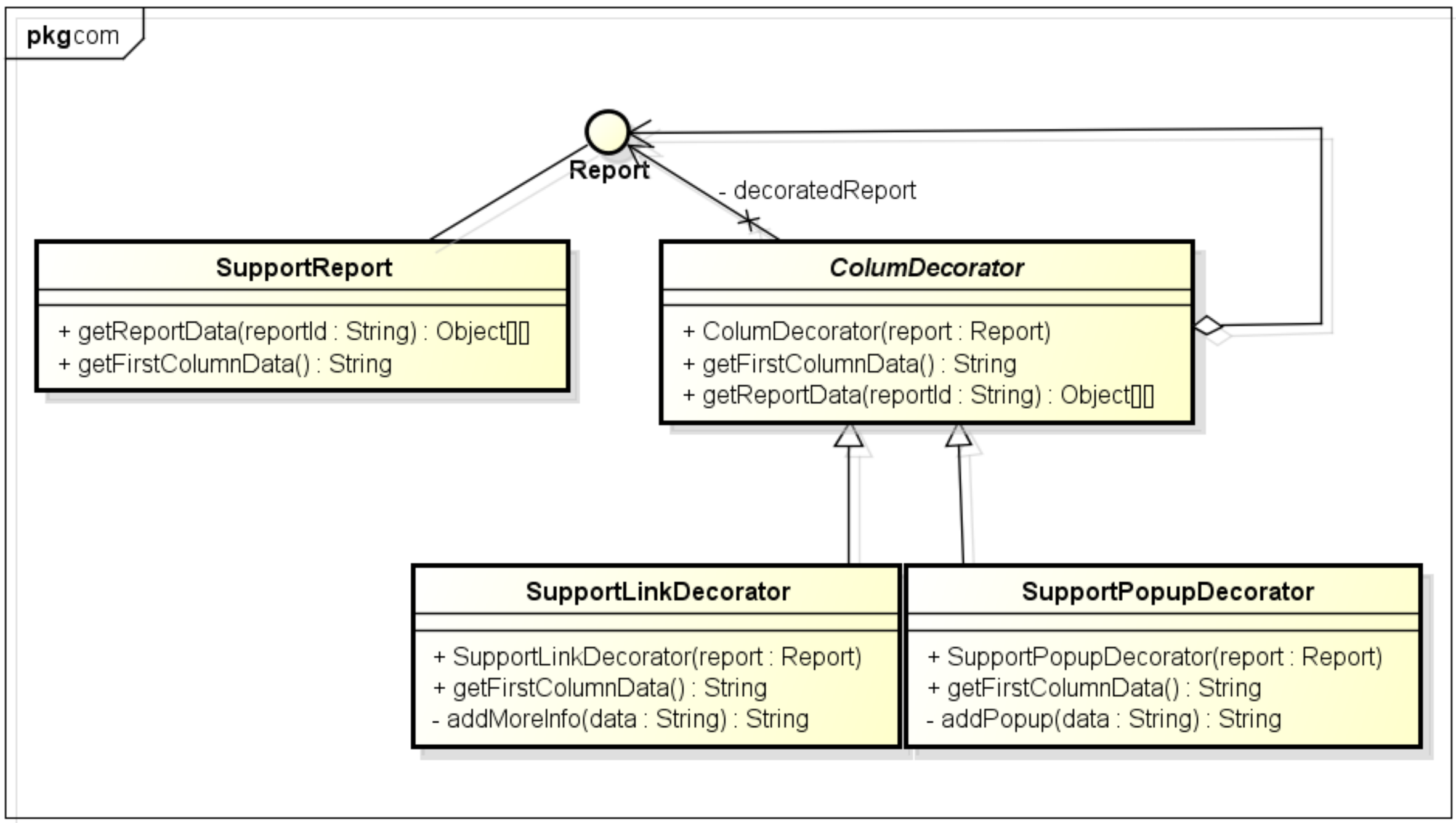

pkg factory
<<enum>>
CarType
+ <<enum constant>> SMALL : CarType
+ <<enum constant>> SEDAN : CarType
+ <<enum constant>> LUXURY : CarType
- model
Car
+ Car(model : CarType)
# construct() : void
+ getModel() : CarType
+ setModel(model : CarType) : void
CarFactory
+ buildCar(model : CarType) : Car
LuxuryCar
~ LuxuryCar()
# construct() : void
SmallCar
~ SmallCar()
# construct() : void
SedanCar
~ SedanCar()
# construct() : void

Abstract Factory

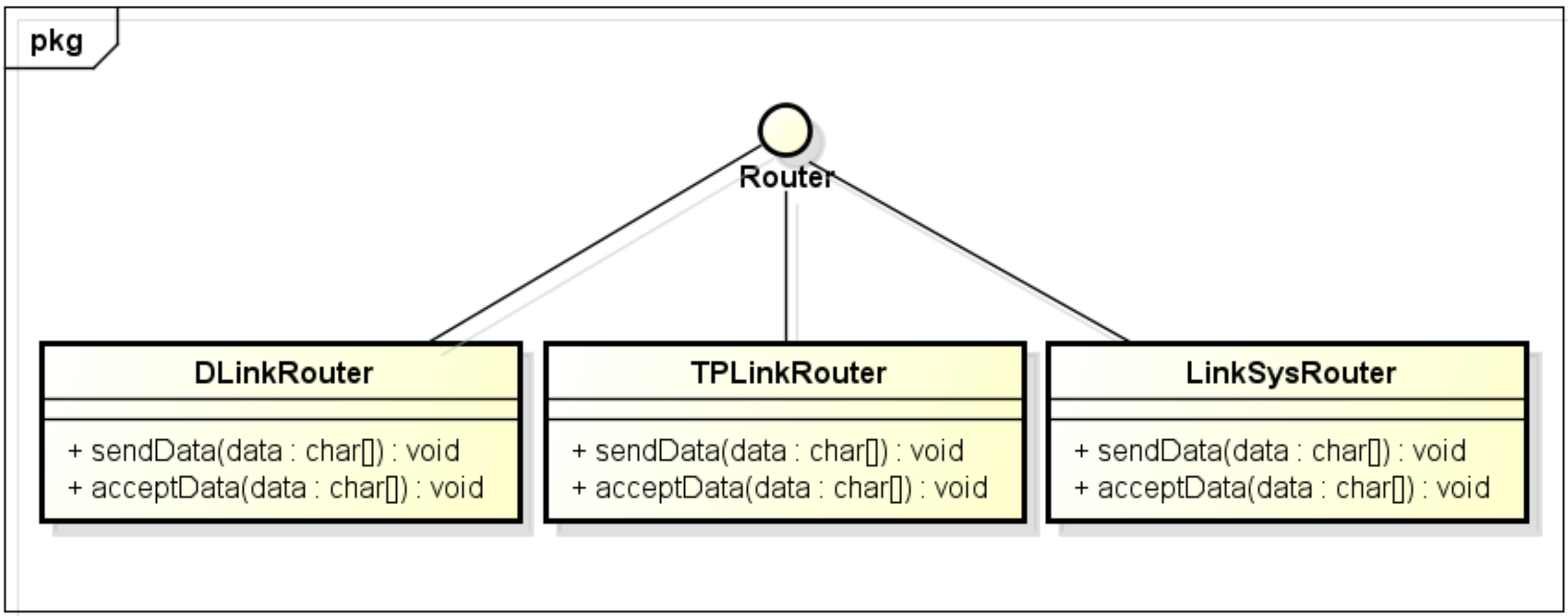



sd Abstract factory
: TestFactoryPattern
: CarFactory
: USACarFactory
: DefaultCarFactory
: AsiaCarFactory
1: buildCar(type:CarType) : Car
1.1: buildCar(model:CarType) : Car
2: buildCar(model:CarType) : Car
3: buildCar(model:CarType) : Car

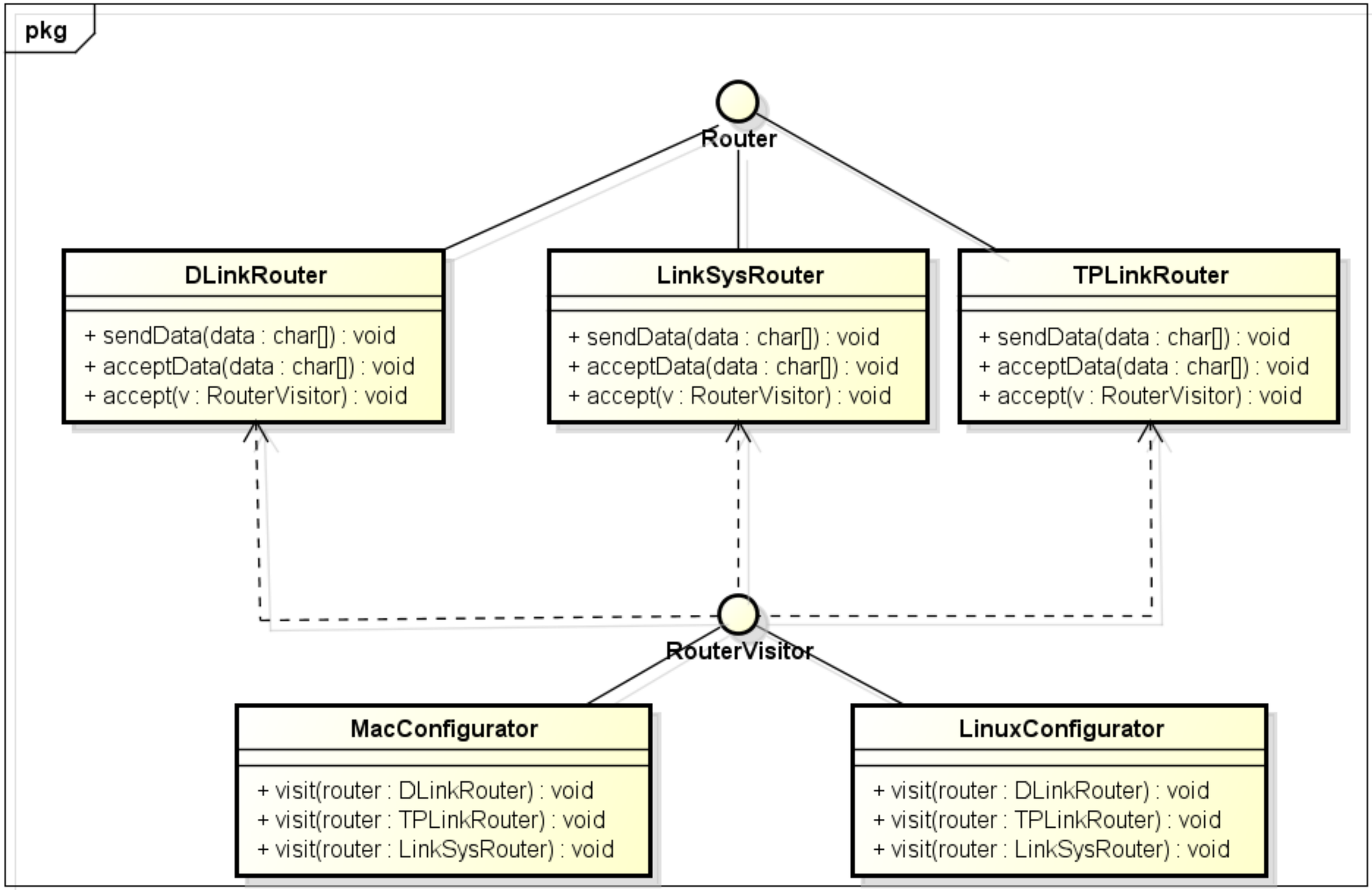
Decorator


pkgcom
Report
- decoratedReport
SupportReport
+ getReportData(reportId : String) : Object[][]
+ getFirstColumnData() : String
ColumDecorator
+ ColumDecorator(report : Report)
+ getFirstColumnData() : String
+ getReportData(reportId : String) : Object[][]
SupportLinkDecorator
+ SupportLinkDecorator(report : Report)
+ getFirstColumnData() : String
- addMoreInfo(data : String) : String
SupportPopupDecorator
+ SupportPopupDecorator(report : Report)
+ getFirstColumnData() : String
- addPopup(data : String) : String

Visitor


pkg
Router
DLinkRouter
+ sendData(data : char[]) : void
+ acceptData(data : char[]) : void
TPLinkRouter
+ sendData(data : char[]) : void
+ acceptData(data : char[]) : void
LinkSysRouter
+ sendData(data : char[]) : void
+ acceptData(data : char[]) : void

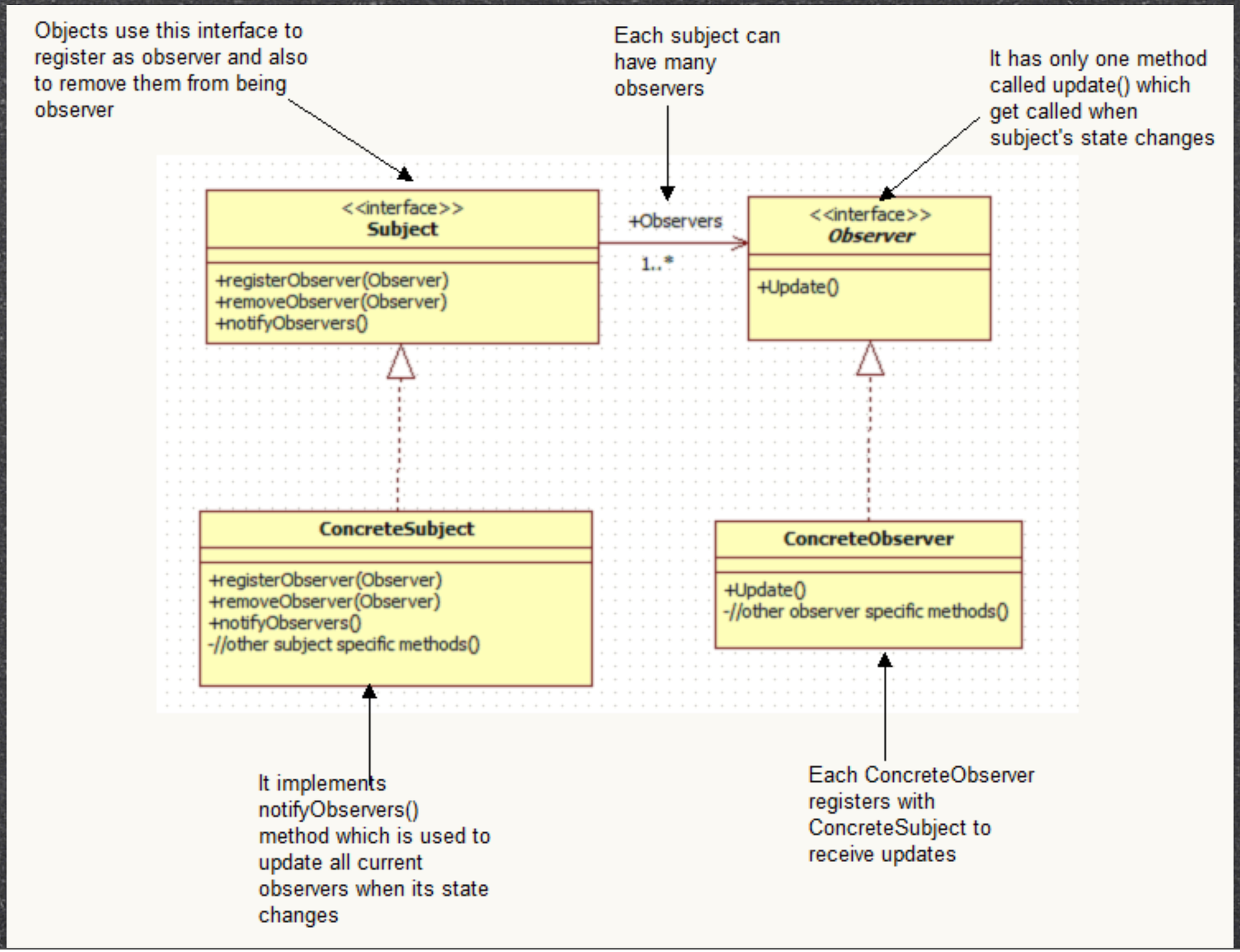
Visitor ...


pkg
Router
DLinkRouter
+ sendData(data : char[]) : void
+ acceptData(data : char[]) : void
+ accept(v : RouterVisitor) : void
LinkSysRouter
+ sendData(data : char[]) : void
+ acceptData(data : char[]) : void
+ accept(v : RouterVisitor) : void
TPLinkRouter
+ sendData(data : char[]) : void
+ acceptData(data : char[]) : void
+ accept(v : RouterVisitor) : void
RouterVisitor
MacConfigurator
+ visit(router : DLinkRouter) : void
+ visit(router : TPLinkRouter) : void
+ visit(router : LinkSysRouter) : void
LinuxConfigurator
+ visit(router : DLinkRouter) : void
+ visit(router : TPLinkRouter) : void
+ visit(router : LinkSysRouter) : void

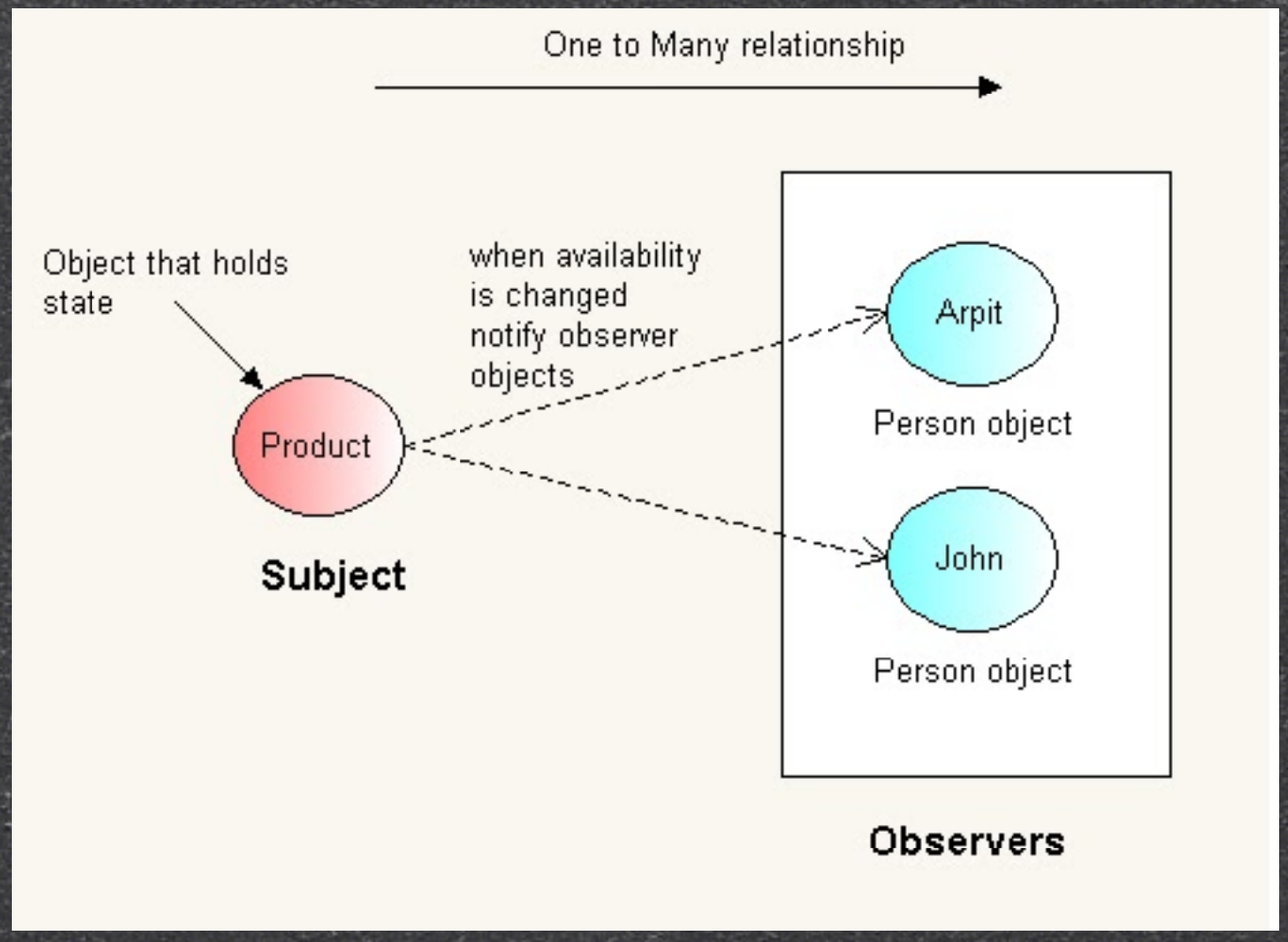
Builder

- We looked at it before remember NutritionFacts class...
- What you folks feel about builder?

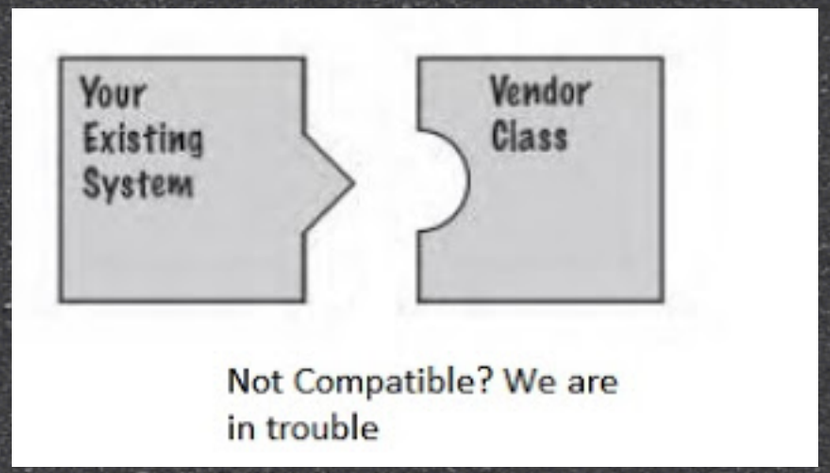
Observer:don't call us, we will call you

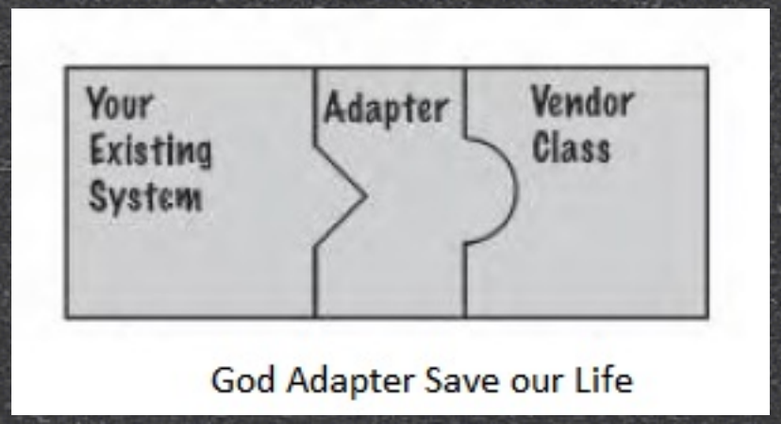


Observer...

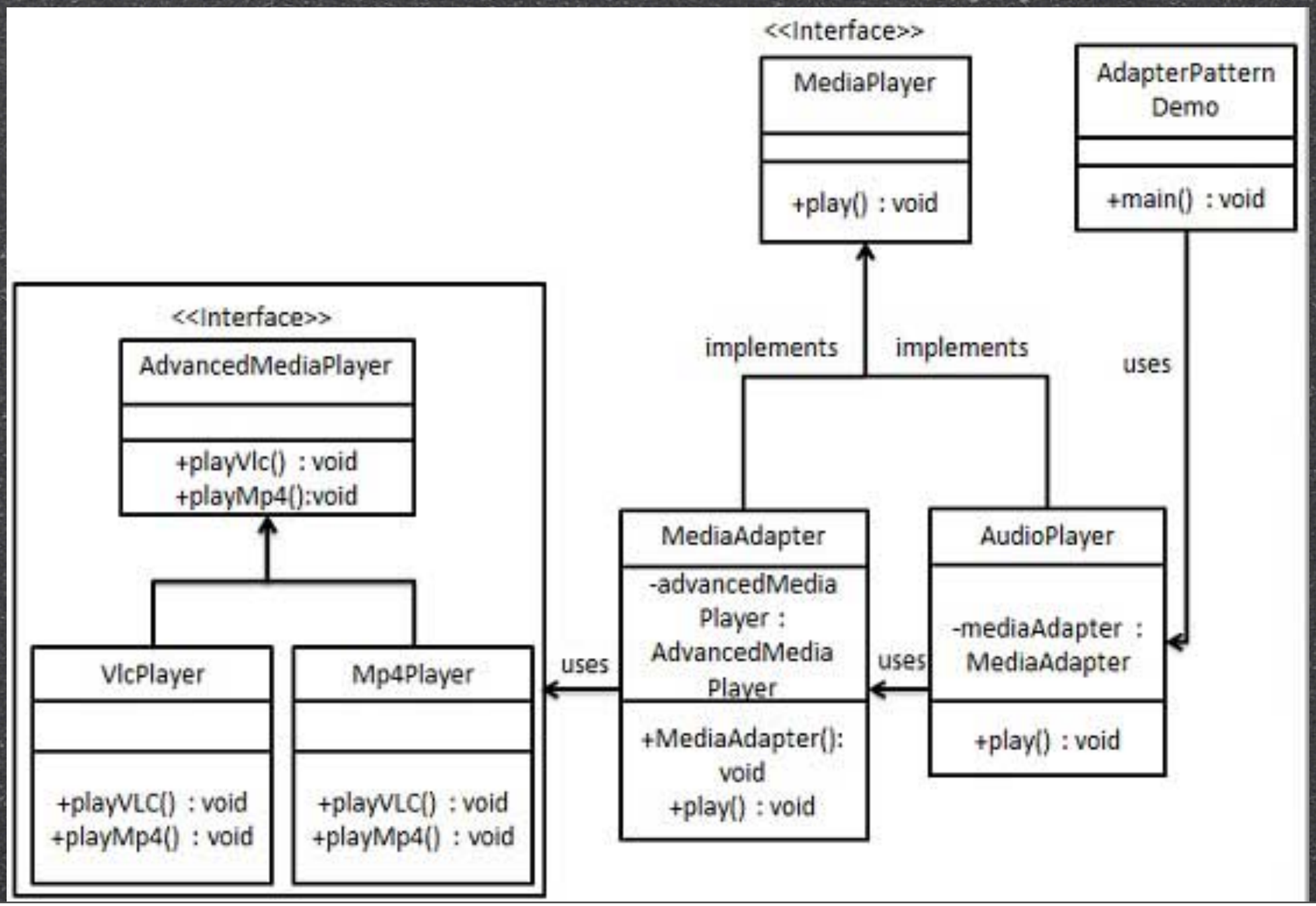

One to Many relationship
Object that holds state
when availability is changed notify observer objects
Product
Subject
Arpit
Person object
John
Person object
Observers

Adaptor

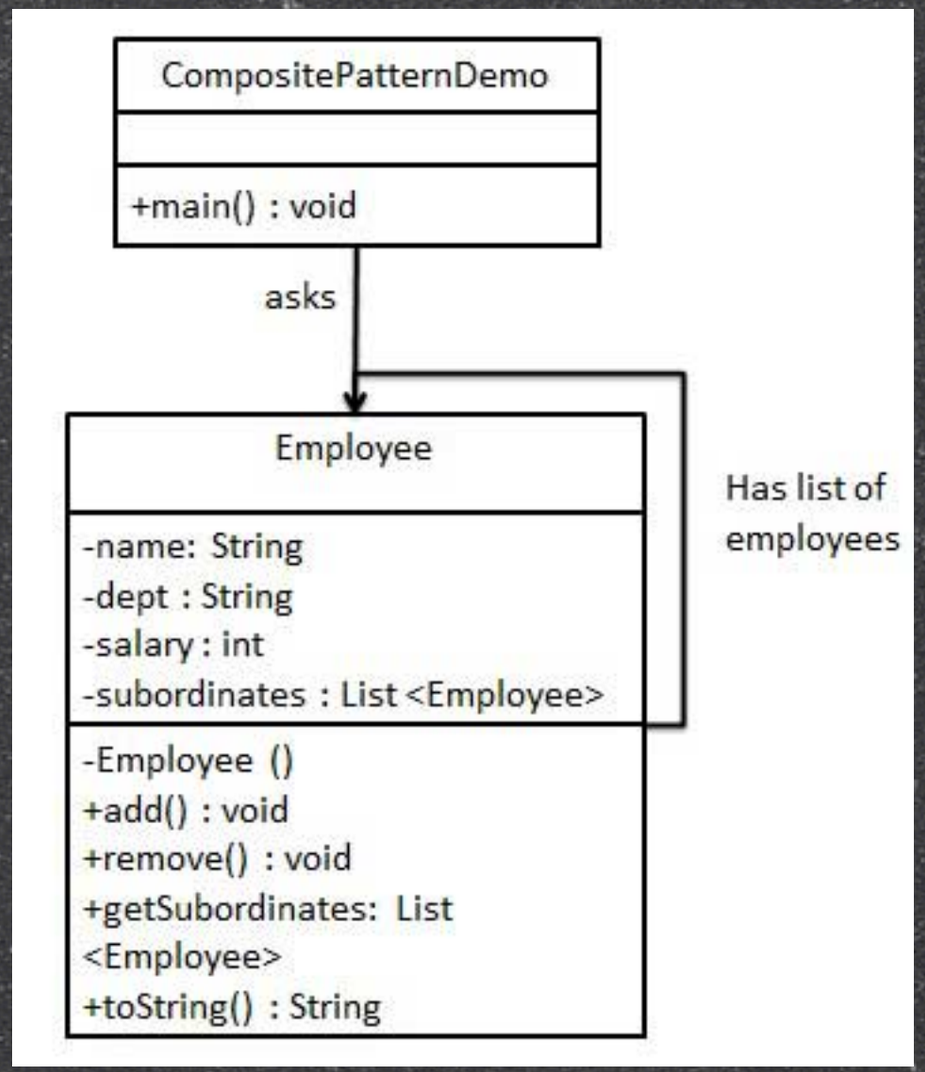

Your Existing System
Vendor Class
Not Compatible? We are in trouble


Your Existing System
Adapter
Vendor Class
God Adapter Save our Life

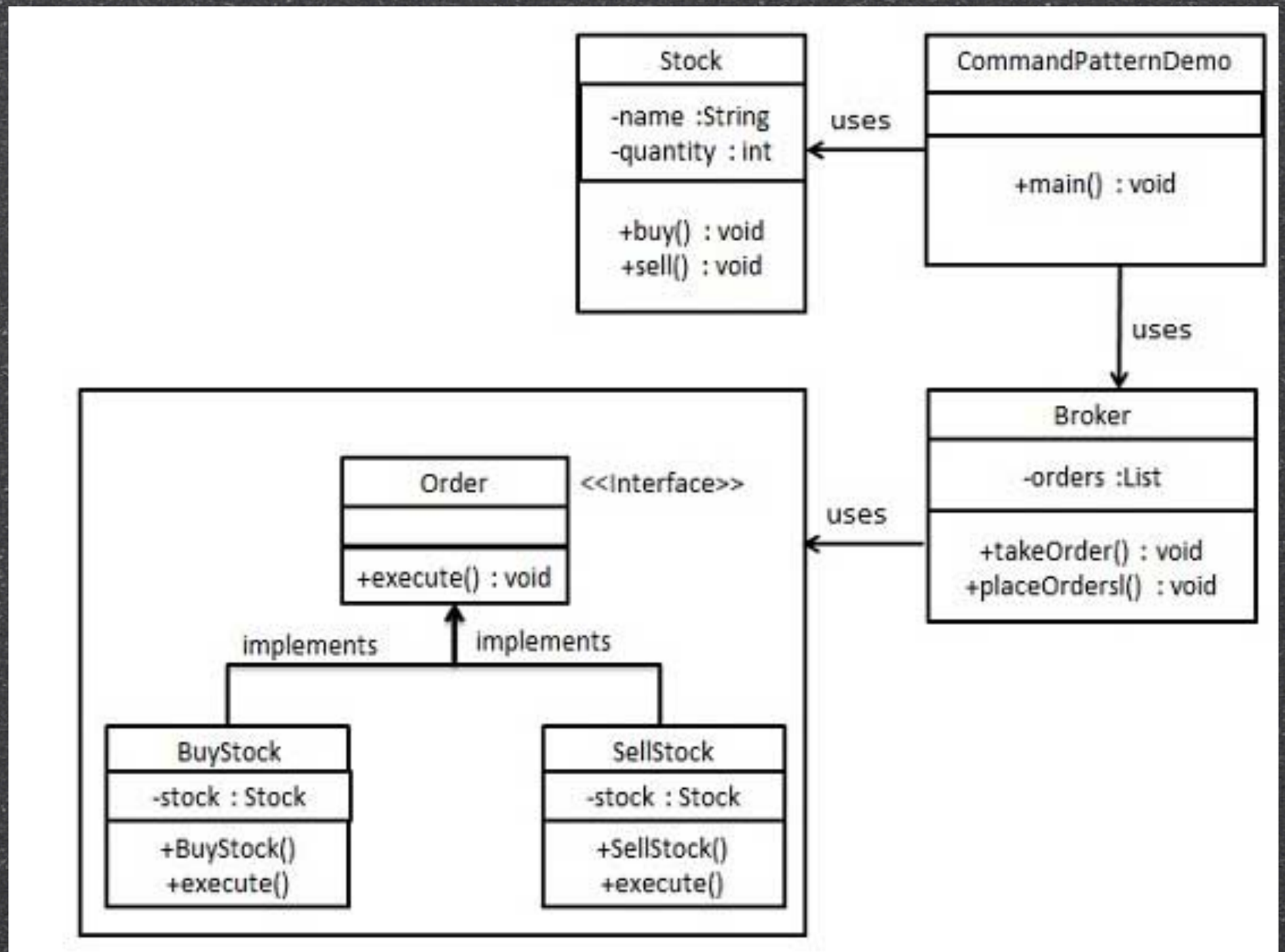
Adapter...


<<Interface>>
MediaPlayer
+play() : void
AdapterPattern Demo
+main() : void
<<Interface>>
AdvancedMediaPlayer
+playVlc() : void
+playMp4():void
implements
implements
uses
MediaAdapter
-advancedMedia Player : AdvancedMedia Player
+MediaAdapter(): void
+play() : void
AudioPlayer
-mediaAdapter : MediaAdapter
+play() : void
VlcPlayer
+playVLC() : void
+playMp4() : void
Mp4Player
+playVLC() : void
+playMp4() : void
uses
uses

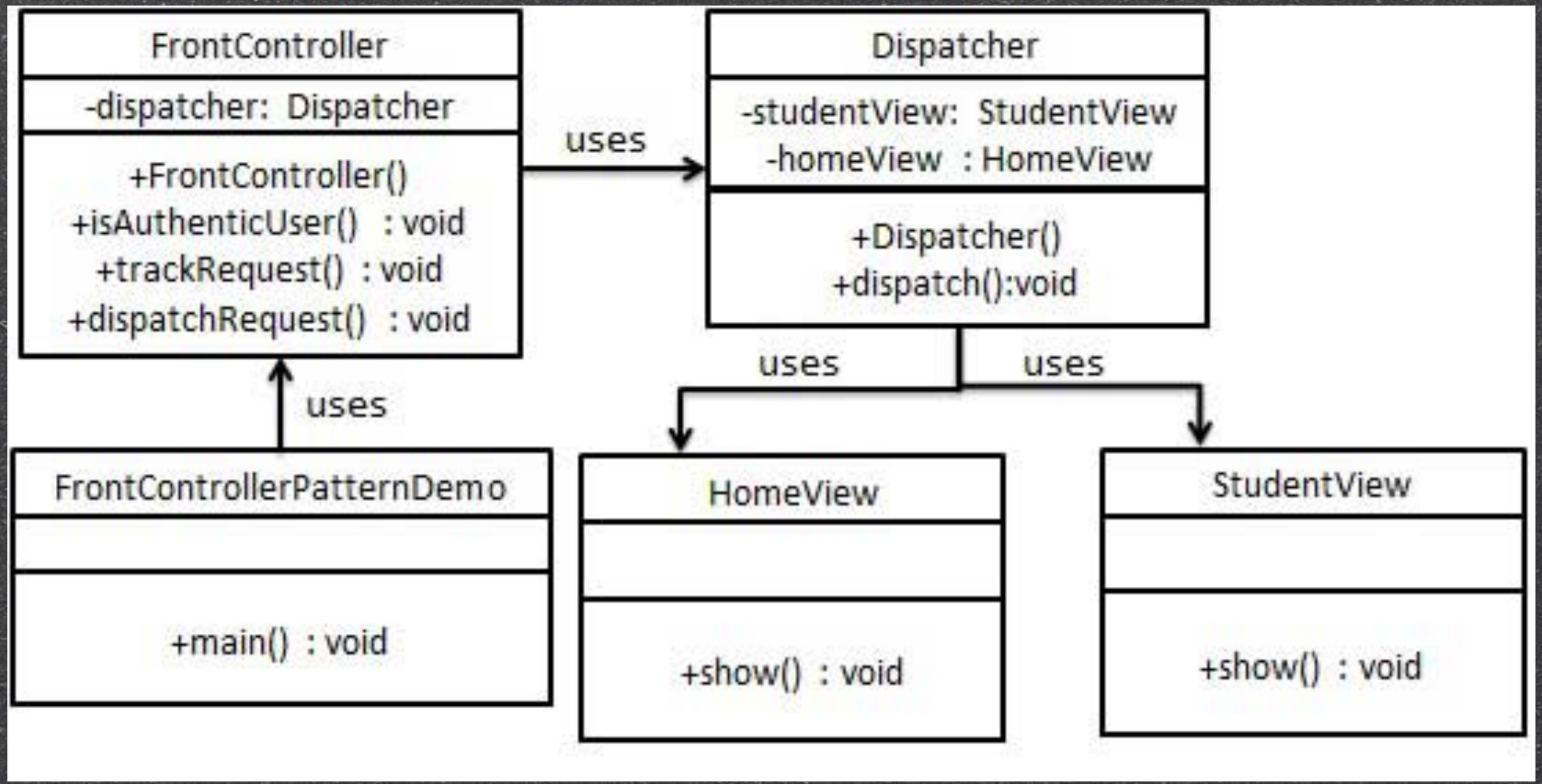
Composite


CompositePatternDemo
+main() : void
asks
Employee
-name: String
-dept : String
-salary : int
-subordinates : List <Employee>
-Employee ()
+add() : void
+remove() : void
+getSubordinates: List
<Employee>
+toString() : String
Has list of
employees

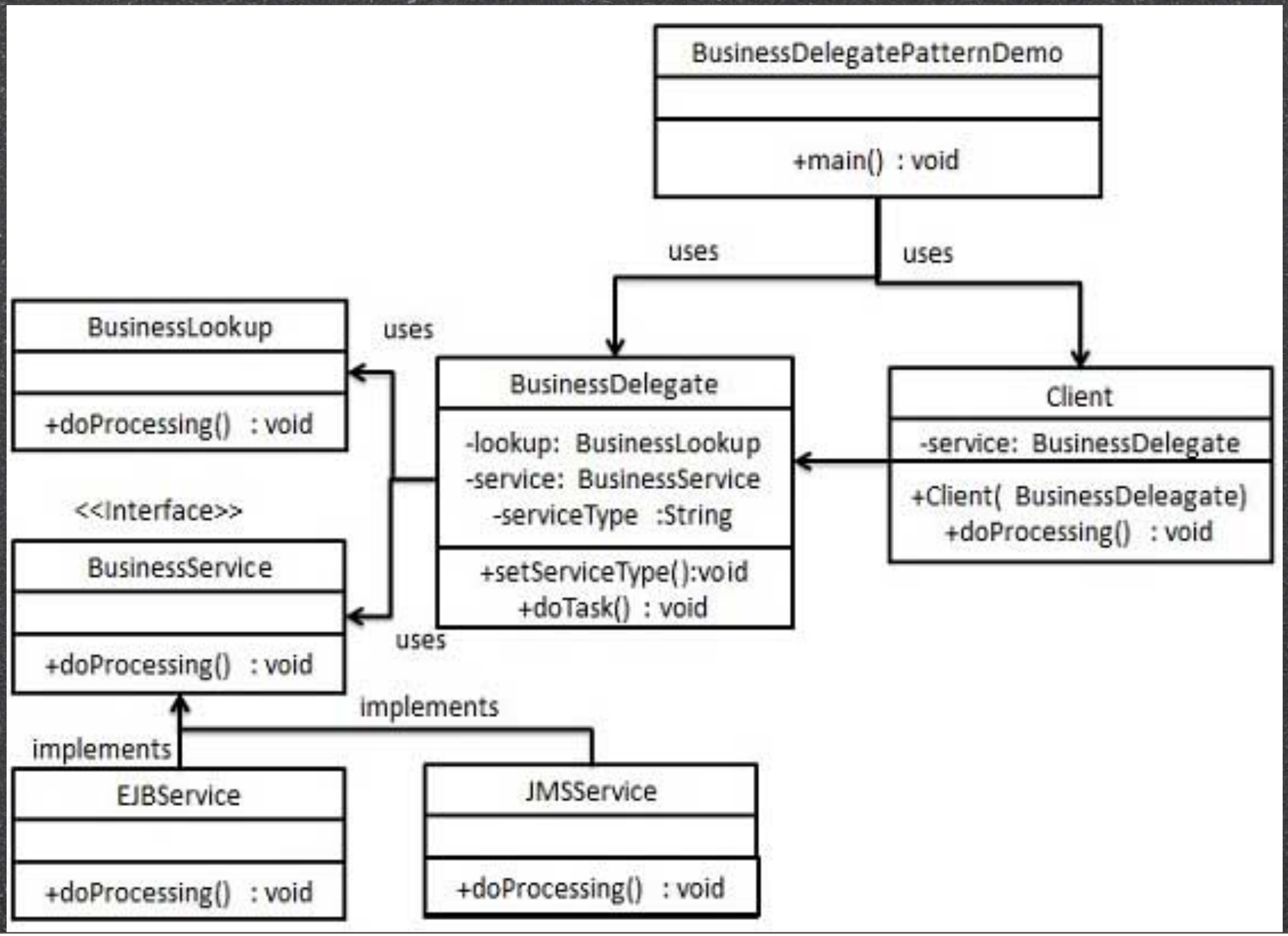
Command Pattern



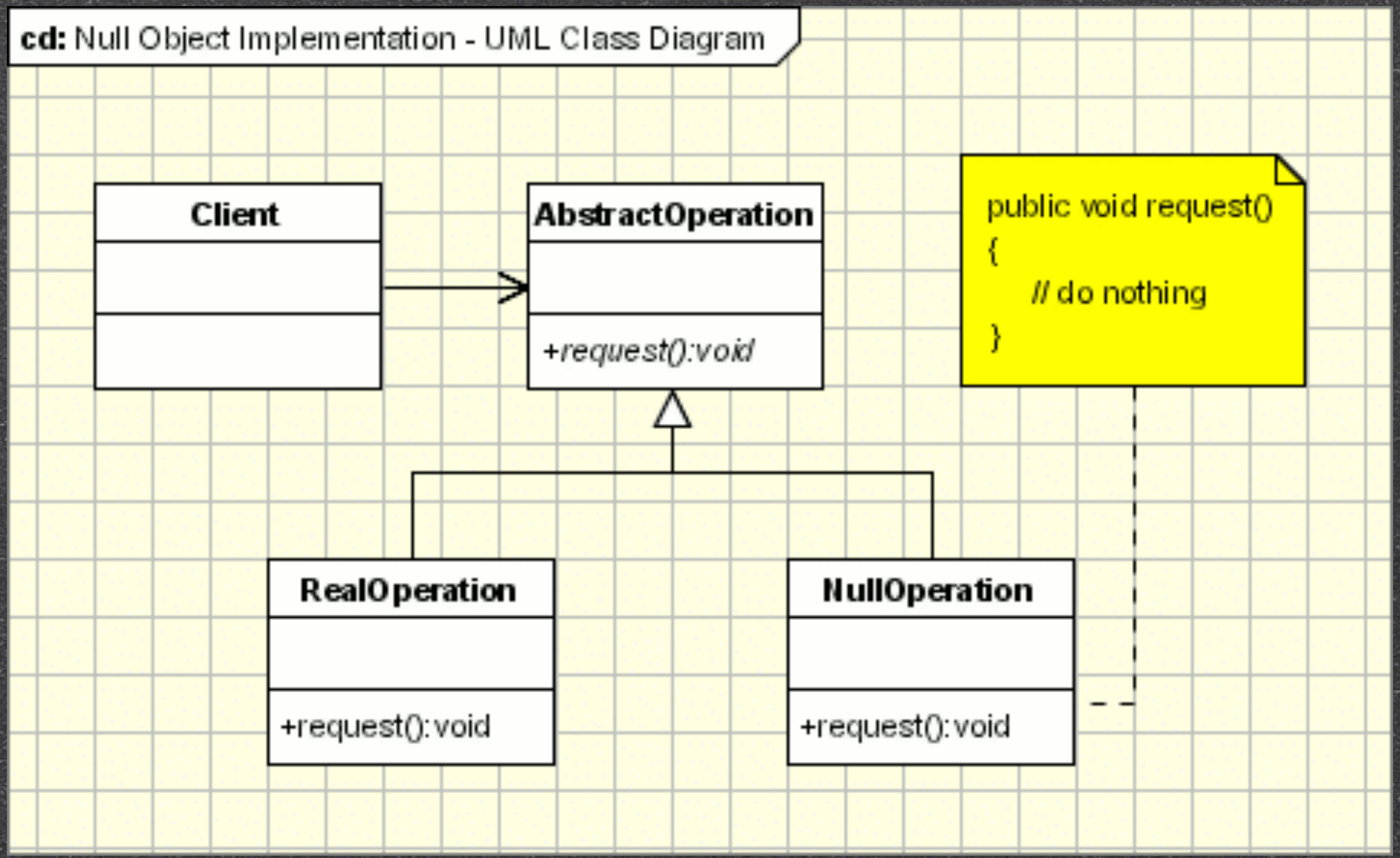
Front Controller



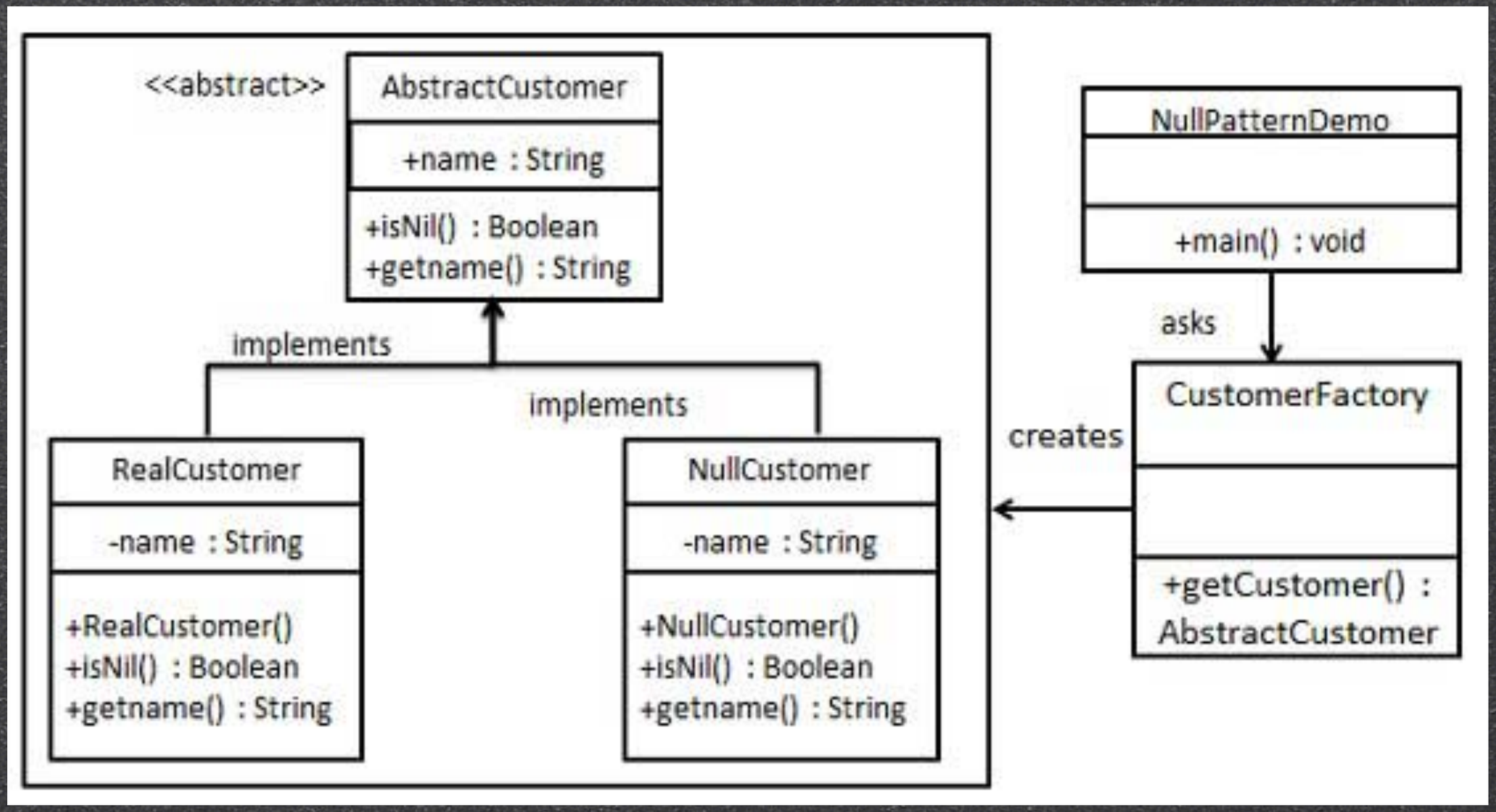
Business Delegate



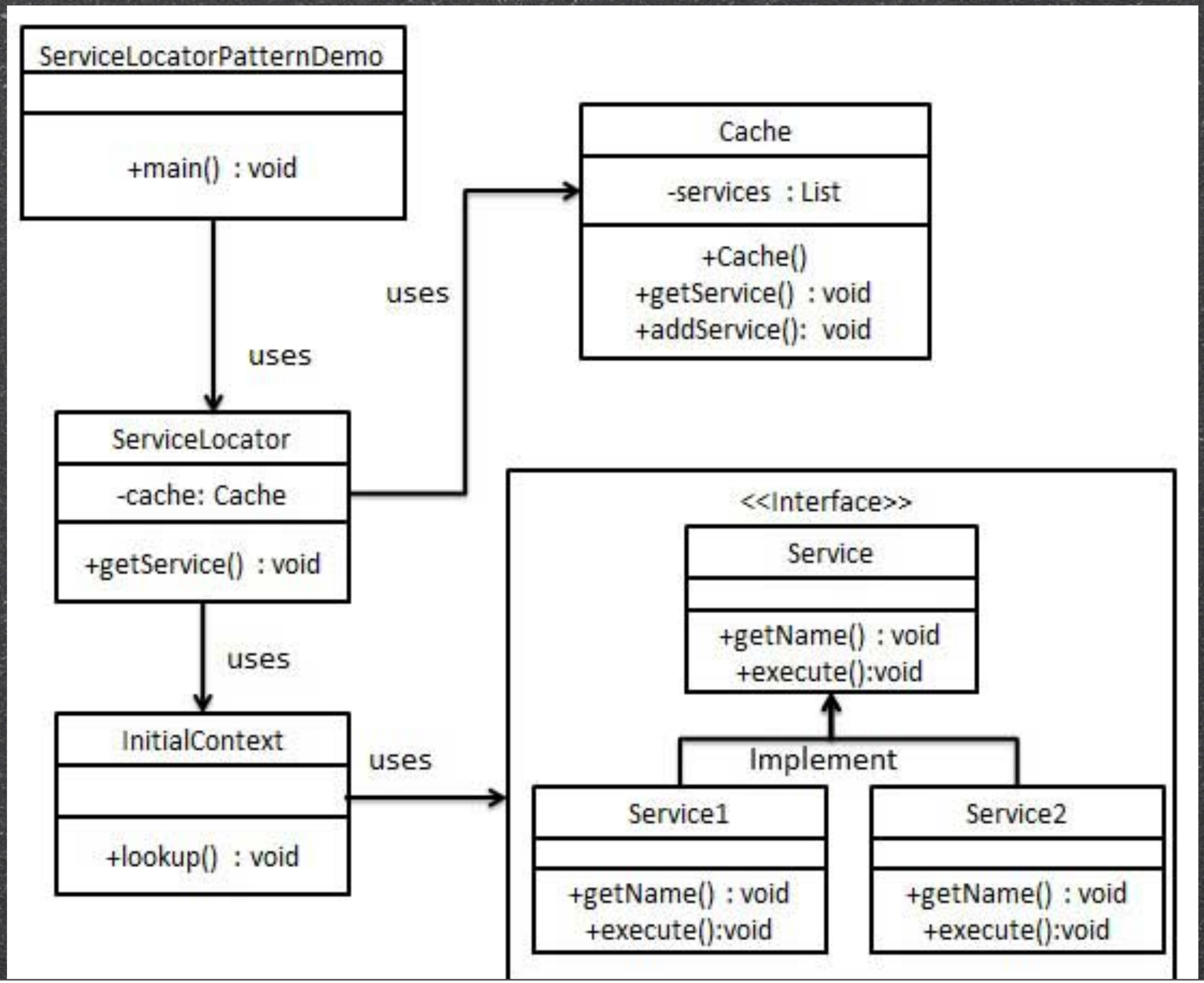
Null Object Pattern



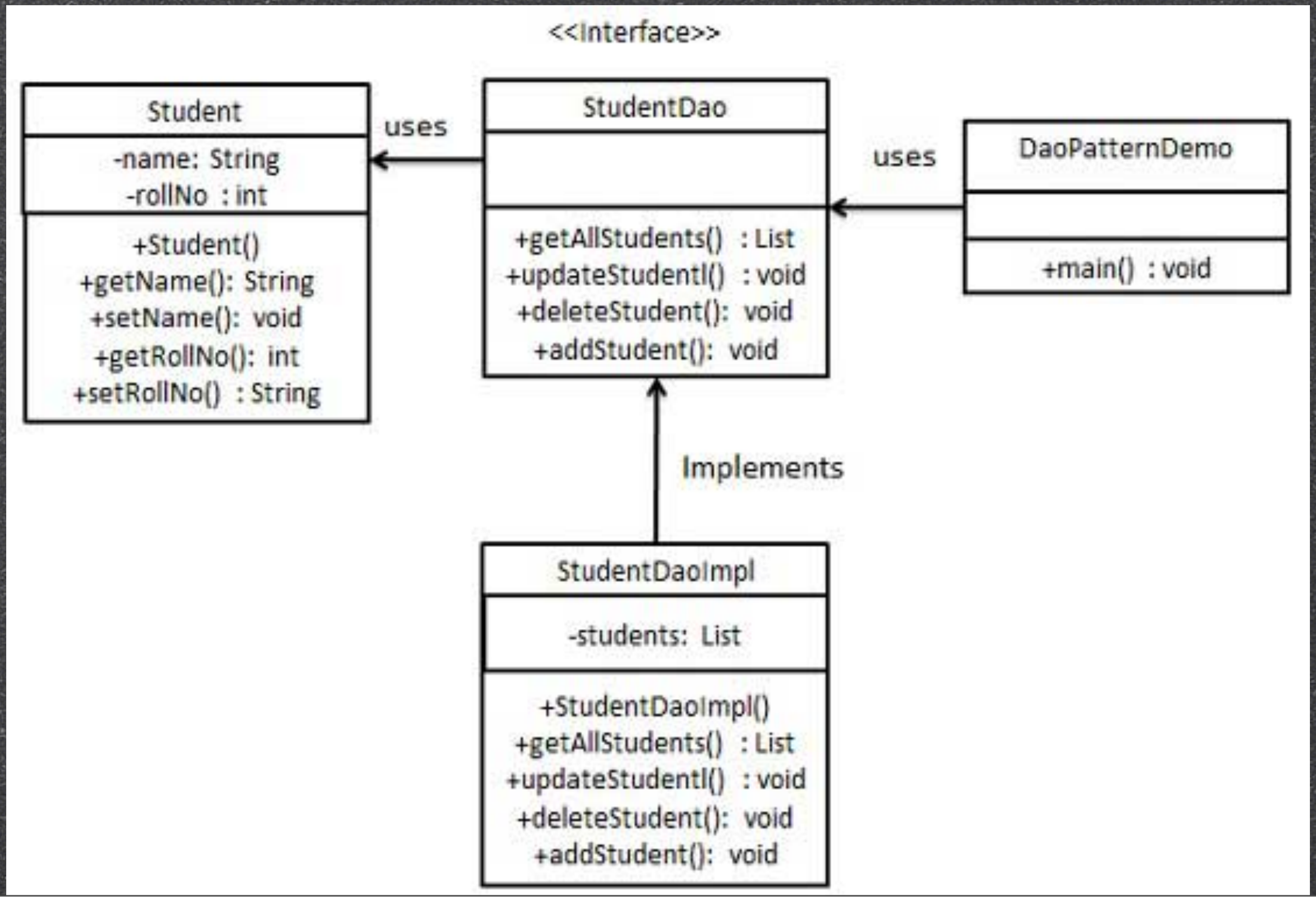
Null Object Pattern



Service Locator



DAO (Data Access Object)



EAA Patterns

- http://martinfowler.com/eaaDev/index.html

Java Ecosystem

- Development Tools
  - Eclipse/IntelliJ/NetBeans/VI/EMAC
- Deployment Tools and Build tools
  - ANT/GANT/Gradle/SBT/MAVEN/
- Servers
  - tomcat/jetty/netty/weblogic/websphere/
- Testing Tools
  - Jasmine/HTTPUnit/AB Tools/JMeter/Grinder/ JUnit/EasyMock/Mockito/JMockit/Fitness/ Cucumber/Watir/Selenium

- Web Building Tools
  - JavaScript/CSS/Markup Language/Browsers
- Debugging Tools
  - IDE/FireBug/Chrome Inspect/ IE Inspect
- Logging
  - Logback/Log4J
- Tracking tools
  - VisualVM/JStack/

Java Support for FP...

- Need for Functional Programming
- Why do you thing after so many years people are going crazy about FP
- Is OOP Bad and if so what is bad in OO and the reason it’s bad...
- What is current trend people should follow?

Java 8 /google fp lib..

- What is functional Programming?
- Referential transparent
- No side effect
- no mutability
- ..... lets see some code example

EOF

@mmahawar