

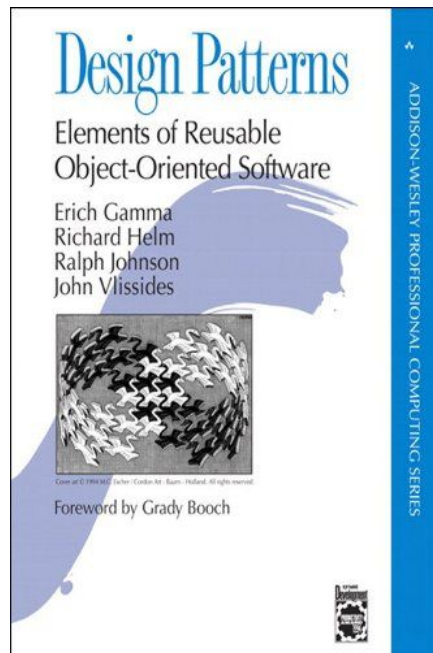
Padrões de Projeto

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

23 padrões

(8) Observador

Contexto: Sistema de uma Estação Meteorológica

- Duas classes principais:
 - Temperatura
 - Termômetro
- Diversos termômetros: digital, analógico, web, celular, etc
- Se a temperatura mudar, os termômetros devem ser atualizados

Problema

- Não queremos acoplar Temperatura a Termômetros
- Motivos:
 - Tornar classe de dados (modelo) independente de classes de visão (ou de interface com usuários)
 - Tornar flexível a adição de um novo tipo de termômetro no sistema

Solução: **Padrão Observador**

- Implementa uma relação do tipo um-para-muitos entre os seguintes objetos
 - Sujeito (Temperatura)
 - Observadores (Termômetros)
- Quando o estado de um Sujeito muda, seus Observadores são notificados.
- Mas Sujeito não conhece o tipo concreto de seus Observadores

Programa Principal

```
void main() {  
    Temperatura t = new Temperatura();  
    t.addObserver(new TermometroCelsius());  
    t.addObserver(new TermometroFahrenheit());  
    t.setTemp(100.0);  
}
```



Sujeito

Programa Principal

```
void main() {  
    Temperatura t = new Temperatura();  
    t.addObserver(new TermometroCelsius());  
    t.addObserver(new TermometroFahrenheit());  
    t.setTemp(100.0);  
}
```



Dois observadores

Programa Principal

```
void main() {  
    Temperatura t = new Temperatura();  
    t.addObserver(new TermometroCelsius());  
    t.addObserver(new TermometroFahrenheit());  
    t.setTemp(100.0);  
}
```



Notifica os termômetros

Classe Temperatura

```
class Temperatura extends Subject {  
    private double temp;  
  
    public double getTemp() {  
        return temp;  
    }  
  
    public void setTemp(double temp) {  
        this.temp = temp;  
        notifyObservers();  
    }  
}
```

Classe Temperatura

```
class Temperatura extends Subject {  
  
    private double temp;  
  
    public double getTemp() {  
        return temp;  
    }  
  
    public void setTemp(double temp) {  
        this.temp = temp;  
        notifyObservers();  
    }  
  
}
```

Classe que
implementa
addObservers e
notifyObservers

Classe Temperatura

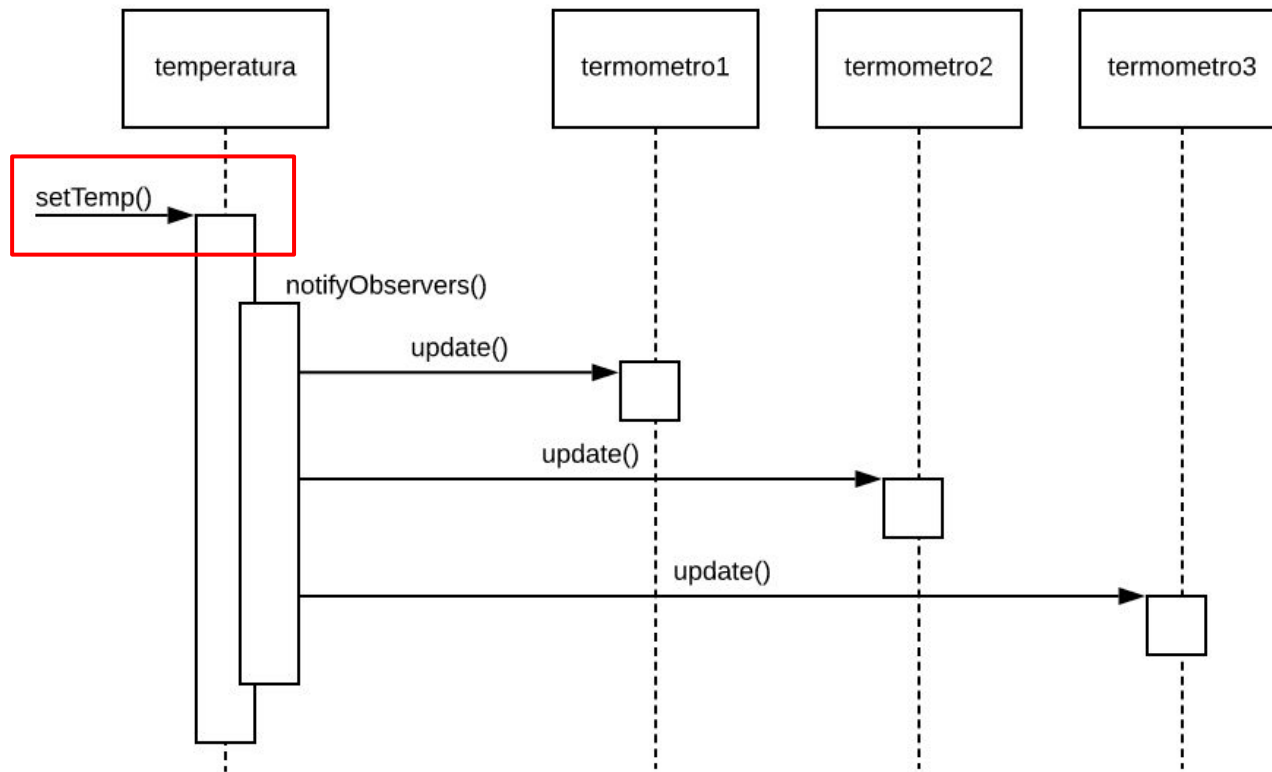
```
class Temperatura extends Subject {  
  
    private double temp;  
  
    public double getTemp() {  
        return temp;  
    }  
  
    public void setTemp(double temp) {  
        this.temp = temp;  
        notifyObservers();  
    }  
  
}
```

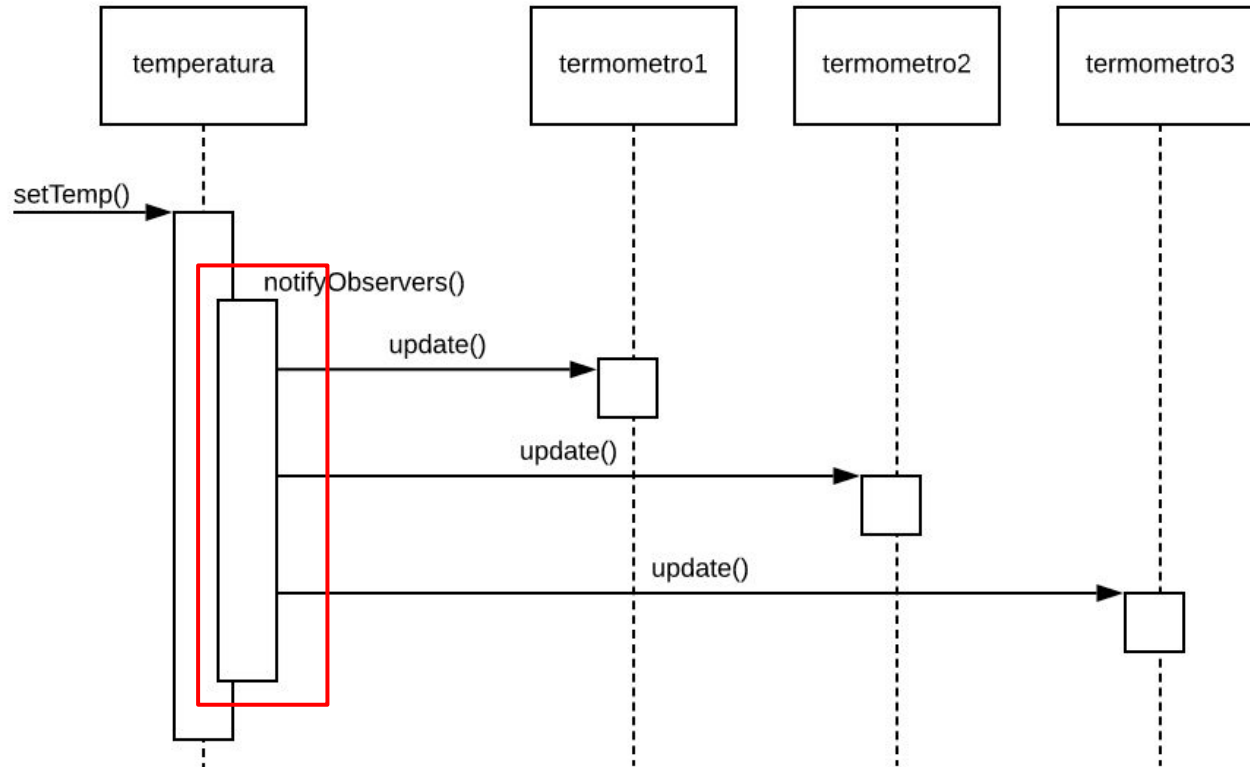
Notifica todos os observadores (isto é, Termômetros) que foram adicionados a uma Temperatura

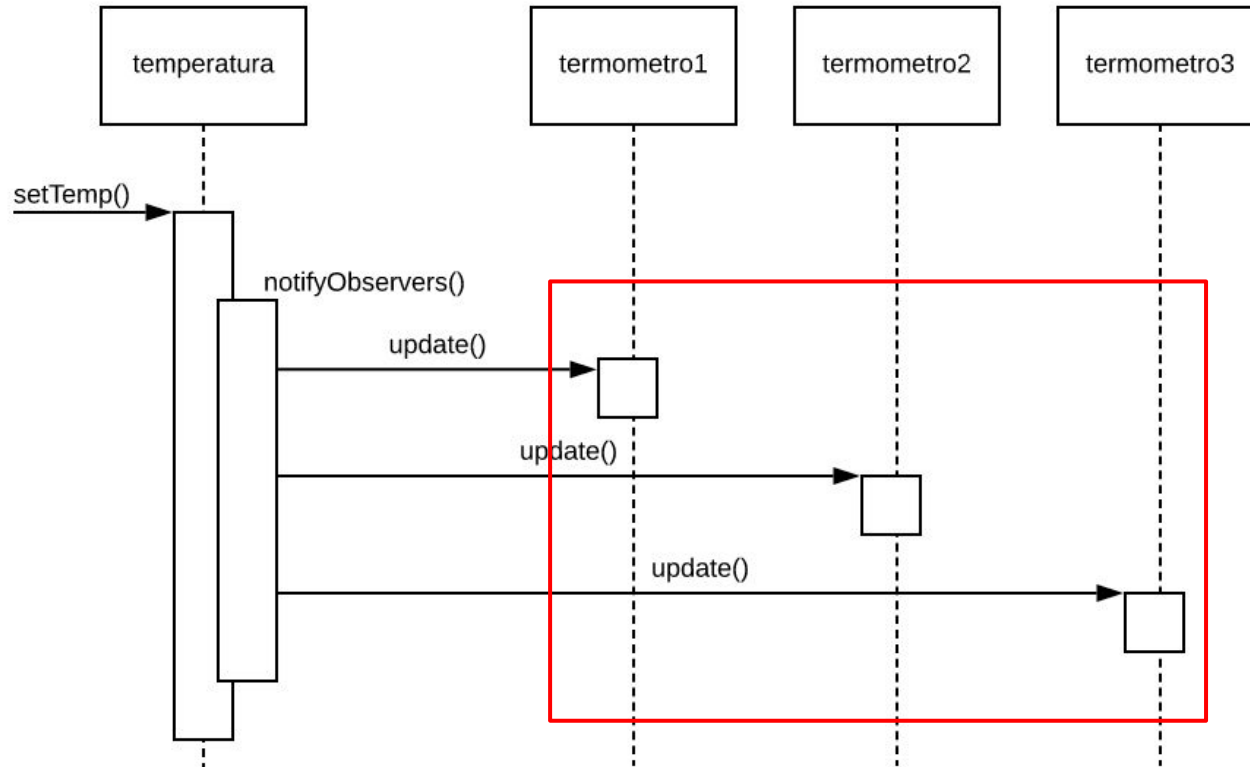
Uma classe Termômetro

```
class TermometroCelsius implements Observer{  
  
    public void update(Subject s){  
        double temp = ((Temperatura) s).getTemp();  
        System.out.println("Temperatura Celsius: " + temp);  
    }  
  
}
```

Todos Observadores devem implementar esse método.
Chamada de **notifyObservers** (em Temperatura) resulta na execução de **update** de cada um de seus observadores







(9) Template Method

Contexto: Folha de Pagamento

- Uma classe base: Funcionario
- Duas subclasses: FuncionarioPublico e FuncionarioCLT

Problema

- Função que calcula salário de funcionários:
 - Passos são semelhantes para func. públicos e CLT
 - Porém, existem alguns detalhes diferentes
- Na classe pai (Funcionario) queremos definir o "workflow principal" (ou o template) para cálculo de salários
- E deixar aberto para as subclasses os refinamentos desses passos

Solução: Padrão **Template Method**

- Especifica como implementar o esqueleto de um algoritmo em uma classe abstrata X
- Mas deixando pendente alguns passos (ou métodos abstratos) para serem implementados nas subclasses
- Permite que subclasses customizem um algoritmo, mas sem mudar sua estrutura

```
abstract class Funcionario {
```

```
    double salario;
```

```
    ...
```

```
private abstract double calcDescontosPrevidencia();
```

```
private abstract double calcDescontosPlanoSaude();
```

```
private abstract double calcOutrosDescontos();
```

```
public double calcSalarioLiquido { // template method
```

```
    double prev = calcDescontosPrevidencia();
```

```
    double saude = calcDescontosPlanoSaude();
```

```
    double outros = calcOutrosDescontos();
```

```
    return salario - prev - saude - outros;
```

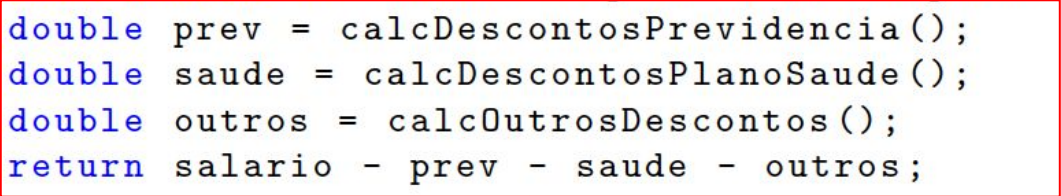
```
}
```

```
}
```

Serão
implementados
pelas
subclasses

```
abstract class Funcionario {  
  
    double salario;  
    ...  
    private abstract double calcDescontosPrevidencia();  
    private abstract double calcDescontosPlanoSaude();  
    private abstract double calcOutrosDescontos();  
  
    public double calcSalarioLiquido { // template method  
        double prev = calcDescontosPrevidencia();  
        double saude = calcDescontosPlanoSaude();  
        double outros = calcOutrosDescontos();  
        return salario - prev - saude - outros;  
    }  
}
```

```
abstract class Funcionario {  
  
    double salario;  
  
    ...  
    private abstract double calcDescontosPrevidencia();  
    private abstract double calcDescontosPlanoSaude();  
    private abstract double calcOutrosDescontos();  
  
    public double calcSalarioLiquido { // template method  
        double prev = calcDescontosPrevidencia();  
        double saude = calcDescontosPlanoSaude();  
        double outros = calcOutrosDescontos();  
        return salario - prev - saude - outros;  
    }  
}
```

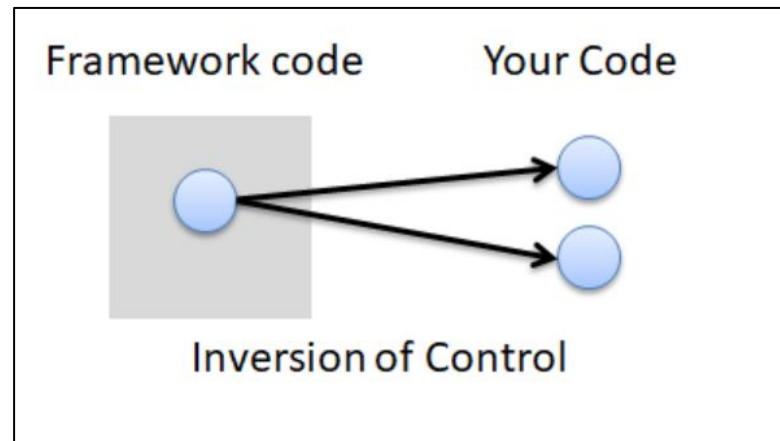
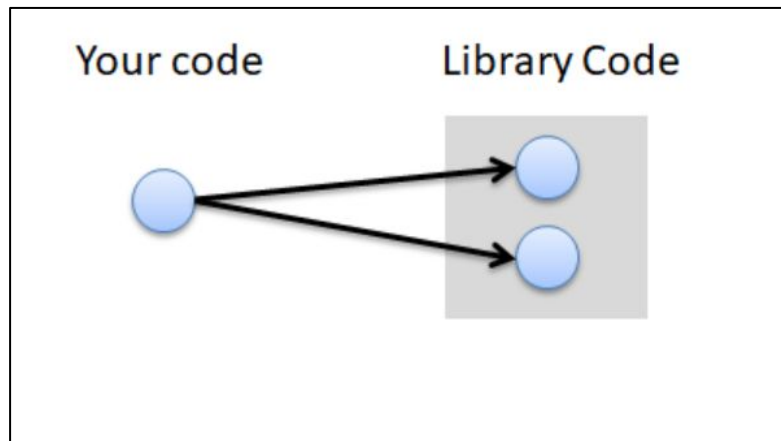


Passos principais (ou template, ou modelo)
para cálculo de salário líquido

Inversão de Controle

- Template Method é usado para implementar **Inversão de Controle**, principalmente em frameworks
- Framework: define o "modelo" de um algoritmo/sistema
 - Mas clientes podem parametrizar alguns passos
 - Framework chama esse código dos clientes
 - Daí o termo inversão de controle

Frameworks vs Bibliotecas



Fonte: <https://github.com/prmr/SoftwareDesign/blob/master/modules/Module-06.md>

Exercícios de Fixação

Assinale V ou F

() Observador é um padrão que permite que um objeto avise outros objetos de que seu estado mudou.

() Template Method é um padrão que define o esqueleto de um algoritmo em uma classe base e delega a implementação de alguns passos para subclasses.

() Strategy usa herança para variar partes de um algoritmo.
Template Method usa composição para variar um algoritmo inteiro.

Assinale V ou F

- () No padrão Observador, os objetos sujeito (Subject) conhecem e possuem dependências para as classes concretas dos observadores.
- () Template Methods são usados para implementar inversão de controle, quando, por exemplo, uma classe pai chama métodos concretos implementados em uma classe filha.
- () Template Methods são muito usados para implementar frameworks. Quando isso acontece, os métodos template são implementados por classes clientes do framework.

Créditos

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

Assinale V ou F

(V) Observador é um padrão que permite que um objeto avise outros objetos de que seu estado mudou.

(V) Template Method é um padrão que define o esqueleto de um algoritmo em uma classe base e delega a implementação de alguns passos para subclasses.

(F) Strategy usa herança para variar partes de um algoritmo.
Template Method usa composição para variar um algoritmo inteiro.

Assinale V ou F

(F) No padrão Observador, os objetos sujeito (Subject) conhecem e possuem dependências para as classes concretas dos observadores.

(V) Template Methods são usados para implementar inversão de controle, quando, por exemplo, uma classe pai chama métodos concretos implementados em uma classe filha.

(F) Template Methods são muito usados para implementar frameworks. Quando isso acontece, os métodos template são implementados por classes clientes do framework.

Fim