



Patrón de diseño

# Observer Design.

JOSE ANTONIO PEREZ DURAN  
ISAAC EDUARDO ODRIozOLA DIAZ





# Definición

Define una dependencia de uno a muchos objetos de tal manera que cuando un objeto cambio de estado, todos sus dependientes son notificados y actualizados automáticamente

Del libro «Design patterns: elements of reusable object-oriented software»

# Uso de observer Suscripciones y Notificaciones



YouTube MX

Buscar

Iniciar Explorar Suscripciones Biblioteca

Xbox 4,45 M de suscriptores SUSCRITO

xbox button but it's everywhere

Xbox 388.207 visualizaciones • hace 3 semanas

welcome to my life Subscribe to Xbox <https://xbx.lv/2EEjmaR> FOLLOW XBOX: Facebook: <https://www.facebook.com/Xbox> Twitter: <https://www.twitter.com/Xbox> Instagram: <https://www.instagram.com/Xbox>

Notificaciones

- HBO Max ha subido: Adventure Time: Distant Lands - Together Again | HBO Max hace 13 horas
- Apple ha subido: iPhone 12 | Mmmmm, purple | Apple hace 18 horas
- Recomendación: Diego Boneta - Suave (Letra/Lyrics) hace 18 horas
- Microsoft Surface ha subido: Hospital gains transformational efficiency and effectiveness boost from Surface Hub 2S hace 1 día
- Lenovo ha subido: Lenovo ThinkPad X Series Gen 2 Product Tour Video hace 1 día



# ¿Cómo funciona Observer?

01

Por medio de una clase observador podemos hacer una relación uno a muchos objetos para saber cual fue modificado.

02

Para poder utilizar el patron observer es necesario definir el alcance para las clases por observar.

03

Se necesitan iniciar los constructores para su correcto funcionamiento





# Ejemplo:

Supongamos que tenemos clases con java y vamos a modificar las entradas para que cambien los valores numéricos en la entrada

```
public class ObserverPatternDemo {  
    public static void main(String[] args) {  
        Subject subject = new Subject();  
  
        new HexaObserver(subject);  
        new OctalObserver(subject);  
        new BinaryObserver(subject);  
  
        System.out.println("First state change: 15");  
        subject.setState(15);  
        System.out.println("Second state change: 10");  
        subject.setState(10);  
    }  
}
```



```
First state change: 15  
Hex String: F  
Octal String: 17  
Binary String: 1111  
Second state change: 10  
Hex String: A  
Octal String: 12  
Binary String: 1010
```



# Código:

Pensemos en lo siguiente:  
Tenemos las clases  
Observer, Video y Usuario



```
1 // Publisher
2 class Video {
3   constructor(observable, name, content) {
4     this.observable = observable;
5     this.name = name;
6     this.content = content;
7     // publish the 'video-uploaded' event
8     this.observable.publish('video-uploaded', {
9       name,
10      content,
11    });
12  }
13 }
14 // Subscriber
15 class User {
16   constructor(observable) {
17     this.observable = observable;
18     this.interestedVideos = [];
19     // subscribe with the event name and the call back function
20     this.observable.subscribe('video-uploaded', this.addVideo.bind(this));
21   }
22
23   addVideo(video) {
24     this.interestedVideos.push(video);
25   }
26 }
27 // Observer
28 class Observable {
29   constructor() {
30     this.handlers = [];
31   }
32
33   subscribe(event, handler) {
34     this.handlers[event] = this.handlers[event] || [];
35     this.handlers[event].push(handler);
36   }
37
38   publish(event, eventData) {
39     const eventHandlers = this.handlers[event];
40
41     if (eventHandlers) {
42       for (var i = 0, l = eventHandlers.length; i < l; ++i) {
43         eventHandlers[i].call({}, eventData);
44       }
45     }
46   }
47 }
48 // usage
49 const observable = new Observable();
50 const user = new User(observable);
51 const video = new Video(observable, 'ES6 Design Patterns', videoFile);
```



# Ejercicio:

Deberás completar el siguiente código, donde crearas una clase para incrementar el valor numérico de su propiedad y cada vez que este valor sea alterado se deberá notificar el cambio de este estado a través de la consola.



==



# Preguntas y Respuestas





¡Gracias por su  
atención!

