Arquitetura de ‘Software’ {

[Padrões de Projeto]

< Builder and State >

}

TADS.html

seminario.css

Integrantes da ‘Equipe’;

Arthur Pinheiro de Barros

Davi Dantas Soares

João Pedro Silva de Queiroz

Pedro Henrique da Costa **Barros**

Stefaneo Ribeiro

Tabela de ‘Conteúdos’ {

}

01.1 {

[Visão Geral do BUILDER]

< O que é o padrão Builder? >

}

Conceito < /1 > {

< Permite a construção de objetos complexos passo a passo. >

}

Concepts < /2 > {

< Diferente de outros padrões, não exige que os produtos tenham uma interface comum. Isso torna possível produzir produtos diferentes usando o mesmo processo de construção>

}

01.2 {

[Visão Geral do STATE]

< O que é o padrão State? >

}

Conceito < /1 > {

<É um padrão comportamental que permite que um objeto altere o seu comportamento quando seu estado interno muda. No fim, parece como se o objeto mudasse de classe >

}

Conceito < /2 > {

<Muda o comportamento do contexto ao delegar algum trabalho para objetos auxiliares e não restringe dependências entre estados concretos, permitindo que eles alterem o estado do contexto à vontade>

}

02 {

[Qual problema resolve?]

< Como os padrões de projeto resolvem problemas? >

}

Builder; {

‘O Problema:’

 Imagine um objeto complexo que necessite de uma inicialização passo a passo trabalhosa, de muitos campos e objetos agrupados. Tal código de inicialização fica geralmente enterrado dentro de um construtor monstruoso com vários parâmetros. Ou pior: Espalhado por todo o código cliente

}

Builder; {

‘A solução:’

 O padrão Builder sugere que você extraia o código de construção do objeto para fora de sua própria classe e mova ele para os objetos separados chamados de builders. A construção de objetos é separada em etapas. Assim, na hora de criar um objeto, você chama apenas as etapas necessárias para a configuração específica daquele objeto.

}

```
State; {

    'O Problema:'

        <p> Máquinas de estado geralmente tem muitos
        if ou switch, que selecionam o comportamento
        apropriado dependendo do estado atual do
        objeto. A fraqueza em máquinas baseadas em
        condicionais está quando se adiciona mais e
        mais estados, pois a maioria dos métodos
        passará a conter condicionais monstruosas.

        </p>
}
```

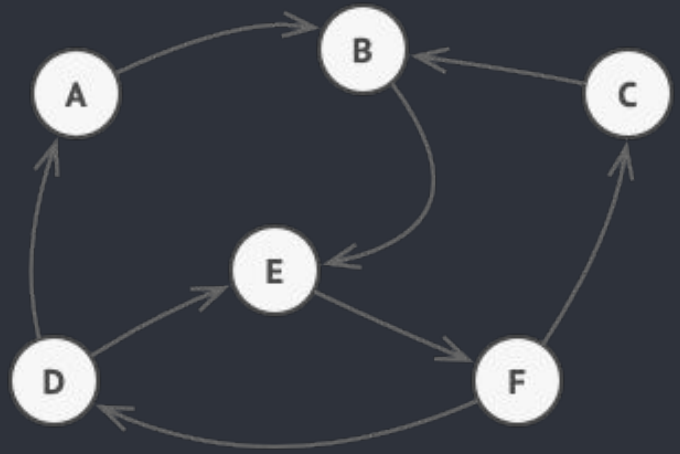
TADS.html

seminario.css

Máquina de estado finito; {

‘O Problema:’

<figure>



</figure>

}

State; {

‘A solução:’

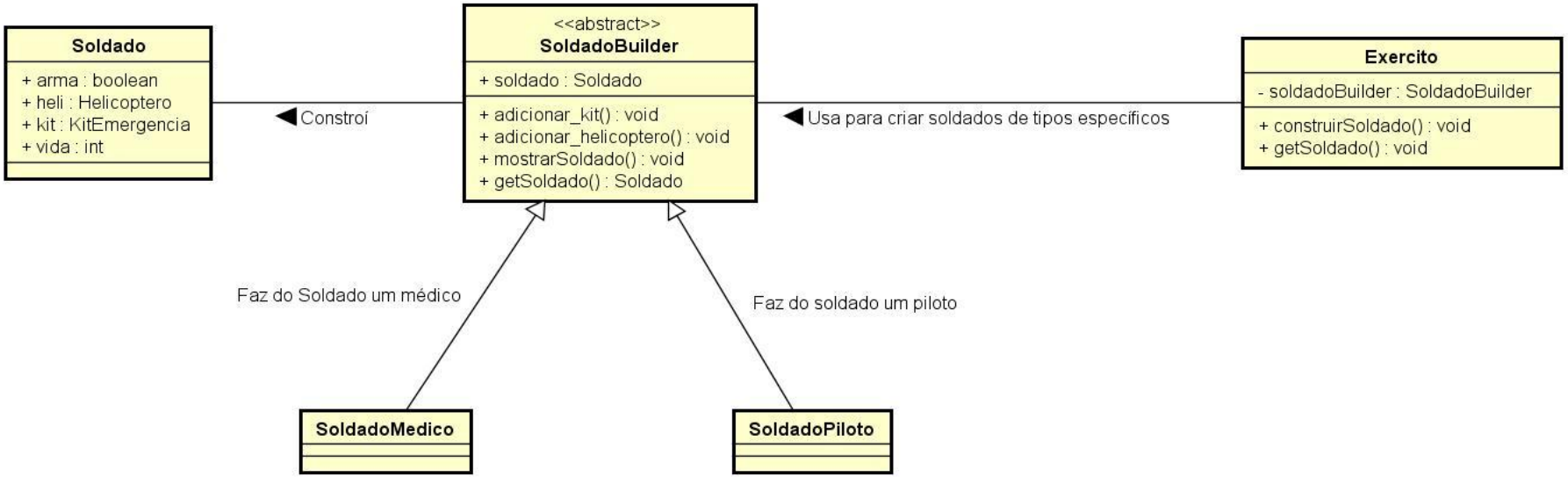
 O State sugere que você crie novas classes para todos os estados possíveis de um objeto e extraia todos os comportamentos específicos de estados para dentro dessas classes.

}

pkg

Diagrama de classes{

Builder

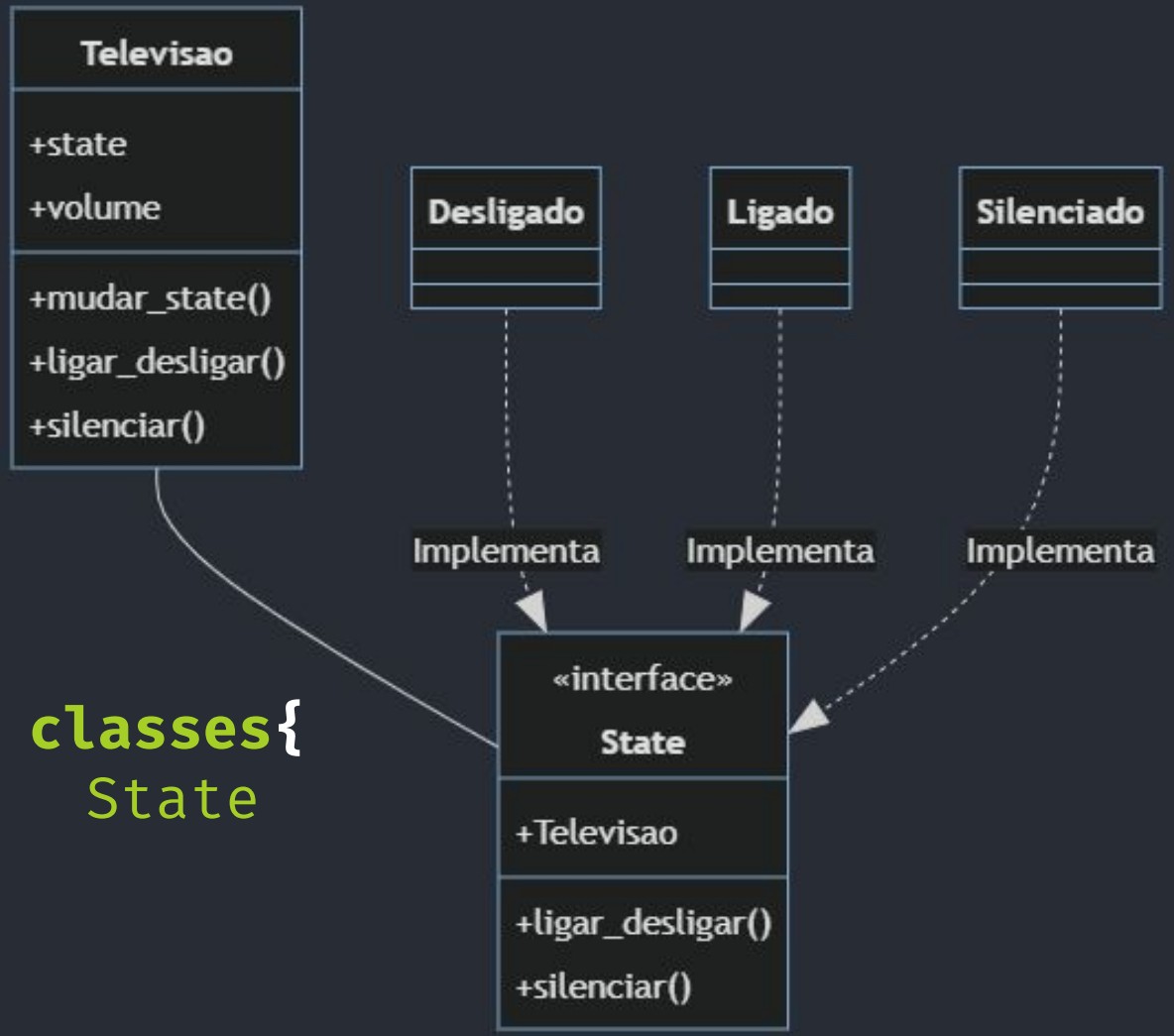


}

Diagrama de classes{

State

}



Por ‘Fim’ {

A implementação dos Padrões

Repositório: https://github.com/JoPedro/seminario-2-builder-e-state

}