Ejemplo de Patrón Observer

DigitalHouse >
Coding School

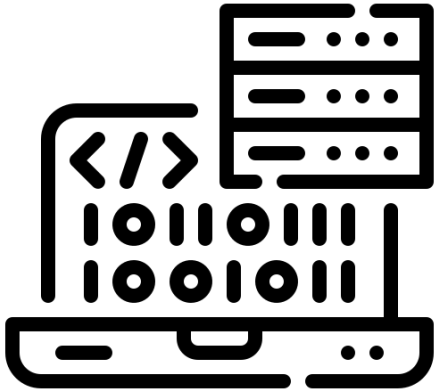
Certified Tech Developer
The Ultimate Degree

Ejemplo de Patrón Observer

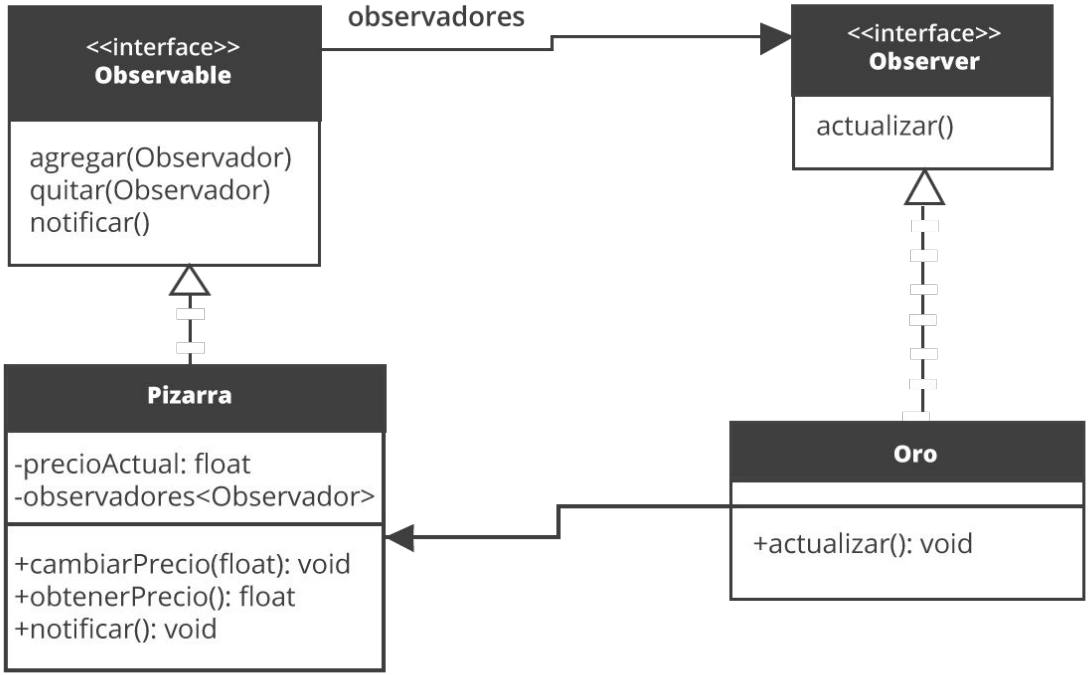
Ahora imaginemos una implementación.

Supongamos que tenemos un sistema que muestra en una pizarra el precio del oro y que al actualizarse se sincroniza con cada instancia suscrita informando el cambio del precio.

Veamos el diagrama UML a seguir.



Solución


<<interface>>
Observable
agregar(Observador)
quitar(Observador)
notificar()
observadores
<<interface>>
Observer
actualizar()
Pizarra
-precioActual: float
-observadores<Observador>
+cambiarPrecio(float): void
+obtenerPrecio(): float
+notificar(): void
Oro
+actualizar(): void

Implementación de las clases del diagrama UML

Creación de interfaces “Observador” y “Observable”.

```
public interface Observador{

      String actualizar();

}
```

Código de la interface **Observador**

```
public interface Observable{

      void agregar(Observador o);

      void quitar(Observador o);

      void notificar(String cambio);


}
```

Código de la interface **Observable**

Implementación de las clases del diagrama UML

```
public class Pizarra implements Observable{
      private float precioActual();
       private ArrayList<Observador> observadores = new
       ArrayList<>();
      public void cambiarPrecio(float precio){
            this.precioActual = precio;
            notificar(" precio actualizado a "+obtenerPrecio());
      }
      public float obtenerPrecio(){return precioActual;}
      } ...
```

Código de la clase **Pizarra**

Implementación de las clases del diagrama UML

```
…

@Override

public void agregar(Observador o){

      this.observadores.add(o);

}

@Override

public void quitar(Observador o){

      this.observadores.remove(o);

}

...
```

Código de la clase **Pizarra**

Implementación de las clases del diagrama UML

```
        …

        @Override

        public void notificar(String cambio){

                for(Observador o : observadores)

                System.out.print(o.actualizar() + cambio);

        }

}
```

Código de la clase **Pizarra**

Implementación de las clases del diagrama UML

Para implementar a la interface Observador creamos la clase **Oro** y declaramos su método:

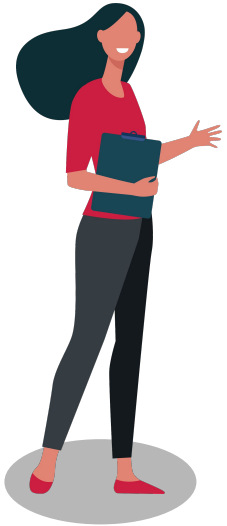
```
public class Oro implements Observador{

      @Override

      public String actualizar(){

            return this + "> Cambio de estado: ";

      }


}
```

Código de la clase **Oro**

```
public class Main{

      public static void main(String[] args){

            Pizarra pizarra = new Pizarra();

            Observador obs1 = new Oro();

            Observador obs1 = new Oro();


            pizarra.agregar(obs1);

            pizarra.agregar(obs1);


            pizarra.cambiarPrecio(42.5f);

            pizarra.cambiarPrecio(44.3f);

      }

 }
```

Código de la clase **Main**

Salida de datos



```
Oro@568db2f2> Cambio de estado:  precio actualizado a 42.5
Oro@378bf509> Cambio de estado:  precio actualizado a 42.5
Oro@568db2f2> Cambio de estado:  precio actualizado a 44.3
Oro@378bf509> Cambio de estado:  precio actualizado a 44.3

Process finished with exit code 0
```


DigitalHouse>
Coding School