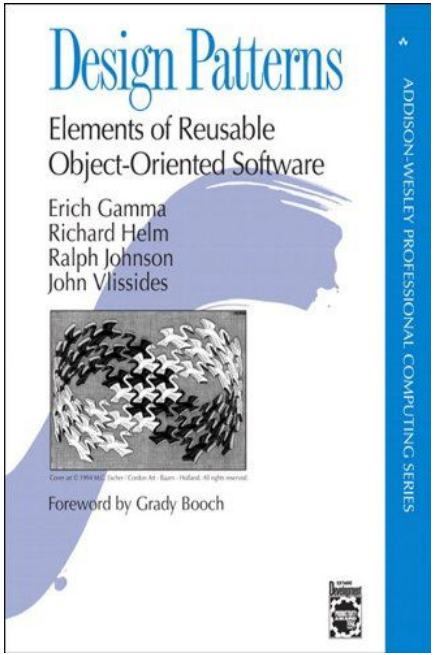
Padrões de Projeto

Profa. Fabíola S. F. Pereira

fabiola.pereira@ufu.br

Padrões de Projeto

- Soluções recorrentes para problemas de projeto enfrentados por engenheiros de software



1994, conhecido como livro da "Gangue dos Quatro" ou GoF

Criacionais	Estruturais	Comportamentais
Abstract Factory	**Proxy**	**Strategy**
Factory Method	**Adapter**	**Observer**
Singleton	**Facade**	**Template Method**
Builder	**Decorator**	Visitor
Prototype	Bridge	Chain of Responsibility
	Composite	Command
	Flyweight	Interpreter
		Iterator
		Mediator
		Memento
		State

23 padrões

(4) Adaptador

Contexto: Sistema para Controlar Projetores Multimídia

```
class ProjetorSamsung {
  public void turnOn() { ... }
  ...
}

class ProjetorLG {
  public void enable(int timer) { ... }
  ...
}
```

Classes fornecidas pelos fabricantes dos projetores

Problema

- No sistema de controle de multimídias, eu gostaria de manipular uma única interface Projetor

```
interface Projetor {

  void liga();

}
...
class SistemaControleProjetores {

  void init(Projetor projetor) {
    projetor.liga();  // liga qualquer projetor
  }

}
```

Sempre vou usar objetos dessa interface, sem me preocupar com a classe concreta que implementa a interface

Problema (cont.)

```
class ProjetorSamsung {
  public void turnOn() { ... }
  ...
}

class ProjetorLG {
  public void enable(int timer) { ... }
  ...
}
```

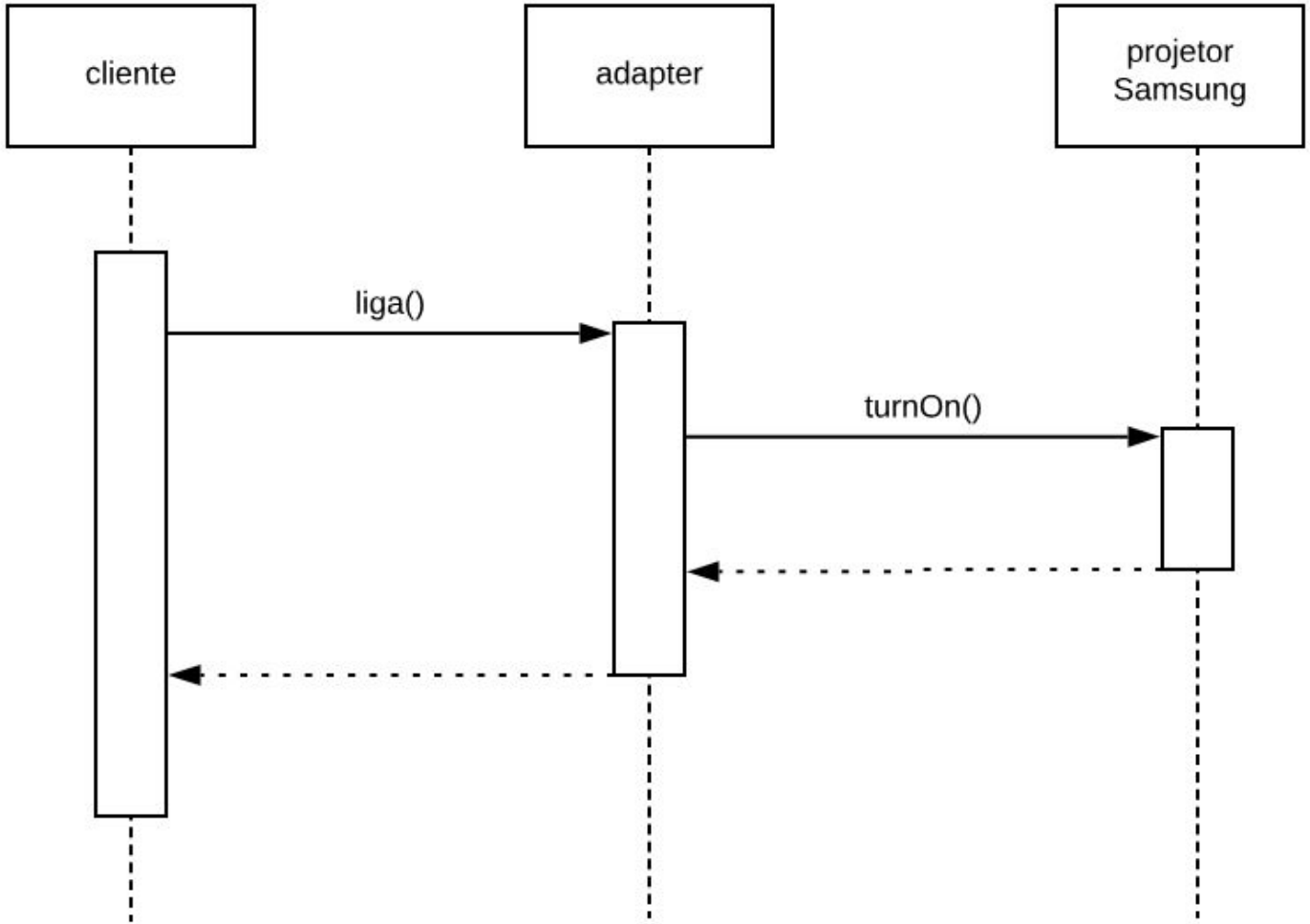
- São classes dos fabricantes dos projetores
- Eu não tenho acesso a elas. Para, por exemplo, fazer com que elas implementem a interface Projetor

Solução: **Padrão Adaptador ou Wrapper**

Padrão (ou interface)
de entrada

Padrão (ou interface)
de saída

Classe Adaptadora


cliente
adapter
projetor
Samsung
liga()
turnOn()

Solução: Implementar uma Classe Adaptadora

```
class AdaptadorProjetorSamsung implements Projetor {

    private ProjetorSamsung projetor;

    AdaptadorProjetorSamsung (ProjetorSamung projetor) {
      this.projetor = projetor;
    }

    public void liga() {
      projetor.turnOn();
    }

}
```

Solução: Implementar uma Classe Adaptadora

```
class AdaptadorProjetorSamsung implements Projetor {

    private ProjetorSamsung projetor;

    AdaptadorProjetorSamsung (ProjetorSamung projetor) {
      this.projetor = projetor;
    }

    public void liga() {
      projetor.turnOn();
    }

}
```

Padrão de Entrada

Padrão de Saída

Projetor

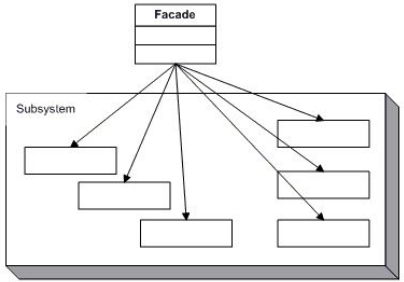
ProjetorSamsung

AdaptadorProjetorSamsung

(5) Fachada

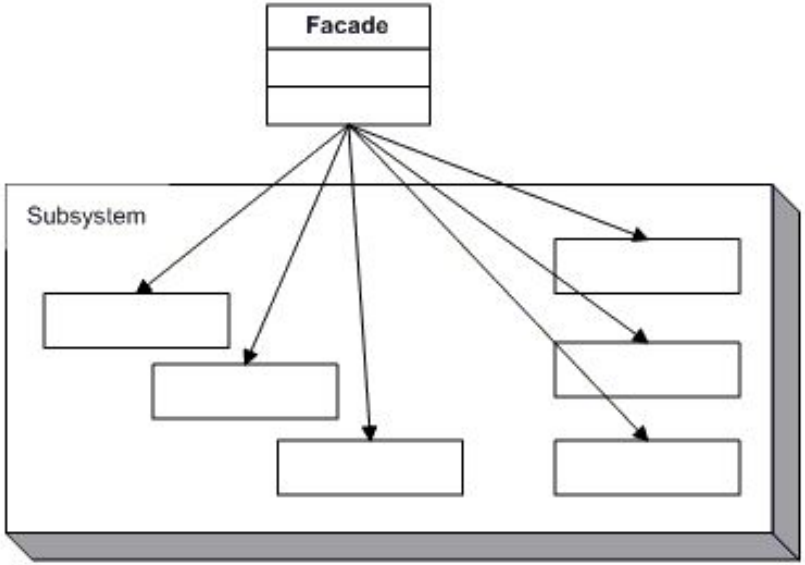
Contexto / Problema / Solução

- Contexto: módulo M usado por diversos outros módulos
- Problema: interface de M é complexa; clientes reclamam que é difícil usar a interface pública de M
- Solução: criar uma interface mais simples para M, chamada de **Fachada**.



Solução

- Uma Fachada é uma classe que oferece uma interface mais simples para um sistema

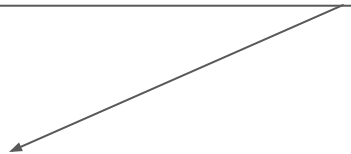


Exemplo: Interpretador

```
Scanner s = new Scanner("prog1.x");
Parser p = new Parser(s);
AST ast = p.parse();
CodeGenerator code = new CodeGenerator(ast);
code.eval();
```

```
Scanner s = new Scanner("prog1.x");
Parser p = new Parser(s);
AST ast = p.parse();
CodeGenerator code = new CodeGenerator(ast);
code.eval();
```

```
new InterpretadorX("prog1.x").eval();
```



Interface mais simples,
que a de cima

Exercícios de Fixação

Assinale V ou F

( ) Wrapper é um padrão que facilita a construção de objetos complexos com vários atributos, sendo alguns deles opcionais.

( ) Adaptador é um padrão que permite adicionar dinamicamente novas funcionalidades a uma classe.

( ) Uma desvantagem de Fachadas é aumentar o acoplamento entre um subsistema e seus clientes.

Assinale V ou F

( ) Um adaptador oferece uma interface diferente para o objeto adaptado. Em contraste, um proxy possui a mesma interface que o seu objeto base.

( ) Como são padrões distintos, uma Fachada não pode ser implementada como um Singleton.

Créditos

- CC-BY: Slides adaptados de Marco Tulio Valente, ESM

Fim

Assinale V ou F

(F) Wrapper é um padrão que facilita a construção de objetos complexos com vários atributos, sendo alguns deles opcionais.

(F) Adaptador é um padrão que permite adicionar dinamicamente novas funcionalidades a uma classe.

(V) Uma desvantagem de Fachadas é aumentar o acoplamento entre um subsistema e seus clientes.

Assinale V ou F

(V) Um adaptador oferece uma interface diferente para o objeto adaptado. Em contraste, um proxy possui a mesma interface que o seu objeto base.

(F) Como são padrões distintos, uma Fachada não pode ser implementada como um Singleton.