



Patrones de diseño 2

OBSERVER PATTERN

OBSERVER PATTERN

OBJECT - BEHAVIORAL

► Intención:

- La intención de este patrón es proveer una forma de comunicación entre objetos con bajo acoplamiento.
- En este patrón existe un objeto que observar y observadores atentos a cambios en el estado del observable

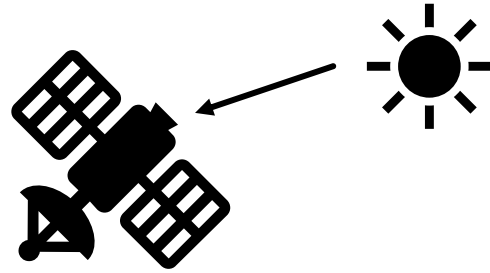
► Motivación:

- Hay ciertos escenarios donde un objeto necesite estar pendiente de los cambios en el estado de algún otro objeto. Usando el patrón Observer logaras este objetivo sin tener que tener un código fuertemente acoplado.

Voyager



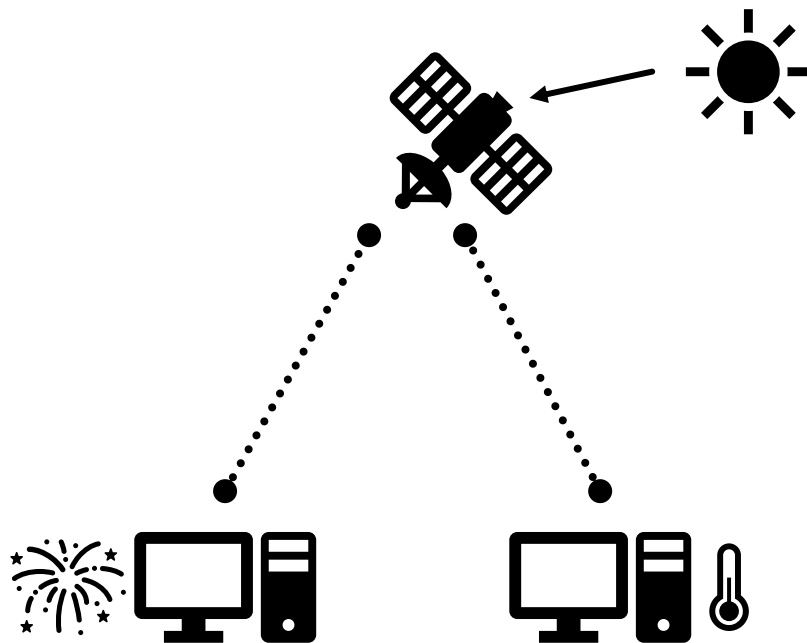
Voyager Sistema de monitored del sol



El Voyager Tiene un nuevo Sistema que mide la temperatura del sol cada 2 segundos.

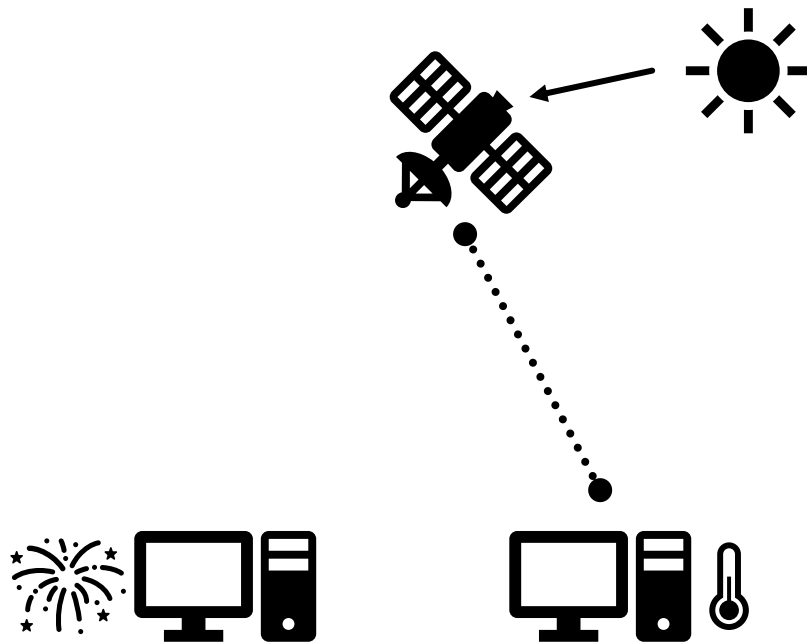


Voyager Sistema de monitored del sol



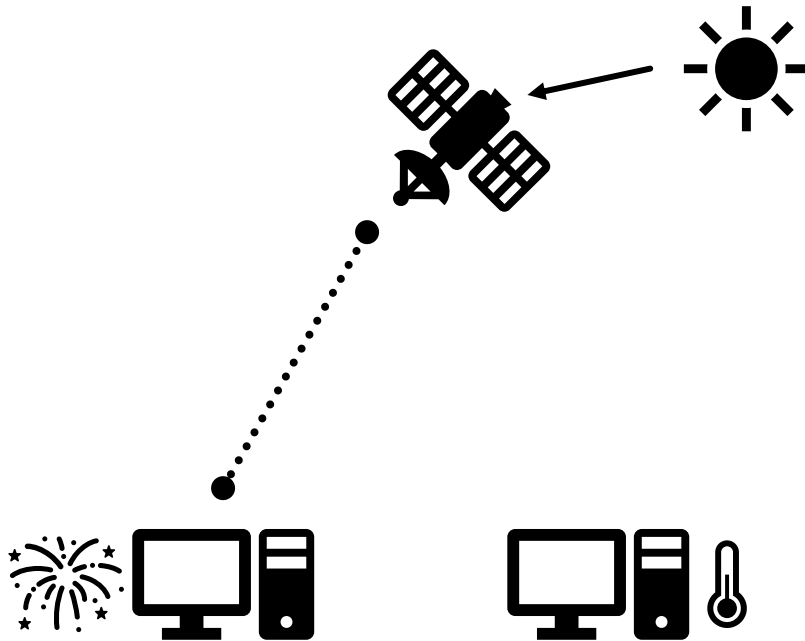
La NASA quiere implementar dos sistemas para registrar cambios en la temperatura solar desde la tierra.

Voyager Sistema de monitored del sol



El primer sistema debe registrar la temperatura actual del sol, y la mas alta y mas baja registrada.

Voyager Sistema de monitored del sol



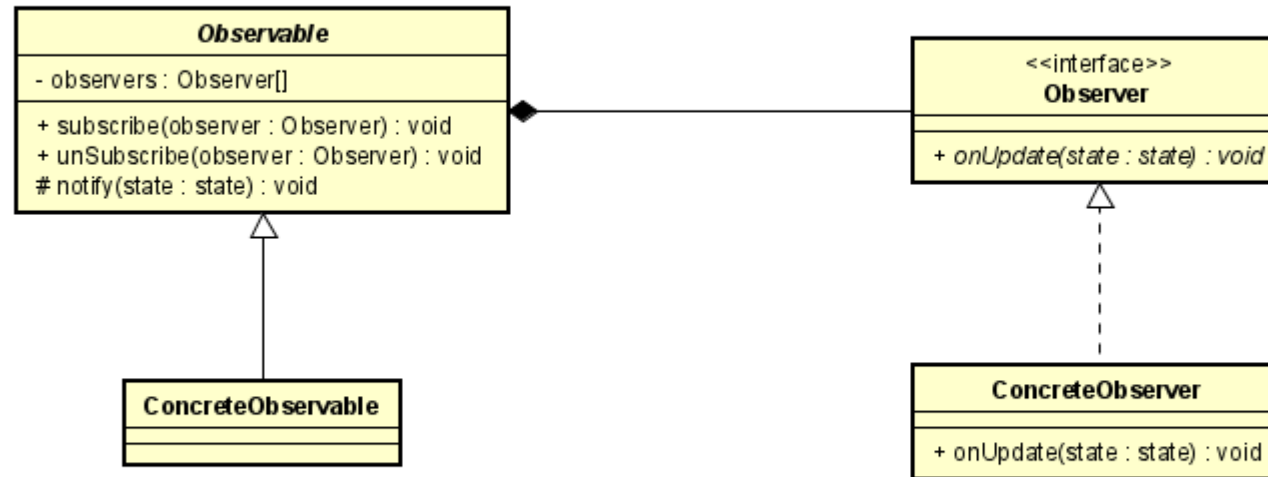
El segundo debe mandar una alerta cuando haya una erupción solar. Una erupción solar se registra cuando se da un subidón de temperatura en la superficie solar de mas de 400 mil grados centígrados.



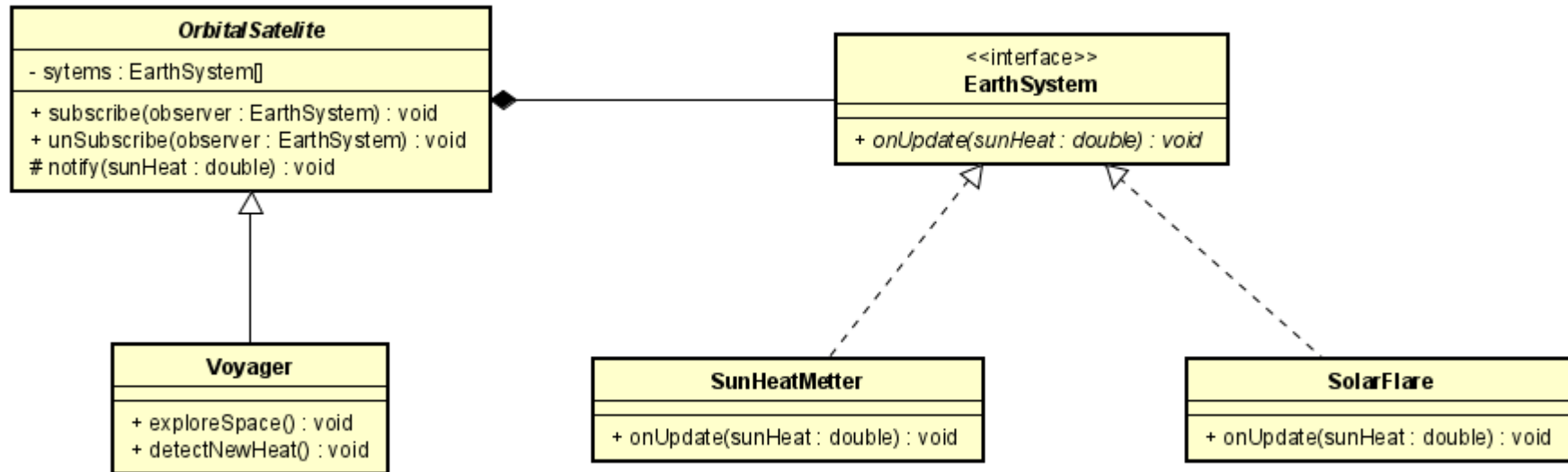
```
export default class Voyager {  
  
  async exploreSpace(): void {  
    //init the voyager system  
  }  
  
  private detectNewHeat(): Promise<any> {  
    //trigger every time a new temperature is registered in the voyager  
  }  
  
}
```

El API del voyager

Implementing the Observer



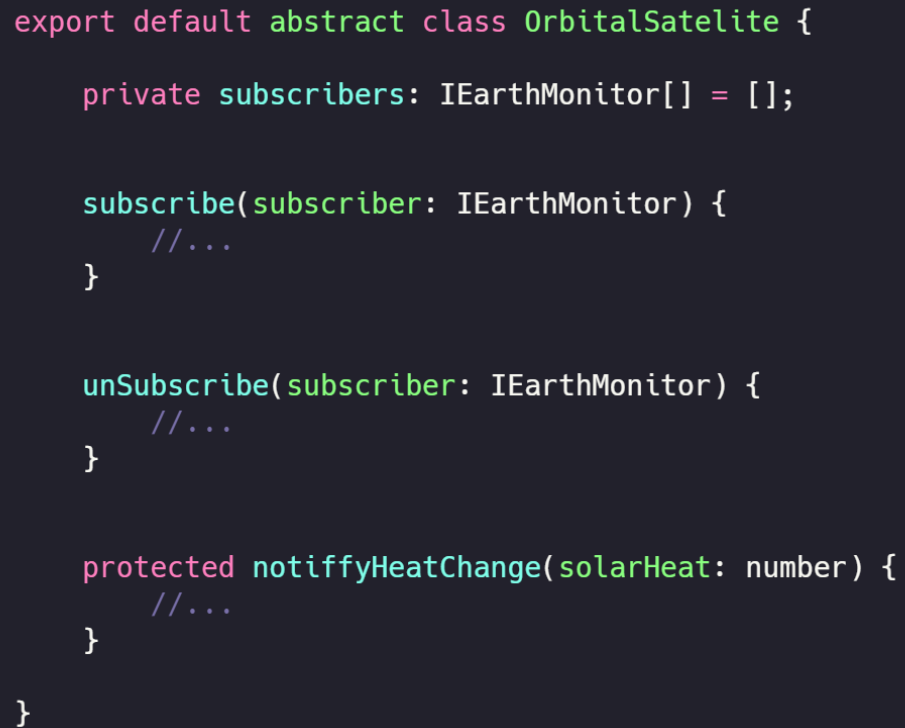
Implementing the Observer





```
export default interface IEarthMonitor {  
  onUpdate(solarHeat: number): void;  
}
```

Interfaces Observer y Observable



```
export default abstract class OrbitalSatelite {  
    private subscribers: IEarthMonitor[] = [];  
  
    subscribe(subscriber: IEarthMonitor) {  
        //...  
    }  
  
    unsubscribe(subscriber: IEarthMonitor) {  
        //...  
    }  
  
    protected notifyHeatChange(solarHeat: number) {  
        //...  
    }  
}
```



Interfaces Observer y Observable



```
export default class Voyager extends OrbitalSatelite {  
    async exploreSpace() {  
        //...  
    }  
  
    private detectNewHeat(): Promise<any> {  
        //...  
        this.notifyHeatChange(solarHeat);  
    }  
}
```

Voyager implementation

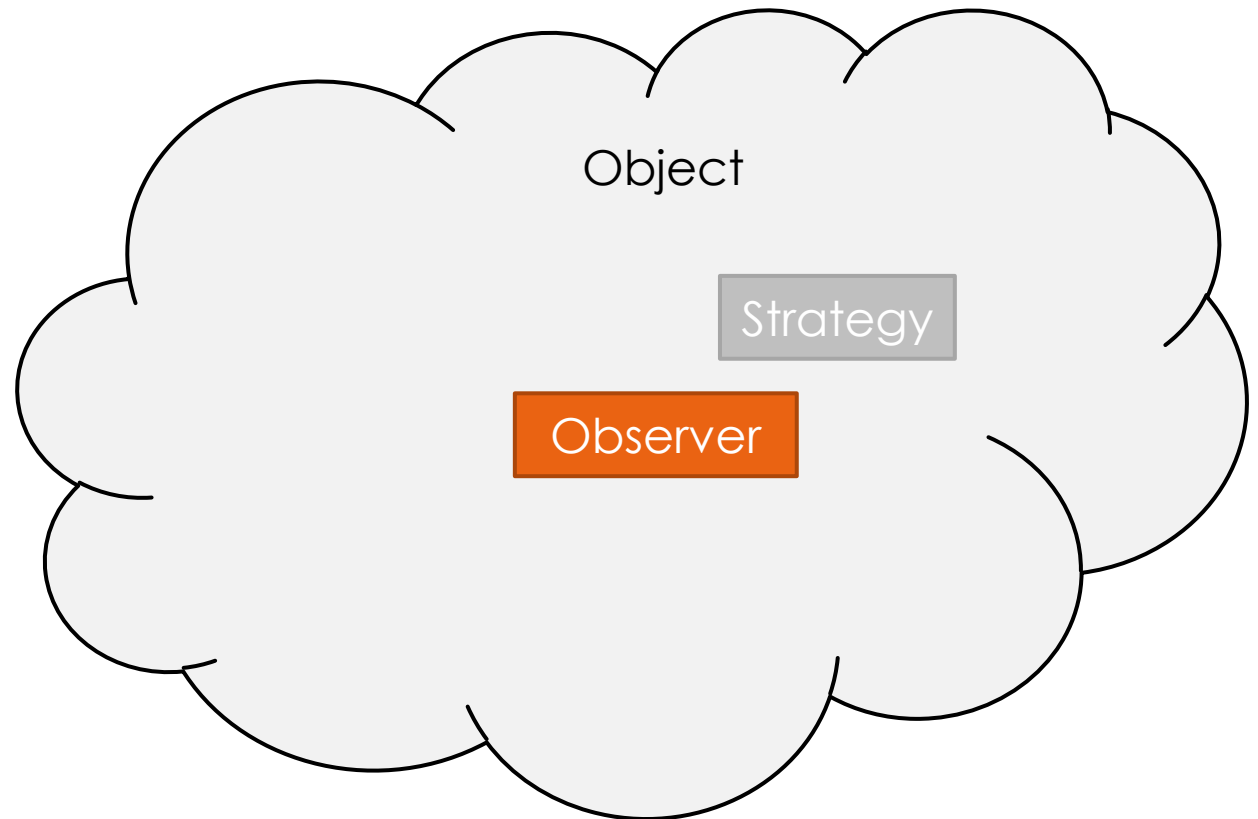
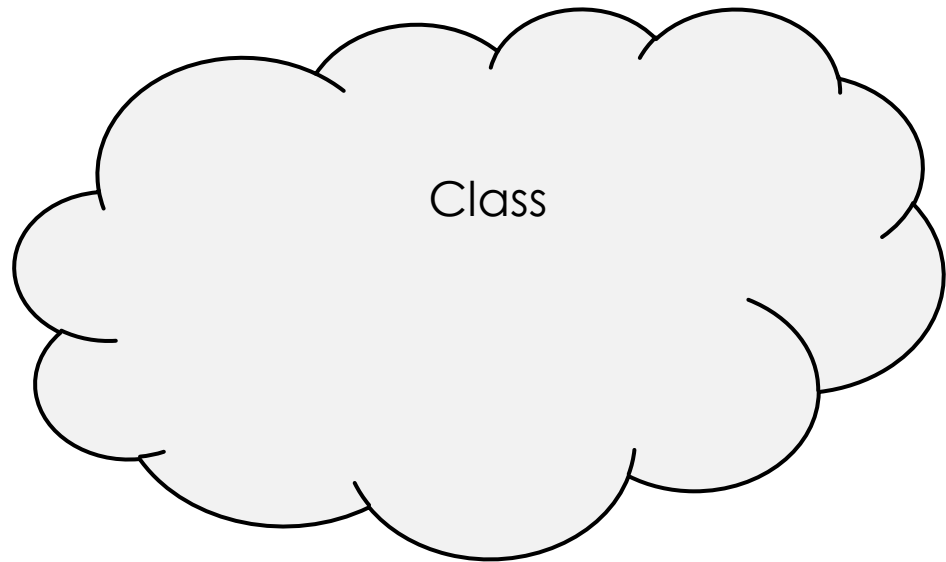


```
export default class SunHeatMetter implements IEarthMonitor {  
  
  constructor(  
    private satellite: OrbitalSatelite  
  ) {  
    this.satellite.subscribe( this );  
  }  
  
  onUpdate(solarHeat: number): void {  
    //...  
  }  
  
}
```

SunHeatMetter impementation

Categorías de patrones

Objects – Classes



Categorías de patrones

Creational – Behavioral – Structural

