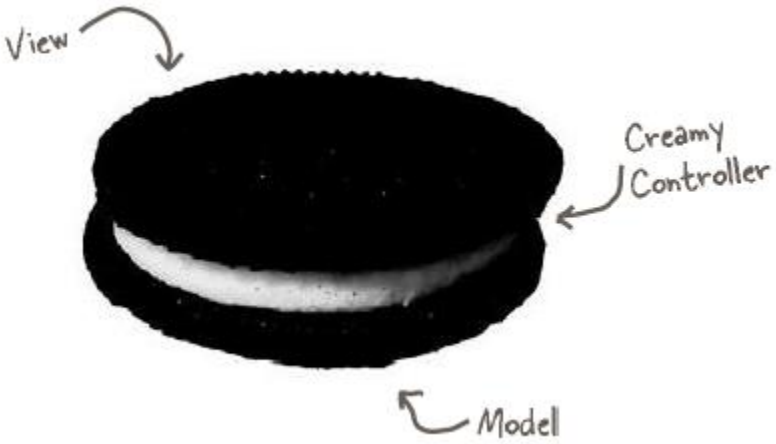
Patrones de Arquitectura
MVC (Model View Controller)

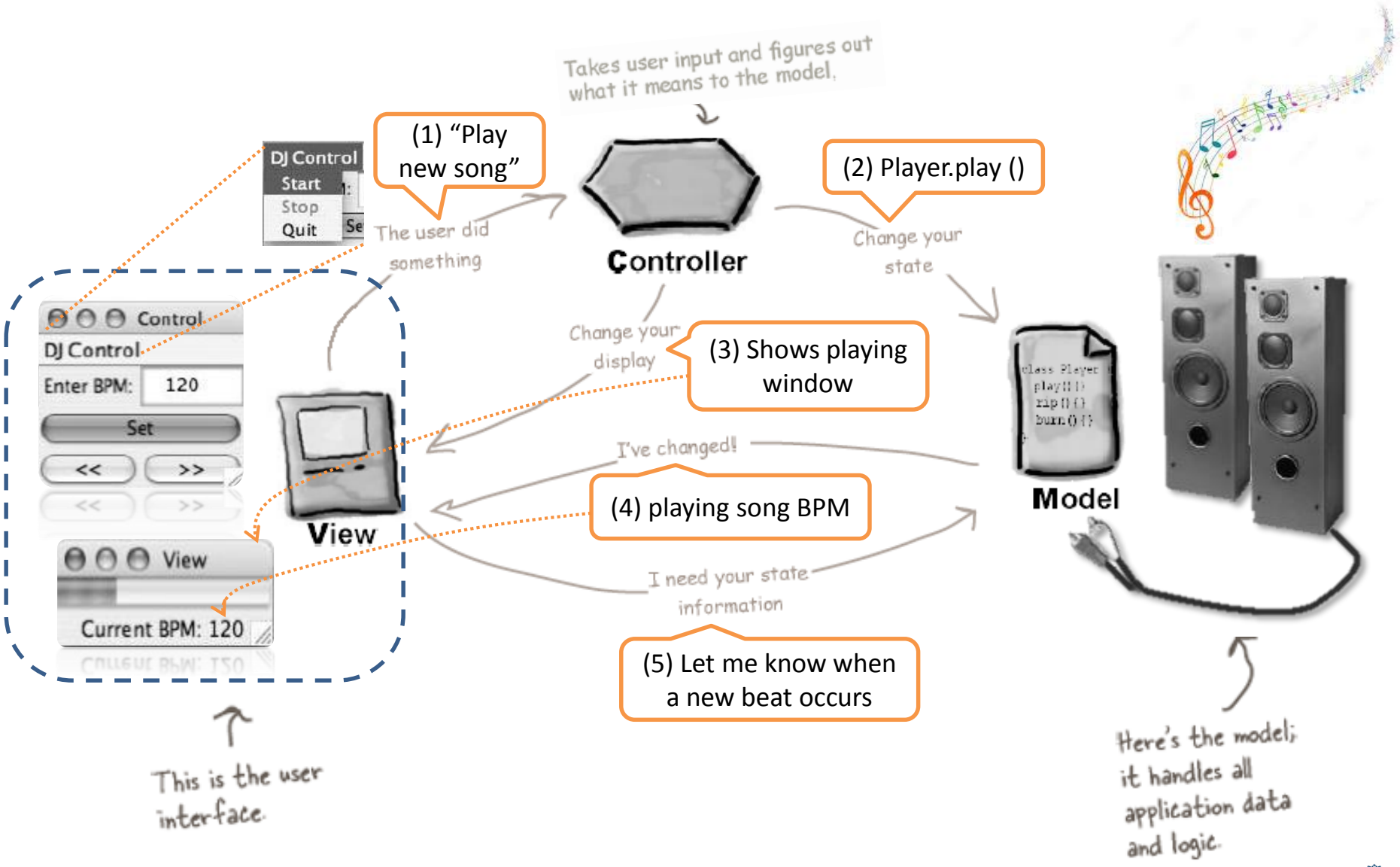


Patrones de diseño compuestos: Observer & Strategy Patterns

Ejemplo MVC: Beat and Heart Monitor

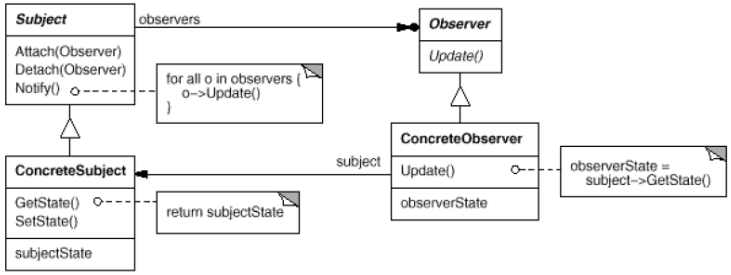
Fuente: HeadFirst DesignPattern

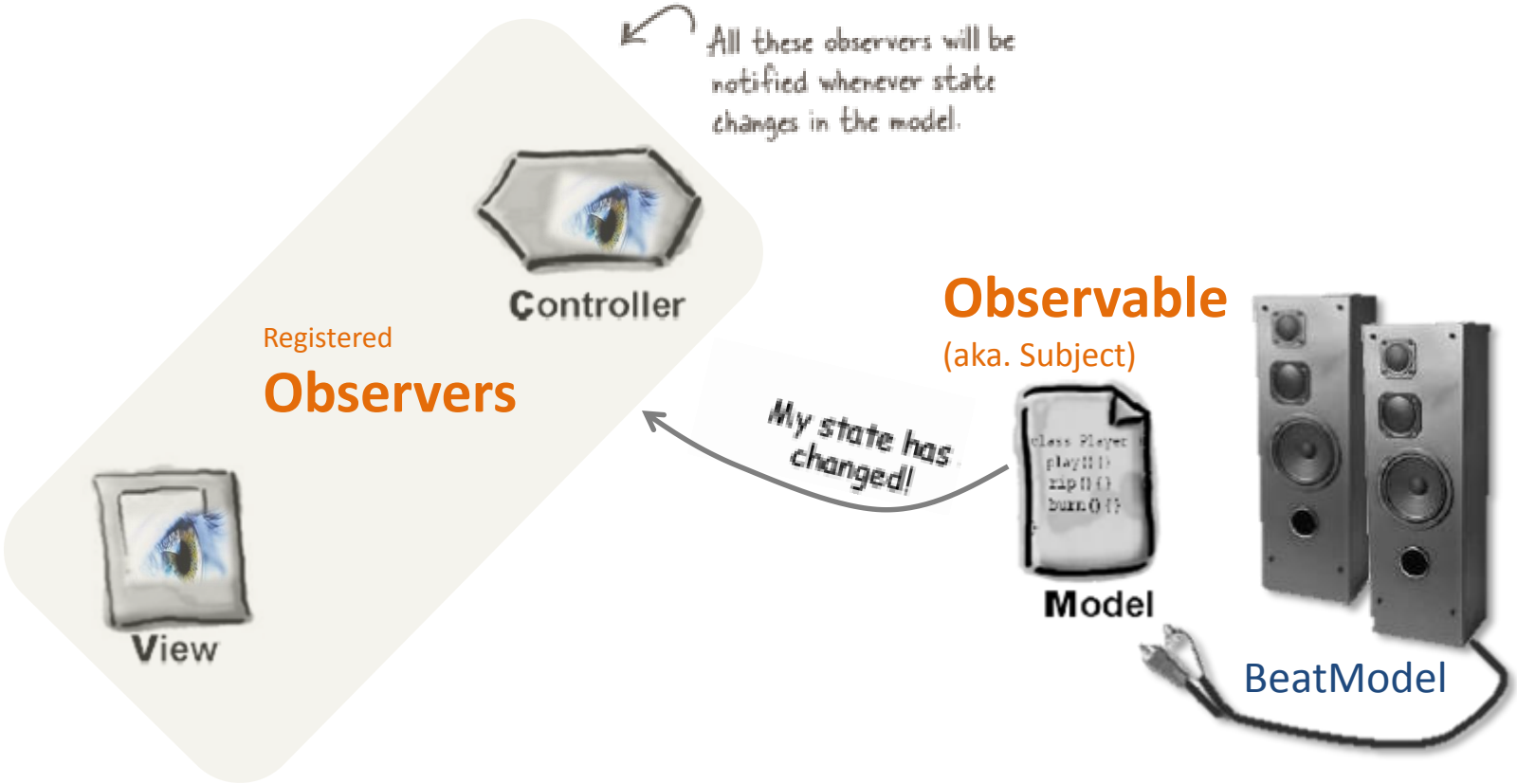
MVC: Model-View-Controller



y los patrones?

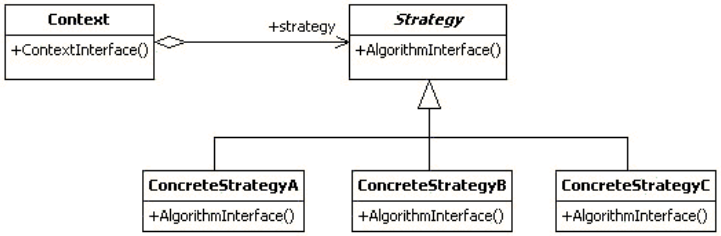
Observer…

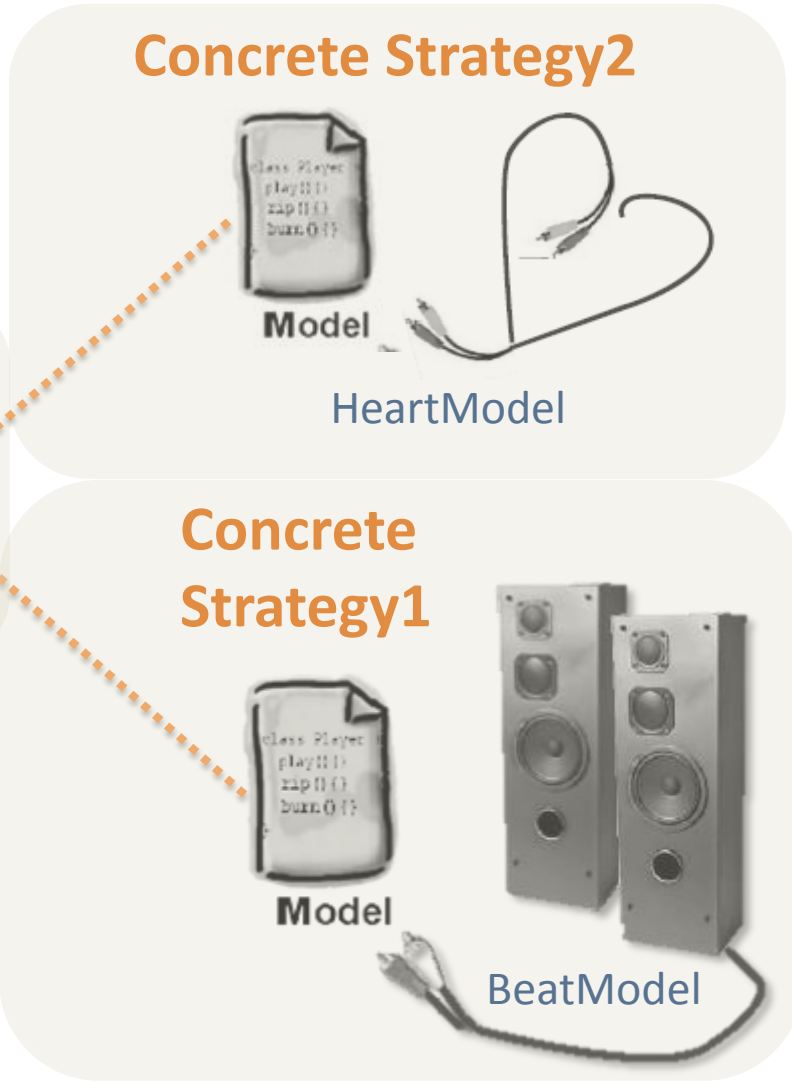


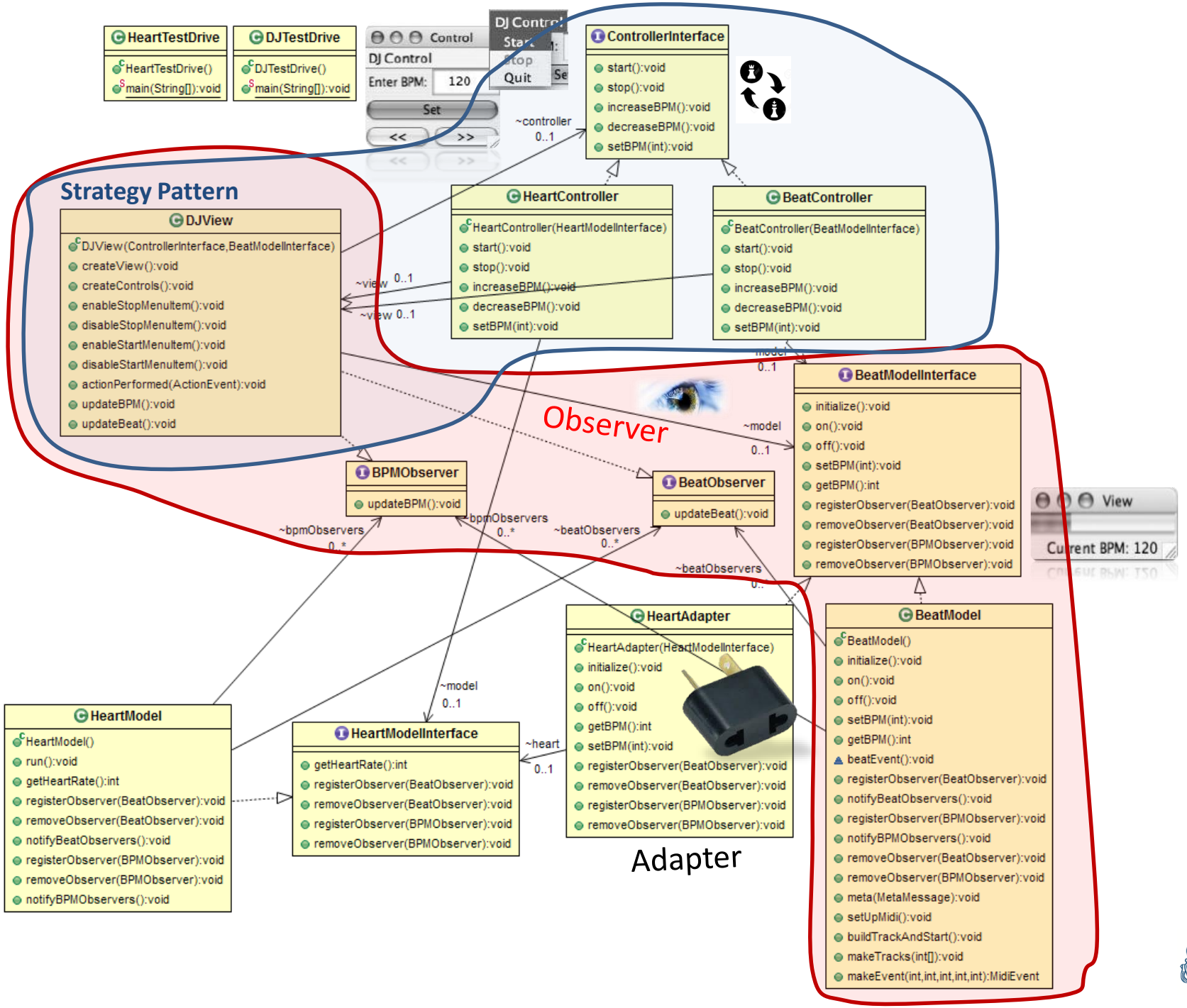


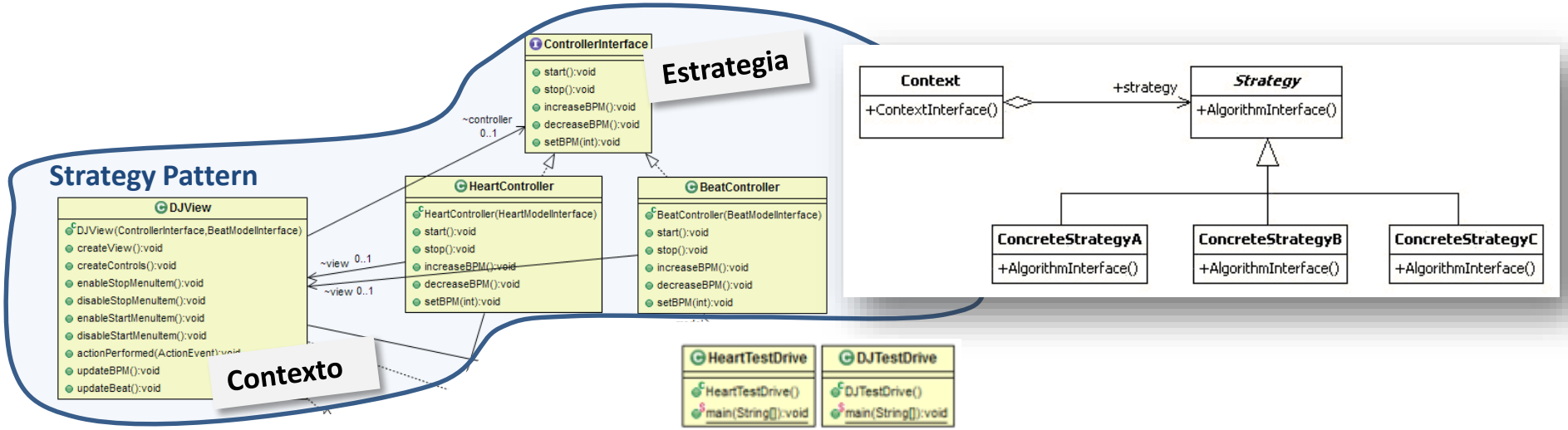
y los patrones?

Strategy…






HeartTestDrive
HeartTestDrive()
main(String[]):void
DJTestDrive
DJTestDrive()
main(String[]):void
Control
DJ Control
Enter BPM:
120
Set
Stop
Quit
ControllerInterface
start():void
stop():void
increaseBPM():void
decreaseBPM():void
setBPM(int):void
~controller
0..1
Strategy Pattern
DJView
DJView(ControllerInterface,BeatModelInterface)
createView():void
createControls():void
enableStopMenuItem():void
disableStopMenuItem():void
enableStartMenuItem():void
disableStartMenuItem():void
actionPerformed(ActionEvent):void
updateBPM():void
updateBeat():void
HeartController
HeartController(HeartModelInterface)
start():void
stop():void
increaseBPM():void
decreaseBPM():void
setBPM(int):void
BeatController
BeatController(BeatModelInterface)
start():void
stop():void
increaseBPM():void
decreaseBPM():void
setBPM(int):void
~view
0..1
~view 0..1
Observer
BeatModelInterface
initialize():void
on():void
off():void
setBPM(int):void
getBPM():int
registerObserver(BeatObserver):void
removeObserver(BeatObserver):void
registerObserver(BPMObserver):void
removeObserver(BPMObserver):void
~model
0..1
BPMObserver
updateBPM():void
BeatObserver
updateBeat():void
~bpmObservers
0..*
~beatObservers
0..*
View
Current BPM: 120
HeartAdapter
HeartAdapter(HeartModelInterface)
initialize():void
on():void
off():void
getBPM():int
setBPM(int):void
registerObserver(BeatObserver):void
removeObserver(BeatObserver):void
registerObserver(BPMObserver):void
removeObserver(BPMObserver):void
BeatModel
BeatModel()
initialize():void
on():void
off():void
setBPM(int):void
getBPM():int
beatEvent():void
registerObserver(BeatObserver):void
notifyBeatObservers():void
registerObserver(BPMObserver):void
notifyBPMObservers():void
removeObserver(BeatObserver):void
removeObserver(BPMObserver):void
meta(MetaMessage):void
setUpMidi():void
buildTrackAndStart():void
makeTracks(int[]):void
makeEvent(int,int,int,int,int):MidiEvent
HeartModel
HeartModel()
run():void
getHeartRate():int
registerObserver(BeatObserver):void
removeObserver(BeatObserver):void
notifyBeatObservers():void
registerObserver(BPMObserver):void
removeObserver(BPMObserver):void
notifyBPMObservers():void
HeartModelInterface
getHeartRate():int
registerObserver(BeatObserver):void
removeObserver(BeatObserver):void
registerObserver(BPMObserver):void
removeObserver(BPMObserver):void
~heart
0..1
Adapter



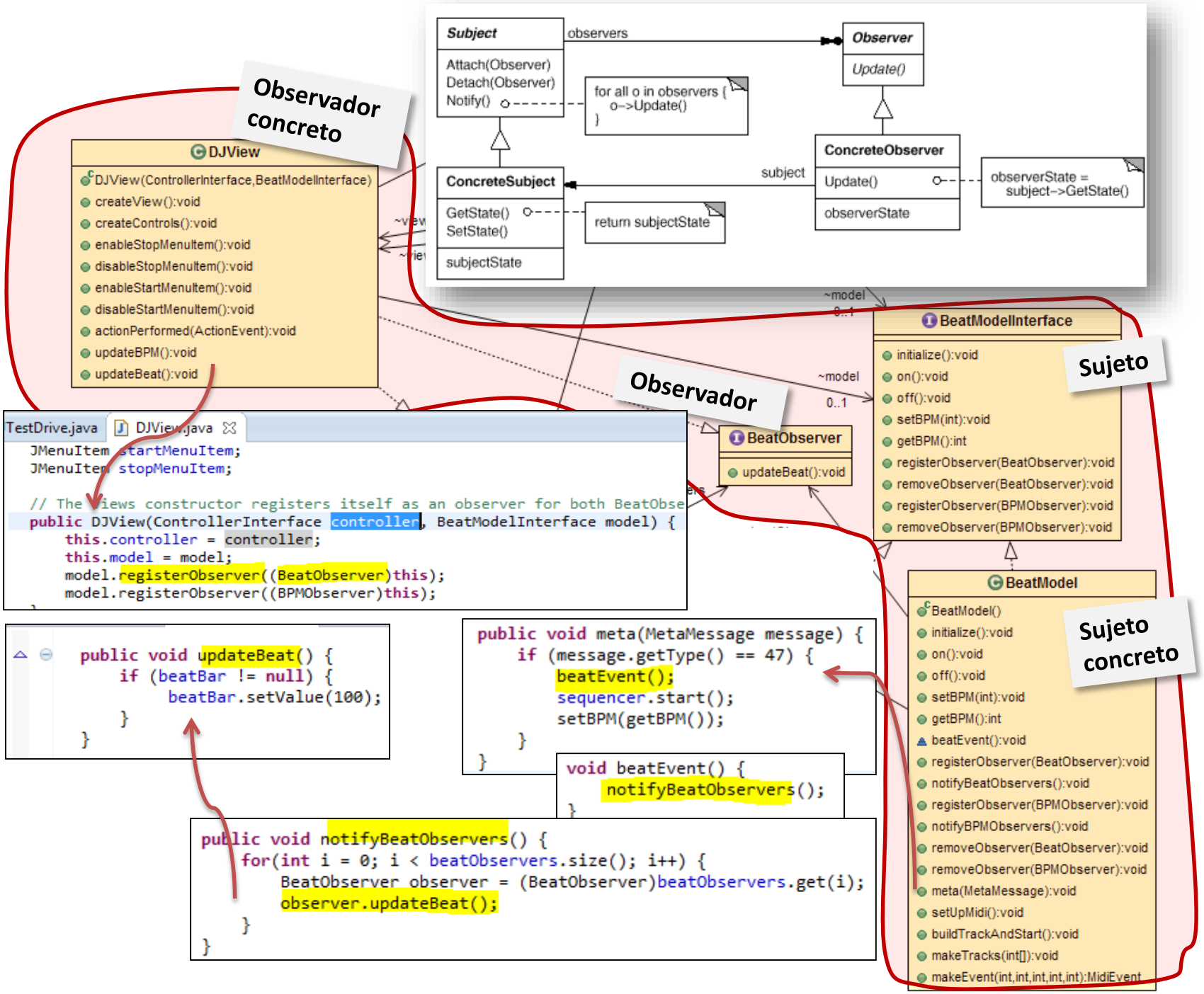

```
public class HeartTestDrive {

    public static void main (String[] args) {
        HeartModel heartModel = new HeartModel();
        @SuppressWarnings("unused")
        ControllerInterface heartController = new HeartController(heartModel);
    }
}
```

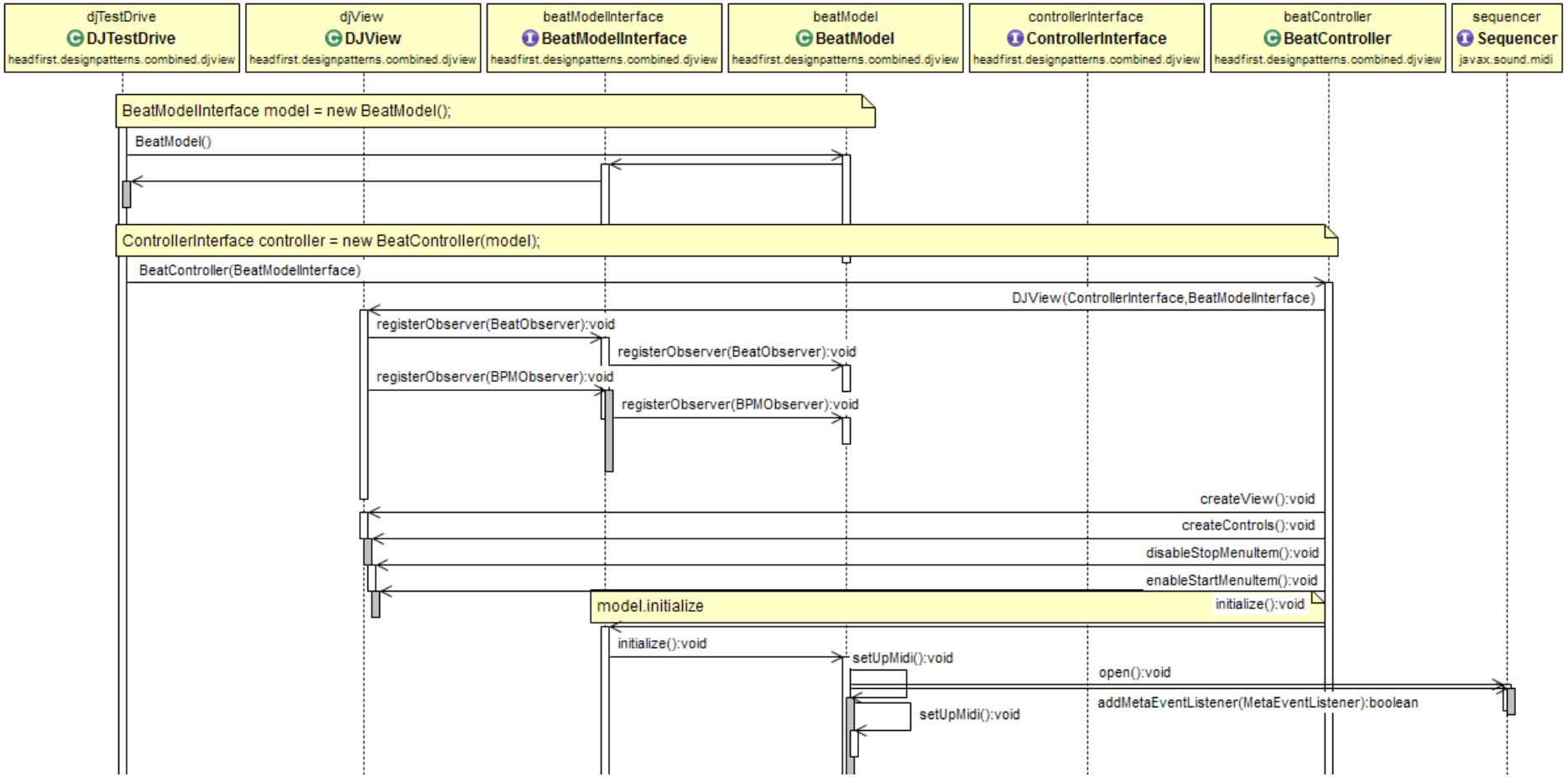
```
public class HeartController implements ControllerInterface {
    HeartModelInterface model;
    DJView view;

    public HeartController(HeartModelInterface model) {
        this.model = model;
        view = new DJView(this, new HeartAdapter(model));
        view.createView();
        view.createControls();
        view.disableStopMenuItem();
        view.disableStartMenuItem();
    }
```

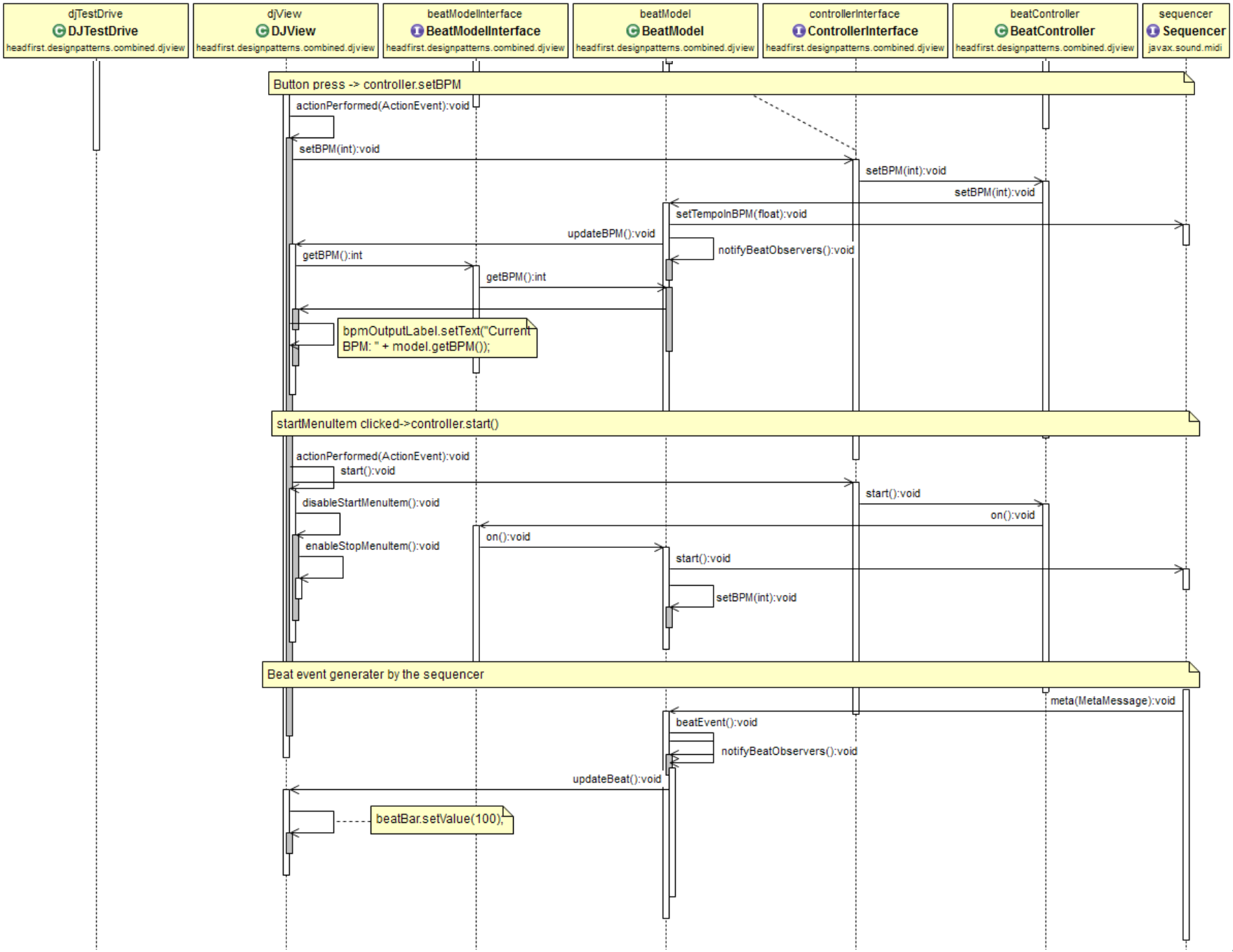
```
public class DJTestDrive {

    public static void main (String[] args) {
        BeatModelInterface model = new BeatModel();
        ControllerInterface controller = new BeatController(model);
    }
}
```

```
public class BeatController implements ControllerInterface {
    BeatModelInterface model;
    DJView view;

    public BeatController(BeatModelInterface model) {
        this.model = model;
        view = new DJView(this, model);
        view.createView();
        view.createControls();
        view.disableStopMenuItem();
        view.enableStartMenuItem();
        model.initialize();
    }
```

```
// The views constructor registers itself as an observer for both BeatObserver
public DJView(ControllerInterface controller, BeatModelInterface model) {
    this.controller = controller;
    this.model = model;
    model.registerObserver((BeatObserver)this);
    model.registerObserver((BPMObserver)this);
}
```



Observador concreto
Observador
Sujeto
Sujeto concreto
Subject
observers
Attach(Observer)
Detach(Observer)
Notify()
for all o in observers {
o->Update()
}
Observer
Update()
ConcreteSubject
GetState()
SetState()
subjectState
return subjectState
subject
ConcreteObserver
Update()
observerState
observerState = subject->GetState()
DJView
DJView(ControllerInterface,BeatModelInterface)
createView():void
createControls():void
enableStopMenuItem():void
disableStopMenuItem():void
enableStartMenuItem():void
disableStartMenuItem():void
actionPerformed(ActionEvent):void
updateBPM():void
updateBeat():void
~model
0..1
BeatModelInterface
initialize():void
on():void
off():void
setBPM(int):void
getBPM():int
registerObserver(BeatObserver):void
removeObserver(BeatObserver):void
registerObserver(BPMObserver):void
removeObserver(BPMObserver):void
BeatObserver
updateBeat():void
BeatModel
BeatModel()
initialize():void
on():void
off():void
setBPM(int):void
getBPM():int
beatEvent():void
registerObserver(BeatObserver):void
notifyBeatObservers():void
registerObserver(BPMObserver):void
notifyBPMObservers():void
removeObserver(BeatObserver):void
removeObserver(BPMObserver):void
meta(MetaMessage):void
setUpMidi():void
buildTrackAndStart():void
makeTracks(int[]):void
makeEvent(int,int,int,int,int):MidiEvent
TestDrive.java
DJView.java
JMenuItem startMenuItem;
JMenuItem stopMenuItem;
// The views constructor registers itself as an observer for both BeatObse
public DJView(ControllerInterface controller, BeatModelInterface model) {
this.controller = controller;
this.model = model;
model.registerObserver((BeatObserver)this);
model.registerObserver((BPMObserver)this);
public void updateBeat() {
if (beatBar != null) {
beatBar.setValue(100);
}
}
public void meta(MetaMessage message) {
if (message.getType() == 47) {
beatEvent();
sequencer.start();
setBPM(getBPM());
}
}
void beatEvent() {
notifyBeatObservers();
}
public void notifyBeatObservers() {
for(int i = 0; i < beatObservers.size(); i++) {
BeatObserver observer = (BeatObserver)beatObservers.get(i);
observer.updateBeat();
}
}


djTestDrive
DJTestDrive
headfirst.designpatterns.combined.djview
djView
DJView
headfirst.designpatterns.combined.djview
beatModelInterface
BeatModelInterface
headfirst.designpatterns.combined.djview
beatModel
BeatModel
headfirst.designpatterns.combined.djview
controllerInterface
ControllerInterface
headfirst.designpatterns.combined.djview
beatController
BeatController
headfirst.designpatterns.combined.djview
sequencer
Sequencer
javax.sound.midi
BeatModelInterface model = new BeatModel();
BeatModel()
ControllerInterface controller = new BeatController(model);
BeatController(BeatModelInterface)
DJView(ControllerInterface,BeatModelInterface)
registerObserver(BeatObserver):void
registerObserver(BeatObserver):void
registerObserver(BPMObserver):void
registerObserver(BPMObserver):void
createView():void
createControls():void
disableStopMenuItem():void
enableStartMenuItem():void
model.initialize
initialize():void
initialize():void
setUpMidi():void
open():void
addMetaEventListener(MetaEventListener):boolean
setUpMidi():void


djTestDrive
DJTestDrive
headfirst.designpatterns.combined.djview
djView
DJView
headfirst.designpatterns.combined.djview
beatModelInterface
BeatModelInterface
headfirst.designpatterns.combined.djview
beatModel
BeatModel
headfirst.designpatterns.combined.djview
controllerInterface
ControllerInterface
headfirst.designpatterns.combined.djview
beatController
BeatController
headfirst.designpatterns.combined.djview
sequencer
Sequencer
javax.sound.midi
Button press -> controller.setBPM
actionPerformed(ActionEvent):void
setBPM(int):void
setBPM(int):void
setBPM(int):void
setTempoInBPM(float):void
updateBPM():void
notifyBeatObservers():void
getBPM():int
getBPM():int
bpmOutputLabel.setText("Current BPM: " + model.getBPM());
startMenuItem clicked->controller.start()
actionPerformed(ActionEvent):void
start():void
start():void
disableStartMenuItem():void
on():void
on():void
enableStopMenuItem():void
start():void
setBPM(int):void
Beat event generater by the sequencer
meta(MetaMessage):void
beatEvent():void
notifyBeatObservers():void
updateBeat():void
beatBar.setValue(100);