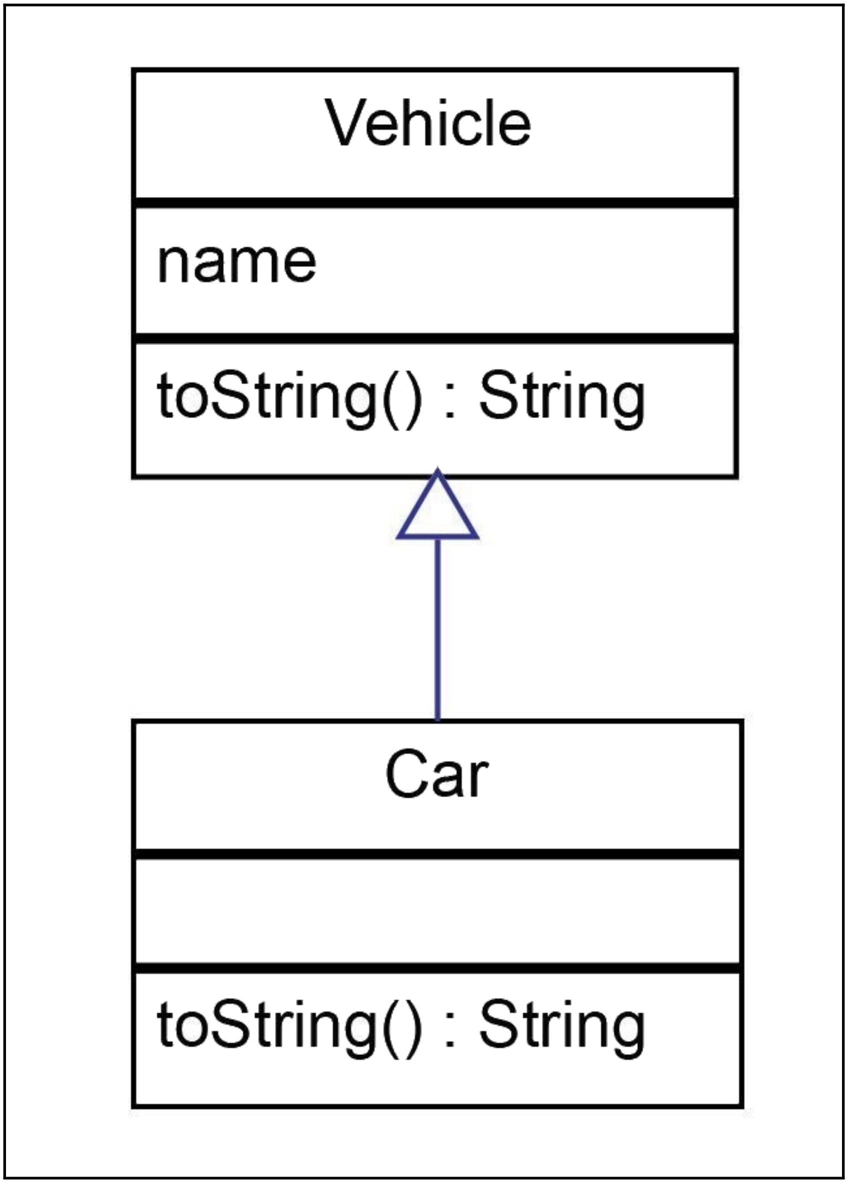
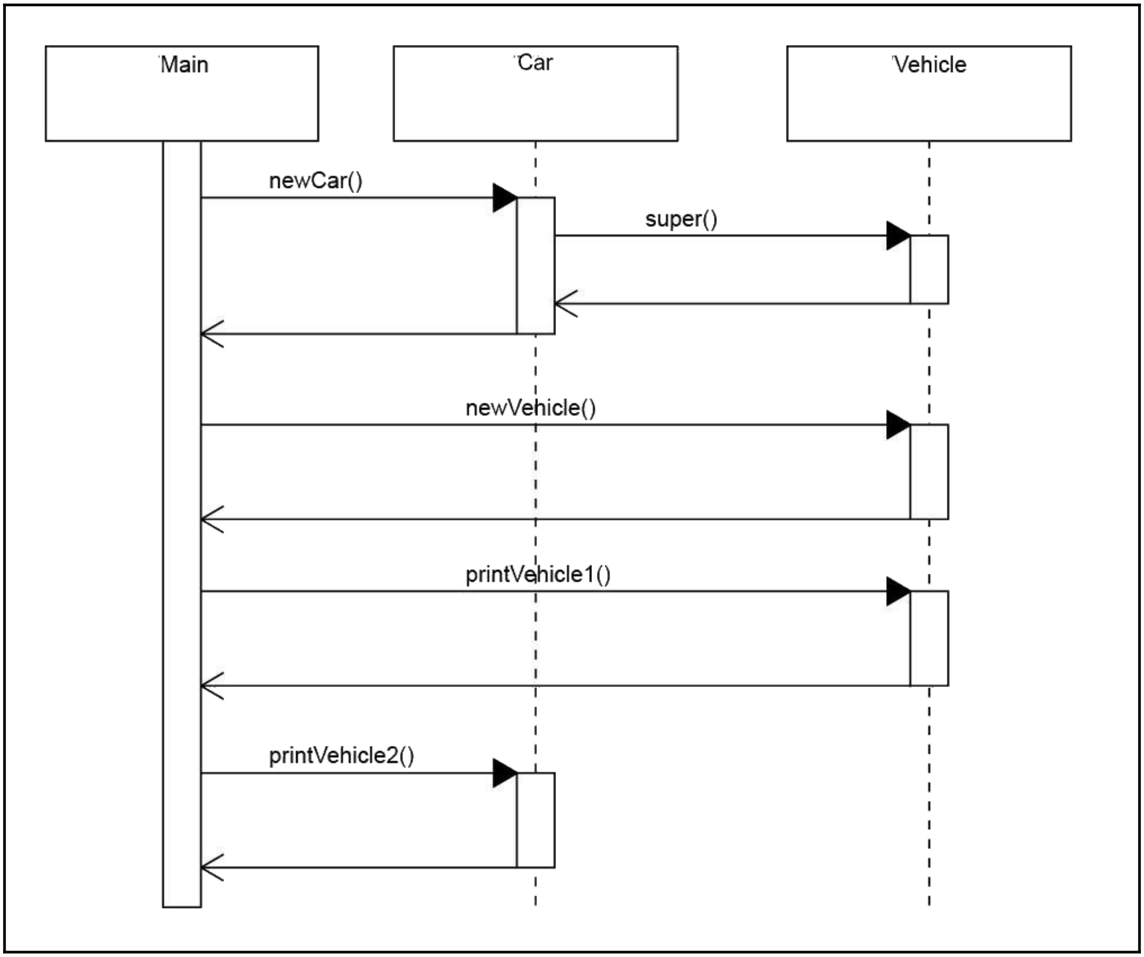
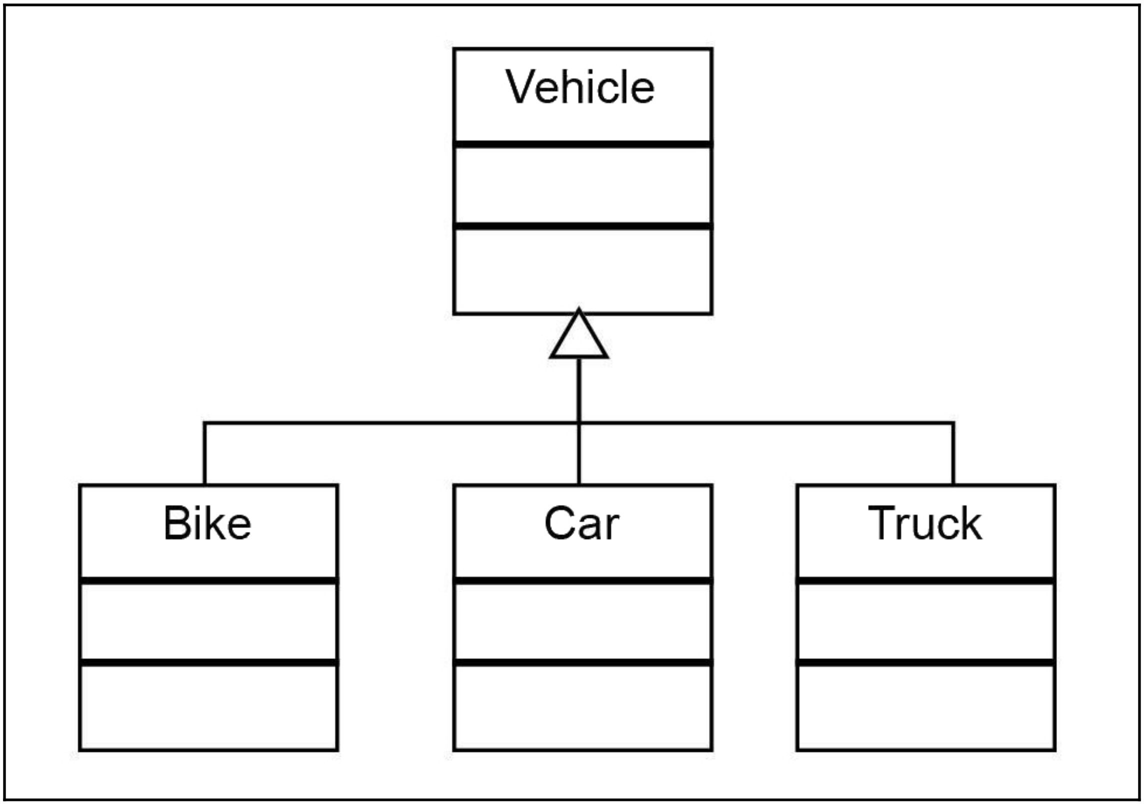
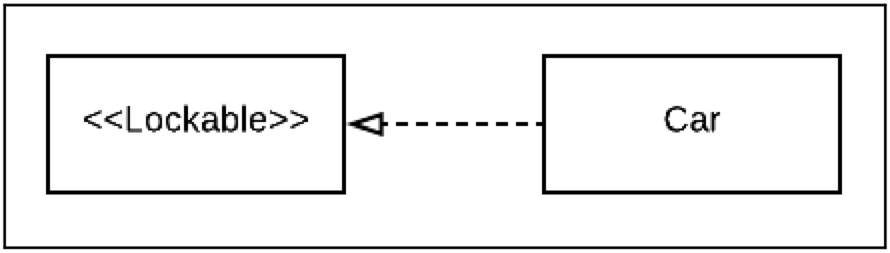
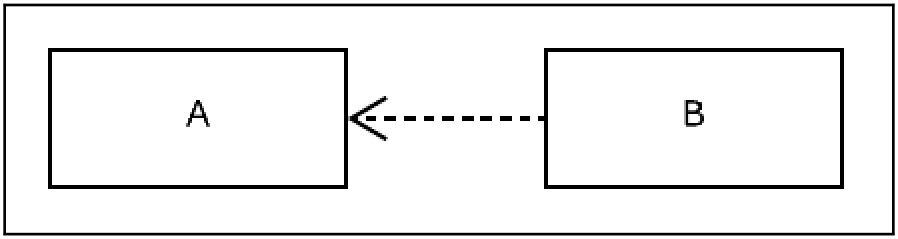
Chapter 1: From Object-Oriented to Functional Programming

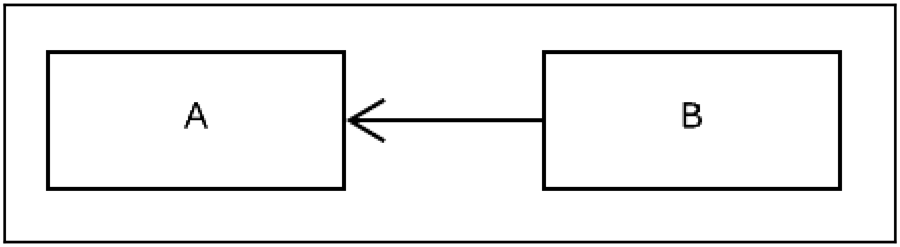


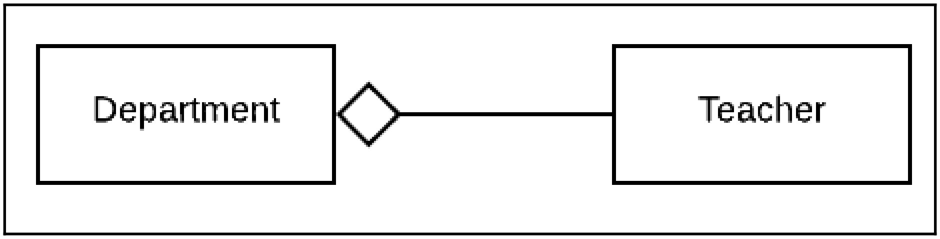

Main
Car
Vehicle
newCar()
super()
newVehicle()
printVehicle1()
printVehicle2()

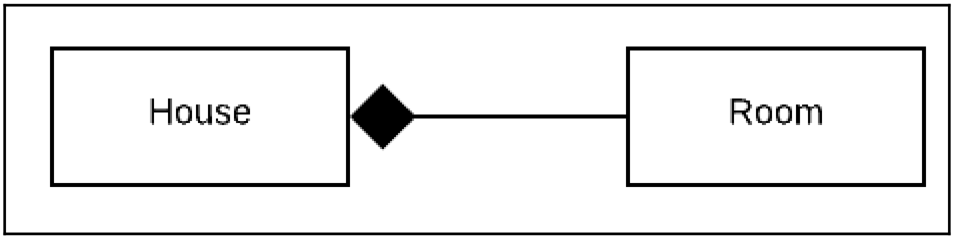

Vehicle
Bike
Car
Truck

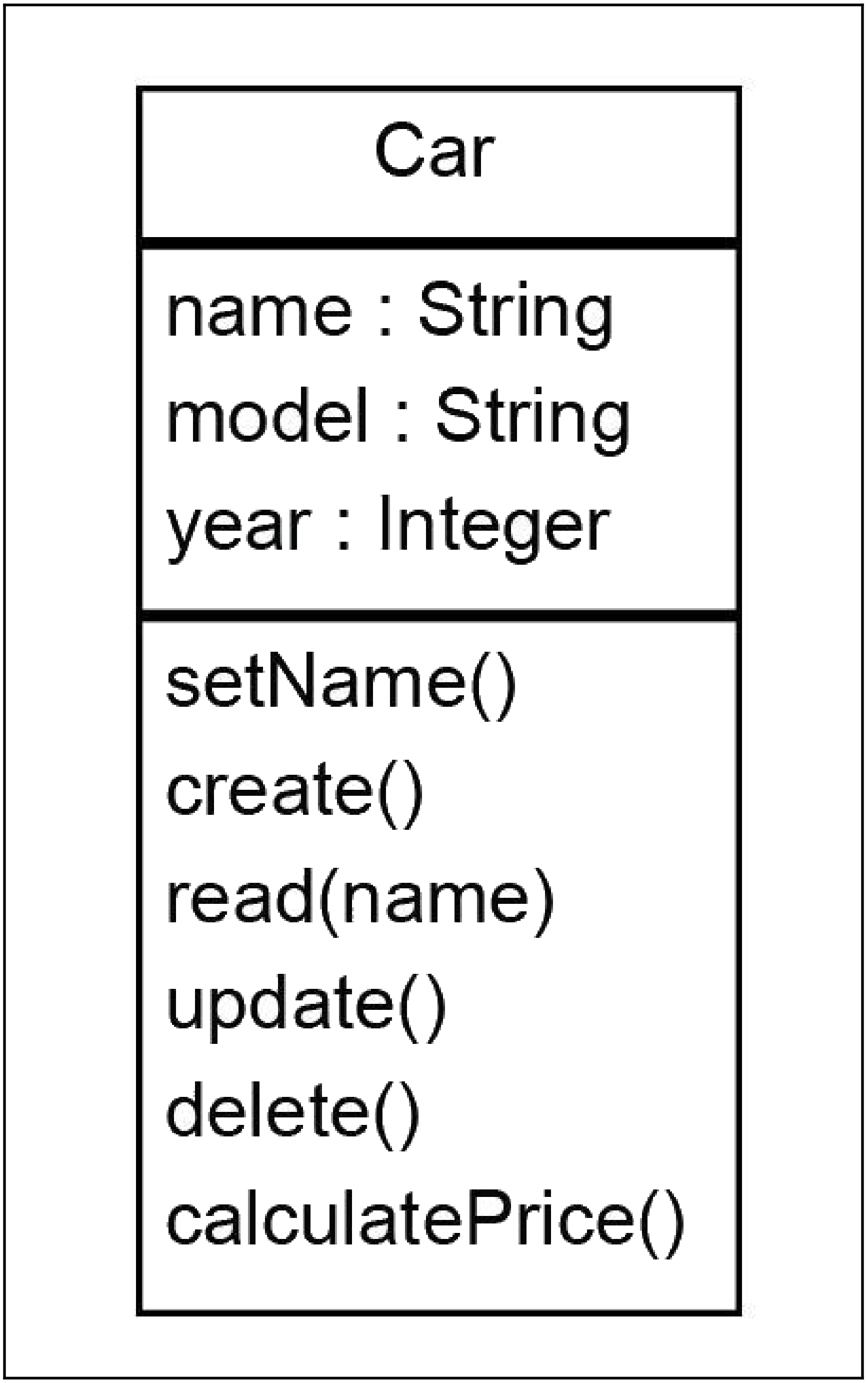

<<Lockable>>
Car

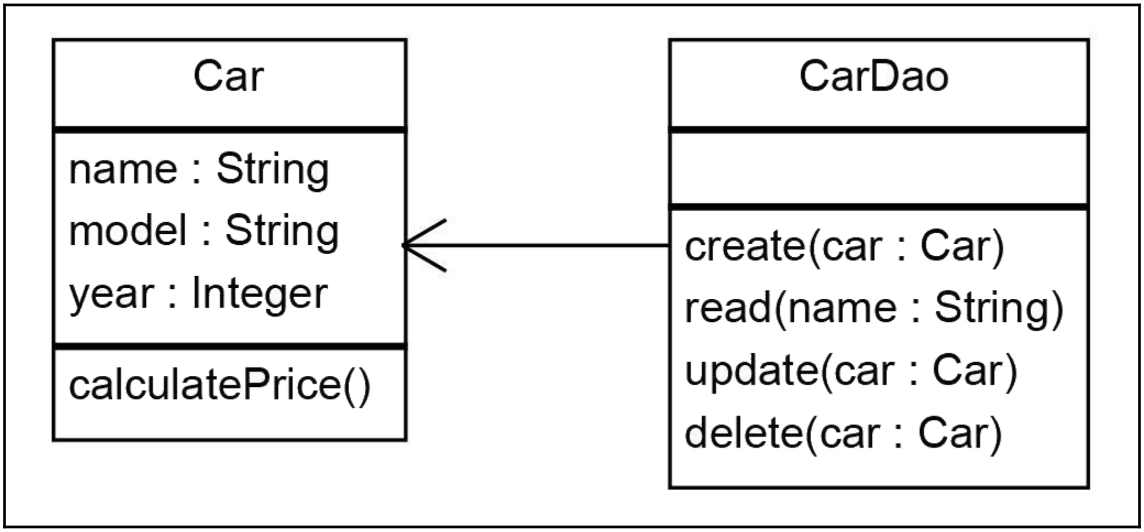

A
B

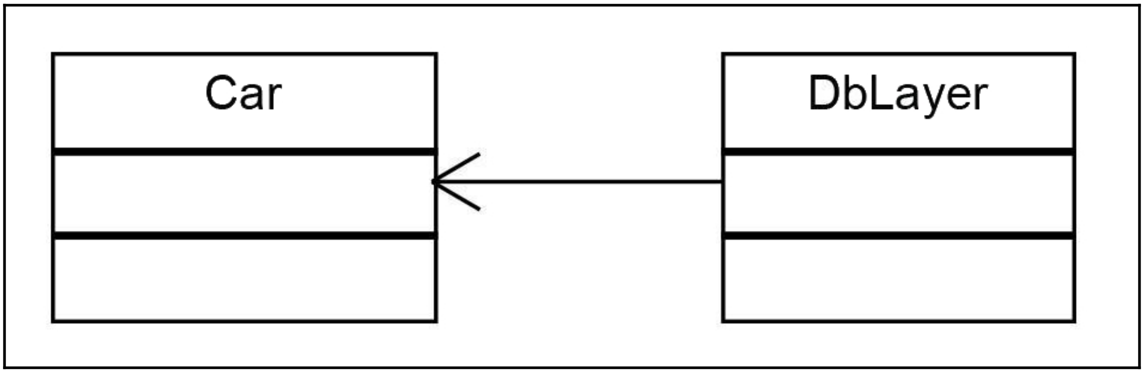

A
B

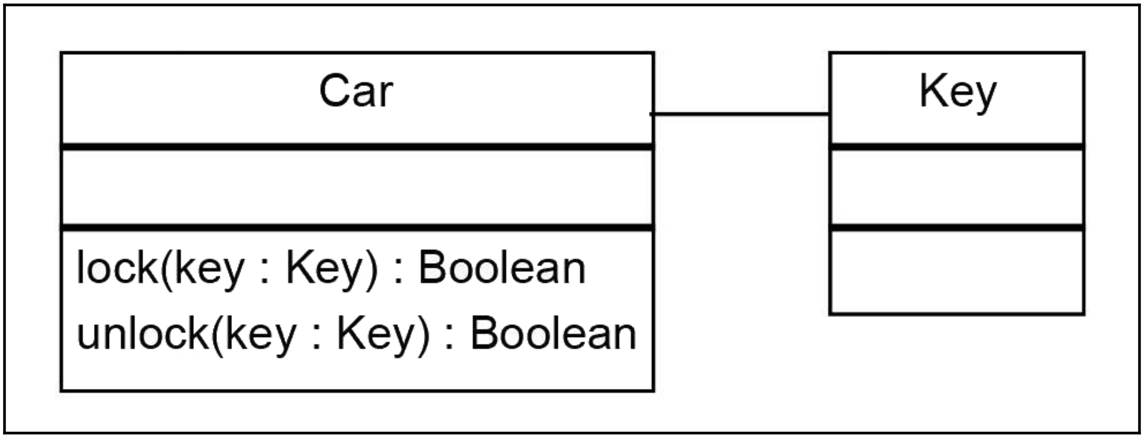

Department
Teacher

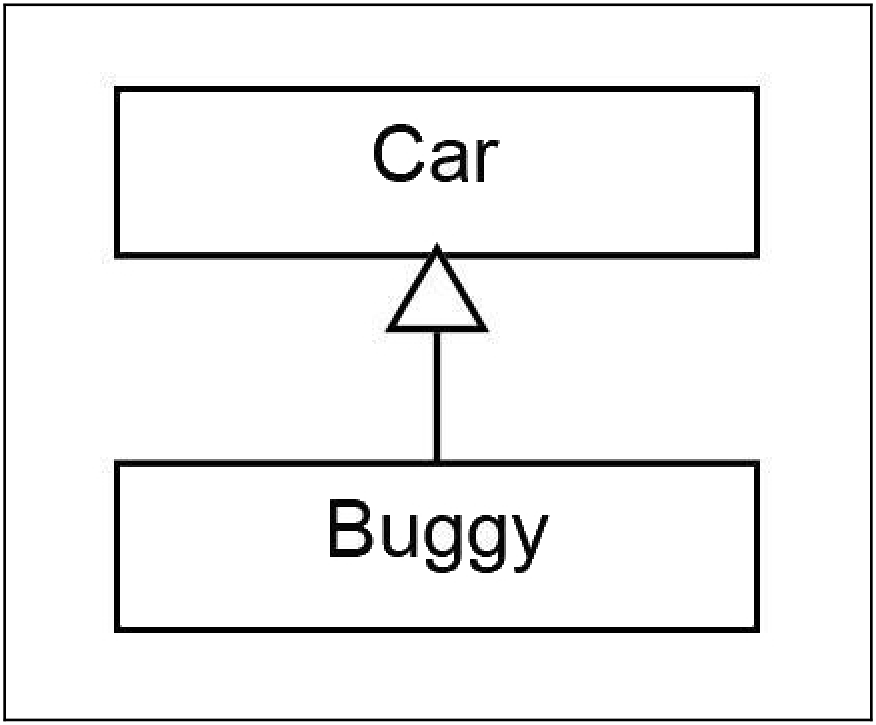

House
Room

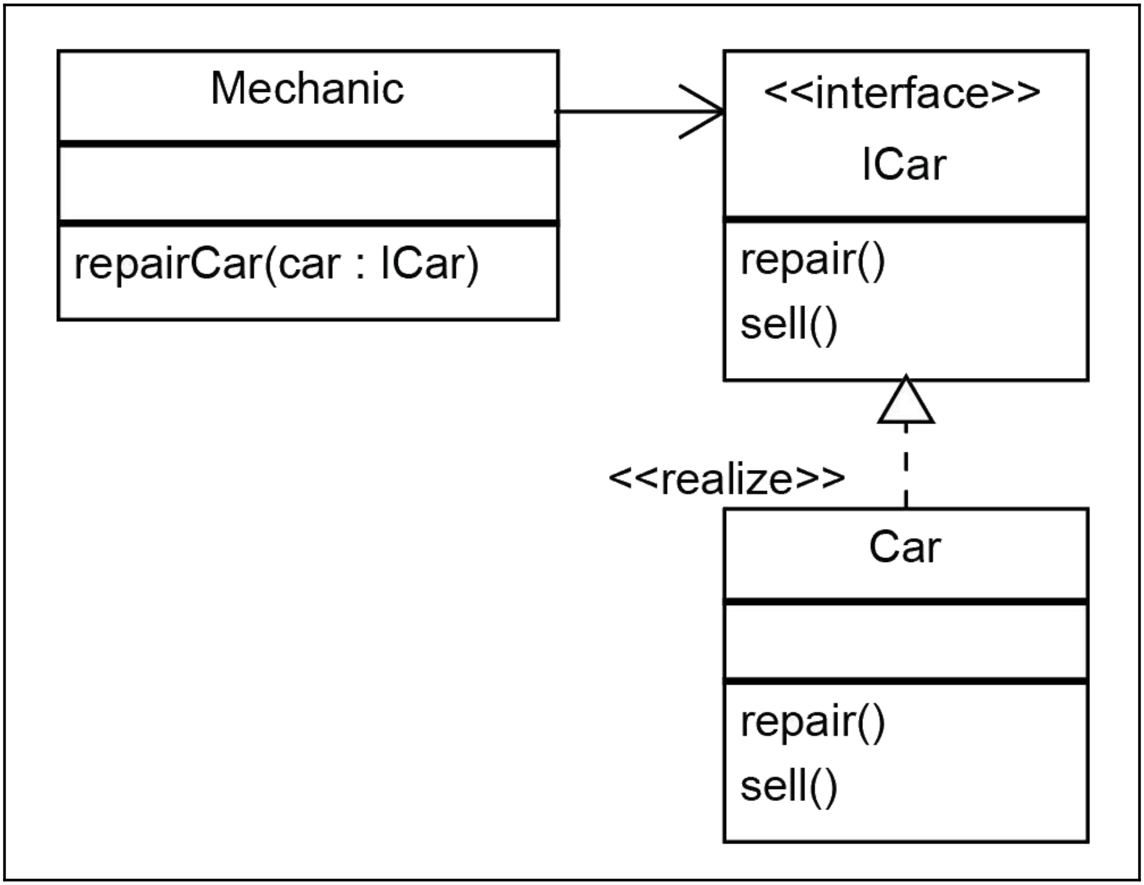

Car
name : String
model : String
year : Integer
setName()
create()
read(name)
update()
delete()
calculatePrice()

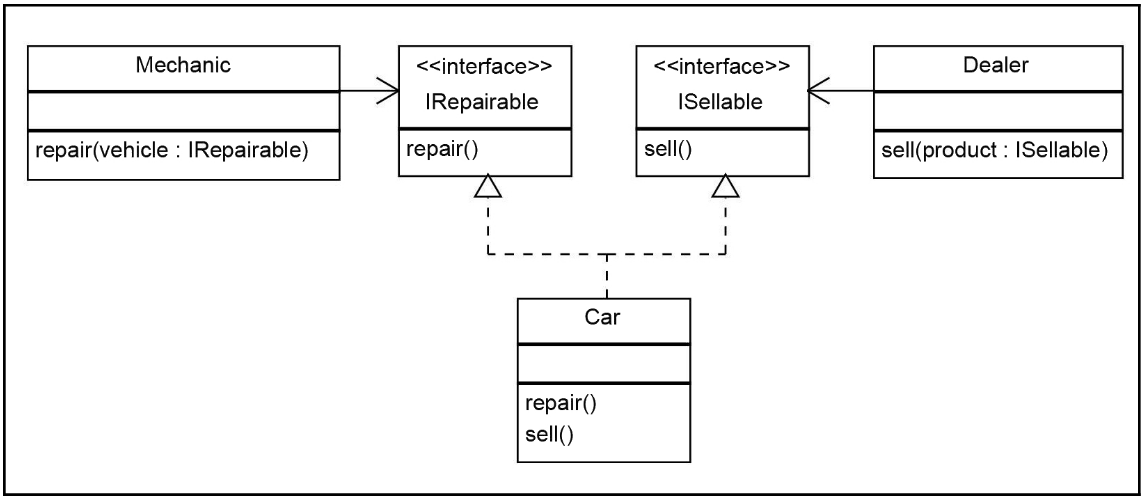

Car
name : String
model : String
year : Integer
calculatePrice()
CarDao
create(car : Car)
read(name : String)
update(car : Car)
delete(car : Car)

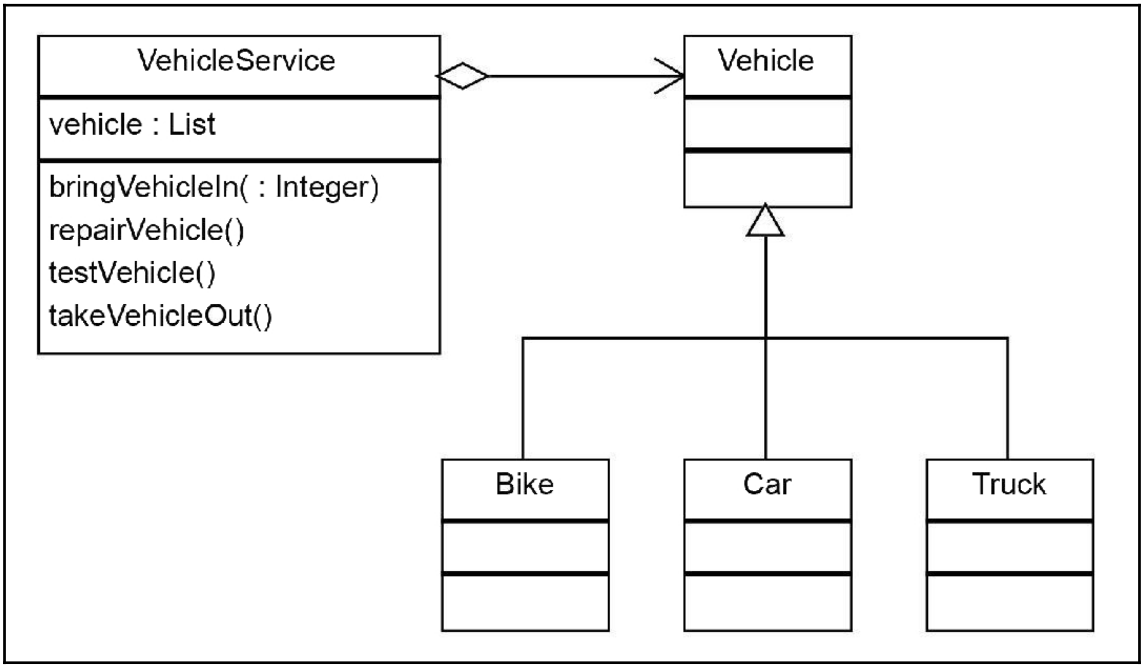

Car
DbLayer


Car
lock(key : Key) : Boolean
unlock(key : Key) : Boolean
Key

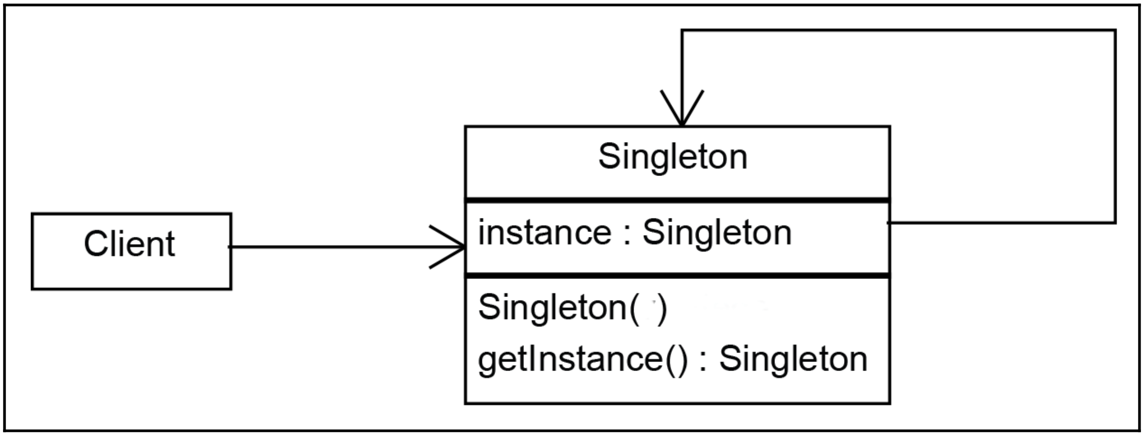

Car
Buggy

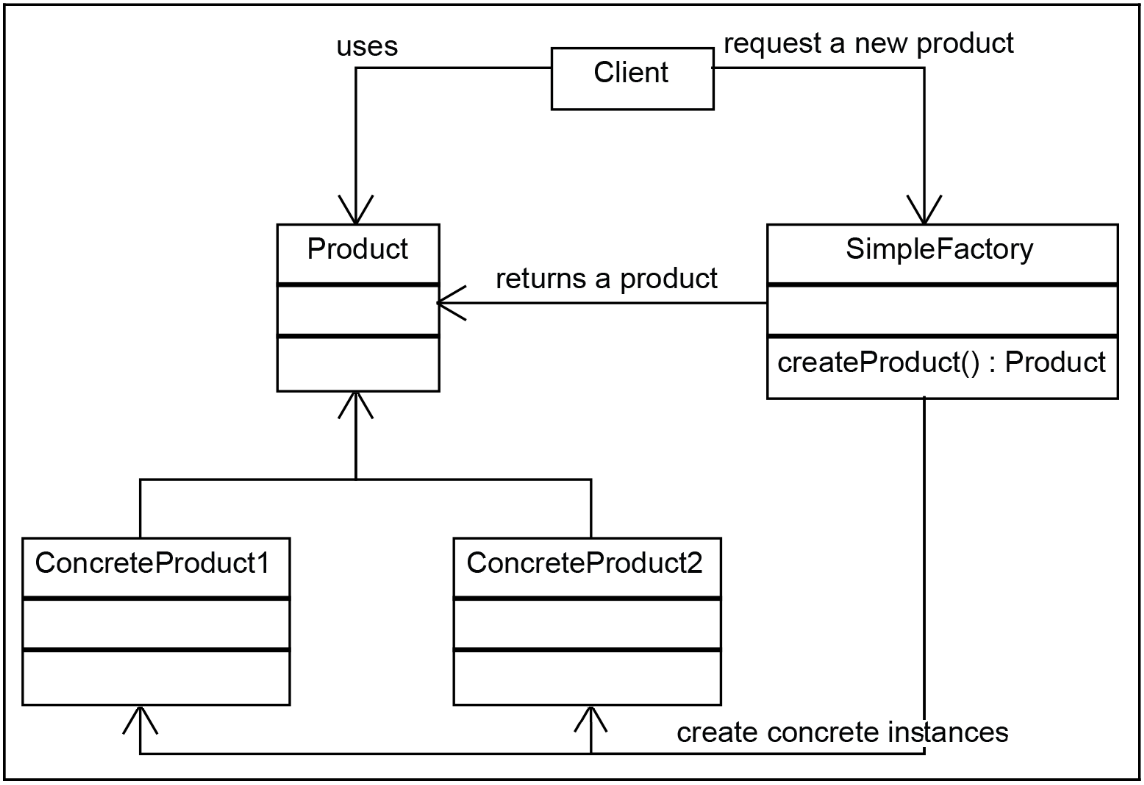

Mechanic
repairCar(car : ICar)
<<interface>>
ICar
repair()
sell()
<<realize>>
Car
repair()
sell()

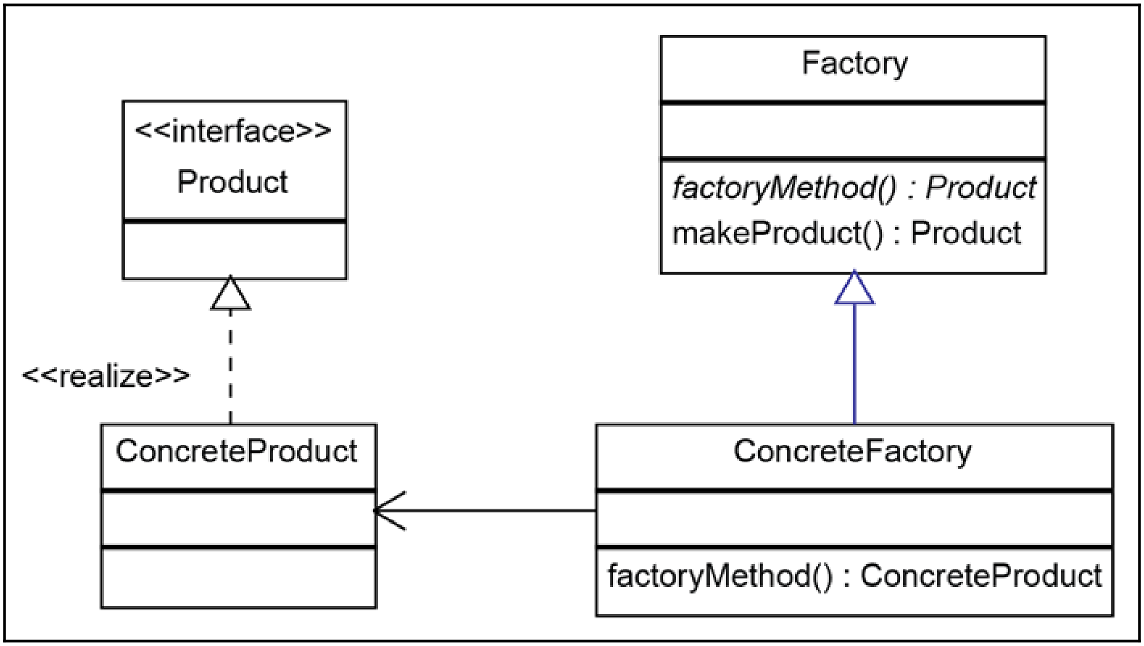

Mechanic
repair(vehicle : IRepairable)
<<interface>>
IRepairable
repair()
<<interface>>
ISellable
sell()
Dealer
sell(product : ISellable)
Car
repair()
sell()

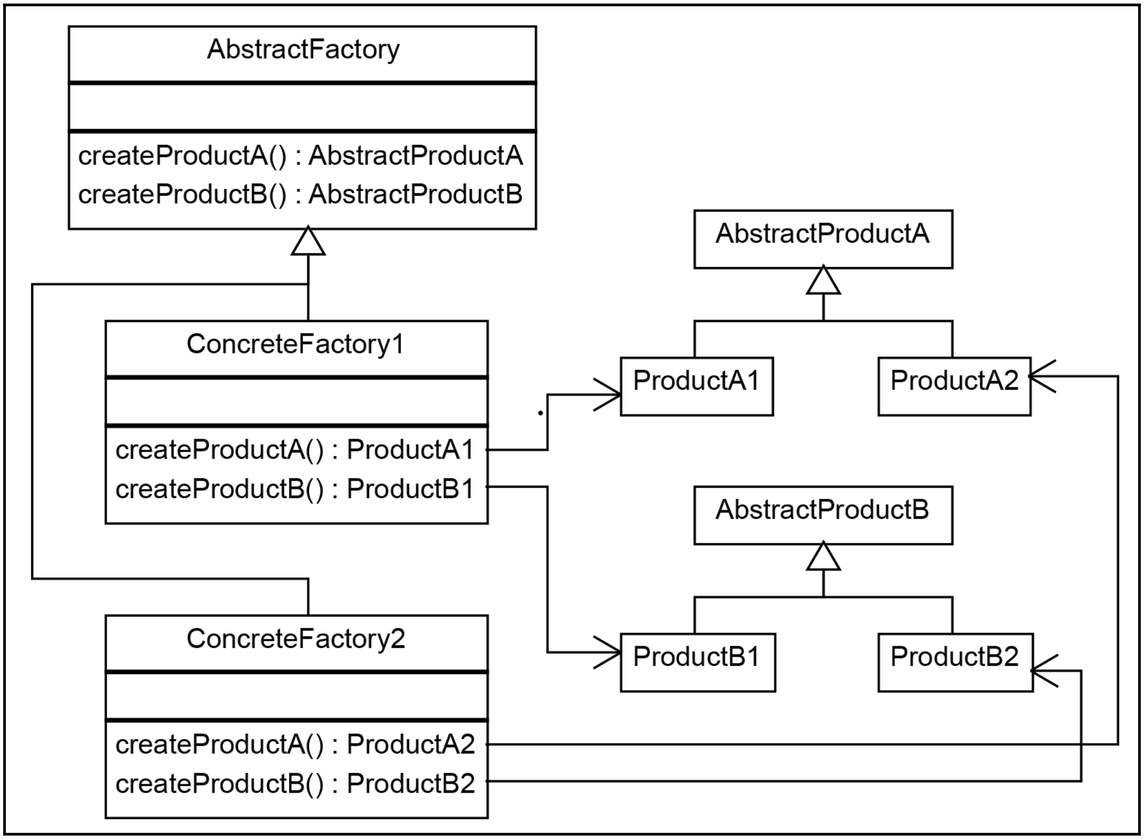

VehicleService
vehicle : List
bringVehicleIn(: Integer)
repairVehicle()
testVehicle()
takeVehicleOut()
Vehicle
Bike
Car
Truck

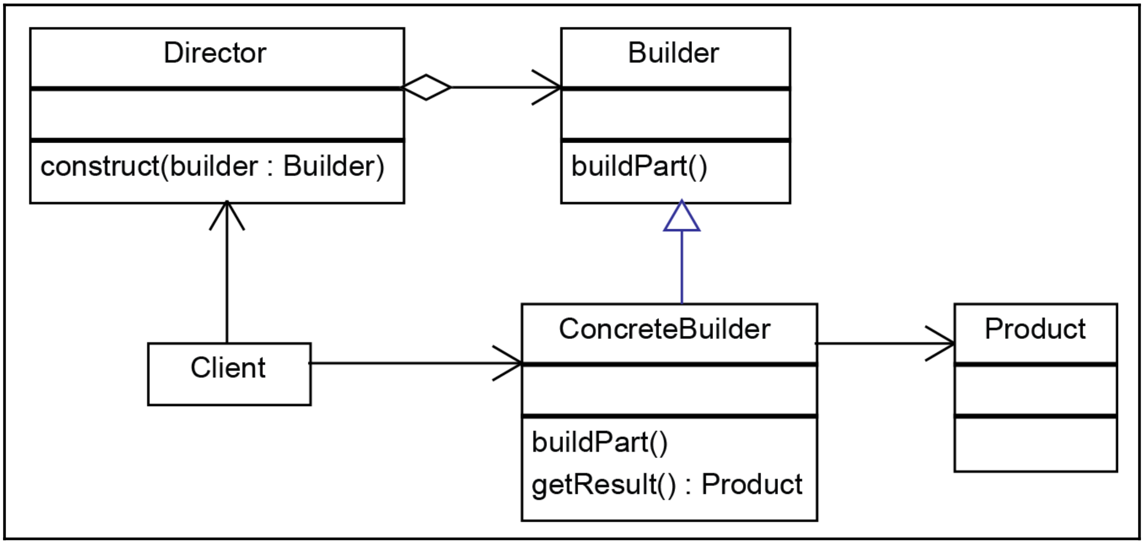
Chapter 2: Creational Patterns

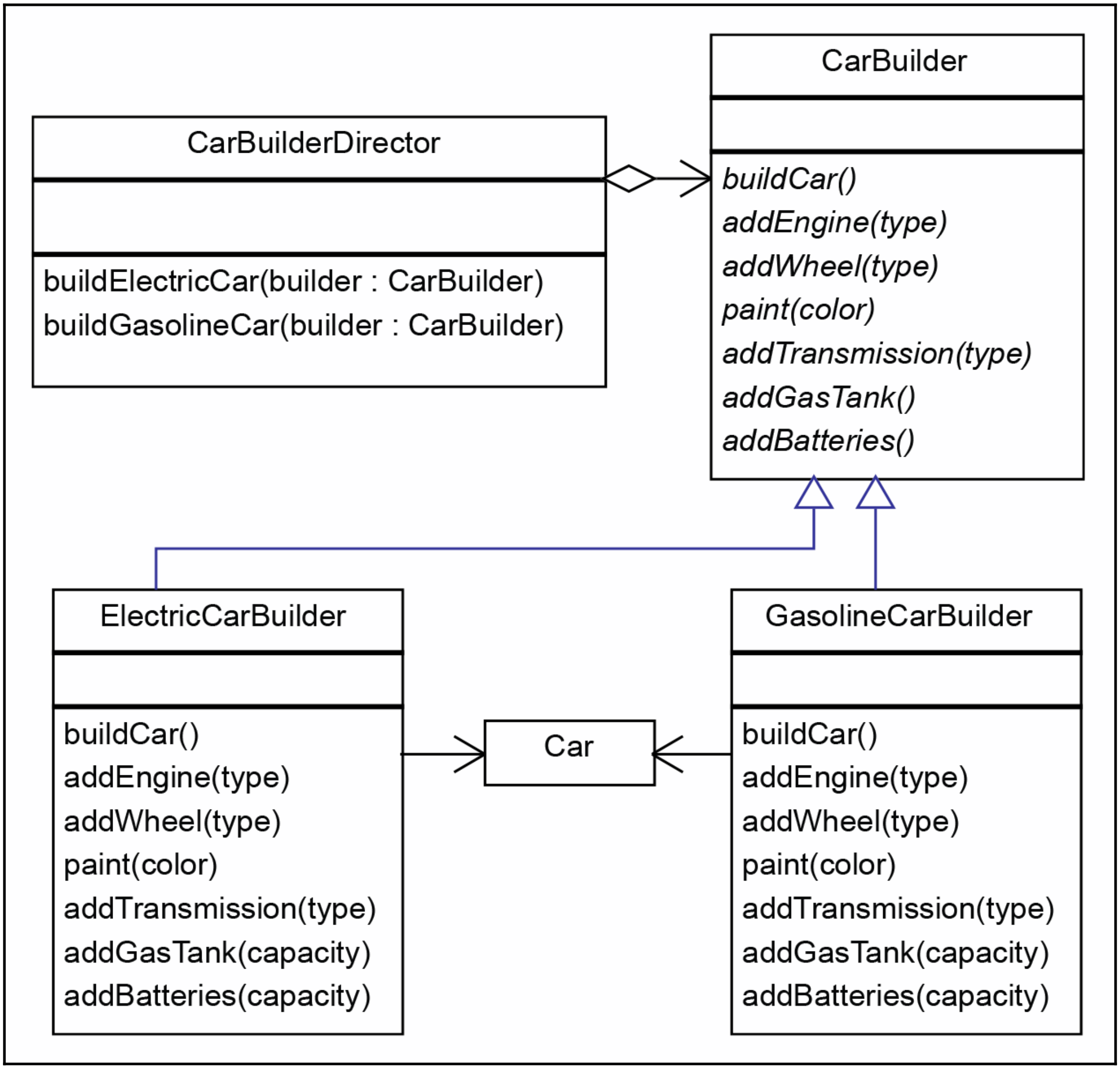


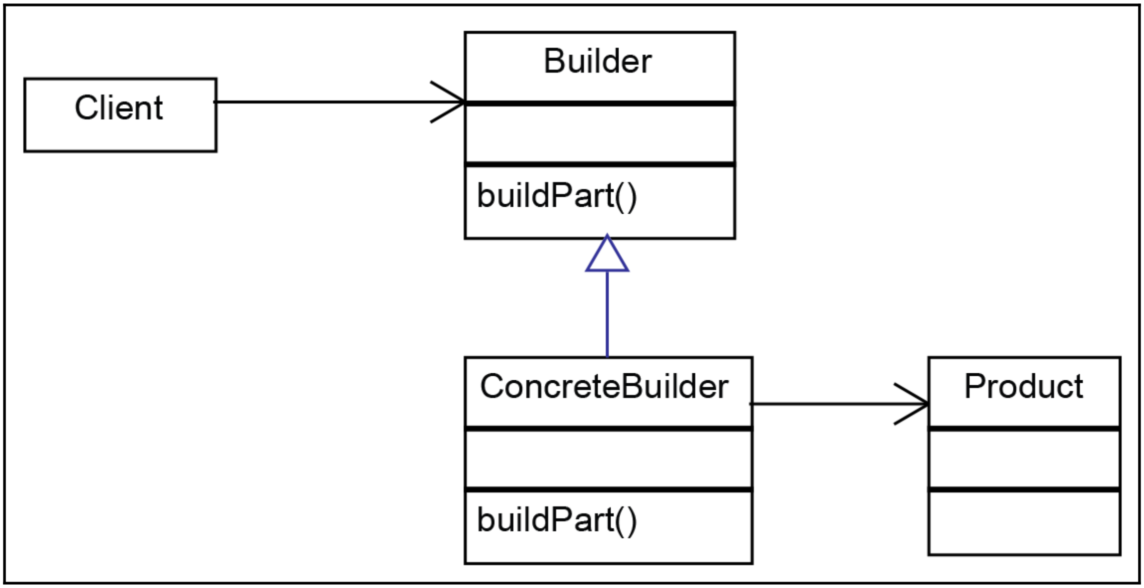


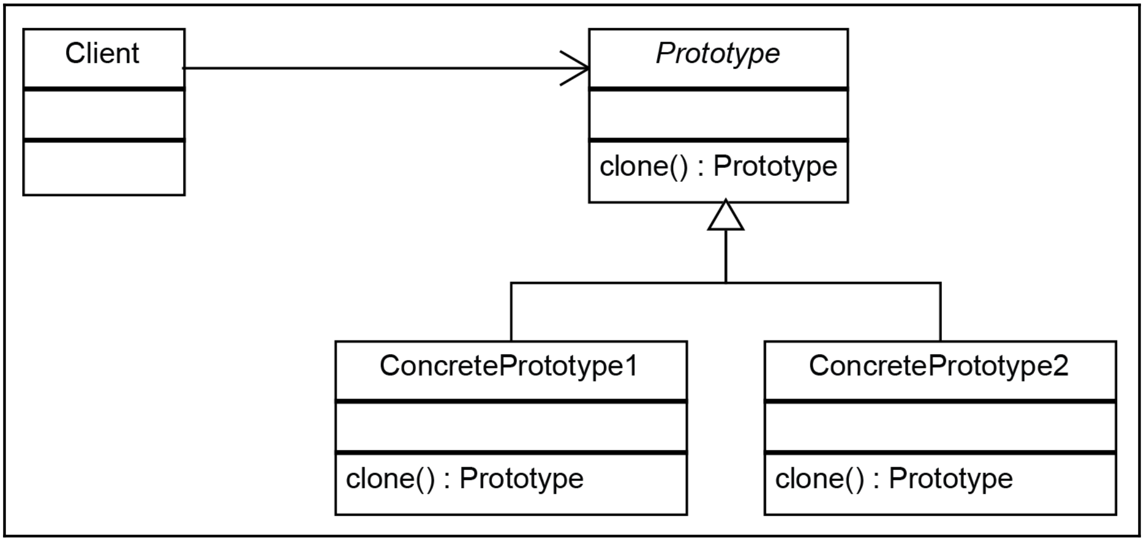

<<interface>>
Product
Factory
factoryMethod() : Product
makeProduct() : Product
<<realize>>
ConcreteProduct
ConcreteFactory
factoryMethod() : ConcreteProduct

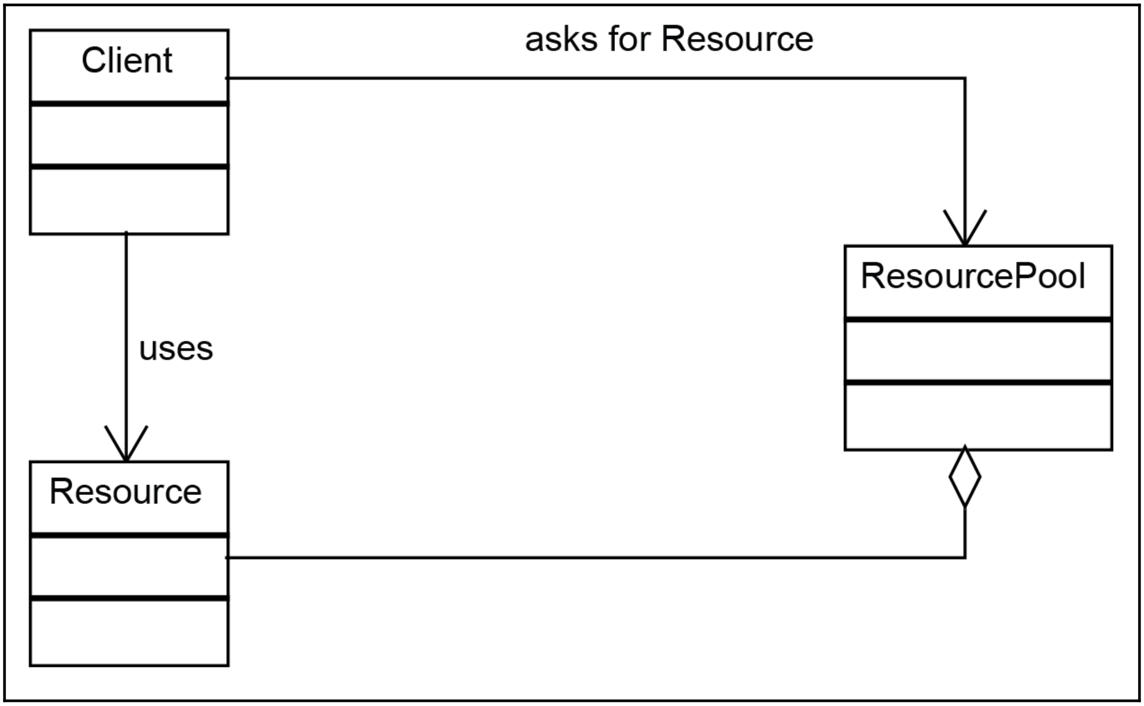

AbstractFactory
createProductA() : AbstractProductA
createProductB() : AbstractProductB
AbstractProductA
ConcreteFactory1
ProductA1
ProductA2
createProductA() : ProductA1
createProductB() : ProductB1
AbstractProductB
ConcreteFactory2
ProductB1
ProductB2
createProductA() : ProductA2
createProductB() : ProductB2


Director
construct(builder : Builder)
Builder
buildPart()
Client
ConcreteBuilder
buildPart()
getResult() : Product
Product

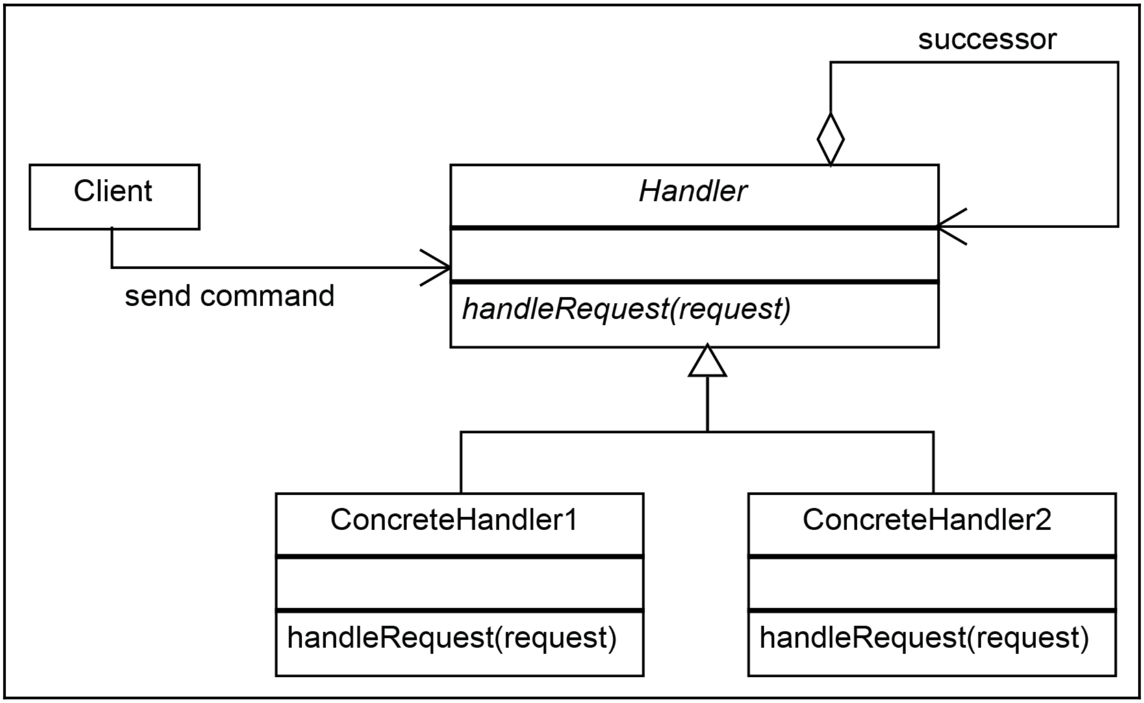

CarBuilder
buildCar()
addEngine(type)
addWheel(type)
paint(color)
addTransmission(type)
addGasTank()
addBatteries()
CarBuilderDirector
buildElectricCar(builder : CarBuilder)
buildGasolineCar(builder : CarBuilder)
ElectricCarBuilder
buildCar()
addEngine(type)
addWheel(type)
paint(color)
addTransmission(type)
addGasTank(capacity)
addBatteries(capacity)
Car
GasolineCarBuilder
buildCar()
addEngine(type)
addWheel(type)
paint(color)
addTransmission(type)
addGasTank(capacity)
addBatteries(capacity)

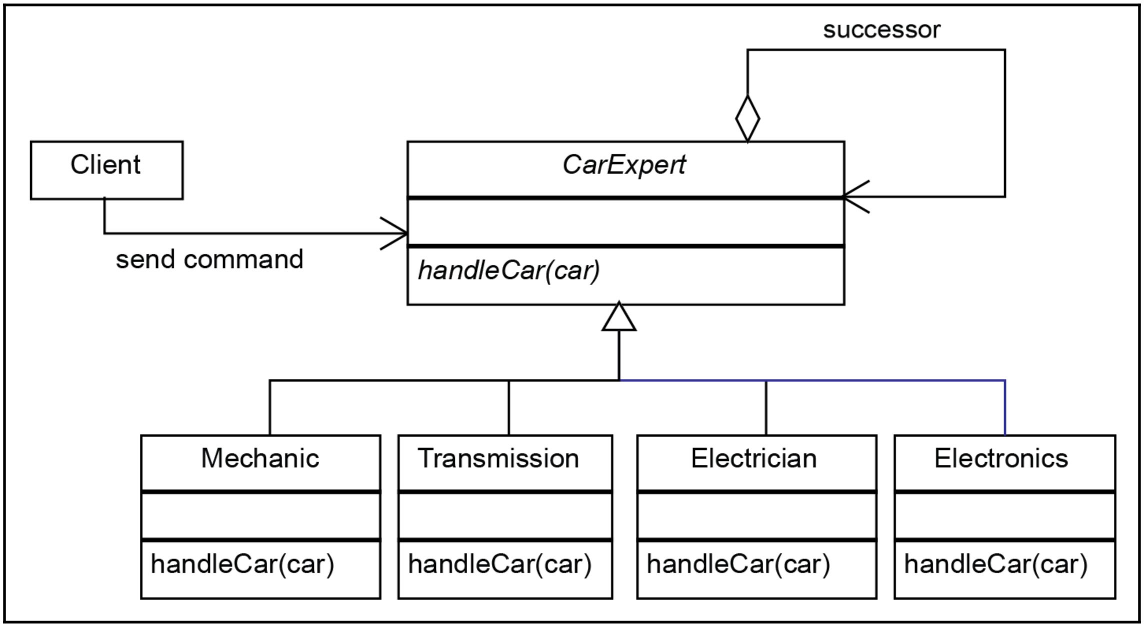

Client
Builder
buildPart()
ConcreteBuilder
buildPart()
Product

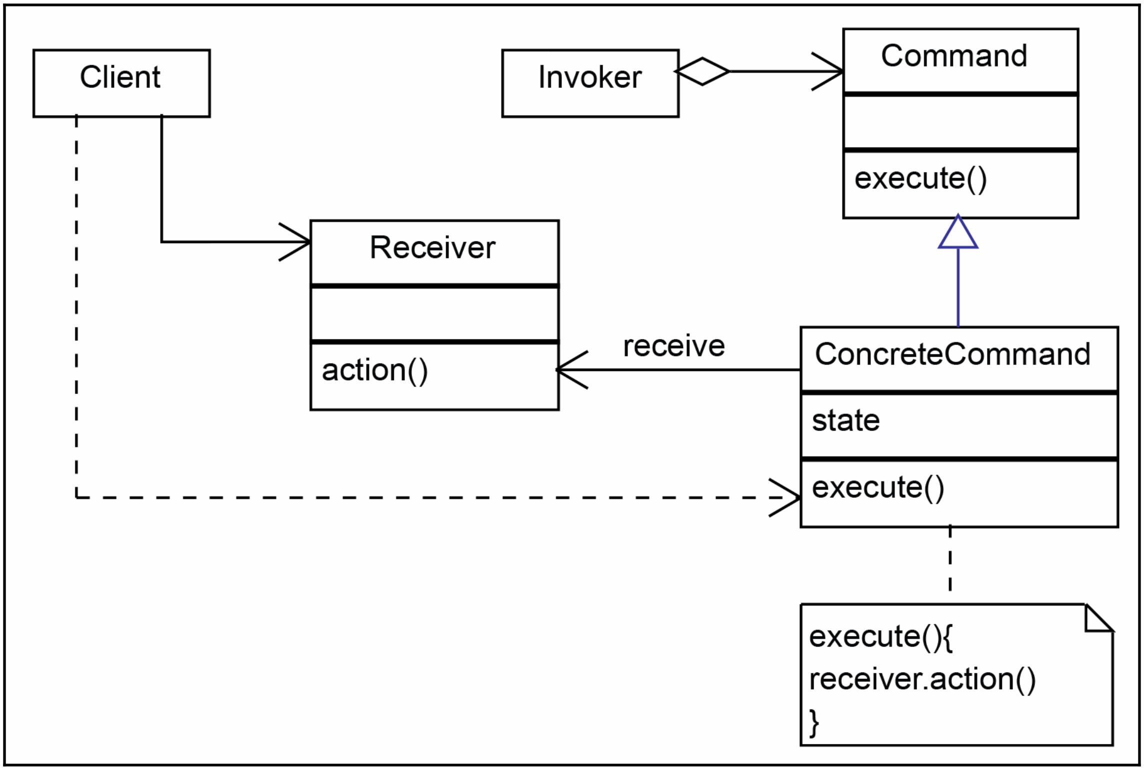

Client
Prototype
clone() : Prototype
ConcretePrototype1
clone() : Prototype
ConcretePrototype2
clone() : Prototype

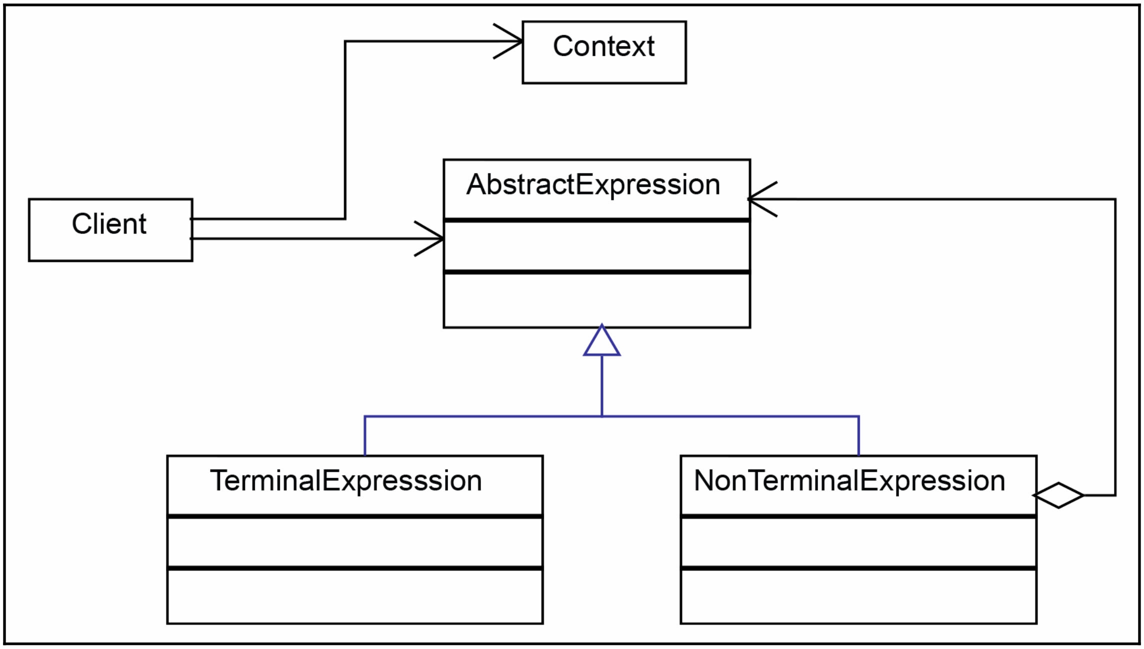

Client
asks for Resource
ResourcePool
uses
Resource

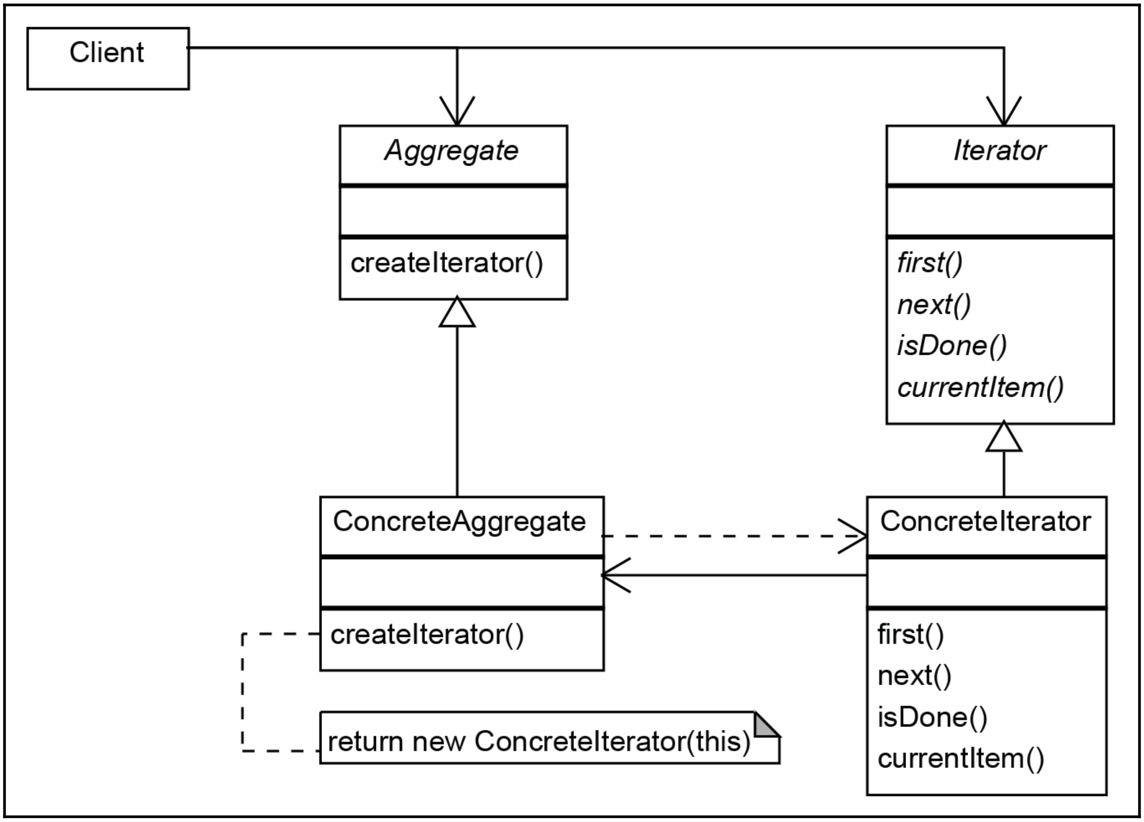
Chapter 3: Behavioral Patterns

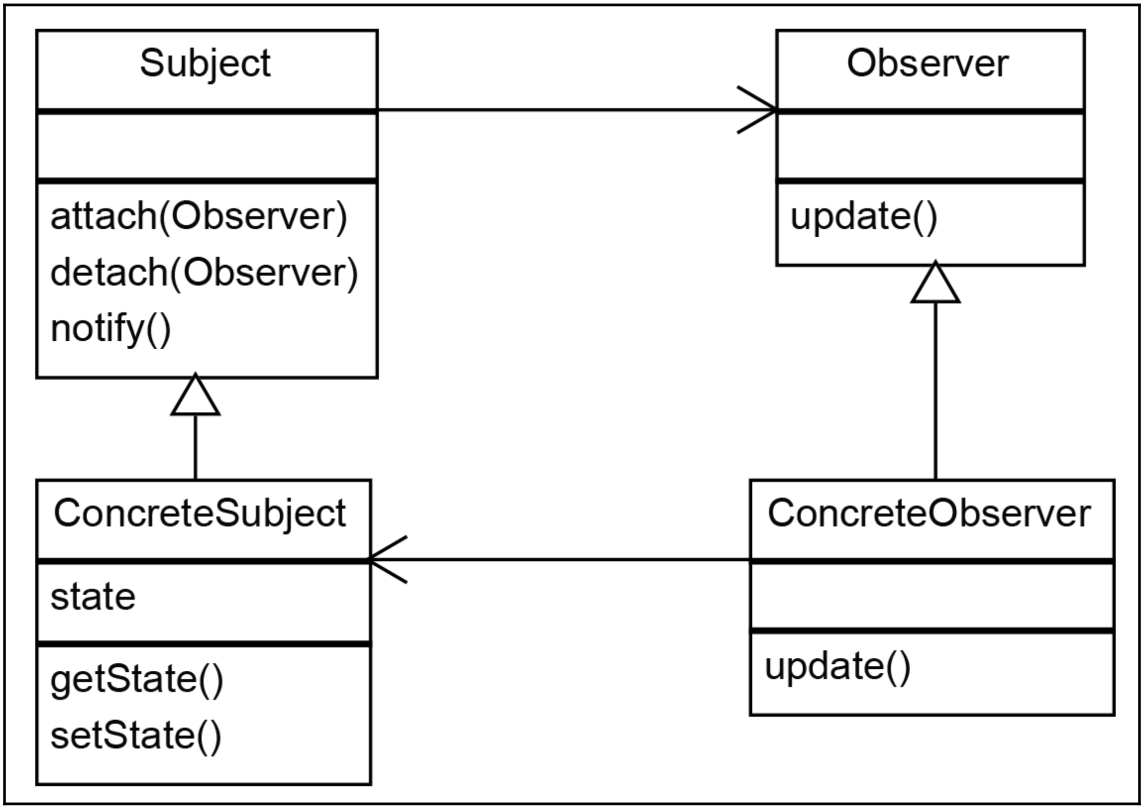


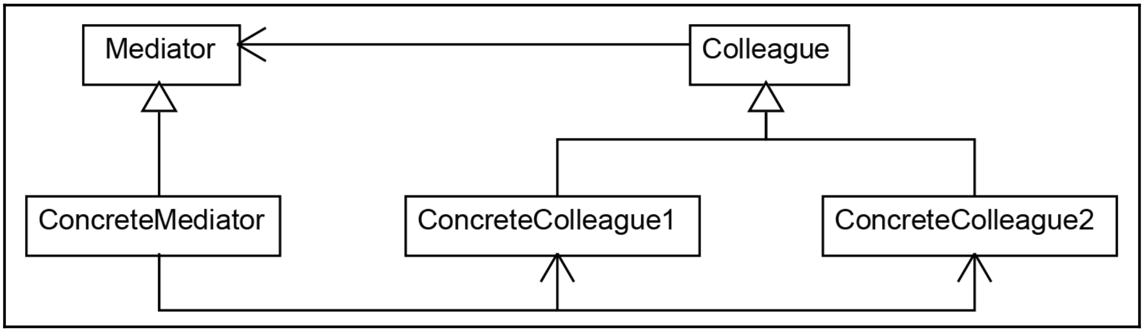

successor
Client
CarExpert
handleCar(car)
send command
Mechanic
handleCar(car)
Transmission
handleCar(car)
Electrician
handleCar(car)
Electronics
handleCar(car)

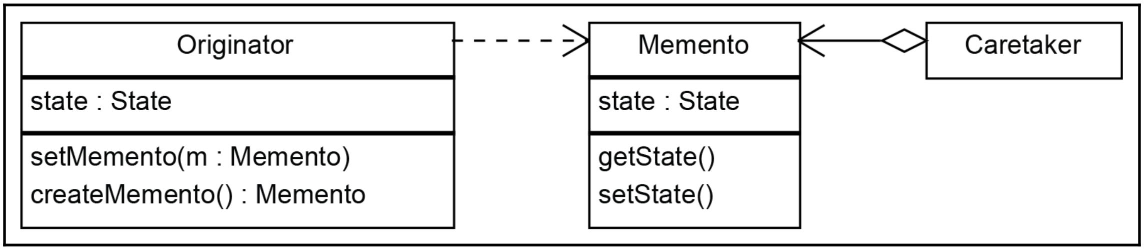

Client
Invoker
Command
execute()
Receiver
action()
receive
ConcreteCommand
state
execute()
execute(){
receiver.action()
}

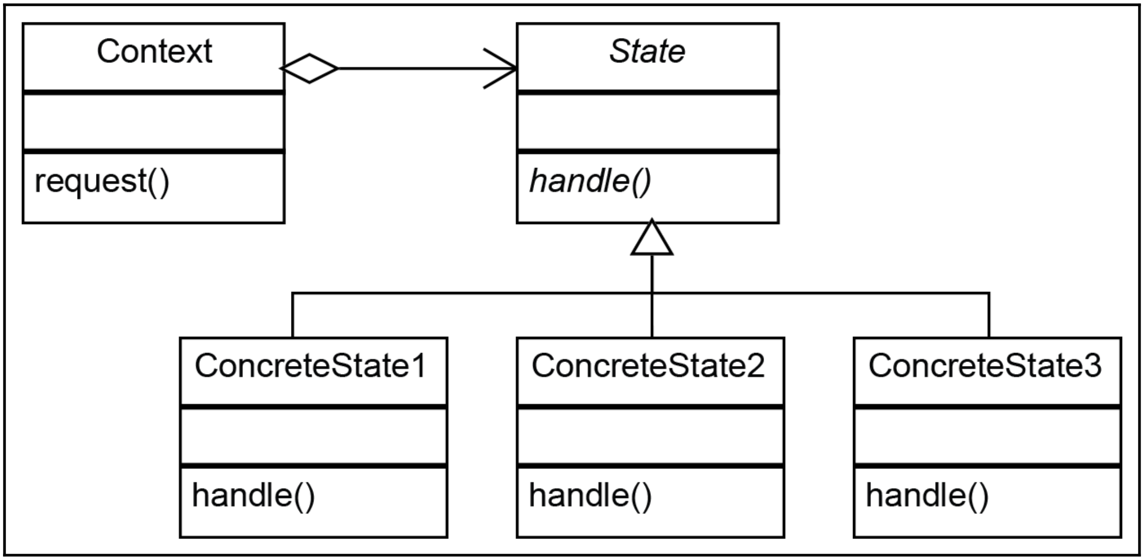

Context
Client
AbstractExpression
TerminalExpresssion
NonTerminalExpression

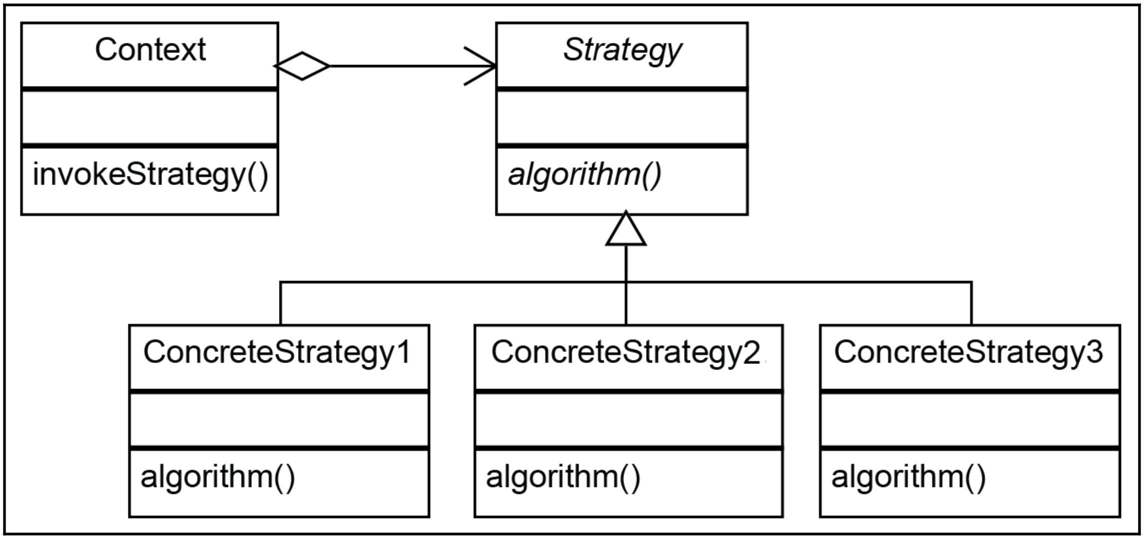

Client
Aggregate
createIterator()
Iterator
first()
next()
isDone()
currentItem()
ConcreteAggregate
createIterator()
ConcreteIterator
first()
next()
isDone()
currentItem()
return new ConcreteIterator(this)

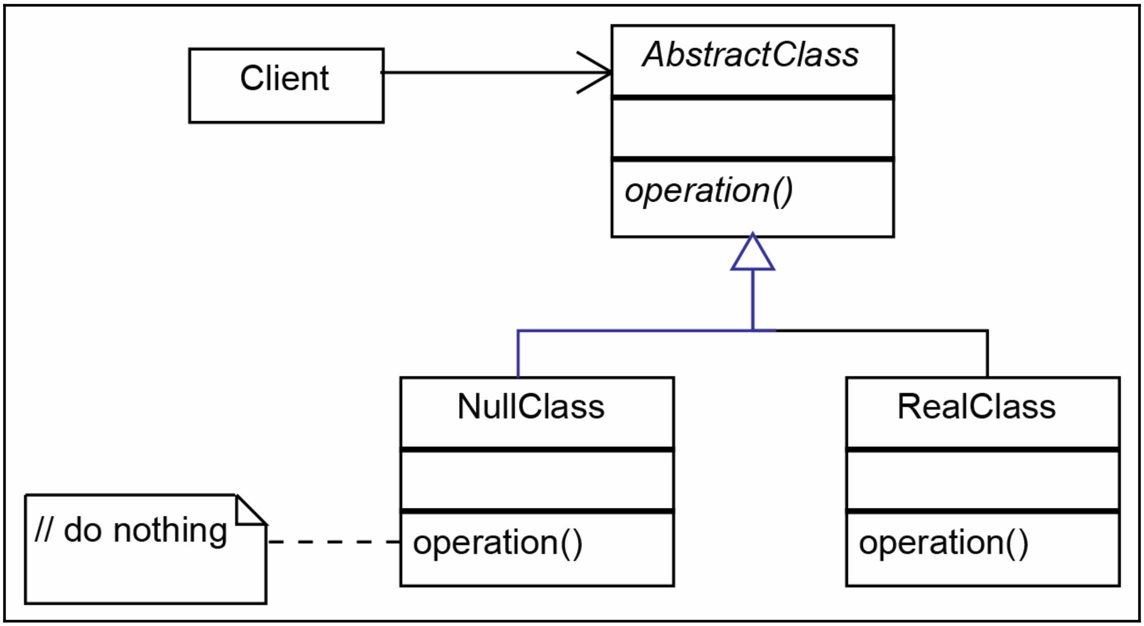

Subject
attach(Observer)
detach(Observer)
notify()
Observer
update()
ConcreteSubject
state
getState()
setState()
ConcreteObserver
update()

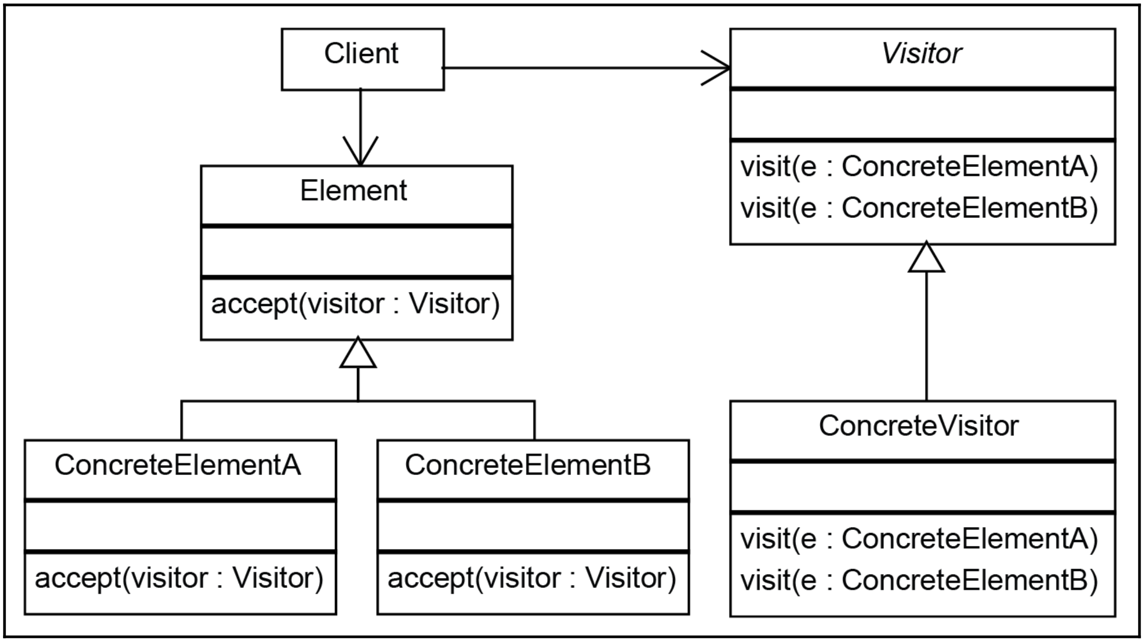

Mediator
Colleague
ConcreteMediator
ConcreteColleague1
ConcreteColleague2


Originator
state : State
setMemento(m : Memento)
createMemento() : Memento
Memento
state : State
getState()
setState()
Caretaker

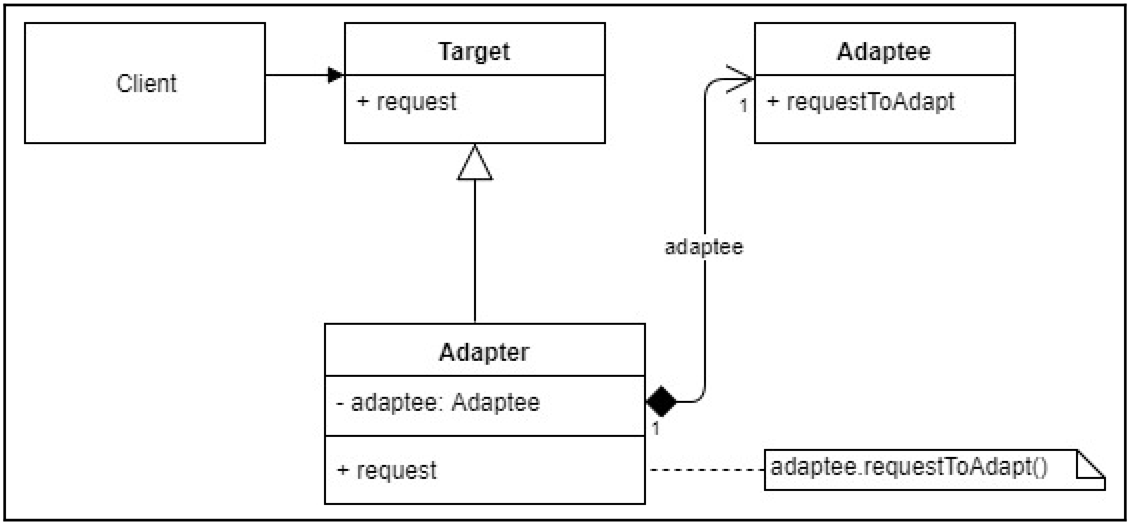

Context
request()
State
handle()
ConcreteState1
handle()
ConcreteState2
handle()
ConcreteState3
handle()

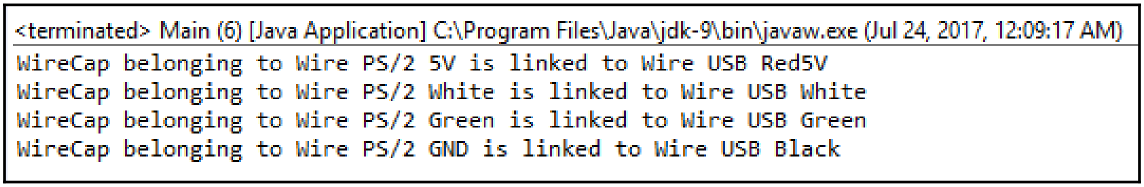

Context
invokeStrategy()
Strategy
algorithm()
ConcreteStrategy1
algorithm()
ConcreteStrategy2
algorithm()
ConcreteStrategy3
algorithm()


Client
AbstractClass
operation()
NullClass
operation()
RealClass
operation()
// do nothing

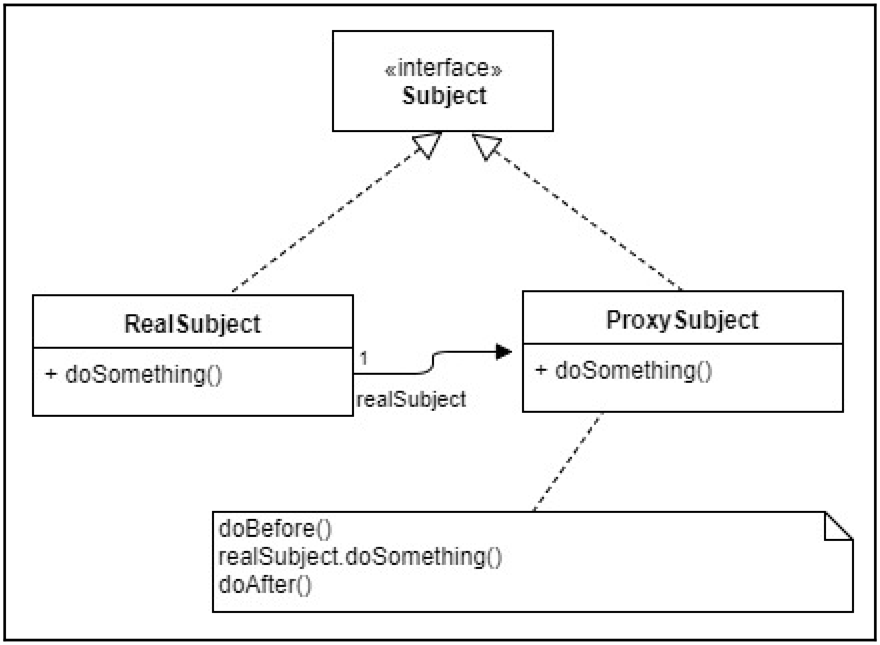

Client
Visitor
Element
accept(visitor : Visitor)
visit(e : ConcreteElementA)
visit(e : ConcreteElementB)
ConcreteElementA
accept(visitor : Visitor)
ConcreteElementB
accept(visitor : Visitor)
ConcreteVisitor
visit(e : ConcreteElementA)
visit(e : ConcreteElementB)

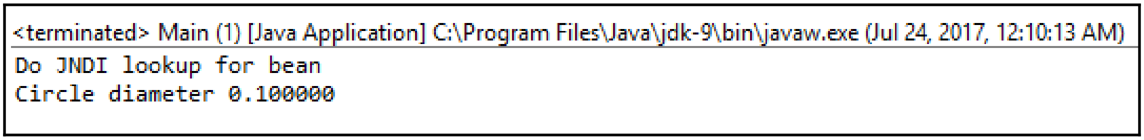
Chapter 4: Structural Patterns



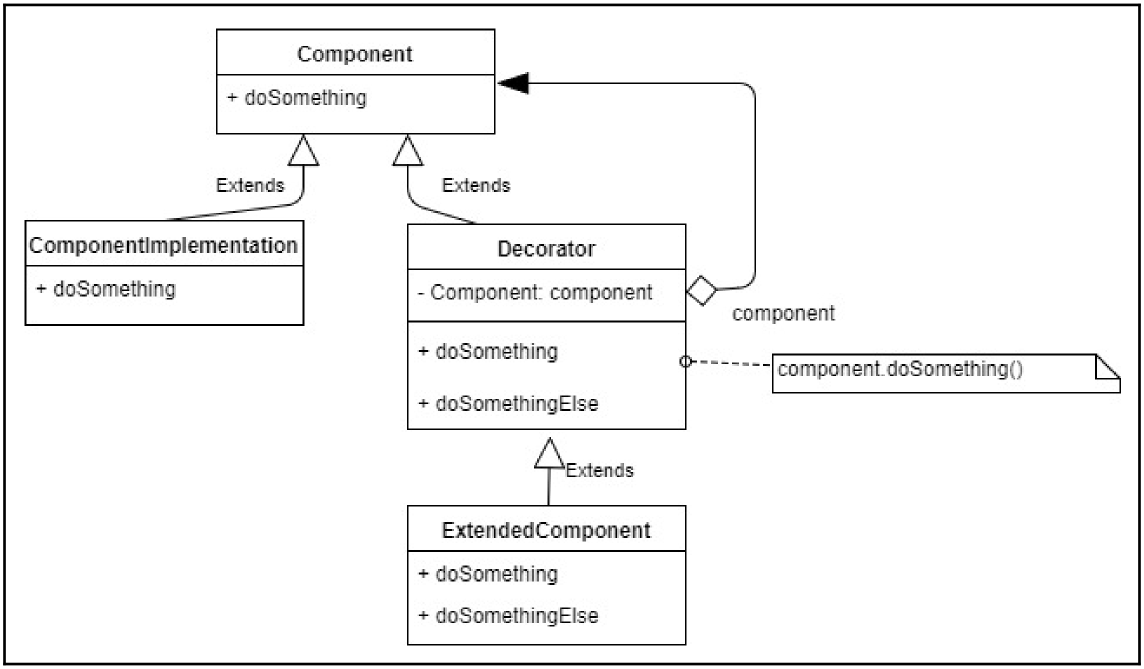


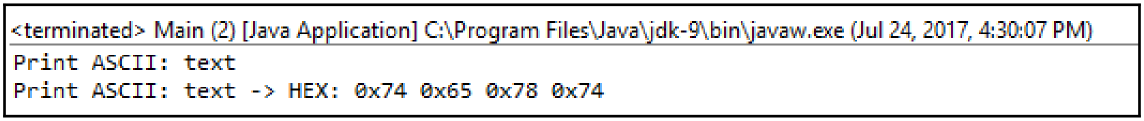
```
<terminated> Main (6) [Java Application] C:\Program Files\Java\jdk-9\bin\javaw.exe (Jul 24, 2017, 12:09:17 AM)
WireCap belonging to Wire PS/2 5V is linked to Wire USB Red5V
WireCap belonging to Wire PS/2 White is linked to Wire USB White
WireCap belonging to Wire PS/2 Green is linked to Wire USB Green
WireCap belonging to Wire PS/2 GND is linked to Wire USB Black
```



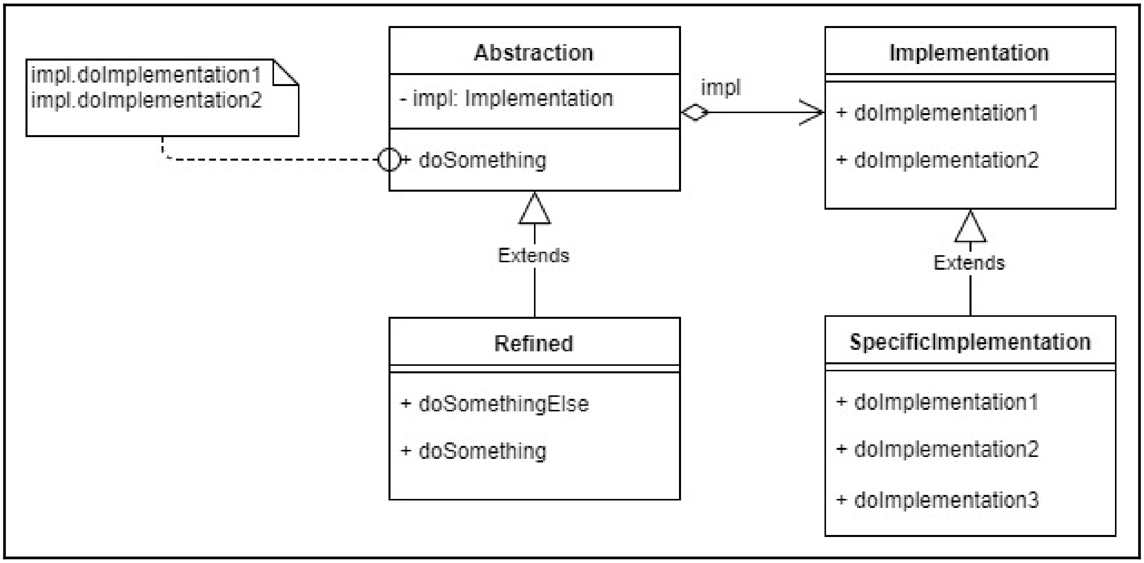


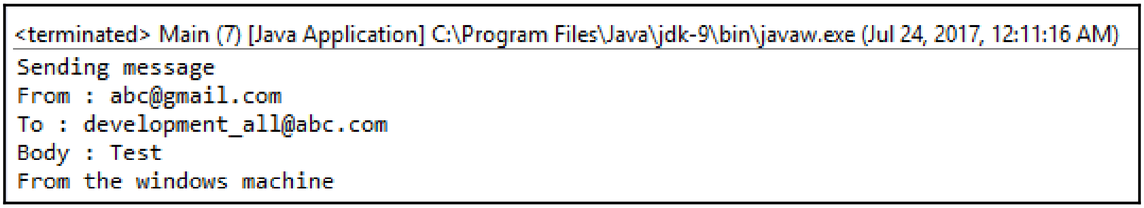
```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-9\bin\javaw.exe (Jul 24, 2017, 12:10:13 AM)
Do JNDI lookup for bean
Circle diameter 0.100000
```

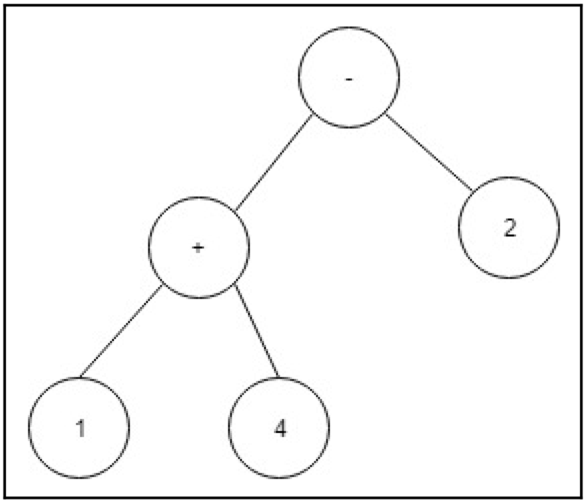


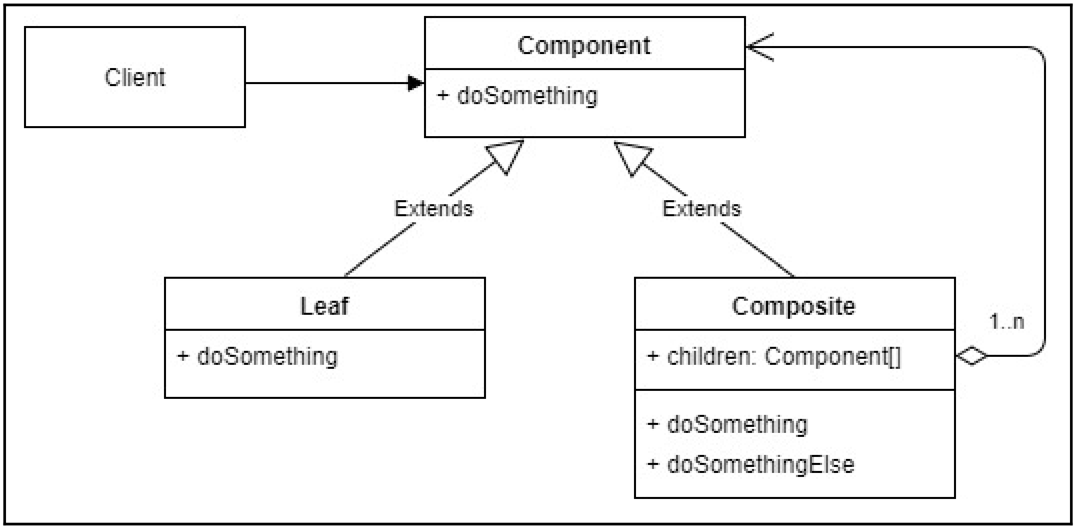


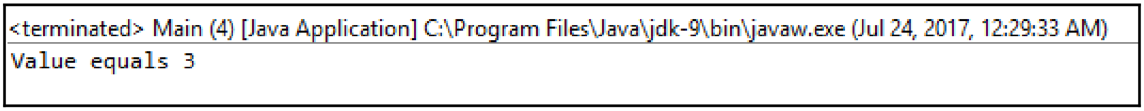
```
<terminated> Main (2) [Java Application] C:\Program Files\Java\jdk-9\bin\javaw.exe (Jul 24, 2017, 4:30:07 PM)
Print ASCII: text
Print ASCII: text -> HEX: 0x74 0x65 0x78 0x74
```



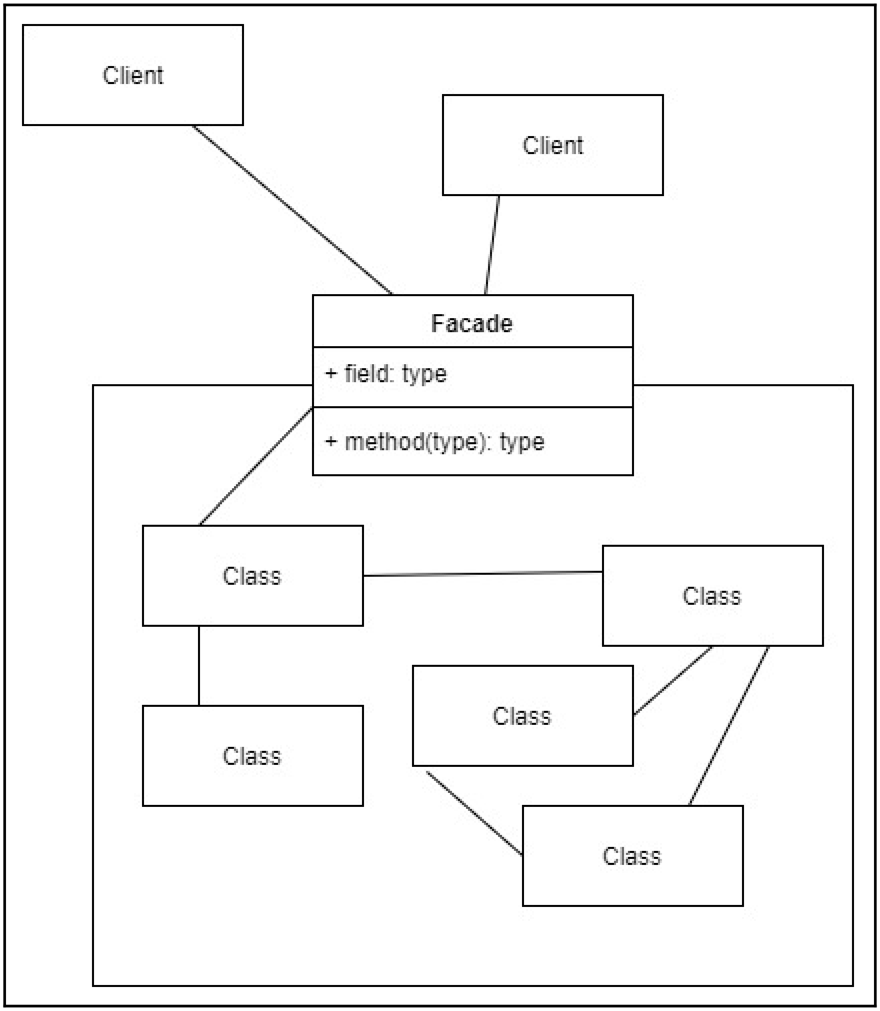

```
<terminated> Main (7) [Java Application] C:\Program Files\Java\jdk-9\bin\javaw.exe (Jul 24, 2017, 12:11:16 AM)
Sending message
From : abc@gmail.com
To : development_all@abc.com
Body : Test
From the windows machine
```

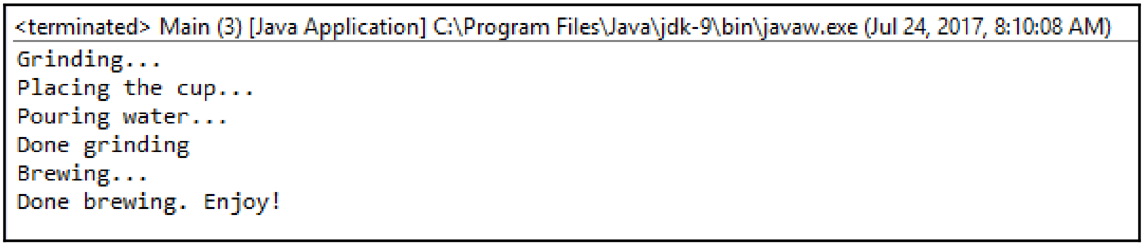




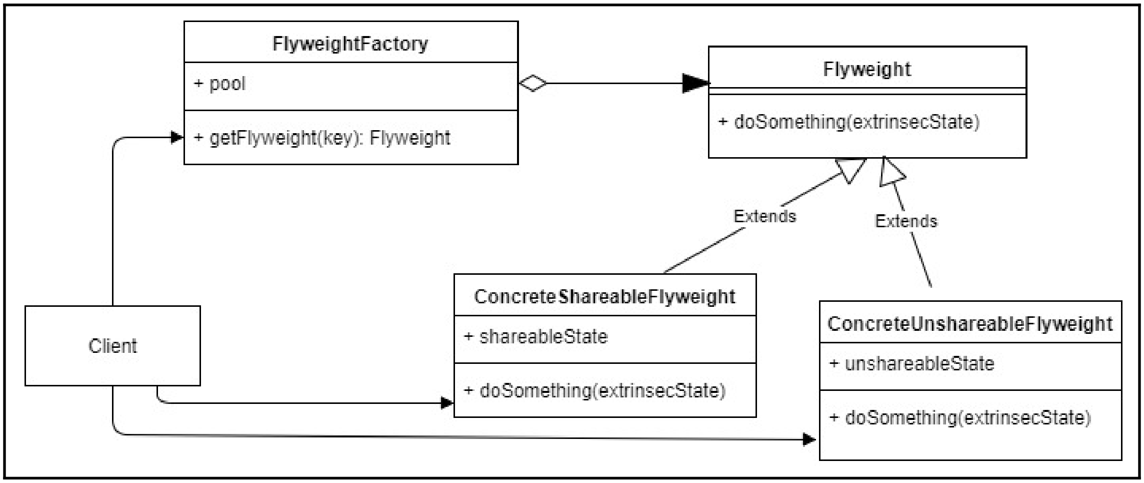


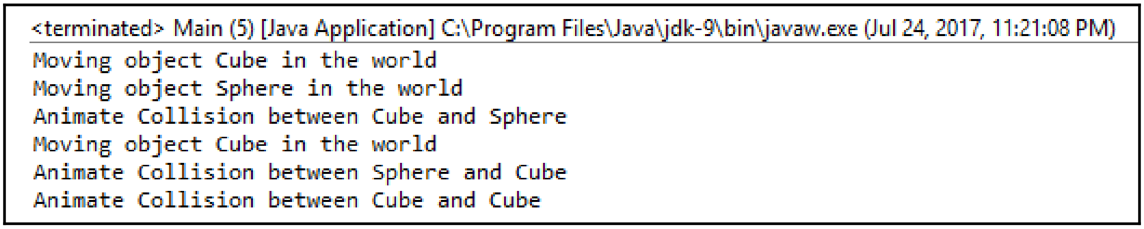
```
<terminated> Main (4) [Java Application] C:\Program Files\Java\jdk-9\bin\javaw.exe (Jul 24, 2017, 12:29:33 AM)
Value equals 3
```





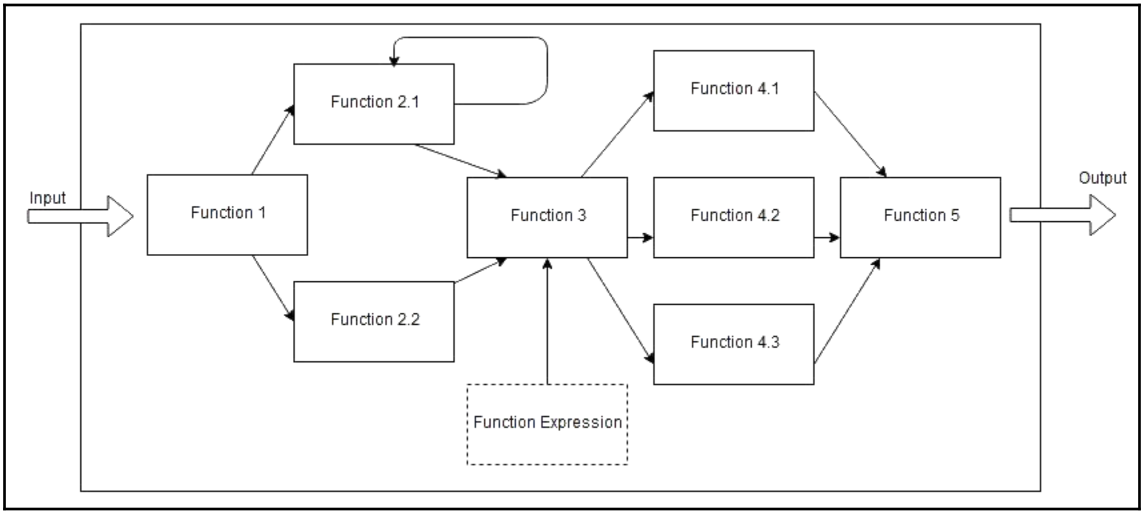
```
<terminated> Main (3) [Java Application] C:\Program Files\Java\jdk-9\bin\javaw.exe (Jul 24, 2017, 8:10:08 AM)
Grinding...
Placing the cup...
Pouring water...
Done grinding
Brewing...
Done brewing. Enjoy!
```


FlyweightFactory
+ pool
+ getFlyweight(key): Flyweight
Flyweight
+ doSomething(extrinsecState)
Extends
Extends
ConcreteShareableFlyweight
+ shareableState
+ doSomething(extrinsecState)
ConcreteUnshareableFlyweight
+ unshareableState
+ doSomething(extrinsecState)
Client


<terminated> Main (5) [Java Application] C:\Program Files\Java\jdk-9\bin\javaw.exe (Jul 24, 2017, 11:21:08 PM)

```
Moving object Cube in the world
Moving object Sphere in the world
Animate Collision between Cube and Sphere
Moving object Cube in the world
Animate Collision between Sphere and Cube
Animate Collision between Cube and Cube
```

Chapter 5: Functional Patterns



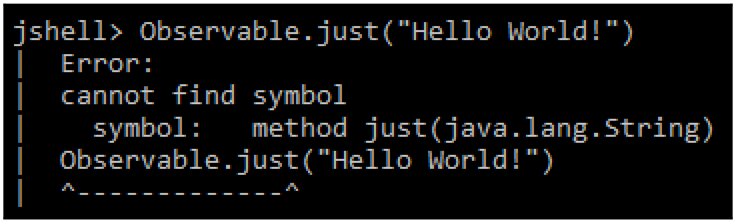
$$(x, y) \to x^2 + y^2$$

$$\lambda xy.x * x + y * y$$

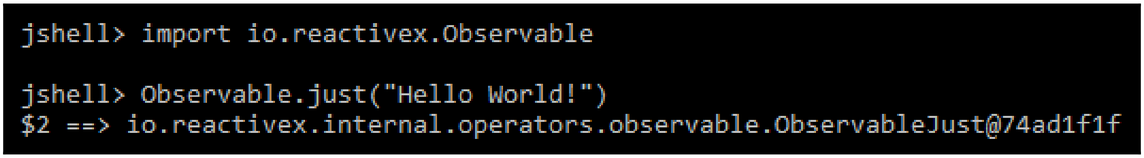
$$\left((x, y) \to x^2 + y^2\right)(1, 2) = 5$$

$$\left(\left((x, y) \to x^2 + y^2\right)(1)\right)(2) = 5$$

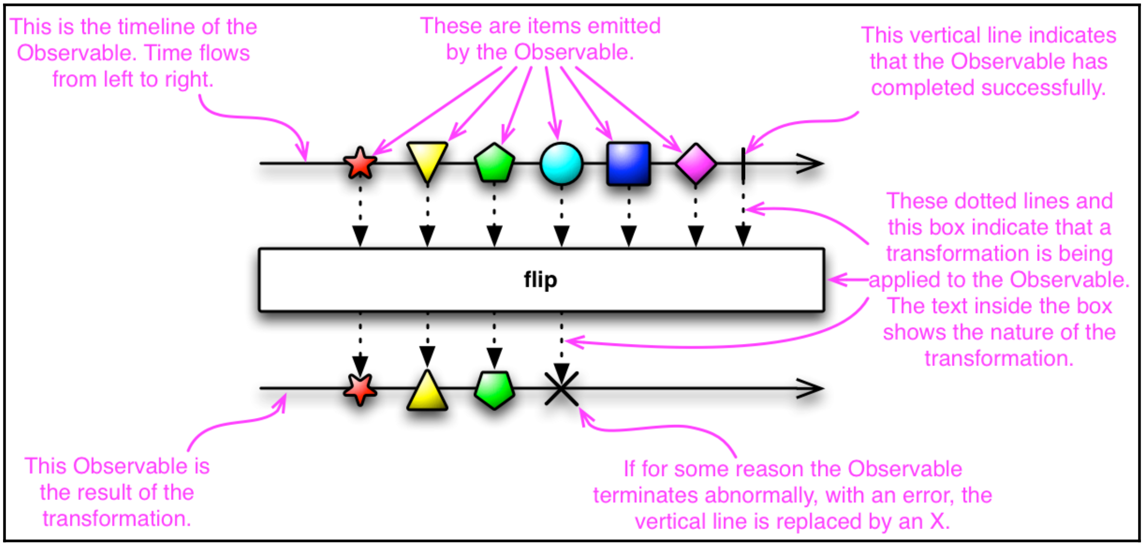
Chapter 6: Let's Get Reactive



```
jshell> Observable.just("Hello World!")
|   Error:
|   cannot find symbol
|     symbol:   method just(java.lang.String)
|   Observable.just("Hello World!")
|   ^-------------^
```



```
jshell> import io.reactivex.Observable

jshell> Observable.just("Hello World!")
$2 ==> io.reactivex.internal.operators.observable.ObservableJust@74ad1f1f
```



This is the timeline of the Observable. Time flows from left to right.

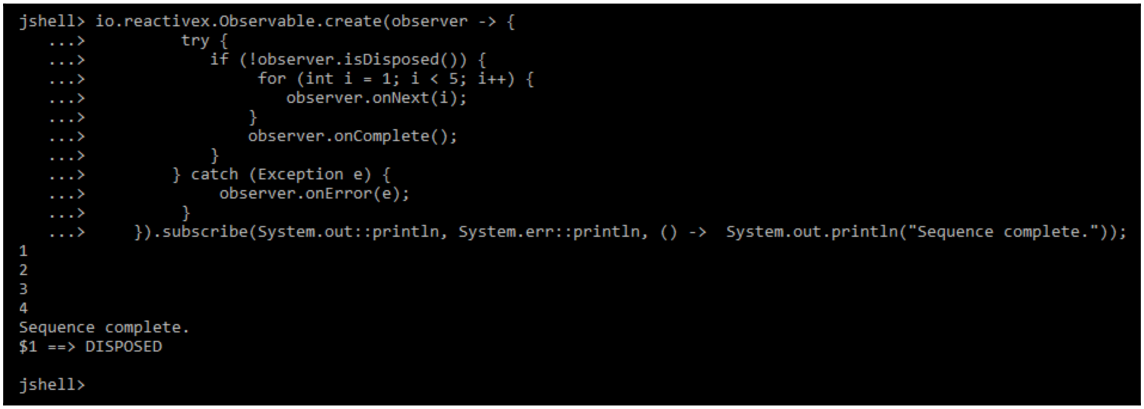
These are items emitted by the Observable.

This vertical line indicates that the Observable has completed successfully.

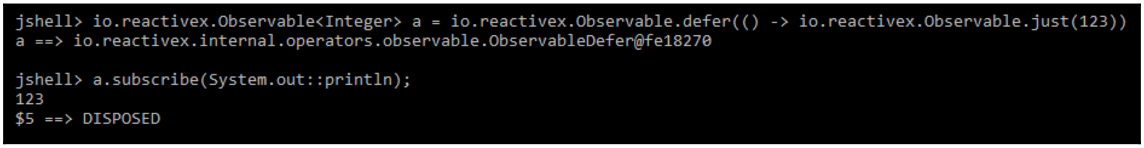
These dotted lines and this box indicate that a transformation is being applied to the Observable. The text inside the box shows the nature of the transformation.

This Observable is the result of the transformation.

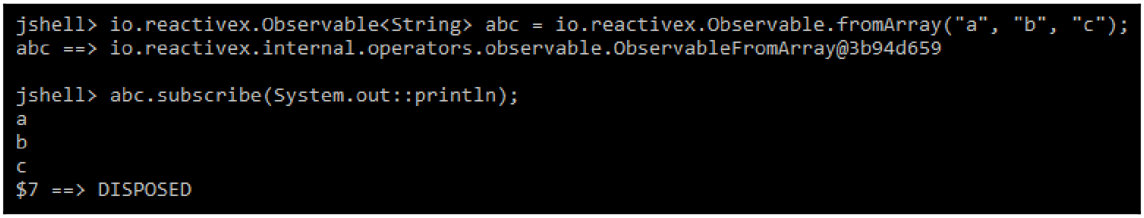
If for some reason the Observable terminates abnormally, with an error, the vertical line is replaced by an X.



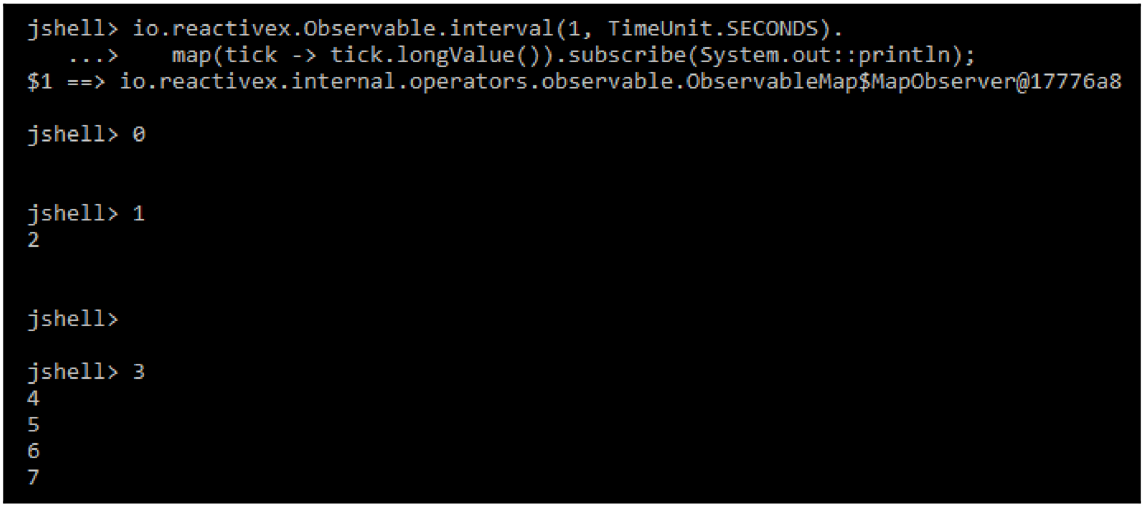
```
jshell> io.reactivex.Observable.create(observer -> {
   ...>          try {
   ...>             if (!observer.isDisposed()) {
   ...>                  for (int i = 1; i < 5; i++) {
   ...>                     observer.onNext(i);
   ...>                 }
   ...>                 observer.onComplete();
   ...>             }
   ...>         } catch (Exception e) {
   ...>              observer.onError(e);
   ...>          }
   ...>     }).subscribe(System.out::println, System.err::println, () ->  System.out.println("Sequence complete."));
1
2
3
4
Sequence complete.
$1 ==> DISPOSED

jshell>
```



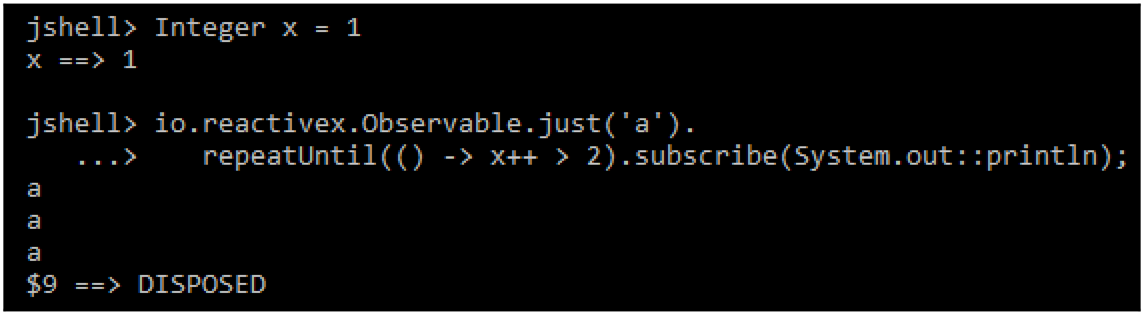
```
jshell> io.reactivex.Observable<Integer> a = io.reactivex.Observable.defer(() -> io.reactivex.Observable.just(123))
a ==> io.reactivex.internal.operators.observable.ObservableDefer@fe18270

jshell> a.subscribe(System.out::println);
123
$5 ==> DISPOSED
```



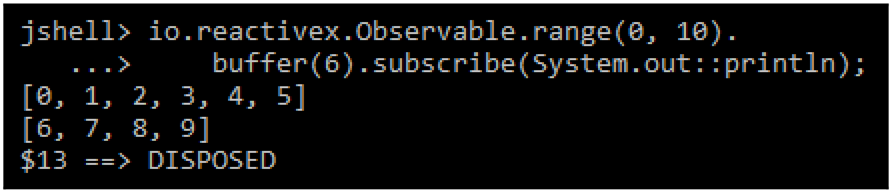
```
jshell> io.reactivex.Observable<String> abc = io.reactivex.Observable.fromArray("a", "b", "c");
abc ==> io.reactivex.internal.operators.observable.ObservableFromArray@3b94d659

jshell> abc.subscribe(System.out::println);
a
b
c
$7 ==> DISPOSED
```



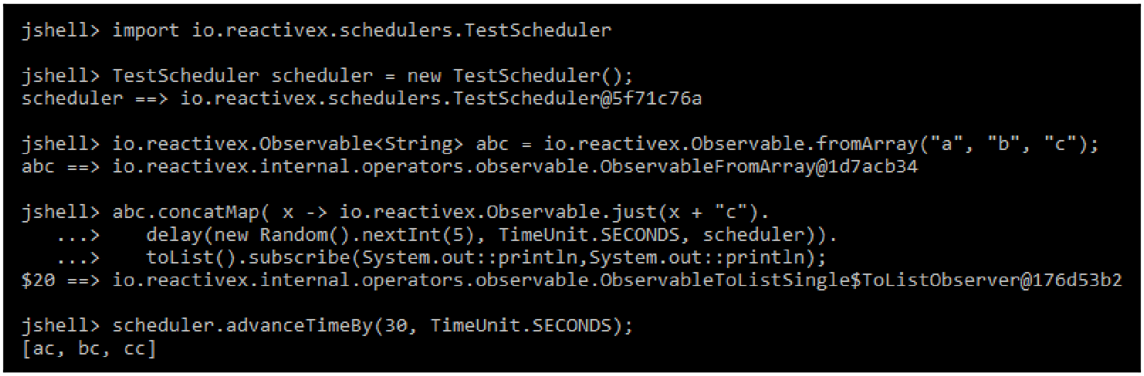
```
jshell> io.reactivex.Observable.interval(1, TimeUnit.SECONDS).
   ...>    map(tick -> tick.longValue()).subscribe(System.out::println);
$1 ==> io.reactivex.internal.operators.observable.ObservableMap$MapObserver@17776a8

jshell> 0


jshell> 1
2


jshell>

jshell> 3
4
5
6
7
```



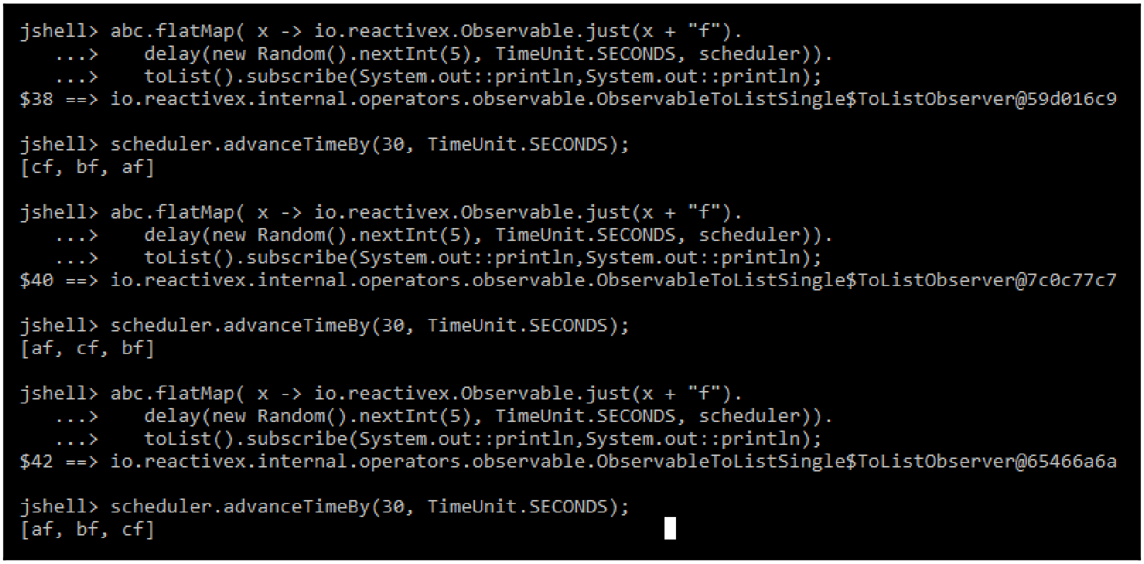
```
jshell> Integer x = 1
x ==> 1

jshell> io.reactivex.Observable.just('a').
   ...>    repeatUntil(() -> x++ > 2).subscribe(System.out::println);
a
a
a
$9 ==> DISPOSED
```



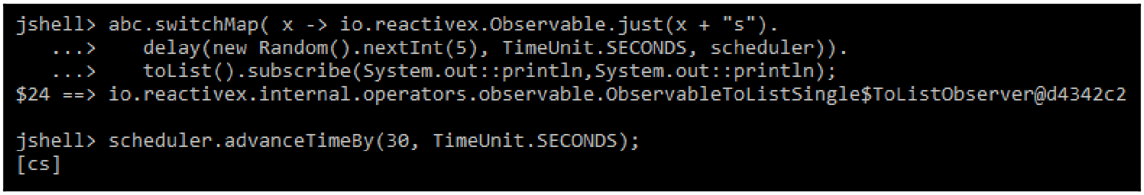
```
jshell> io.reactivex.Observable.range(0, 10).
   ...>     buffer(6).subscribe(System.out::println);
[0, 1, 2, 3, 4, 5]
[6, 7, 8, 9]
$13 ==> DISPOSED
```



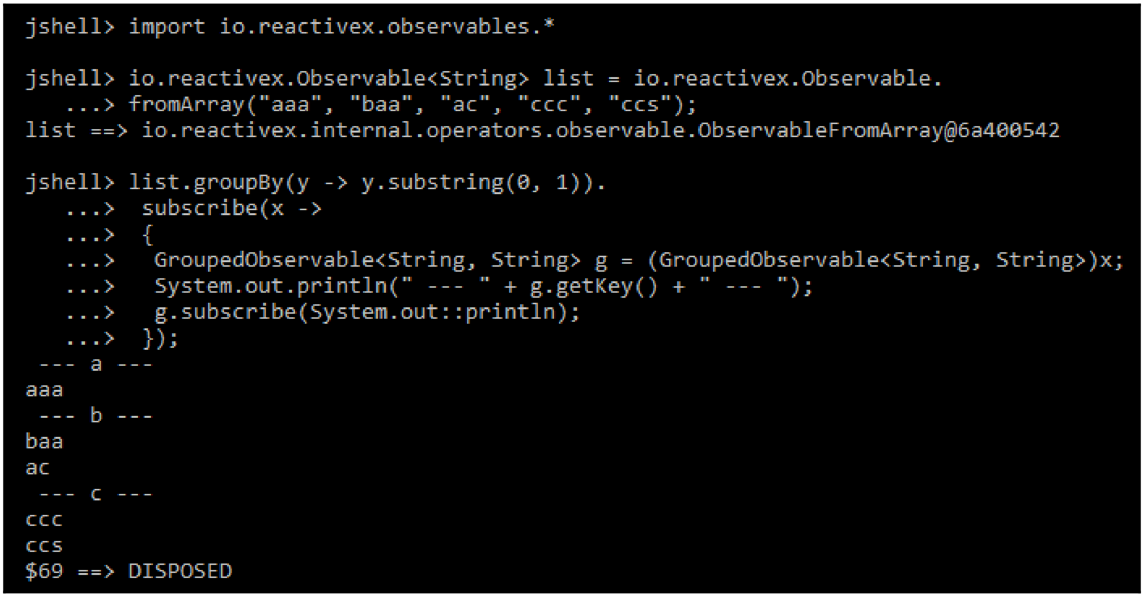
```
jshell> import io.reactivex.schedulers.TestScheduler

jshell> TestScheduler scheduler = new TestScheduler();
scheduler ==> io.reactivex.schedulers.TestScheduler@5f71c76a

jshell> io.reactivex.Observable<String> abc = io.reactivex.Observable.fromArray("a", "b", "c");
abc ==> io.reactivex.internal.operators.observable.ObservableFromArray@1d7acb34

jshell> abc.concatMap( x -> io.reactivex.Observable.just(x + "c").
   ...>    delay(new Random().nextInt(5), TimeUnit.SECONDS, scheduler)).
   ...>    toList().subscribe(System.out::println,System.out::println);
$20 ==> io.reactivex.internal.operators.observable.ObservableToListSingle$ToListObserver@176d53b2

jshell> scheduler.advanceTimeBy(30, TimeUnit.SECONDS);
[ac, bc, cc]
```



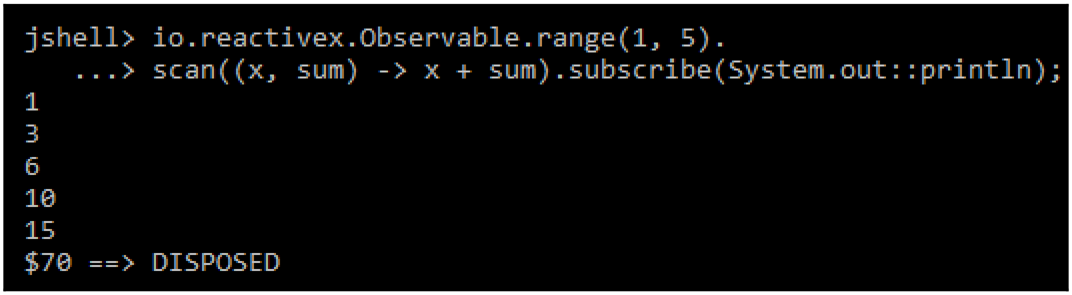
```
jshell> abc.flatMap( x -> io.reactivex.Observable.just(x + "f").
   ...>    delay(new Random().nextInt(5), TimeUnit.SECONDS, scheduler)).
   ...>    toList().subscribe(System.out::println,System.out::println);
$38 ==> io.reactivex.internal.operators.observable.ObservableToListSingle$ToListObserver@59d016c9

jshell> scheduler.advanceTimeBy(30, TimeUnit.SECONDS);
[cf, bf, af]

jshell> abc.flatMap( x -> io.reactivex.Observable.just(x + "f").
   ...>    delay(new Random().nextInt(5), TimeUnit.SECONDS, scheduler)).
   ...>    toList().subscribe(System.out::println,System.out::println);
$40 ==> io.reactivex.internal.operators.observable.ObservableToListSingle$ToListObserver@7c0c77c7

jshell> scheduler.advanceTimeBy(30, TimeUnit.SECONDS);
[af, cf, bf]

jshell> abc.flatMap( x -> io.reactivex.Observable.just(x + "f").
   ...>    delay(new Random().nextInt(5), TimeUnit.SECONDS, scheduler)).
   ...>    toList().subscribe(System.out::println,System.out::println);
$42 ==> io.reactivex.internal.operators.observable.ObservableToListSingle$ToListObserver@65466a6a

jshell> scheduler.advanceTimeBy(30, TimeUnit.SECONDS);
[af, bf, cf]
```



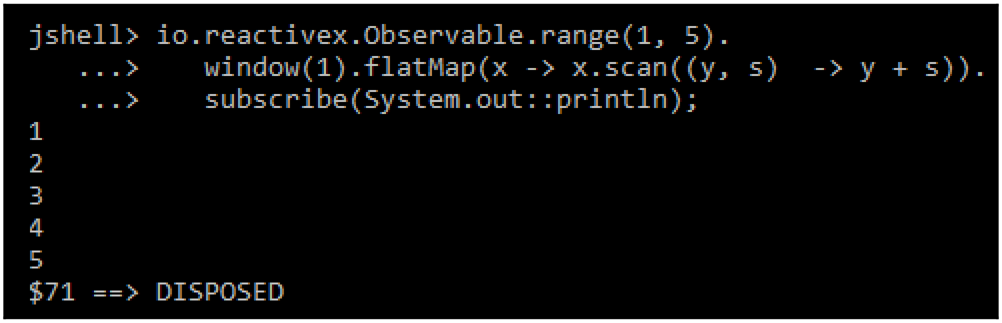
```
jshell> abc.switchMap( x -> io.reactivex.Observable.just(x + "s").
   ...>    delay(new Random().nextInt(5), TimeUnit.SECONDS, scheduler)).
   ...>    toList().subscribe(System.out::println,System.out::println);
$24 ==> io.reactivex.internal.operators.observable.ObservableToListSingle$ToListObserver@d4342c2

jshell> scheduler.advanceTimeBy(30, TimeUnit.SECONDS);
[cs]
```



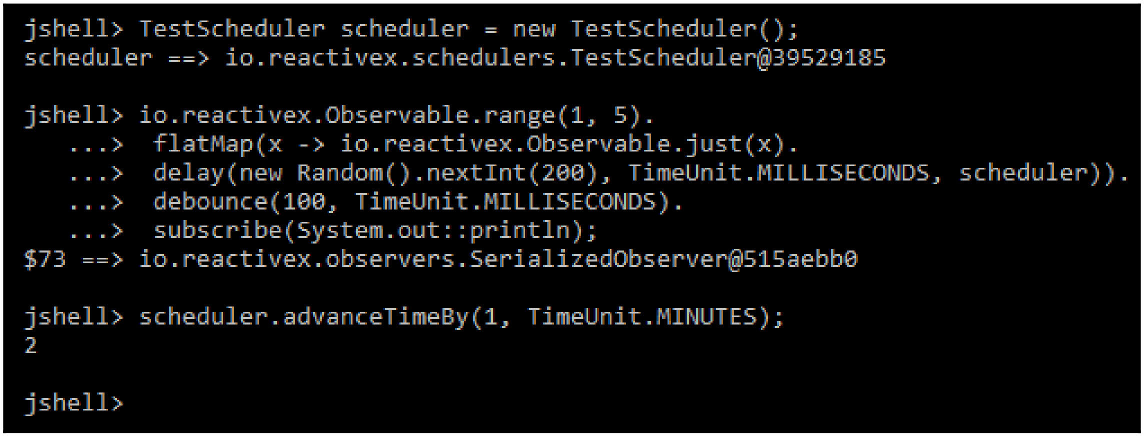
```
jshell> import io.reactivex.observables.*

jshell> io.reactivex.Observable<String> list = io.reactivex.Observable.
   ...> fromArray("aaa", "baa", "ac", "ccc", "ccs");
list ==> io.reactivex.internal.operators.observable.ObservableFromArray@6a400542

jshell> list.groupBy(y -> y.substring(0, 1)).
   ...>  subscribe(x ->
   ...>  {
   ...>   GroupedObservable<String, String> g = (GroupedObservable<String, String>)x;
   ...>   System.out.println(" --- " + g.getKey() + " --- ");
   ...>   g.subscribe(System.out::println);
   ...>  });
 --- a ---
aaa
 --- b ---
baa
ac
 --- c ---
ccc
ccs
$69 ==> DISPOSED
```



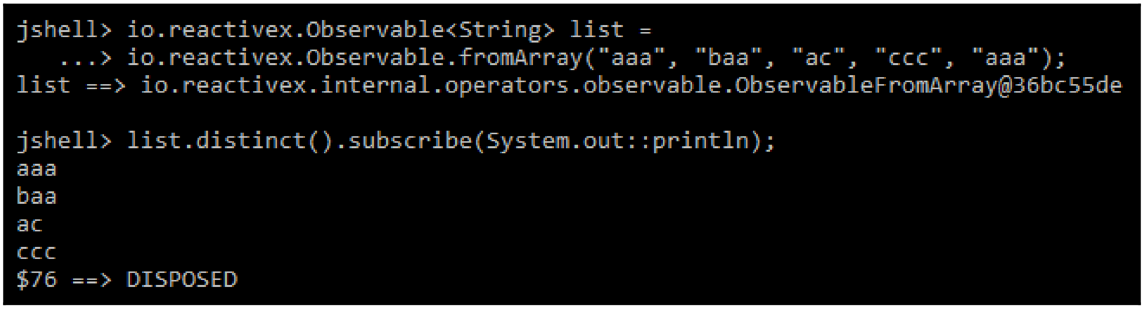
```
jshell> io.reactivex.Observable.range(1, 5).
   ...> scan((x, sum) -> x + sum).subscribe(System.out::println);
1
3
6
10
15
$70 ==> DISPOSED
```



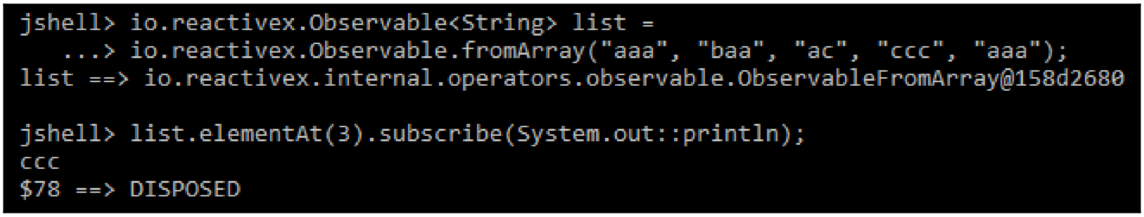
```
jshell> io.reactivex.Observable.range(1, 5).
   ...>    window(1).flatMap(x -> x.scan((y, s)  -> y + s)).
   ...>    subscribe(System.out::println);
1
2
3
4
5
$71 ==> DISPOSED
```



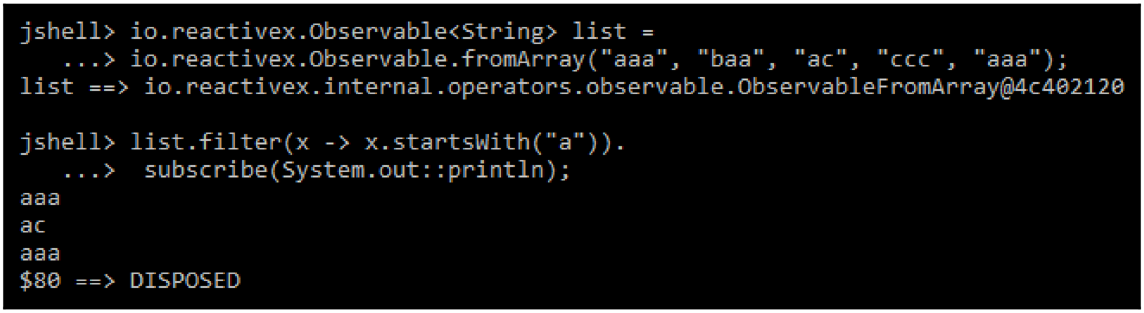
```
jshell> TestScheduler scheduler = new TestScheduler();
scheduler ==> io.reactivex.schedulers.TestScheduler@39529185

jshell> io.reactivex.Observable.range(1, 5).
   ...>  flatMap(x -> io.reactivex.Observable.just(x).
   ...>  delay(new Random().nextInt(200), TimeUnit.MILLISECONDS, scheduler)).
   ...>  debounce(100, TimeUnit.MILLISECONDS).
   ...>  subscribe(System.out::println);
$73 ==> io.reactivex.observers.SerializedObserver@515aebb0

jshell> scheduler.advanceTimeBy(1, TimeUnit.MINUTES);
2

jshell>
```



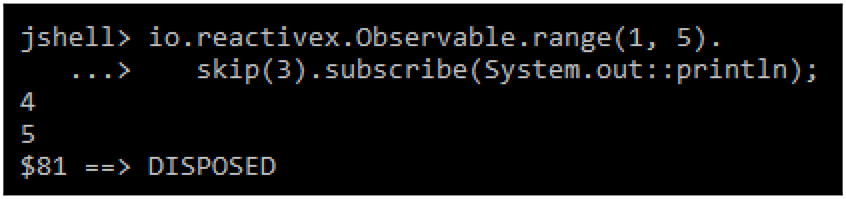
```
jshell> io.reactivex.Observable<String> list =
   ...> io.reactivex.Observable.fromArray("aaa", "baa", "ac", "ccc", "aaa");
list ==> io.reactivex.internal.operators.observable.ObservableFromArray@36bc55de

jshell> list.distinct().subscribe(System.out::println);
aaa
baa
ac
ccc
$76 ==> DISPOSED
```



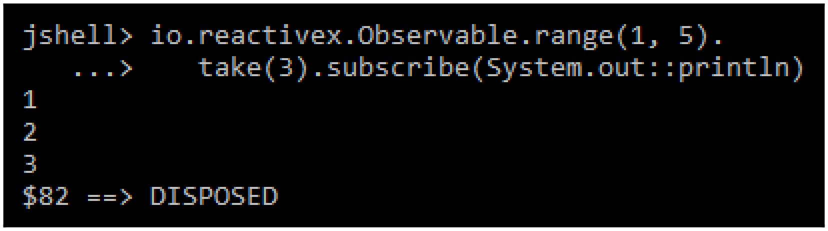
```
jshell> io.reactivex.Observable<String> list =
   ...> io.reactivex.Observable.fromArray("aaa", "baa", "ac", "ccc", "aaa");
list ==> io.reactivex.internal.operators.observable.ObservableFromArray@158d2680

jshell> list.elementAt(3).subscribe(System.out::println);
ccc
$78 ==> DISPOSED
```



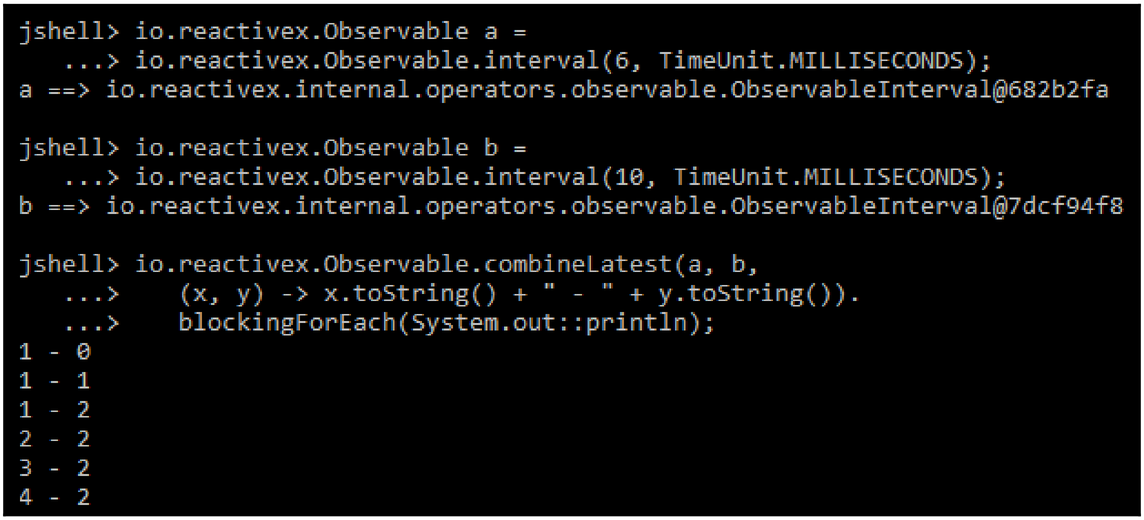
```
jshell> io.reactivex.Observable<String> list =
   ...> io.reactivex.Observable.fromArray("aaa", "baa", "ac", "ccc", "aaa");
list ==> io.reactivex.internal.operators.observable.ObservableFromArray@4c402120

jshell> list.filter(x -> x.startsWith("a")).
   ...>  subscribe(System.out::println);
aaa
ac
aaa
$80 ==> DISPOSED
```



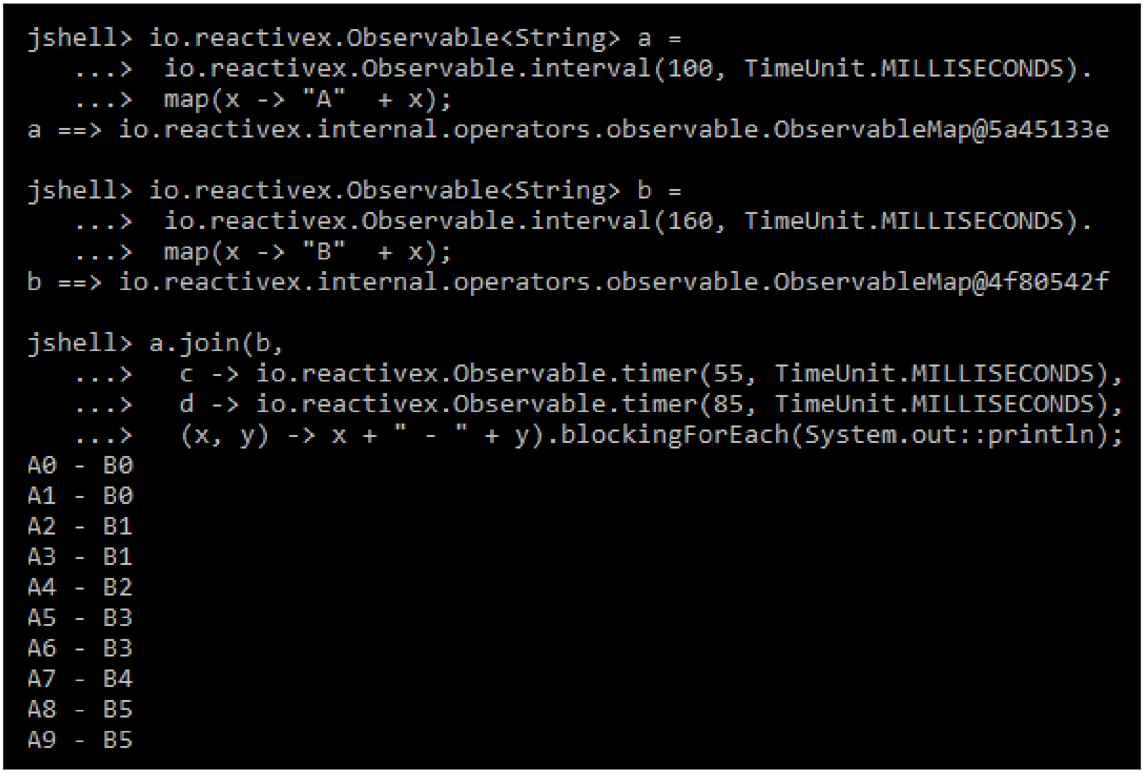
```
jshell> io.reactivex.Observable.range(1, 5).
   ...>     skip(3).subscribe(System.out::println);
4
5
$81 ==> DISPOSED
```



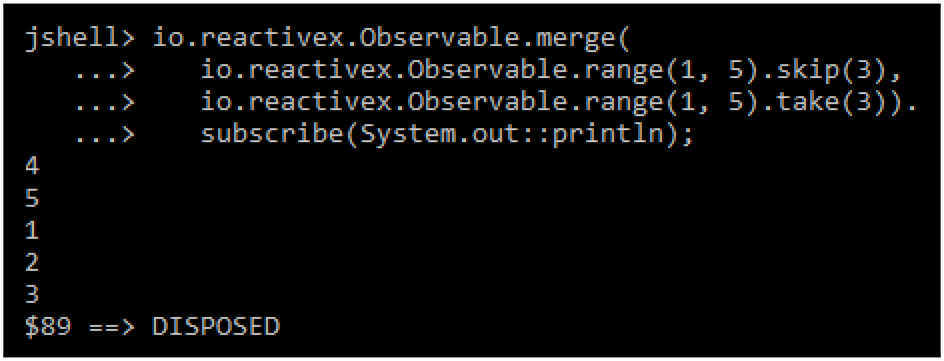
```
jshell> io.reactivex.Observable.range(1, 5).
   ...>     take(3).subscribe(System.out::println)
1
2
3
$82 ==> DISPOSED
```



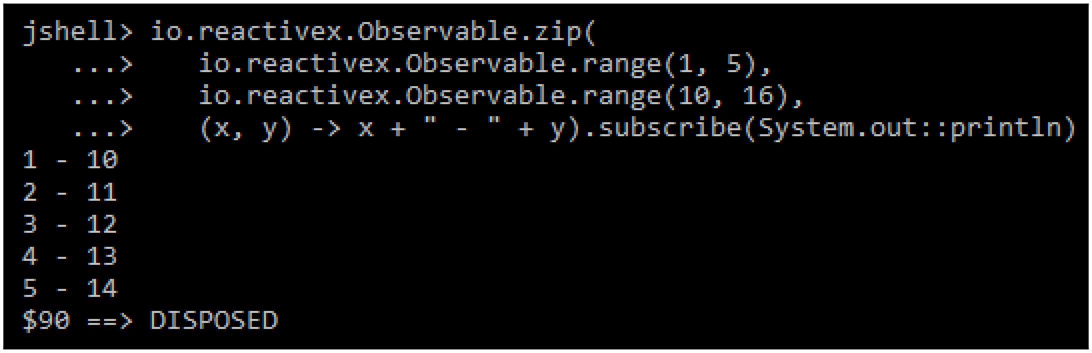
```
jshell> io.reactivex.Observable a =
   ...> io.reactivex.Observable.interval(6, TimeUnit.MILLISECONDS);
a ==> io.reactivex.internal.operators.observable.ObservableInterval@682b2fa

jshell> io.reactivex.Observable b =
   ...> io.reactivex.Observable.interval(10, TimeUnit.MILLISECONDS);
b ==> io.reactivex.internal.operators.observable.ObservableInterval@7dcf94f8

jshell> io.reactivex.Observable.combineLatest(a, b,
   ...>     (x, y) -> x.toString() + " - " + y.toString()).
   ...>     blockingForEach(System.out::println);
1 - 0
1 - 1
1 - 2
2 - 2
3 - 2
4 - 2
```



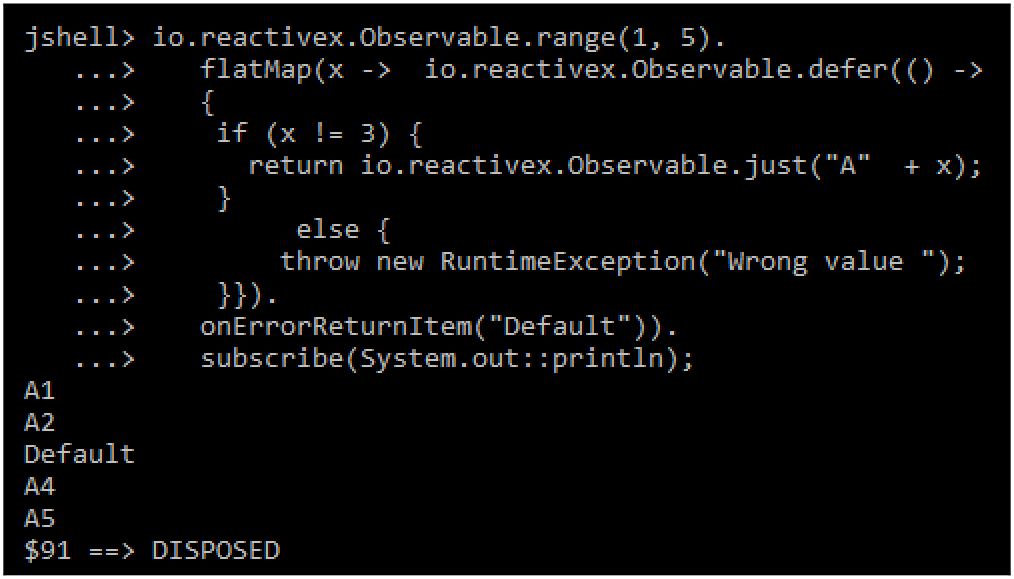
```
jshell> io.reactivex.Observable<String> a =
   ...>  io.reactivex.Observable.interval(100, TimeUnit.MILLISECONDS).
   ...>  map(x -> "A"  + x);
a ==> io.reactivex.internal.operators.observable.ObservableMap@5a45133e

jshell> io.reactivex.Observable<String> b =
   ...>  io.reactivex.Observable.interval(160, TimeUnit.MILLISECONDS).
   ...>  map(x -> "B"  + x);
b ==> io.reactivex.internal.operators.observable.ObservableMap@4f80542f

jshell> a.join(b,
   ...>   c -> io.reactivex.Observable.timer(55, TimeUnit.MILLISECONDS),
   ...>   d -> io.reactivex.Observable.timer(85, TimeUnit.MILLISECONDS),
   ...>   (x, y) -> x + " - " + y).blockingForEach(System.out::println);
A0 - B0
A1 - B0
A2 - B1
A3 - B1
A4 - B2
A5 - B3
A6 - B3
A7 - B4
A8 - B5
A9 - B5
```



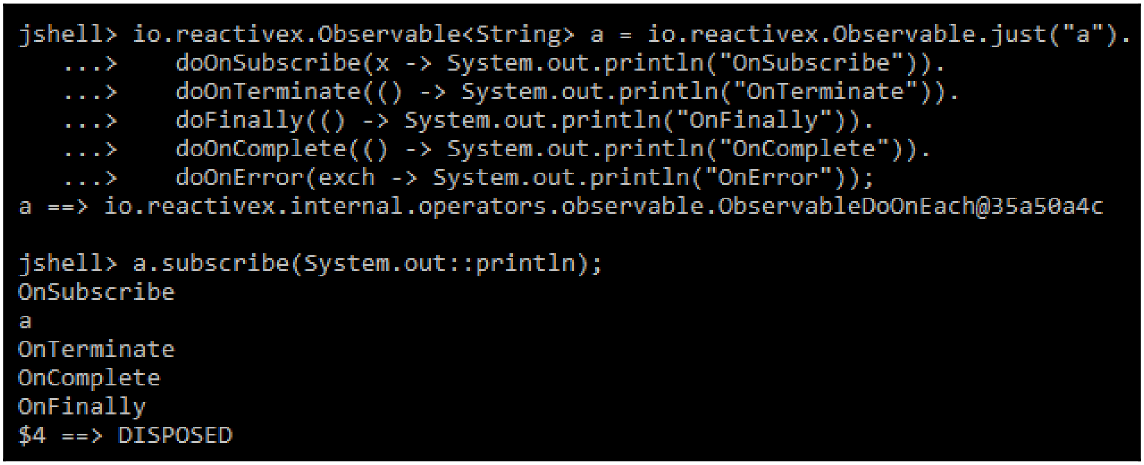
```
jshell> io.reactivex.Observable.merge(
   ...>    io.reactivex.Observable.range(1, 5).skip(3),
   ...>    io.reactivex.Observable.range(1, 5).take(3)).
   ...>    subscribe(System.out::println);
4
5
1
2
3
$89 ==> DISPOSED
```



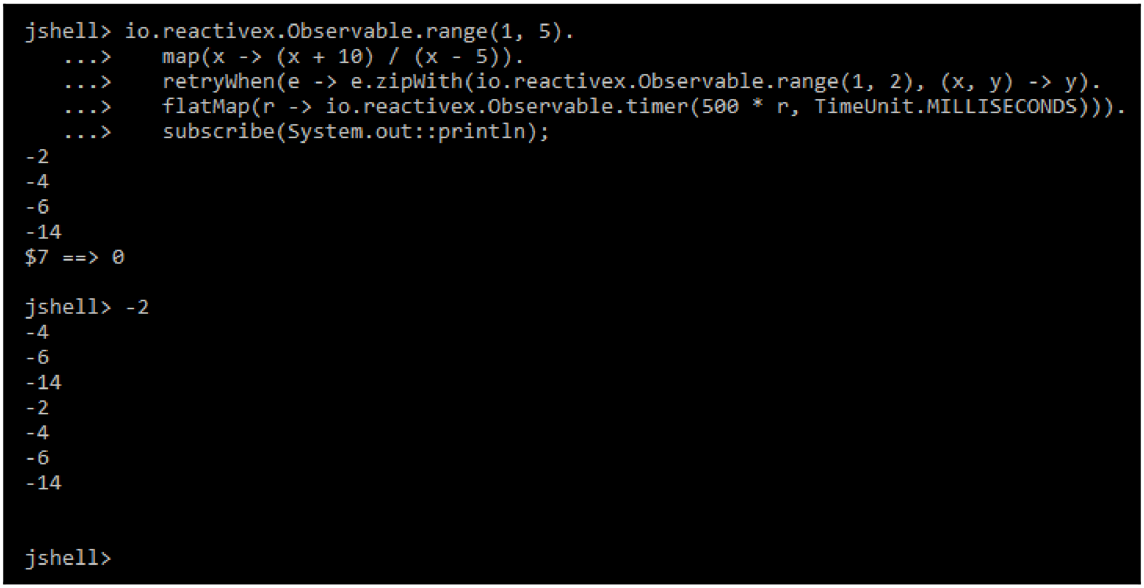
```
jshell> io.reactivex.Observable.zip(
   ...>    io.reactivex.Observable.range(1, 5),
   ...>    io.reactivex.Observable.range(10, 16),
   ...>    (x, y) -> x + " - " + y).subscribe(System.out::println)
1 - 10
2 - 11
3 - 12
4 - 13
5 - 14
$90 ==> DISPOSED
```



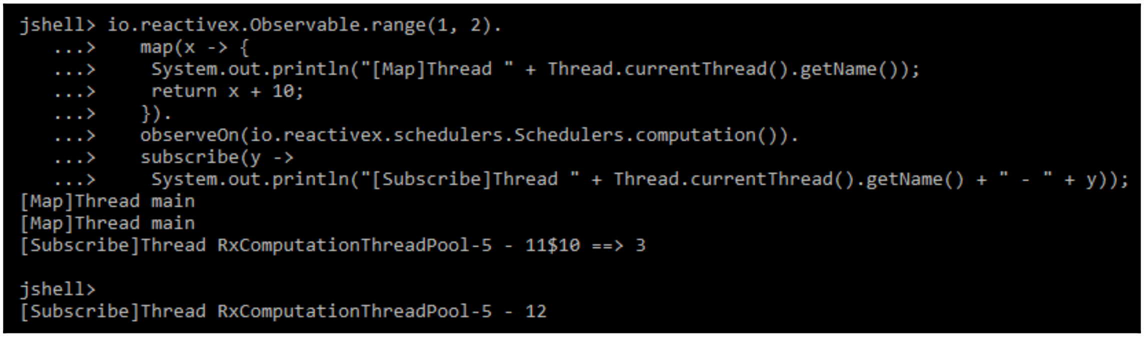
```
jshell> io.reactivex.Observable.range(1, 5).
   ...>    flatMap(x ->  io.reactivex.Observable.defer(() ->
   ...>    {
   ...>     if (x != 3) {
   ...>       return io.reactivex.Observable.just("A"  + x);
   ...>     }
   ...>          else {
   ...>         throw new RuntimeException("Wrong value ");
   ...>     }}).
   ...>    onErrorReturnItem("Default")).
   ...>    subscribe(System.out::println);
A1
A2
Default
A4
A5
$91 ==> DISPOSED
```



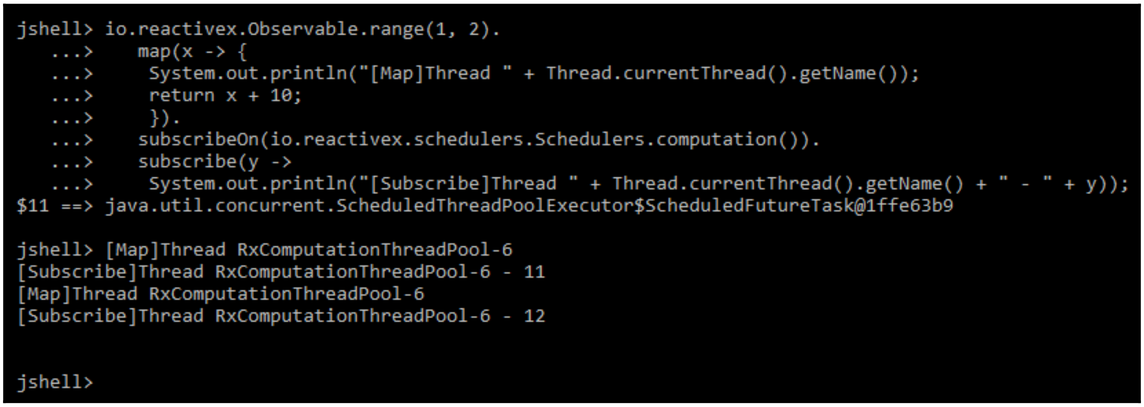
```
jshell> io.reactivex.Observable<String> a = io.reactivex.Observable.just("a").
   ...>    doOnSubscribe(x -> System.out.println("OnSubscribe")).
   ...>    doOnTerminate(() -> System.out.println("OnTerminate")).
   ...>    doFinally(() -> System.out.println("OnFinally")).
   ...>    doOnComplete(() -> System.out.println("OnComplete")).
   ...>    doOnError(exch -> System.out.println("OnError"));
a ==> io.reactivex.internal.operators.observable.ObservableDoOnEach@35a50a4c

jshell> a.subscribe(System.out::println);
OnSubscribe
a
OnTerminate
OnComplete
OnFinally
$4 ==> DISPOSED
```



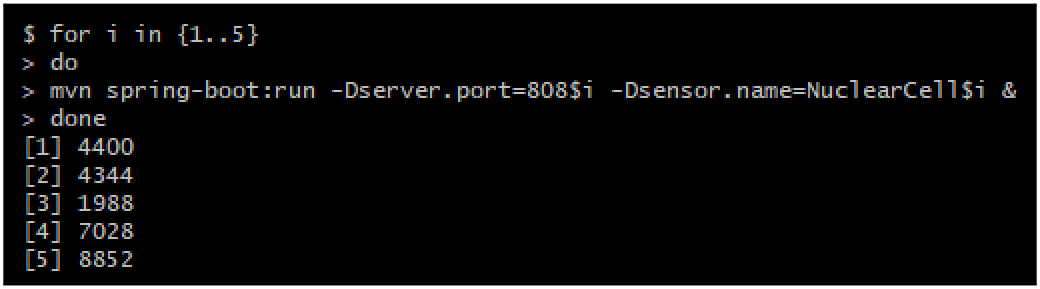
```
jshell> io.reactivex.Observable.range(1, 5).
   ...>    map(x -> (x + 10) / (x - 5)).
   ...>    retryWhen(e -> e.zipWith(io.reactivex.Observable.range(1, 2), (x, y) -> y).
   ...>    flatMap(r -> io.reactivex.Observable.timer(500 * r, TimeUnit.MILLISECONDS))).
   ...>    subscribe(System.out::println);
-2
-4
-6
-14
$7 ==> 0

jshell> -2
-4
-6
-14
-2
-4
-6
-14


jshell>
```



```
jshell> io.reactivex.Observable.range(1, 2).
   ...>    map(x -> {
   ...>     System.out.println("[Map]Thread " + Thread.currentThread().getName());
   ...>     return x + 10;
   ...>    }).
   ...>    observeOn(io.reactivex.schedulers.Schedulers.computation()).
   ...>    subscribe(y ->
   ...>     System.out.println("[Subscribe]Thread " + Thread.currentThread().getName() + " - " + y));
[Map]Thread main
[Map]Thread main
[Subscribe]Thread RxComputationThreadPool-5 - 11$10 ==> 3

jshell>
[Subscribe]Thread RxComputationThreadPool-5 - 12
```

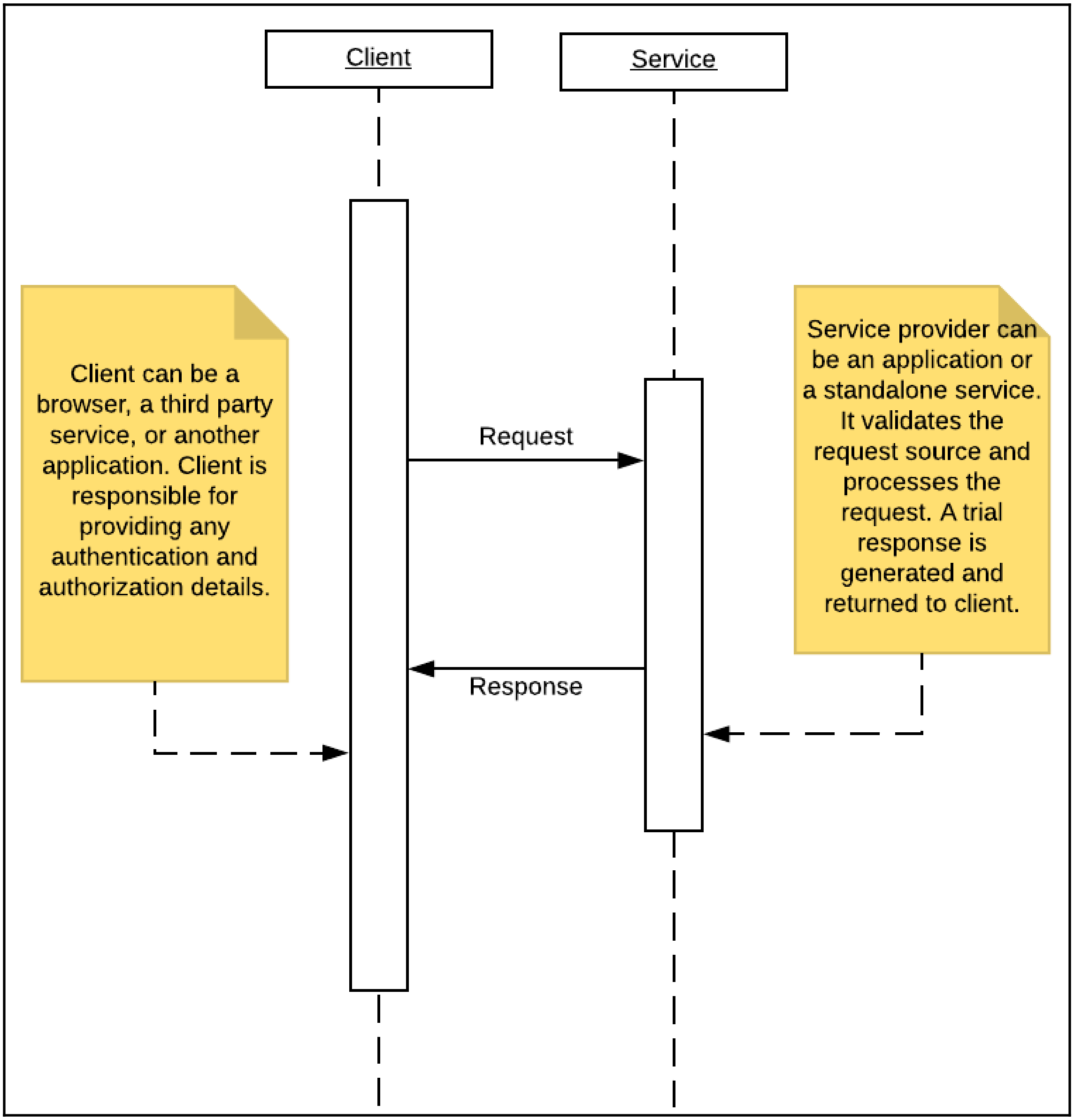


```
jshell> io.reactivex.Observable.range(1, 2).
   ...>    map(x -> {
   ...>     System.out.println("[Map]Thread " + Thread.currentThread().getName());
   ...>     return x + 10;
   ...>     }).
   ...>    subscribeOn(io.reactivex.schedulers.Schedulers.computation()).
   ...>    subscribe(y ->
   ...>     System.out.println("[Subscribe]Thread " + Thread.currentThread().getName() + " - " + y));
$11 ==> java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask@1ffe63b9

jshell> [Map]Thread RxComputationThreadPool-6
[Subscribe]Thread RxComputationThreadPool-6 - 11
[Map]Thread RxComputationThreadPool-6
[Subscribe]Thread RxComputationThreadPool-6 - 12


jshell>
```

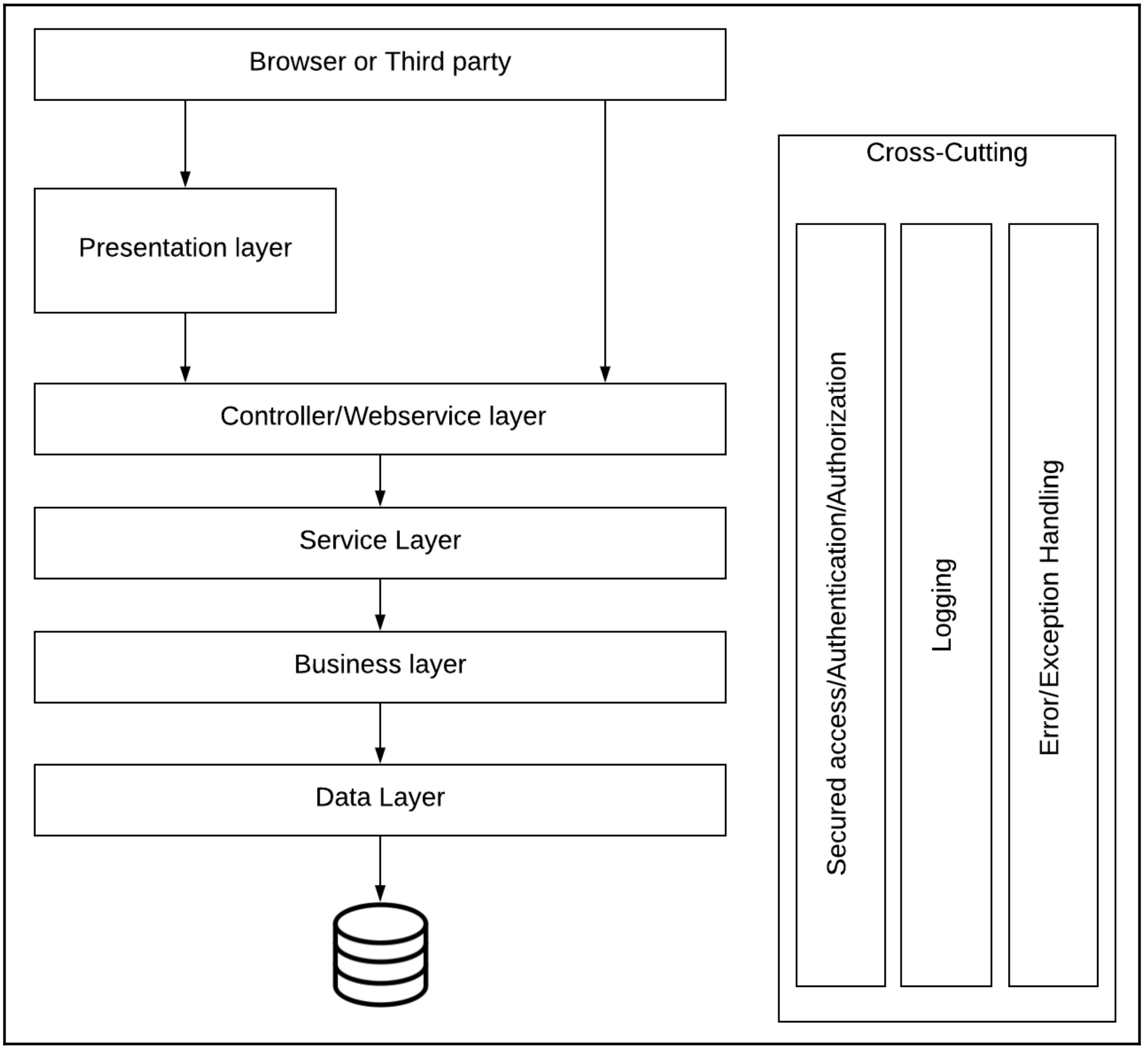


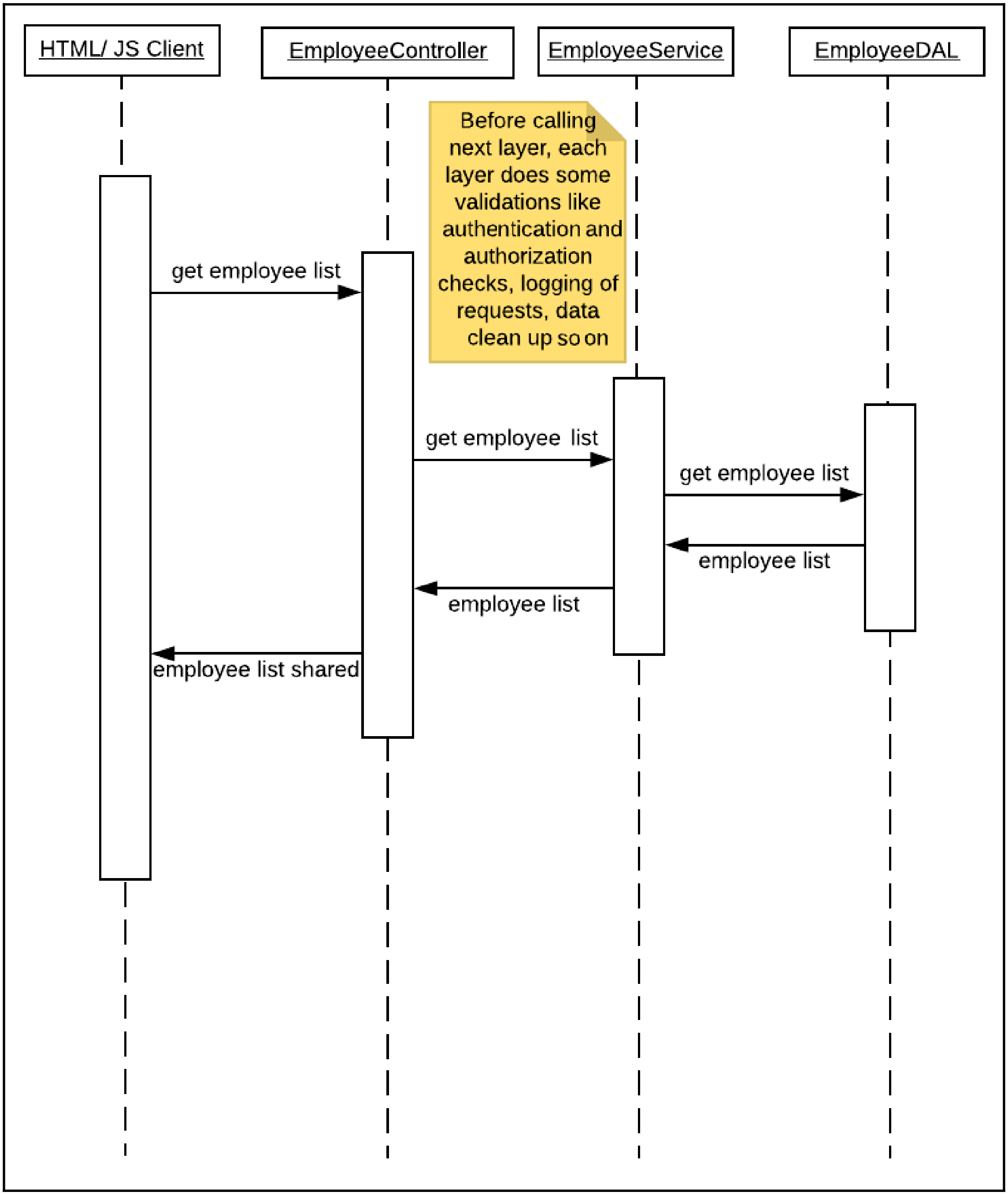
```
$ for i in {1..5}
> do
> mvn spring-boot:run -Dserver.port=808$i -Dsensor.name=NuclearCell$i &
> done
[1] 4400
[2] 4344
[3] 1988
[4] 7028
[5] 8852
```

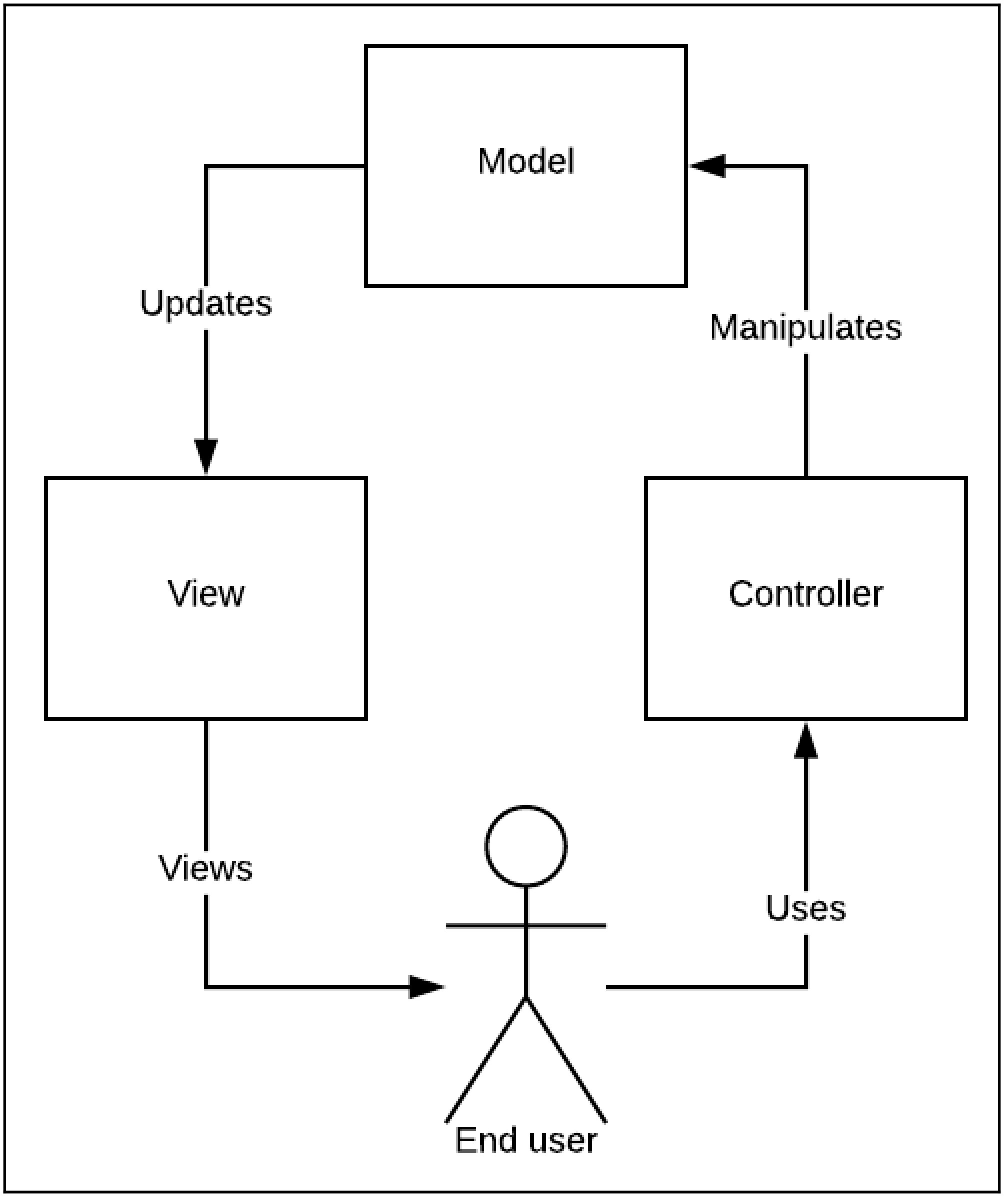
Chapter 7: Reactive Design Patterns

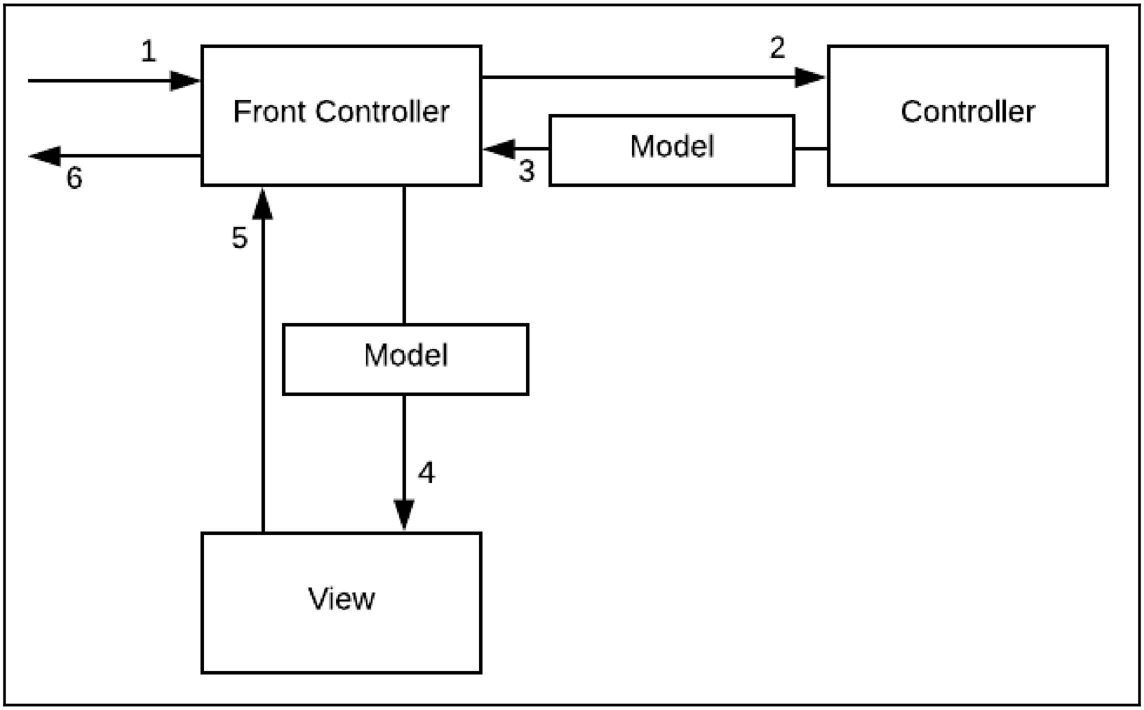

Client
Service
Client can be a browser, a third party service, or another application. Client is responsible for providing any authentication and authorization details.
Request
Service provider can be an application or a standalone service. It validates the request source and processes the request. A trial response is generated and returned to client.
Response

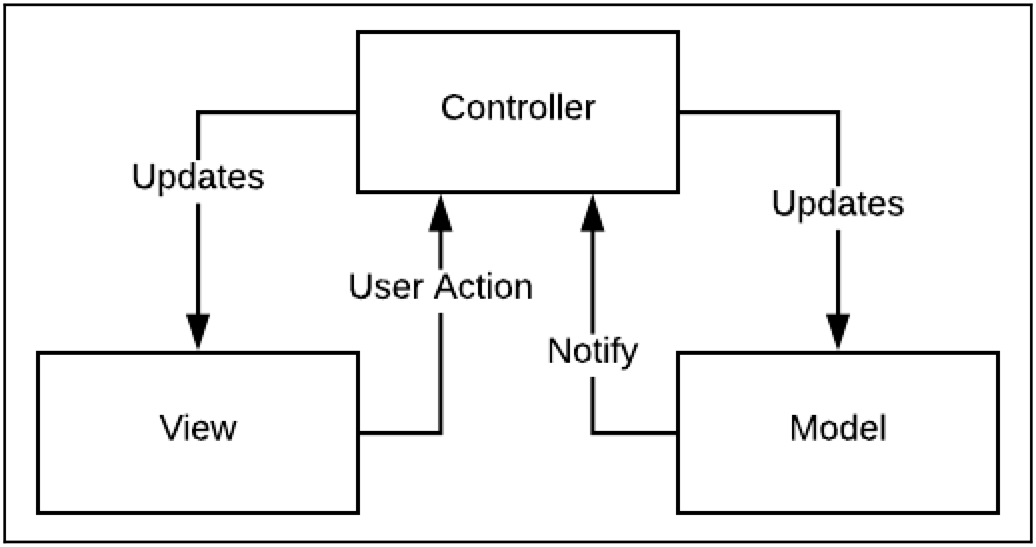
Chapter 8: Trends in Application Architecture

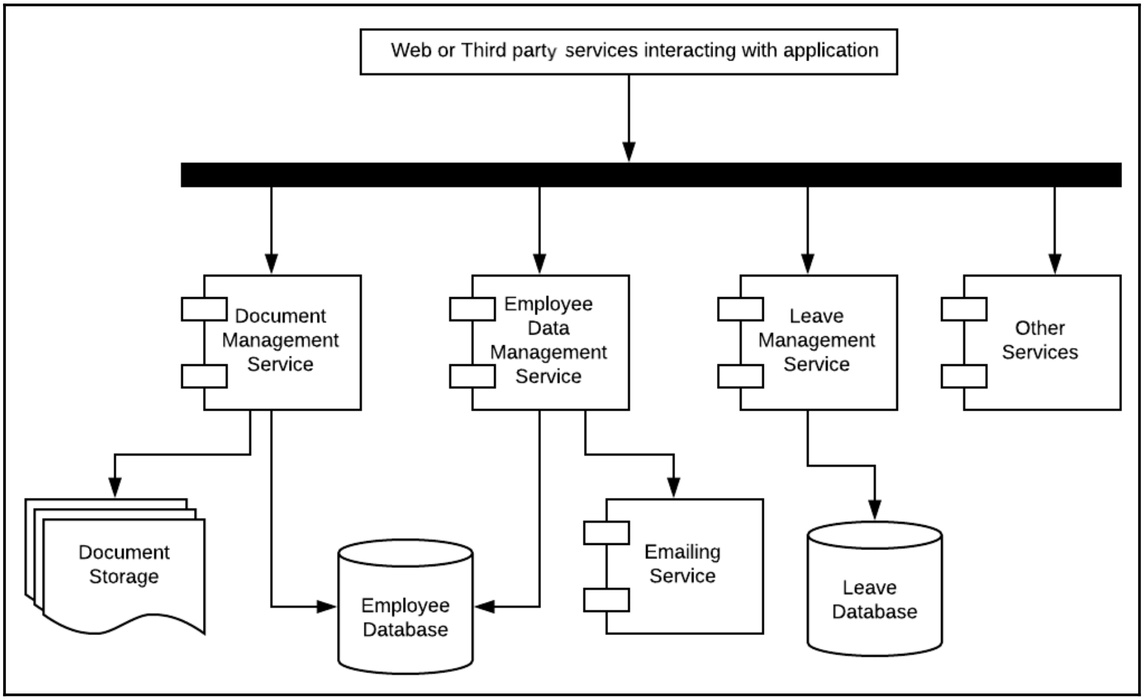


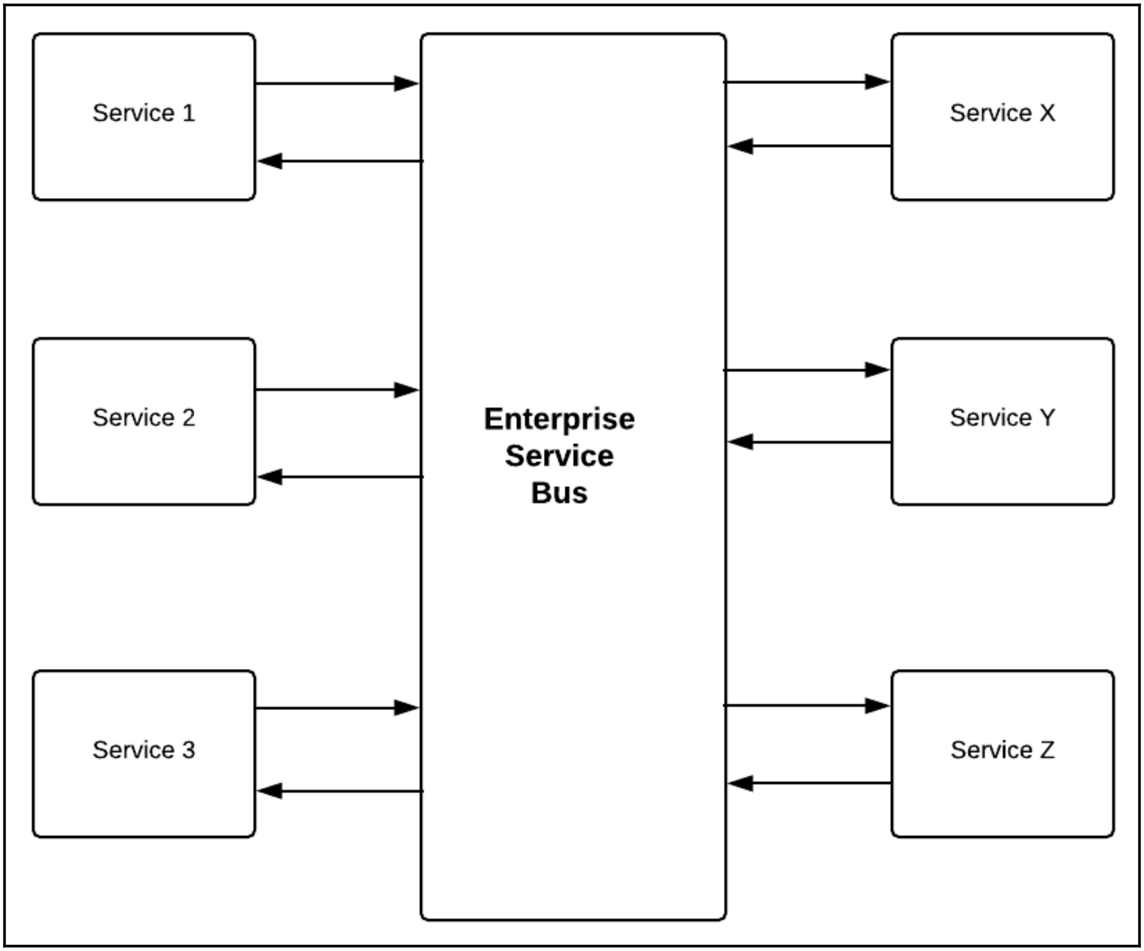

HTML/ JS Client
EmployeeController
EmployeeService
EmployeeDAL
Before calling next layer, each layer does some validations like authentication and authorization checks, logging of requests, data clean up so on
get employee list
get employee list
get employee list
employee list
employee list
employee list shared

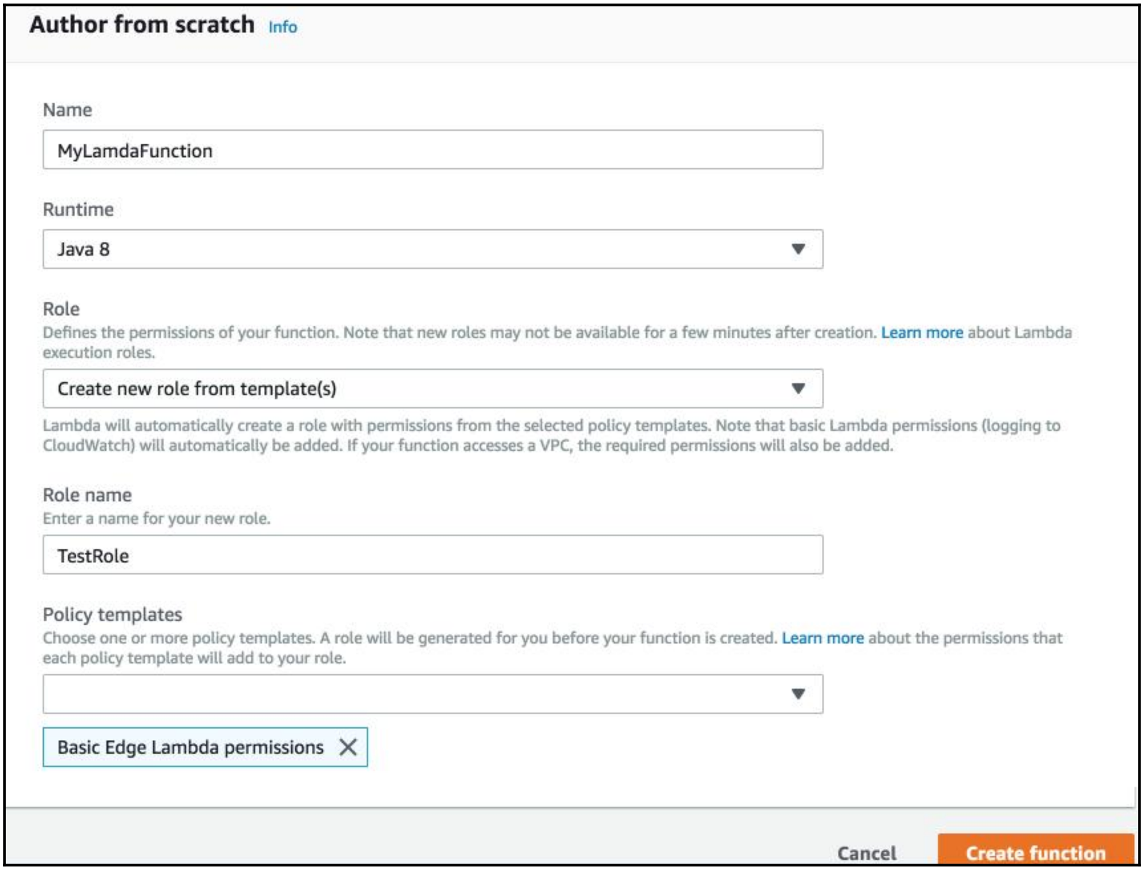

Model
Updates
Manipulates
View
Controller
Views
Uses
End user


1
2
Front Controller
Controller
Model
6
3
5
Model
4
View


Controller
Updates
Updates
User Action
Notify
View
Model


Web or Third party services interacting with application
Document Management Service
Employee Data Management Service
Leave Management Service
Other Services
Document Storage
Employee Database
Emailing Service
Leave Database


Service 1
Service 2
Service 3
Enterprise Service Bus
Service X
Service Y
Service Z



Author from scratch Info

Name

MyLamdaFunction

Runtime

Java 8

Role

Defines the permissions of your function. Note that new roles may not be available for a few minutes after creation. Learn more about Lambda execution roles.

Create new role from template(s)

Lambda will automatically create a role with permissions from the selected policy templates. Note that basic Lambda permissions (logging to CloudWatch) will automatically be added. If your function accesses a VPC, the required permissions will also be added.

Role name

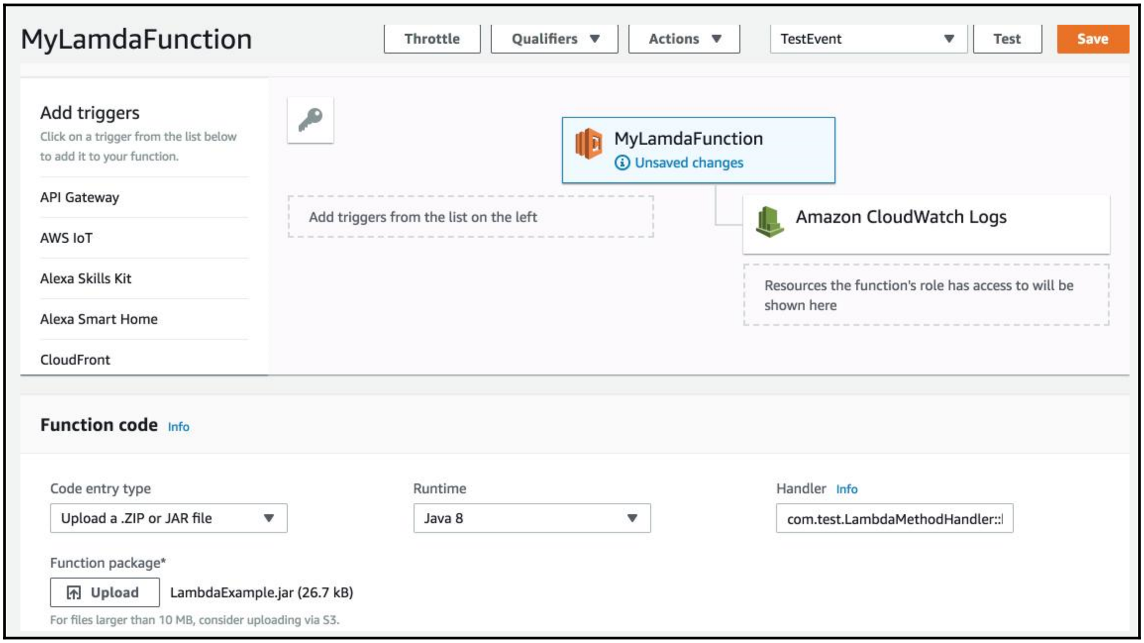
Enter a name for your new role.

TestRole

Policy templates

Choose one or more policy templates. A role will be generated for you before your function is created. Learn more about the permissions that each policy template will add to your role.

Basic Edge Lambda permissions ✕



MyLamdaFunction

Add triggers

Click on a trigger from the list below to add it to your function.

MyLamdaFunction

ⓘ Unsaved changes

Add triggers from the list on the left

Amazon CloudWatch Logs

Resources the function's role has access to will be shown here

Function code

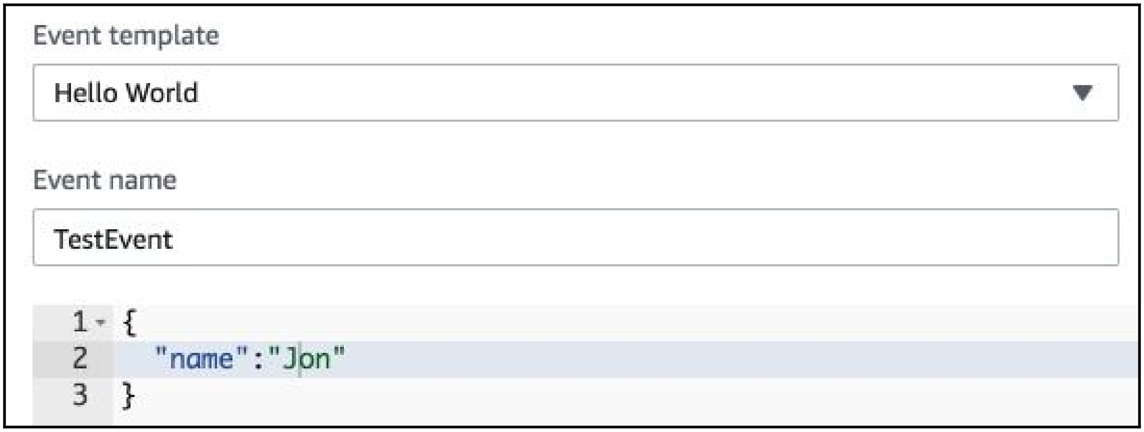
Code entry type: Upload a .ZIP or JAR file

Runtime: Java 8

Handler: com.test.LambdaMethodHandler::l

Function package*: LambdaExample.jar (26.7 kB)

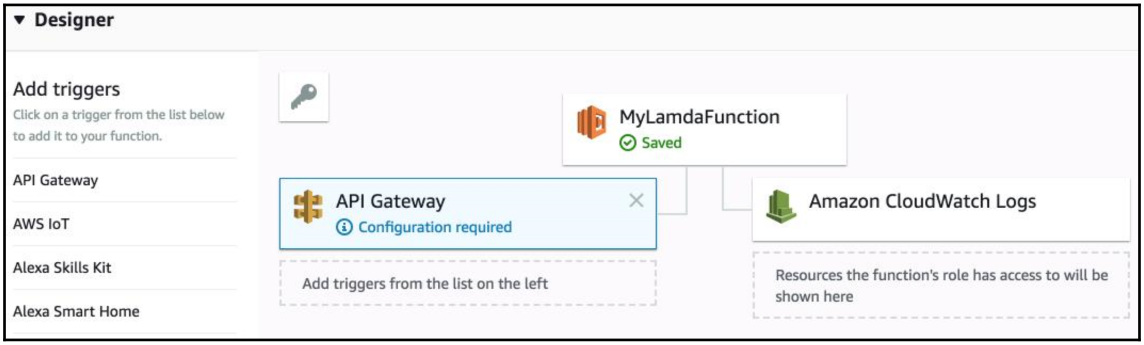
For files larger than 10 MB, consider uploading via S3.



Event template: Hello World

Event name: TestEvent

```
{
  "name":"Jon"
}
```



▼ Designer

Add triggers

Click on a trigger from the list below to add it to your function.

API Gateway

AWS IoT

Alexa Skills Kit

Alexa Smart Home

MyLamdaFunction

Saved

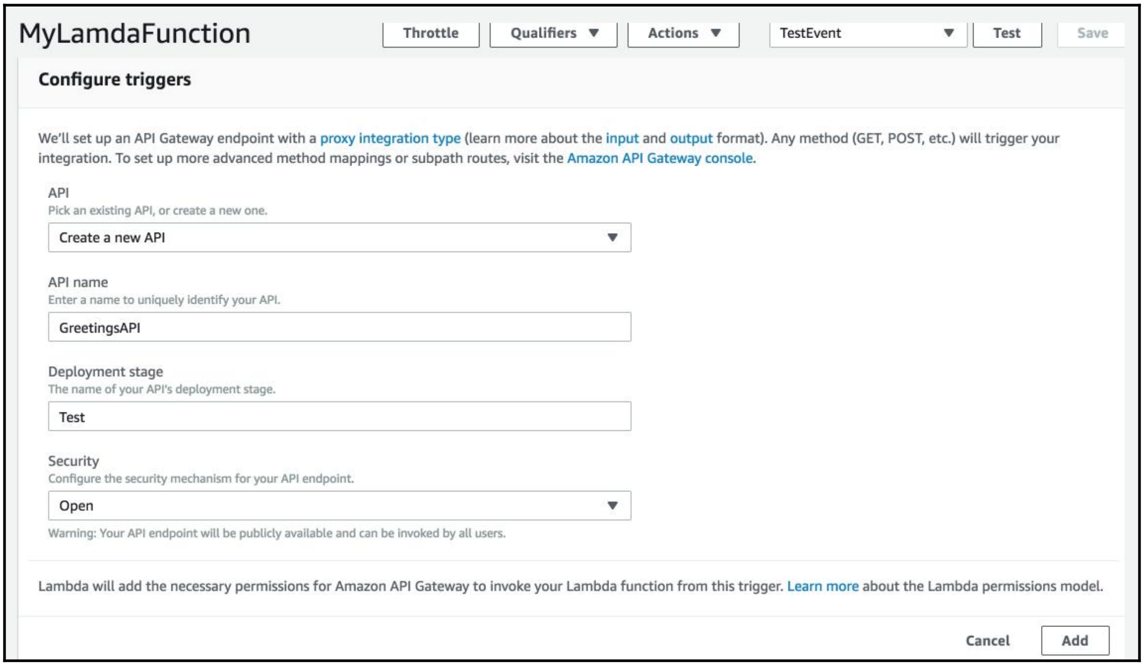
API Gateway

Configuration required

Amazon CloudWatch Logs

Add triggers from the list on the left

Resources the function's role has access to will be shown here



MyLamdaFunction

Configure triggers

We'll set up an API Gateway endpoint with a proxy integration type (learn more about the input and output format). Any method (GET, POST, etc.) will trigger your integration. To set up more advanced method mappings or subpath routes, visit the Amazon API Gateway console.

API

Pick an existing API, or create a new one.

Create a new API ▼

API name

Enter a name to uniquely identify your API.

GreetingsAPI

Deployment stage

The name of your API's deployment stage.

Test

Security

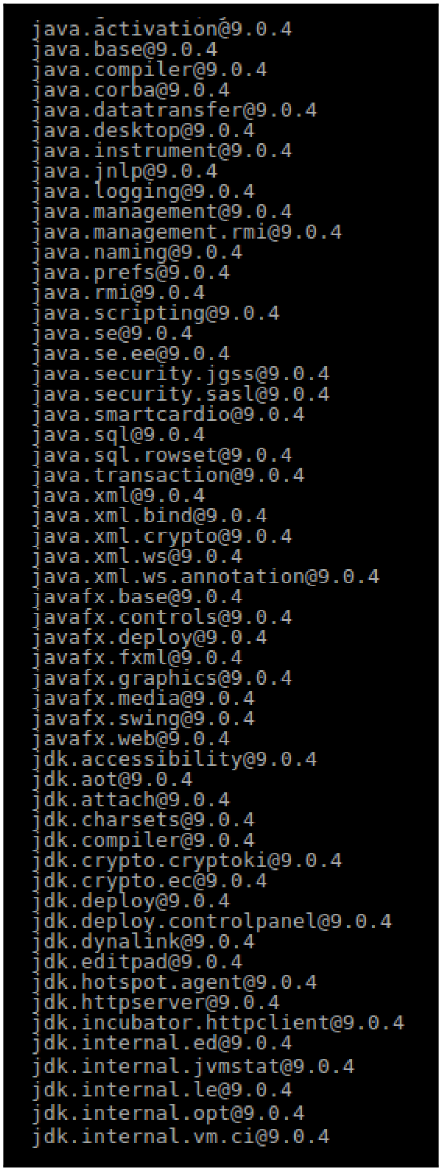
Configure the security mechanism for your API endpoint.

Open ▼

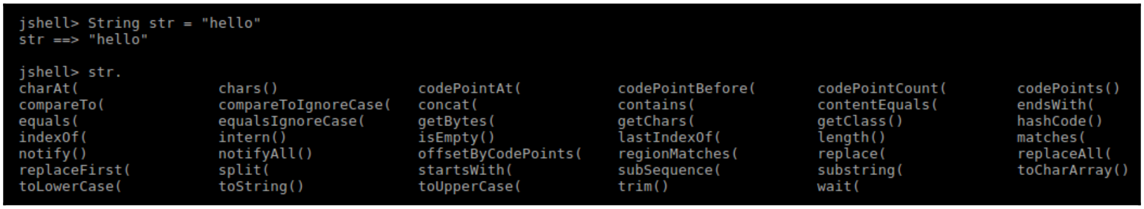
Warning: Your API endpoint will be publicly available and can be invoked by all users.

Lambda will add the necessary permissions for Amazon API Gateway to invoke your Lambda function from this trigger. Learn more about the Lambda permissions model.

Chapter 9: Best Practices in Java



```
java.activation@9.0.4
java.base@9.0.4
java.compiler@9.0.4
java.corba@9.0.4
java.datatransfer@9.0.4
java.desktop@9.0.4
java.instrument@9.0.4
java.jnlp@9.0.4
java.logging@9.0.4
java.management@9.0.4
java.management.rmi@9.0.4
java.naming@9.0.4
java.prefs@9.0.4
java.rmi@9.0.4
java.scripting@9.0.4
java.se@9.0.4
java.se.ee@9.0.4
java.security.jgss@9.0.4
java.security.sasl@9.0.4
java.smartcardio@9.0.4
java.sql@9.0.4
java.sql.rowset@9.0.4
java.transaction@9.0.4
java.xml@9.0.4
java.xml.bind@9.0.4
java.xml.crypto@9.0.4
java.xml.ws@9.0.4
java.xml.ws.annotation@9.0.4
javafx.base@9.0.4
javafx.controls@9.0.4
javafx.deploy@9.0.4
javafx.fxml@9.0.4
javafx.graphics@9.0.4
javafx.media@9.0.4
javafx.swing@9.0.4
javafx.web@9.0.4
jdk.accessibility@9.0.4
jdk.aot@9.0.4
jdk.attach@9.0.4
jdk.charsets@9.0.4
jdk.compiler@9.0.4
jdk.crypto.cryptoki@9.0.4
jdk.crypto.ec@9.0.4
jdk.deploy@9.0.4
jdk.deploy.controlpanel@9.0.4
jdk.dynalink@9.0.4
jdk.editpad@9.0.4
jdk.hotspot.agent@9.0.4
jdk.httpserver@9.0.4
jdk.incubator.httpclient@9.0.4
jdk.internal.ed@9.0.4
jdk.internal.jvmstat@9.0.4
jdk.internal.le@9.0.4
jdk.internal.opt@9.0.4
jdk.internal.vm.ci@9.0.4
```



```
jshell> String str = "hello"
str ==> "hello"

jshell> str.
charAt(                chars()                codePointAt(           codePointBefore(       codePointCount(        codePoints()
compareTo(             compareToIgnoreCase(   concat(                contains(              contentEquals(         endsWith(
equals(                equalsIgnoreCase(      getBytes(              getChars(              getClass()             hashCode()
indexOf(               intern()               isEmpty()              lastIndexOf(           length()               matches(
notify()               notifyAll()            offsetByCodePoints(    regionMatches(         replace(               replaceAll(
replaceFirst(          split(                 startsWith(            subSequence(           substring(             toCharArray()
toLowerCase(           toString()             toUpperCase(           trim()                 wait(
```