

Padrões de Projeto

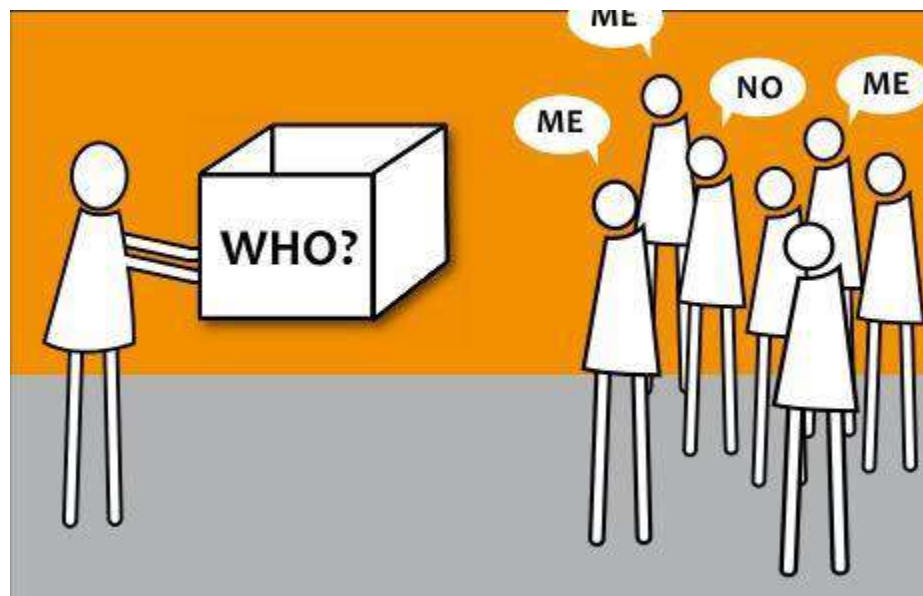
Prof. Adilson Vahldick

Departamento de Engenharia de Software

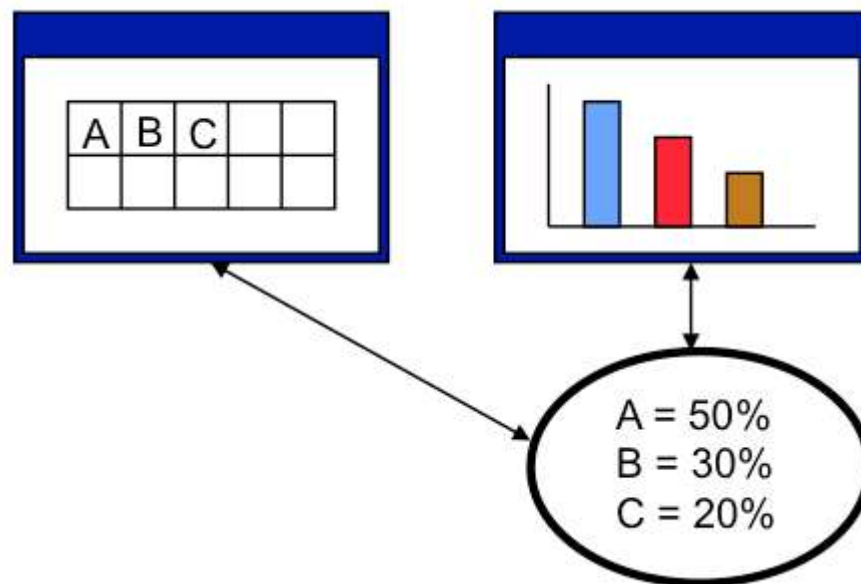
Udesc Ibirama

Objetivos da aula

- Conhecer e aplicar o padrão
 - Observer

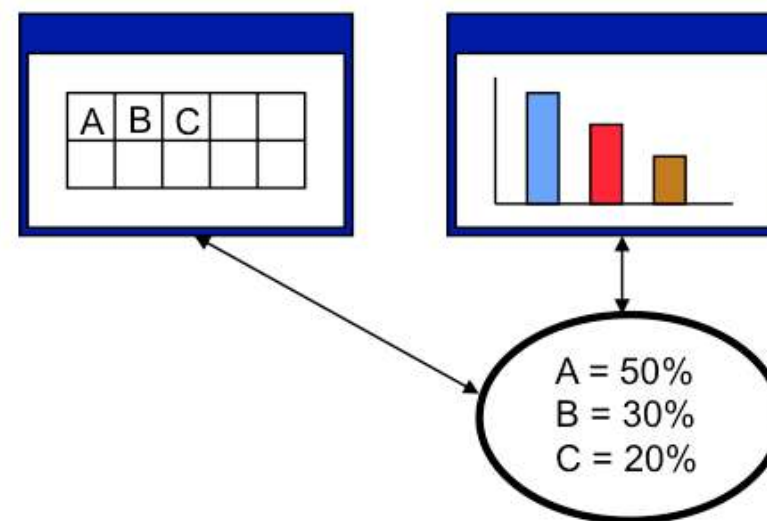


Problema (1)

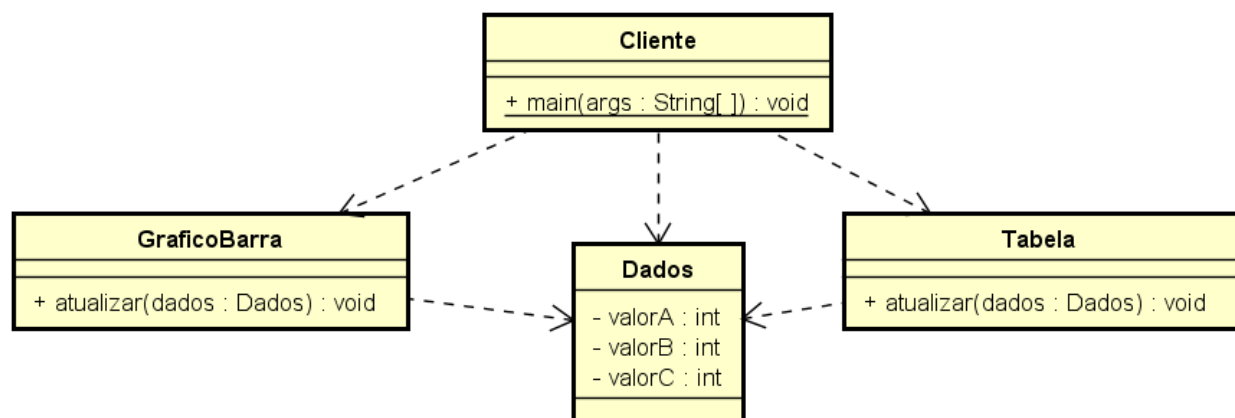


Problema (2)

- Sempre que os dados forem alterados, é preciso atualizar cada uma de suas representações
- O cliente pode exigir novas representações (requisitos mudam !!!)



Problema (3)



powered by Astah

```
Dados d = new Dados(7, 3, 1);
GráficoBarra gb = new GráficoBarra();
Tabela tb = new Tabela();
```

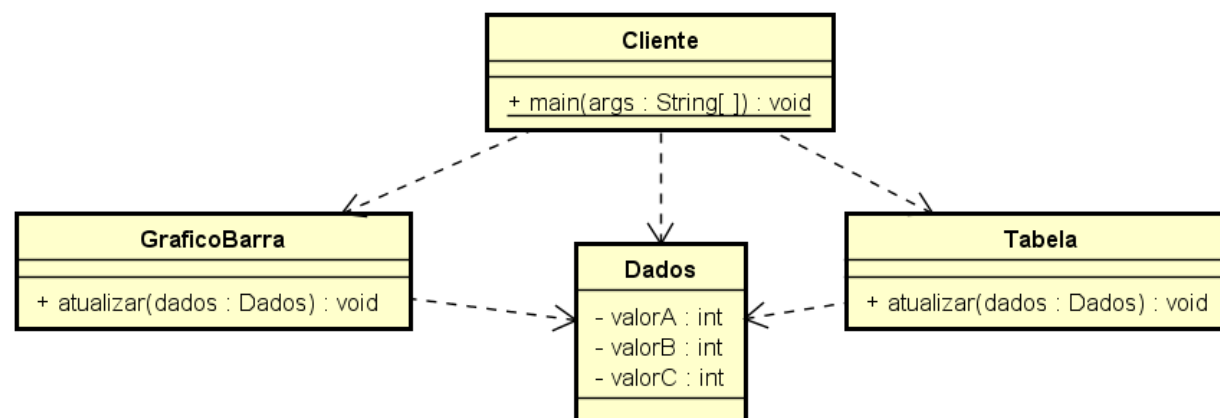
```
gb.atualizar(d);
tb.atualizar(d);
```

```
d.setValorA(100);
gb.atualizar(d);
tb.atualizar(d);
```

Objetivos de Qualquer Solução OO

... úteis para aplicação de Observer

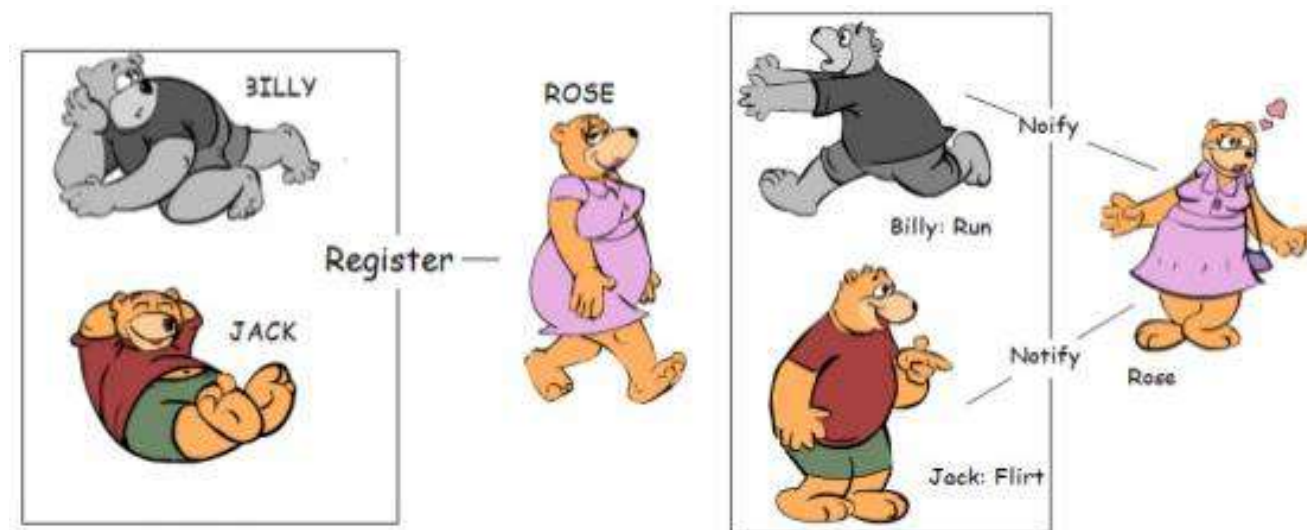
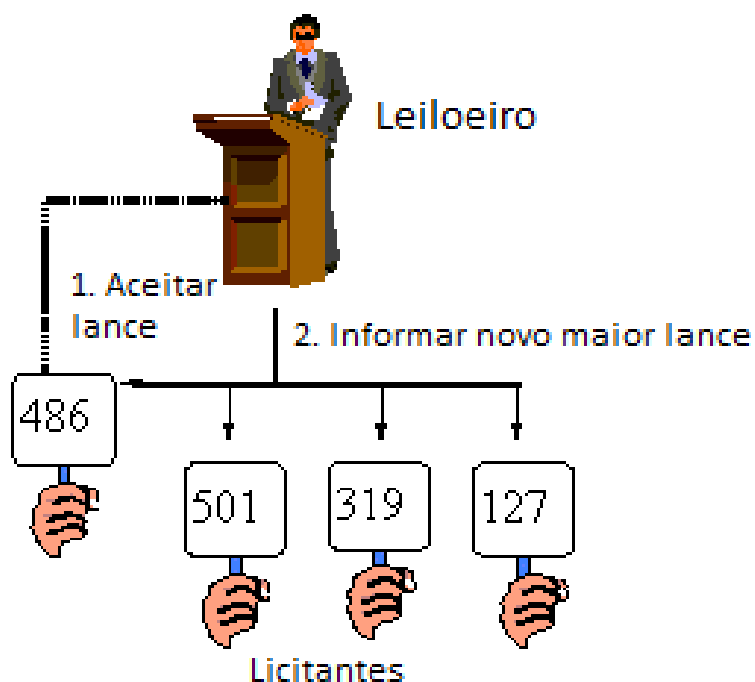
- Programe para uma interface, não para uma implementação
- Baixar o **acoplamento**



powered by Astah

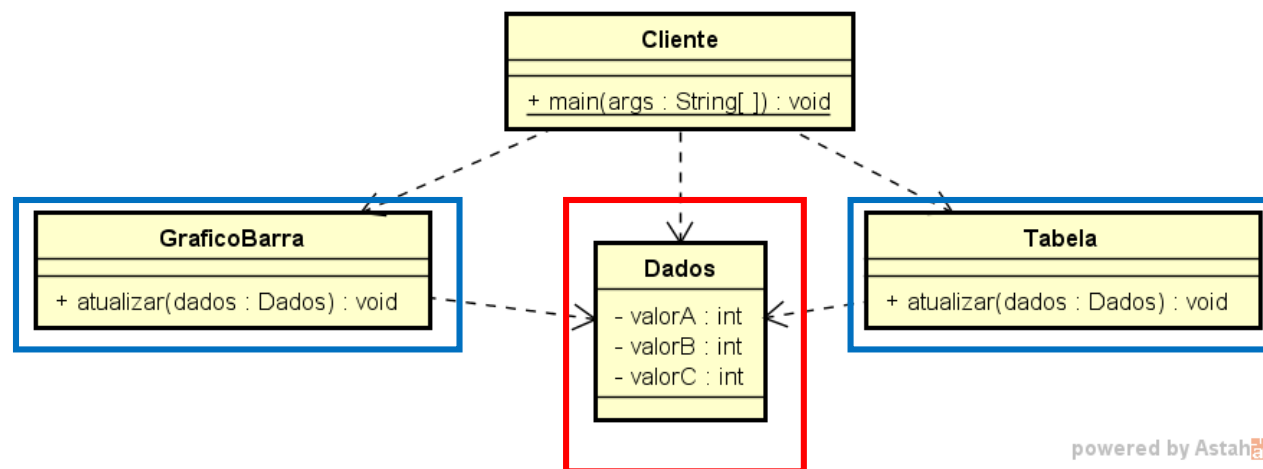
Observer (1)

- **Observer:** definir uma dependência um-para-muitos entre objetos, de maneira que quando um objeto muda de estado todos os seus dependentes são notificados e atualizados automaticamente.
- Os dependentes podem pertencer a diferentes tipos.

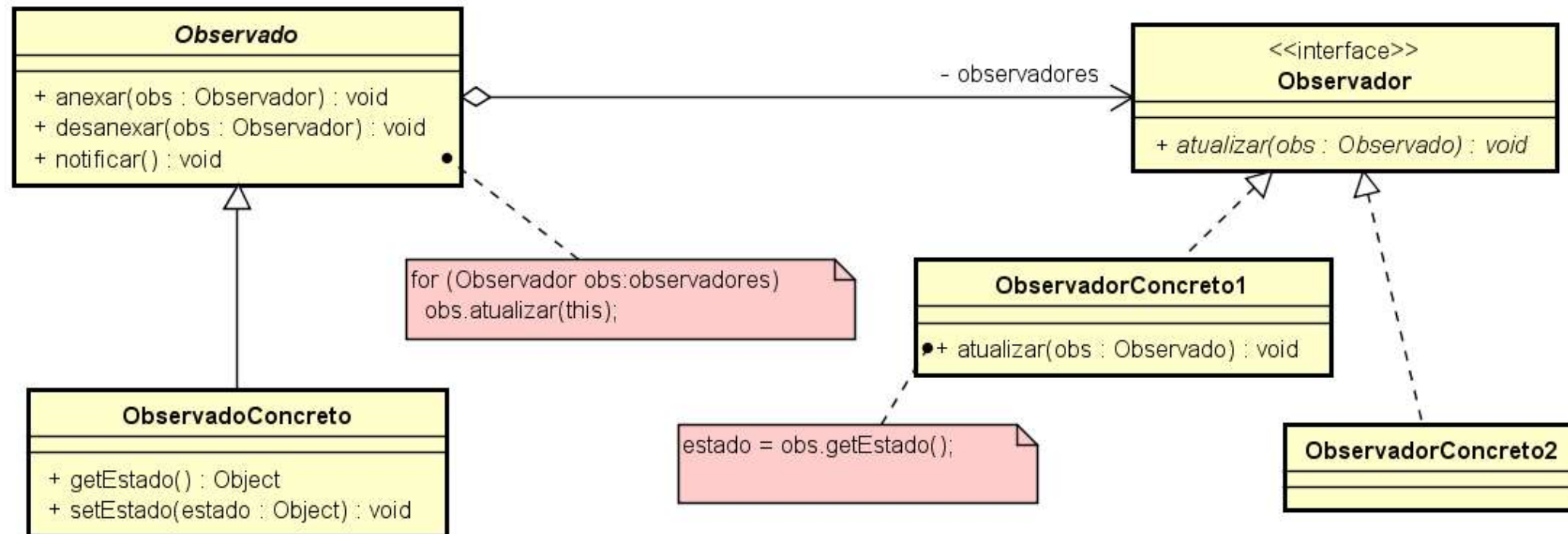


Observer (2)

- **Observer:** definir uma dependência um-para-muitos entre objetos, de maneira que quando um **objeto muda de estado** todos os seus **dependentes são notificados e atualizados automaticamente**.
- Os dependentes podem pertencer a diferentes tipos.

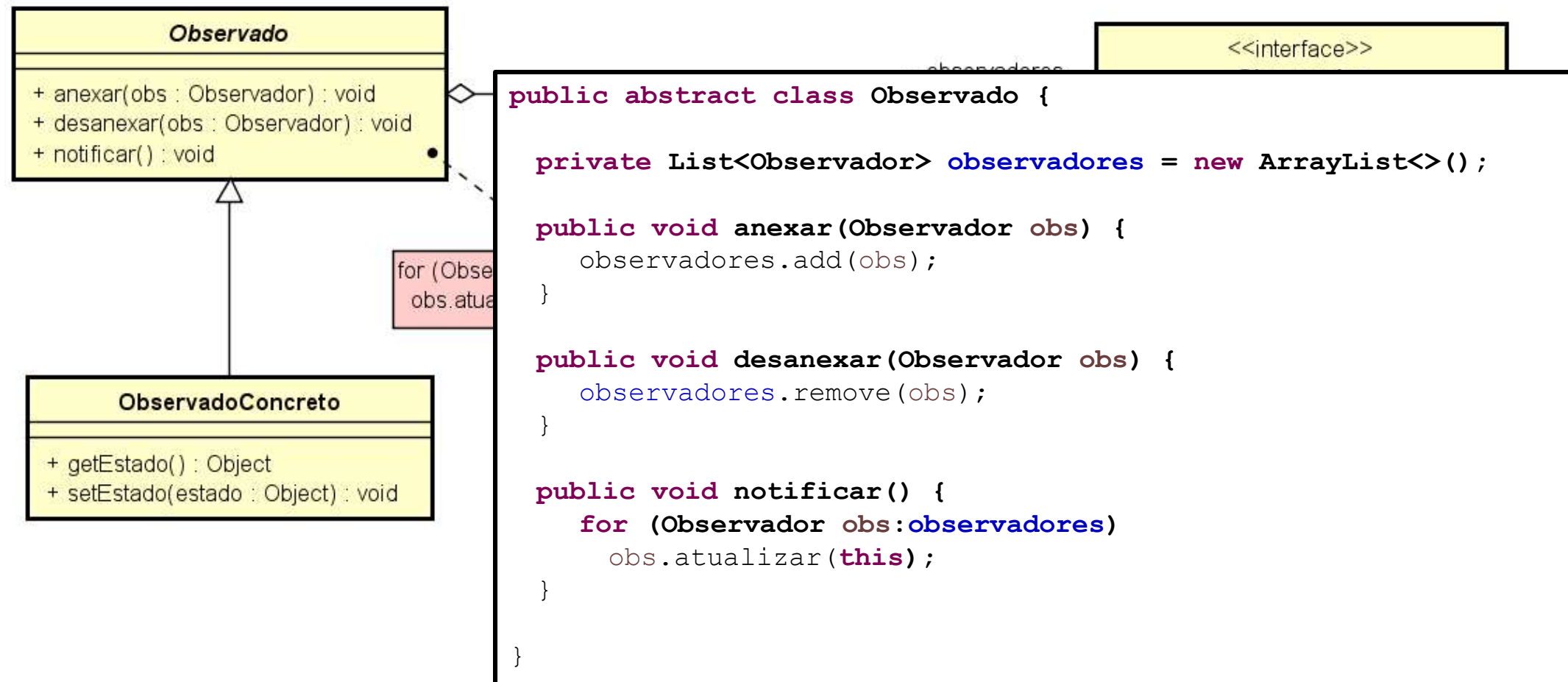


Observer (3)

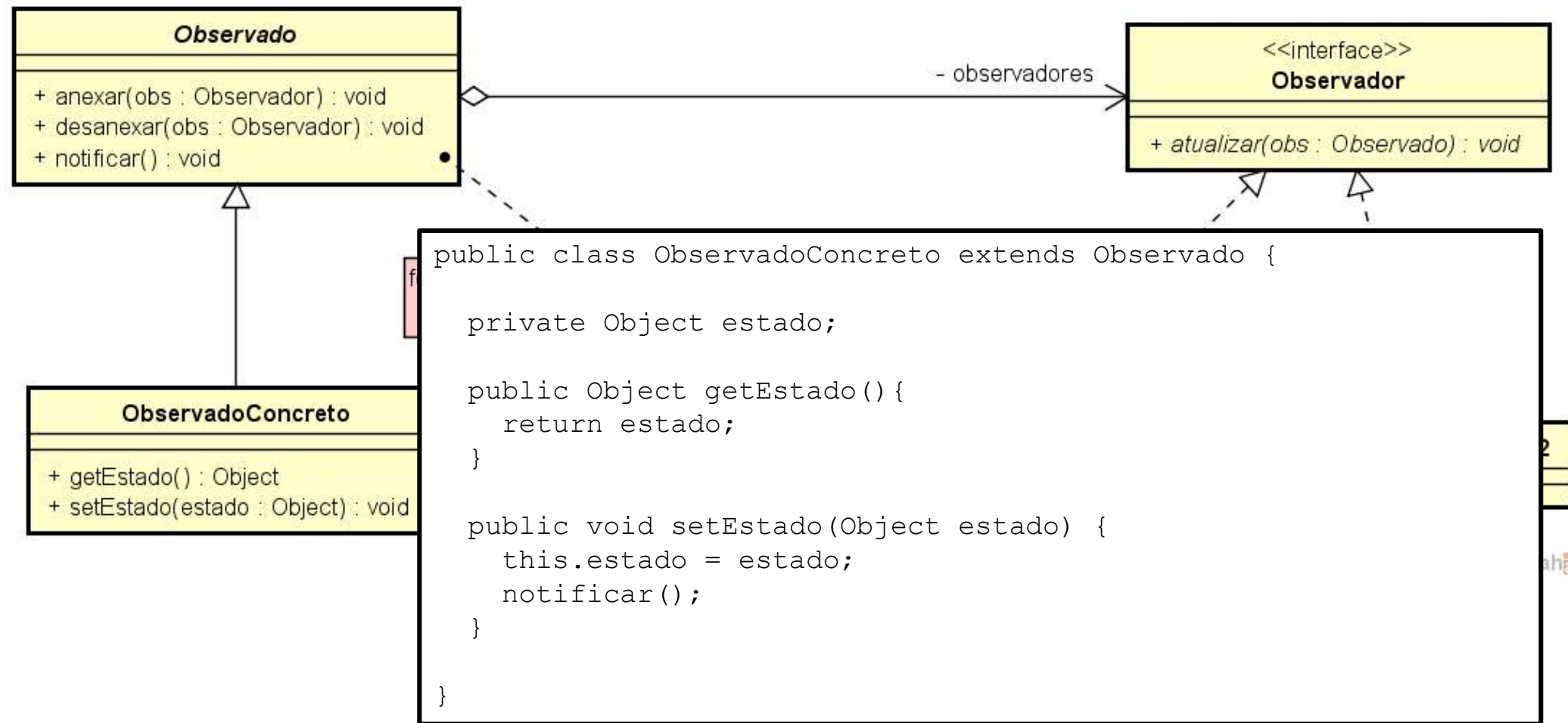


powered by Astah

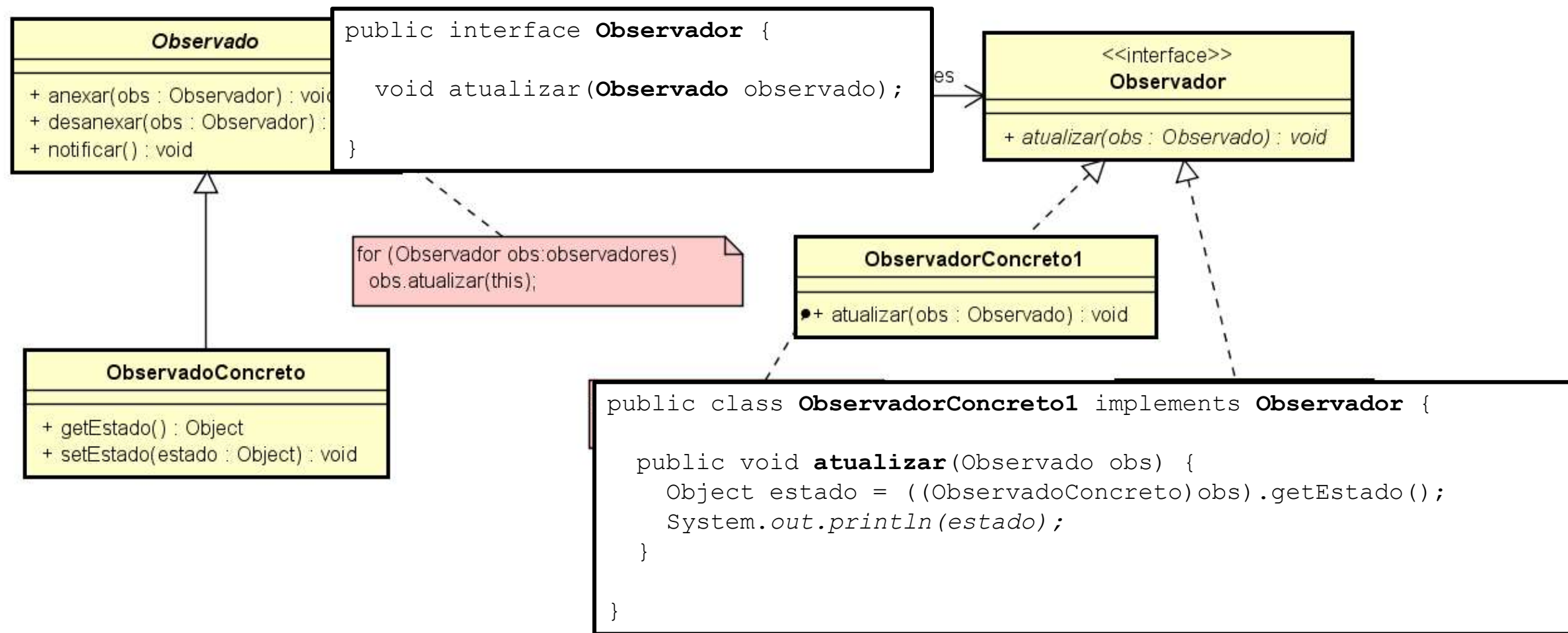
Observer (4)



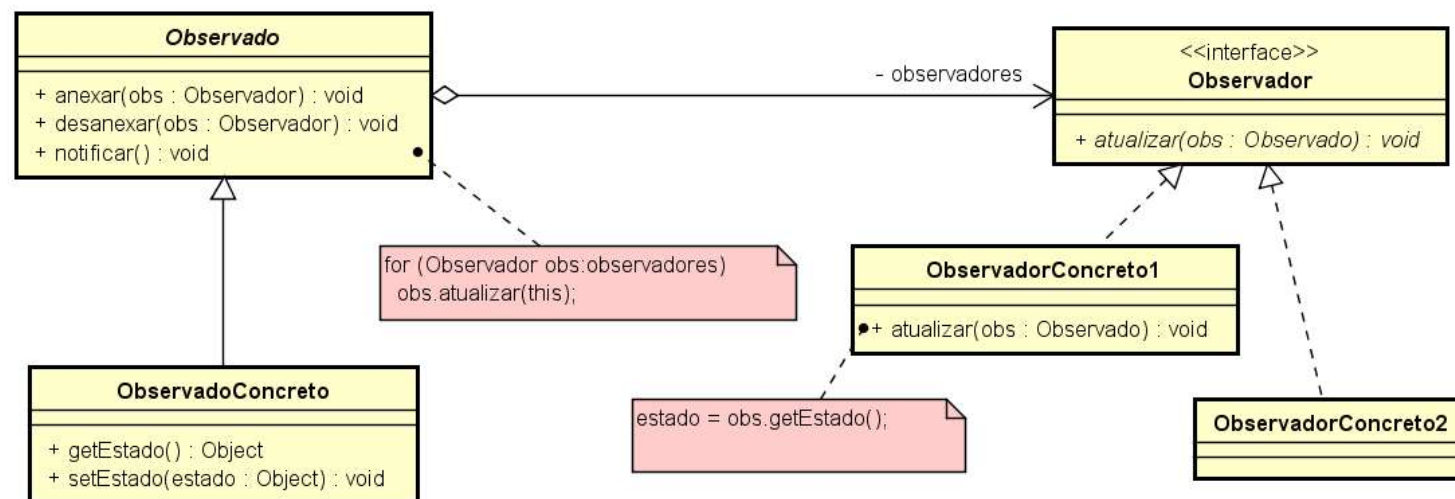
Observer (5)



Observer (6)



Observer (7)



powered by Astah

o:Observado

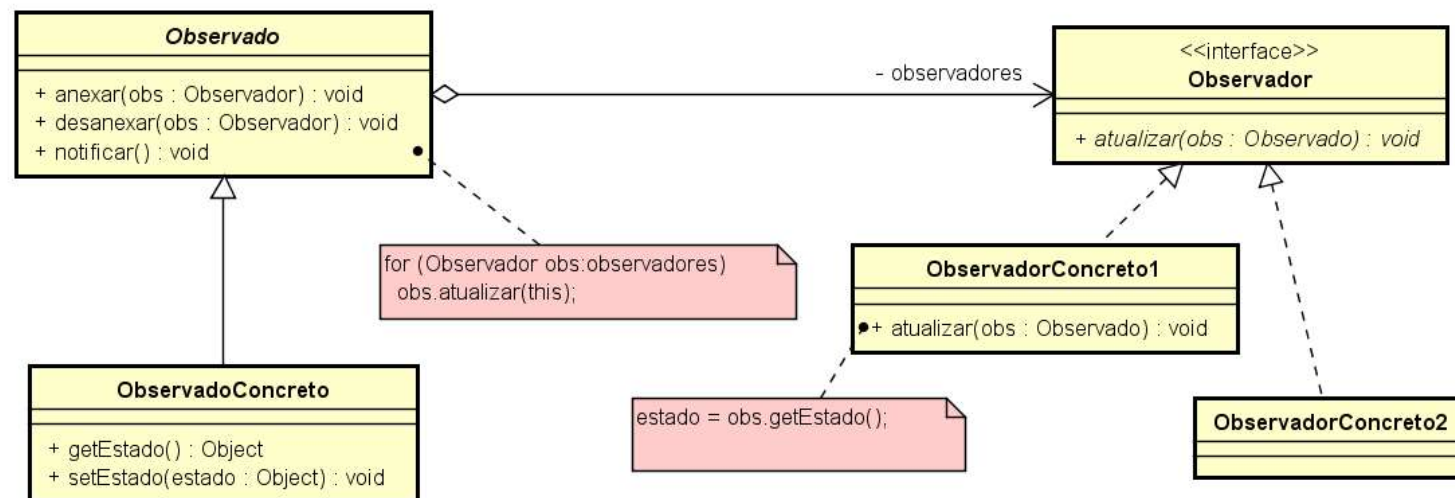
estado: "XX"

o1:Observador

o2:Observador

Observer (8)

o.anexar(o1)

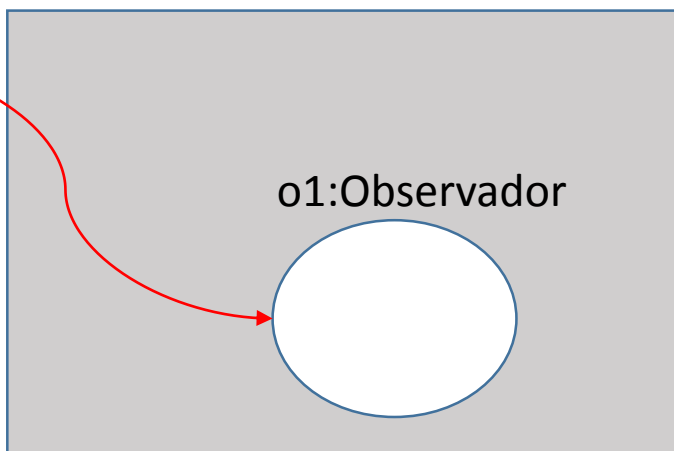


powered by Astah

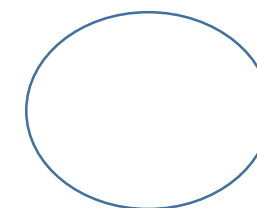
o:Observado

observadores

estado: "XX"

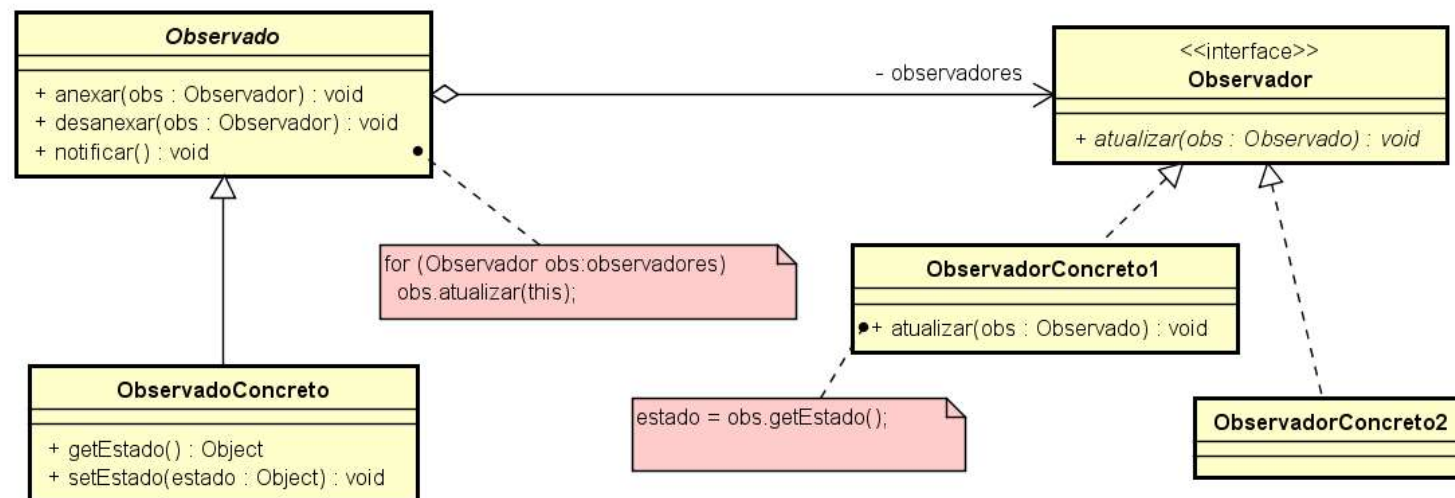


o2:Observador

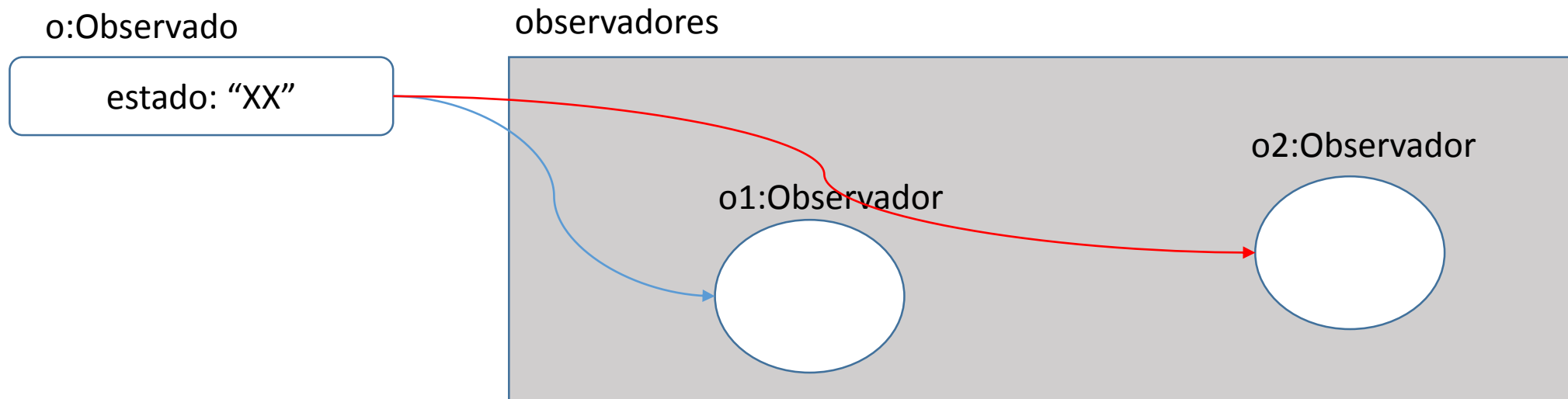


Observer (8)

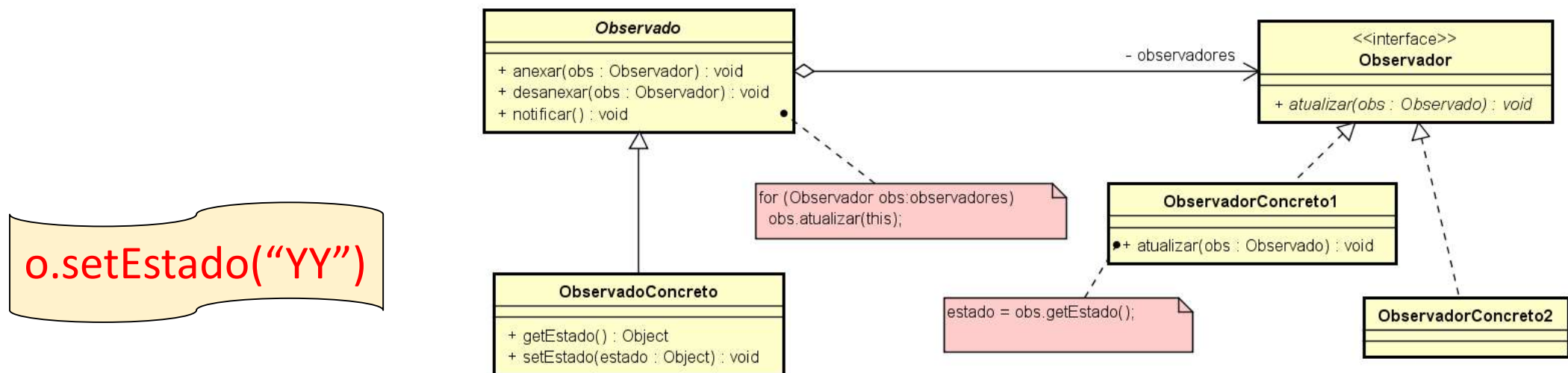
o.anexar(o2)



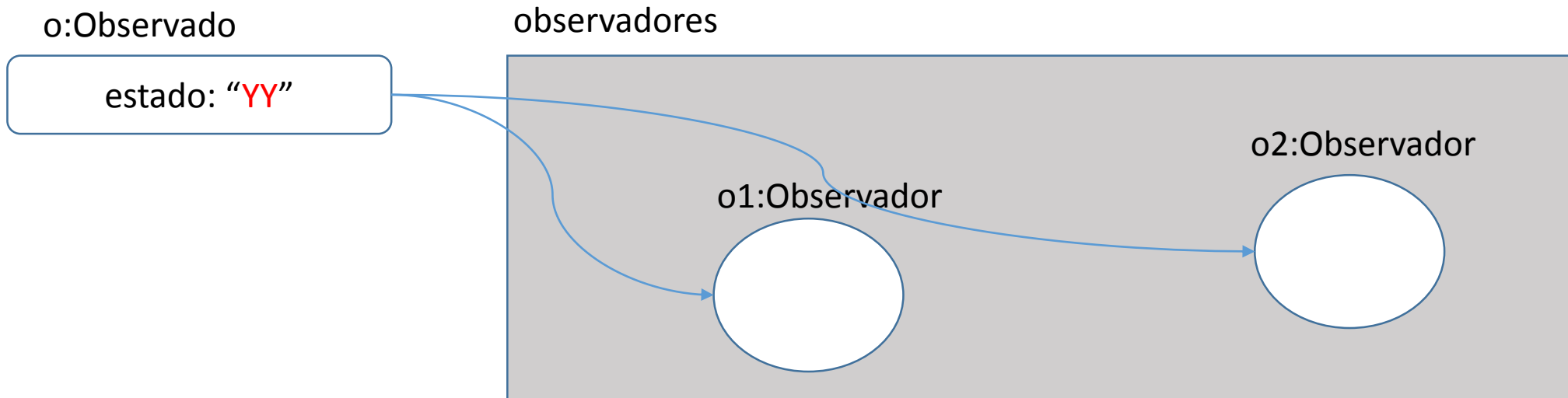
powered by Astah



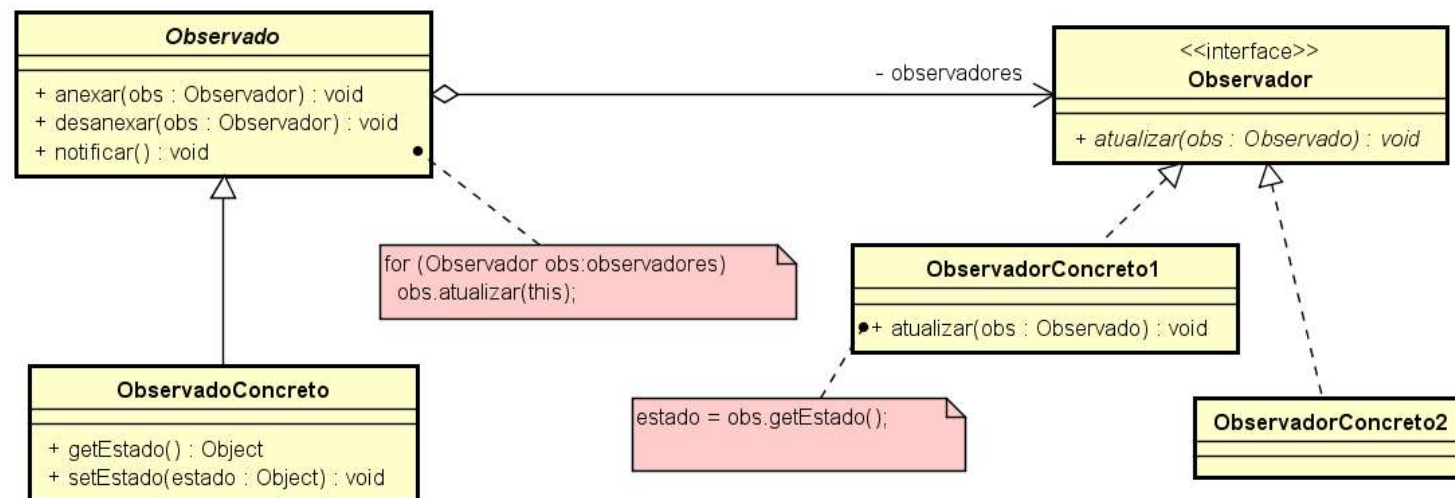
Observer (9)



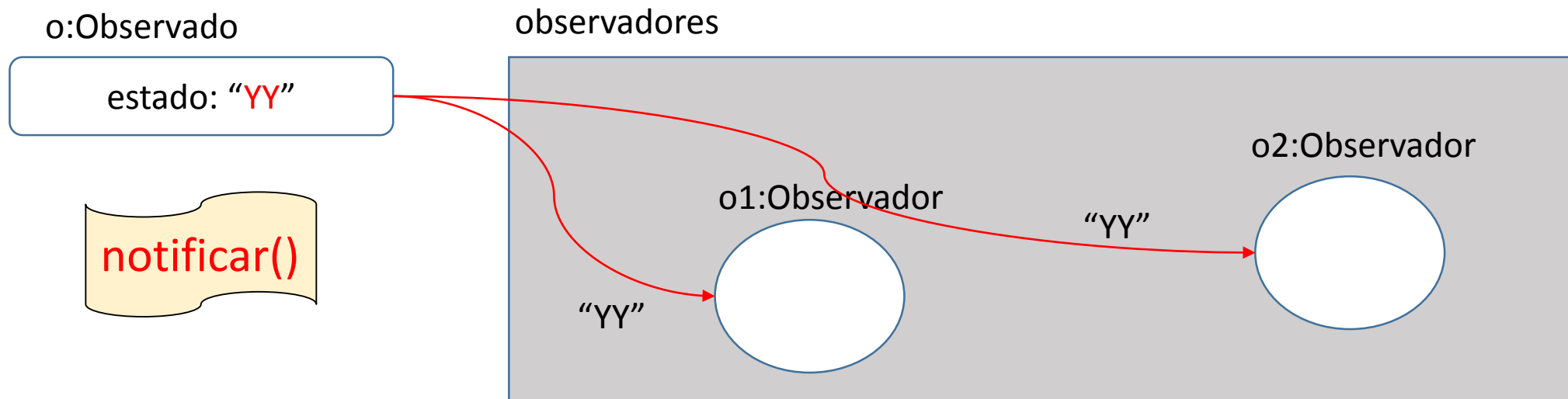
powered by Astah



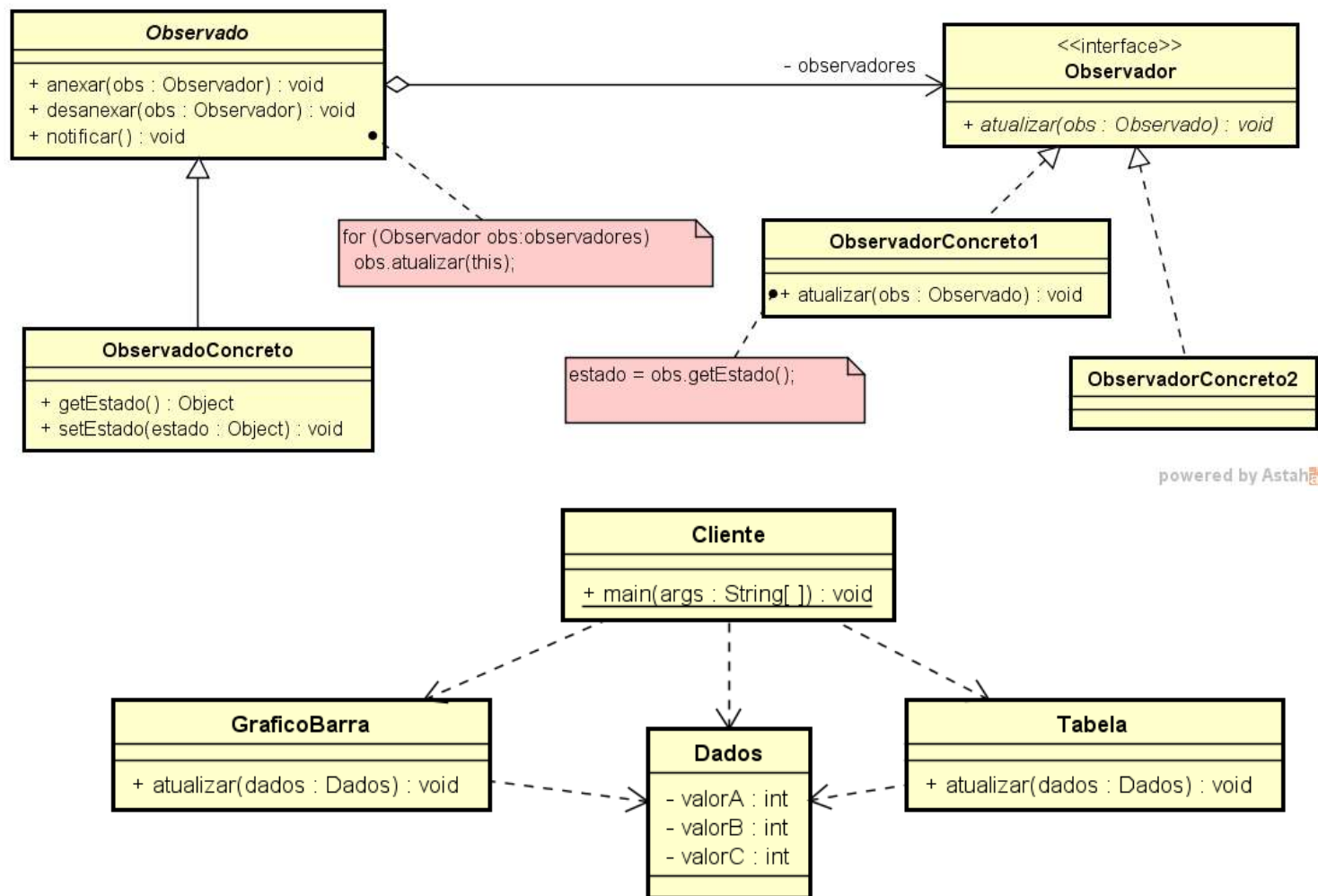
Observer (9)



powered by Astah



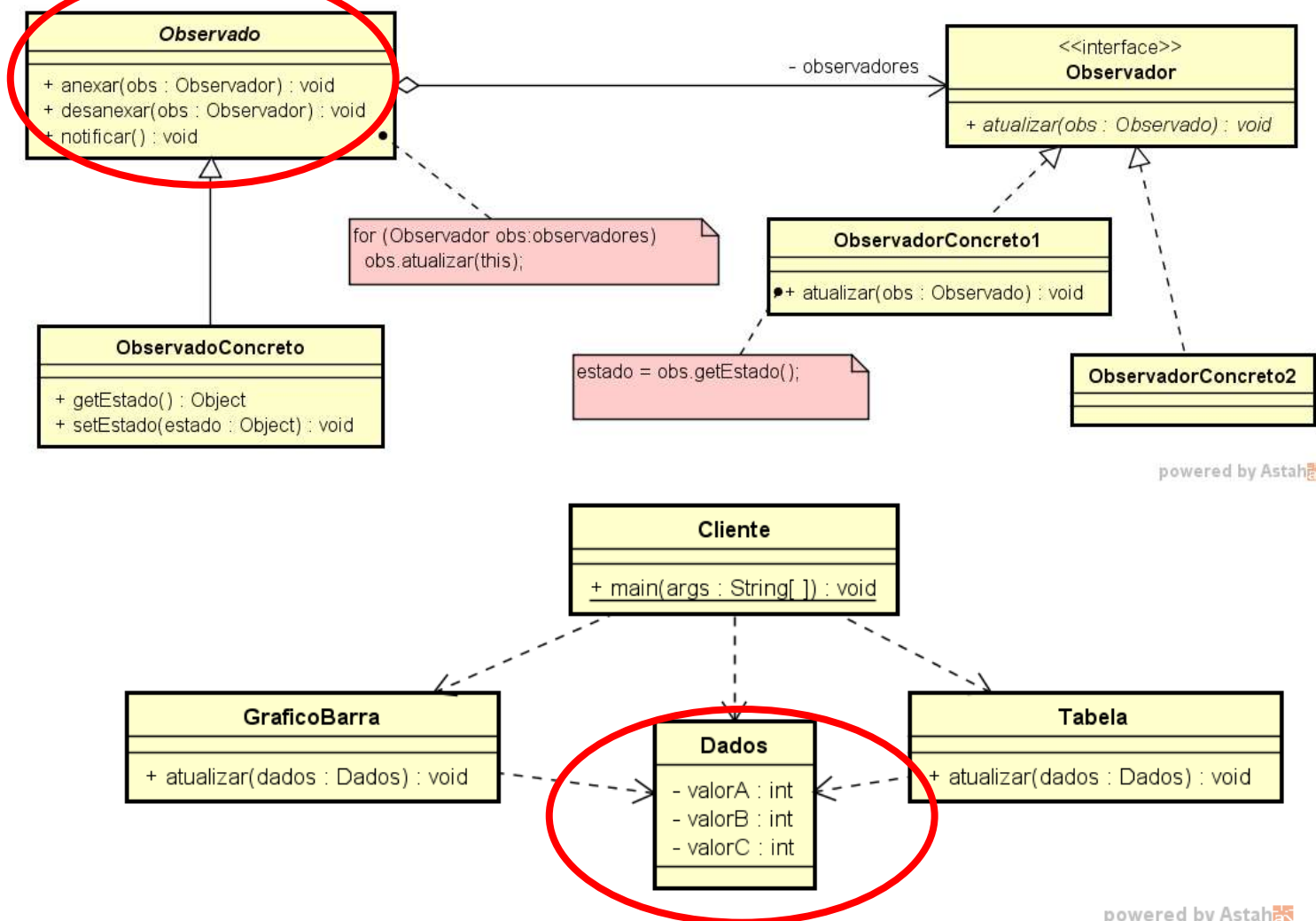
Solução (1)



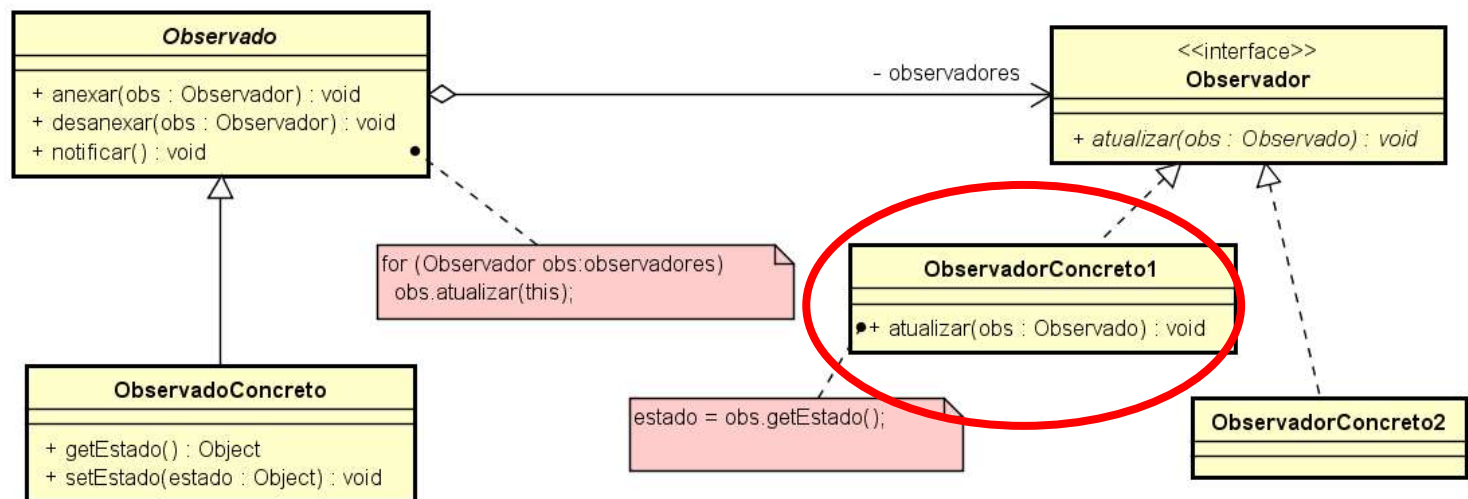
powered by Astah

powered by Astah

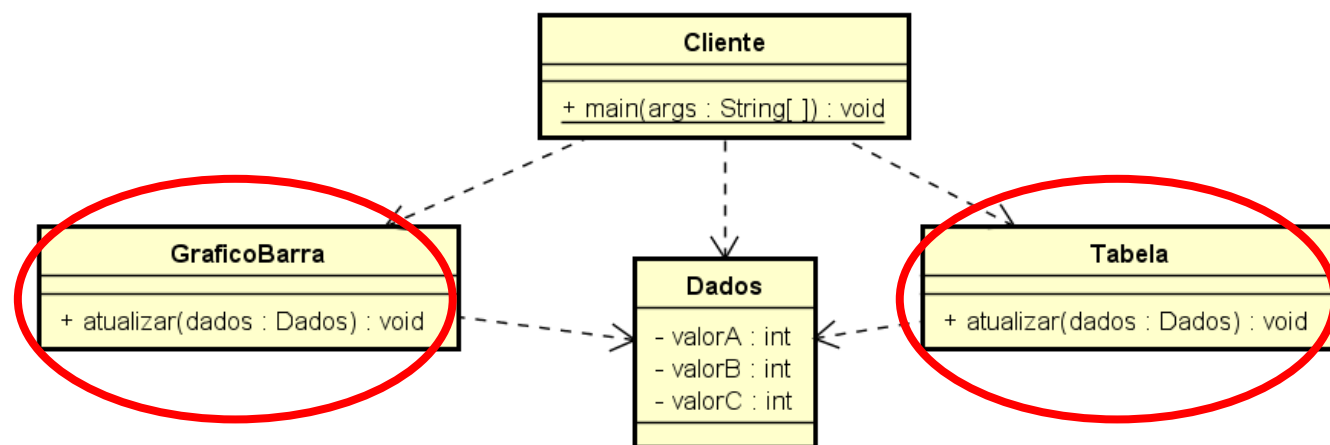
Solução (2)



Solução (3)

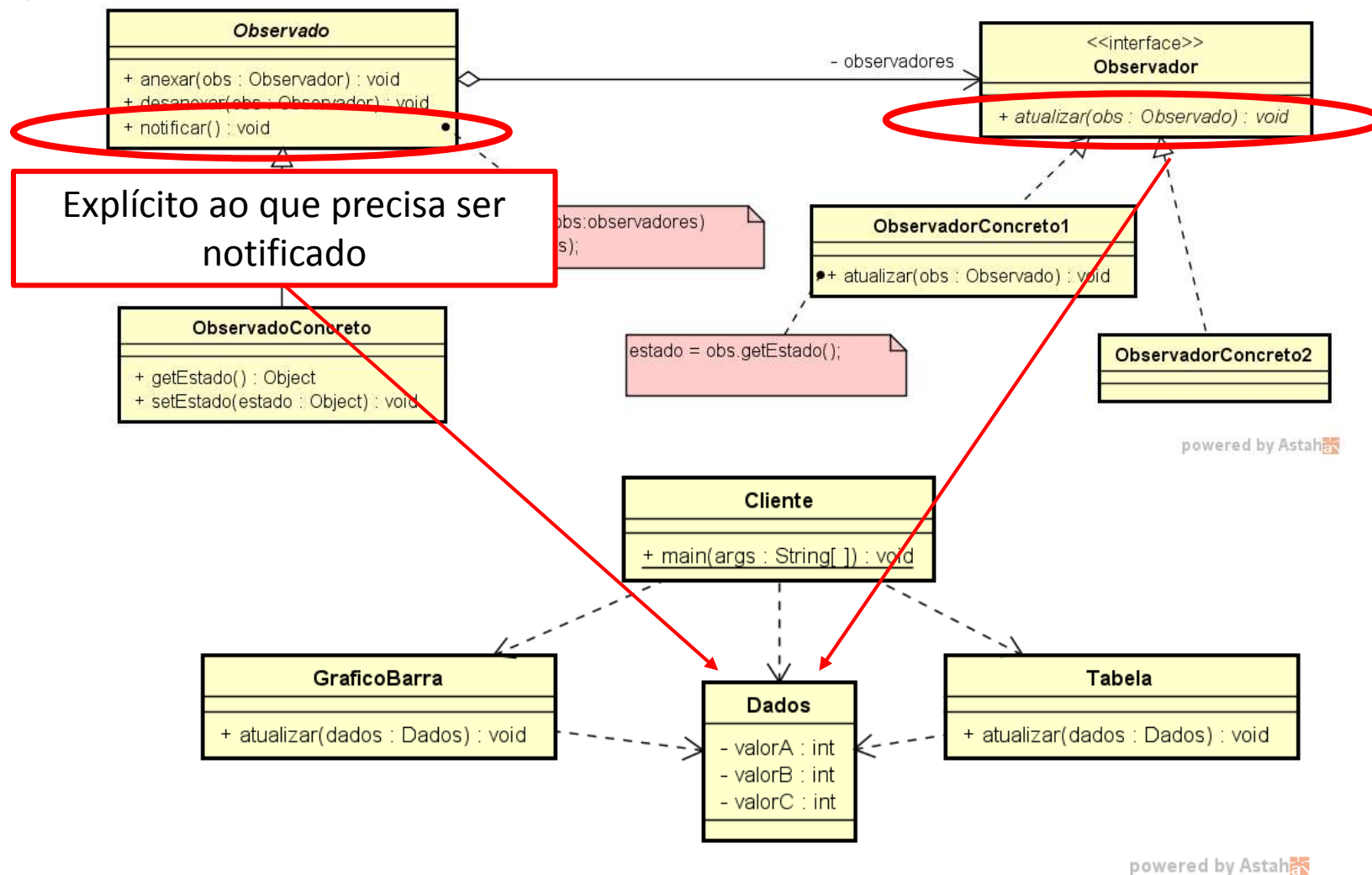


powered by Astah



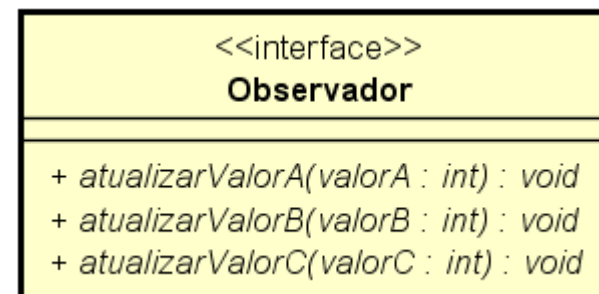
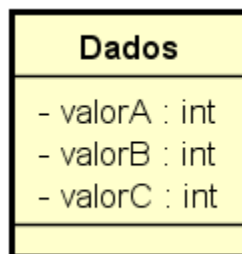
powered by Astah

Solução (4)



Solução (5)

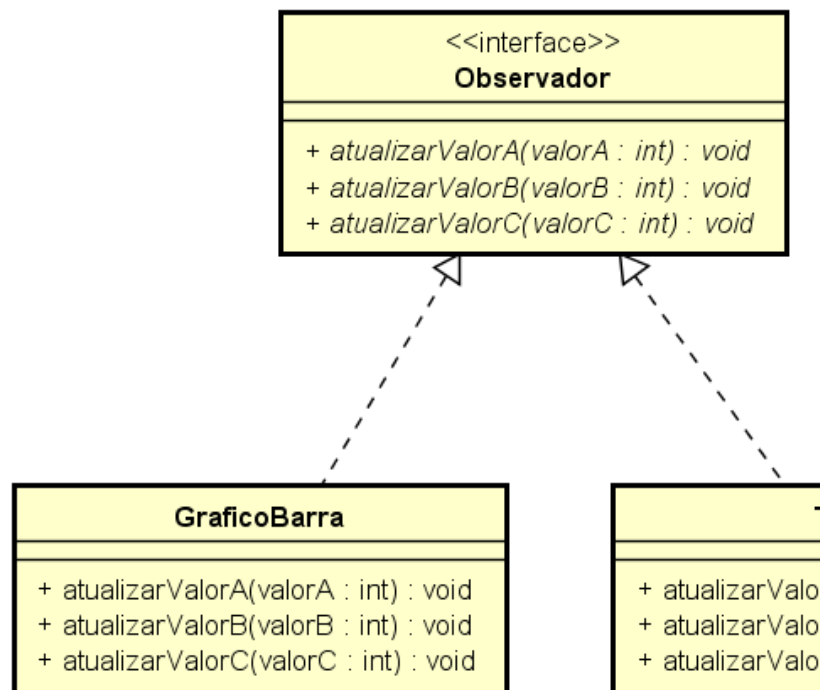
1. Identifique o que precisa ser notificado do objeto observado. Crie uma interface com métodos suficientes para a notificação de cada mudança a ser observada



```
public interface Observador {  
  
    void atualizarValorA(int valorA);  
    void atualizarValorB(int valorB);  
    void atualizarValorC(int valorC);  
  
}
```

Solução (6)

2. Aplique a interface observadora em cada classe receptora.



```

public class GraficoBarra implements Observador {

    private String barraA, barraB, barraC;

    private void desenhar() {
        System.out.println("Barras:\n Valor A: " + barraA + "\nValor B: "
            + barraB + "\nValor C: " + barraC);
    }

    @Override
    public void atualizarValorA(int valorA) {
        barraA = "";
        for (int i = 0; i < valorA; i++) {
            barraA += '=';
        }
        desenhar();
    }
    ...
}
    
```

Solução (7)

3. Refatore a classe observada para suportar as listas de observadores.

```
public class Dados {

    private int valorA, valorB, valorC;

    public int getValorC() { return valorC; }

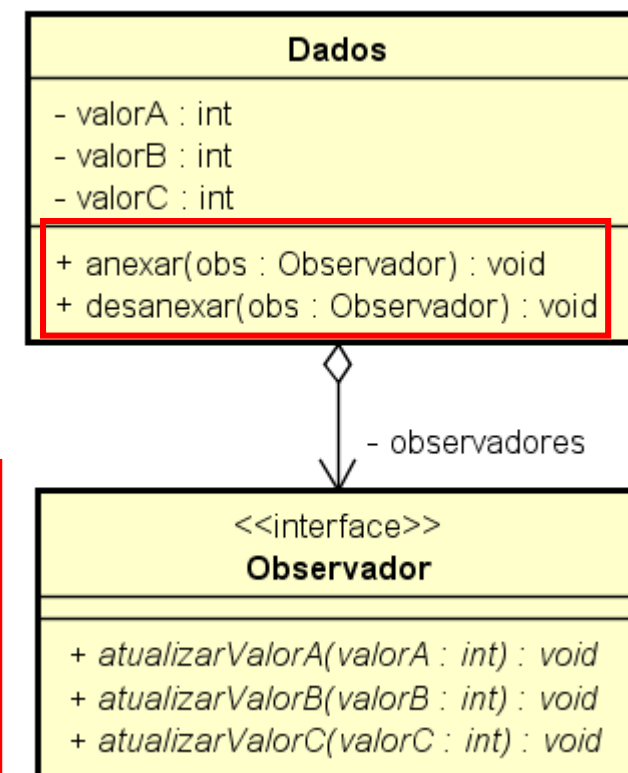
    public void setValorC(int valorC) {
        this.valorC = valorC;
        for (Observador obs:observadores)
            obs.atualizarValorC(valorC);
    }

    private List<Observador> observadores = new ArrayList<>();

    public void anexar(Observador obs) { this.observadores.add(obs); }

    public void desanexar(Observador obs) {
        this.observadores.remove(obs);
    }

    ...
}
```



Solução (8)

```
public static void main(String[] args) {  
    Dados dados = new Dados(7, 3, 1);  
    dados.anexar(new Tabela(dados));  
    dados.anexar(new GraficoBarra(dados));  
  
    dados.setValorA(10);  
    dados.setValorB(20);  
    dados.setValorC(15);  
}
```

Exercícios

- observador1 – completo
- observador2 – falta aplicar o padrão
- observador3 – fazer (classes de domínio no diagrama)
- observador4 – fazer (GUI pronta)

Referências

- FREEMAN, E.; FREEMAN, E. **Use a cabeça**: padrões de projetos. Rio de Janeiro: Alta Books, 2005. Cap 2.
- GAMMA, E. et al. **Padrões de projeto**: soluções reutilizáveis de software orientado a objetos. Porto Alegre: Bookman, 2000. pp 274-284.
- KERIEVSKY, J. **Refatoração para padrões**. Porto Alegre: Bookman, 2008. pp 268-278.
- SHALLOWAY, A.; TROTT, J. **Explicando padrões de projeto**: uma nova perspectiva em projeto orientado a objeto. Porto Alegre: Bookman, 2004. Cap 17.