DESIGN PRINCIPLES AI

06. DESIGN PAT

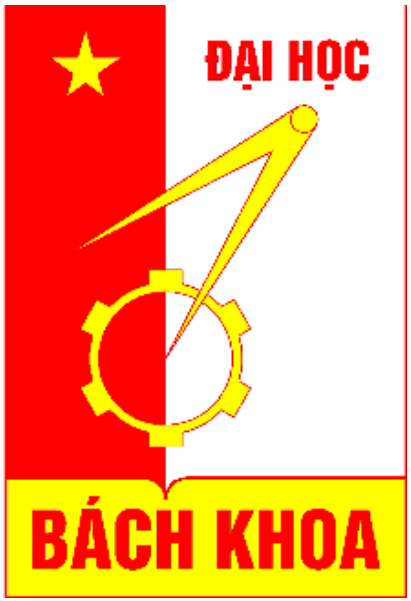
D PATTERNS

TERN 2

Nguyen Thi Thu Tran
trangntt@soict.hust.edu

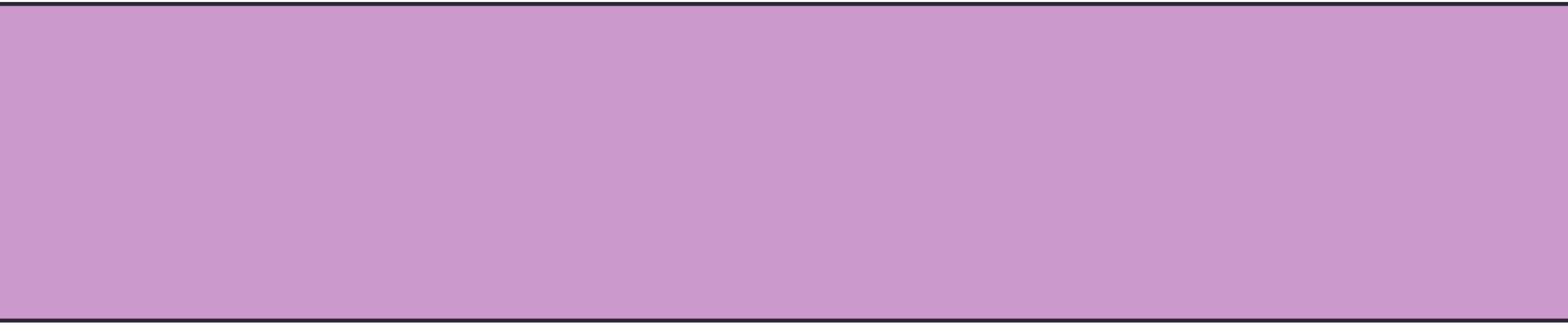
g

vn



Content

1. Strategy
2. Observer



3. Adapter

4. State

Example 1: Street

A character has four
roll and jump

- Mandatory: kick and p

Fighter Game

noves: kick, punch,

nch moves

Optional: roll and jump

How would you mode

- Suppose initially you u
  abstract out the comm
  class and let other cha
  class

l your classes?

se inheritance and

on features in a Fighter

racters subclass Fighter

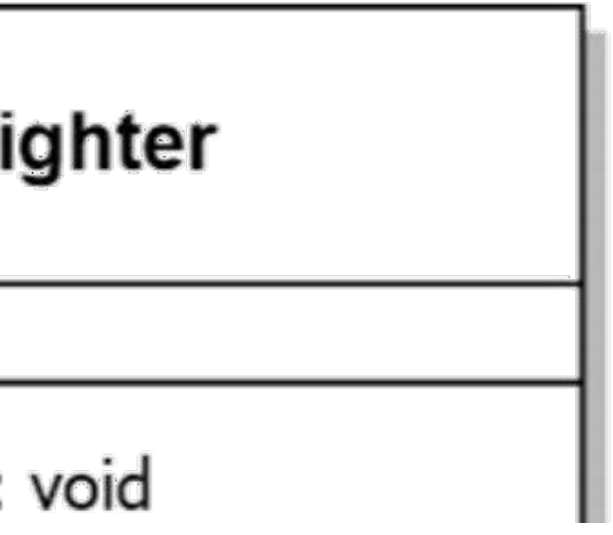
Street Fighter Gam

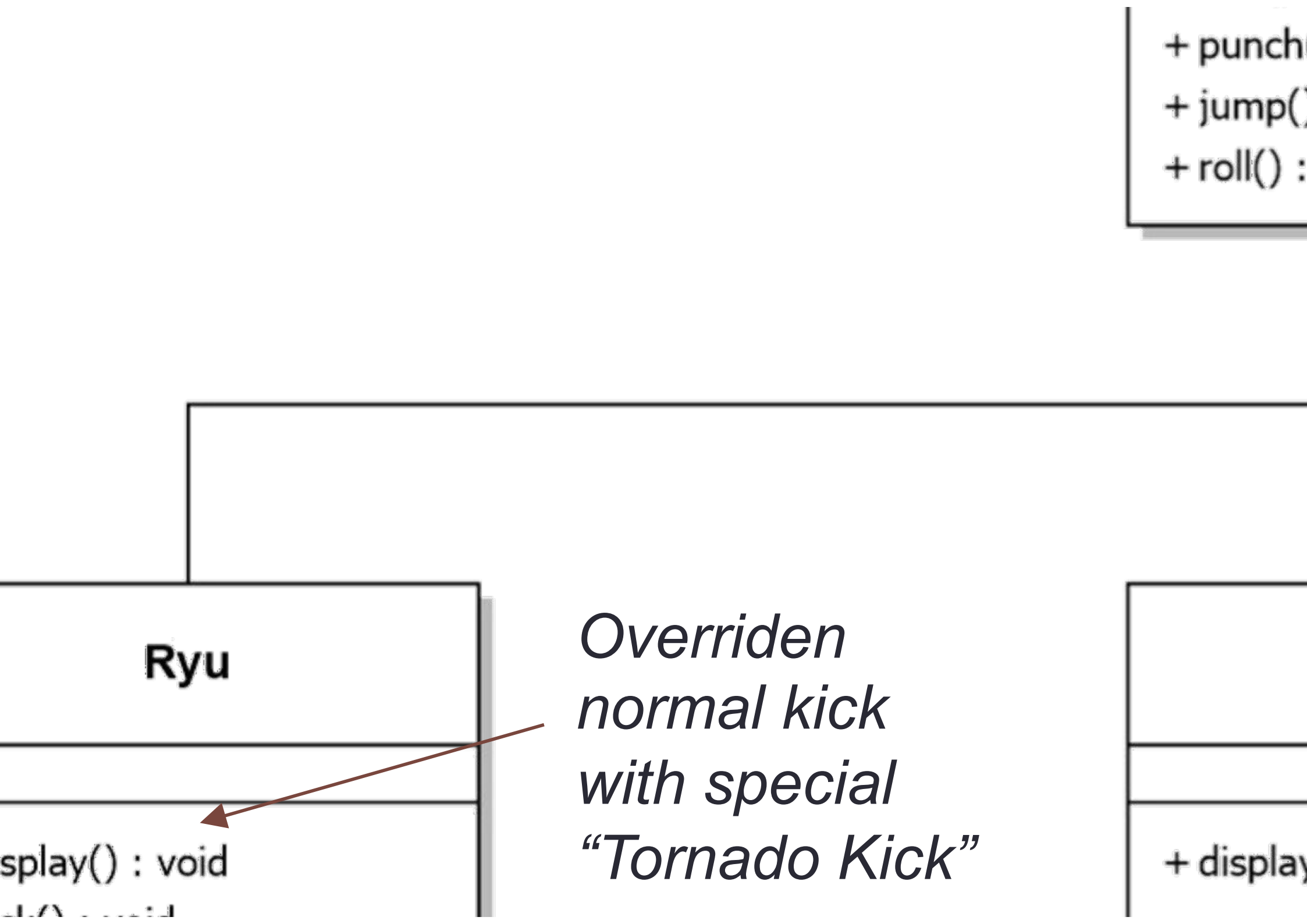
Any character with spec
that action in its subclas

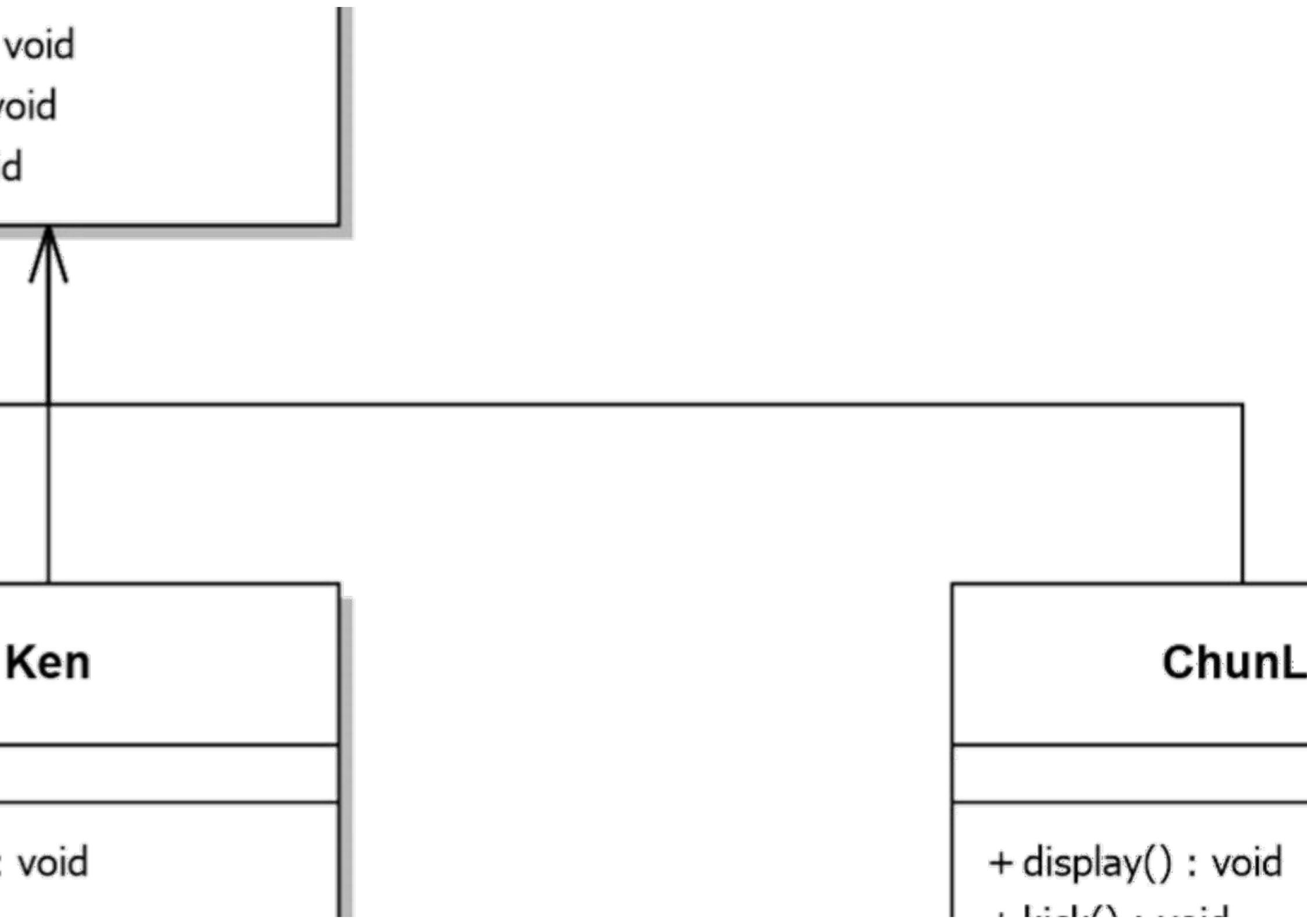
e: Subclass

alized move can override

s




+ punch
+ jump(
+ roll() :
Ryu
splay() : void
+ display
Overriden
normal kick
with special
“Tornado Kick”


void
oid
d
Ken
: void
ChunL
+ display() : void

Street Fighter Gam

What if a character does

- It still inherits the jump bel
- Although you can override
  case but you may have to

e: Subclass (2)

n't perform jump move?

avior from super class

jump to do nothing in that

do so for many existing

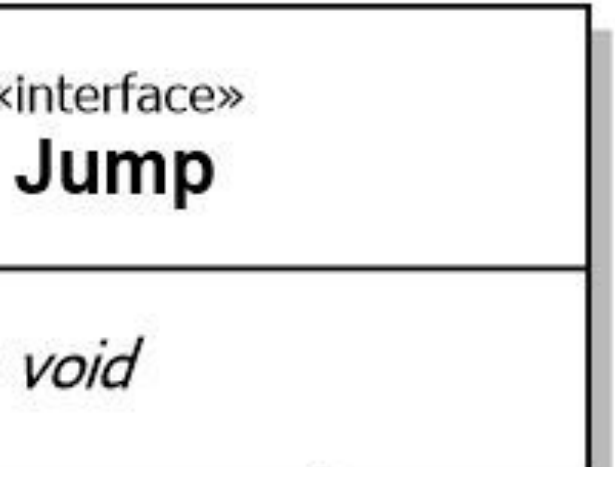
classes and take care of t

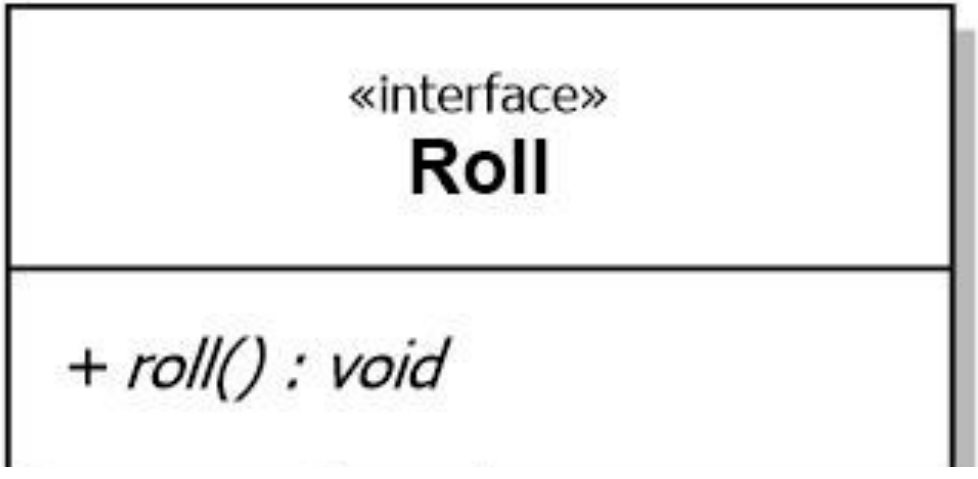
- Difficult to maintain
- Violate LSP

at for future classes too

Street Fighter Game: Su

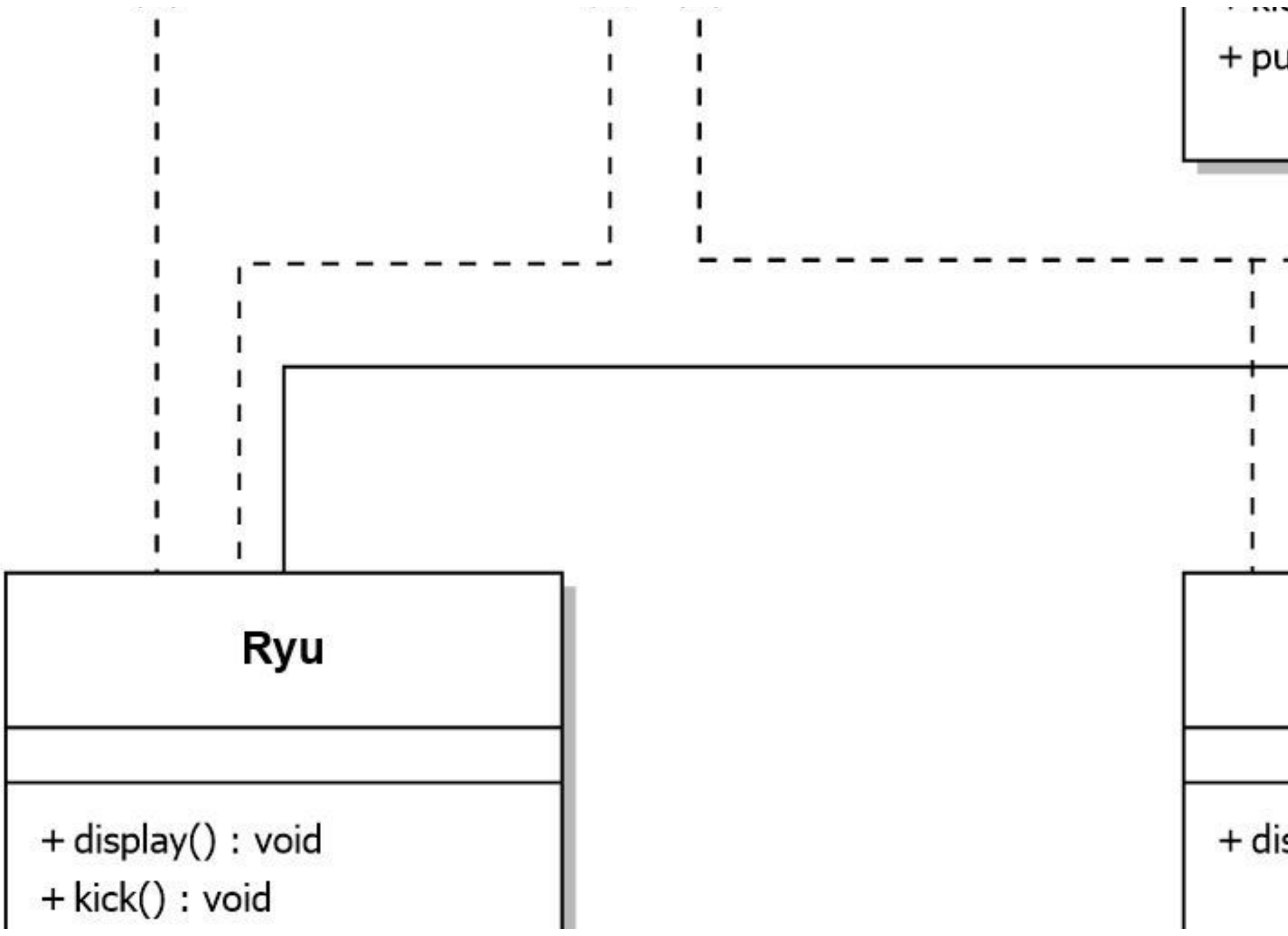
Put optional behaviors t

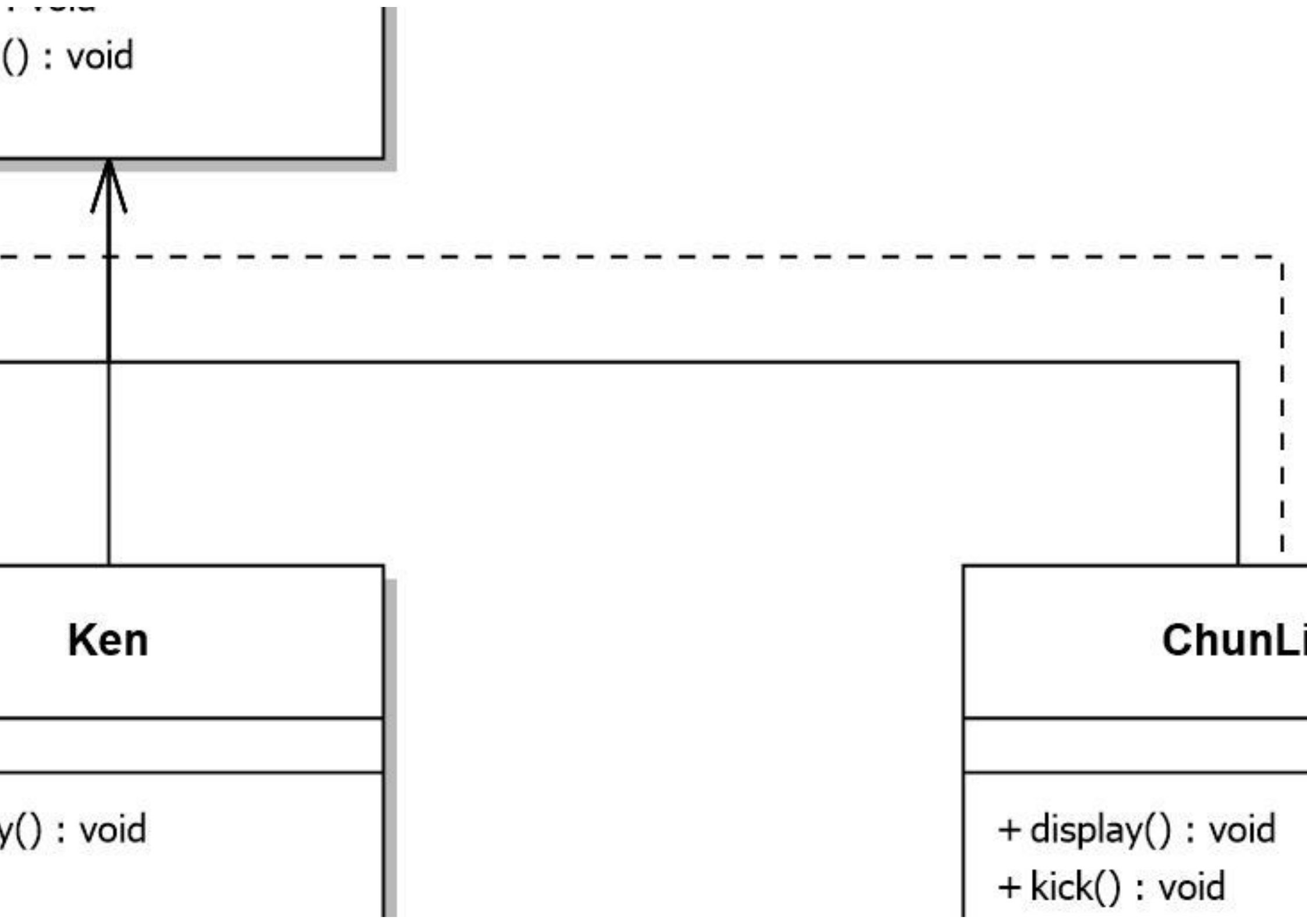




bclass and Interface

interfaces


+ pu
Ryu
+ display() : void
+ kick() : void
+ dis


() : void
Ken
y() : void
+ display() : void
+ kick() : void

Street Fighter Game: Su

What if jump and/or roll
for some or all characte

- Bad code reusability: Dup
- Multiple inheritance?

bclass and Interface (2

behaviors are the same

s?

cation code

- Inheritance duplication, diam
- Not support in many languag

What if the player evolve this character to another

- Nont possible with Inherita

nd problem

s

s/wants to change from

one when PLAYING?

nce

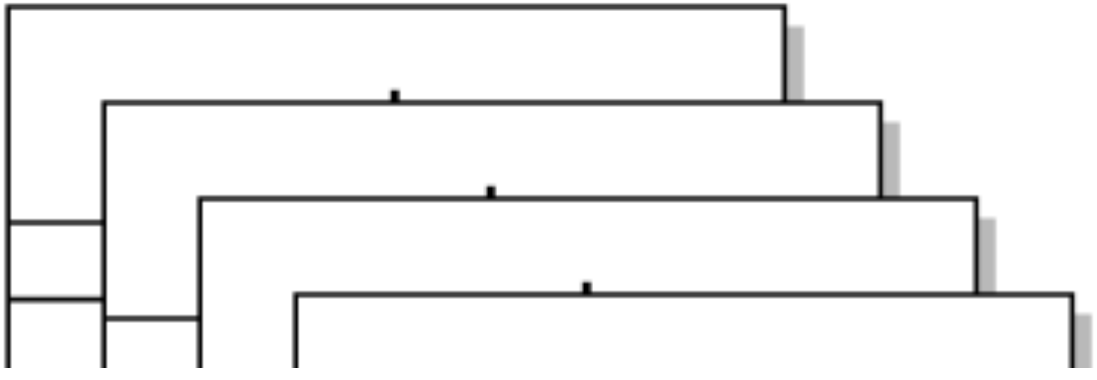
Street Fighter Game

Put behaviors that may var
future and separate them fr

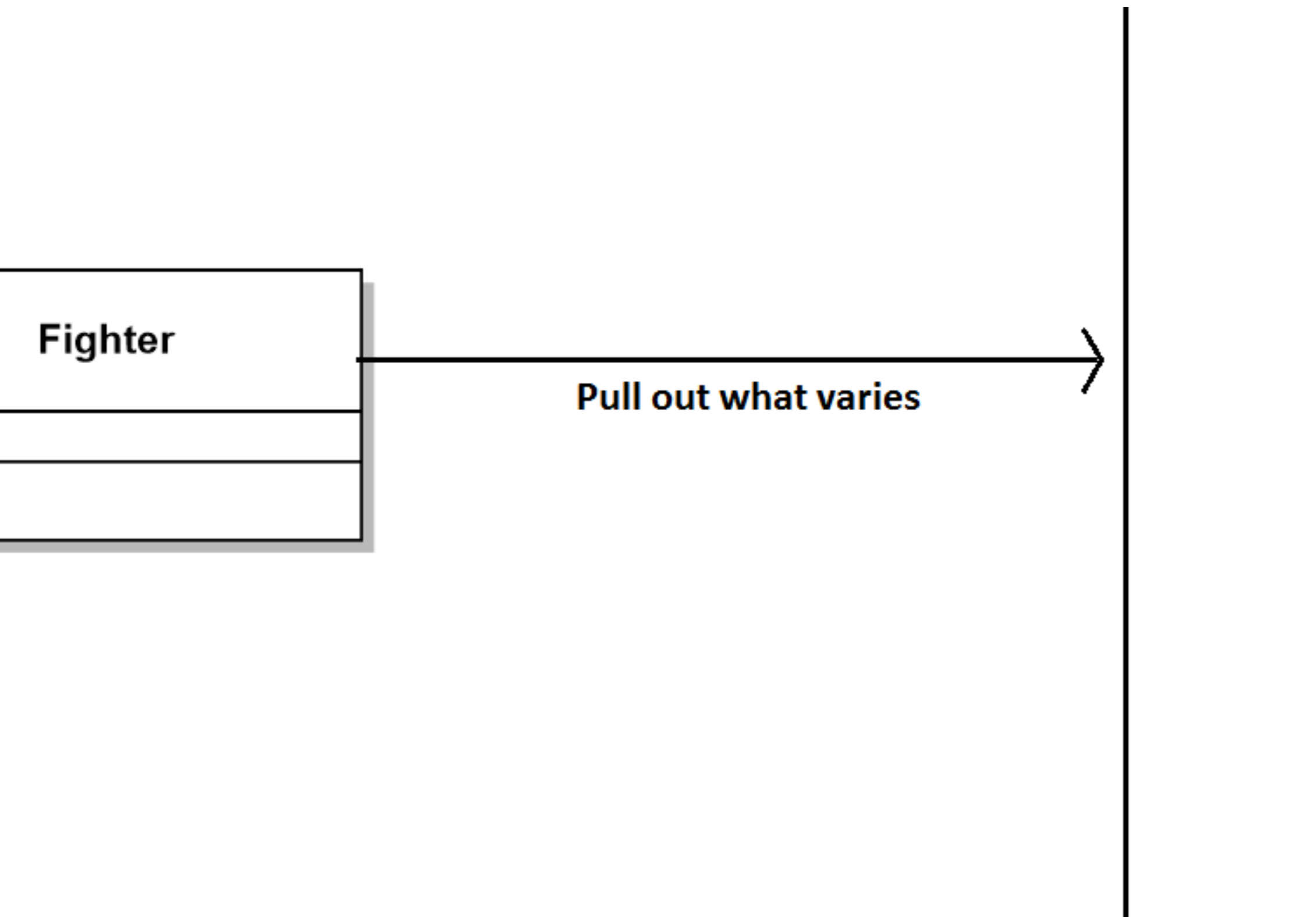
- i.e. jump and roll behaviors

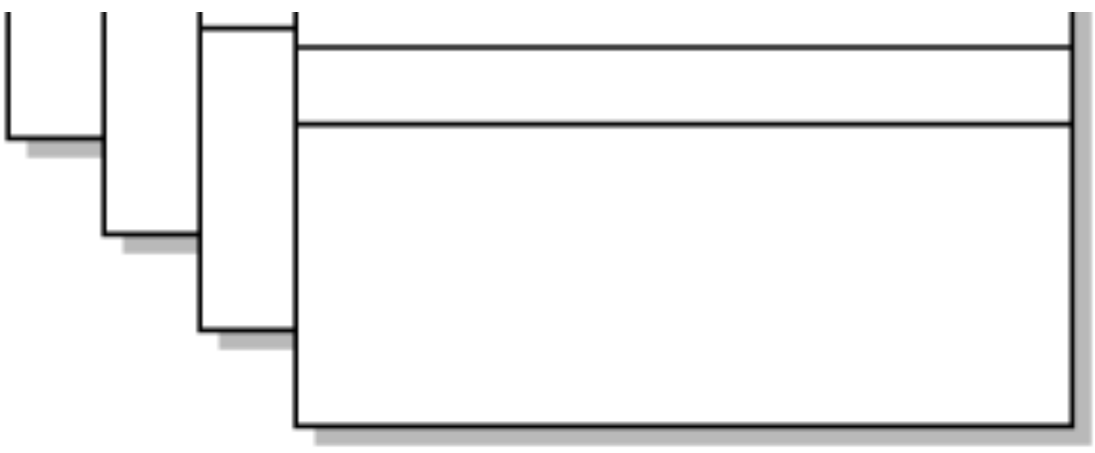
Strategy Pattern

across different classes in
m the rest

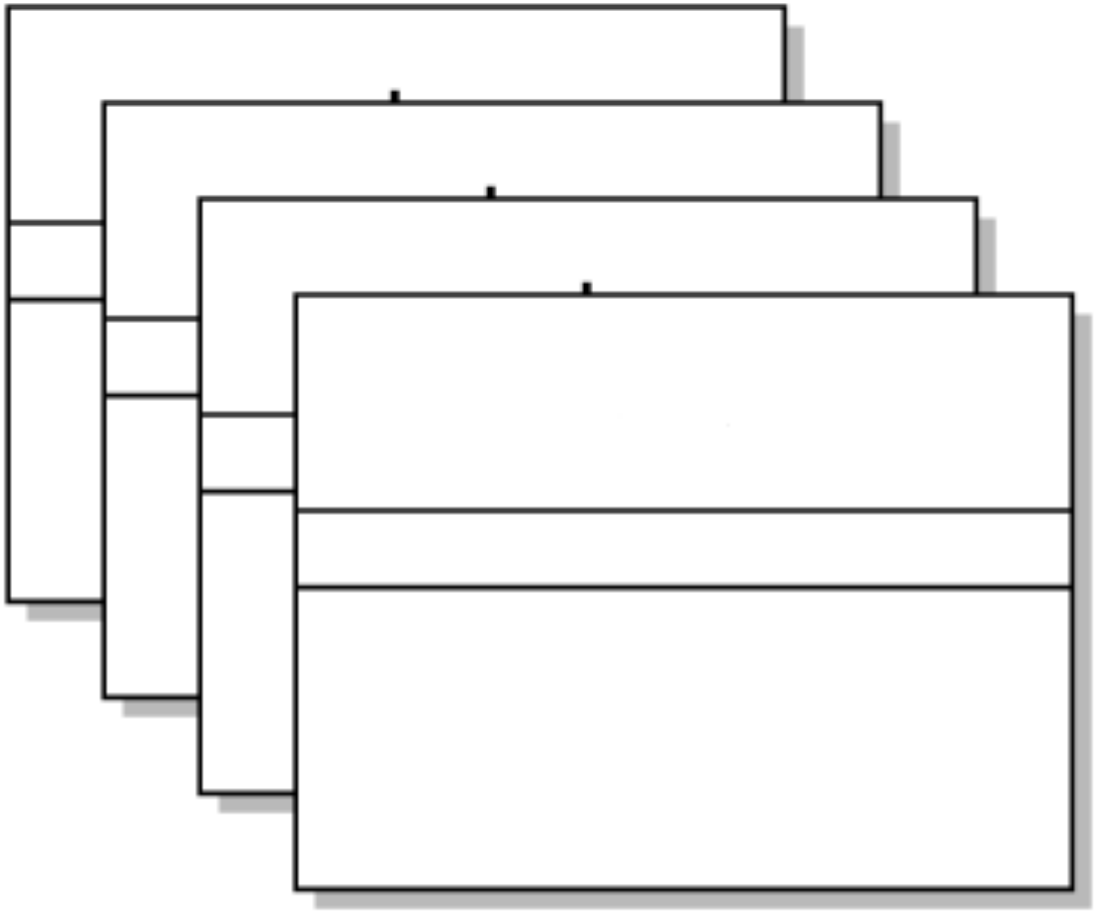


Fighter Behaviors


Fighter
Pull out what varies



Roll Behaviors



Street Fighter Game

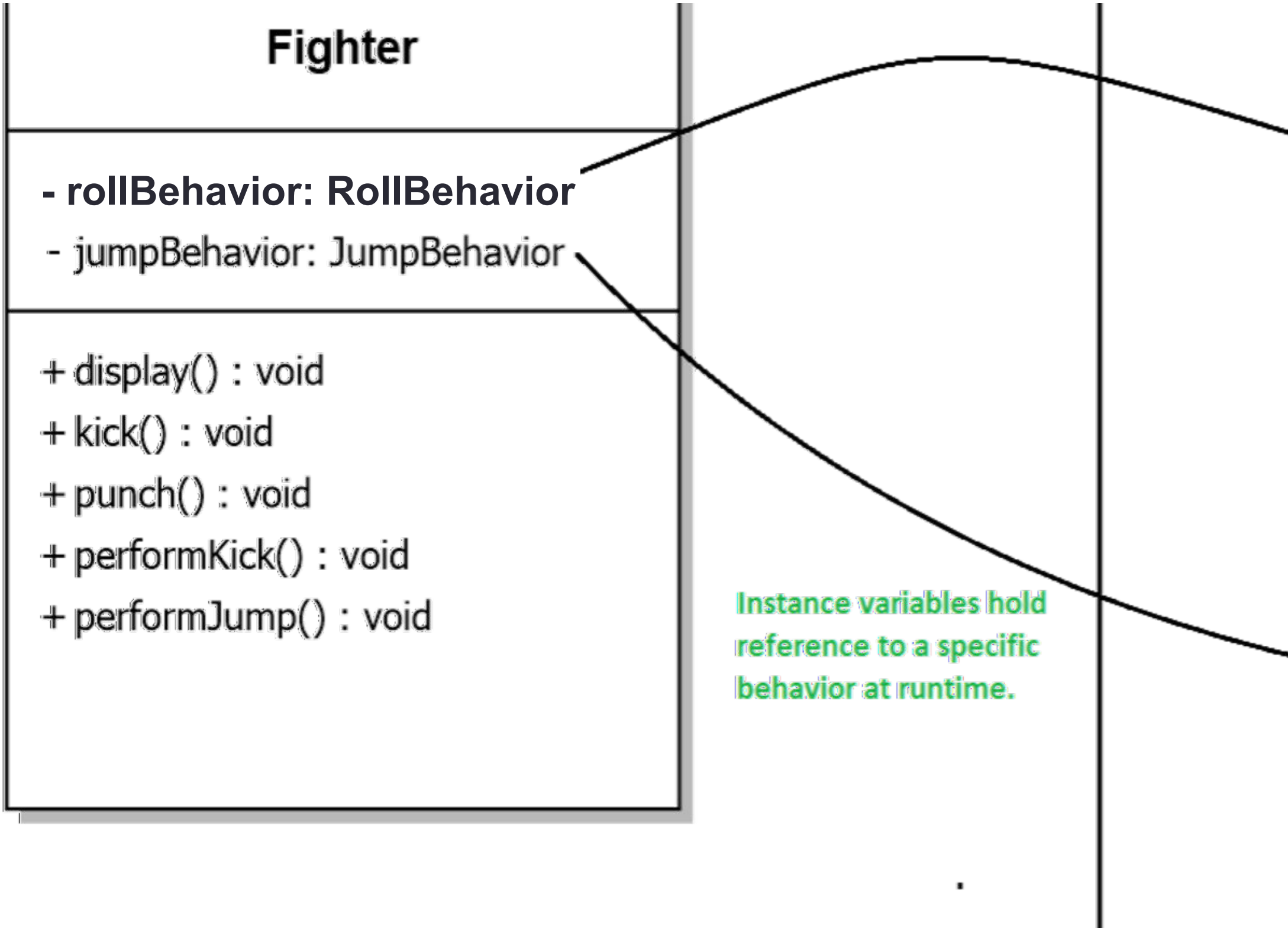
Fighter delegates its jum

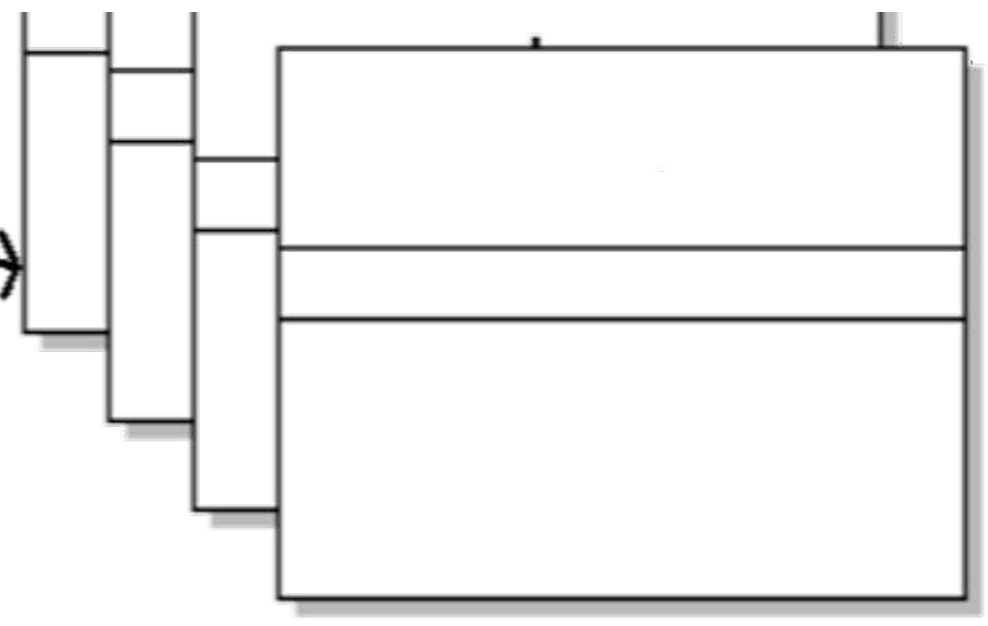
- Instead of using jump and
  Fighter or its subclass

Strategy Pattern (2)

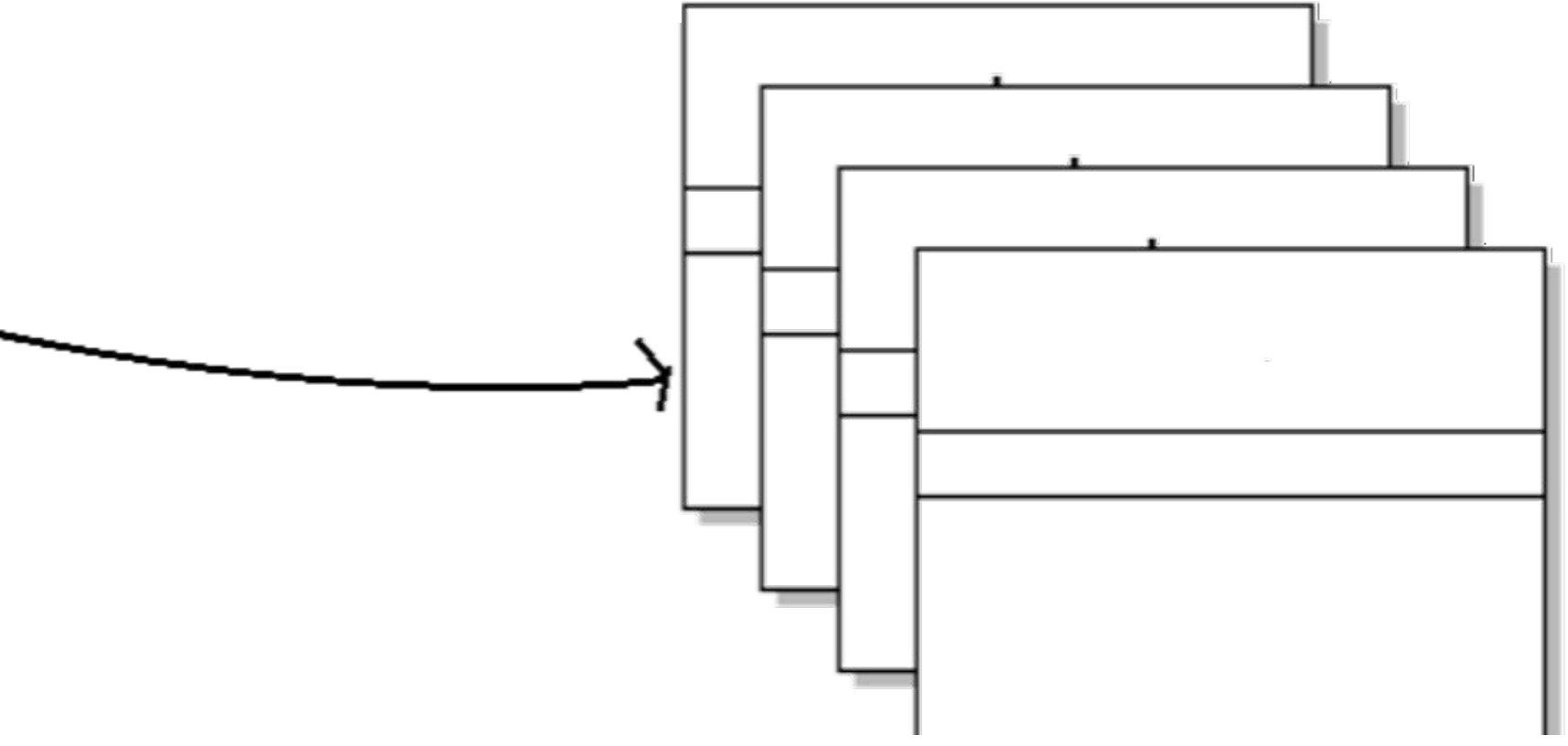
p and roll behaviors

roll methods defined in


Fighter
- rollBehavior: RollBehavior
- jumpBehavior: JumpBehavior
+ display() : void
+ kick() : void
+ punch() : void
+ performKick() : void
+ performJump() : void
Instance variables hold reference to a specific behavior at runtime.

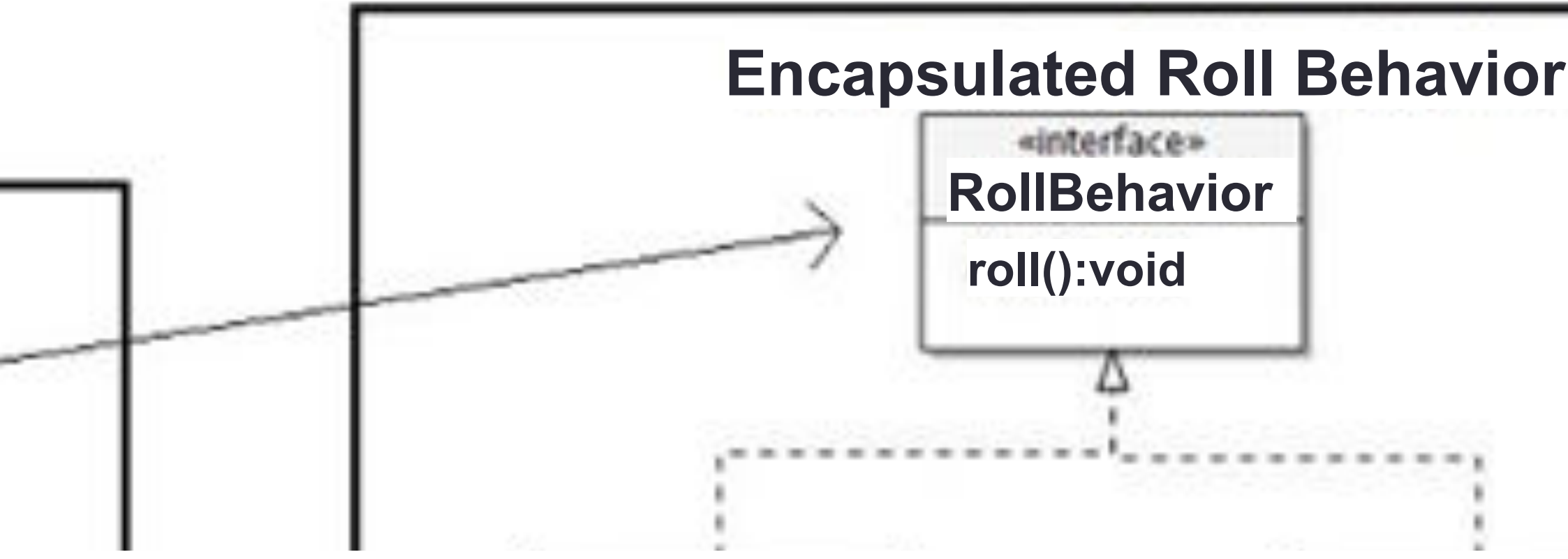


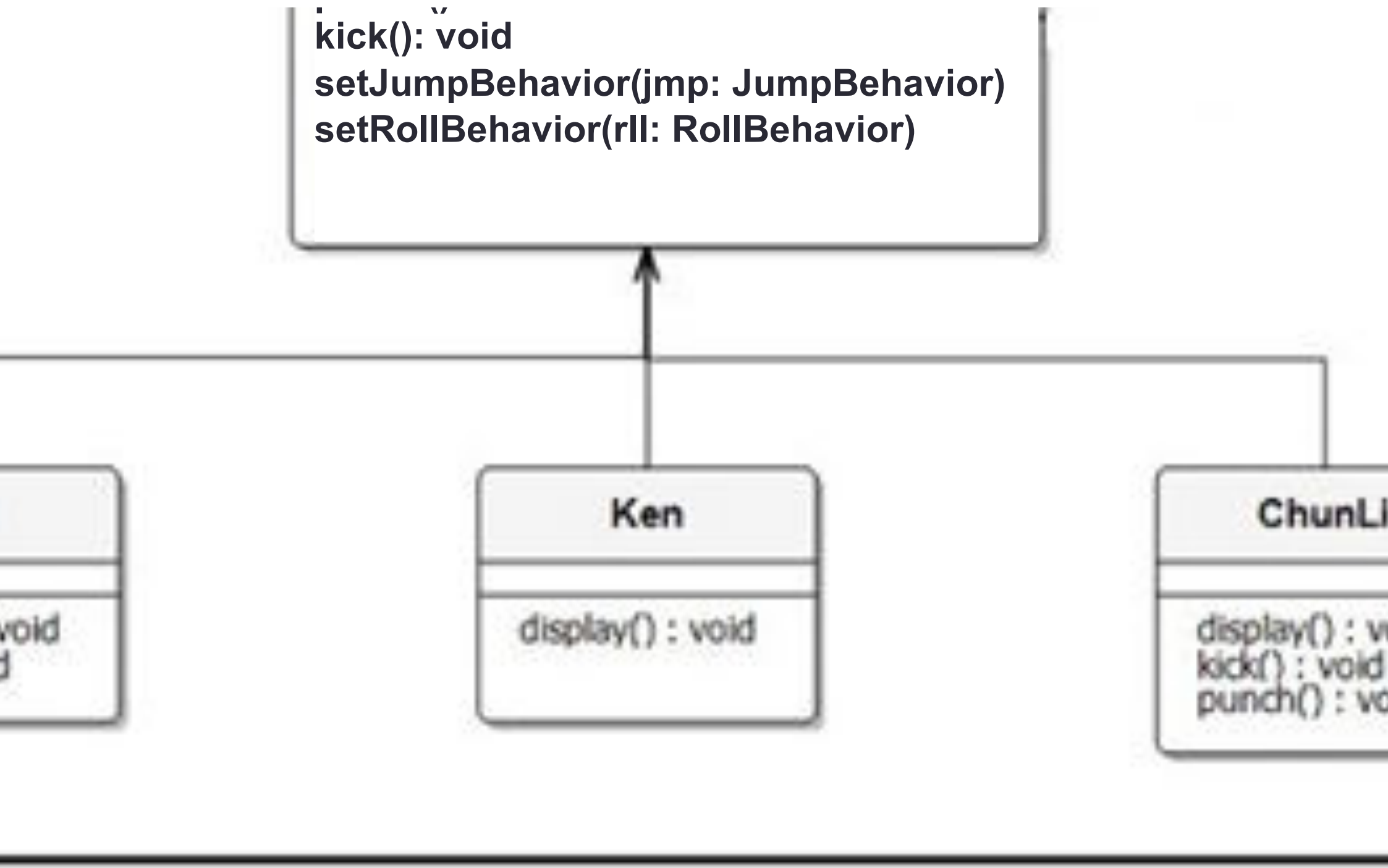
Roll Behaviors

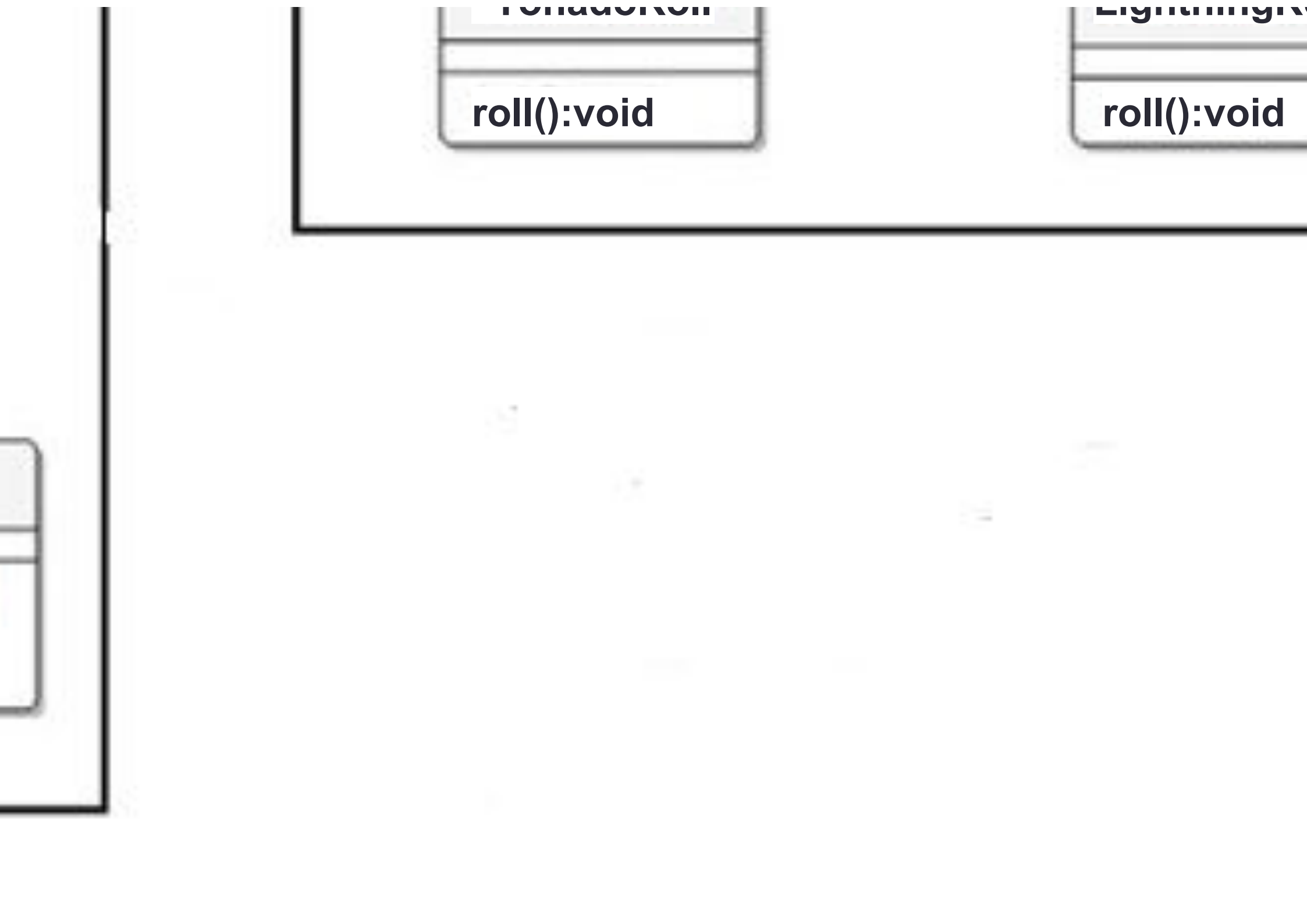


Street Fighter Game

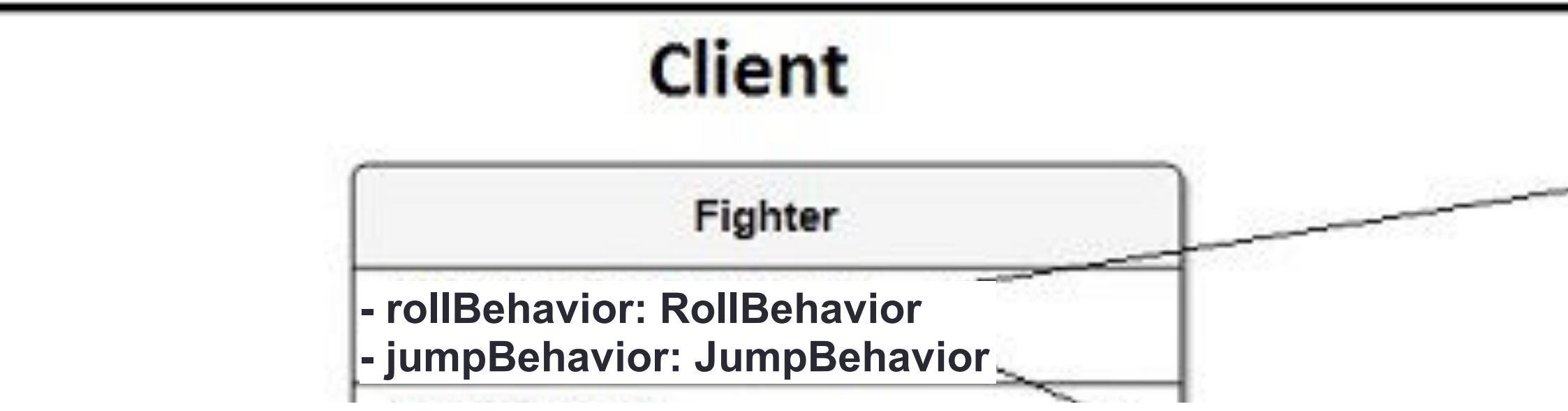
Strategy Pattern (3)



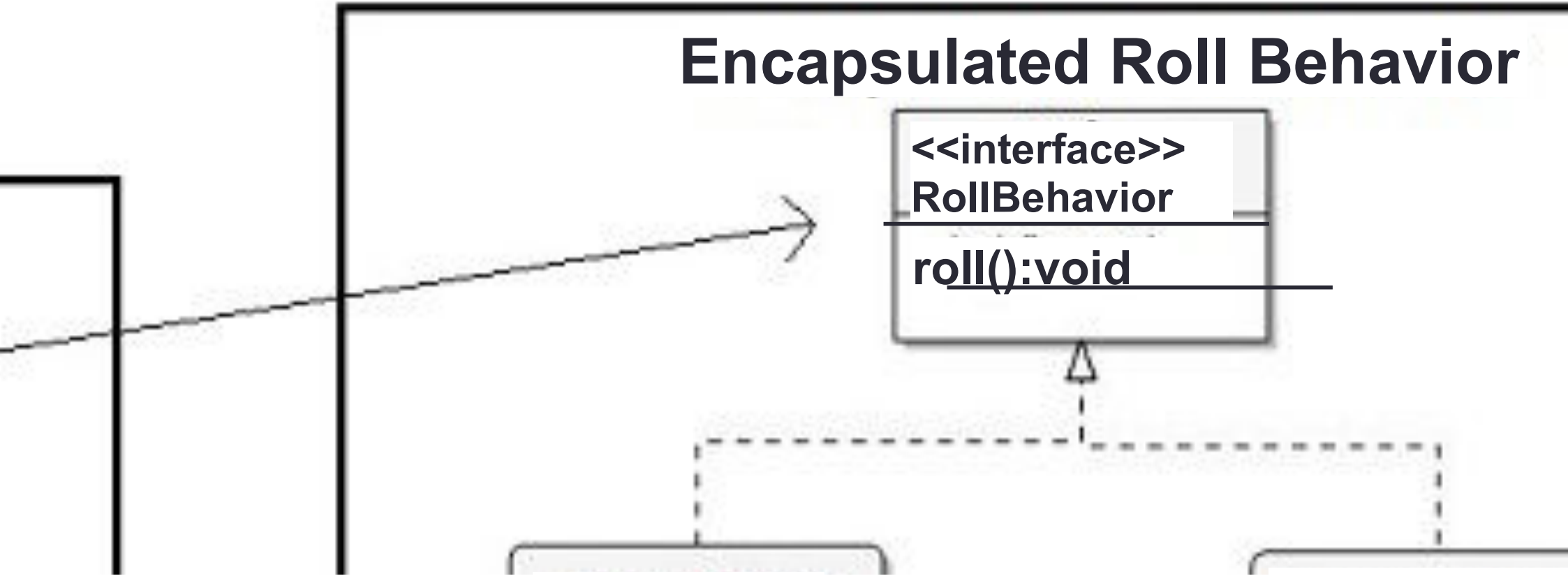

kick(): void
setJumpBehavior(jmp: JumpBehavior)
setRollBehavior(rll: RollBehavior)
Ken
display() : void
ChunLi
display() : v
kick() : void
punch() : vo

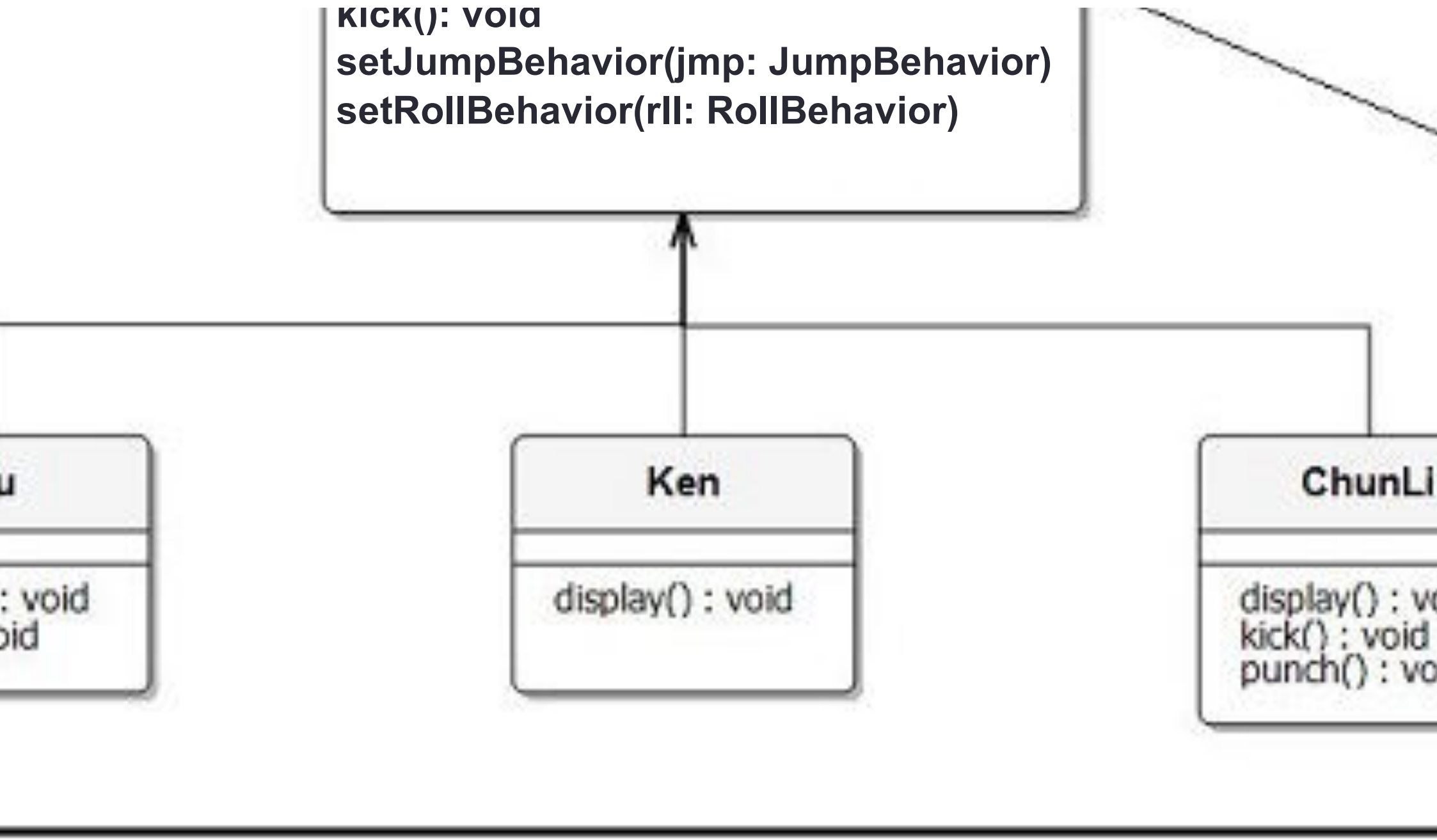

roll():void
roll():void

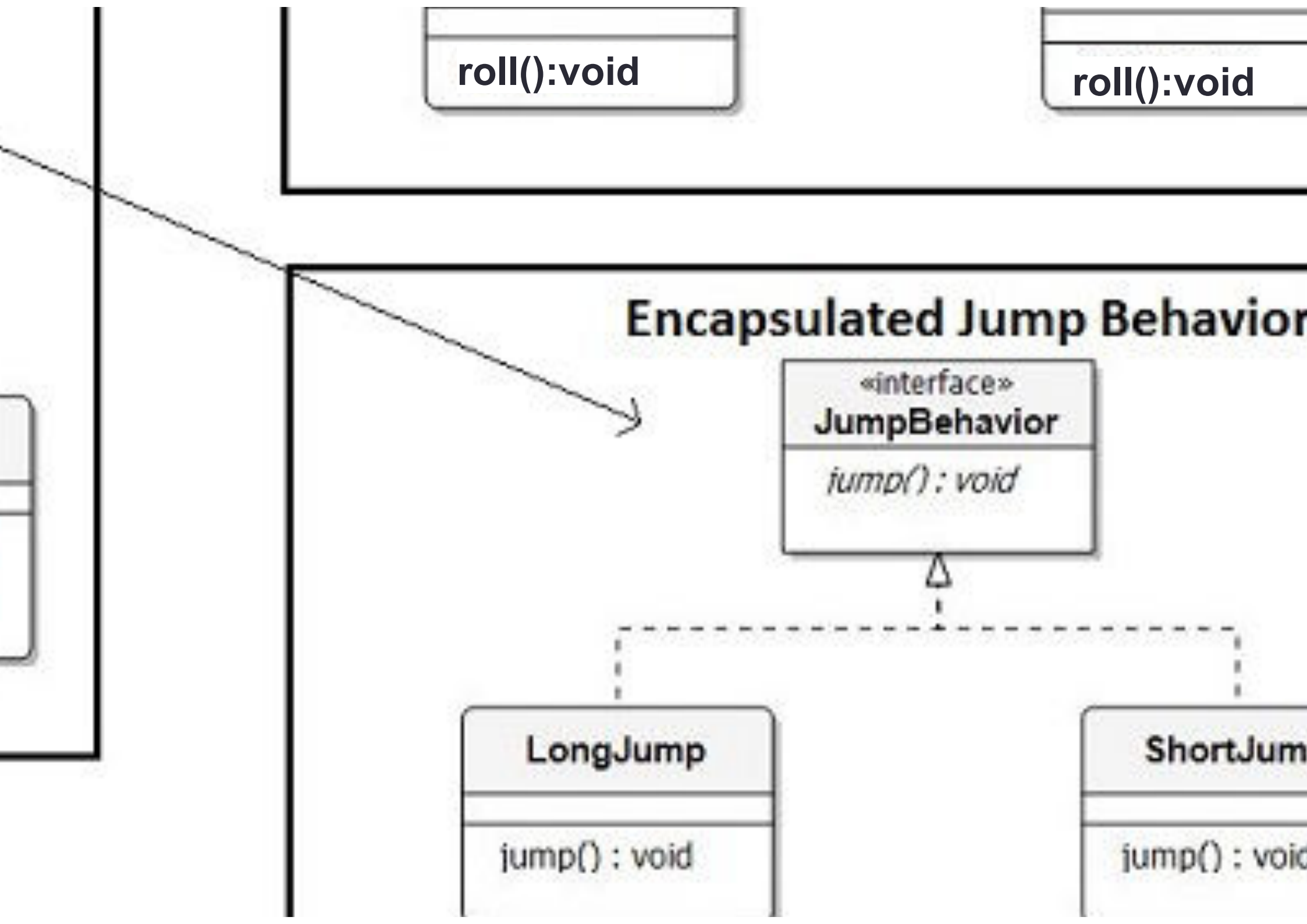
Street Fighter Game



Strategy Pattern (4)




kick(): void
setJumpBehavior(jmp: JumpBehavior)
setRollBehavior(rll: RollBehavior)
Ken
display() : void
ChunLi
display() : vo
kick() : void
punch() : vo


roll():void
roll():void
Encapsulated Jump Behavior
«interface»
JumpBehavior
jump() : void
LongJump
jump() : void
ShortJum
jump() : voi

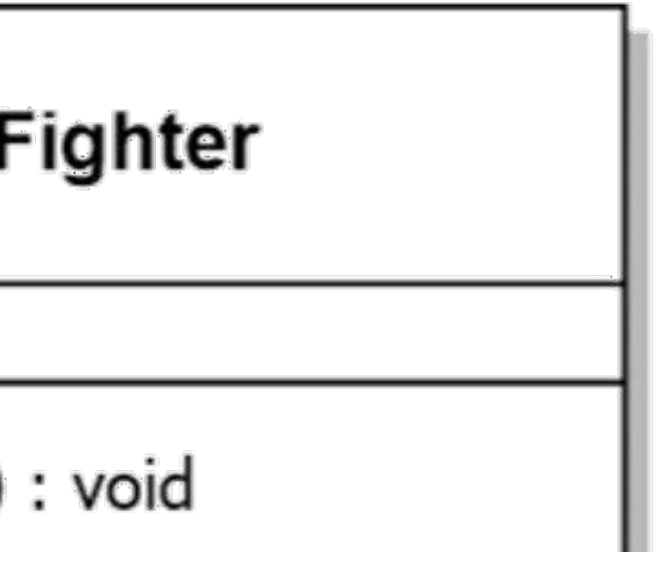
Street Fighter Game

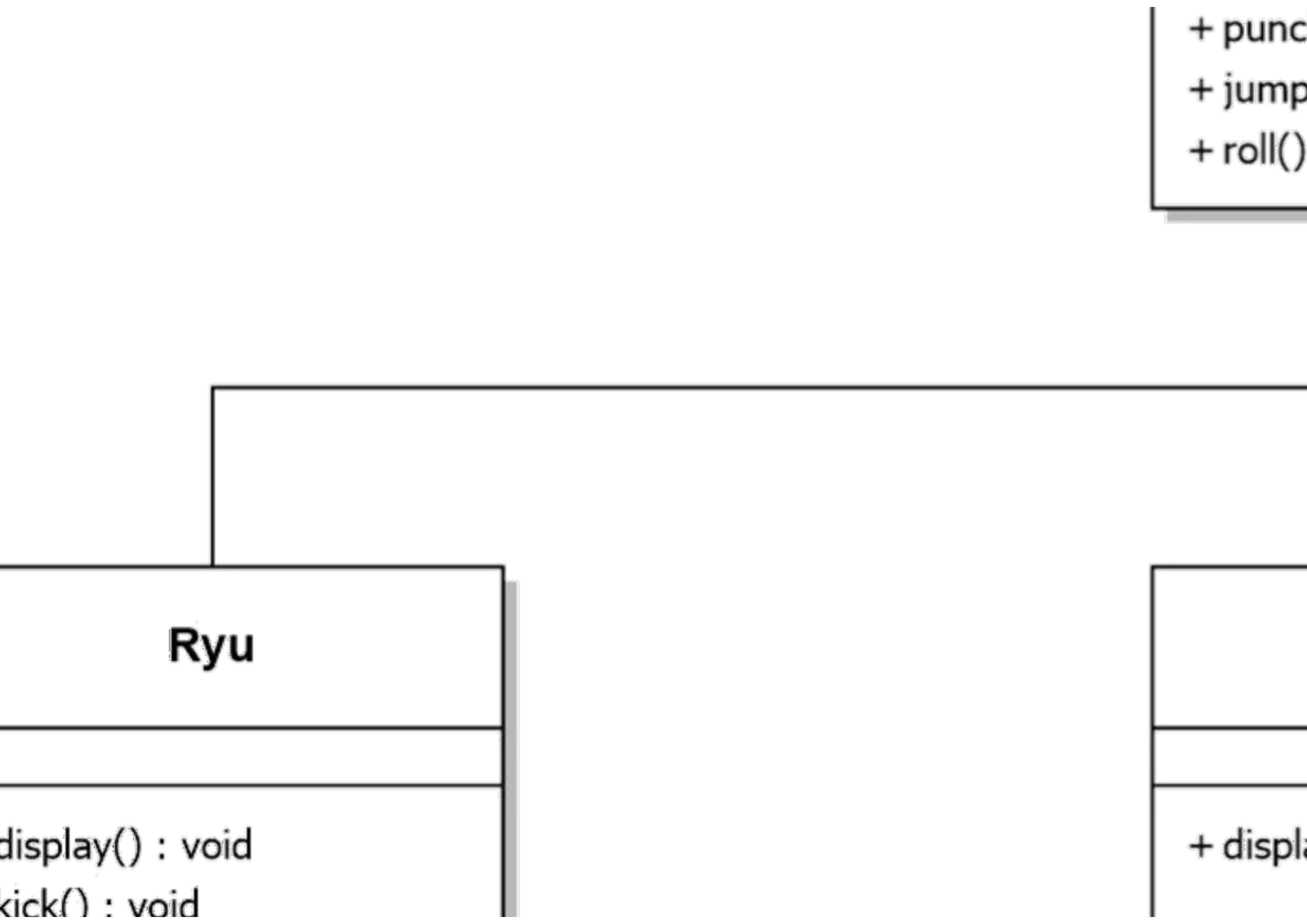
What if mandatory beha
implementations???

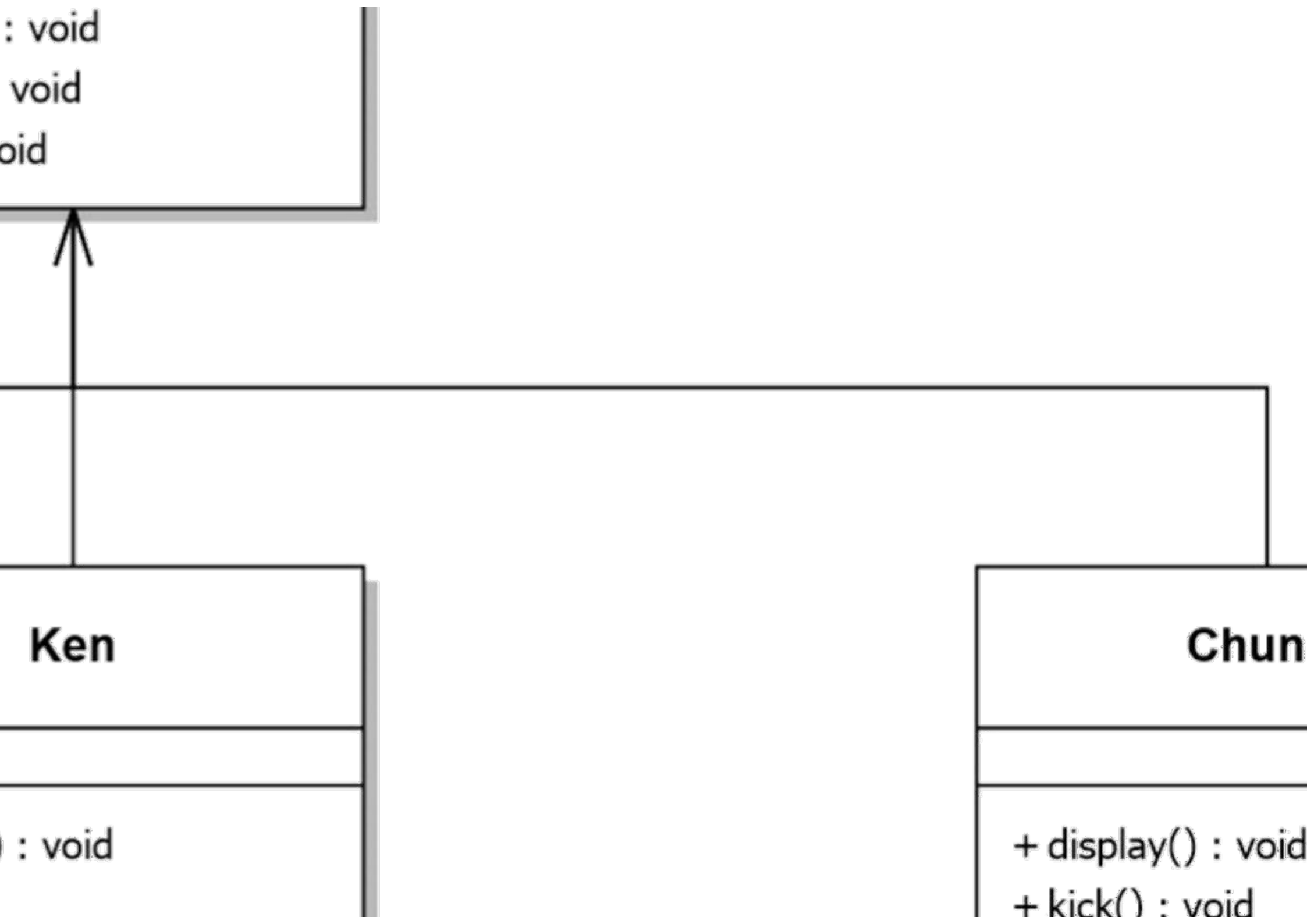
- E.g. kick and punch

More discussion

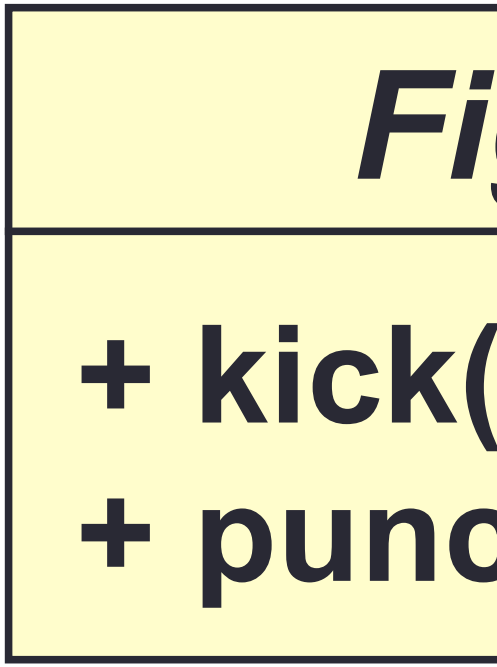
viors have different



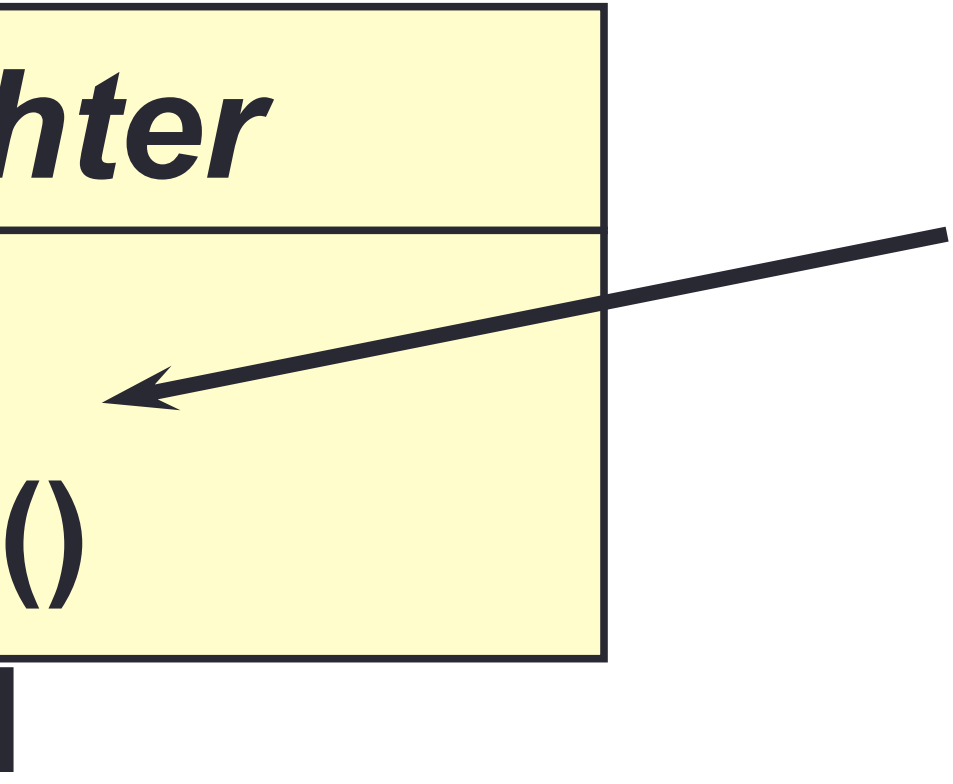

+ punc
+ jump
+ roll()
Ryu
display() : void
kick() : void
+ displa


: void
void
oid
Ken
) : void
Chun
+ display() : void
+ kick() : void

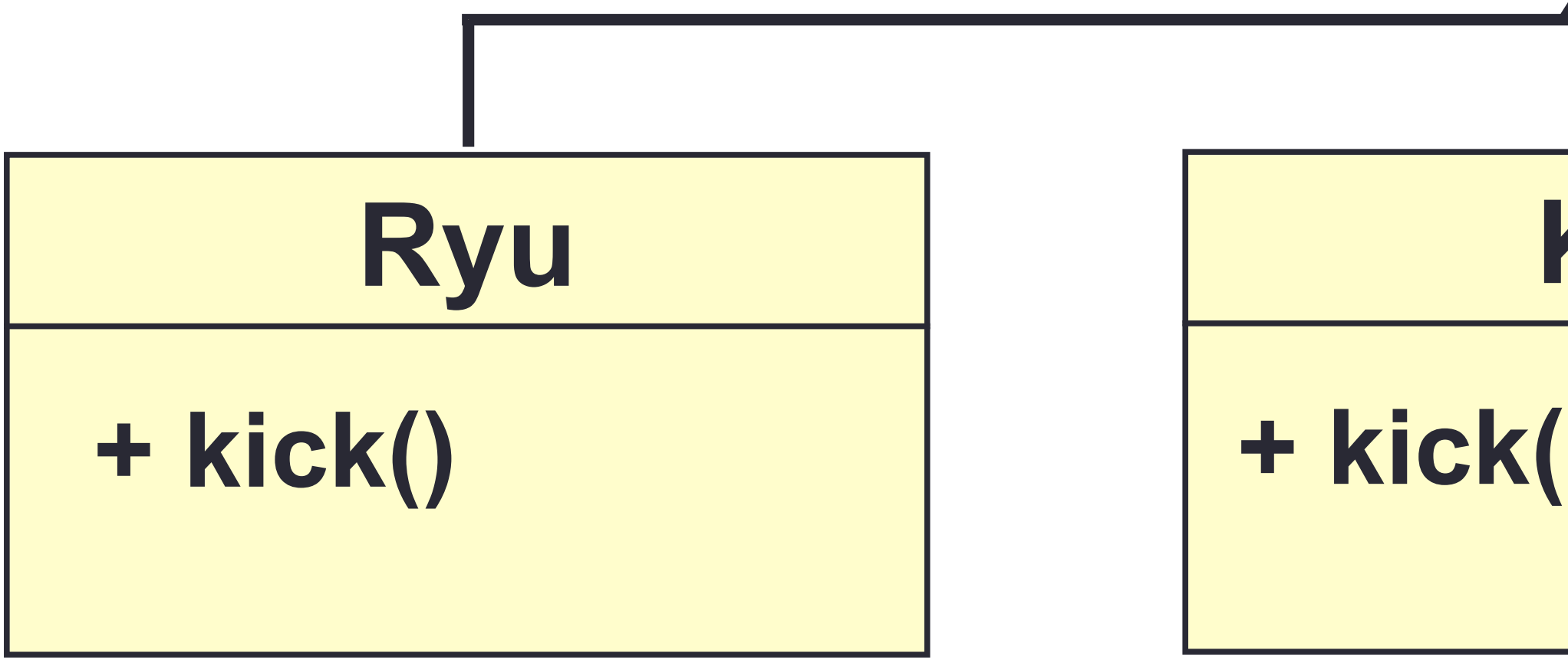
Street Fighter Gam

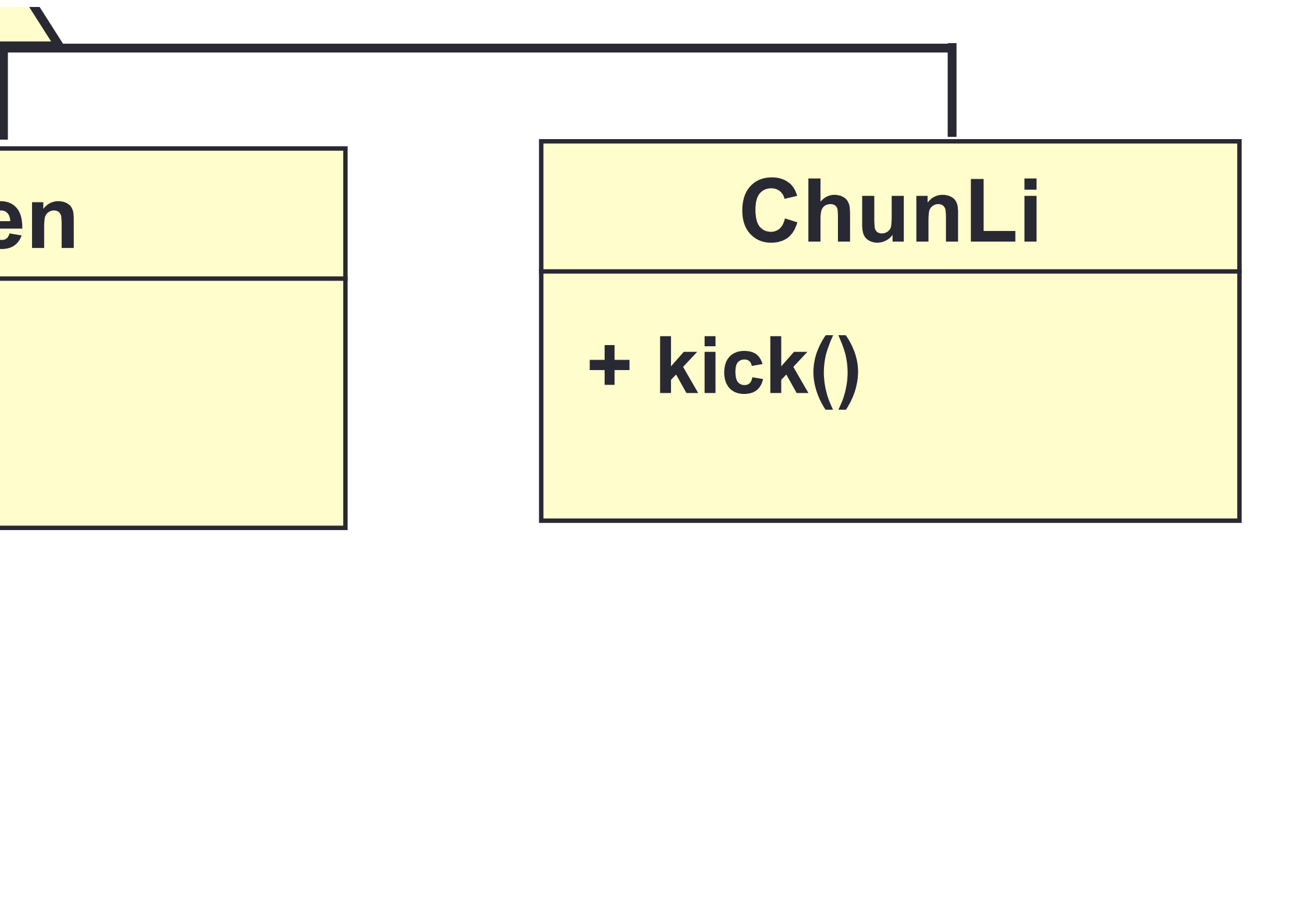


e: Subclassing



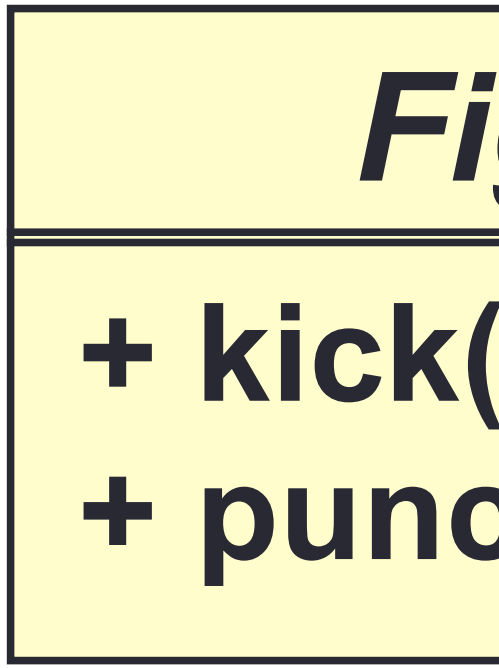
kick() has 3 differe
implementations


Ryu
+ kick()
+ kick(

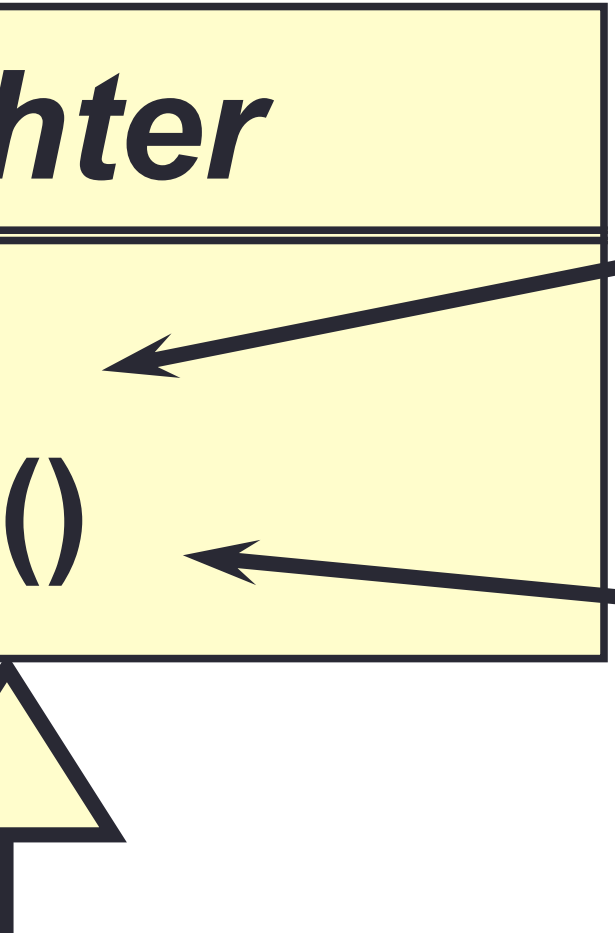

en
ChunLi
+ kick()

Street Fighter Gam

eed SIX classes to
andle both methods!!!

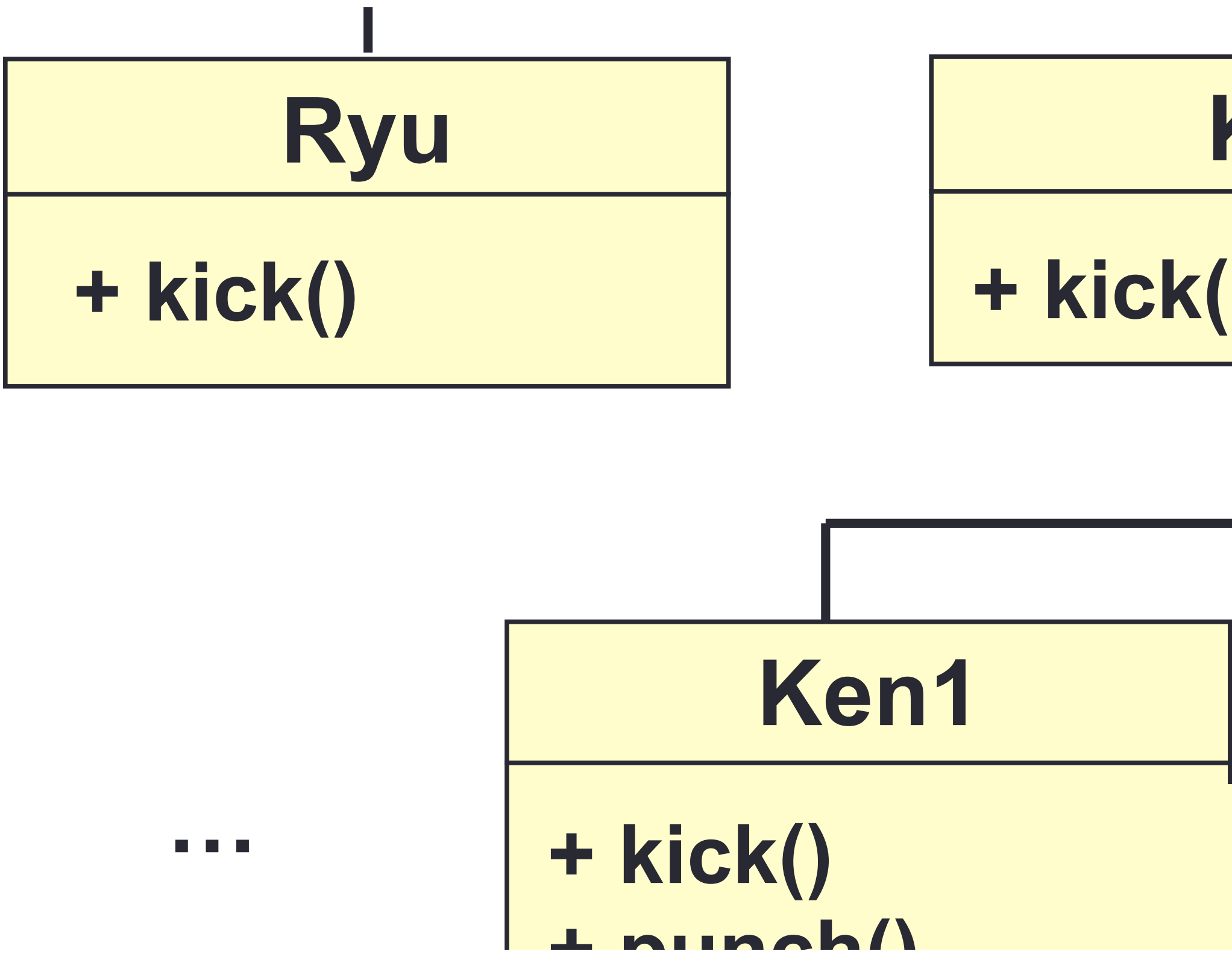


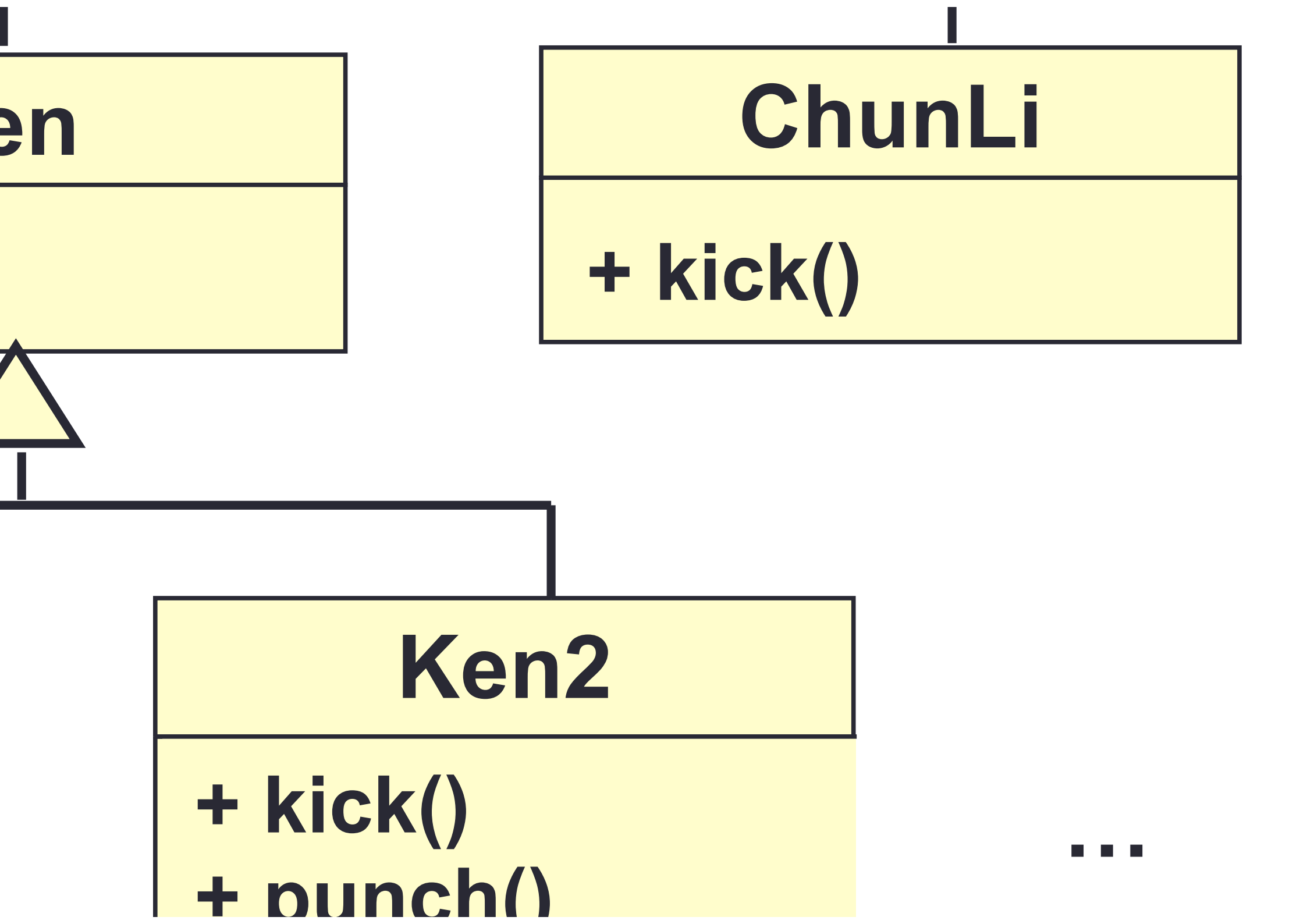
e: Subclassing



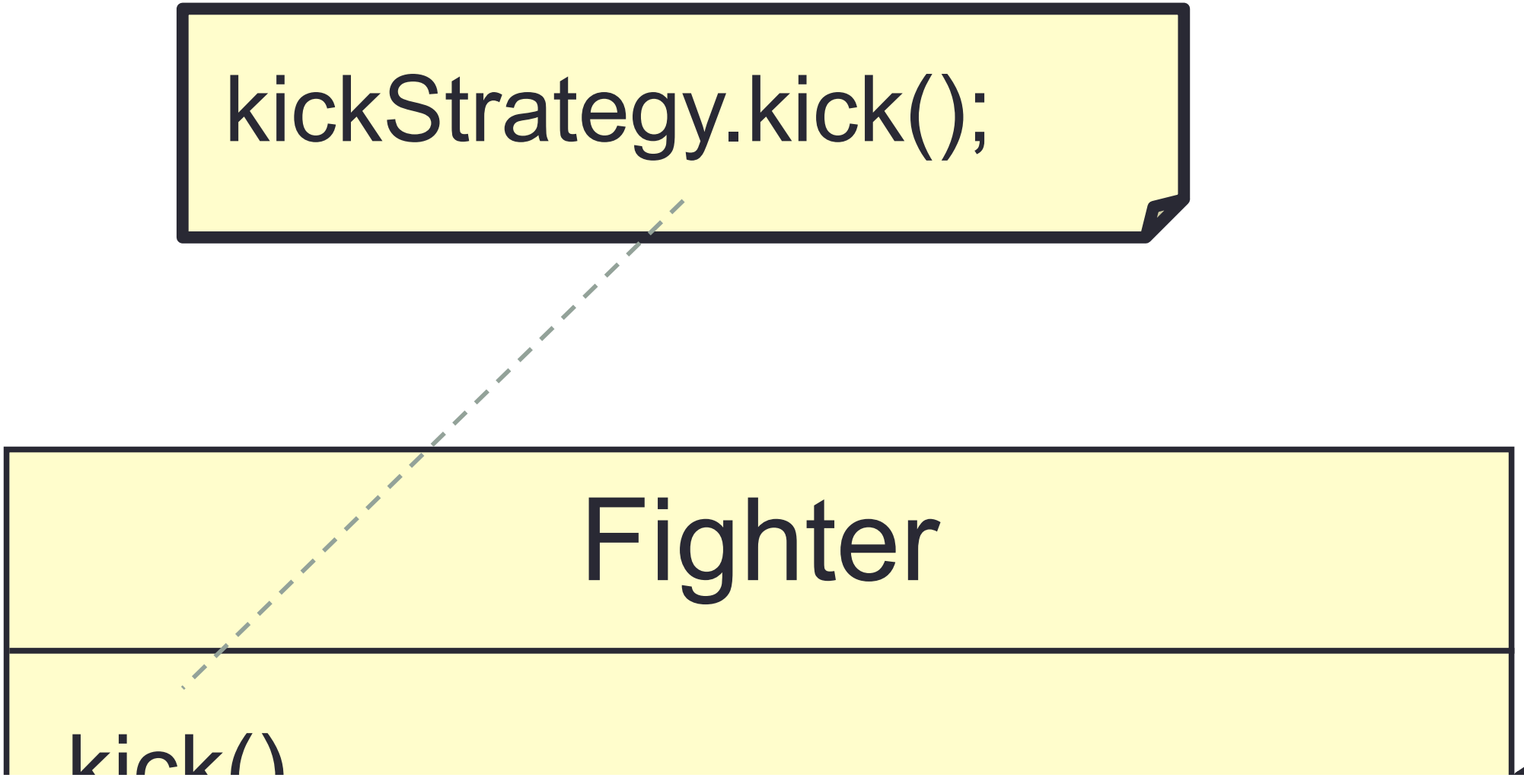
kick() has 3 differe
implementations

punch() has 2 differe
implementations

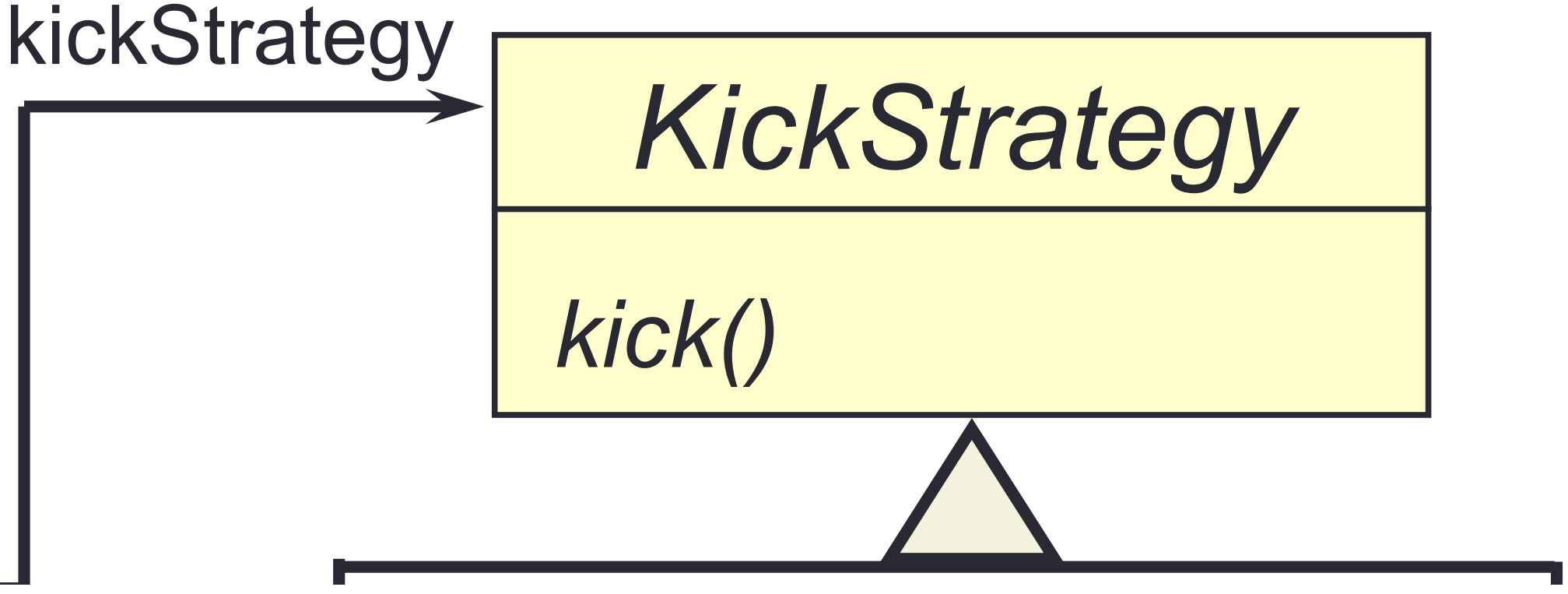

Ryu
+ kick()
+ kick(
Ken1
+ kick()
+ punch()
...


en
ChunLi
+ kick()
Ken2
+ kick()
+ punch()
...

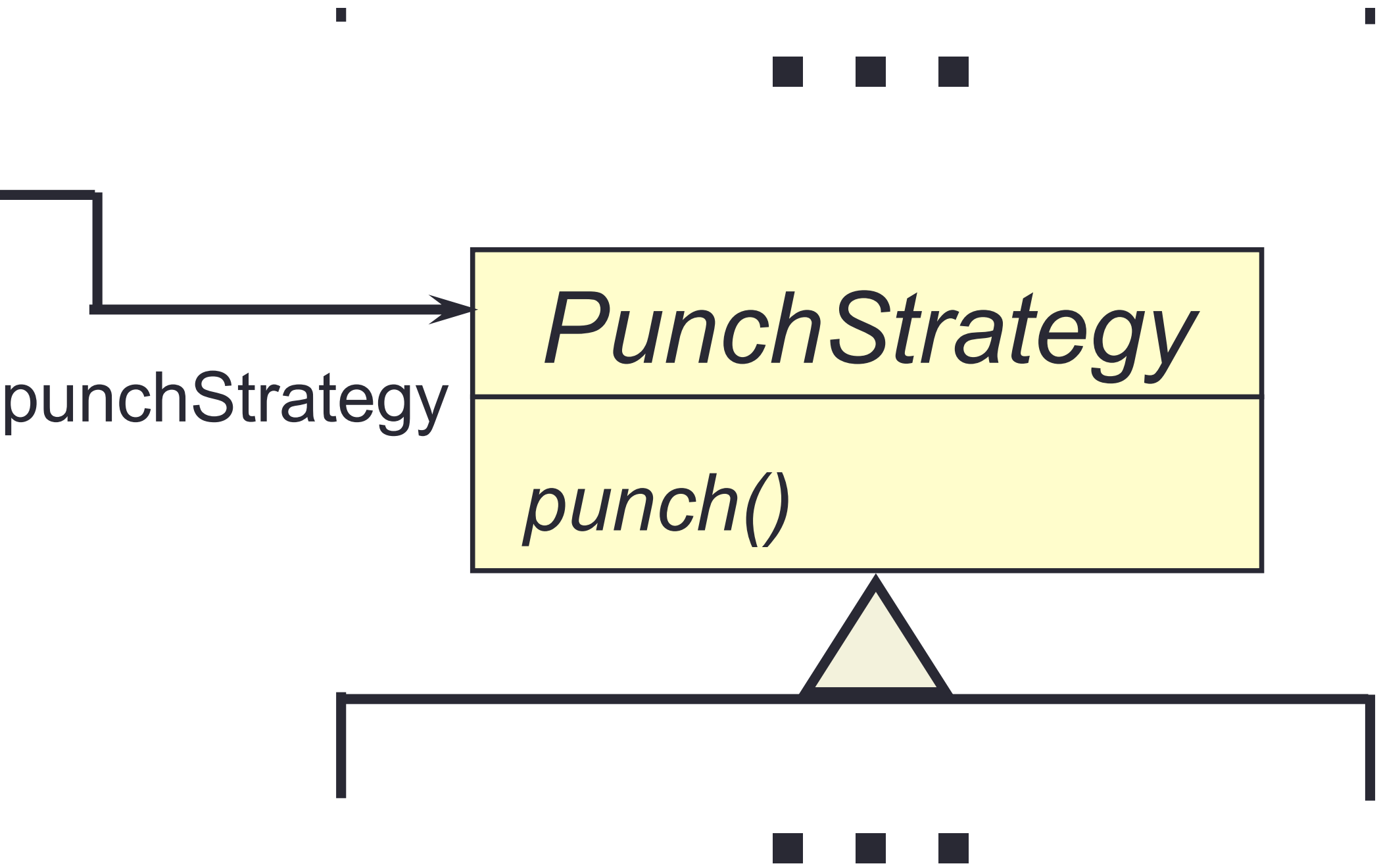
Street Fighter Gam



e: Strategy




punch()
setKickStrategy(KickStrategy)
setPunchStrategy(PunchStrategy)


punchStrategy
PunchStrategy
punch()

Strategy Design Pa

Defines a set of encaps
be swapped to carry out

- Separate behaviors that m
  classes in future from the

ttern

lated algorithms that car
a specific behavior

ay vary across different
est

Enables an algorithm's
runtime

- defines a family of algorith
- encapsulates each algorit
- makes the algorithms inte

➔ Favor Composition to

ehavior to be selected a

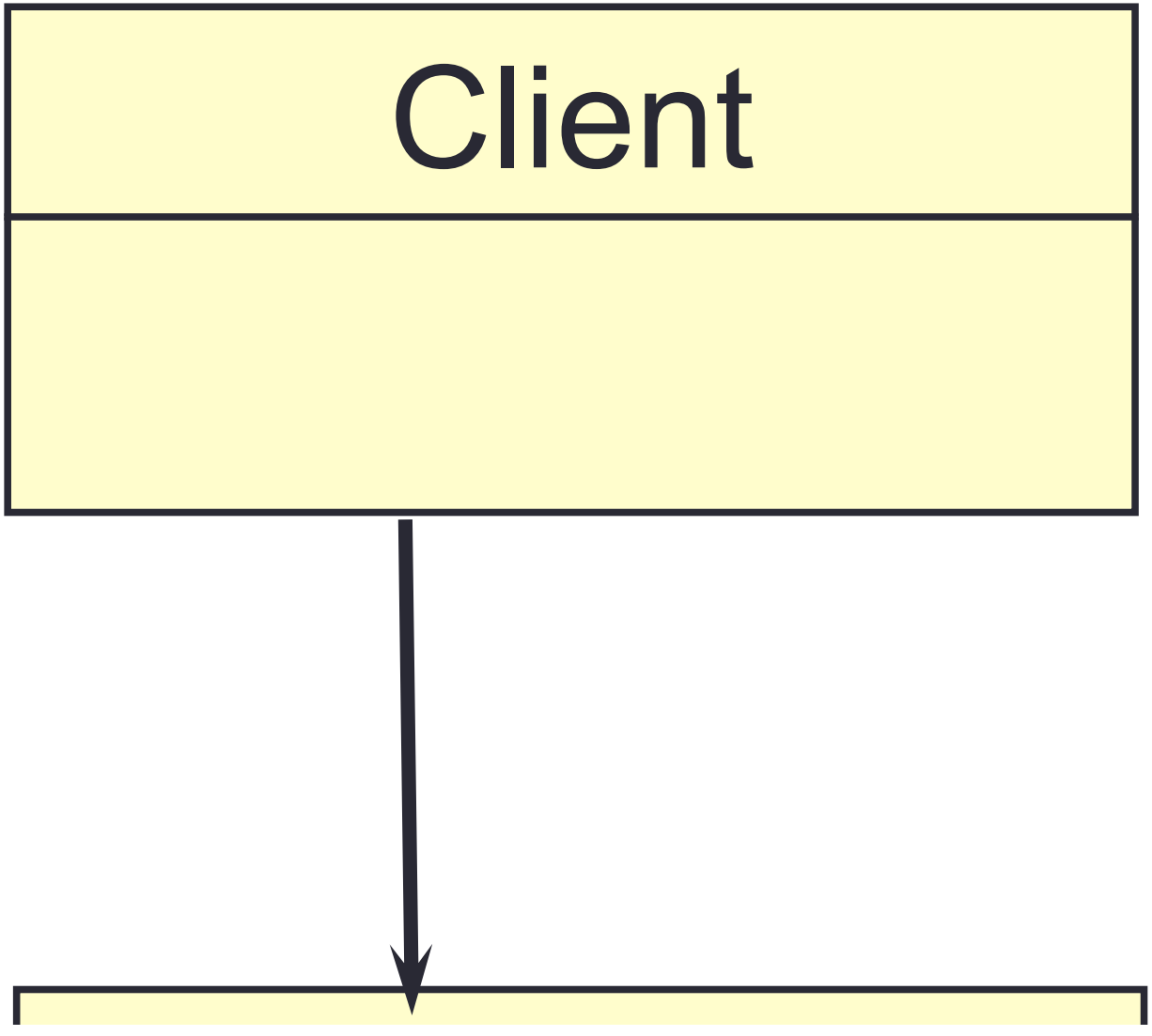
ms,

m, and

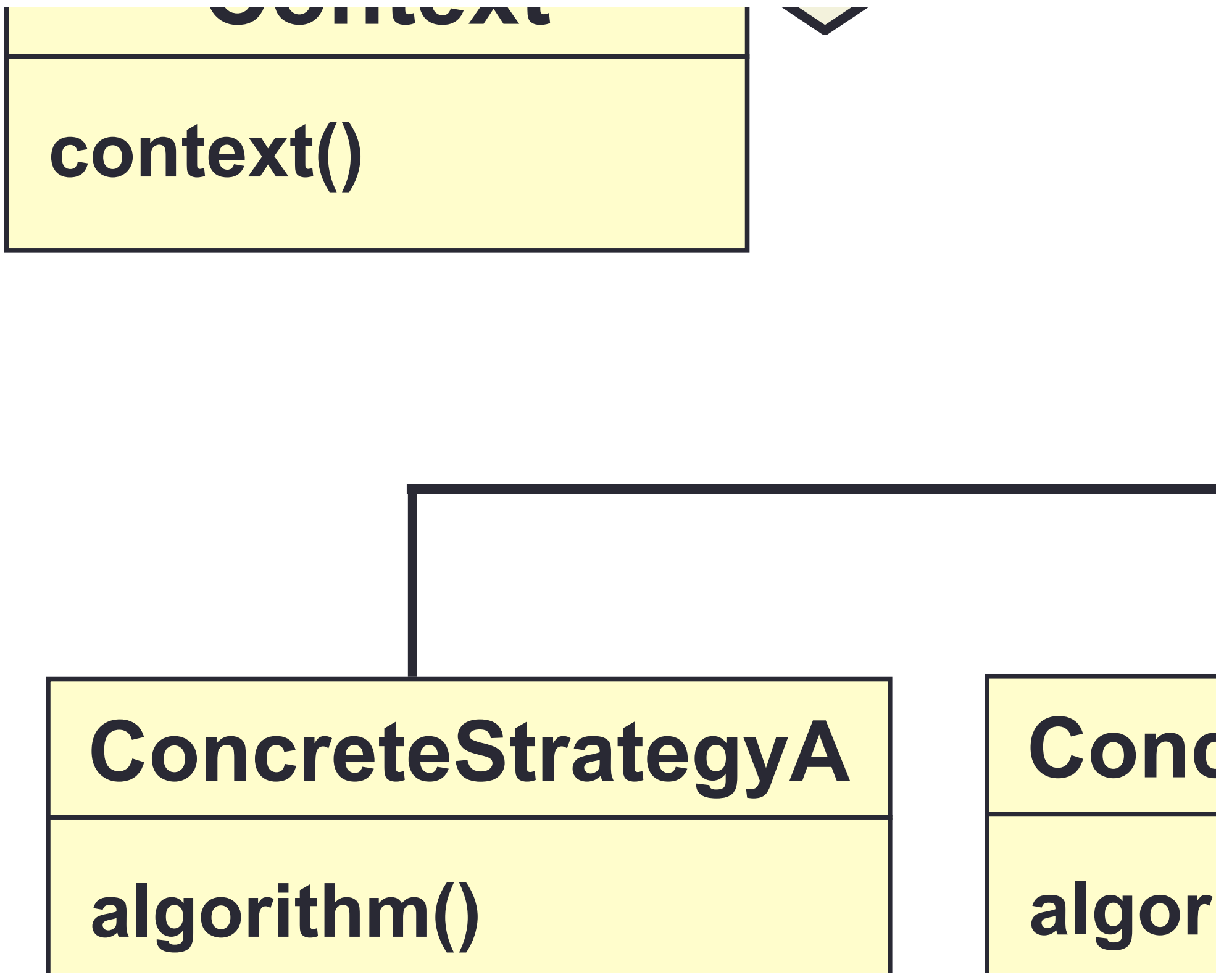
changeable within that family

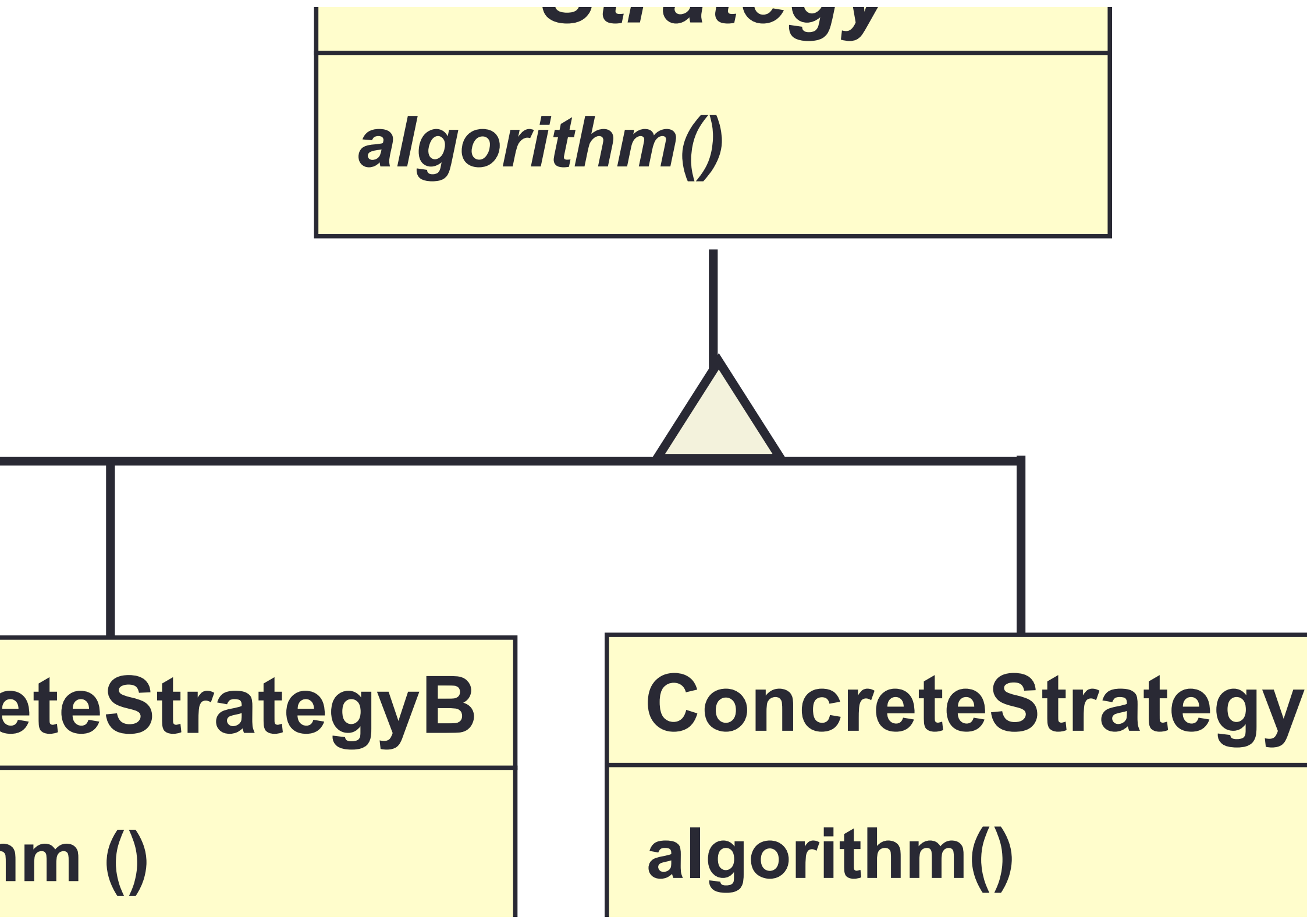
nheritance for Reuse

Strategy Pattern: S

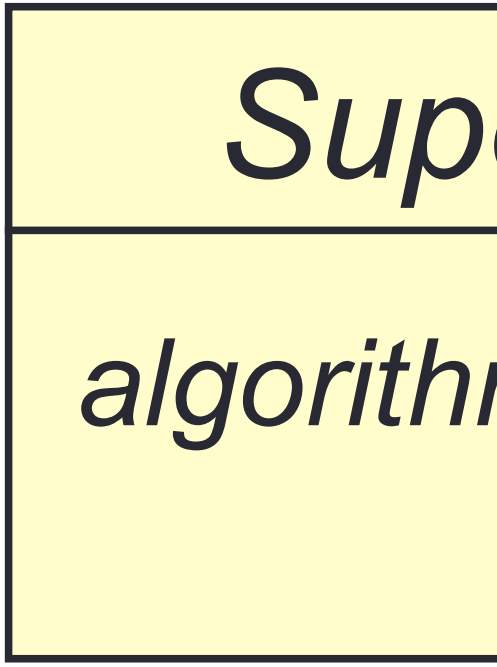


tructure

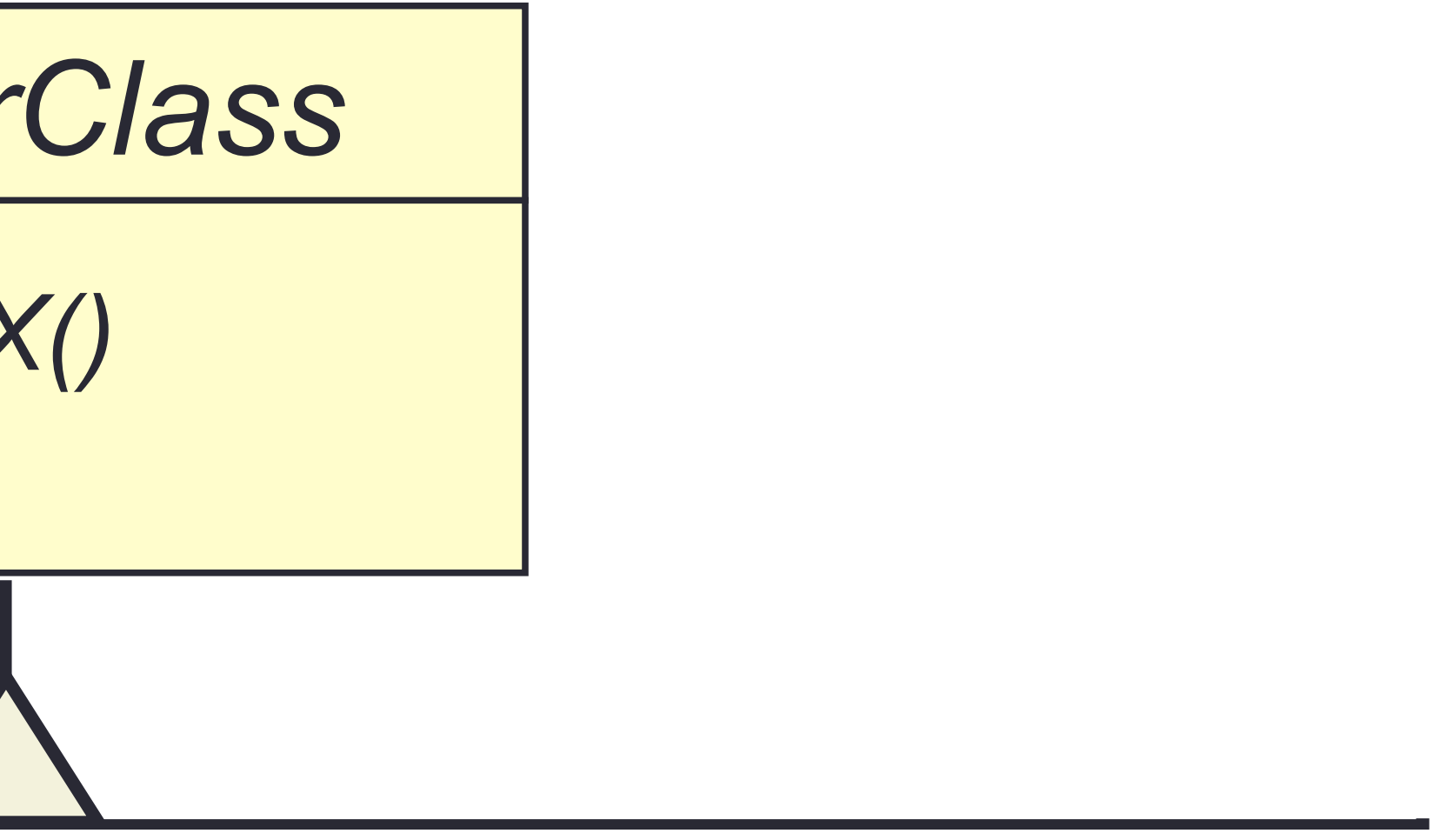

context()
ConcreteStrategyA
algorithm()

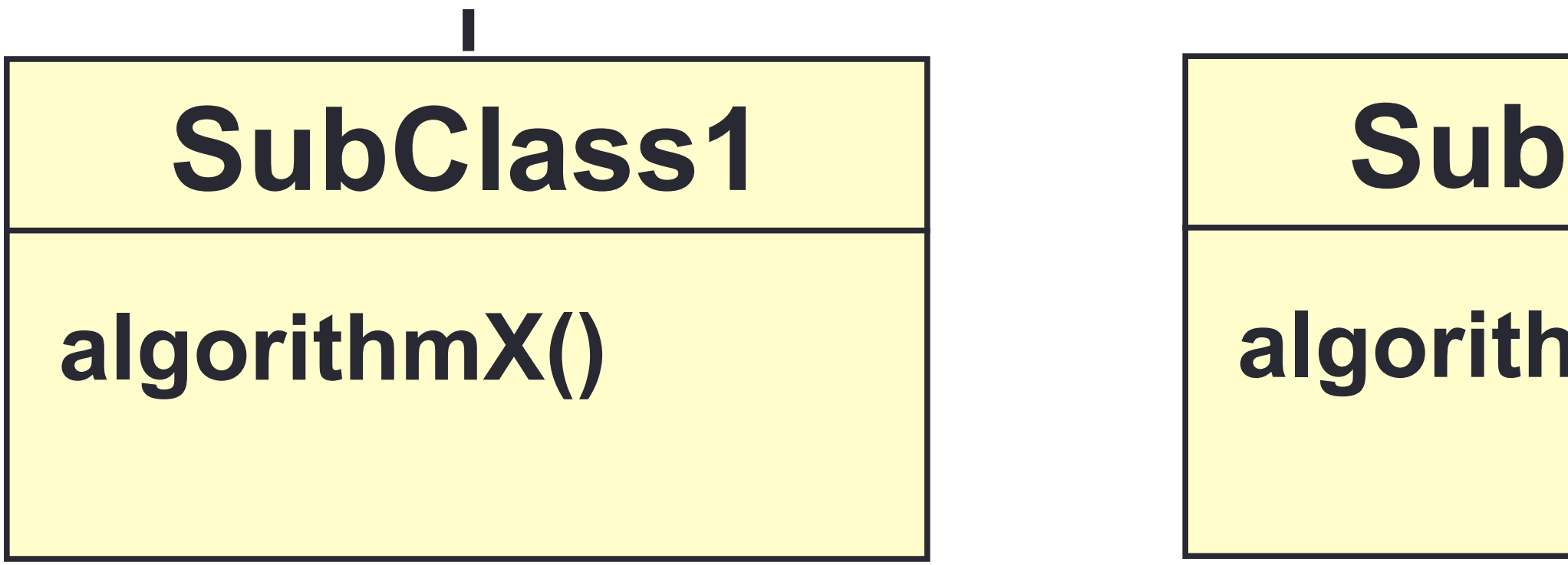

algorithm()
eteStrategyB
hm ()
ConcreteStrategy
algorithm()

Subclassing vs Strate

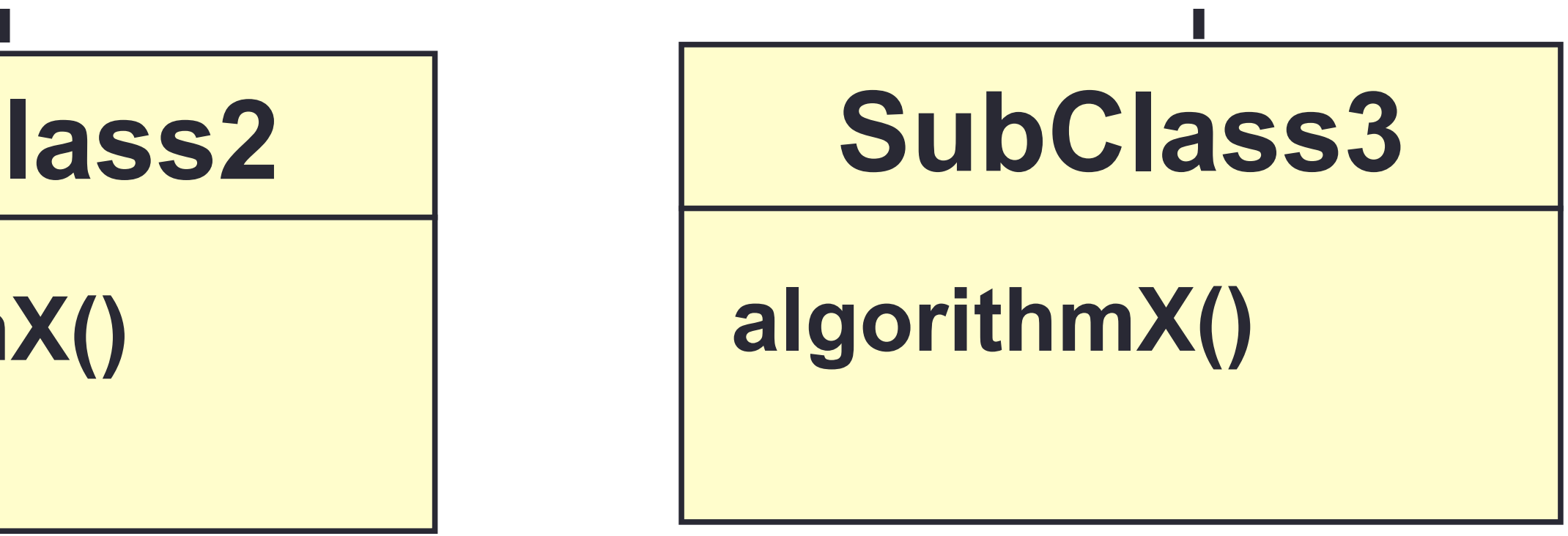


egy: More discussion



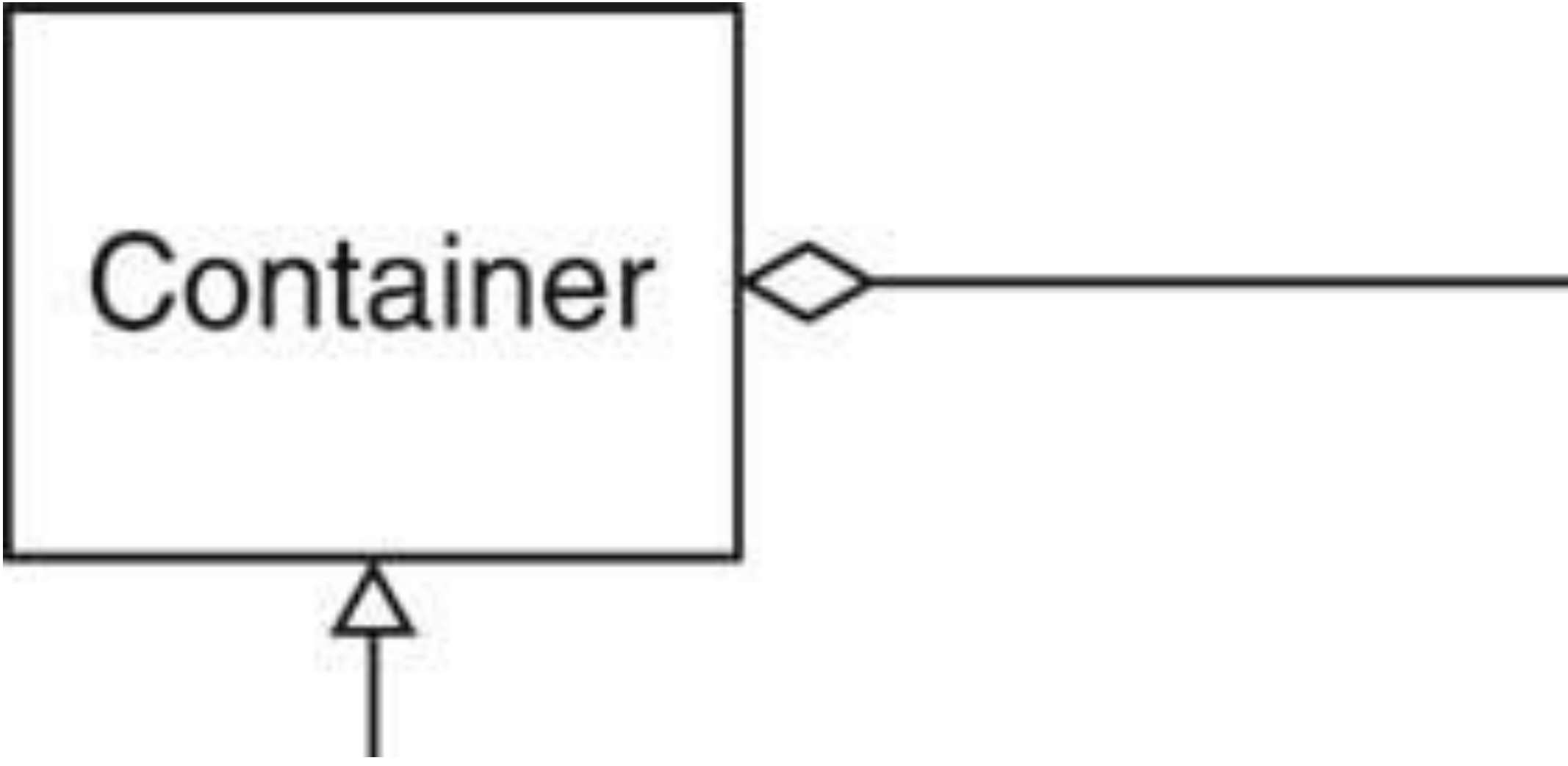


What if SubClass1 r
SubClass2 a

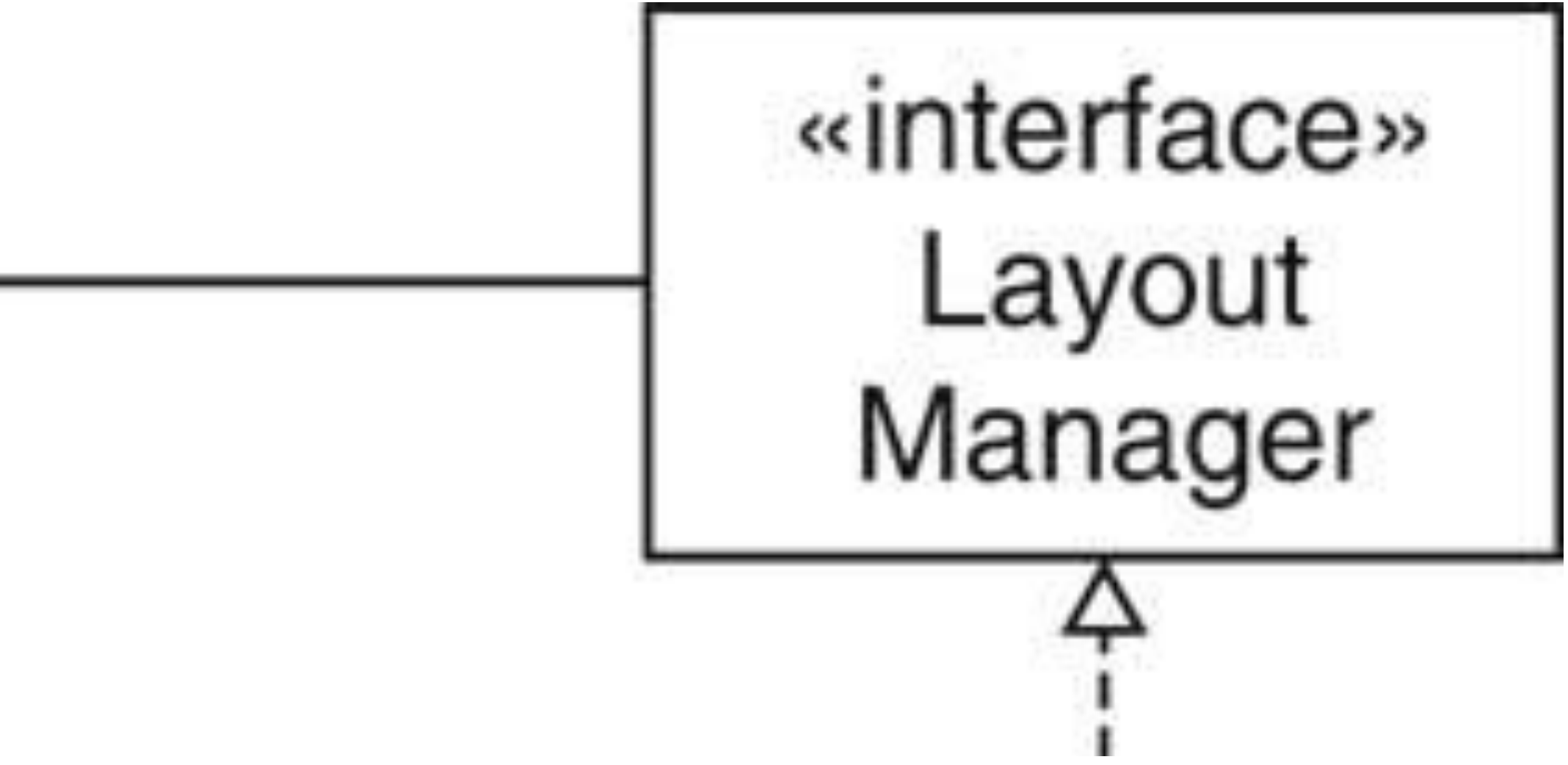


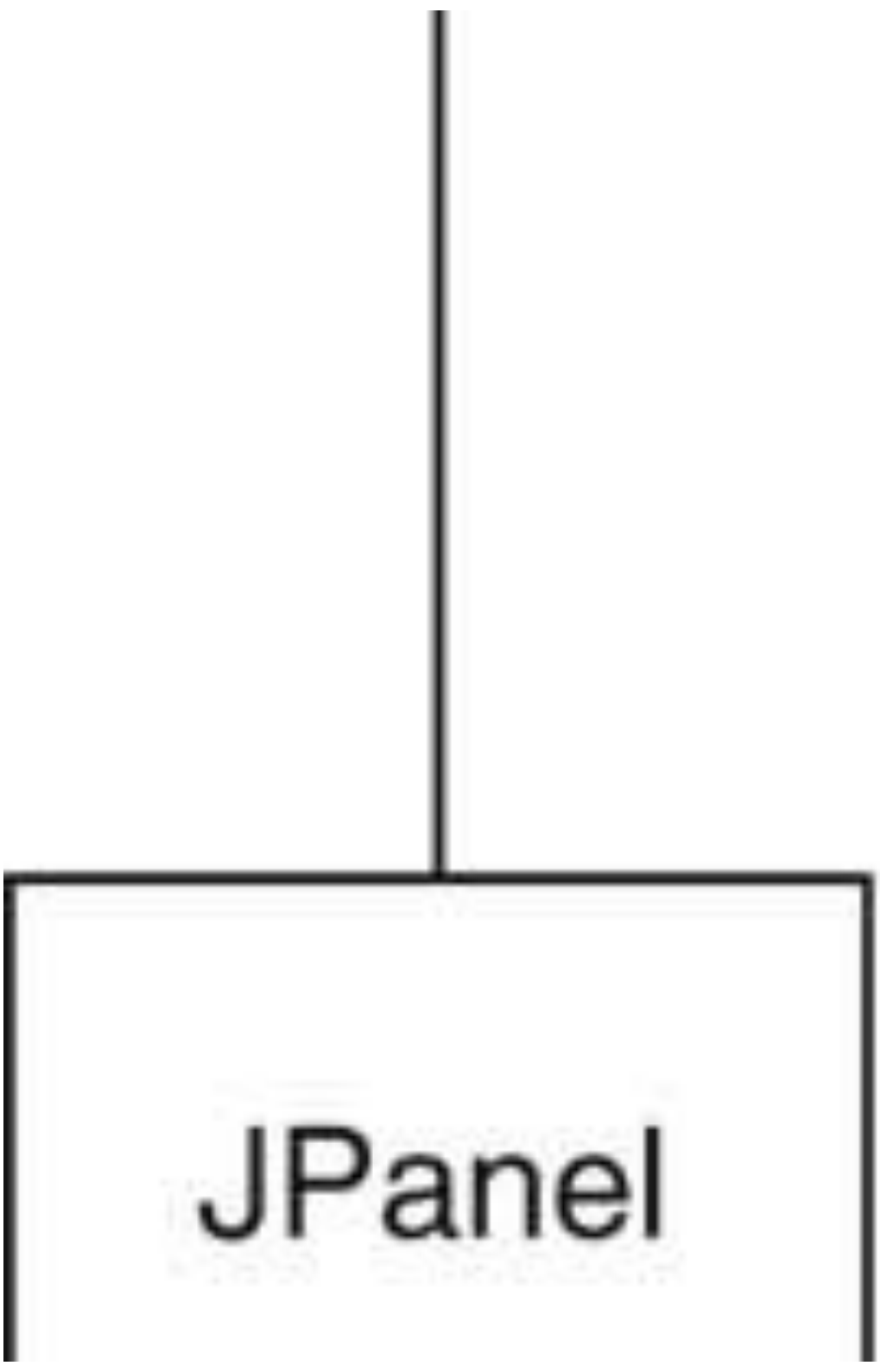
eeds to change to
t runtime?

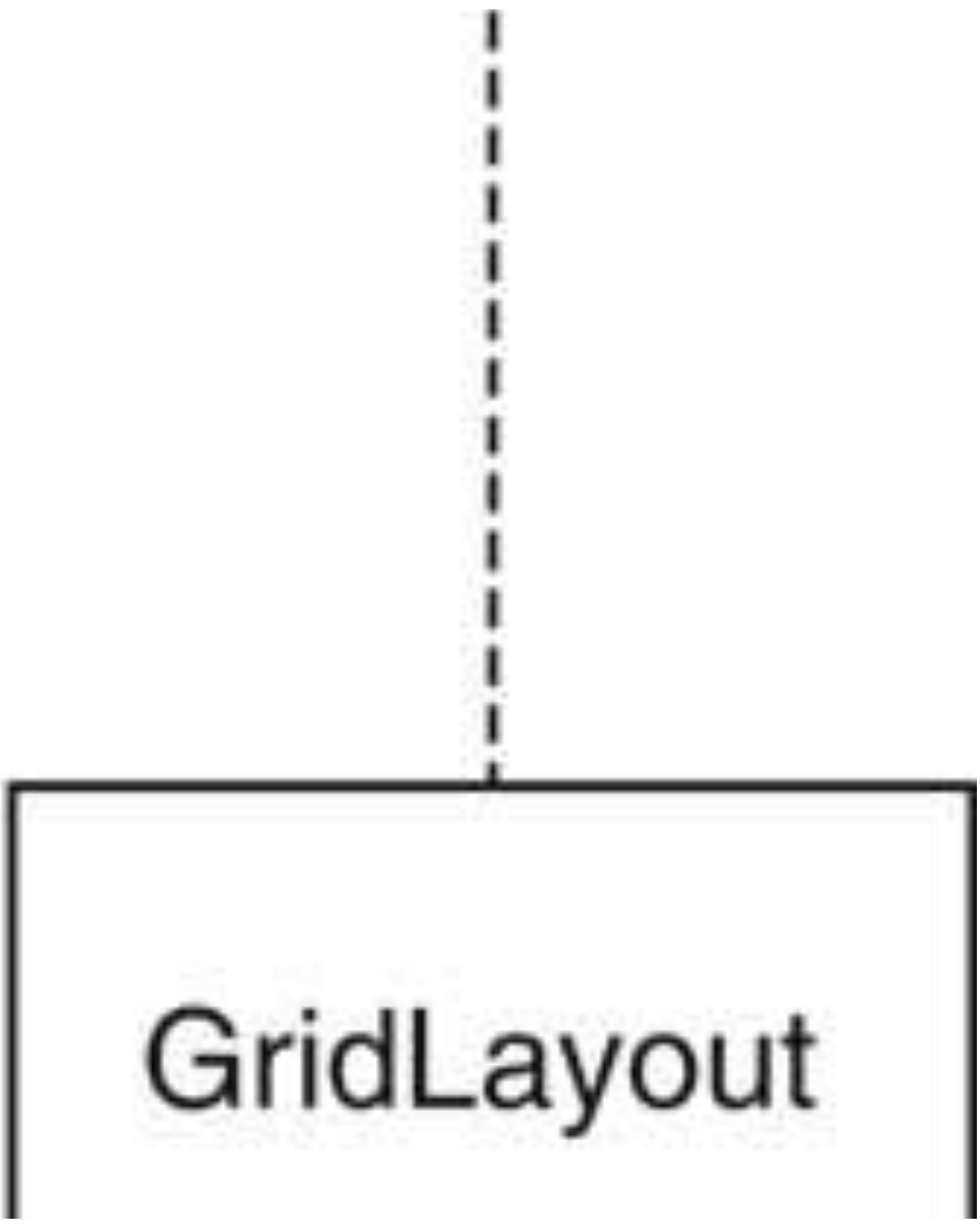
Strategy: Layout M



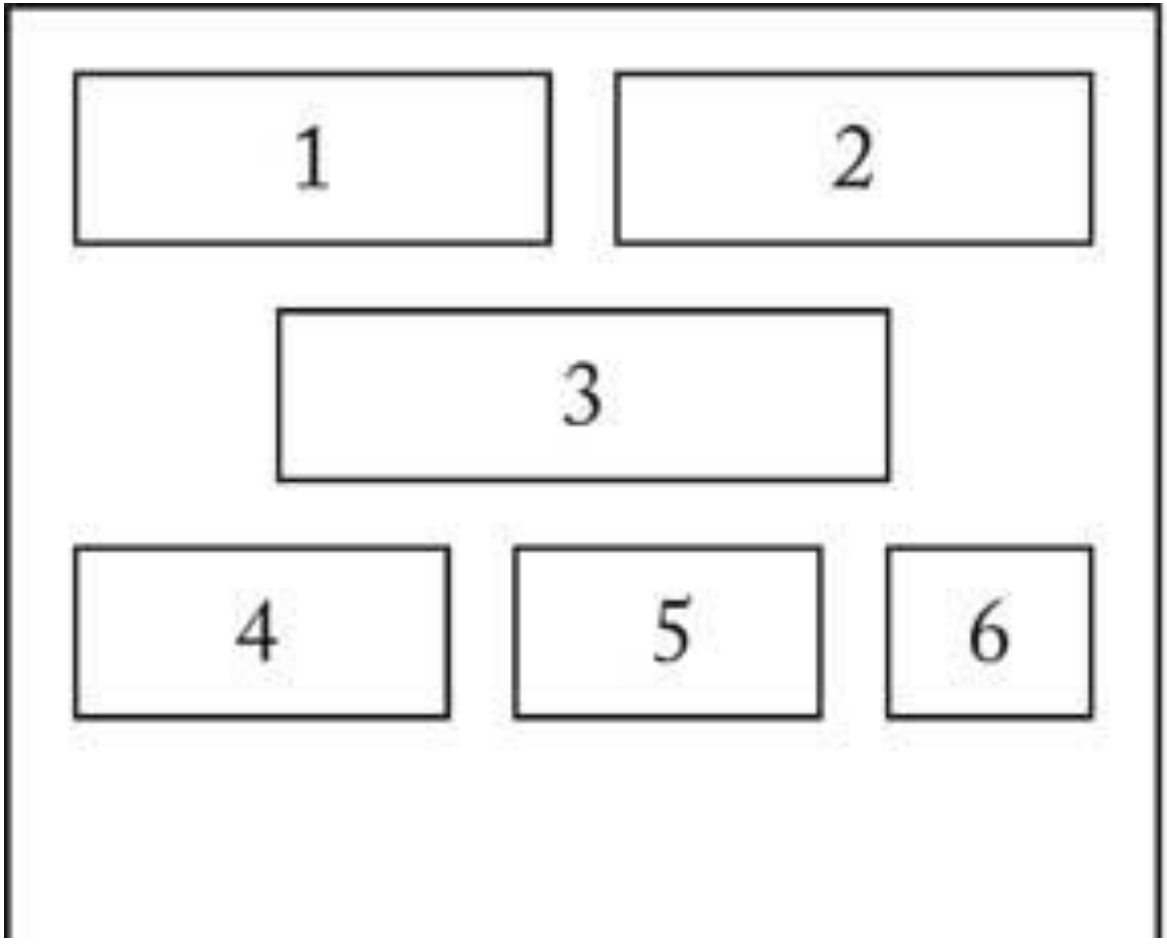
anagers

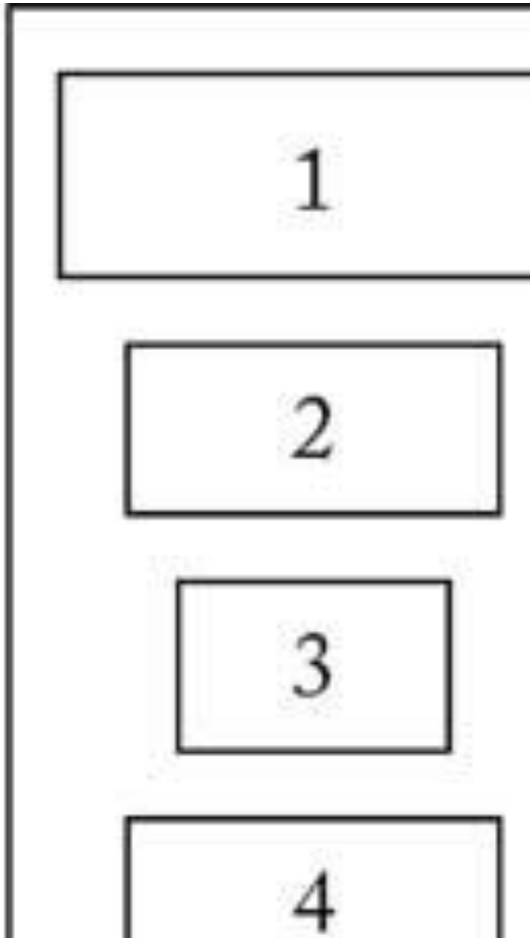



JPanel

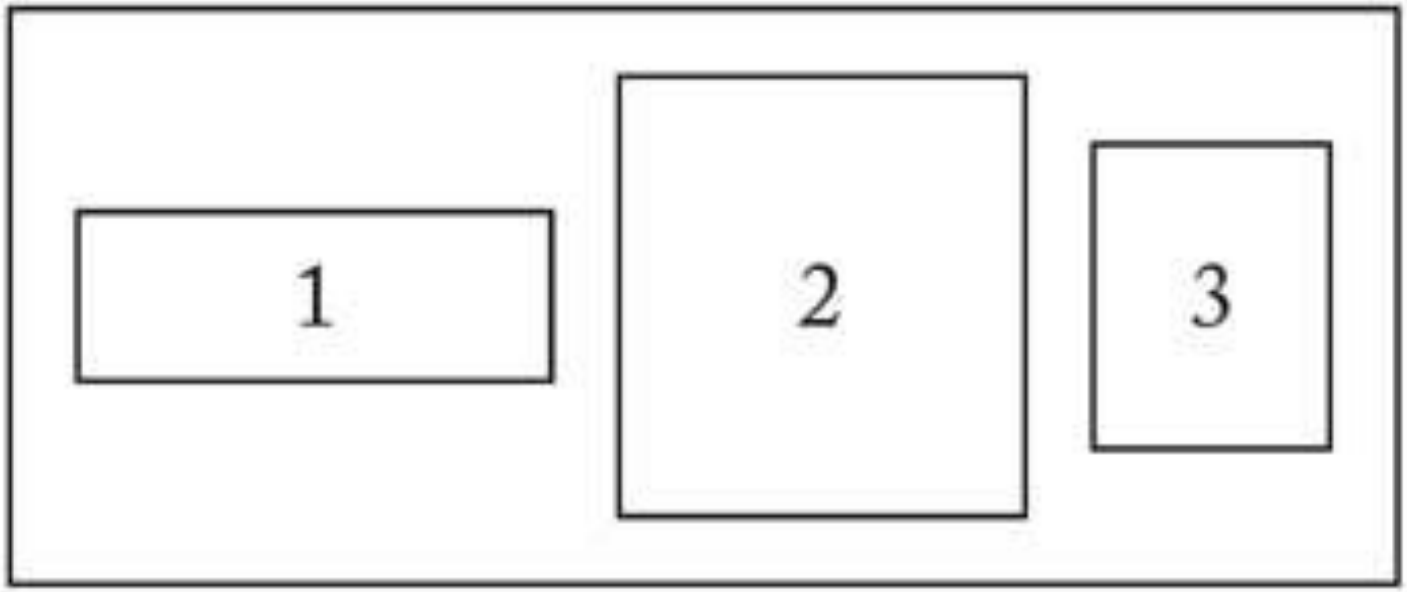

GridLayout

Strategy: Layout M



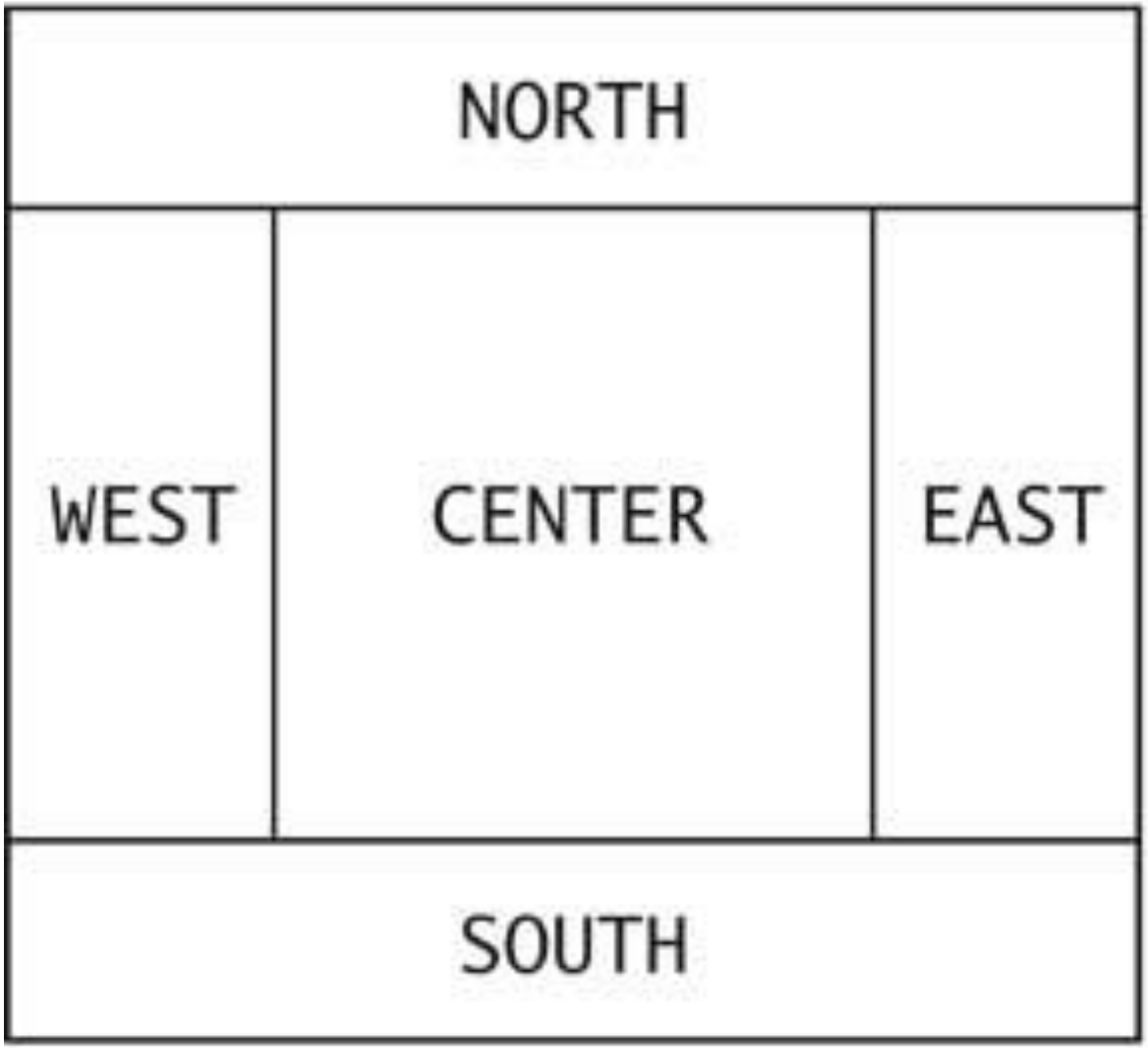


anagers

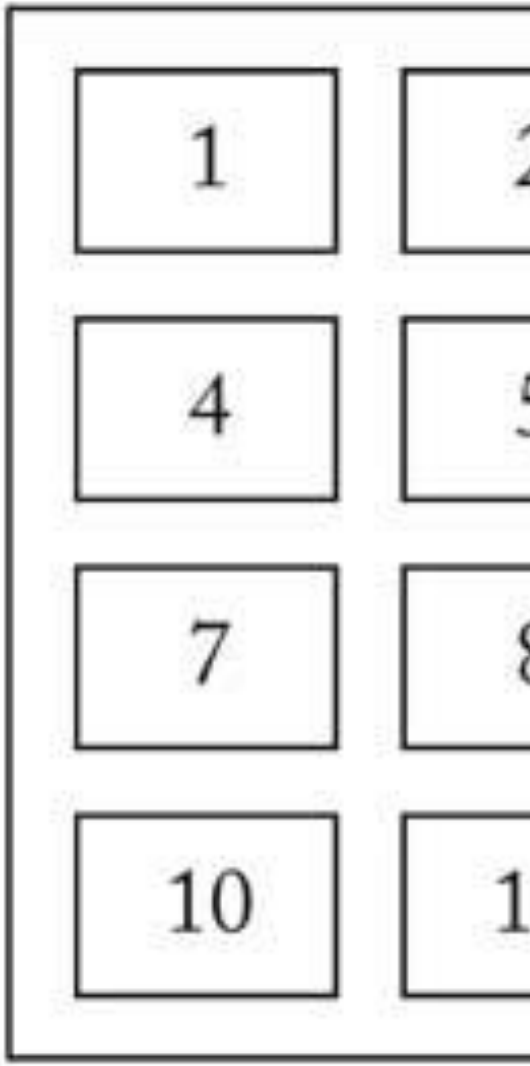


BoxLayout (horizontal)

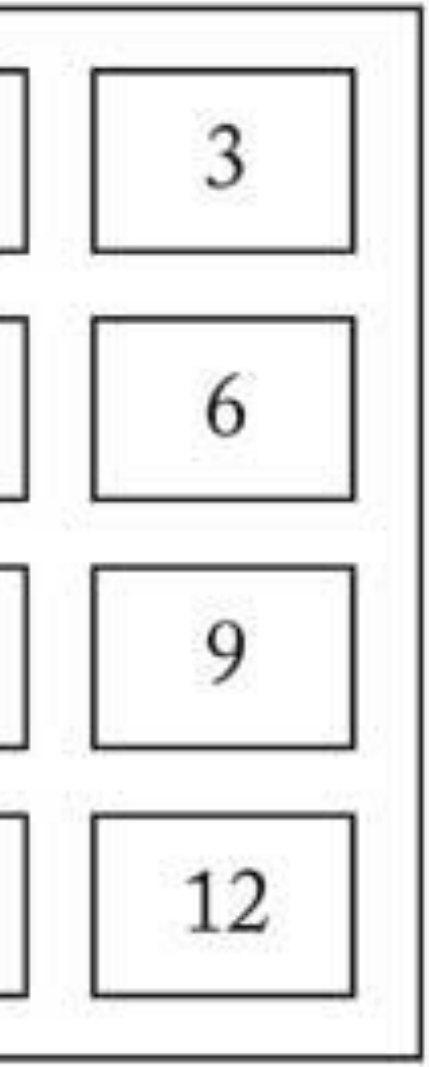
FlowLayout

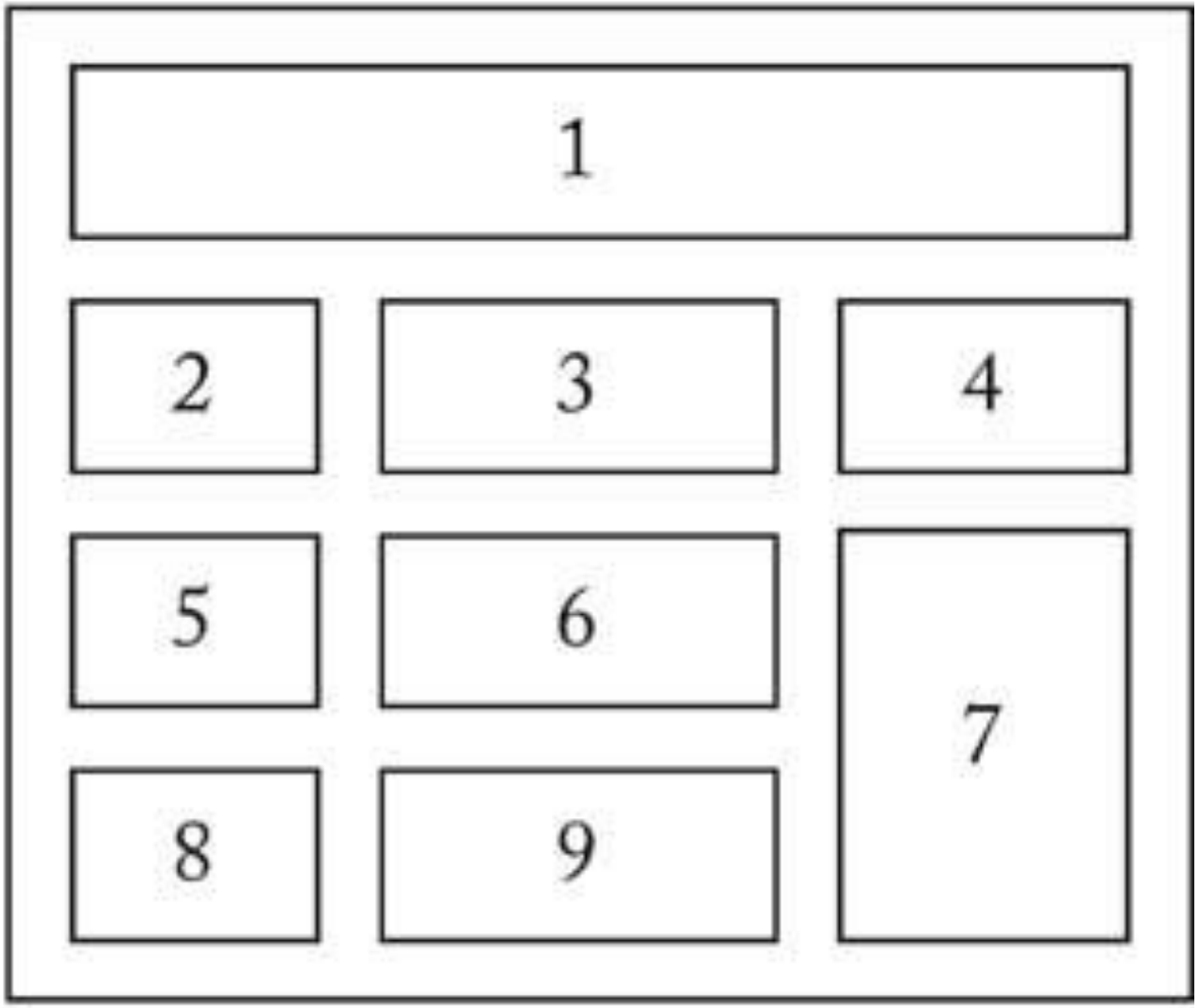


BoxLayout

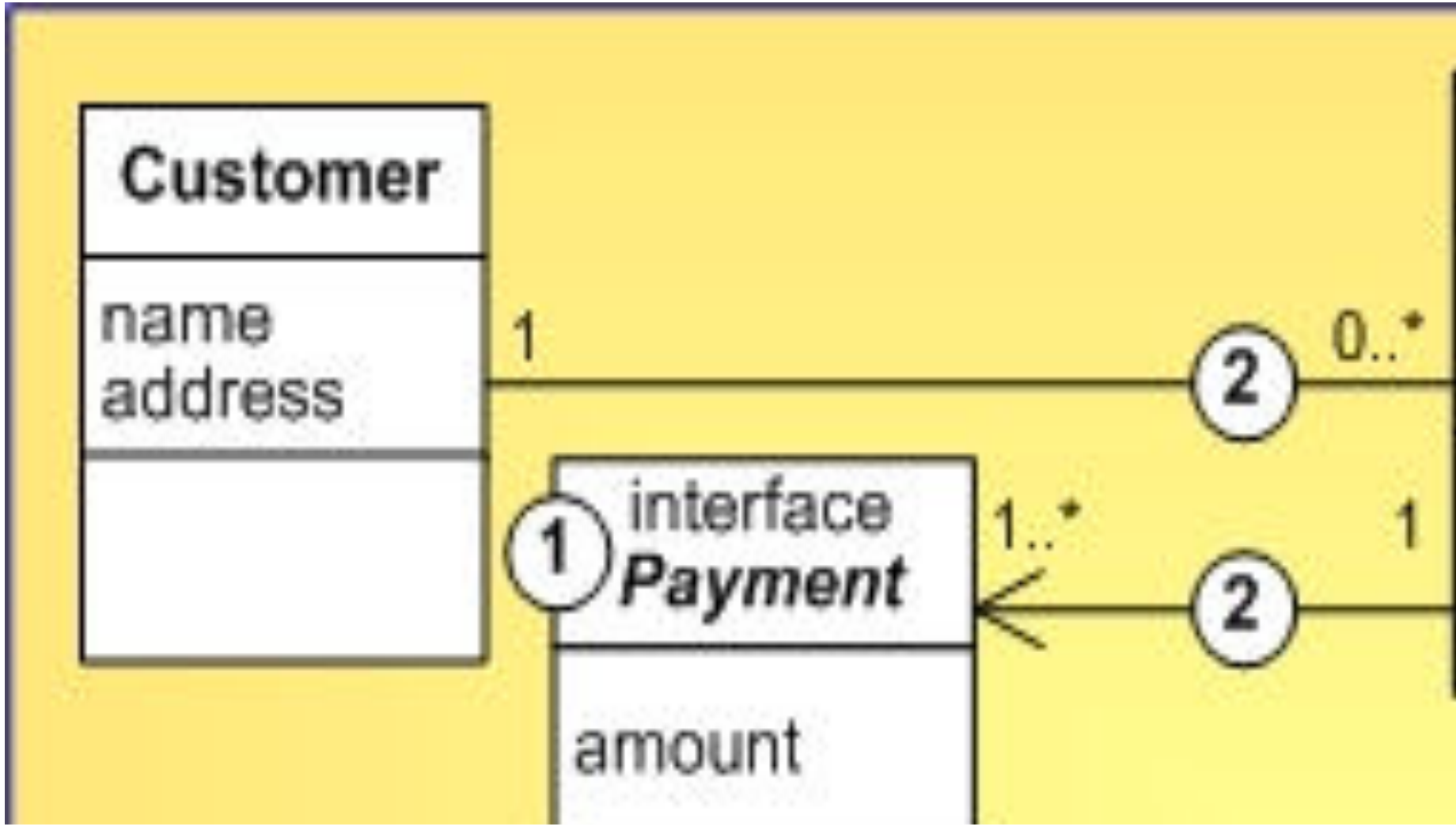


rtical)

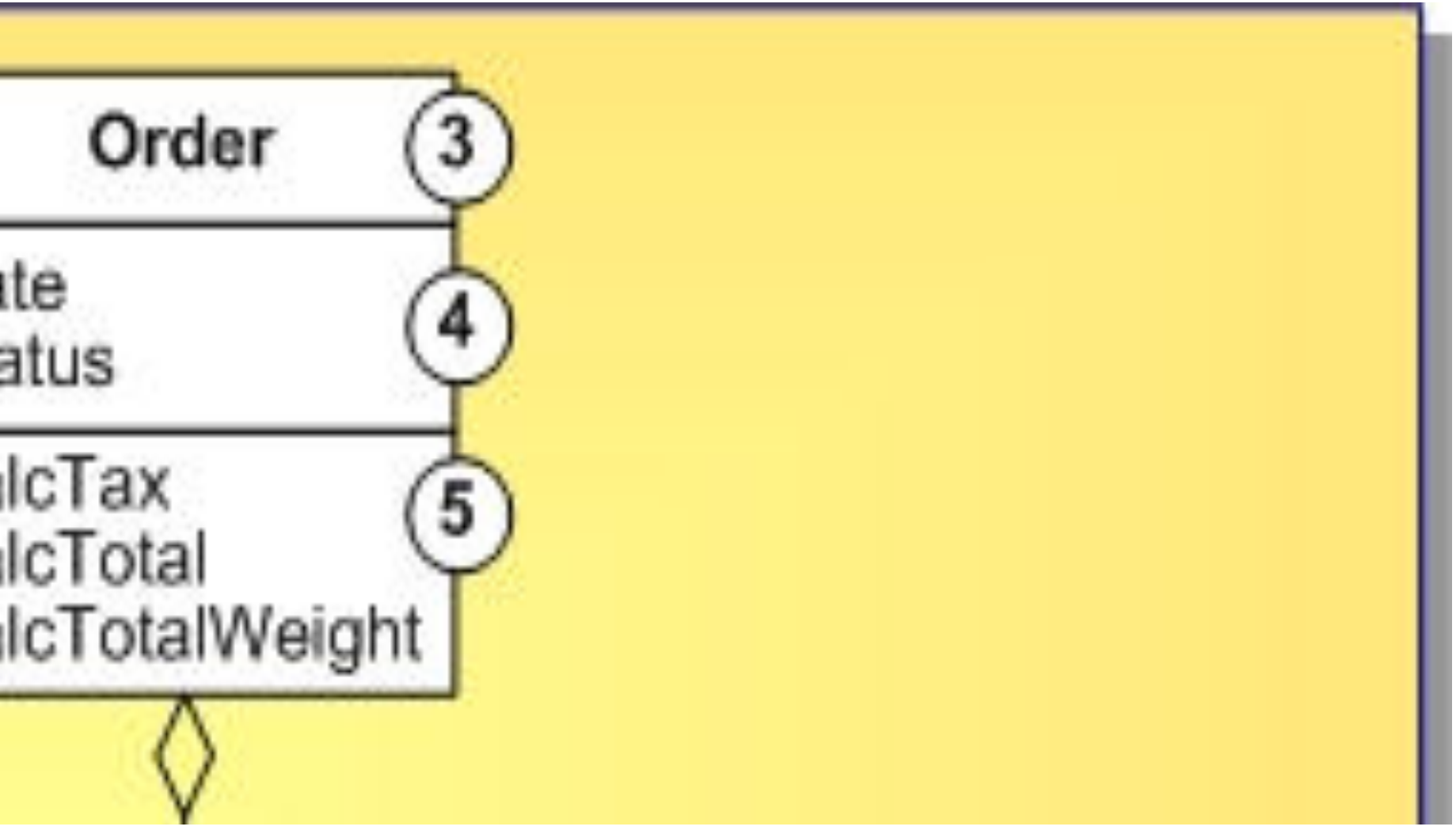


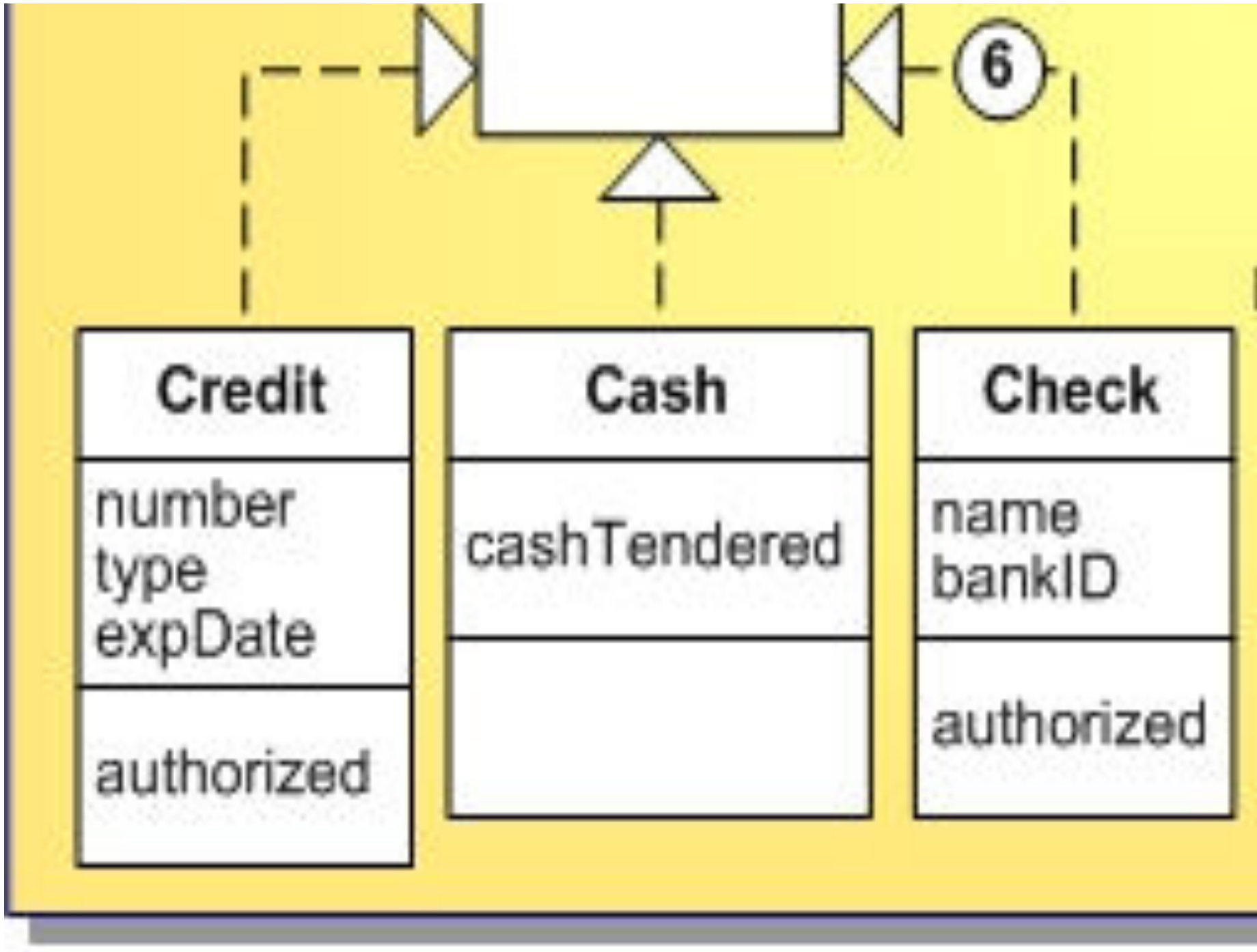


Strategy: Payment

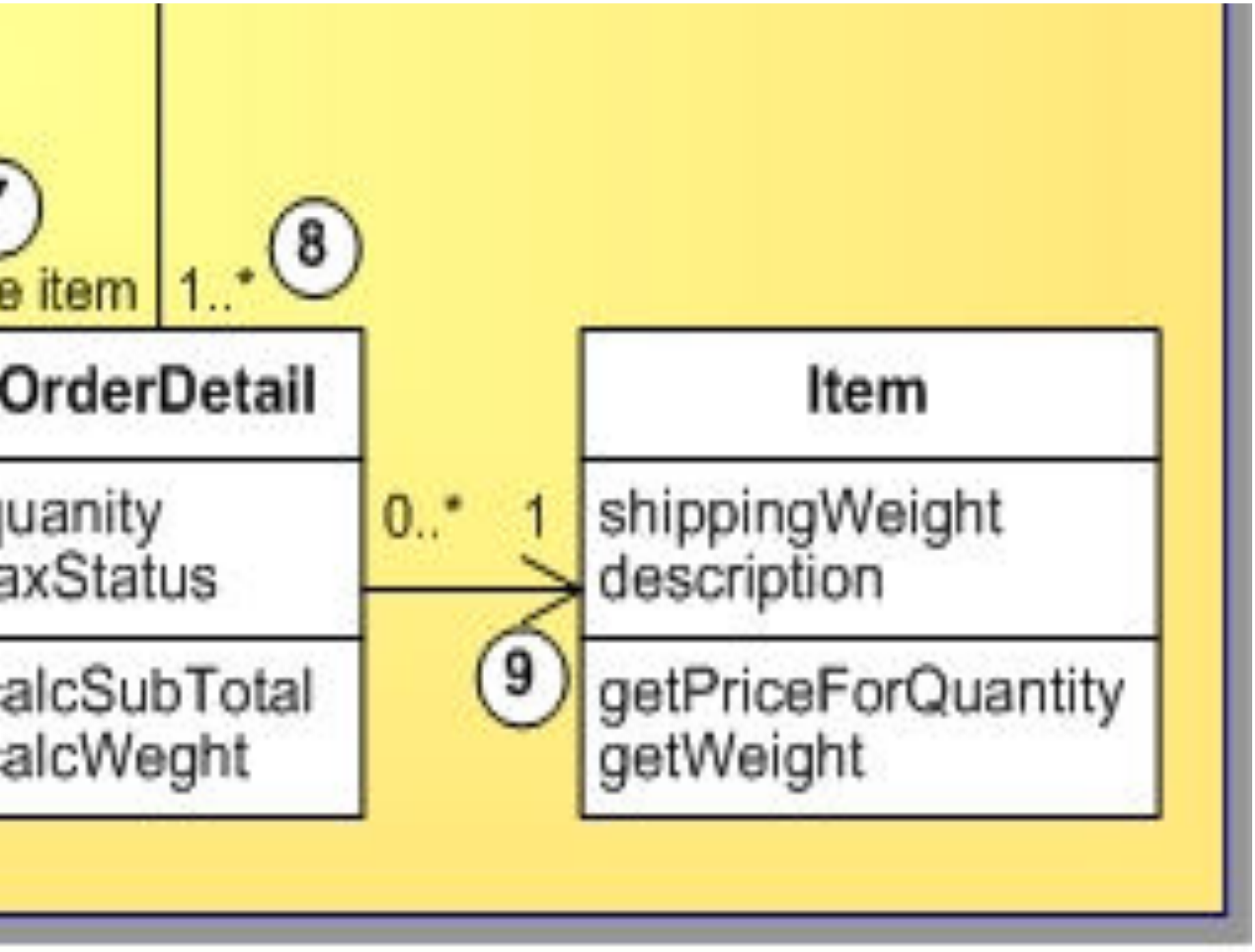


Method





1 Interface 3 Class Name 5 Meth

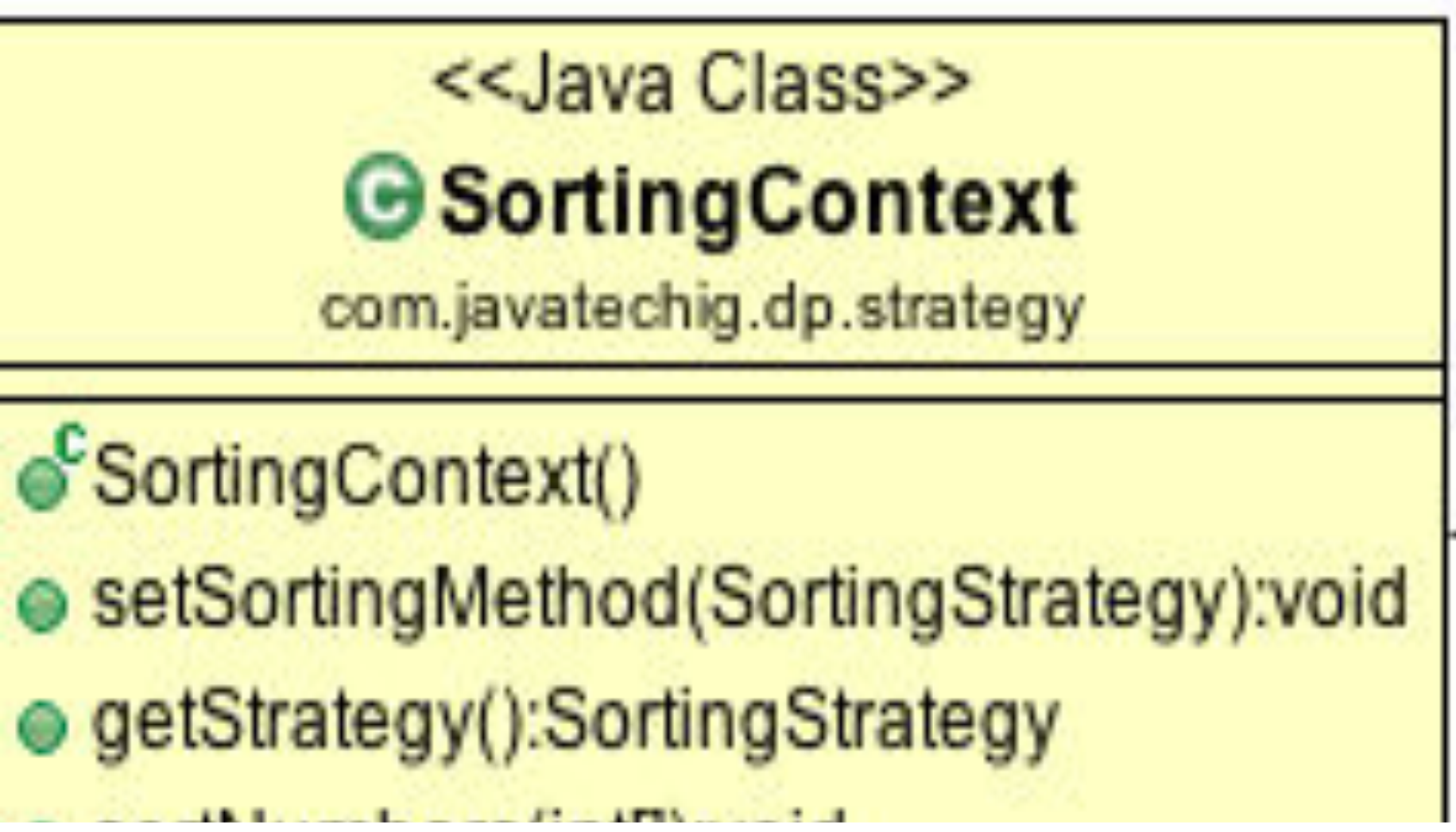


ds

7 Role Name

9 Navigability

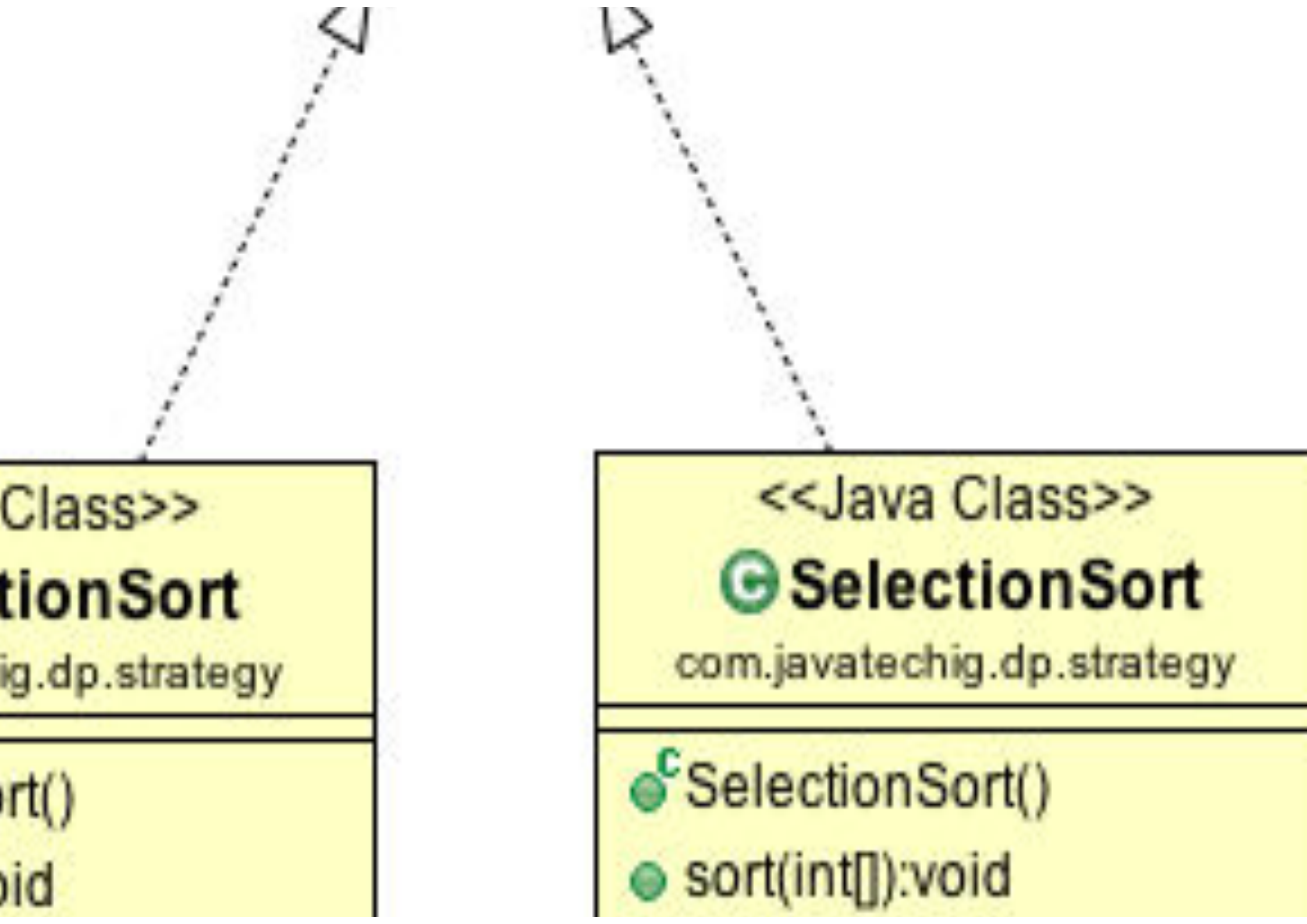
Strategy: Sorting A



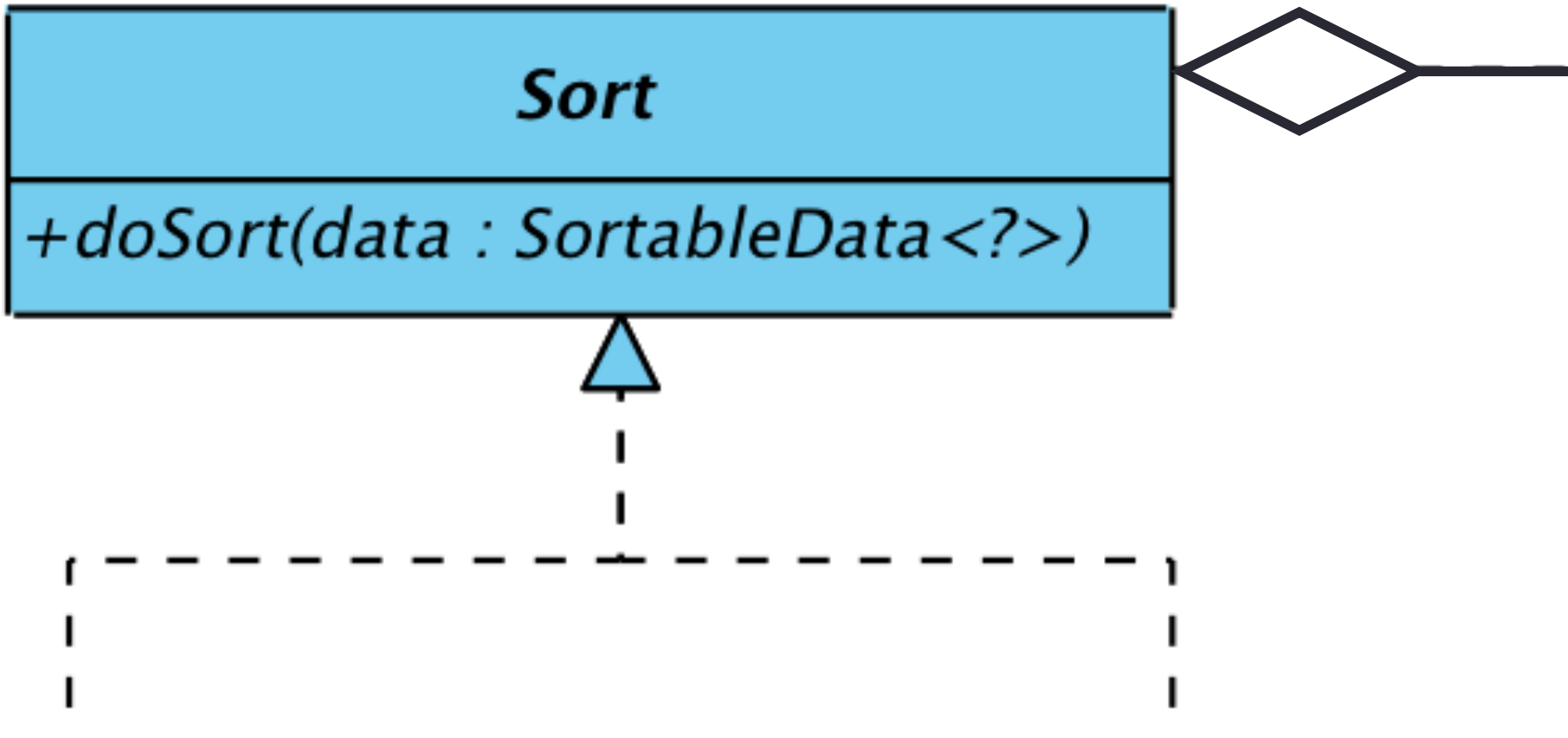
gorithms


egy
0..1
<<Java Interface>>
SortingStrategy
com.javatechig.dp.strategy
sort(int[]):void

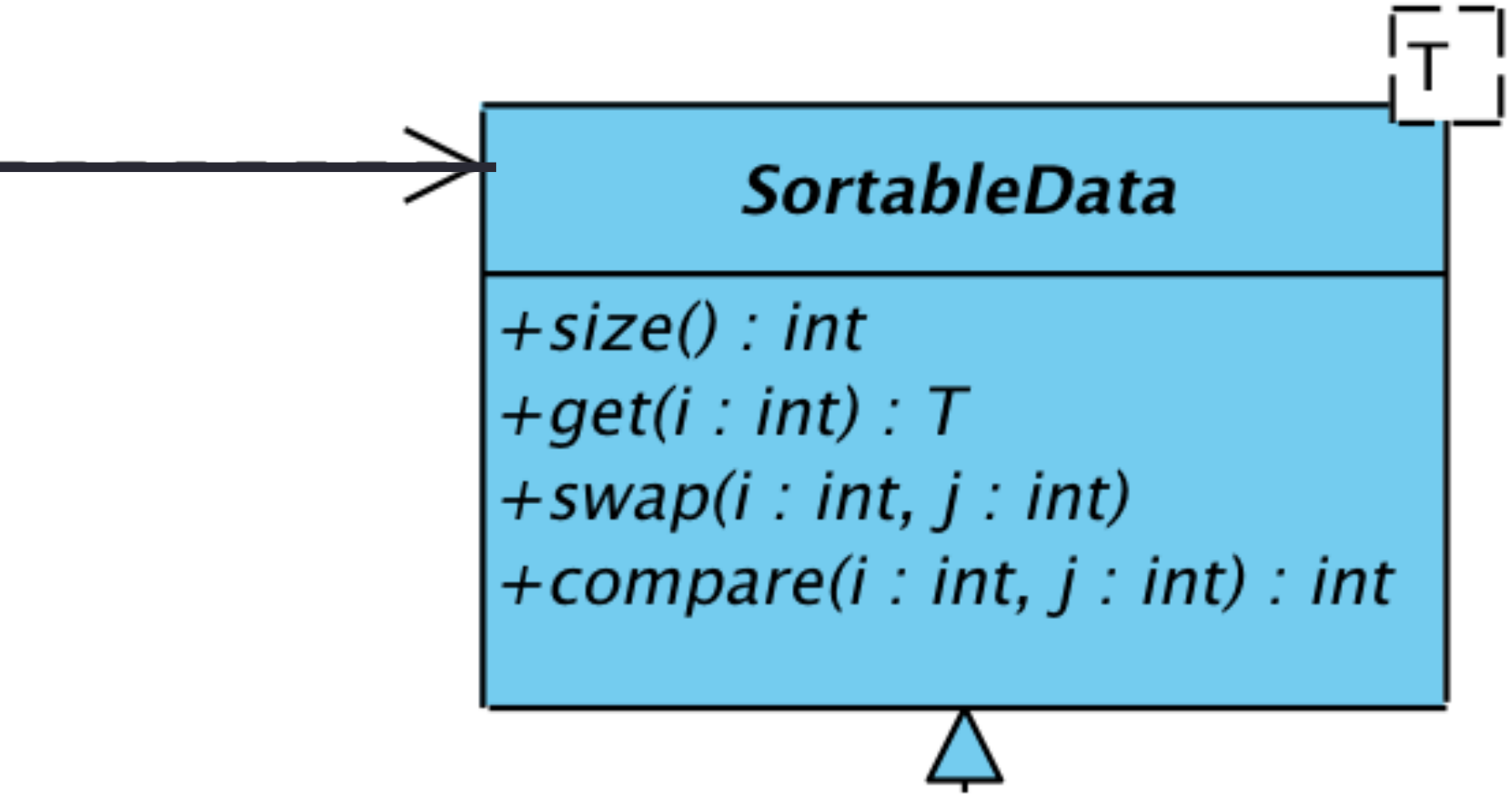

<<Jav
Ins
com.javate
Insertion
sort(int[])


Class>>
tionSort
ig.dp.strategy
rt()
oid
<<Java Class>>
SelectionSort
com.javatechig.dp.strategy
SelectionSort()
sort(int[]):void

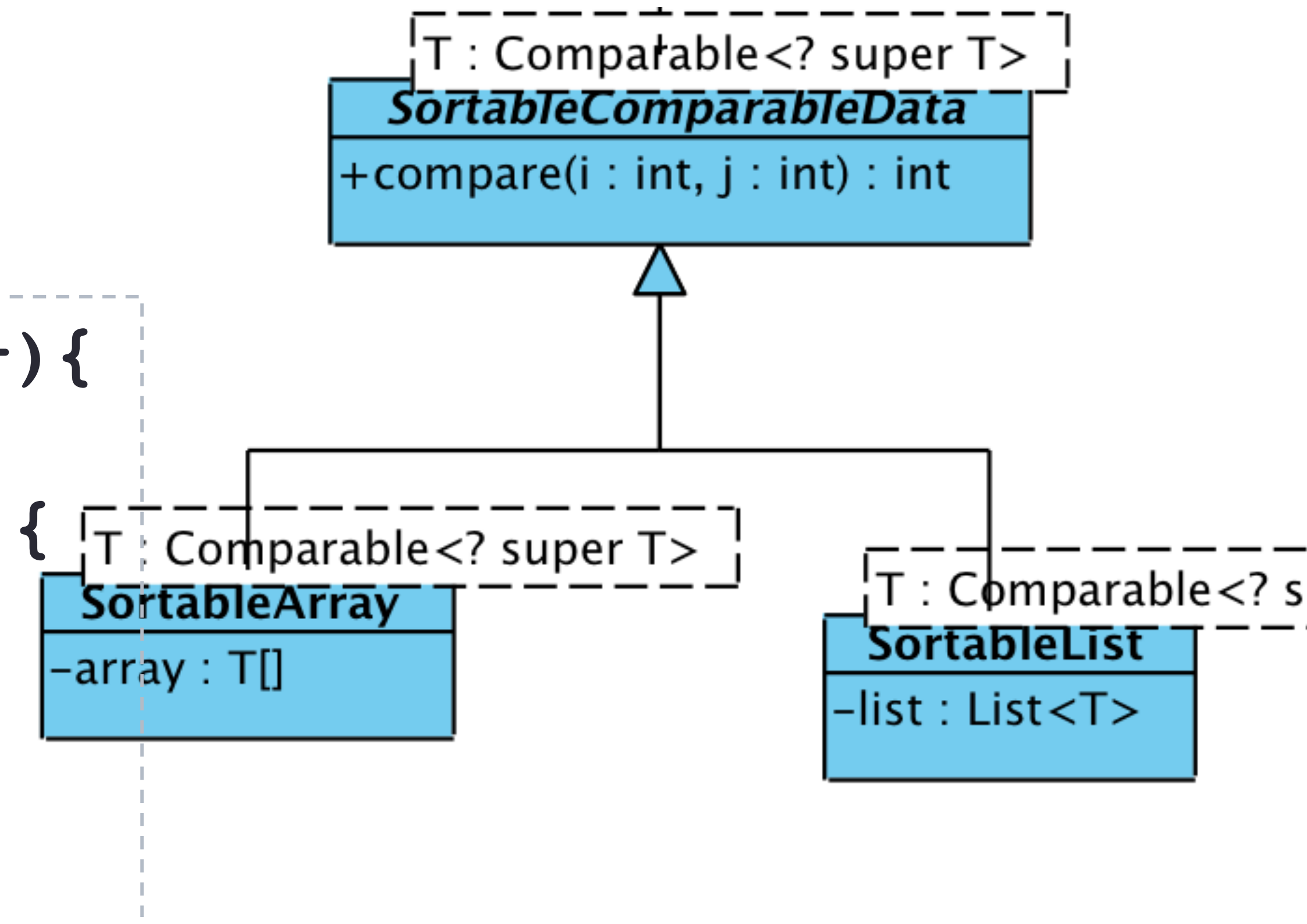
Template Method a



nd Strategy Patte




```
for (int i=0;i<data.size();i
   for (j=i;j>0;j--){
      if (data.compare(j-1,j)>0
         data.swap(j,j-1);
      }
   }
}
```


T : Comparable<? super T>
SortableComparableData
+compare(i : int, j : int) : int
T : Comparable<? super T>
SortableArray
-array : T[]
T : Comparable<? s
SortableList
-list : List<T>

Practice: Applying St

Can we apply Strateg
codebase for a speci
better design?

ategy in Codebase

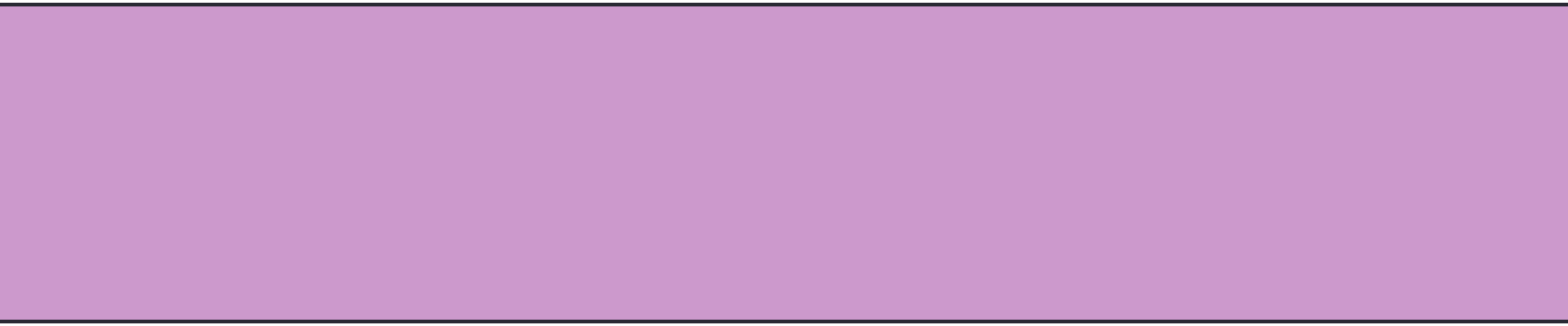
y in any part of the
c requirement/a

TIME FOR


RACTICE

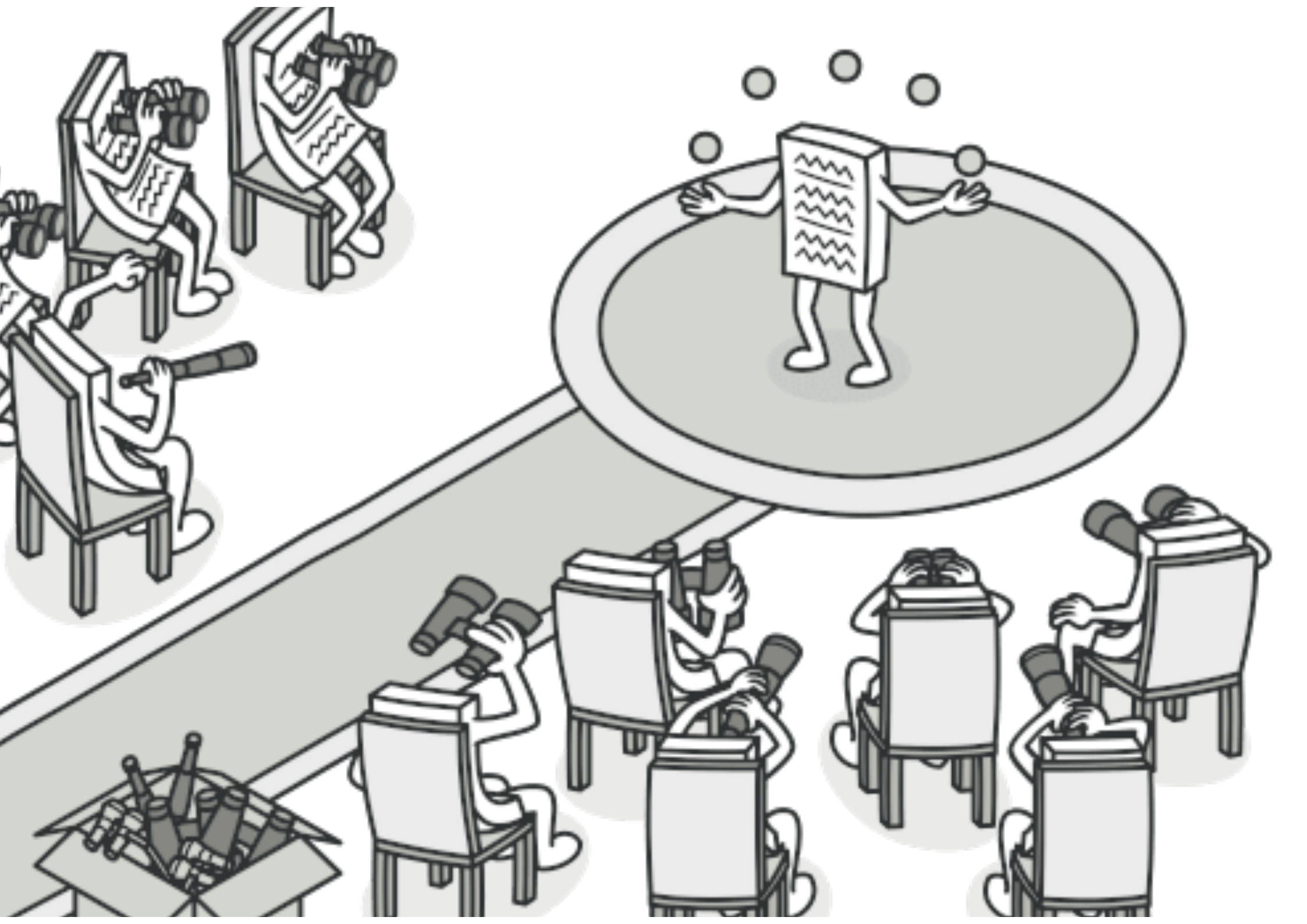
Content

1. Strategy
2. Observer



3. Adapter

4. State



New product in the

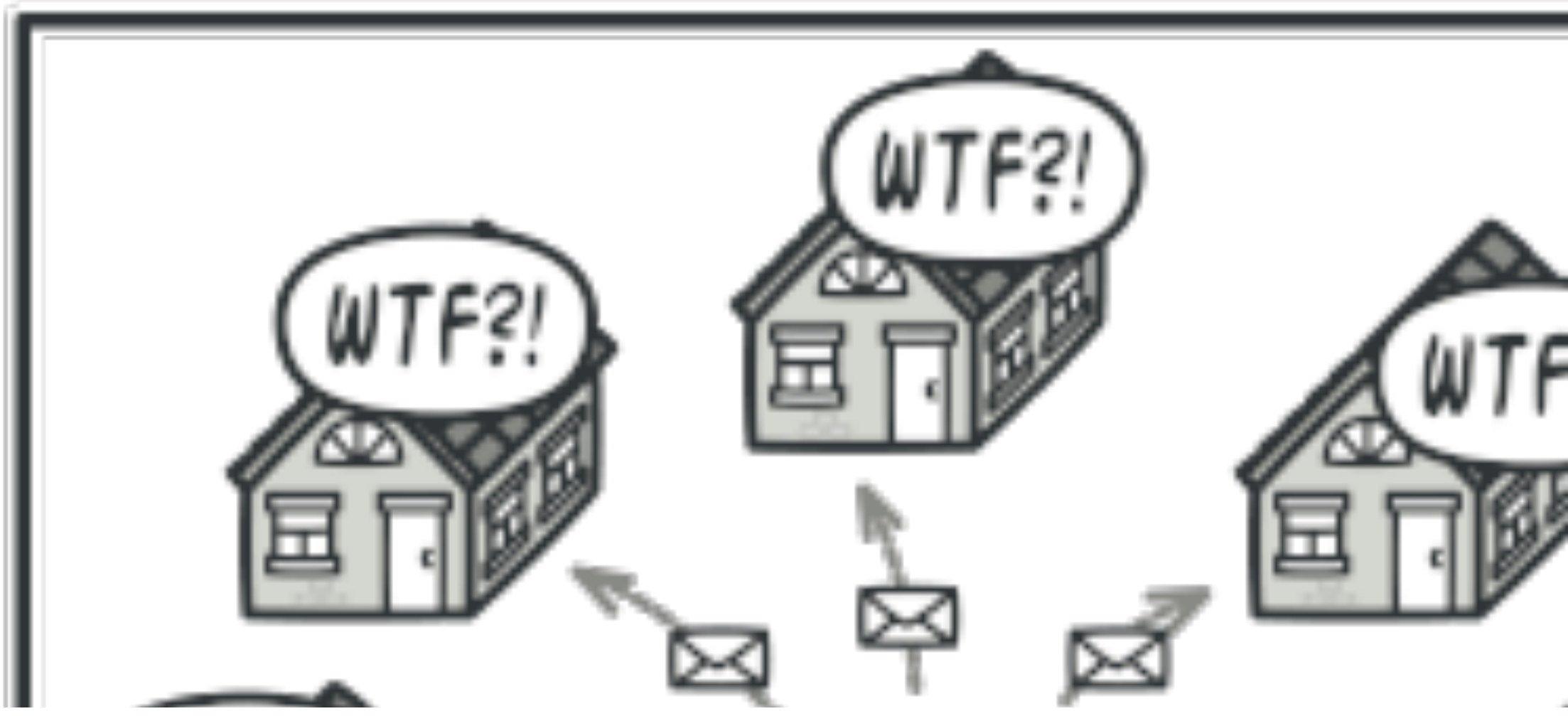
Store

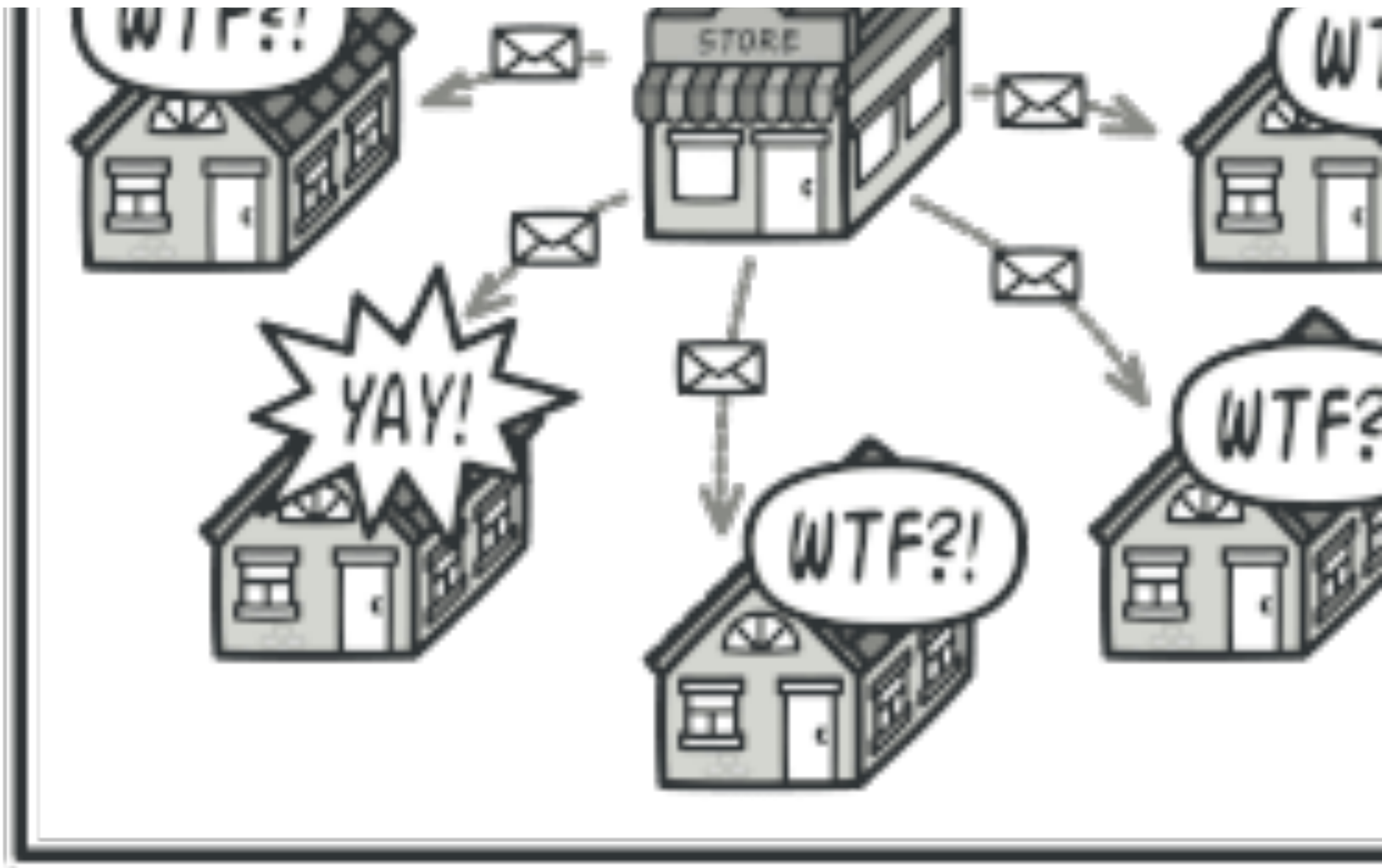
Continuously

checking if the store
has new products o
not

New product in the S

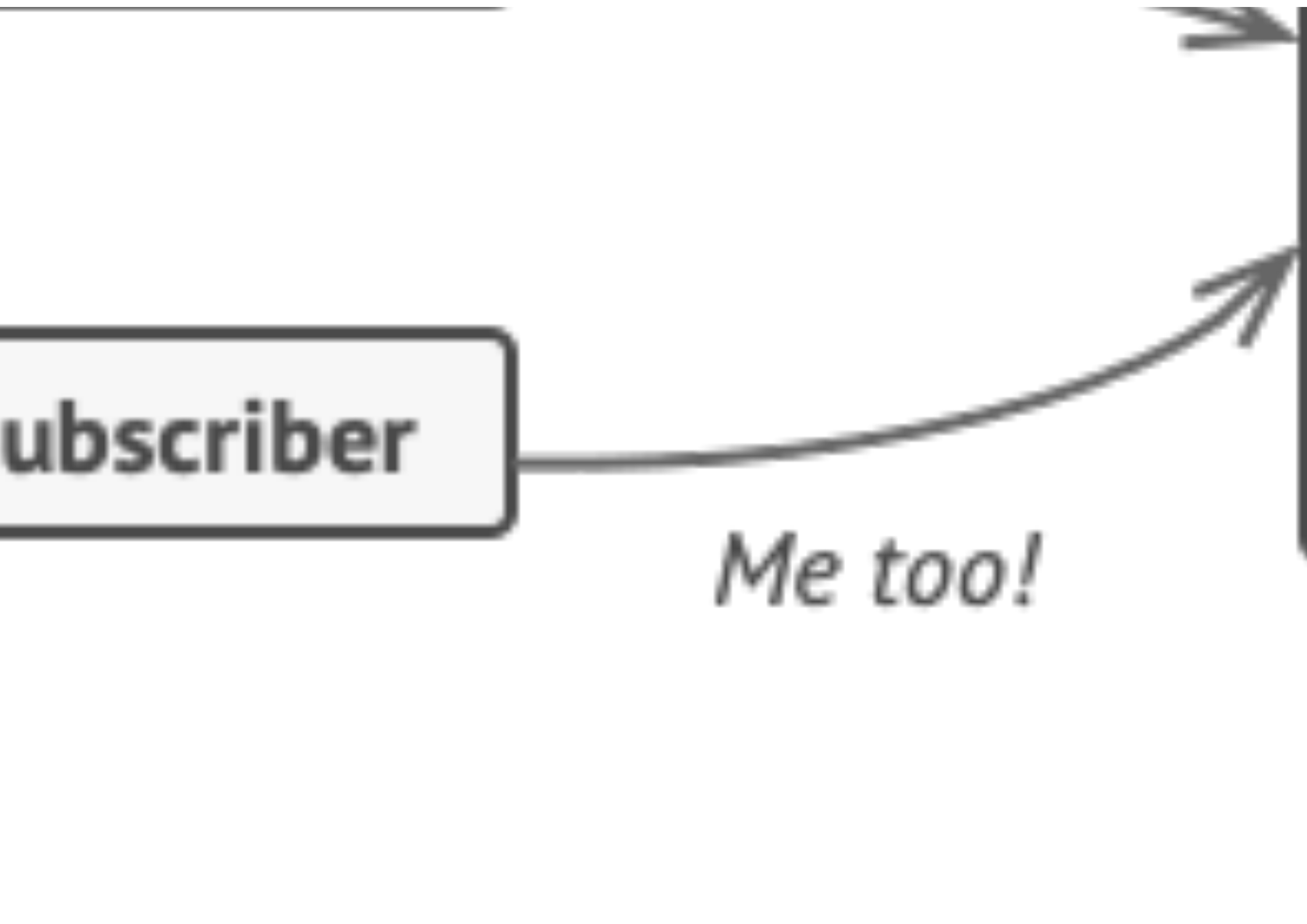
ore – Send to all

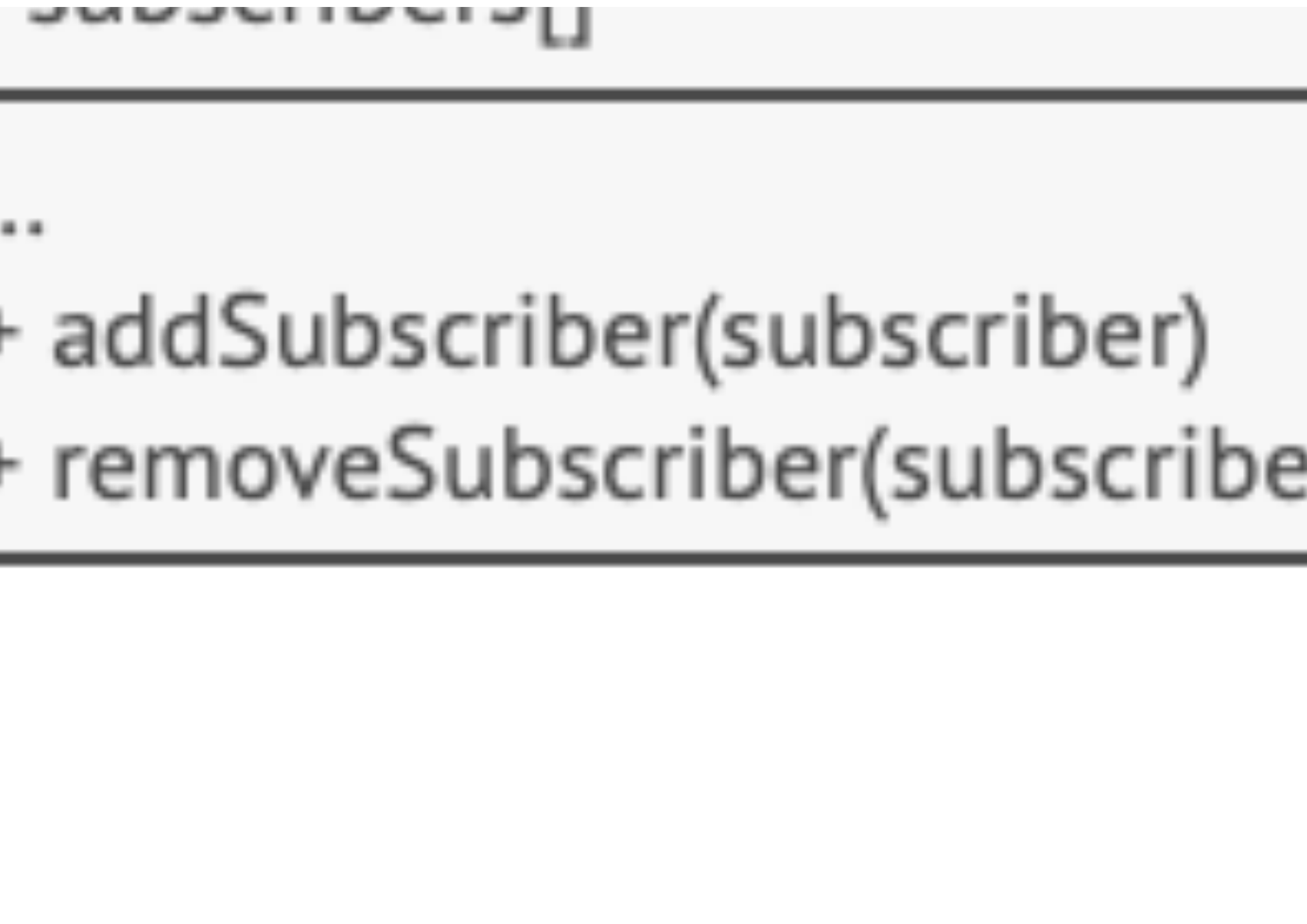



WTF?!
STORE
YAY!
WTF?!
WTF?

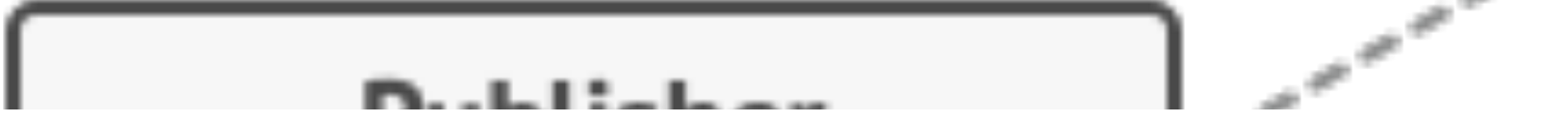
Observer: Subscrip

tion Model


ubscriber
Me too!

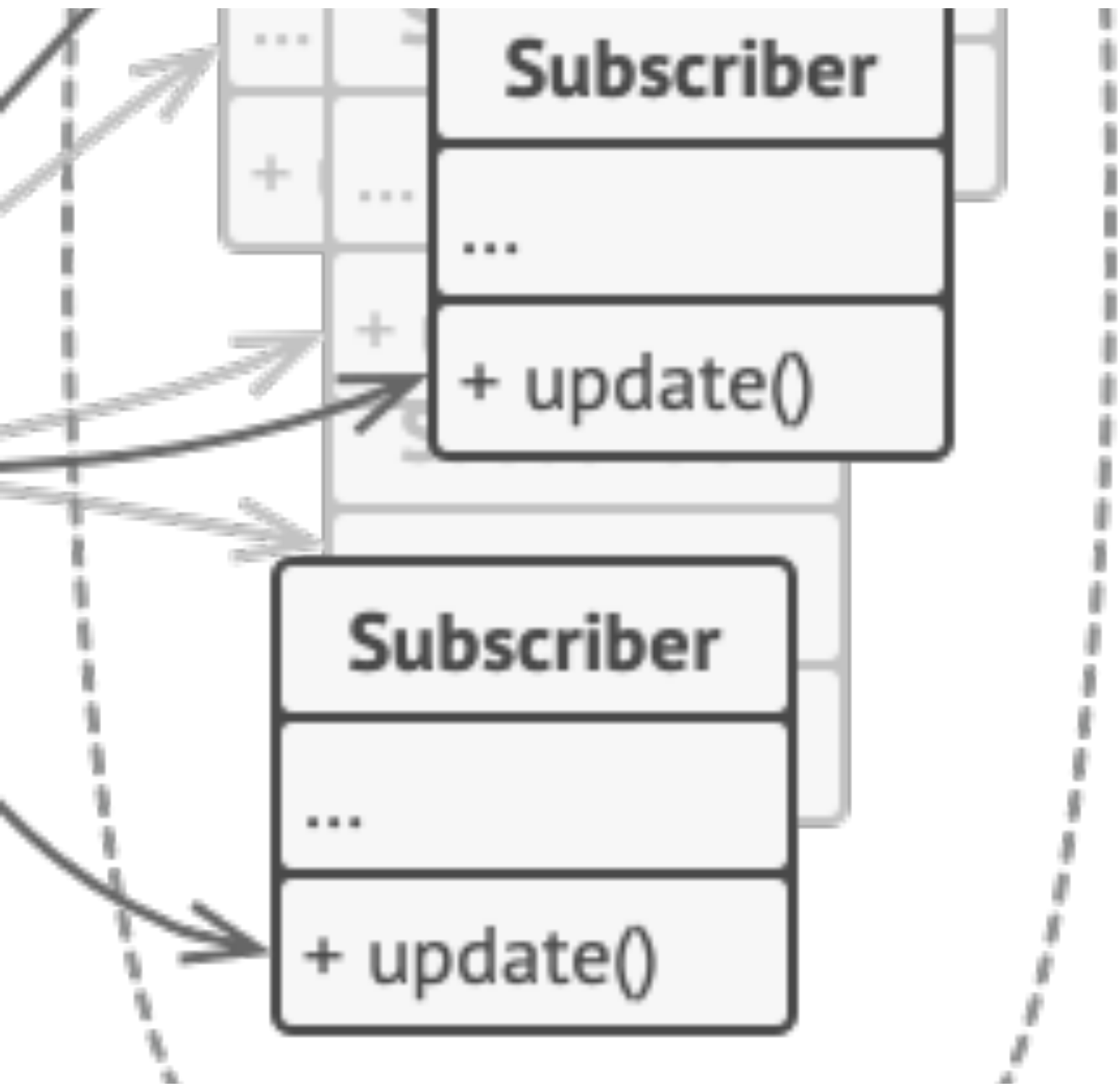

subscribers[]
...
+ addSubscriber(subscriber)
+ removeSubscriber(subscribe

Observer: Notificat



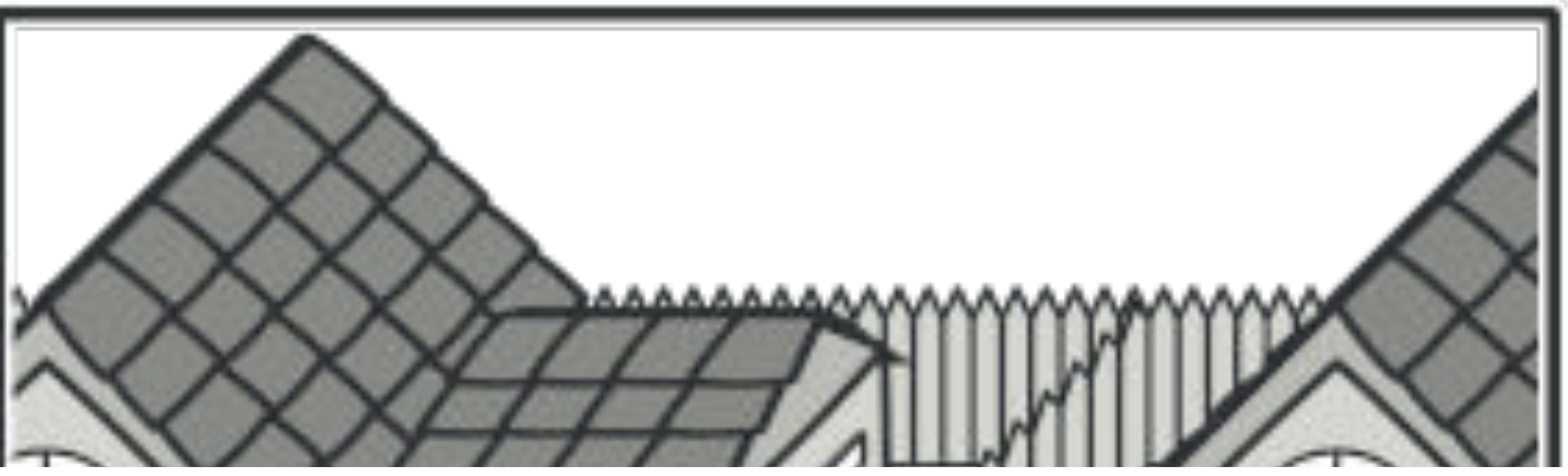
on to Subscribers


- subscribers[]
...
notifySubscribers()
Guys, I just want
to let you know that
something has just
happened to me.

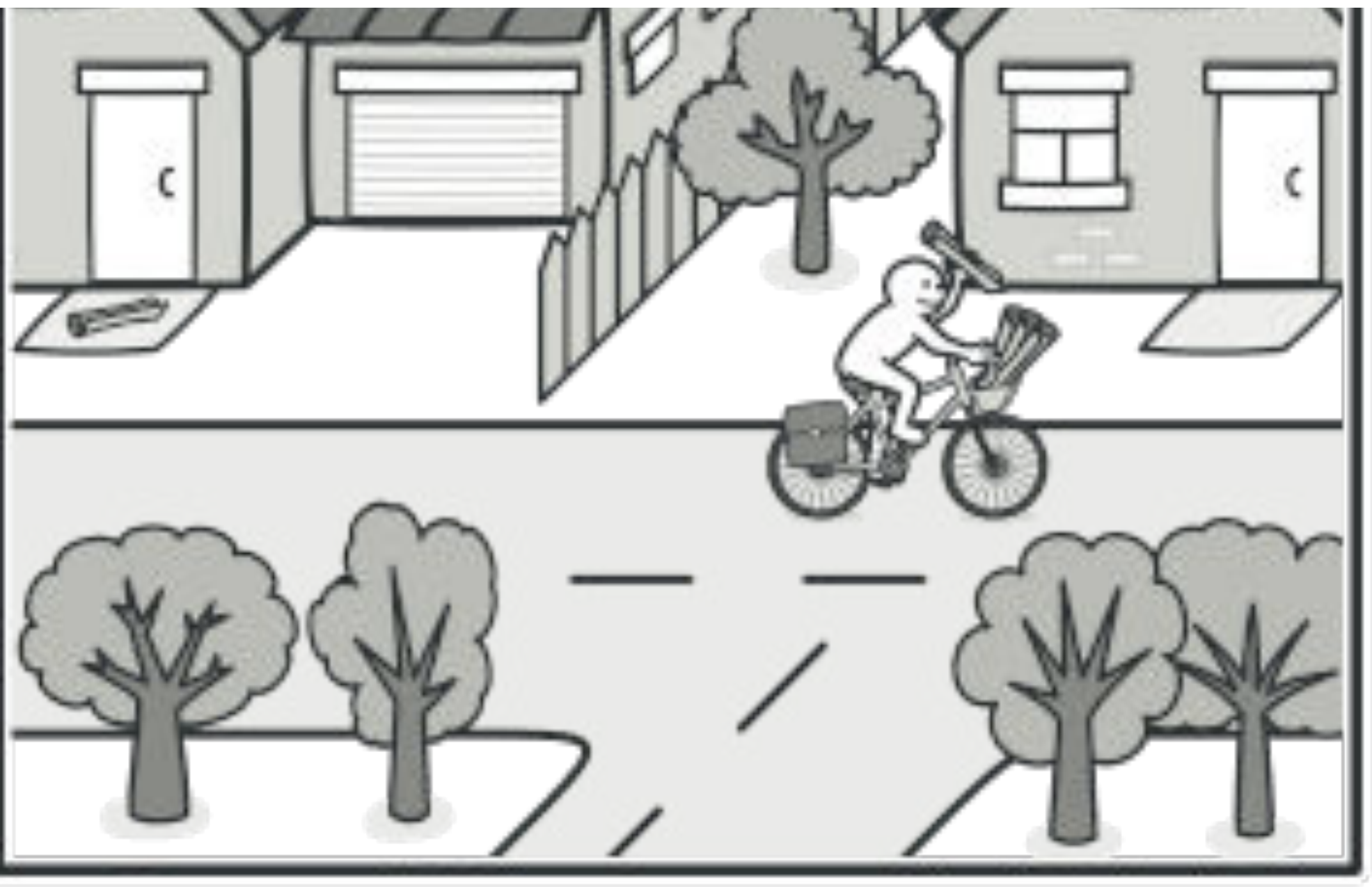

Subscriber
...
+ update()
Subscriber
...
+ update()

Magazine Subscrip

tion Model

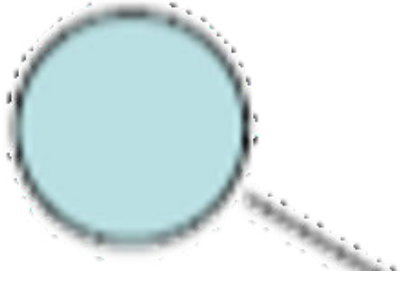



THANKS!
YOU ARE
SUBSCRIBED!



The Weather-O-Ram

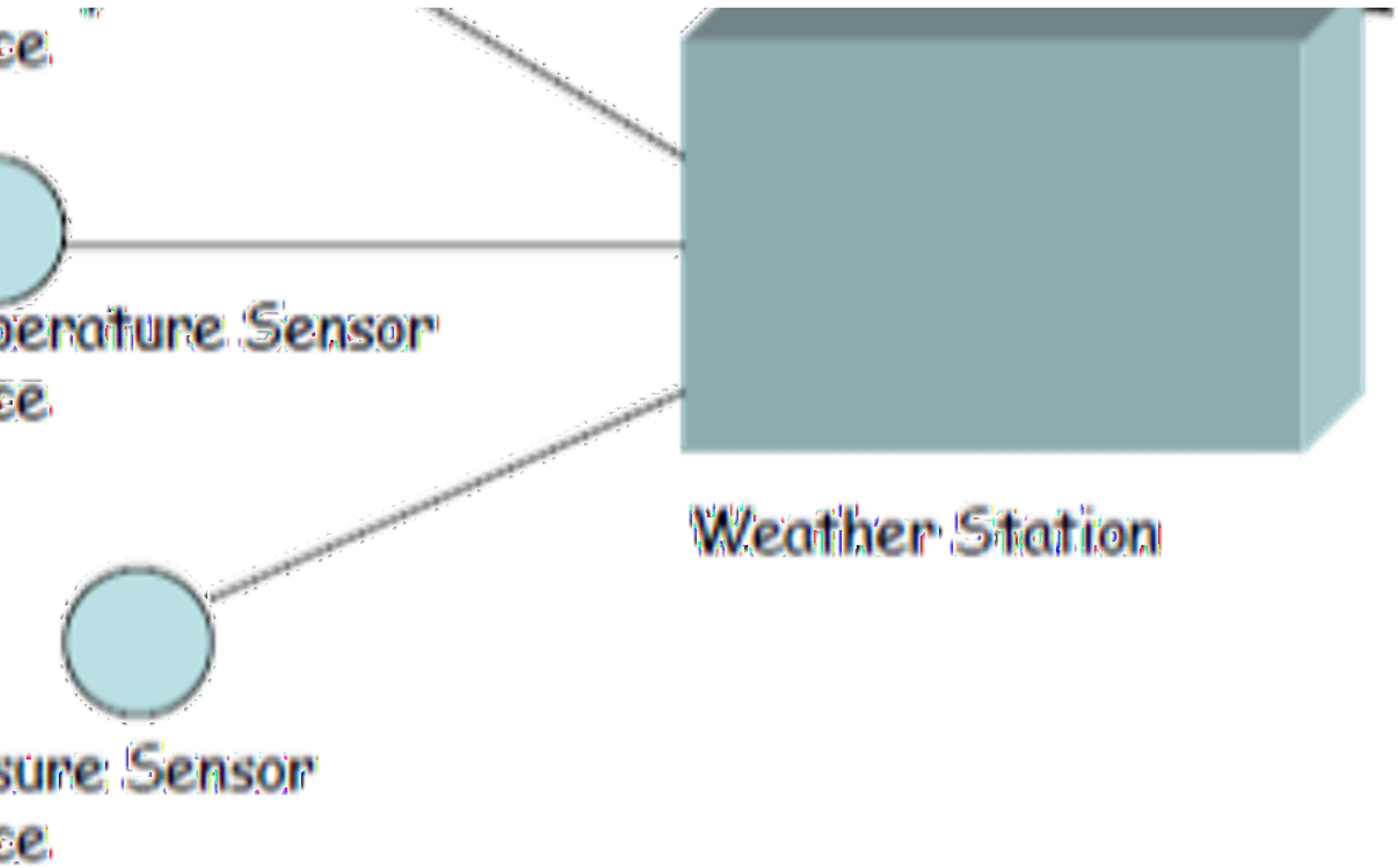
WeatherData has updat
conditions, weather stat

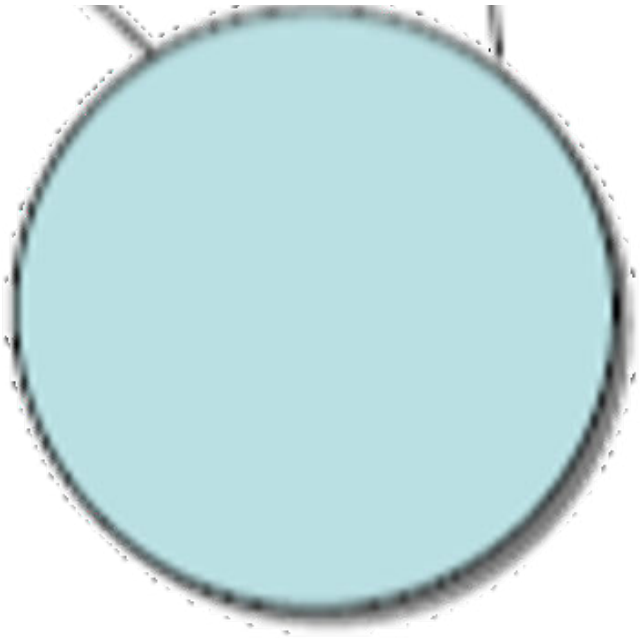


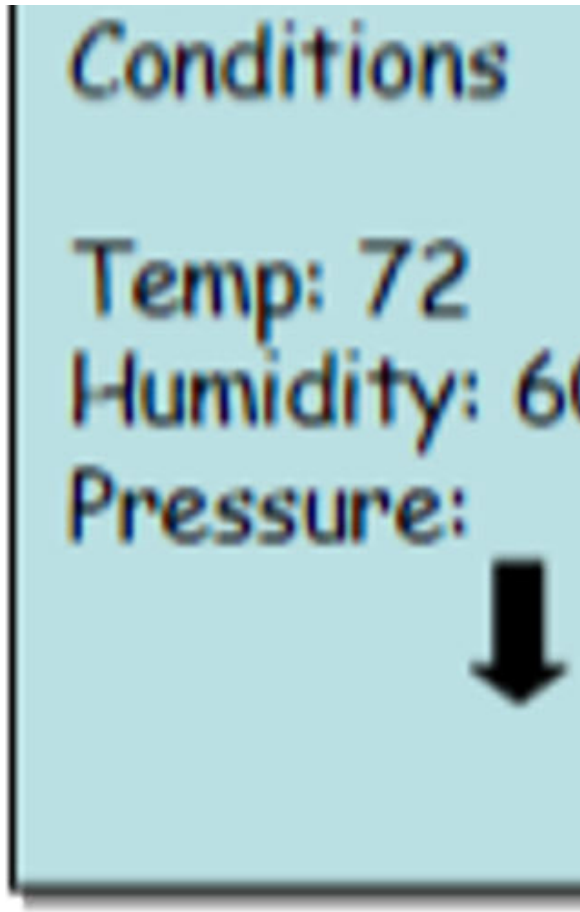
Pu

a: More Example

e three displays for curre
, and a forecast

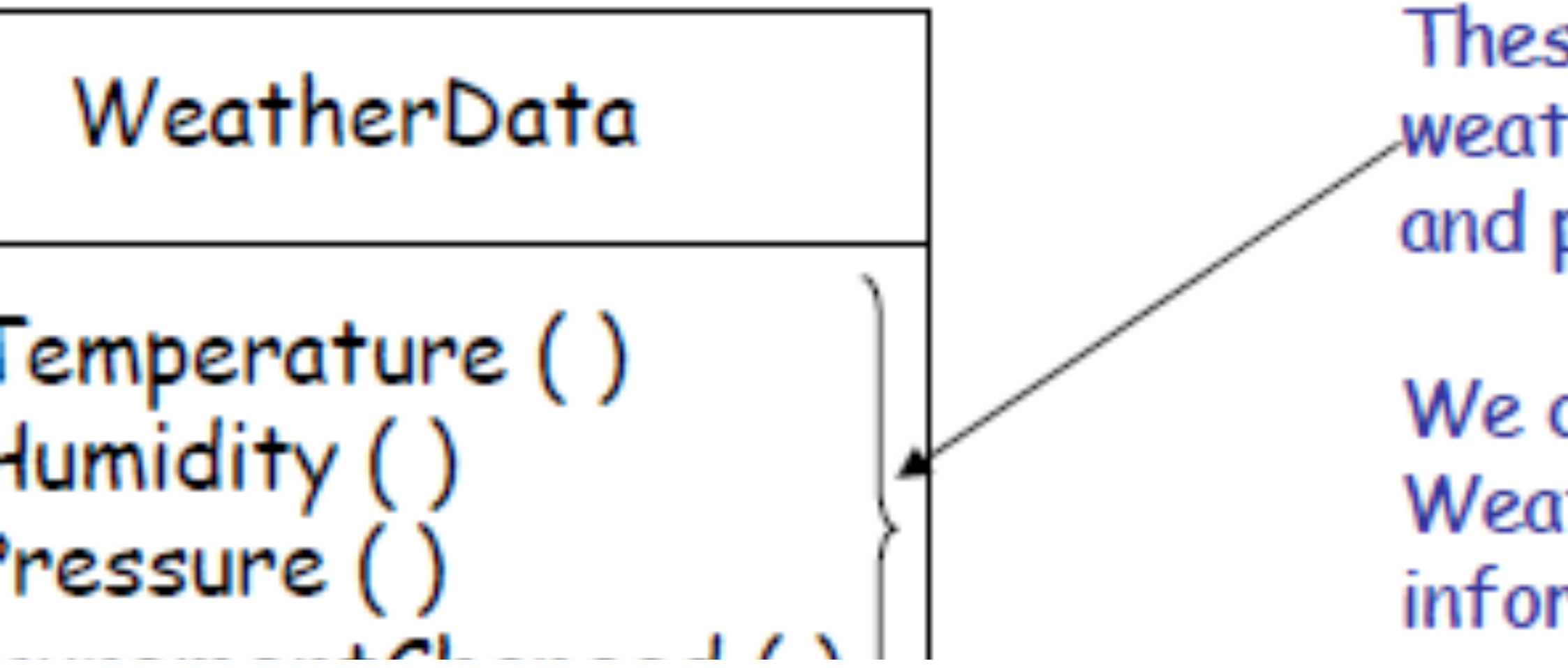

ce
perature Sensor
ce
Weather Station
sure Sensor
ce





Display Devi

WeatherData Clas



s: Idea

three methods return the most recent
er measurements for temperature, humidity
essure respectively.

n't care HOW these variables are set; the
erData object knows how to get updated
ation from the Weather Station

ther methods

A clue: what we need to add!

```
/*
 * This m
 * measu
 * /
public vo
        // Y
}
```

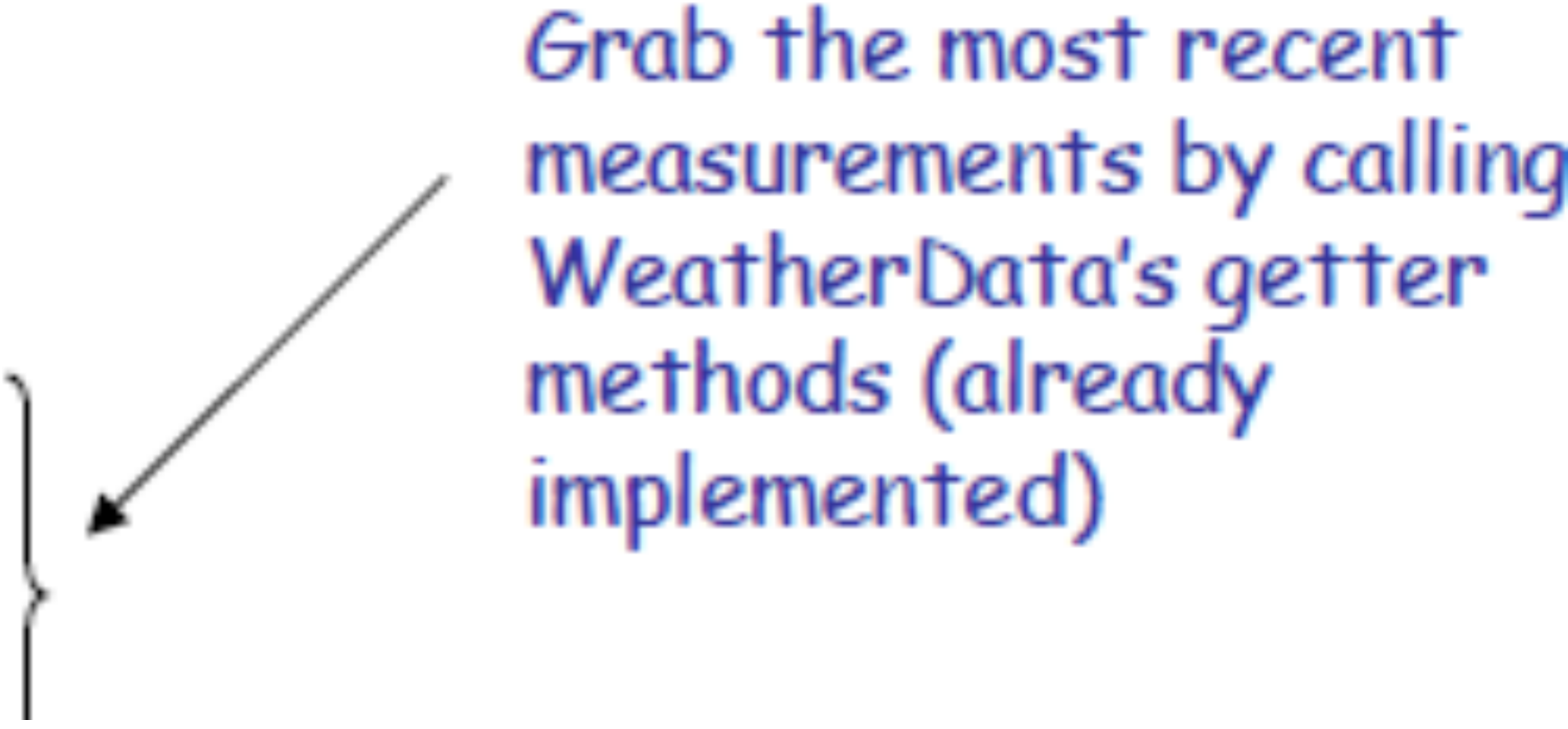
thod gets called whenever the
ements have been updated.

l measurementsChanged ( ){
ur code goes here

WeatherData Clas

```
public class WeatherData {
    // instance variable declarations
    public void measurementsChanged ( )
        float temp = getTemperature ( );
        float humidity = getHumidity ( );
         float pressure = getPressure ( );
```

s: First version



```
        currentConditionsDisplay.update (t
        statisticsDisplay.update (temp, hun
        forecastDisplay.update (temp, hum
    }
    // other WeatherData methods here
```

mp, humidity, pressure);
dity, pressure);
ity, pressure);

Now update the displays.

Call each display element to update its display, passing it th

What's Wrong with

```
public class WeatherData {
    // instance variable declaration
    public void measurementsCha
        float temp = getTemperatu
        float humidity = getHumidit
        float pressure = getPressu
```

the First version?

```
ged ( ) {
 ( );
( );
e ( );
```

Area of change, we n
encapsulate this.

currentConditionsDisplay.

statisticsDisplay.update (te

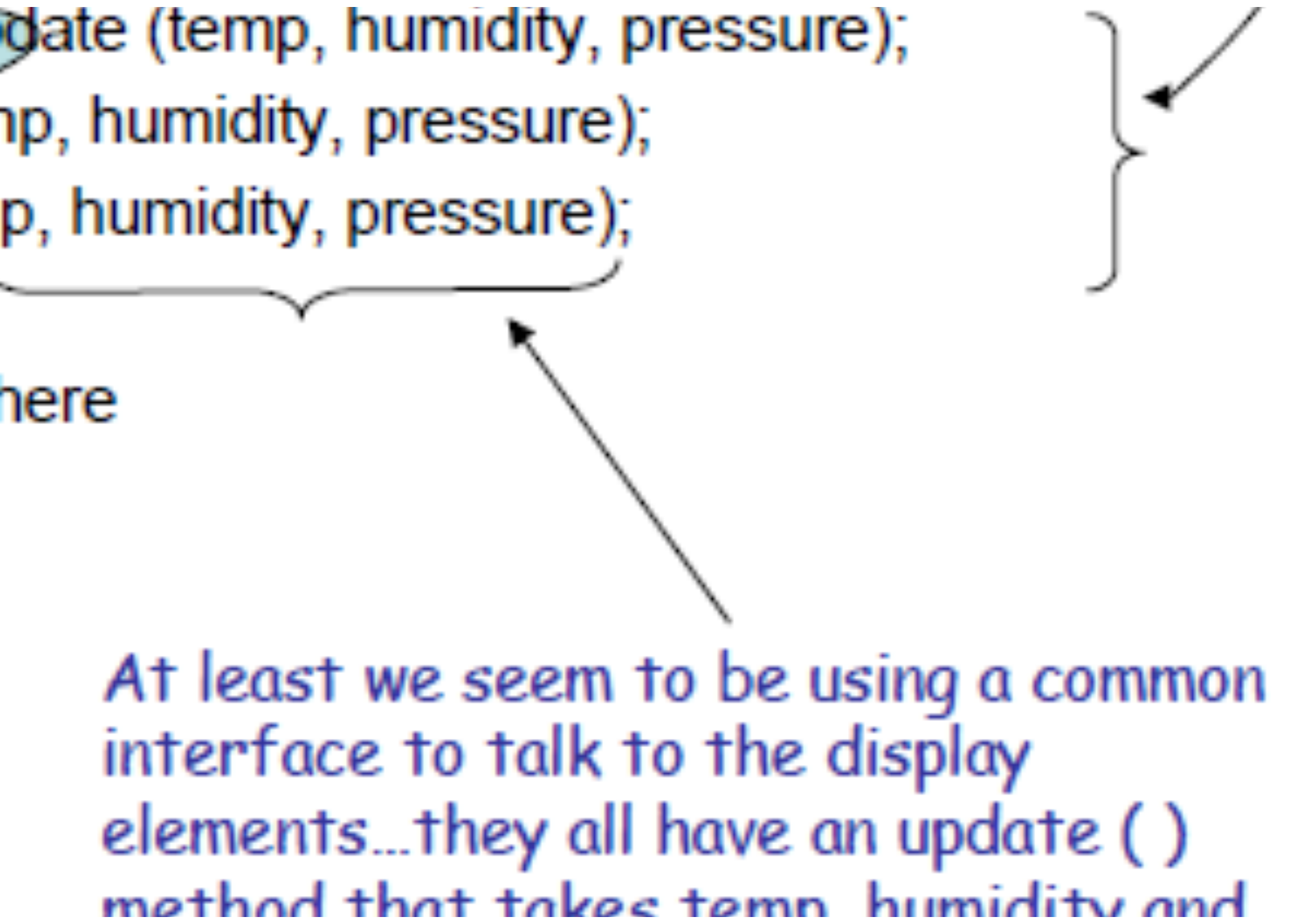
forecastDisplay.update (te

}

// other WeatherData method

}

ling to concrete implementations we
o way to add or remove other
y elements without making changes
program.


date (temp, humidity, pressure);
np, humidity, pressure);
p, humidity, pressure);
here
At least we seem to be using a common interface to talk to the display elements...they all have an update ( )

ublishers + Subscrib

Publisher = Subject

Subscribers = Observers

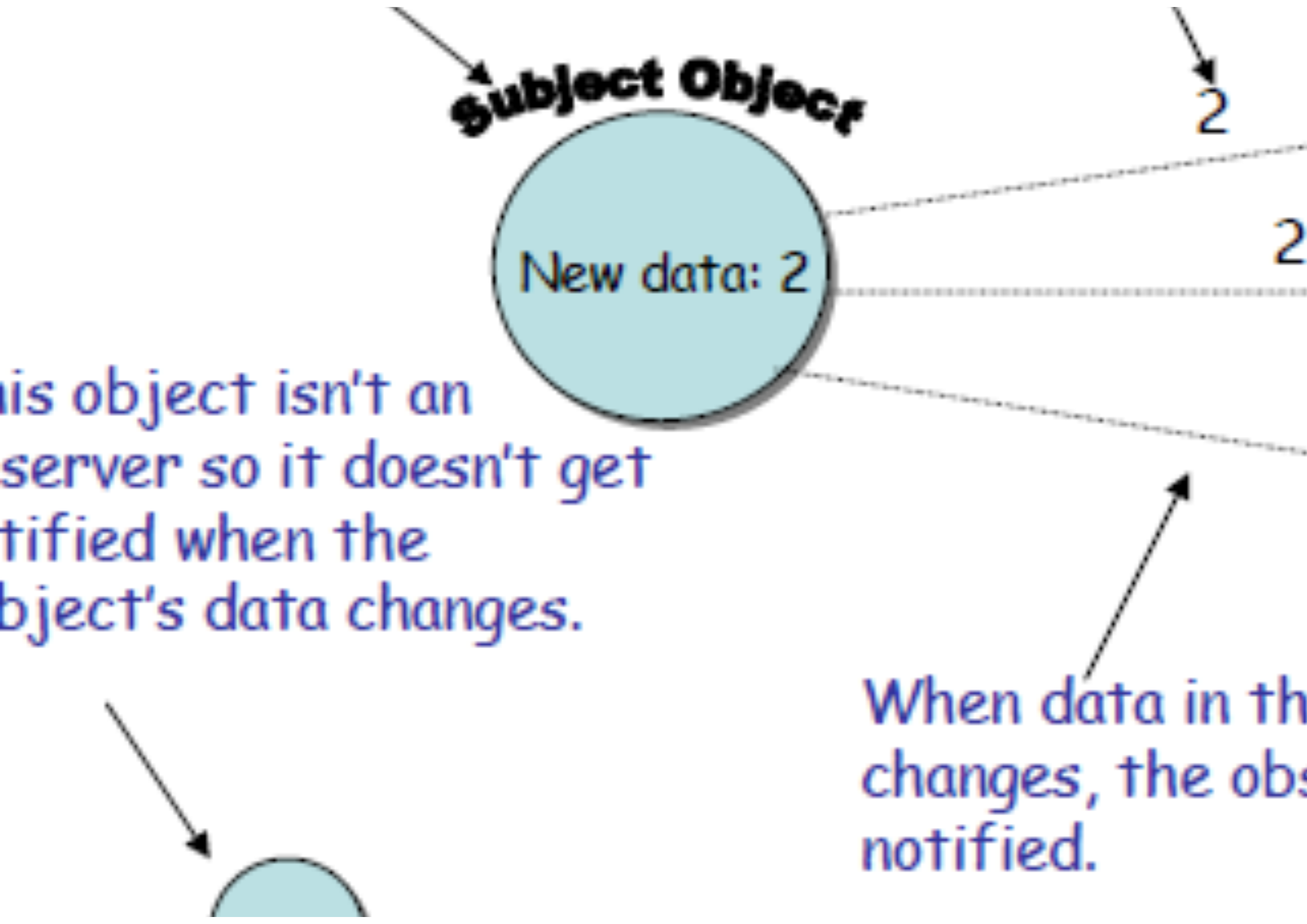
bject object

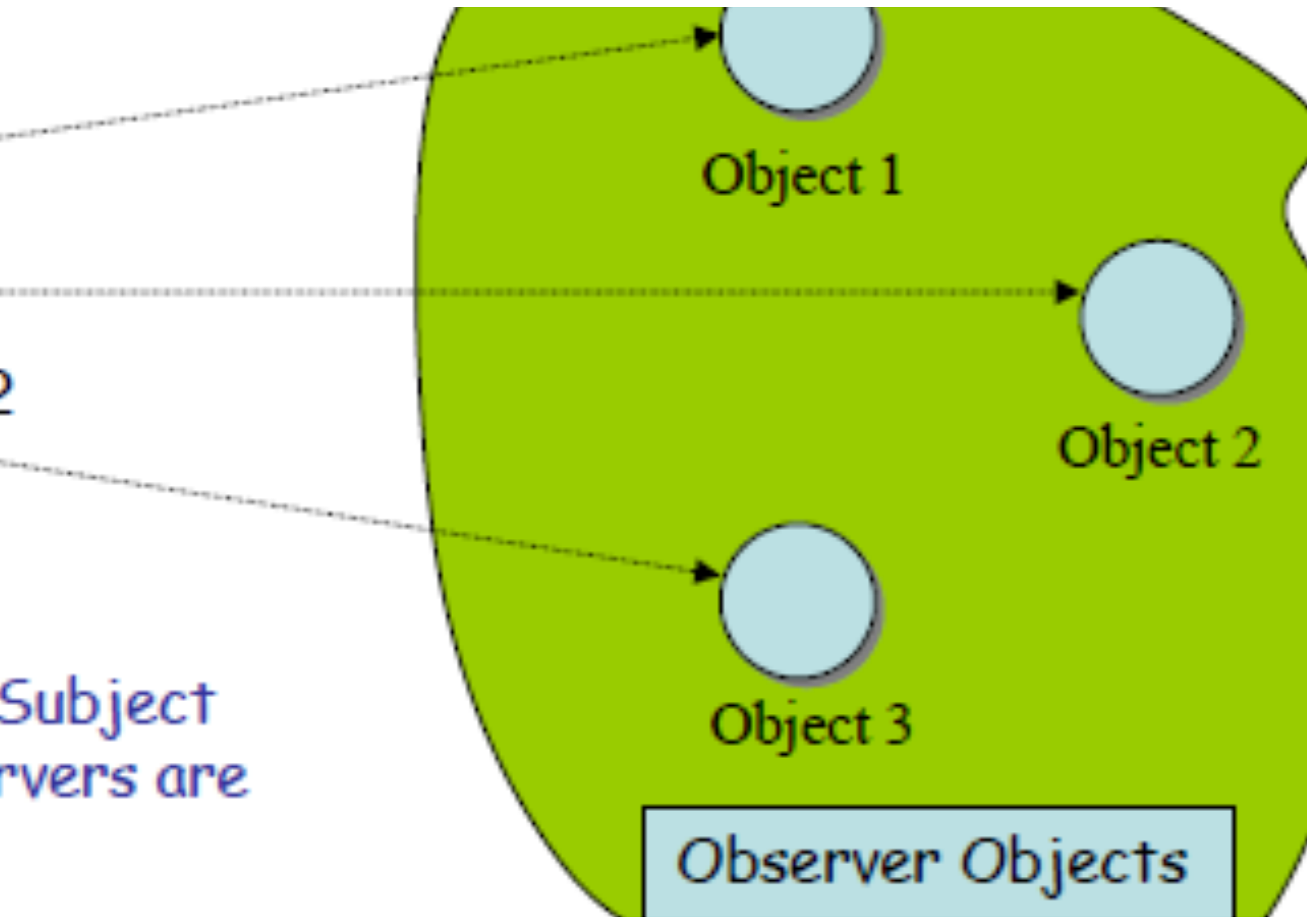
New data values are comm
observers in some form w

ers = Observer Patte

The observers have subscribed to
Subject to receive updates when
subject's data changes.

nicated to the
en they change

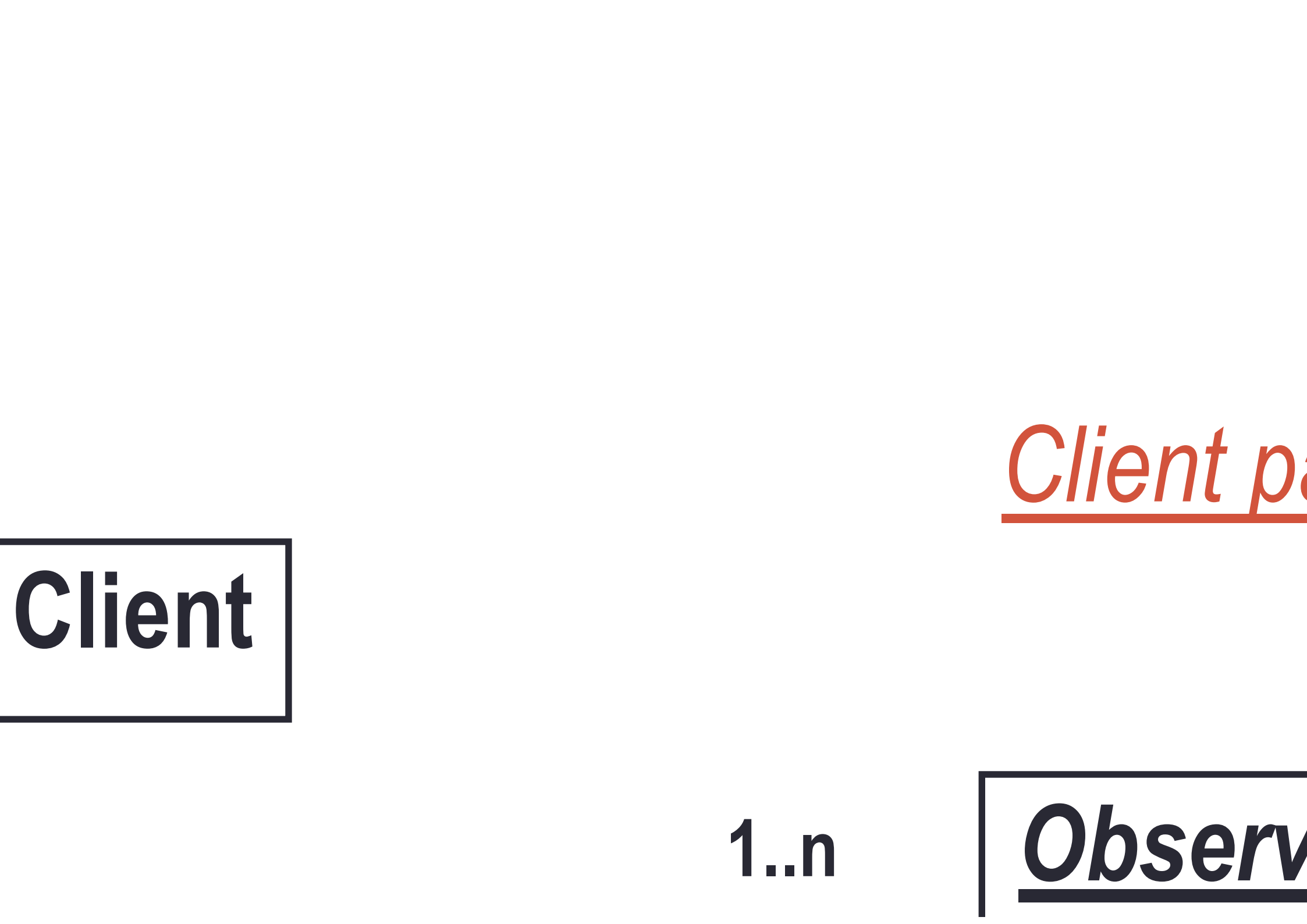

Subject Object
New data: 2
2
2
is object isn't an
server so it doesn't get
tified when the
bject's data changes.
When data in th
changes, the obs
notified.

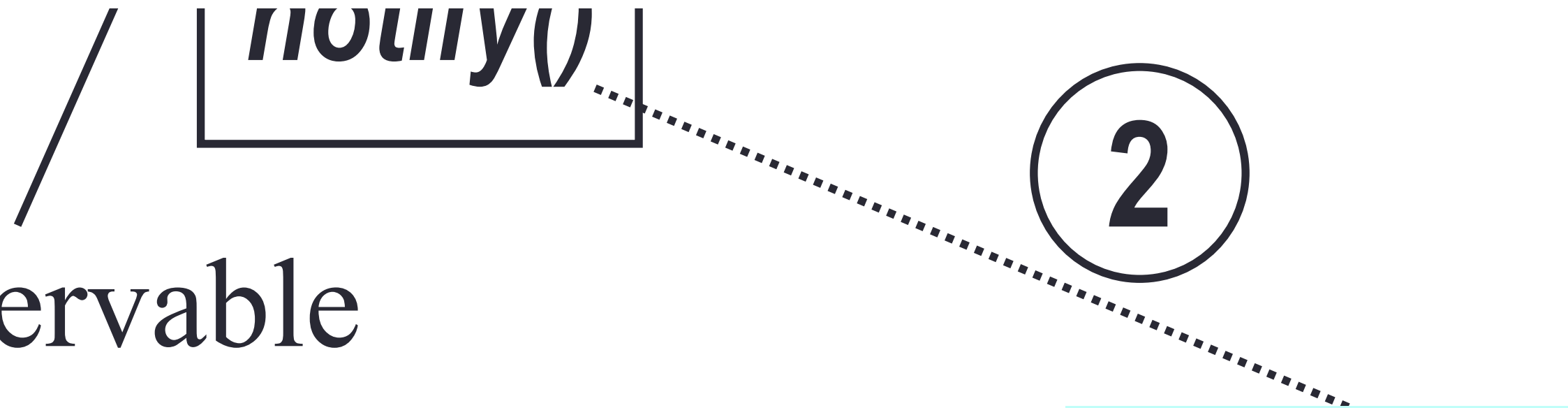


2

Subject
rvers are

erver part


Client pa
Client
1..n
Observ

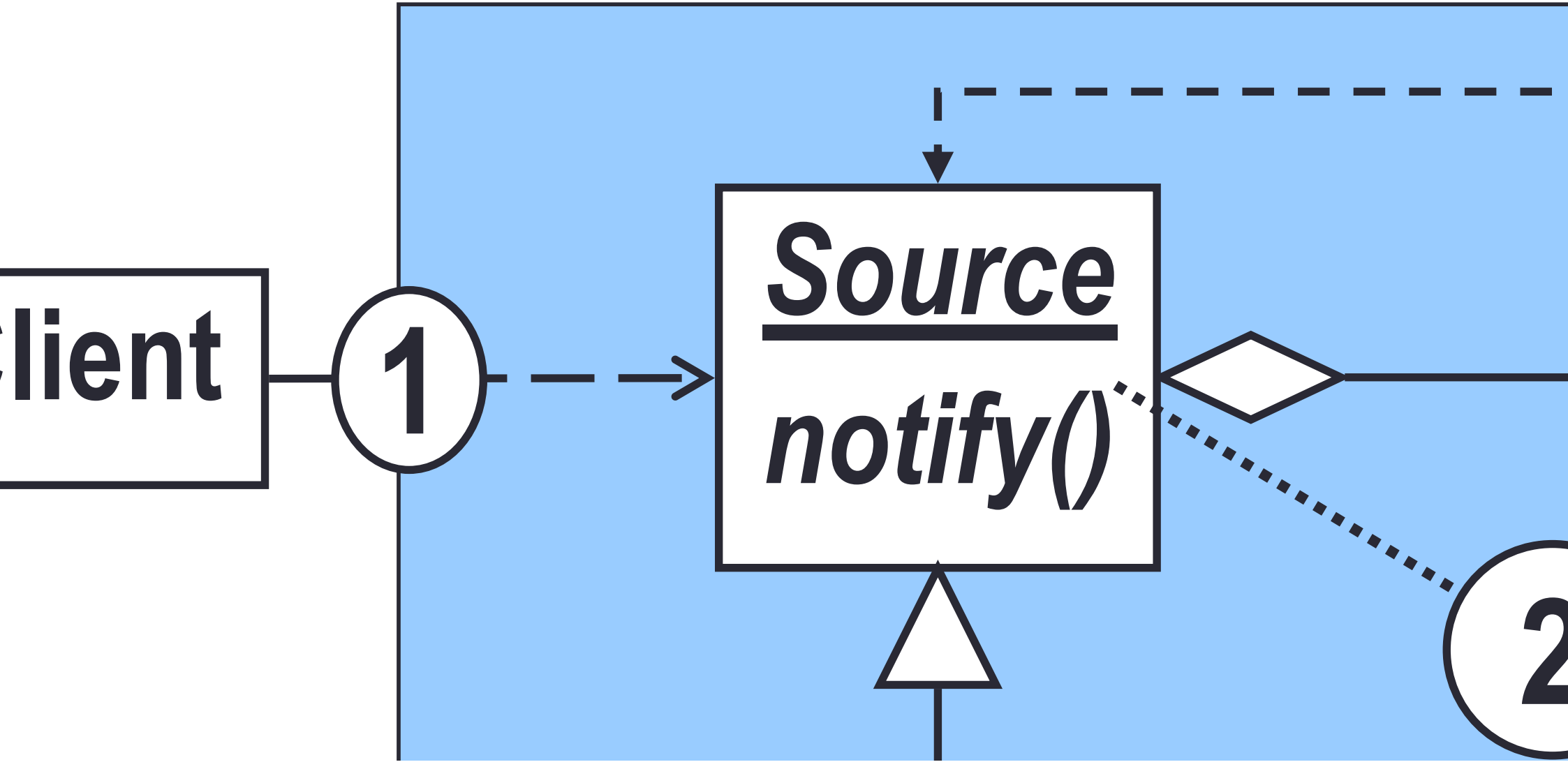

notify()
2
ervable
for all Ob
o.

update

erver's o:
pdate();

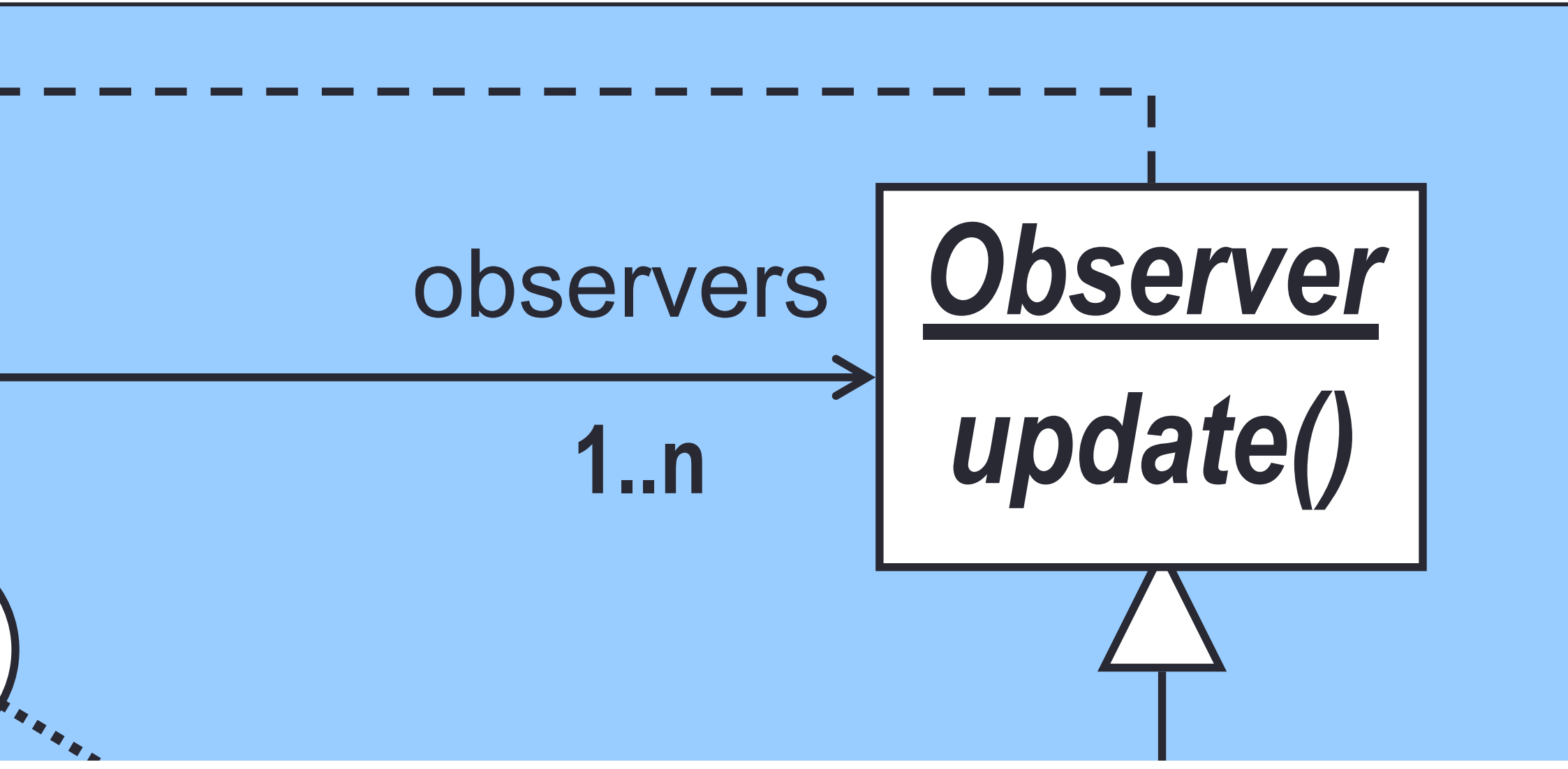
erver part

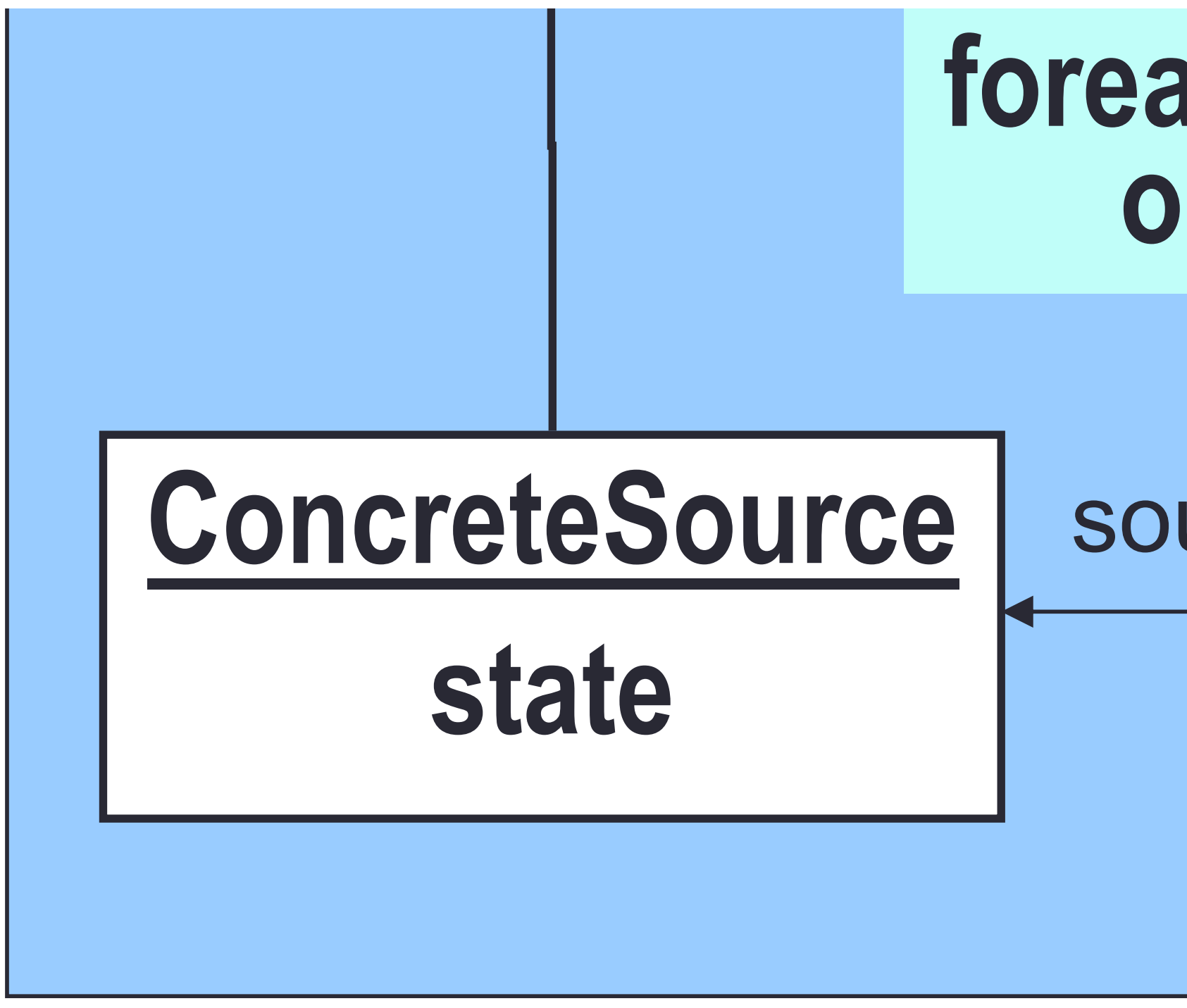
Observer

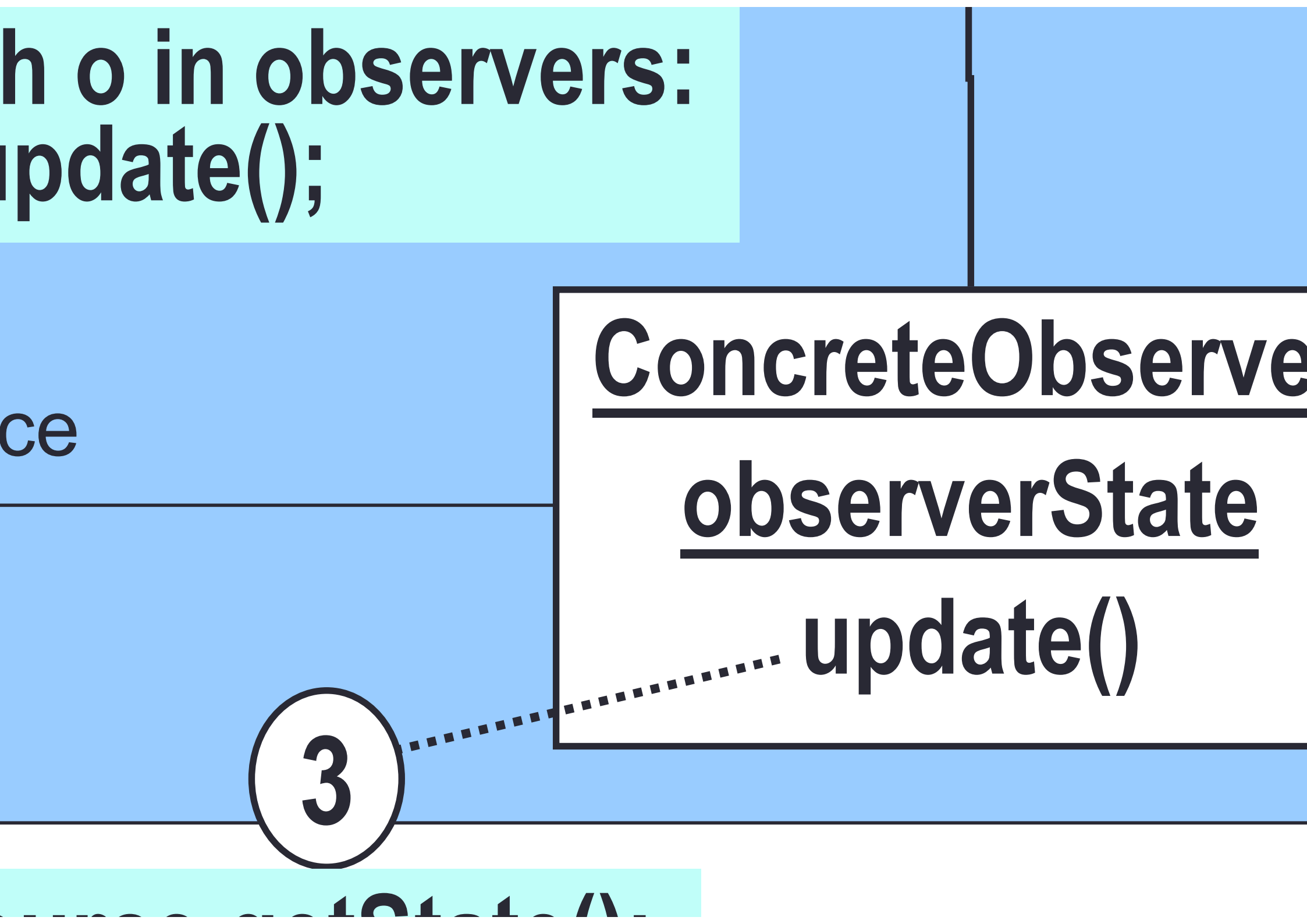


Structure

Client pa




forea
o
ConcreteSource
state
sou


h o in observers:
pdate();
ce
ConcreteObserve
observerState
update()
3

Observer Pattern:

To keep a set of objects of a designated object

- one-to-many dependency one object (subjet) change

ntent

up to date with the state

between objects so that when
s state, all of its dependents

(observers) are notified a

Some observers may ob
subject (many-to-many
The update should spec
changed

d updated automatically

serve more than one
elation)
fy which subject

Design Choices

When to notify Obser

- Automatic on each cha
- Triggered by client

vers:

nge

How to communicate change:

- Push: subject gives de
- Pull: subject notifies o
  occurred, observer qu

information about the

ails to observers

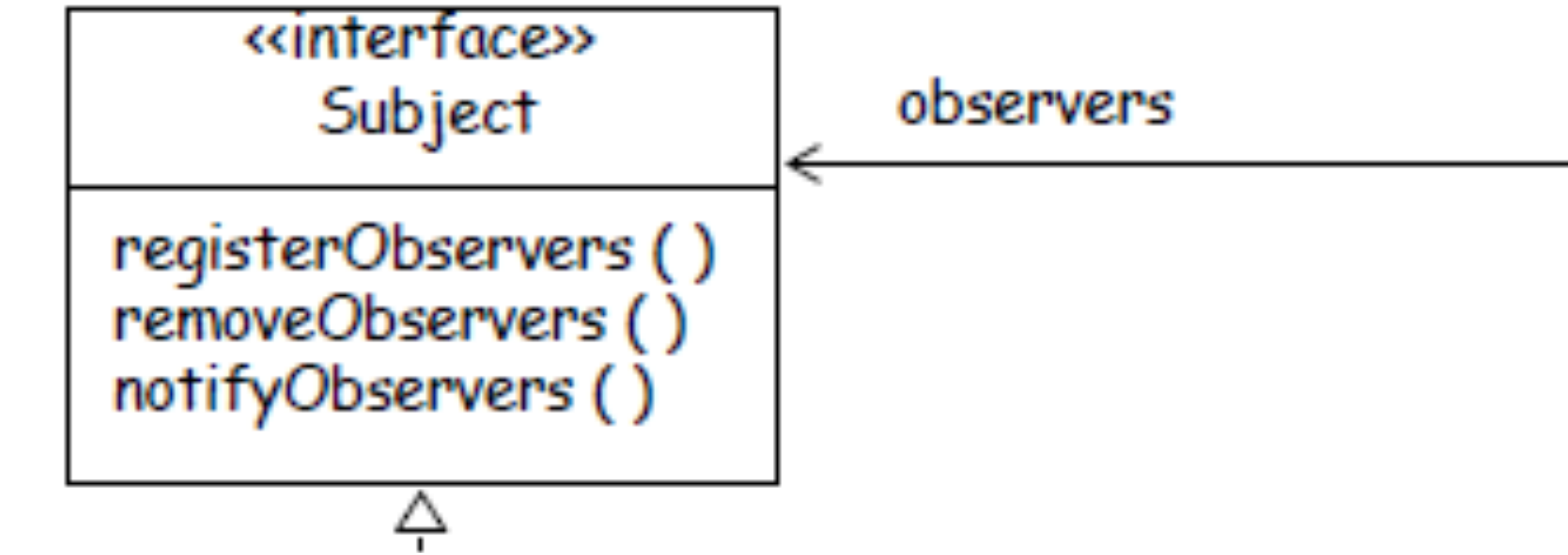
ly that a change has

ries subject about detail

Designing the Weath

ct interface

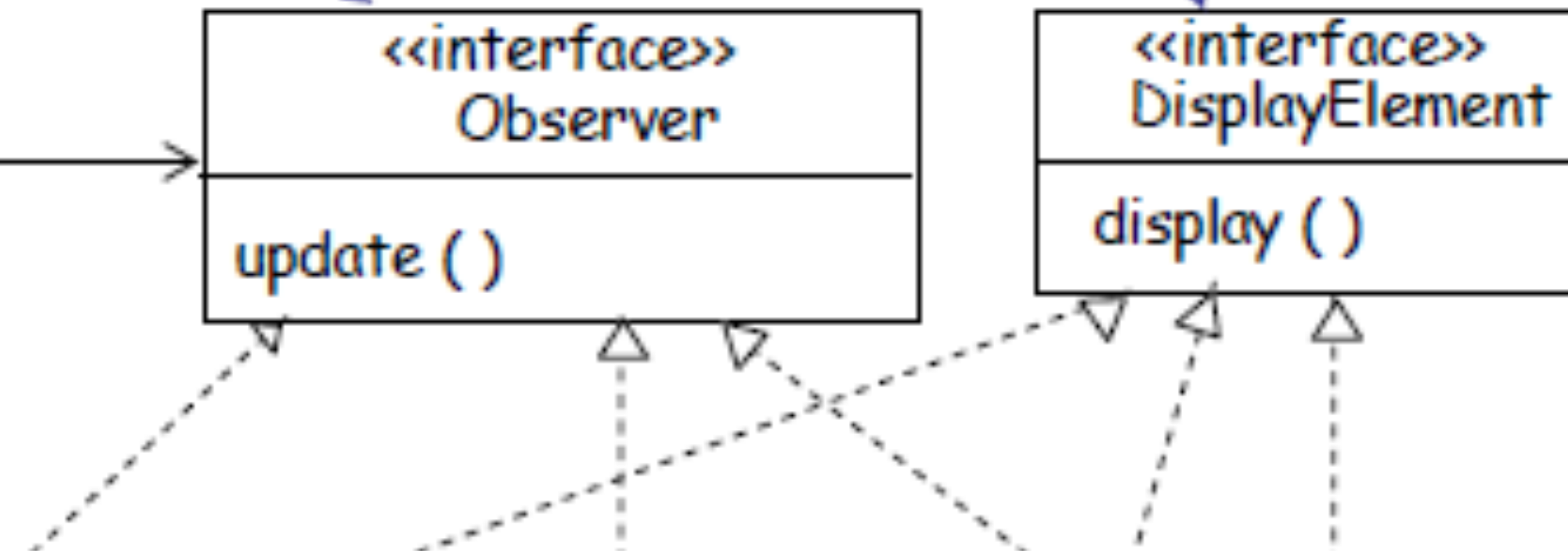
All weather components implem
interface. This gives the subjec
interface to talk to when it com

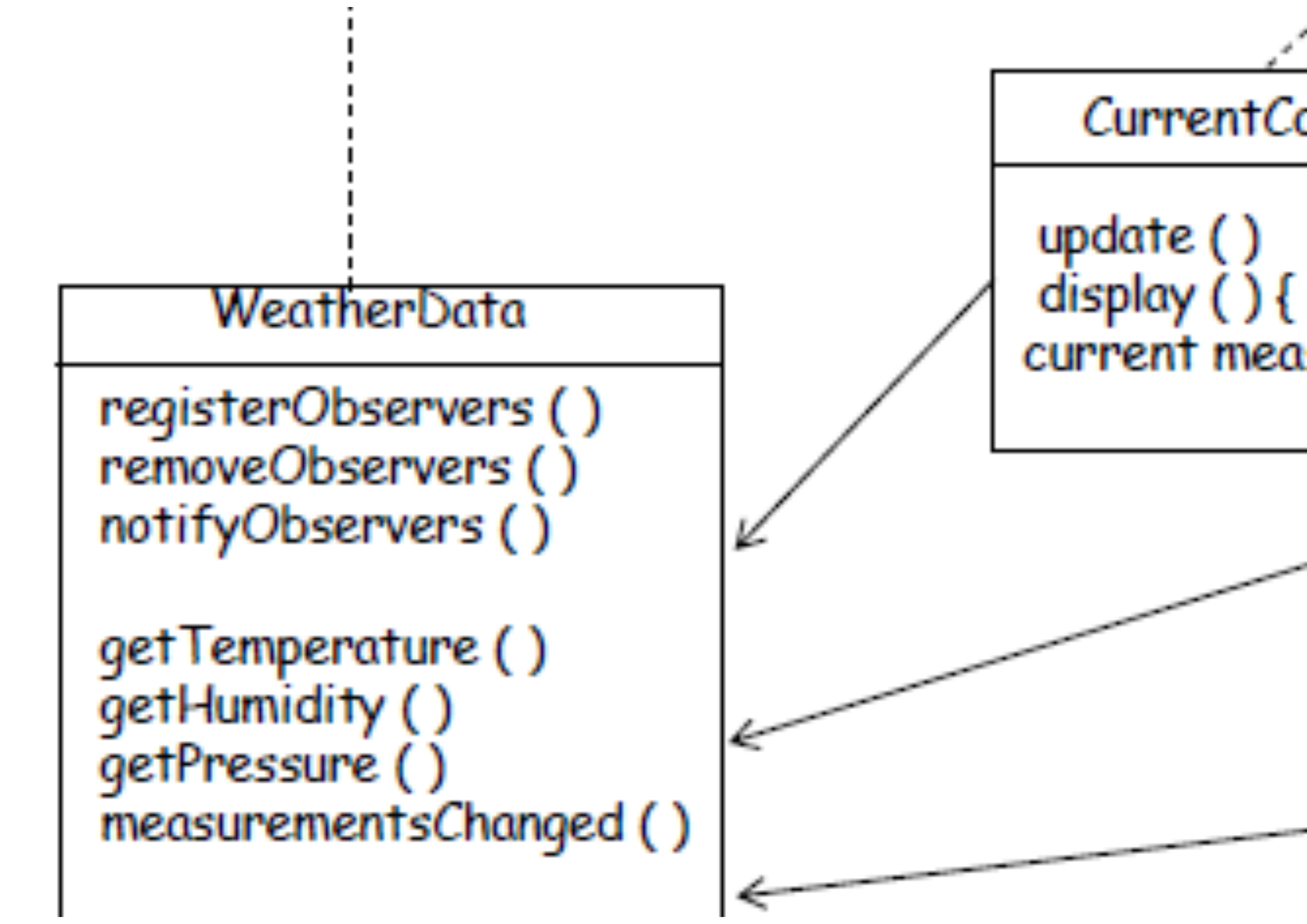


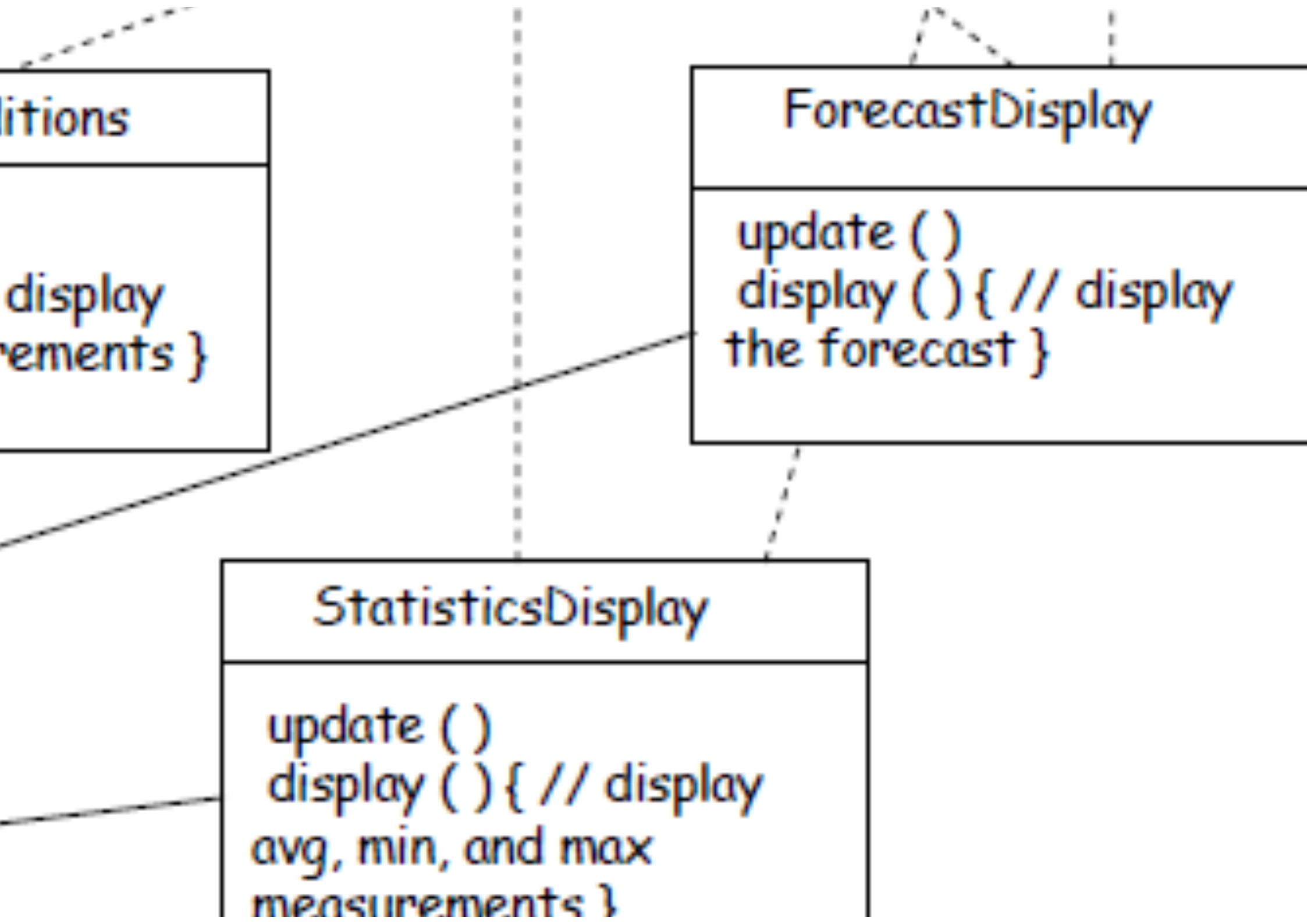
er Station

t the Observer
a common
time to update.

Create an interface for all display elements to implement. The display elements just need to implement a display ( ) method.




WeatherData
registerObservers ( )
removeObservers ( )
notifyObservers ( )
getTemperature ( )
getHumidity ( )
getPressure ( )
measurementsChanged ( )
update ( )
display ( ) {
current mea


itions
display
rements }
ForecastDisplay
update ( )
display ( ) { // display
the forecast }
StatisticsDisplay
update ( )
display ( ) { // display
avg, min, and max
measurements }

mplementing the

```
lic interface Subject {
ublic void registerObserver (Observer o);
ublic void removeObserver (Observer o);
ublic void notifyObservers ( );
```

This
obser
chang

veather Station

Both of these methods take an Obser
as an argument, that is the Observer
be registered or removed.

thod is called to notify all
rs when the Subject's state has
d.

```
public void update (float temp, float humidity, f
```

```
blic interface DisplayElement {
 public void display ( );
```

The DisplayElement interface just includes one method, display( ), that we will call

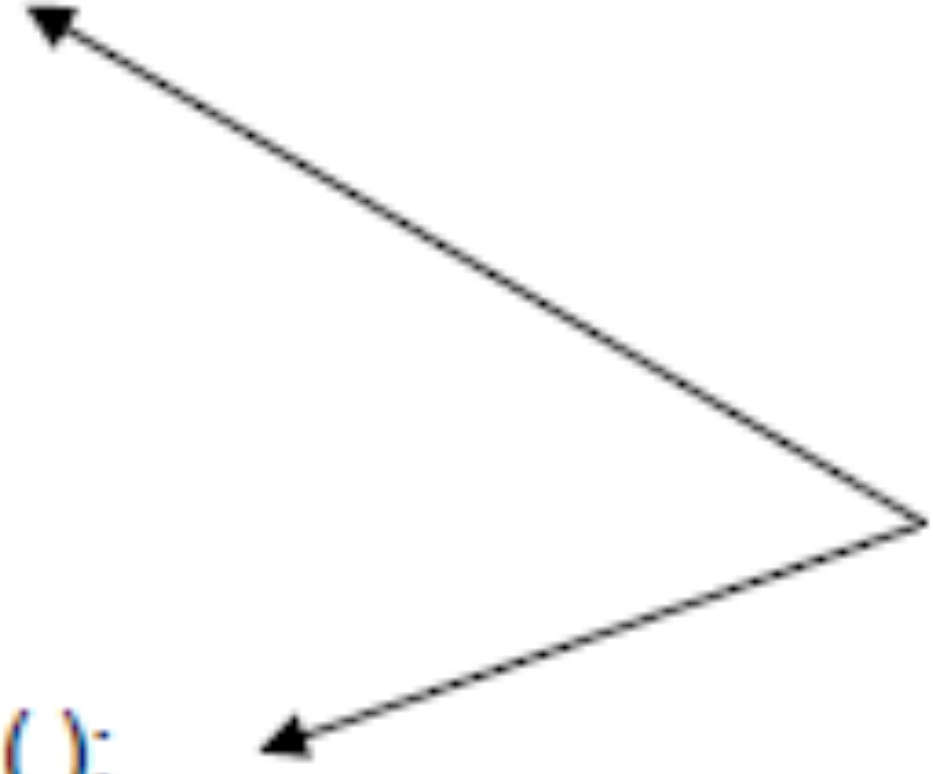
at pressure);

implemented by all obse
so they all have to imple
the update ( ) method.

These are the state values the Observers get from the Subject when a weather measurement changes.

WeatherData and

```
lass WeatherData implements Subject {
vate ArrayList observers;
vate float temperature;
vate float humidity;
vate float pressure;

olic WeatherData ( ){
  observers = new ArrayList ( );

olic void registerObserver (Observer o) {
```

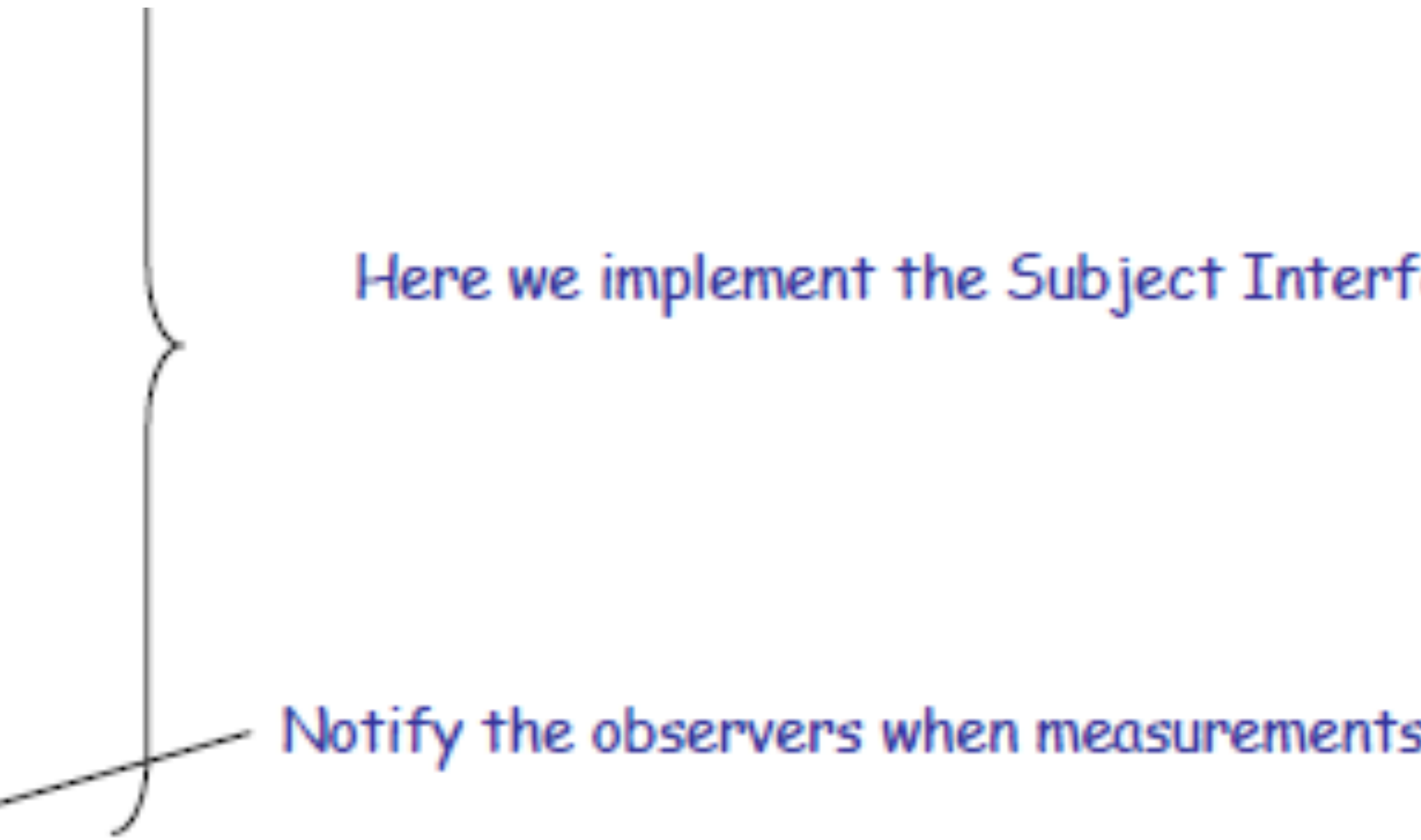


A
ar

Subject Interface

ed an ArrayList to hold the Observers,
we create it in the constructor

```
olic void removeObserver (Observer o) {
  int j = observer.indexOf(o);
  if (j >= 0) {
        observers.remove(j);
  } }
olic void notifyObservers ( ) {
  for (int j = 0; j < observers.size(); j++) {
        Observer observer = (Observer)observers.get(j);
        observer.update(temperature, humidity, pressure)
  }}
olic void measurementsChanged ( ) {
   notifyObservers ( ); }
```


Here we implement the Subject Interf
Notify the observers when measurements

ne Display Eleme

Implements the Observer and Displa

class CurrentConditionsDisplay implements Obser

```
rivate float temperatue;
rivate float humidity;
rivate Subject weatherData;

ublic CurrentConditionsDisplay (Subject weatherDa
```

nts

Element interfaces

r, DisplayElement {

) {

The constructors passed the
weatherData object (the sub
and we use it to register the
as an observer.

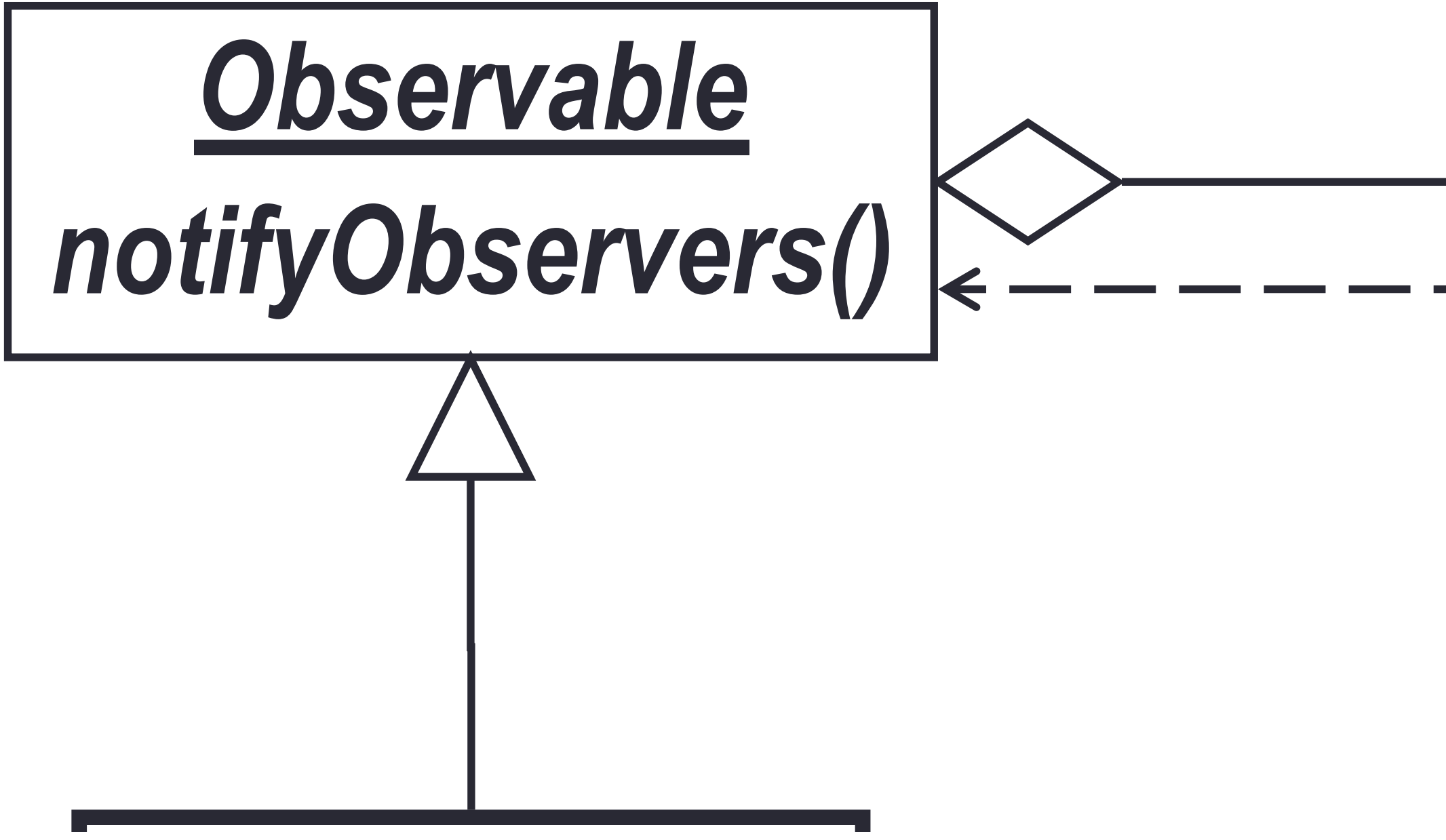
```
    weatherData.registerObserver (this);

ublic void update (float temperature, float humidity,
    this.temperature = temperature;
    this.humidity = humidity;
    display ( );

ublic void display ( ){
    System.out.println(" Current conditions : " + temp
```

at pressure) {

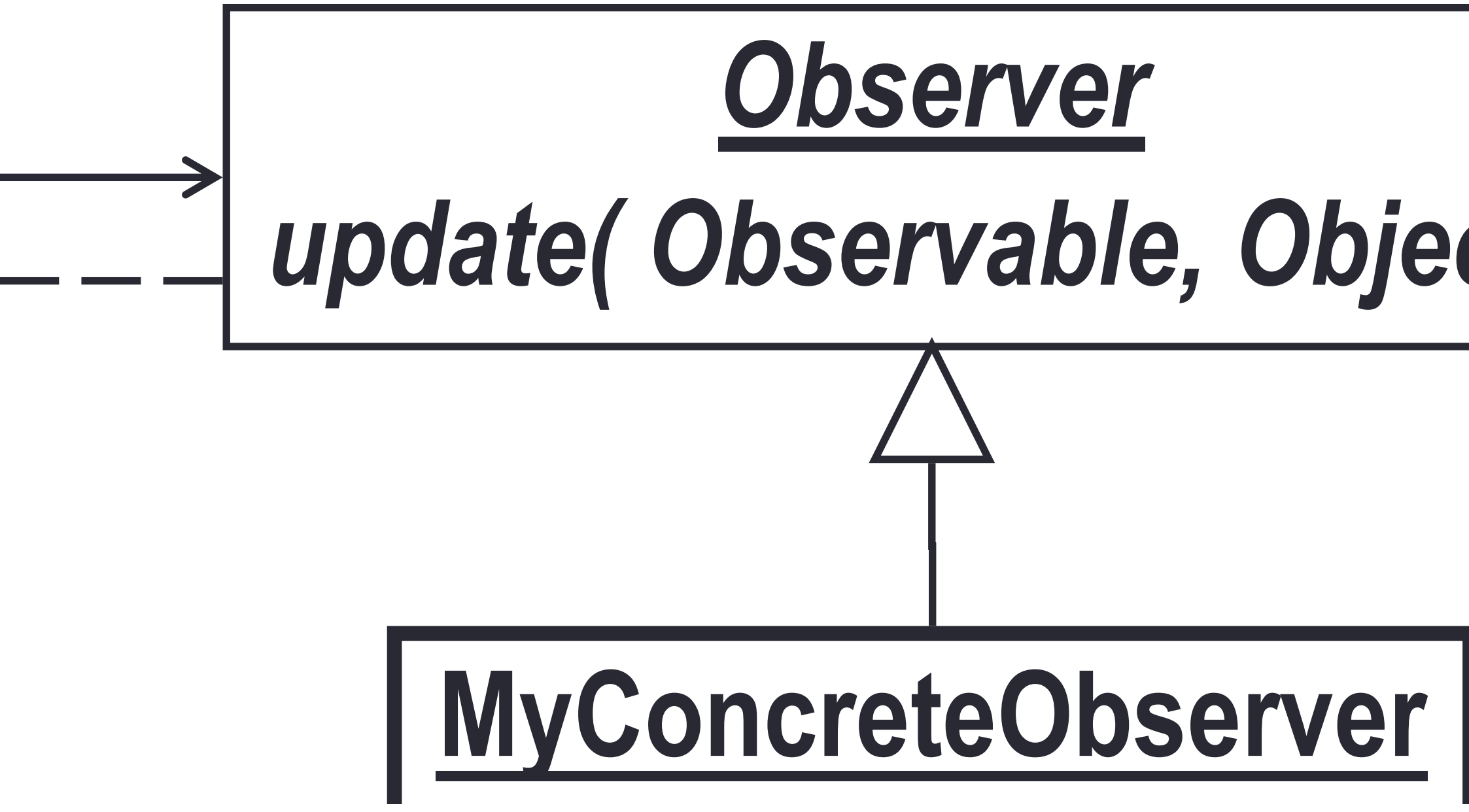
When update ( ) is called, we save the temp and humidity and call display ( )

rature + " F degrees and " + humidity + " % humidity

Example: Observ



el in the Java API



MyObservable

Key: *Java API Class* Deve

Model-View-Controller pattern : (O

observerState

update(...)

oper Class

servable, Observer, (Client & Setup))

Observer: Advanta

- Subject(s) update Ob
common interface
- Observers are loosel

ges

servers using a

coupled in that

Subject(s) knows not
than they implement
interface.

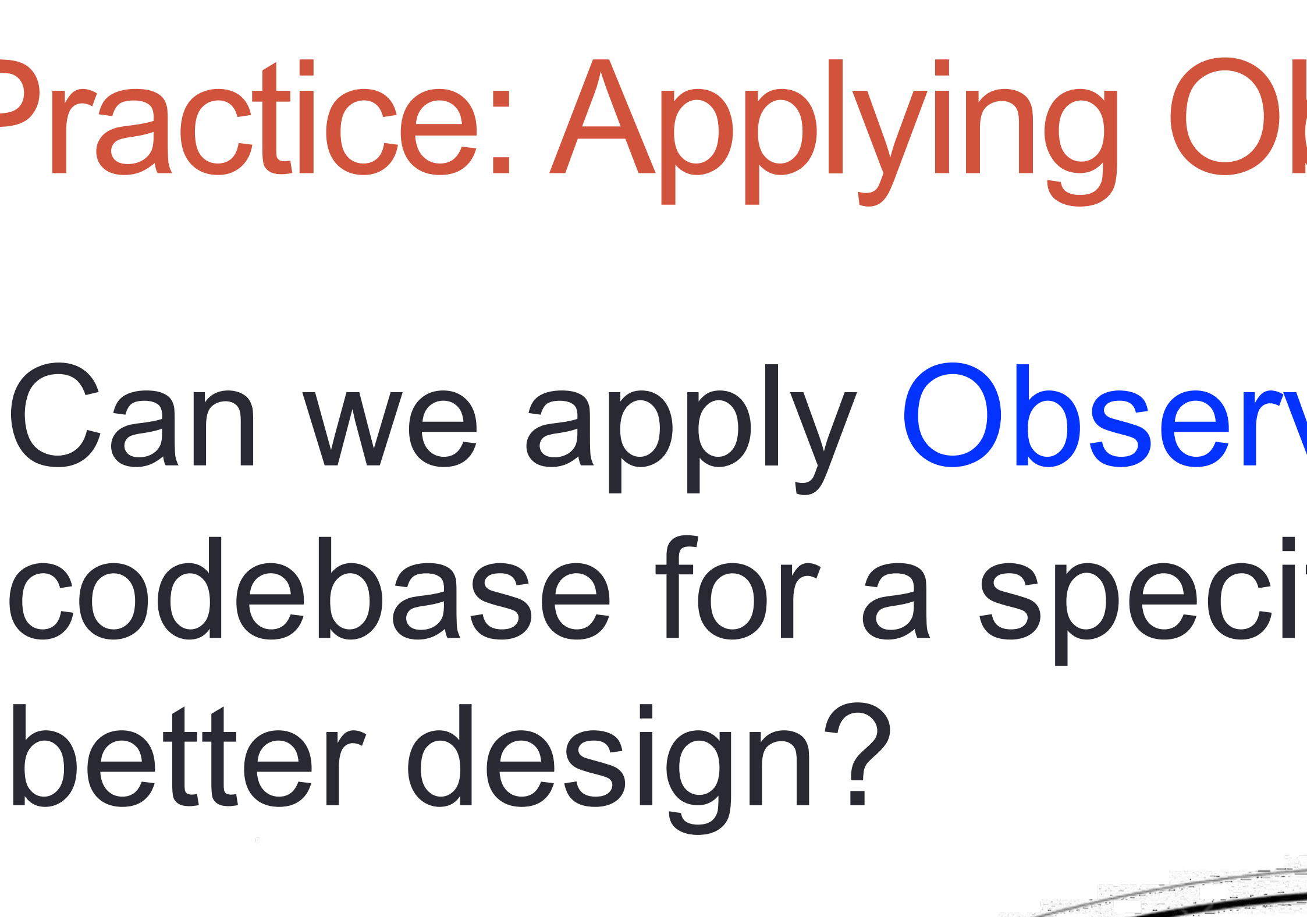
Don’t depend on a sp
notification for your C

ing about them, unle
he Observer

ecific order of
bservers.

Practice: Applying OB

Can we apply Observ
codebase for a speci
better design?



server in Codebase

er in any part of the

c requirement/a

TIME FOR


RACTICE

Content

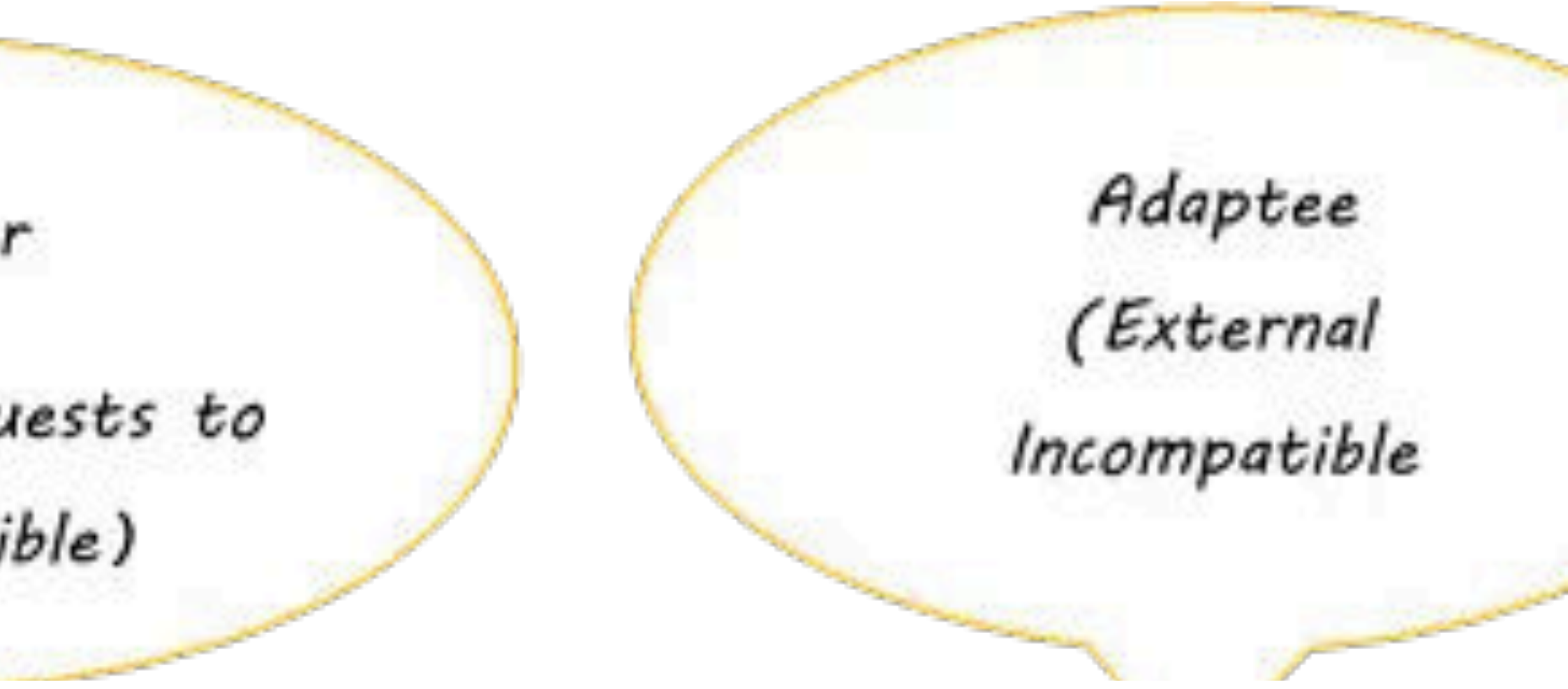
1. Strategy
2. Observer

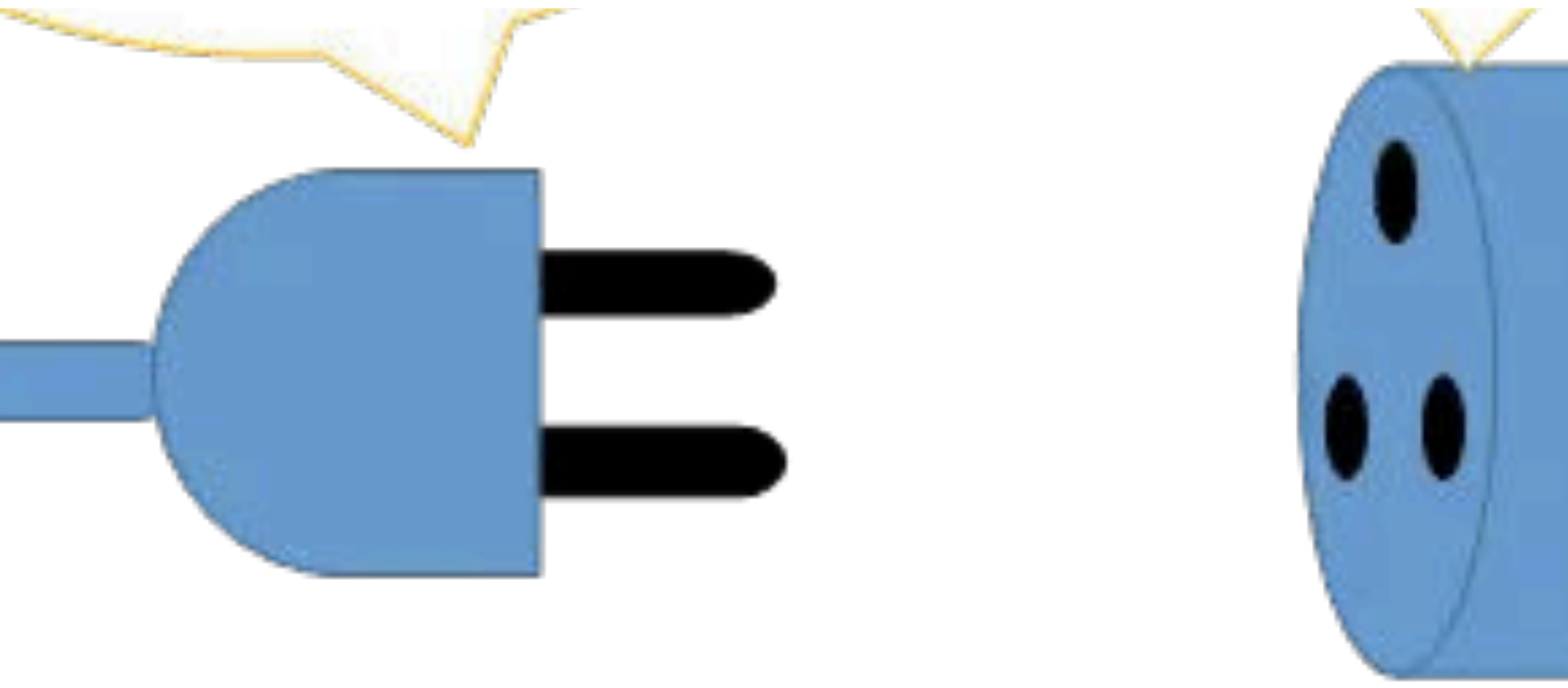
3. Adapter

4. State

Adapter: Connect Inc

ompatible Interfaces





Client needs to get the service fro

cannot inte

Adaptee, which is incompatible &

ct directly

xercise: Connect Media

Existing supported form

New formats from Media

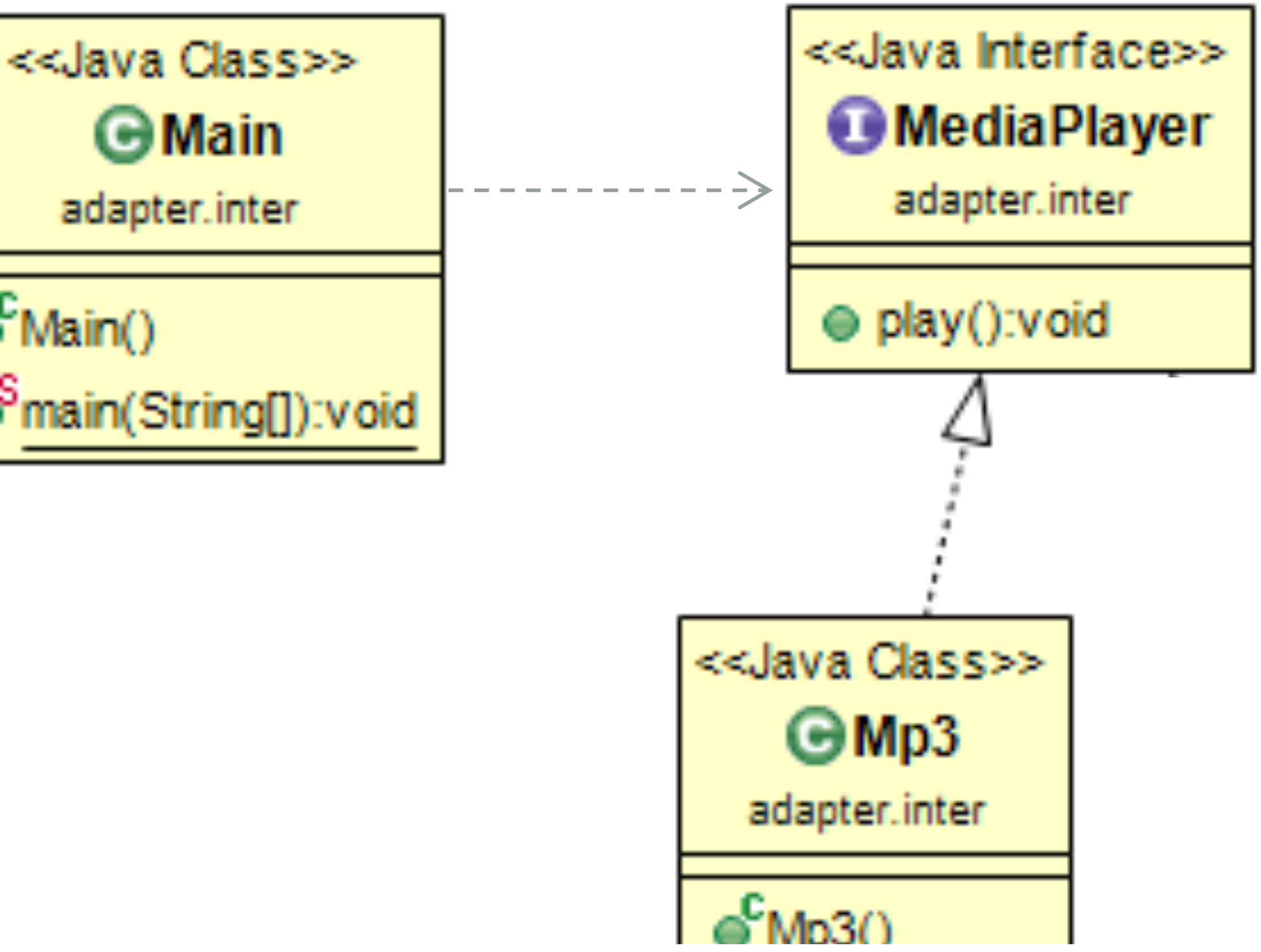
How to support these n

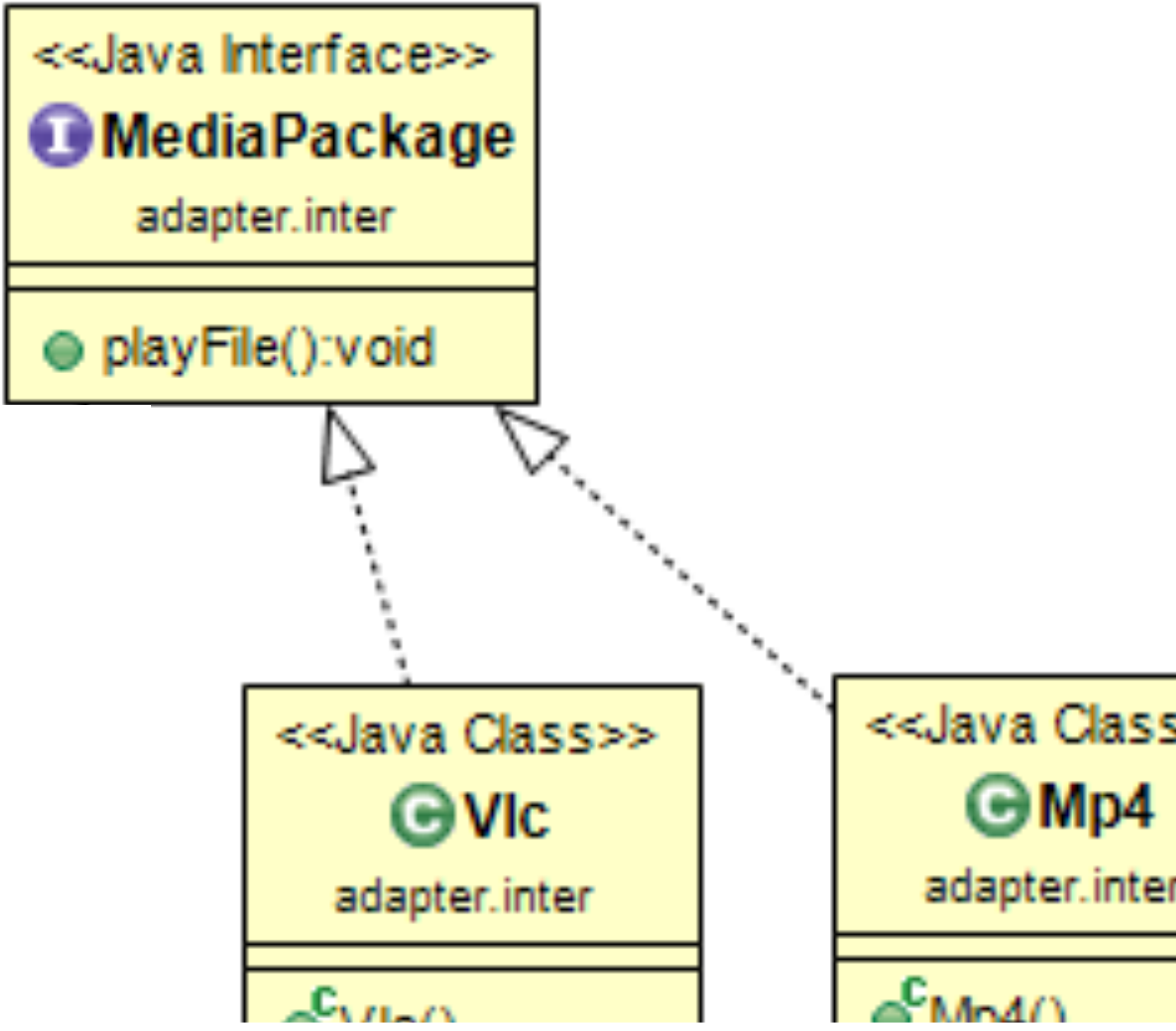
Player to MediaPackag

t: mp3

Package library: VLC, m

w formats?


<<Java Class>>
Main
adapter.inter
Main()
main(String[]):void
<<Java Interface>>
MediaPlayer
adapter.inter
play():void
<<Java Class>>
Mp3
adapter.inter
Mp3()


<<Java Interface>>
MediaPackage
adapter.inter
playFile():void
<<Java Class>>
Vlc
adapter.inter
<<Java Class
Mp4
adapter.inter

Adapter: Intent

Convert the interface of
interface clients expect.

Adapter lets classes wo
otherwise because of in

a class into another

k together that couldn't
ompatible interfaces.

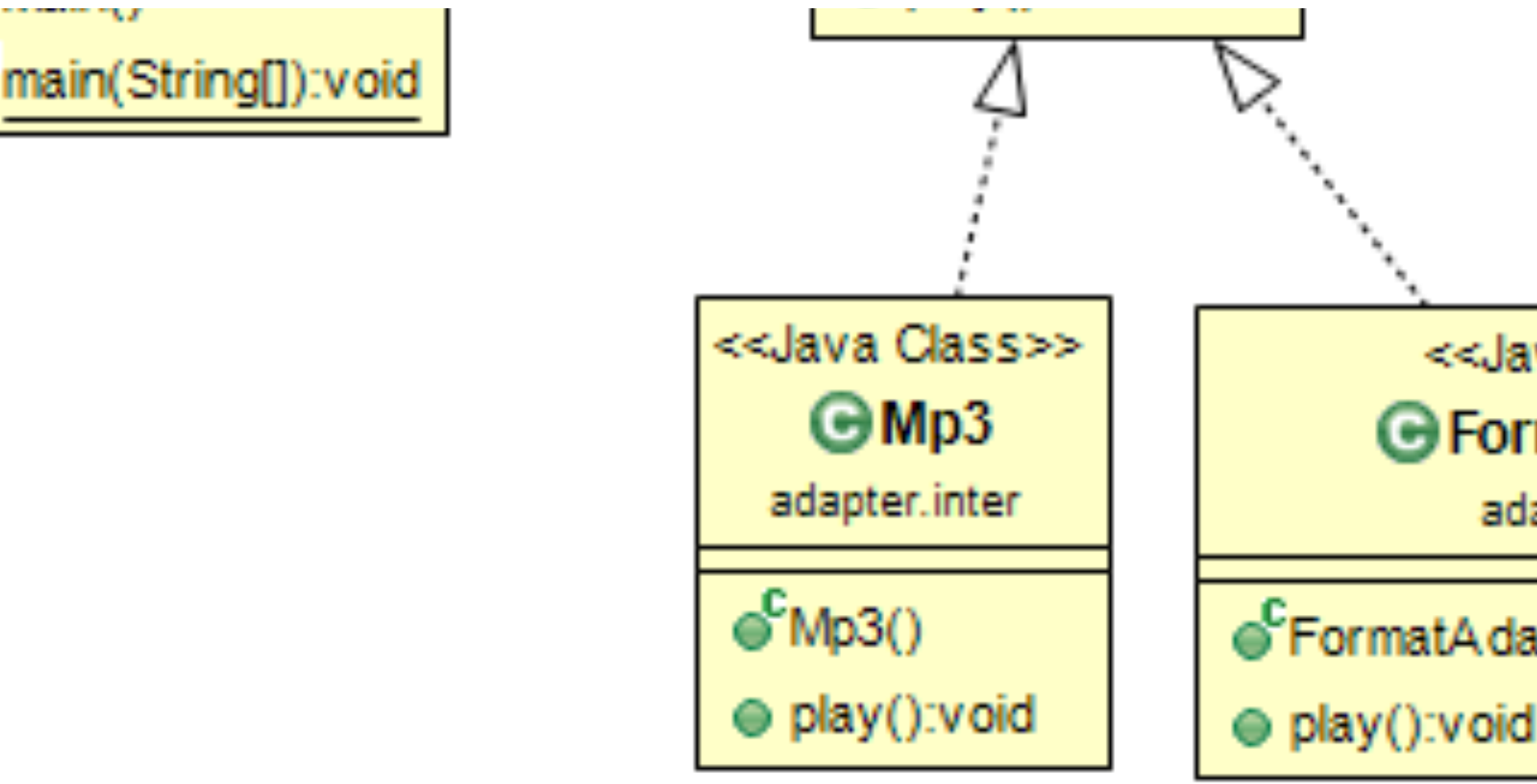
Wrap an existing class w

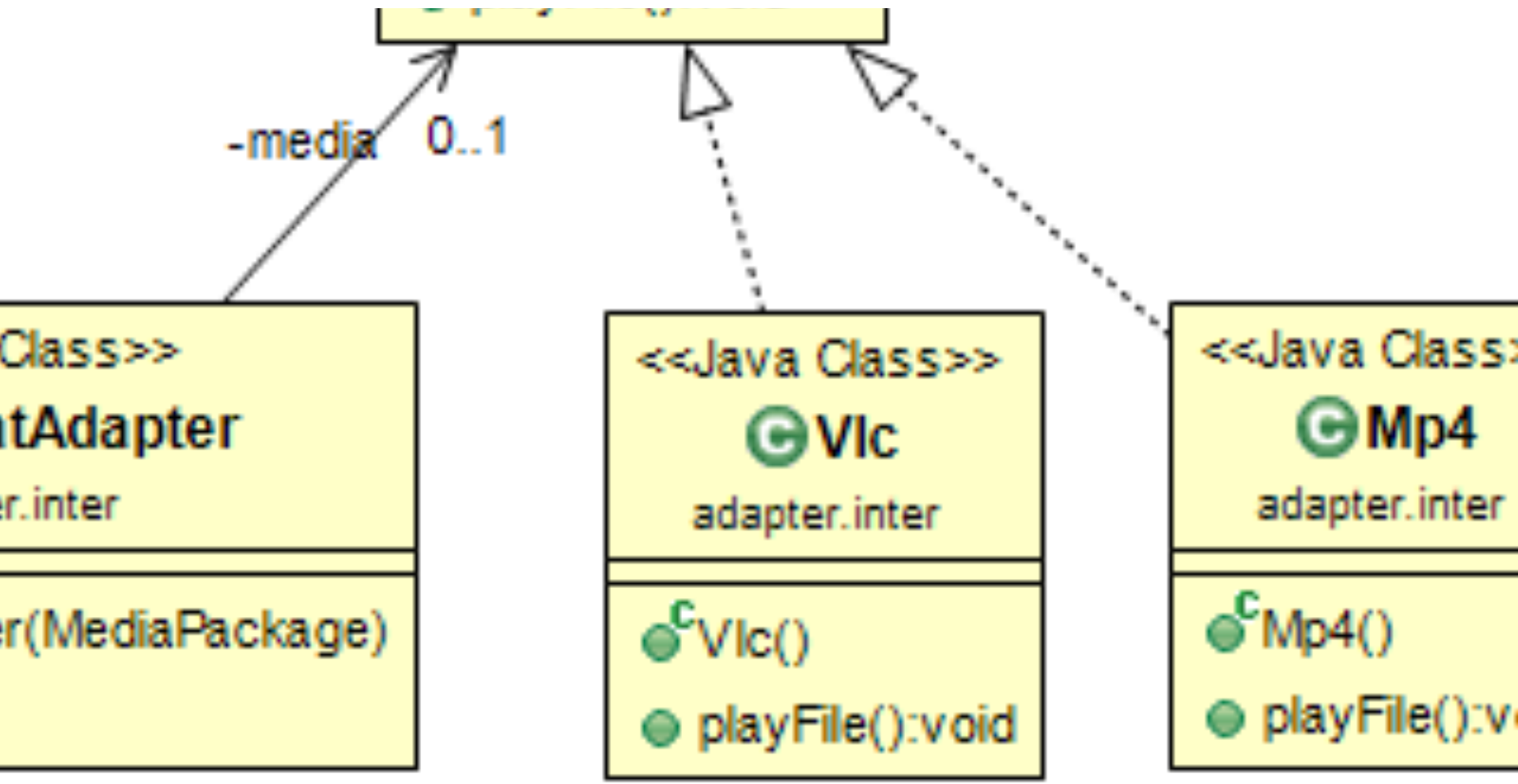
- Also know as Wrapper

ith a new interface

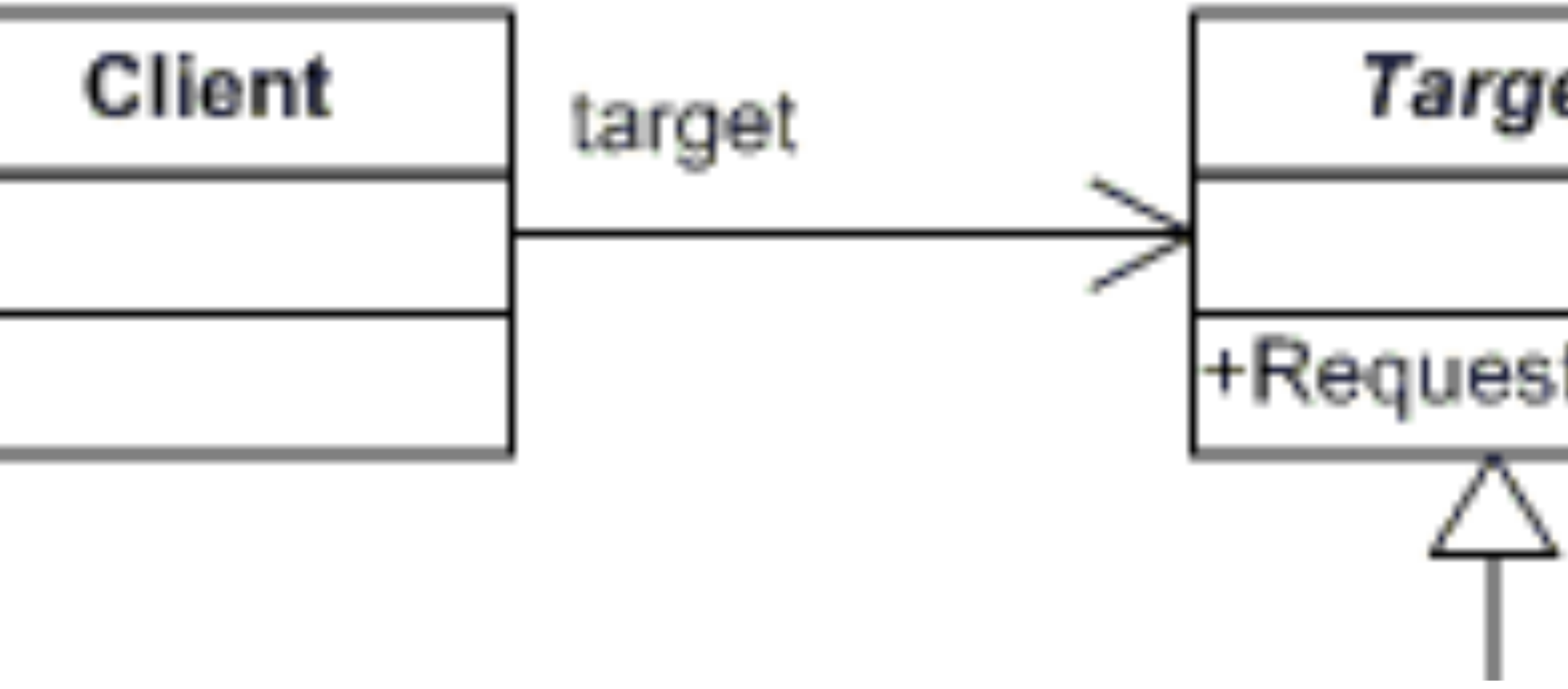
Adapter: Playing N

ew Format

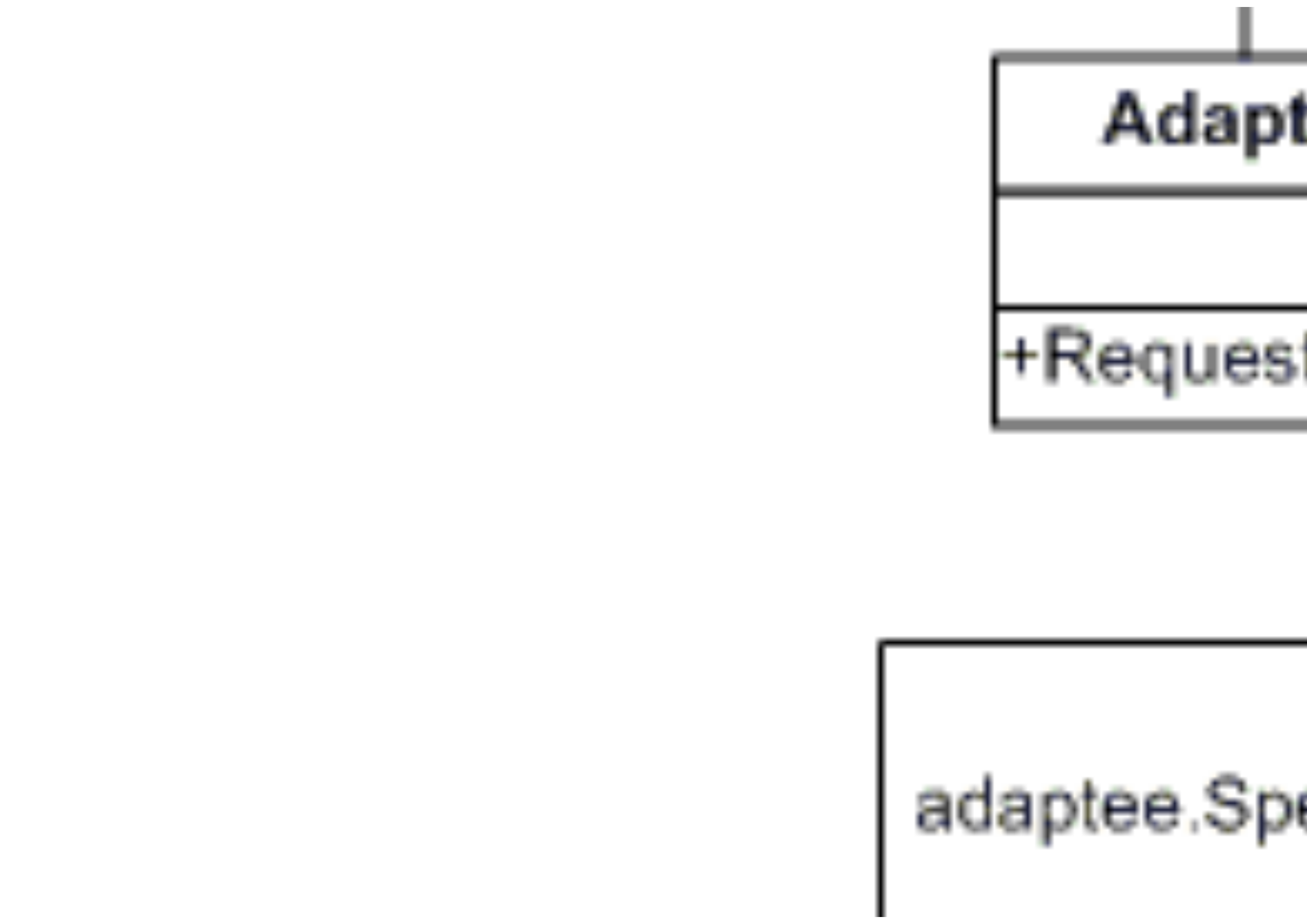

main(String[]):void
<<Java Class>>
Mp3
adapter.inter
Mp3()
play():void
play():void

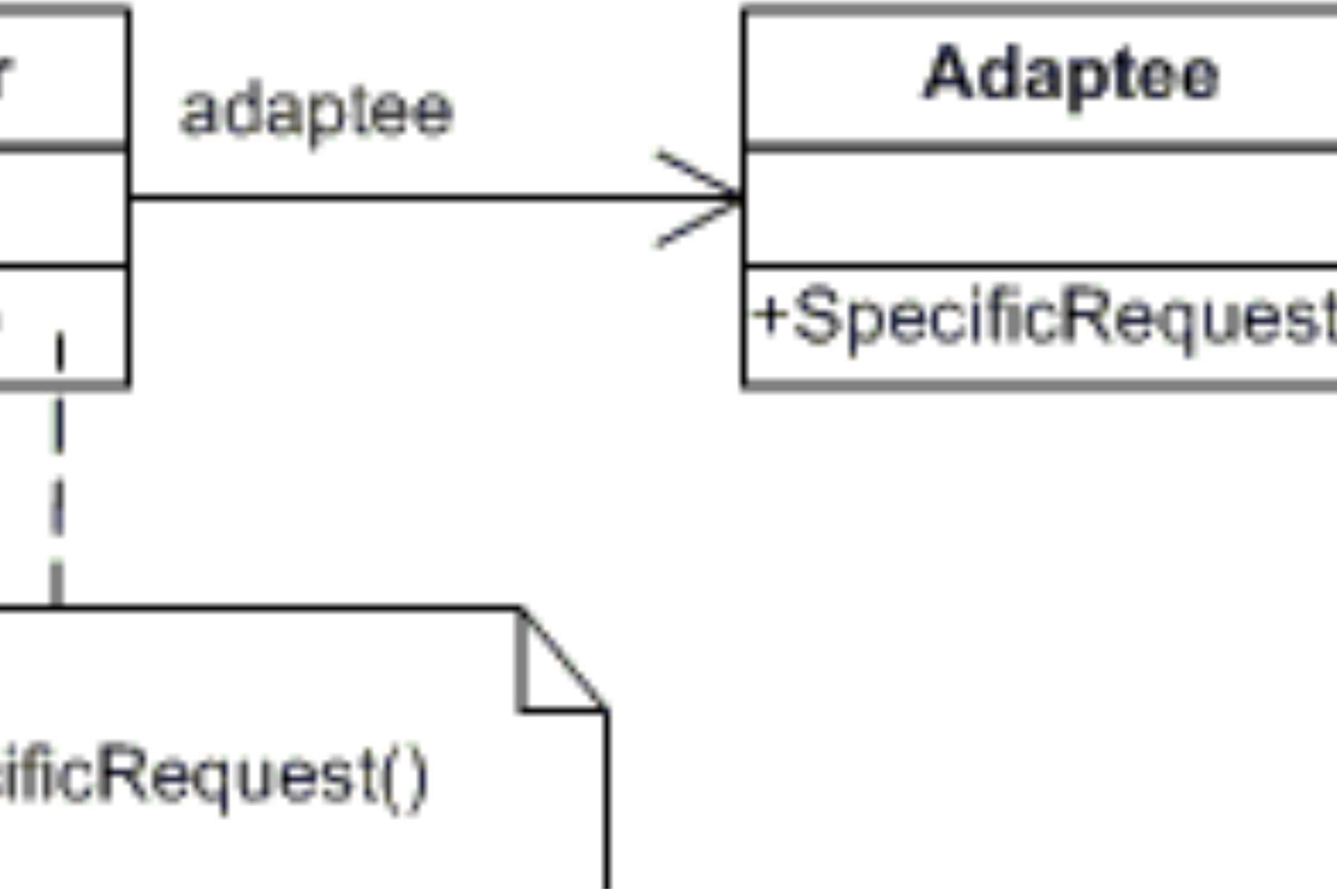

-media
0..1
Class>>
tAdapter
r.inter
r(MediaPackage)
<<Java Class>>
Vlc
adapter.inter
Vlc()
playFile():void
<<Java Class>>
Mp4
adapter.inter
Mp4()
playFile():v

Adapter (object) Pa



ttern: Structure

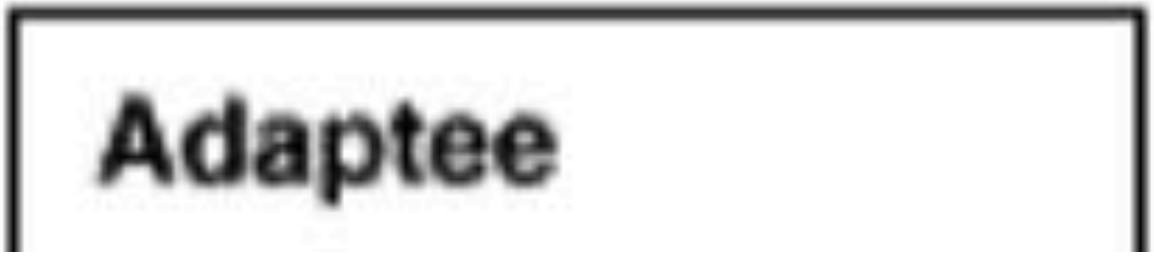

Adapt
+Request
adaptee.Spe


adaptee
Adaptee
+SpecificRequest
ificRequest()

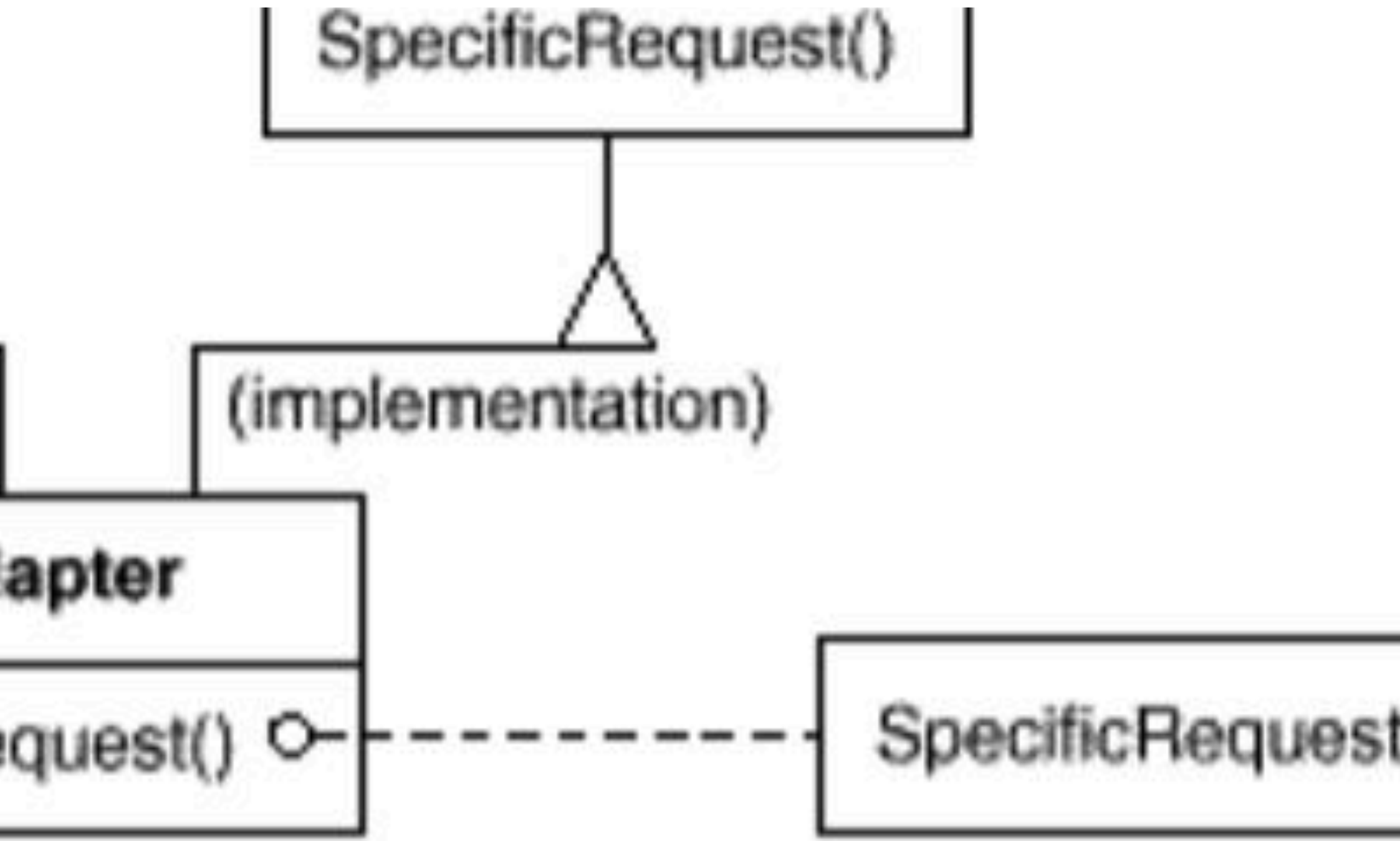
Adapter (class) Pa

Multi-inheritance?

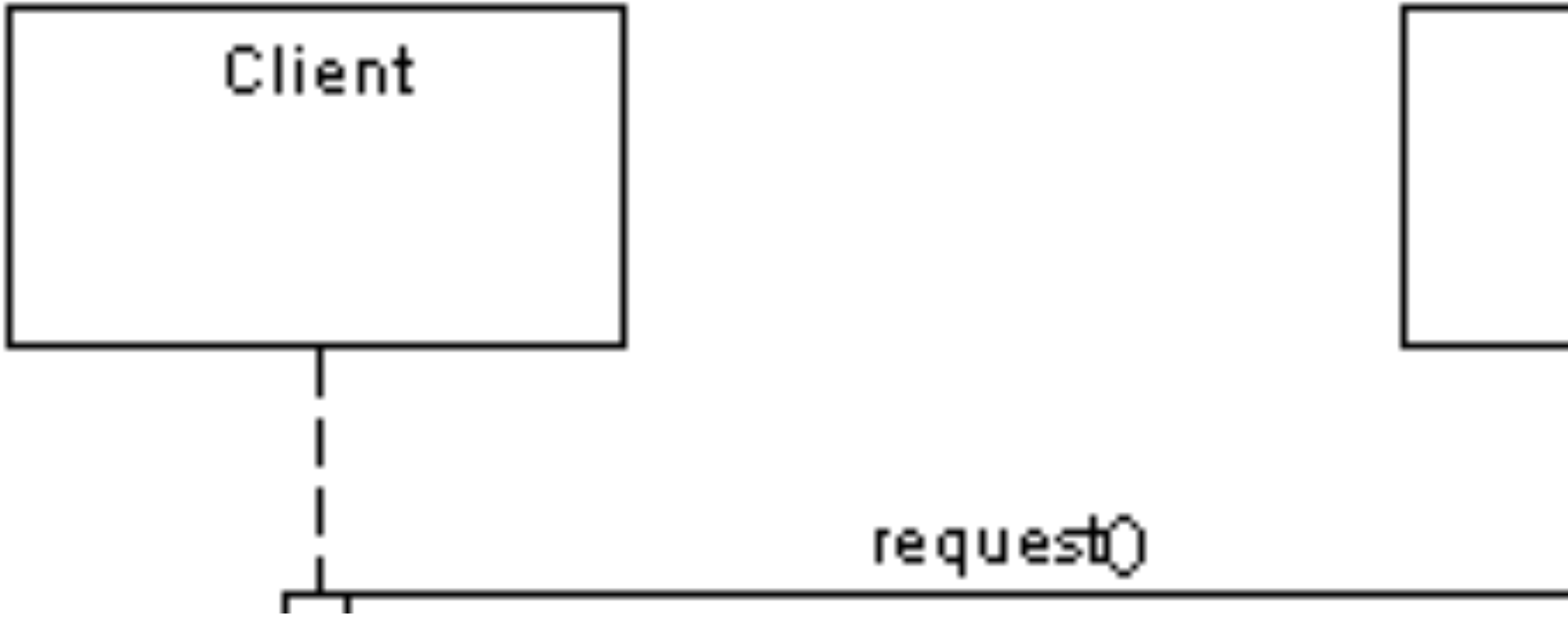
tern: Structure



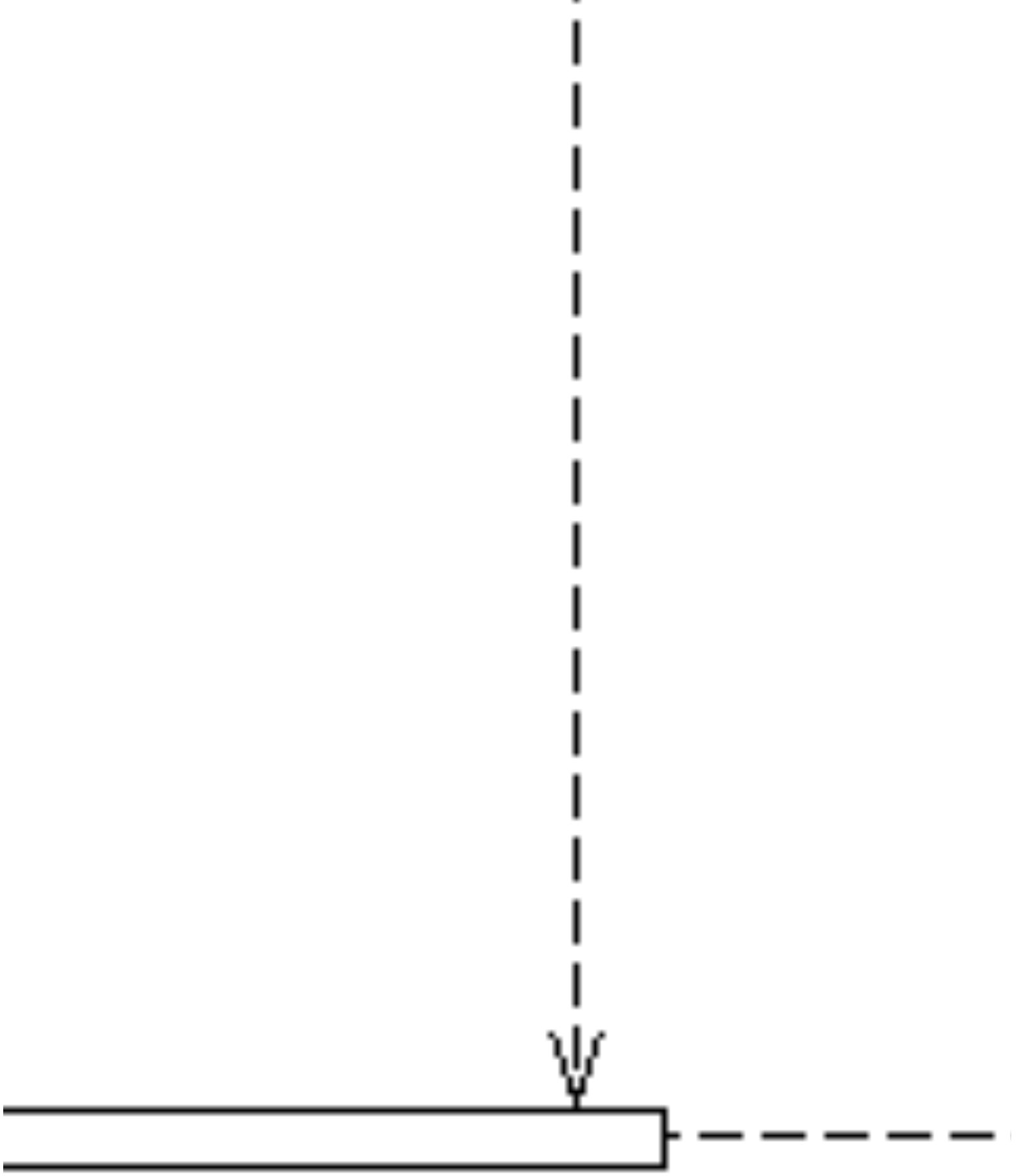

Request()

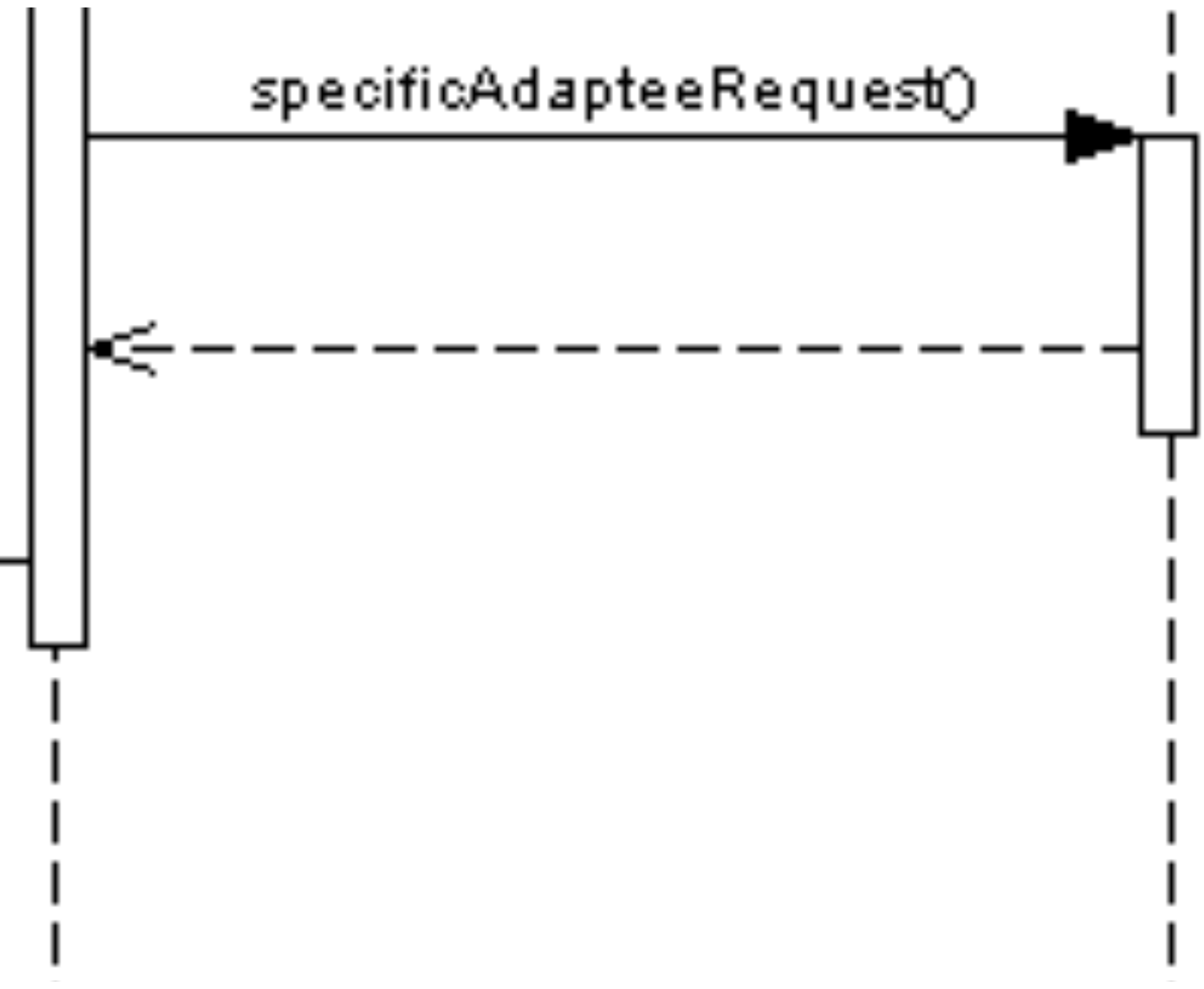

SpecificRequest()
(implementation)
apter
equest()
SpecificRequest

Adapter: Collabora

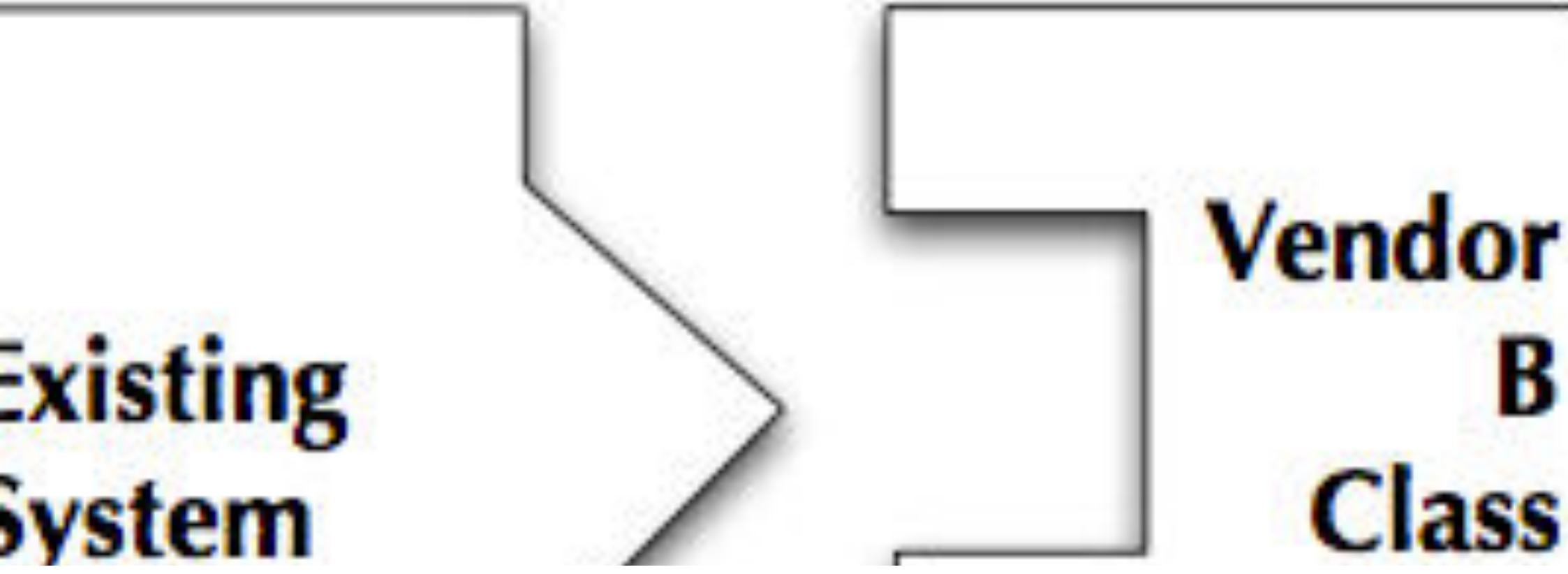


tion

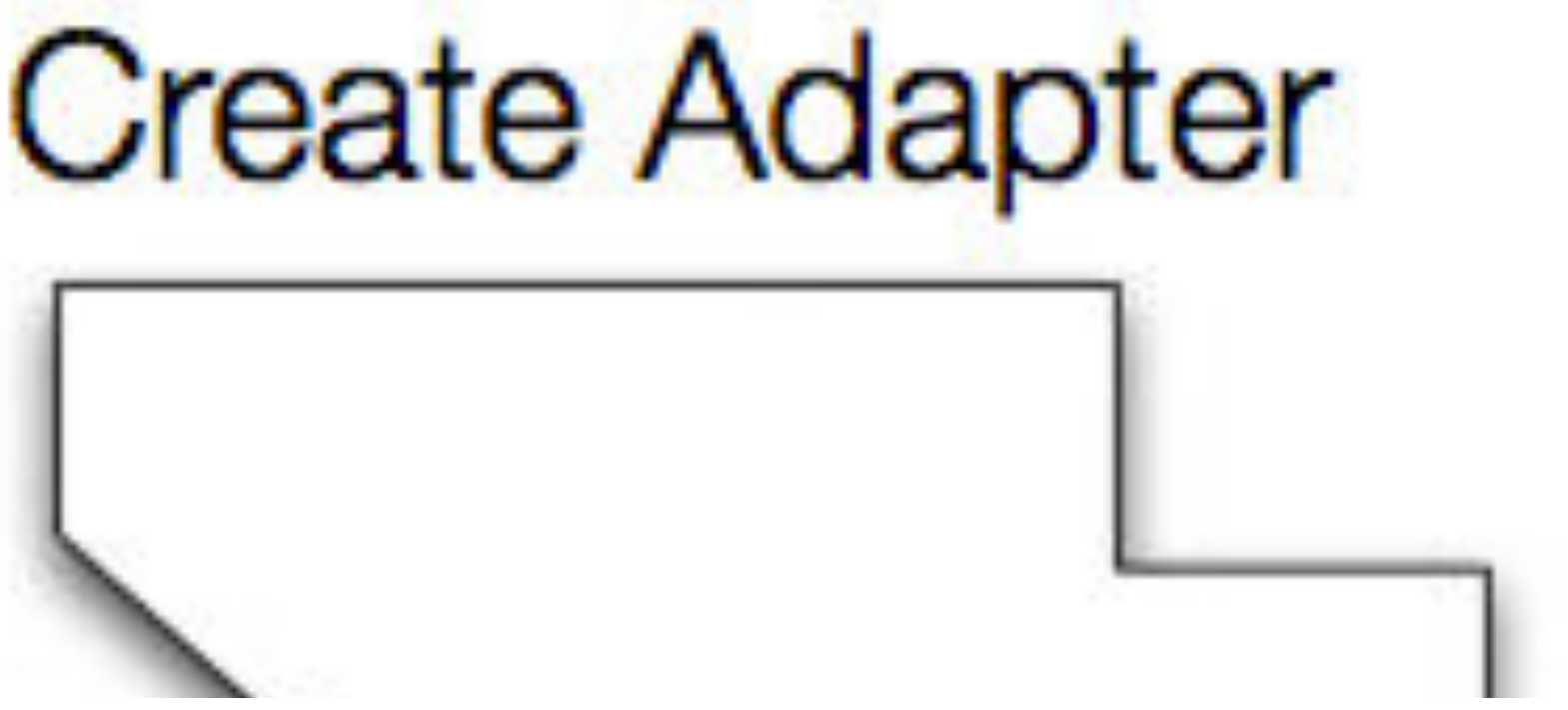



specificAdapteeRequest()

E.g. Software Adap



ters



nterface Mismatch

Need Adapter

And then...

E.g. Software Adap

ters

plug it in

enefit: Existing system and n

w vendor library do not char

Motivation

Sometimes a toolkit c
be used because its
incompatible with the
an application

r class library can not
nterface is
interface required by

We can not change t
since we may not hav
Even if we did have t
probably should not c
each domain-specific

e library interface,
e its source code

ne source code, we
nange the library for
application

Adapter: Applicabil

You want to use an e
interface does not ma
You want to create a

ty

xisting class, and its
tch the one you need
eusable class that

cooperates with an
incompatible interfac

ated classes with

s

Practice: Applying Ad

Can we apply Adapte
codebase for a speci
better design?

apter in Codebase

r in any part of the
c requirement/a

TIME FOR


RACTICE

Content

1. Strategy
2. Observer

3. Adapter

4. State

Example: TV Rem

V Remote object with a

mple button to perform

ction

if the State is ON, it

```
public
    pr
    pu
    }
    pu
```

ote

```
lass TVRemoteBasic {

ate String state="";

ic void setState(String state){
this.state=state;


ic void doAction(){
```

will turn on the TV

if state is OFF, it will

turn off the TV

/e can implement it

sing **if-else condition**

}

pu

```
    System.out.println("TV is turned ON"
}else if(state.equalsIgnoreCase("OFF")){
    System.out.println("TV is turned OFF
}


ic static void main(String args[]){
TVRemoteBasic remote = new TVRemoteBasic

remote.setState("ON");
remote.doAction();

remote.setState("OFF");
remote.doAction();
```

WHAT IS PR
SOLUTION?

OBLEM &

TV Remote

Problem in TV Rem

If else condition f

Client code should know
use for setting the state

note

or different states

the specific values to

of remote

If number of states incre
coupling between imple
code will be very hard to
Design will become mor
and switches or multiple

- But still the above problen

ase then the tight
nentation and the client
maintain and extend
e cleaner by using enum
if then else

Problem in Softwa

Tired of co

When writing code, our classes
transformations

What starts out as a simple cla
added

e Design

ditionals?

often go through a series of

s will grow as behavior is

If you didn't take the necessary
become difficult to understand

Too often, the state of an objec
Boolean attributes and decidin
values

This can become cumbersome
the complexity of your class st

This is a common problem on

precautions, your code will
nd maintain.

is kept by creating multiple
how to behave based on the

and difficult to maintain when
rts to increase

host projects

Solution for TV Re

State Design Pattern

- Creating objects which
  and a context object w
  its state object change

note

(Behavioral Pattern)

represent various states

hose behavior varies as

s

“Allows an object t
when its interna
The object will appea

alter its behaviour

state changes.

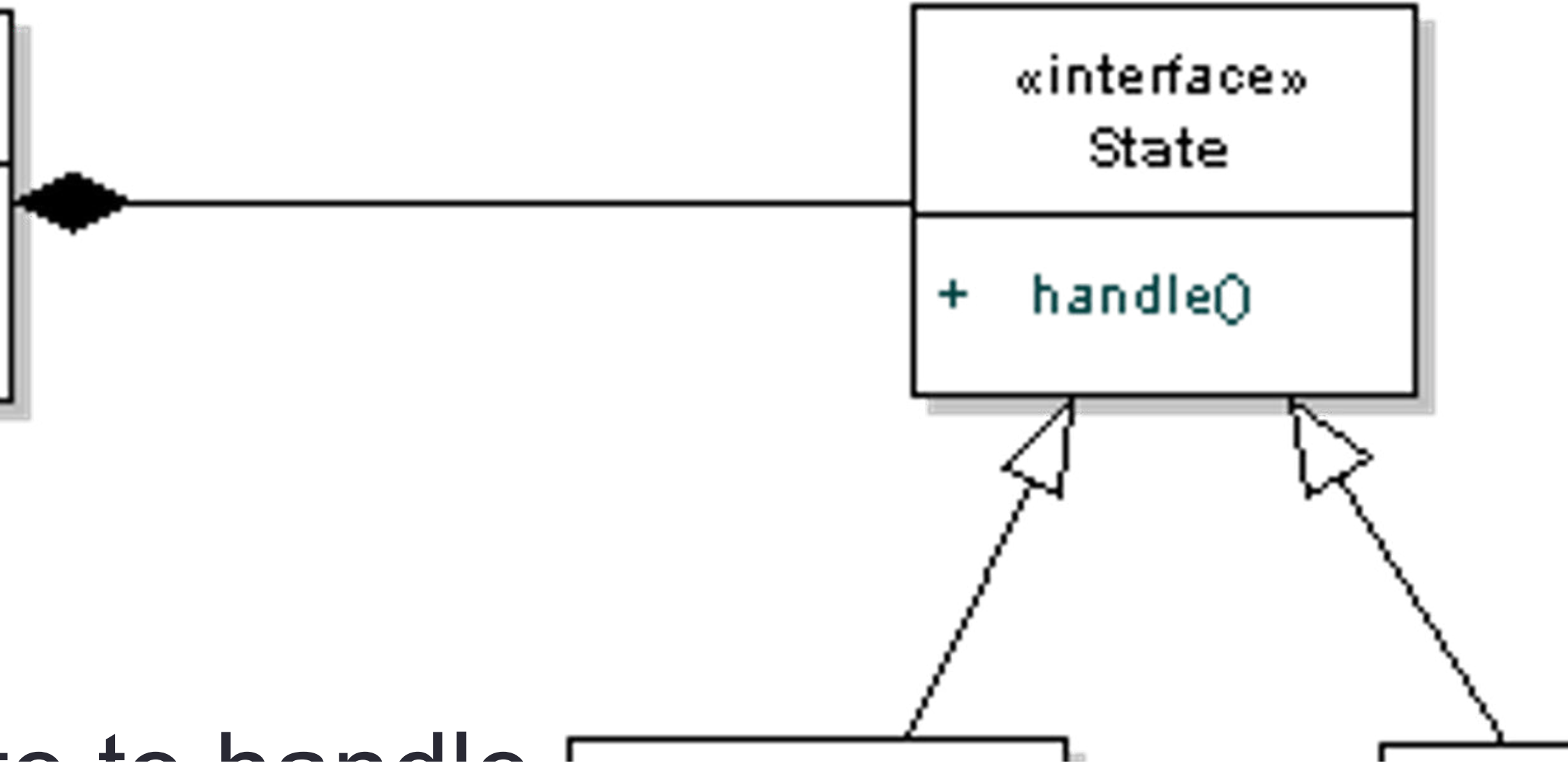
to change its class.”

State Design Patte

Context

- Have a number of internal state

rn: Structure



State interface

- A common interface for all conc

  encapsulating all behaviour ass

ConcreteState

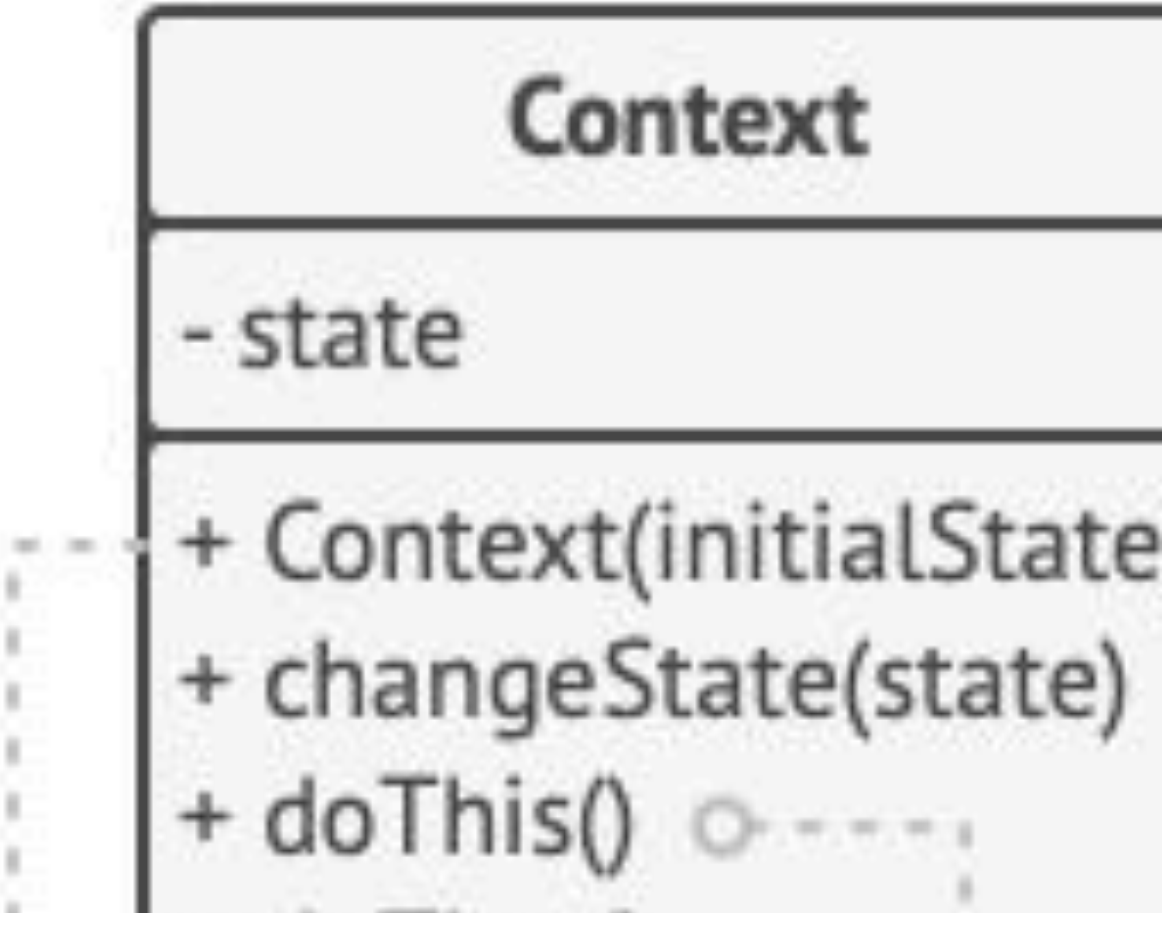
- implements it's own implementa

ete states,

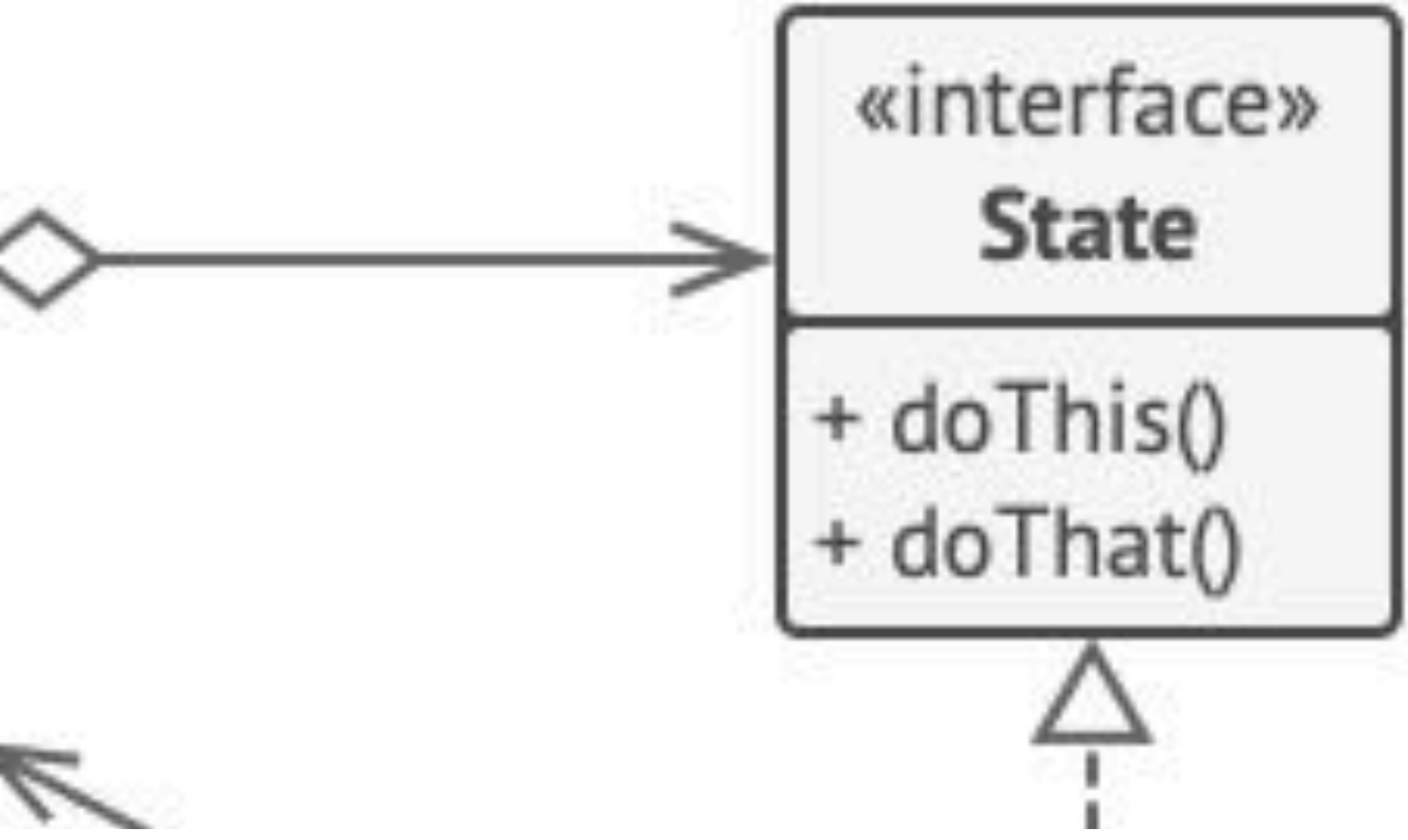
ciated with a particular state.

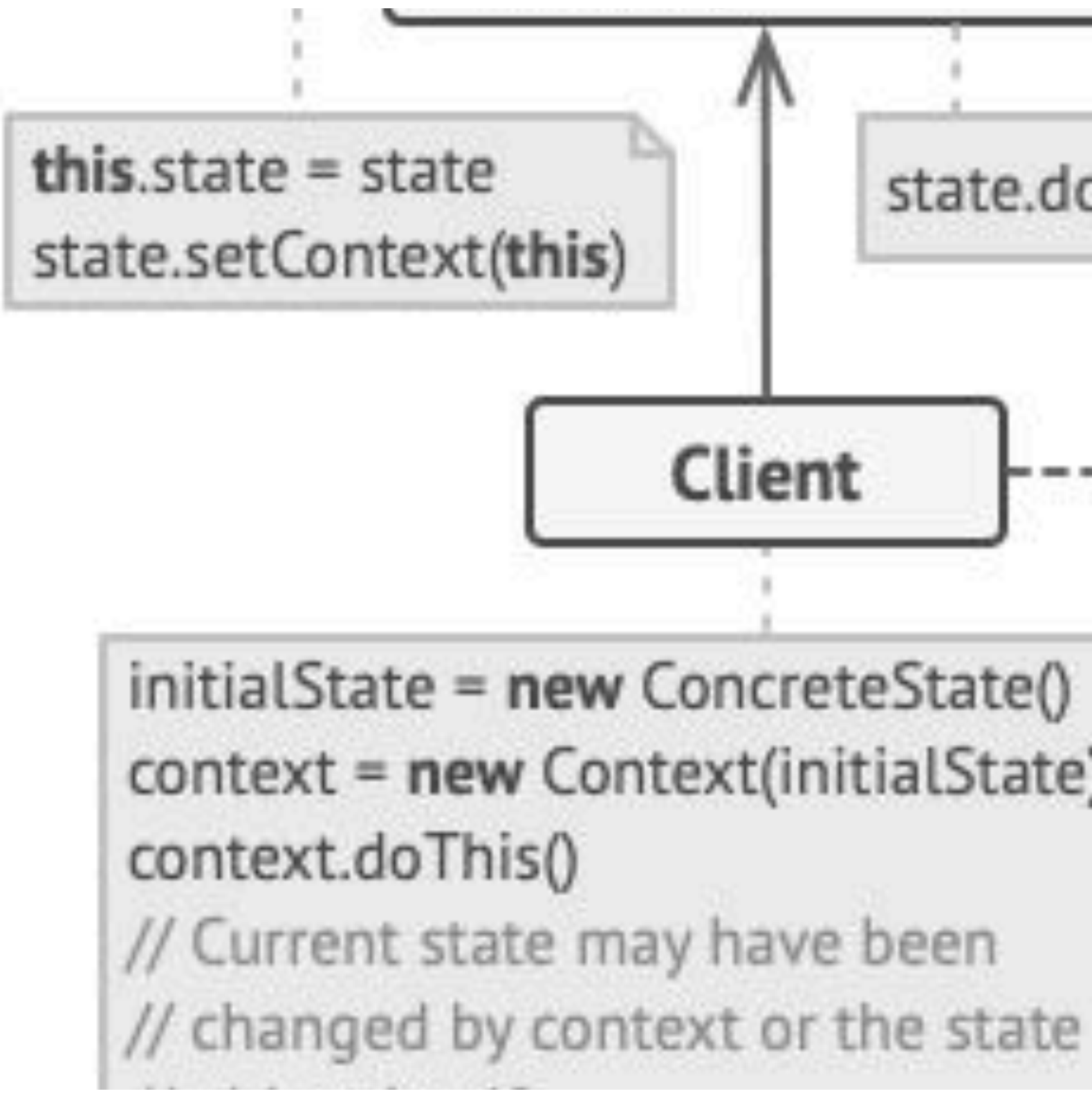
ion for the request

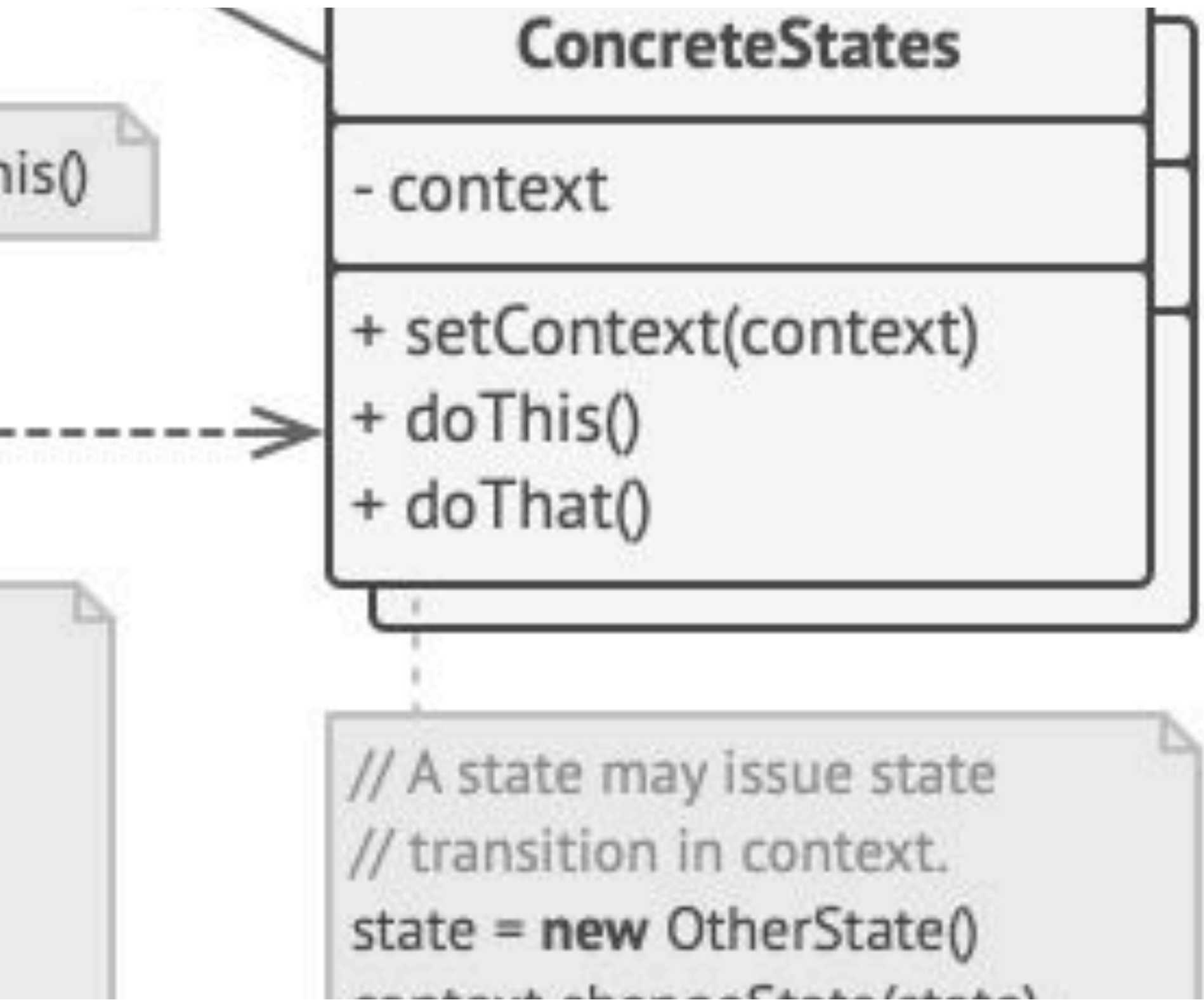
State Design Patte



rn: Structure




this.state = state
state.setContext(this)
state.do
Client
initialState = new ConcreteState()
context = new Context(initialState)
context.doThis()
// Current state may have been
// changed by context or the state


ConcreteStates
- context
+ setContext(context)
+ doThis()
+ doThat()
// A state may issue state
// transition in context.
state = new OtherState()

Quiz: Design State D

P for TV Remote?

Advantages of Sta

- Implement polymorphic
- The chances of error ar
to add more states for a
it more robust.

e DP

behavior is clearly visible

less and it’s very easy

ditional behavior making

Easily maintainable and

Avoiding if-else or switc

this scenario

Avoiding inconsistent st

Putting all associated be

state object

flexible.

-case conditional logic ir

ates

navior together in one

Practice: Applying

Can we apply State i
codebase for a speci
better design?

State in Codebase

any part of the
c requirement/a

TIME FOR


RACTICE