



Instituto Federal de Educação, Ciência e Tecnologia da Bahia - IFBA  
Departamento de Ciência da Computação  
Tecnólogo em Análise e Desenvolvimento de Sistemas

## Padrões de Projeto

André L. R. Madureira <[andre.madureira@ifba.edu.br](mailto:andre.madureira@ifba.edu.br)>  
Doutorando em Ciência da Computação (UFBA)  
Mestre em Ciência da Computação (UFBA)  
Engenheiro da Computação (UFBA)

# Padrões de projeto

---

- Surgiu com Alexander et al. (1977): *“existem padrões comuns de projeto de prédios que são inerentemente agradáveis e eficazes”*
  - Podemos adaptar essa mesma ideia para softwares
- **Padrão:** descrição do problema e da essência de sua solução, de modo que a solução possa ser reusada em diferentes contextos
  - **Um padrão não é uma especificação detalhada**
    - *“Se trata de uma descrição de conhecimento e experiência”*
    - *“Ou uma solução já aprovada para um problema comum”*

# Padrões de projeto

- **Vantagens:**

- Reuso de conhecimentos e experiências que representam as melhores práticas e bons projetos
- Uso de soluções já testadas para problemas comuns
- Ajudam a explicar o projeto



# Elementos Essenciais de Padrões de projeto (GAMMA et al., 1995)

---

- **Nome de referência** (para o padrão)
  - Nome que identifique o padrão
- **Aplicabilidade**
  - Condições para aplicação do modelo / padrão de projeto
- **Descrição da solução** (e do padrão)
  - Estrutura, relacionamentos e implementação
- **Declaração das conseqüências**
  - Resultados e compromissos da aplicação do padrão

# Padrões de projeto

---

- Existem 23 padrões de projeto diferentes, classificados como:
  - **Criacionais (padrões de criação)**
    - Abstraem e ou adiam o processo criação dos objetos
  - **Estruturais**
    - Lidam com a composição de classes e objetos
  - **Comportamentais**
    - Se concentram na comunicação entre objetos

# Padrões de projeto

---

- Existem 23 padrões de projeto diferentes, classificados como:

## *Creational*

1. Singleton
2. Factory
3. Abstract Factory
4. Builder
5. Prototype

## *Structural*

6. Adapter
7. Composite
8. Proxy
9. Fly Weight
10. Facade
11. Bridge
12. Decorator

## *Behavioural*

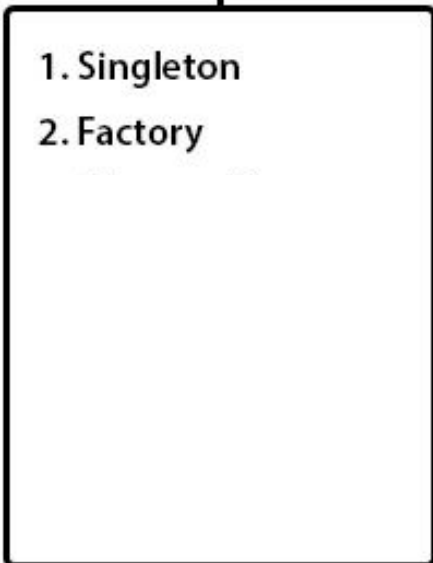
13. Template Method
14. Mediator
15. Chain of Responsibility
16. Observer
17. Strategy
18. Command
19. State
20. Visitor
21. Iterator
22. Interpreter
23. Memento

# Padrões de projeto

---

- Nesta disciplina iremos nos concentrar nos **padrões mais usados**:

## *Creational*

- 
- A vertical rectangle with a black border containing a list of two items. A short vertical line extends from the top center of the rectangle.
1. Singleton
  2. Factory

## *Structural*

- 
- A vertical rectangle with a black border containing one item. A short vertical line extends from the top center of the rectangle.
10. Facade

## *Behavioural*

- 
- A vertical rectangle with a black border containing two items. A short vertical line extends from the top center of the rectangle.
16. Observer
  17. Strategy

# Exercício

---

Considerando os padrões de projeto, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Um padrão de projeto é uma descrição do problema e de sua solução, de forma a permitir a sua reutilização em diferentes contextos.

II - Padrões de projeto são especificações detalhadas.

III - Os elementos essenciais de todo padrão de projeto são o nome, aplicabilidade, descrição da solução e declaração das consequências.

IV - Os padrões de projeto podem ser classificados em composicionais, estáticos, e dinâmicos. O primeiro faz a composição de classes e objetos, o segundo abstrai aspectos estruturais do projeto, e o último tem ênfase maior na comunicação entre objetos.

- ☐ Todas as assertivas são VERDADEIRAS.
- ☐ Somente I, II e III.
- ☐ Somente I e III.
- ☐ Somente II, III e IV.
- ☐ Nenhuma das alternativas anteriores.



# Exercício

---

Considerando os padrões de projeto, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Um padrão de projeto é uma descrição do problema e de sua solução, de forma a permitir a sua reutilização em diferentes contextos. **V**

II - Padrões de projeto são especificações detalhadas.

III - Os elementos essenciais de todo padrão de projeto são o nome, aplicabilidade, descrição da solução e declaração das consequências.

IV - Os padrões de projeto podem ser classificados em composicionais, estáticos, e dinâmicos. O primeiro faz a composição de classes e objetos, o segundo abstrai aspectos estruturais do projeto, e o último tem ênfase maior na comunicação entre objetos.

- ☐ Todas as assertivas são VERDADEIRAS.
- ☐ Somente I, II e III.
- ☐ Somente I e III.
- ☐ Somente II, III e IV.
- ☐ Nenhuma das alternativas anteriores.

# Exercício

Considerando os padrões de projeto, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Um padrão de projeto é uma descrição do problema e de sua solução, de forma a permitir a sua reutilização em diferentes contextos. **V**

II - Padrões de projeto são especificações **detalhadas**. **F**

III - Os elementos essenciais de todo padrão de projeto são o nome, aplicabilidade, descrição da solução e declaração das consequências.

IV - Os padrões de projeto podem ser classificados em composicionais, estáticos, e dinâmicos. O primeiro faz a composição de classes e objetos, o segundo abstrai aspectos estruturais do projeto, e o ultimo tem ênfase maior na comunicação entre objetos.

- ☐ Todas as assertivas são VERDADEIRAS.
- ☐ Somente I, II e III.
- ☐ Somente I e III.
- ☐ Somente II, III e IV.
- ☐ Nenhuma das alternativas anteriores.

# Exercício

Considerando os padrões de projeto, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Um padrão de projeto é uma descrição do problema e de sua solução, de forma a permitir a sua reutilização em diferentes contextos. **V**

II - Padrões de projeto são especificações **detalhadas**. **F**

III - Os elementos essenciais de todo padrão de projeto são o nome, aplicabilidade, descrição da solução e declaração das consequências. **V**

IV - Os padrões de projeto podem ser classificados em composicionais, estáticos, e dinâmicos. O primeiro faz a composição de classes e objetos, o segundo abstrai aspectos estruturais do projeto, e o ultimo tem ênfase maior na comunicação entre objetos.

- ☐ Todas as assertivas são VERDADEIRAS.
- ☐ Somente I, II e III.
- ☐ Somente I e III.
- ☐ Somente II, III e IV.
- ☐ Nenhuma das alternativas anteriores.

# Exercício

Considerando os padrões de projeto, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Um padrão de projeto é uma descrição do problema e de sua solução, de forma a permitir a sua reutilização em diferentes contextos. **V**

II - Padrões de projeto são especificações **detalhadas**. **F**

III - Os elementos essenciais de todo padrão de projeto são o nome, aplicabilidade, descrição da solução e declaração das consequências. **V**

IV - Os padrões de projeto podem ser classificados em **composicionais, estáticos, e dinâmicos**. O primeiro faz a composição de classes e objetos, o segundo abstrai aspectos estruturais do projeto, e o ultimo tem ênfase maior na comunicação entre objetos. **F**

- ☐ Todas as assertivas são VERDADEIRAS.
- ☐ Somente I, II e III.
- ☒ Somente I e III.
- ☐ Somente II, III e IV.
- ☐ Nenhuma das alternativas anteriores.

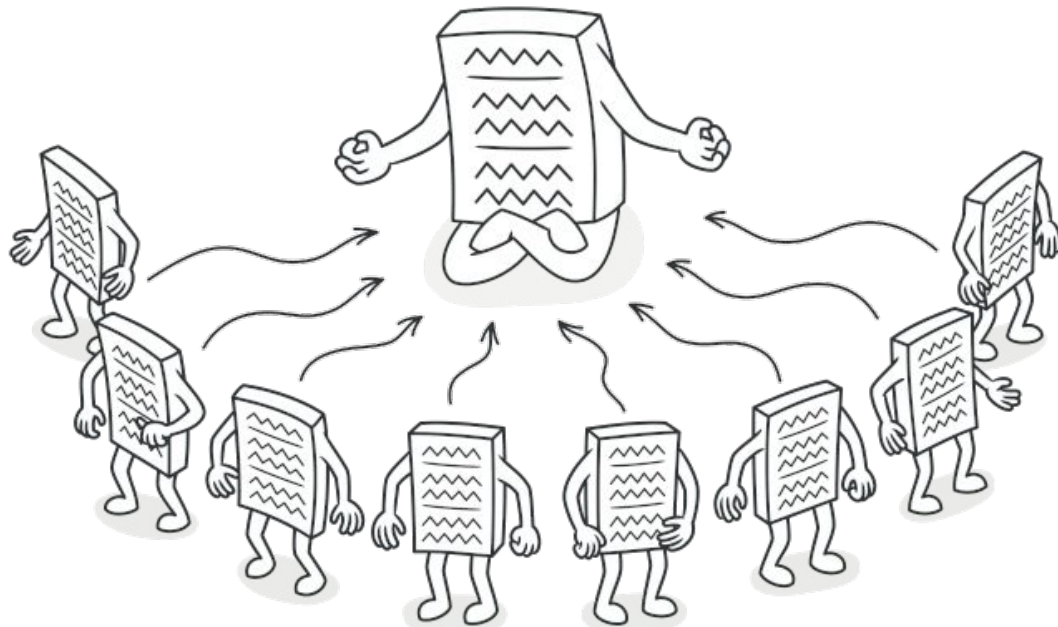
# Padrão Singleton

---

## **Aplicabilidade:**

Garante que uma classe tenha **apenas uma instância**

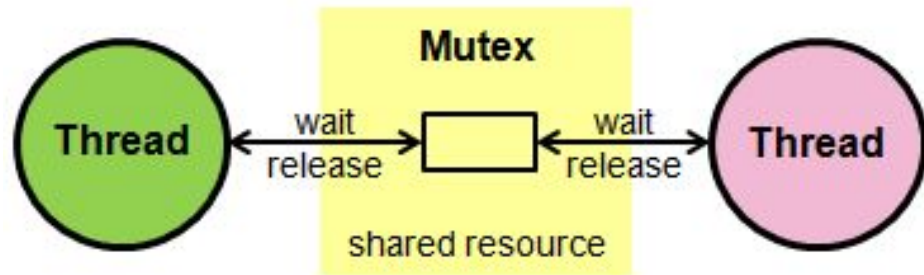
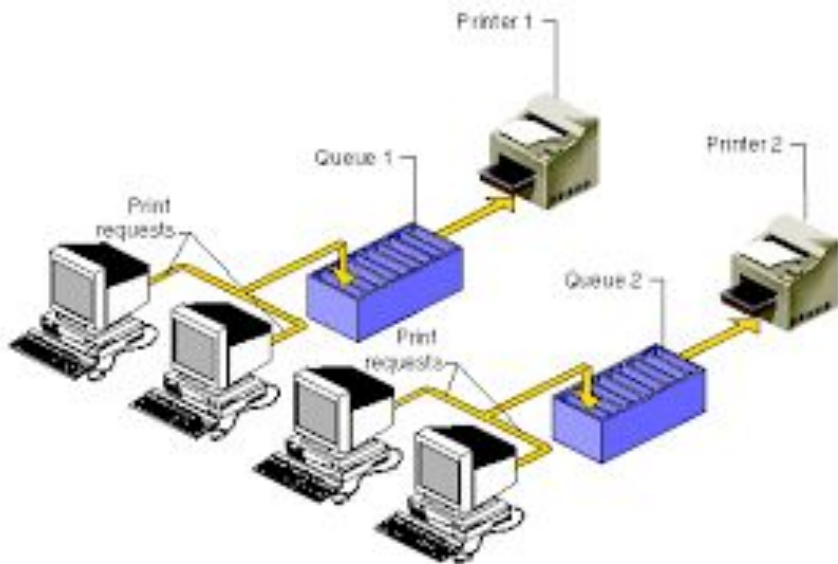
Há um ponto de acesso global para essa instância única





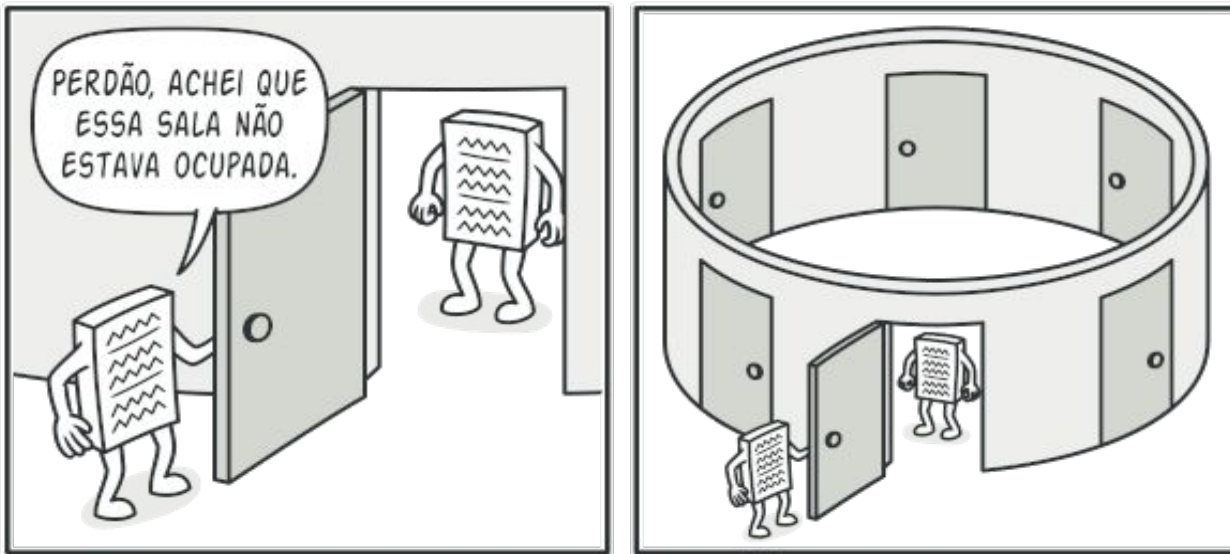
# Porque usar o padrão Singleton ?

**Controlar o acesso a recursos compartilhados**  
(DBs, impressora, CD, pendrive, etc)



# Porque usar o padrão Singleton ?

**Fornecer um ponto de acesso global para a instância**  
(o padrão Singleton permite que você acesse o objeto de qualquer lugar no programa)



# Padrão Singleton

---

## **Consequências:**

Impossibilidade de criar mais de uma instância para a classe Singleton

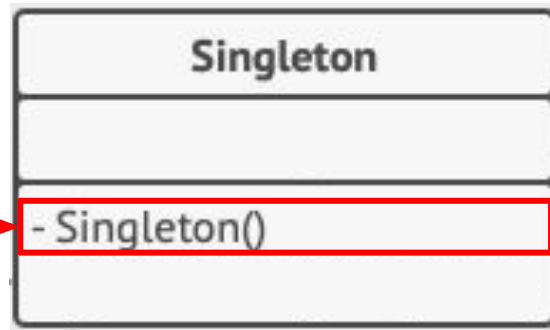
**Há um único ponto de falha no sistema**  
(o objeto Singleton)

Qualquer problema associado ao estado do objeto Singleton **afeta o sistema todo**



# Padrão Singleton - Descrição da Solução

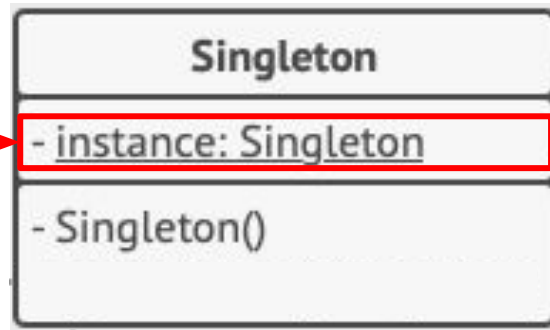
---



**1) Torne o construtor da classe  
um método PRIVADO**  
*(controle da criação de instâncias)*

# Padrão Singleton - Descrição da Solução

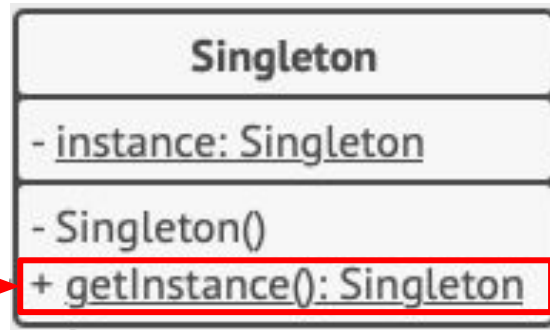
---



2) Crie um **atributo PRIVADO** para armazenar a única instância da classe Singleton

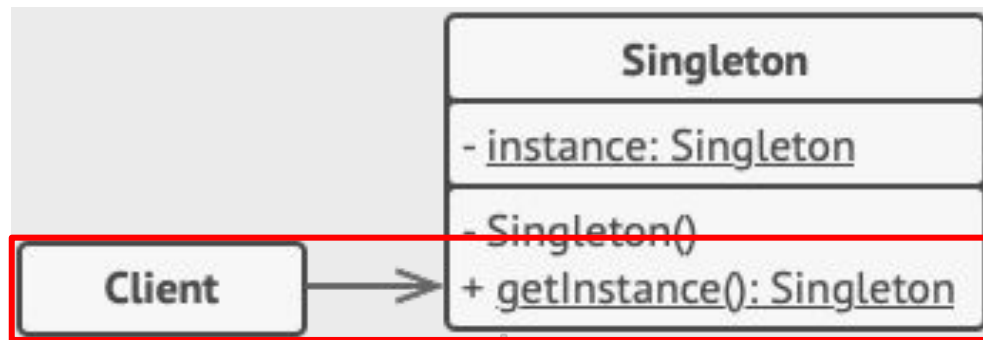
# Padrão Singleton - Descrição da Solução

---



**3) Crie um método ESTÁTICO PÚBLICO** para obter a instância do Singleton

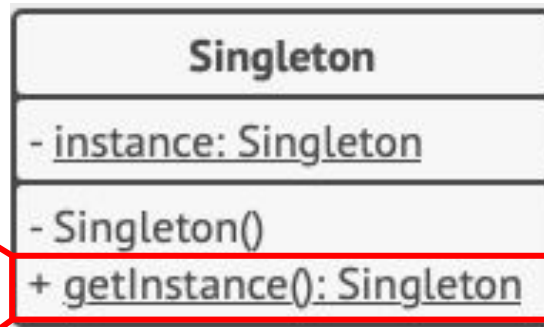
# Padrão Singleton - Descrição da Solução



Quando alguém quiser acessar a instância do Singleton, ele deve chamar o método estático **Singleton.getInstance()**

# Padrão Singleton - Descrição da Solução

```
if (instance == null) {  
    // Atenção: se você está criando uma  
    // aplicação com apoio multithreading,  
    // você deve colocar um thread lock aqui.  
    instance = new Singleton()  
}  
return instance
```



**Se instance == null:**  
Criar a instância do Singleton.  
Retornar a instância criada.

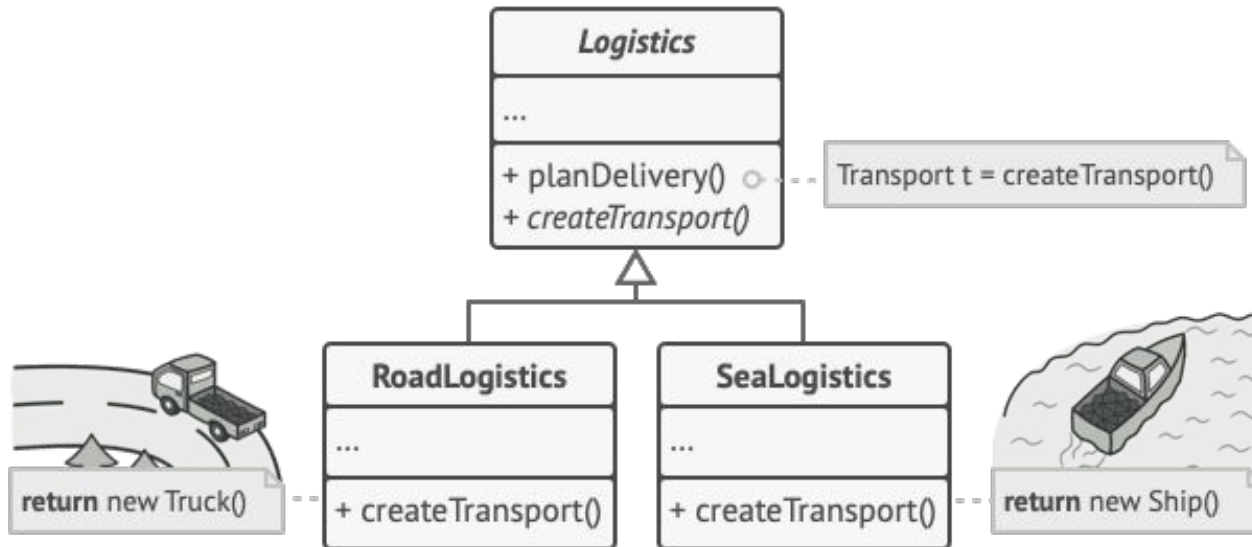
**Se instance != null:**  
Somente retornar a instância.

# Padrão Factory (*Factory Method*)

## Aplicabilidade:

Uma superclasse gerencia a criação de objetos de subclasses

Chamadas diretas de construção de objetos (operador **new**) são substituídas por um **método fábrica** (*factory method*)



# Porque usar o padrão Factory (*Factory Method*)?

**Facilita a evolução do sistema**  
(adição de novas subclasses e recursos)



# Padrão Factory (*Factory Method*)

---

## **Consequências:**

Aumento na complexidade do código

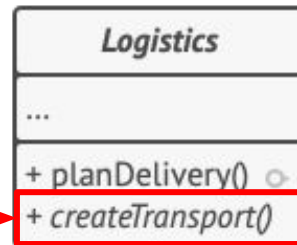
O padrão Factory pode demandar a criação de muitas subclasses, o que torna o código complexo





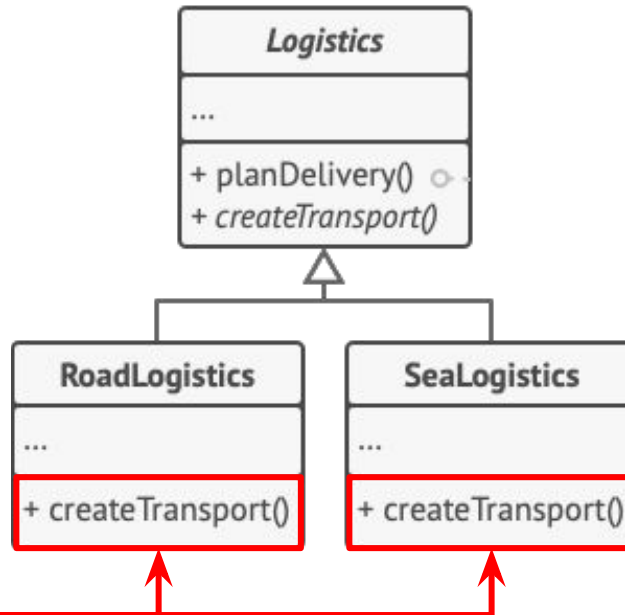
# Padrão Factory - Descrição da Solução

---



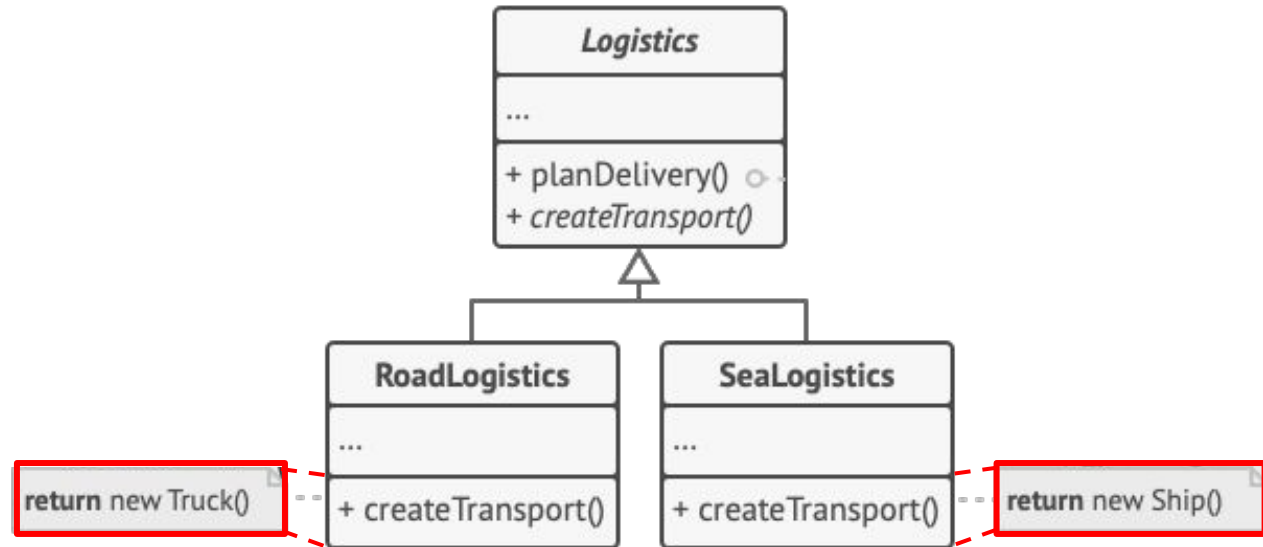
1) Crie uma classe ABSTRATA ou interface com um método abstrato **create()**

# Padrão Factory - Descrição da Solução



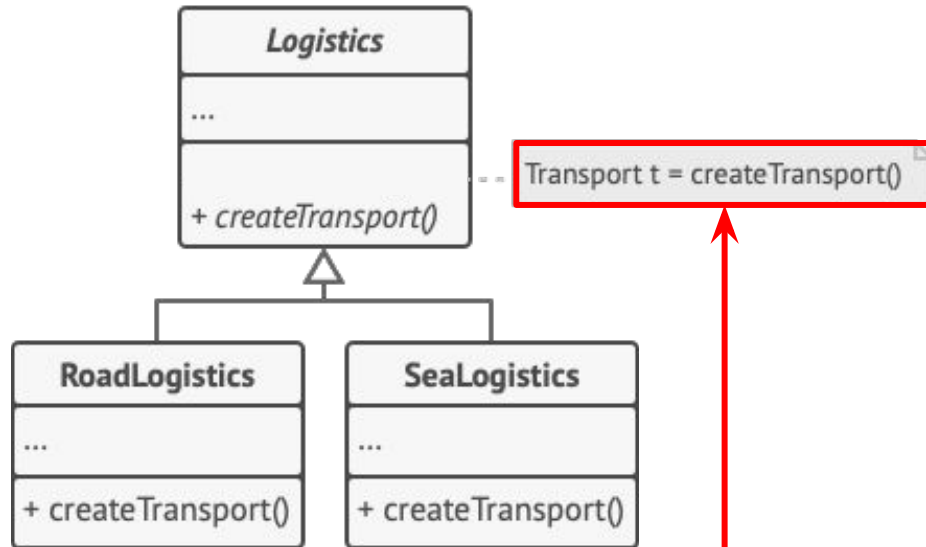
2) Crie classes CONCRETAS que implementem o método **create()**

# Padrão Factory - Descrição da Solução



A implementação do método **create()** é onde ocorre a criação dos objetos

# Padrão Factory - Descrição da Solução



3) Quando for criar uma instância de uma subclasse, use o método **create()**

# Exercício

---

Considerando os padrões de projeto, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - O padrão Singleton garante a existência de apenas uma instância de uma classe.

II - Pelo padrão Singleton, há apenas um ponto de acesso global para instancias de uma classe.

III - O padrão Singleton cria um único ponto de falha no sistema.

IV - O padrão Factory substitui chamadas diretas de construção de objetos por métodos fábrica.

- ☐ Todas as assertivas são VERDADEIRAS.
- ☐ Somente I e IV.
- ☐ Somente I, II e IV.
- ☐ Somente II e III.
- ☐ Nenhuma das alternativas anteriores.

# Exercício

---

Considerando os padrões de projeto, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - O padrão Singleton garante a existência de apenas uma instância de uma classe. **V**

II - Pelo padrão Singleton, há apenas um ponto de acesso global para instancias de uma classe.

III - O padrão Singleton cria um único ponto de falha no sistema.

IV - O padrão Factory substitui chamadas diretas de construção de objetos por métodos fábrica.

- ☐ Todas as assertivas são VERDADEIRAS.
- ☐ Somente I e IV.
- ☐ Somente I, II e IV.
- ☐ Somente II e III.
- ☐ Nenhuma das alternativas anteriores.

# Exercício

---

Considerando os padrões de projeto, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - O padrão Singleton garante a existência de apenas uma instância de uma classe. **V**

II - Pelo padrão Singleton, há apenas um ponto de acesso global para instancias de uma classe. **V**

III - O padrão Singleton cria um único ponto de falha no sistema.

IV - O padrão Factory substitui chamadas diretas de construção de objetos por métodos fábrica.

- ☐ Todas as assertivas são VERDADEIRAS.
- ☐ Somente I e IV.
- ☐ Somente I, II e IV.
- ☐ Somente II e III.
- ☐ Nenhuma das alternativas anteriores.

# Exercício

---

Considerando os padrões de projeto, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - O padrão Singleton garante a existência de apenas uma instância de uma classe. **V**

II - Pelo padrão Singleton, há apenas um ponto de acesso global para instancias de uma classe. **V**

III - O padrão Singleton cria um único ponto de falha no sistema. **V**

IV - O padrão Factory substitui chamadas diretas de construção de objetos por métodos fábrica.

- ☐ Todas as assertivas são VERDADEIRAS.
- ☐ Somente I e IV.
- ☐ Somente I, II e IV.
- ☐ Somente II e III.
- ☐ Nenhuma das alternativas anteriores.



# Exercício

Considerando os padrões de projeto, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - O padrão Singleton garante a existência de apenas uma instância de uma classe. **V**

II - Pelo padrão Singleton, há apenas um ponto de acesso global para instancias de uma classe. **V**

III - O padrão Singleton cria um único ponto de falha no sistema. **V**

IV - O padrão Factory substitui chamadas diretas de construção de objetos por métodos fábrica. **V**

☒ Todas as assertivas são VERDADEIRAS.

☐ Somente I e IV.

☐ Somente I, II e IV.

☐ Somente II e III.

☐ Nenhuma das alternativas anteriores.

# Padrão Facade (tradução livre – *Fachada*)

## Aplicabilidade:

Fornece uma interface simples para um subsistema complexo (biblioteca, framework, ou qualquer conjunto de classes)

**Sistema**  
(Cliente)



**Fachada**  
(Atendente)



**Subsistema**  
(Setores da Loja)



# Porque usar o padrão Facade?

**Facilita a utilização e manutenção de bibliotecas, frameworks e demais códigos**  
(lógica de negócio separada dos detalhes de implementação das classes de terceiros)

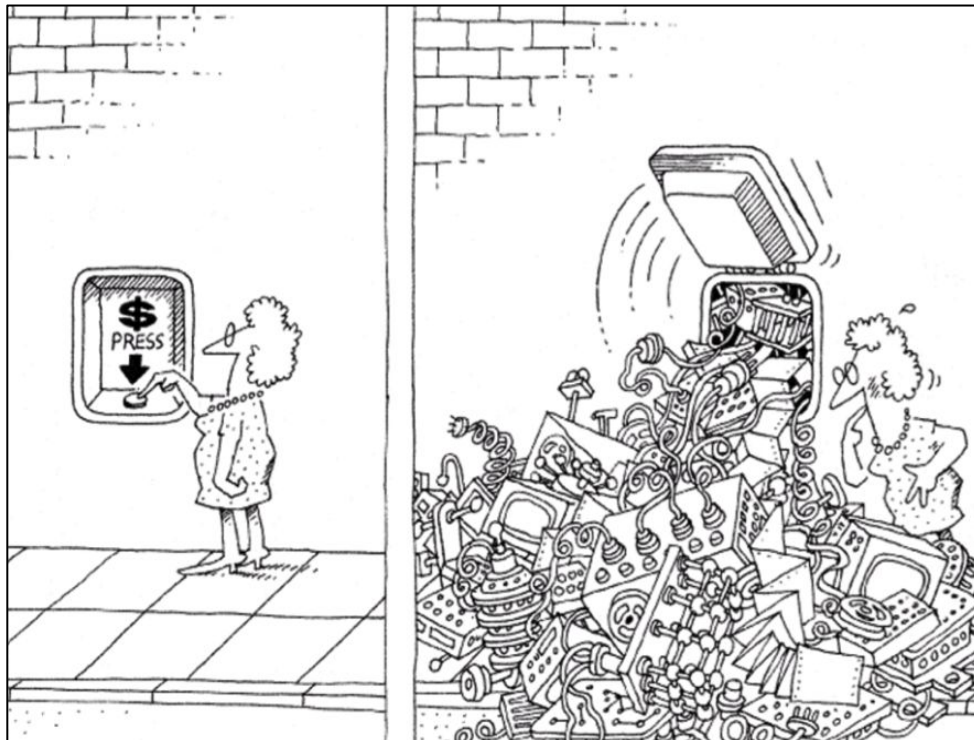


# Porque usar o padrão Facade?

Ocultar detalhes de implementação  
e complexidades dos subsistemas

**KISS**  
**KEEP IT SIMPLE, STUPID**  
**STUPID SIMPLE**

(JOHNSON, 1938)



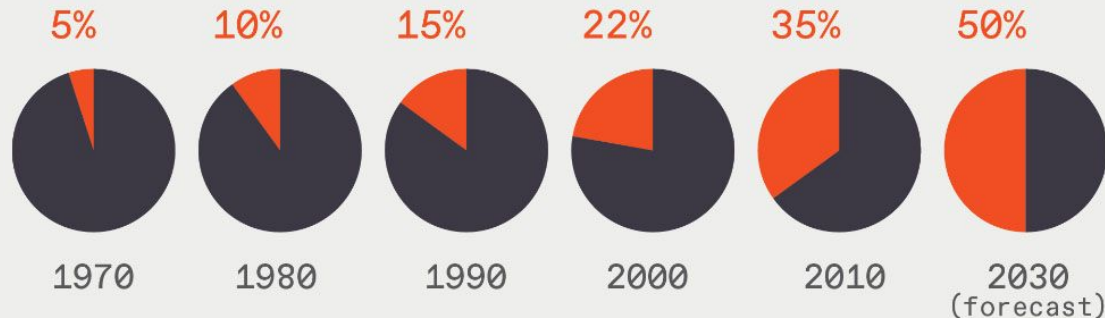
# Padrão Facade

## Consequências:

Ocultar detalhes de implementação reduz a flexibilidade no uso dos subsistemas



## ELECTRONICS SYSTEM AS PERCENT OF TOTAL CAR COST



# Padrão Facade

---

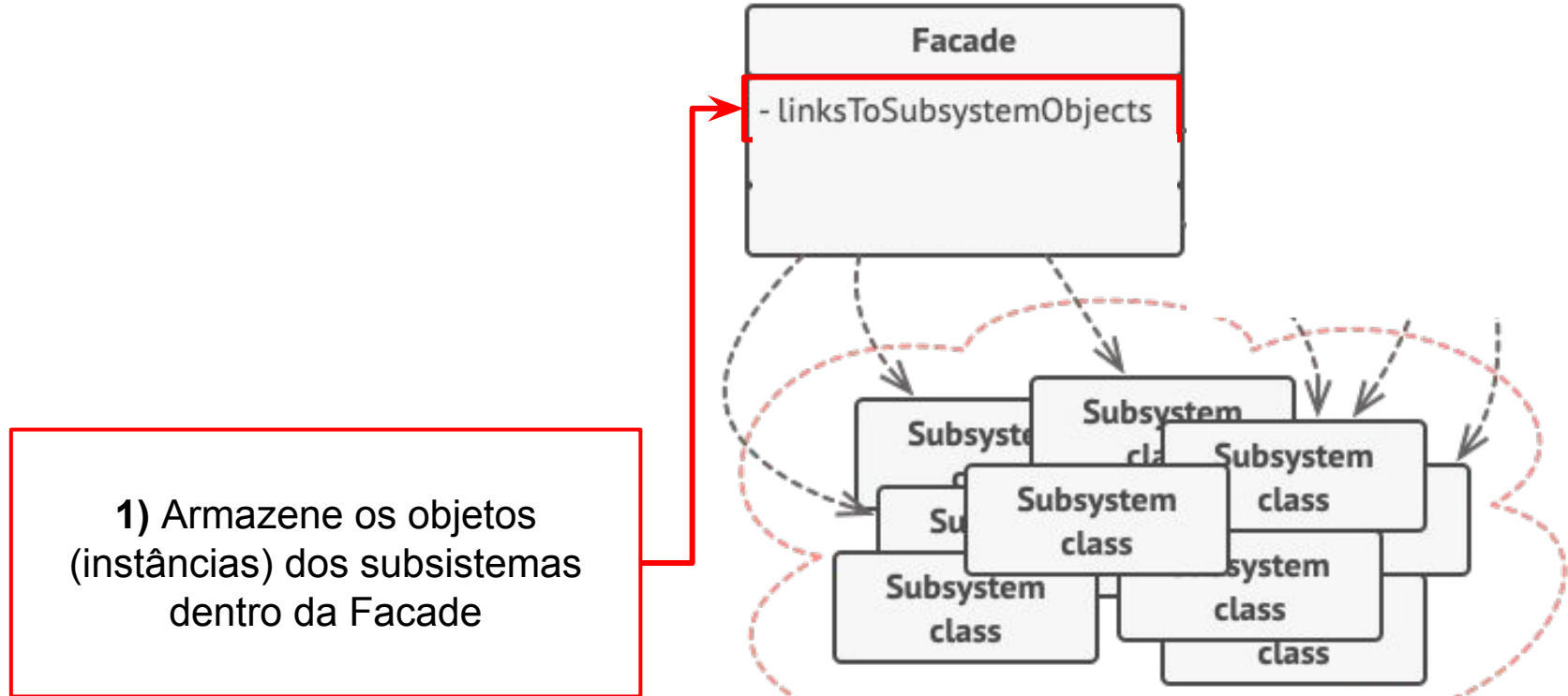
## Consequências:

Ocultar detalhes de implementação reduz a flexibilidade no uso dos subsistemas

Subsistemas complexos possuem muitos recursos, que demandam conhecimento sobre a implementação

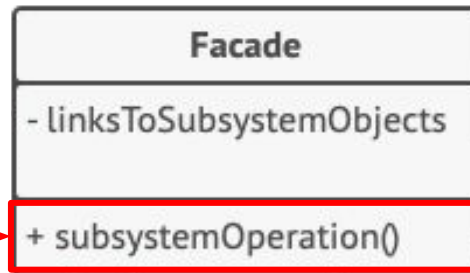
Simplificar o uso do subsistema e ter acesso a todos os recursos disponíveis neste sistema são **alvos antagônicos**

# Padrão Facade - Descrição da Solução



# Padrão Facade - Descrição da Solução

---

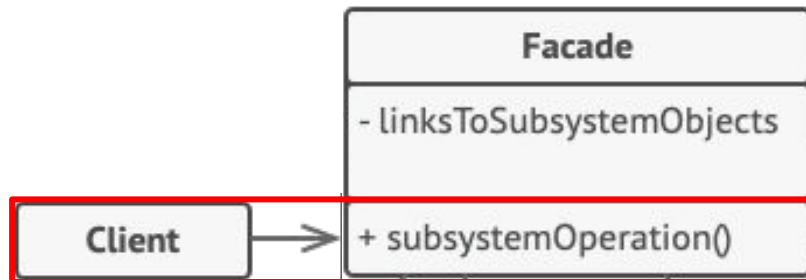


**2)** Crie métodos PÚBLICOS na Facade para utilizar os subsistemas



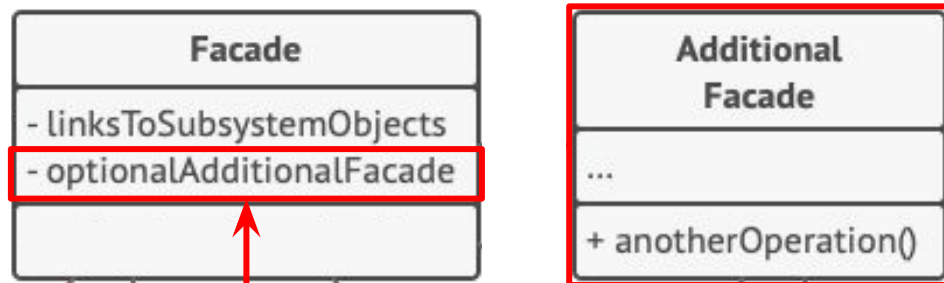
# Padrão Facade - Descrição da Solução

---



**3)** O sistema deve chamar os métodos PÚBLICOS da Facade quando precisar usar recursos de um subsistema

# Padrão Facade - Descrição da Solução



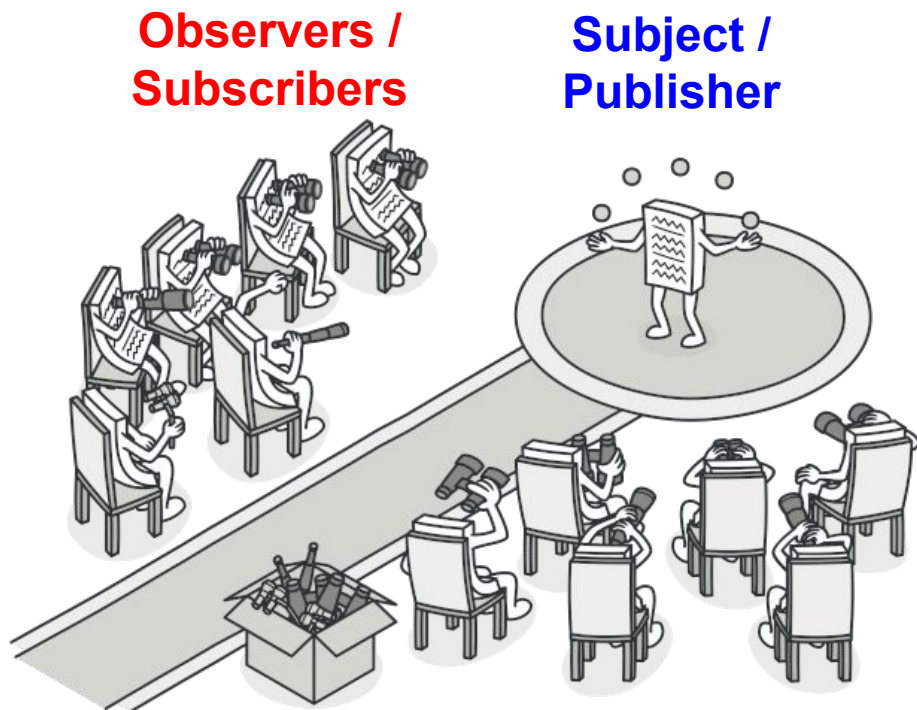
4) Você pode criar mais de uma Facade, conforme necessário

# Padrão Observer (Publish/Subscribe ou PubSub)

## Aplicabilidade:

Permite monitorar um objeto  
(*subject* ou *publisher*)

Permite implementação de **eventos**, disparados quando há mudança no estado do *subject*

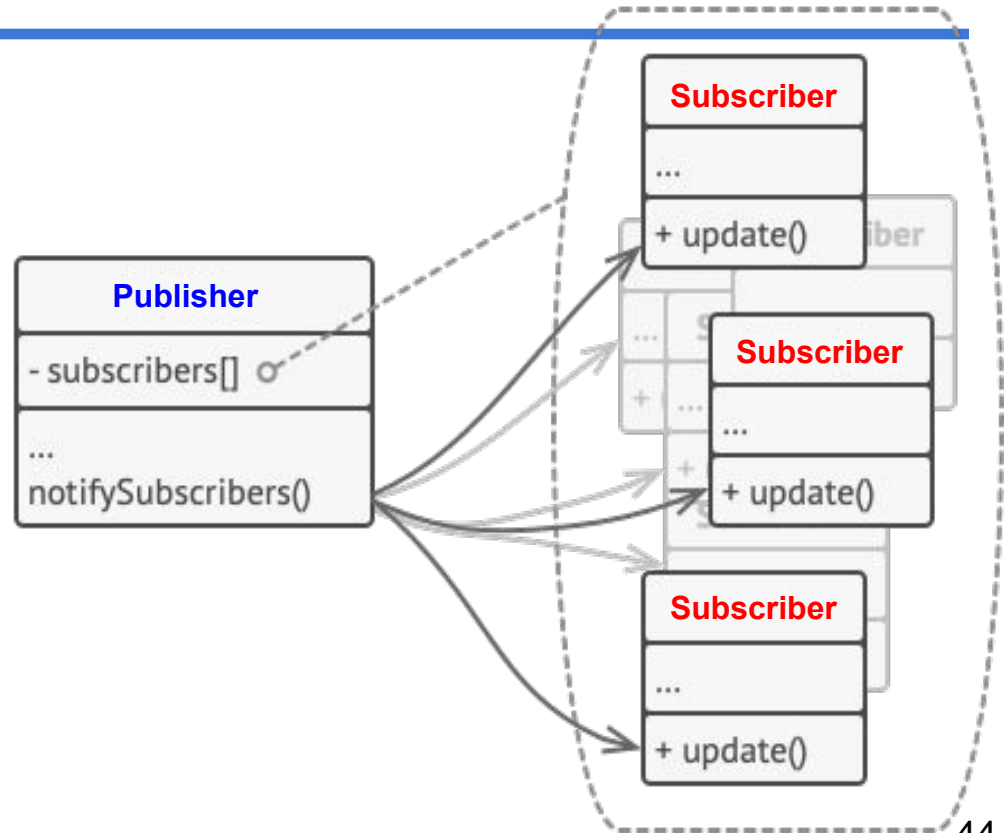


# Padrão Observer (Publish/Subscribe ou PubSub)

## Consequências:

Objeto monitorado (publisher)  
conhece apenas a interface do  
objeto que o monitora (subscriber)

Isto é, *Publisher* conhece apenas  
a interface de *Subscriber*  
( método **update()** )



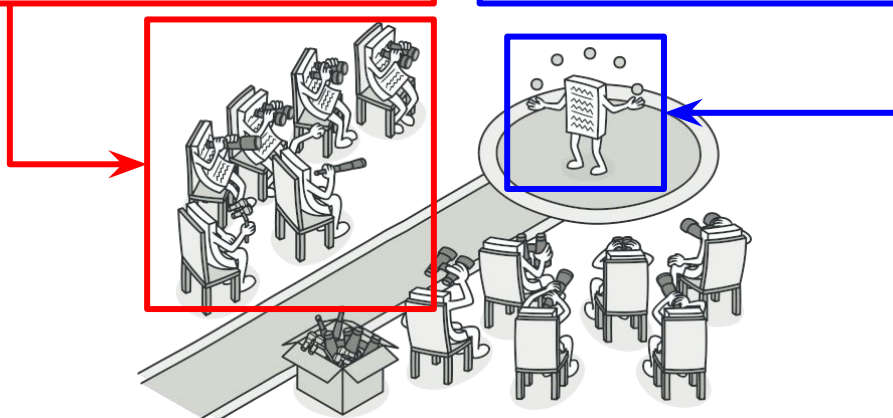
# Padrão Observer - Descrição da Solução

## Observer (*subscriber*):

- Objeto interessado no conteúdo produzido pelo **subject**
- Recebe notificação de atualização do **subject**

## Subject (*publisher*):

- É um classe / objeto que possui **estado**.
- Notifica o **observer** quando o estado muda.



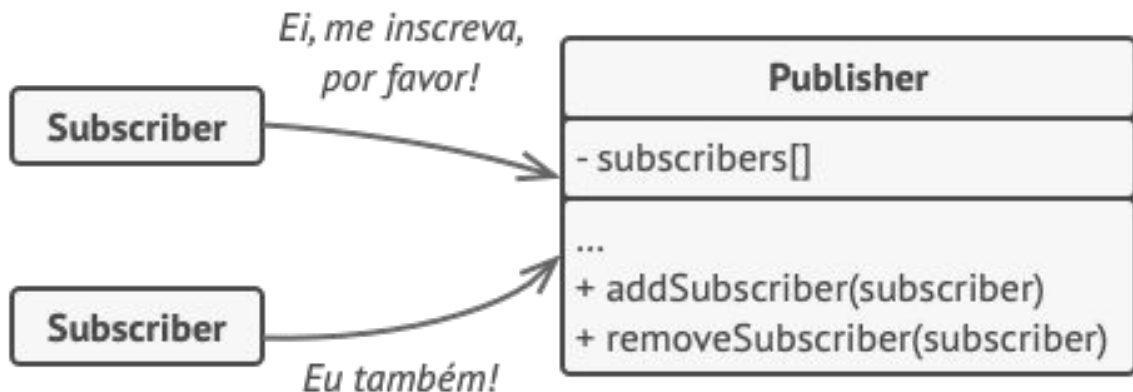
# Padrão Observer - Descrição da Solução

## Observer (*subscriber*):

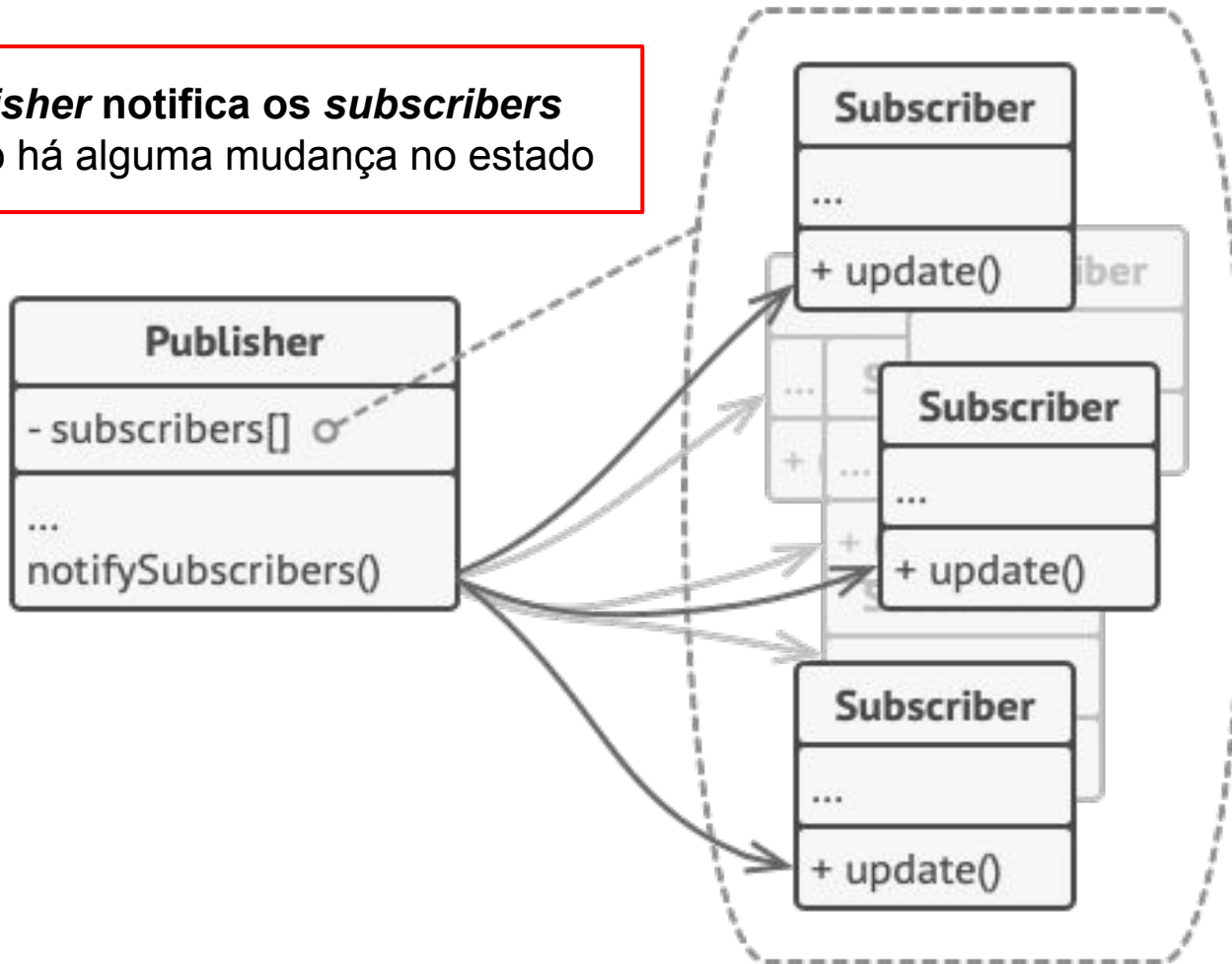
- Objeto interessado no conteúdo produzido pelo **subject**
- Recebe notificação de atualização do **subject**

## Subject (*publisher*):

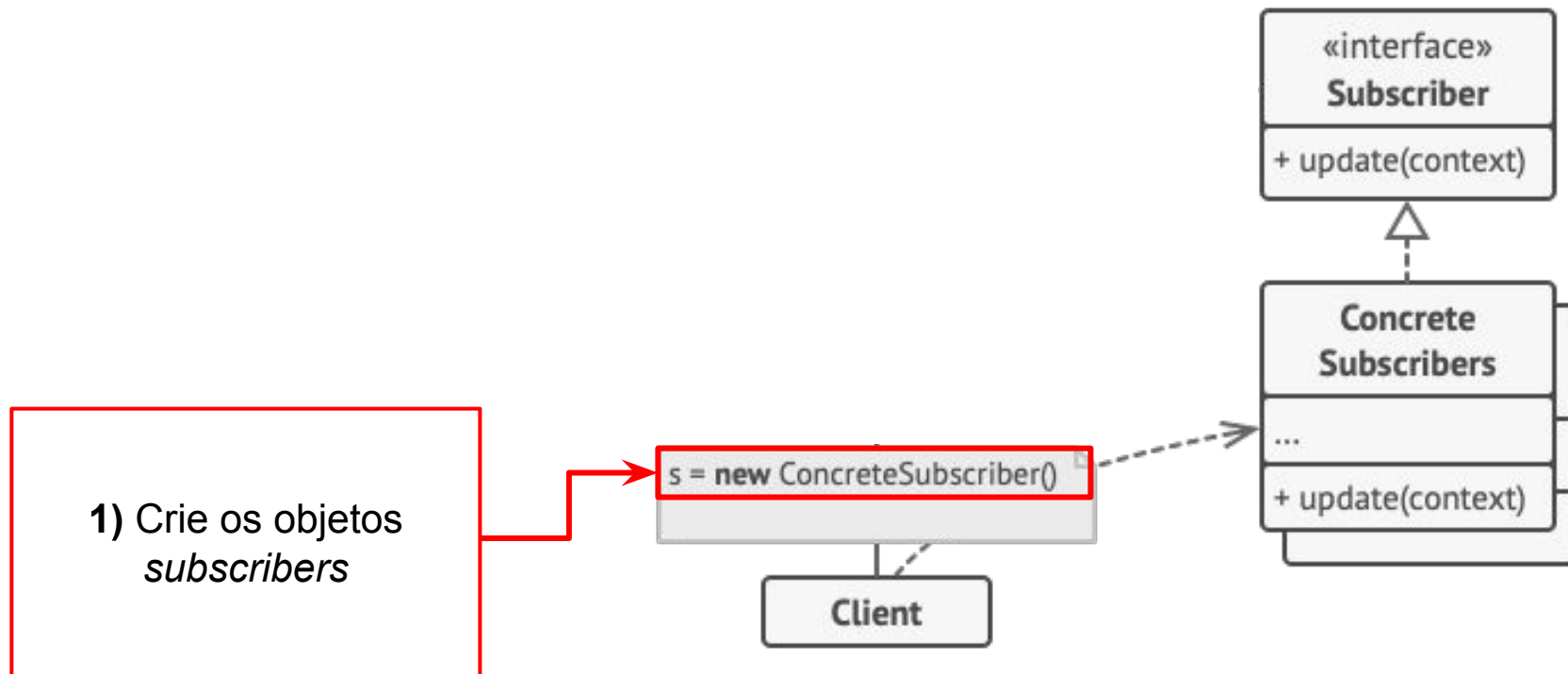
- É um classe / objeto que possui **estado**.
- Notifica o **observer** quando o estado muda.



***Publisher* notifica os *subscribers***  
quando há alguma mudança no estado

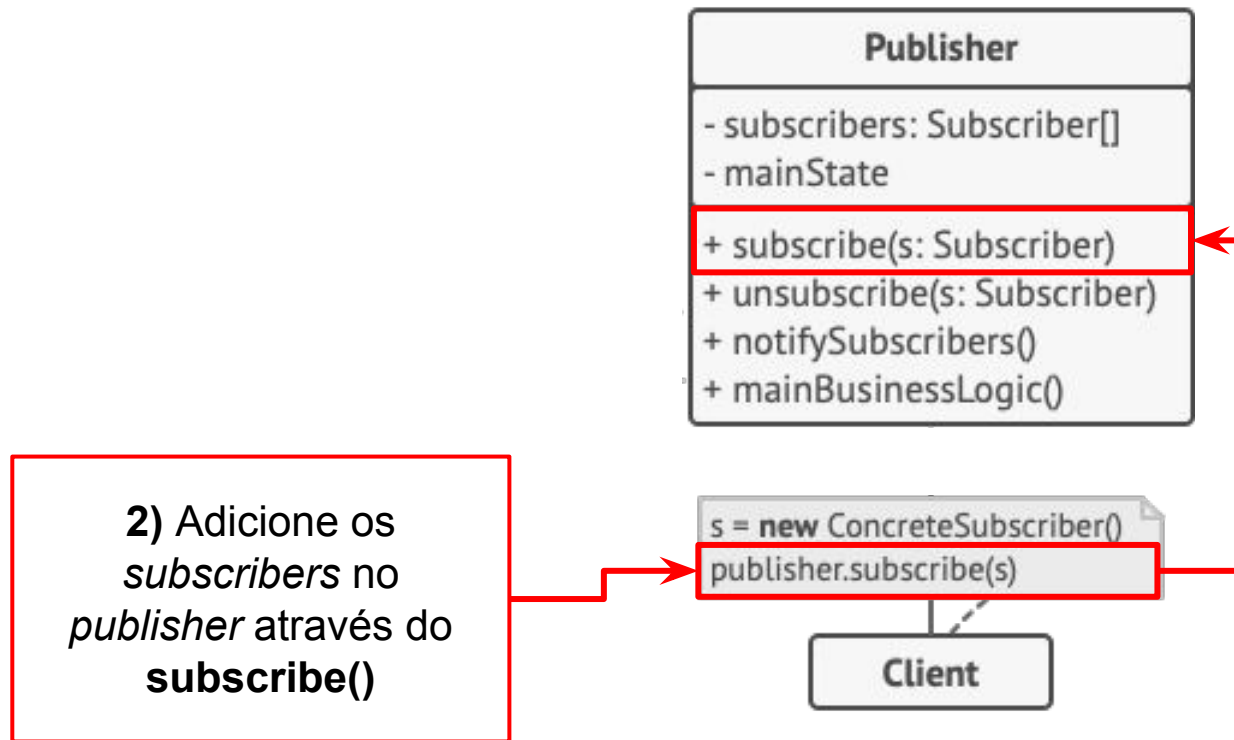


# Padrão Observer - Descrição da Solução



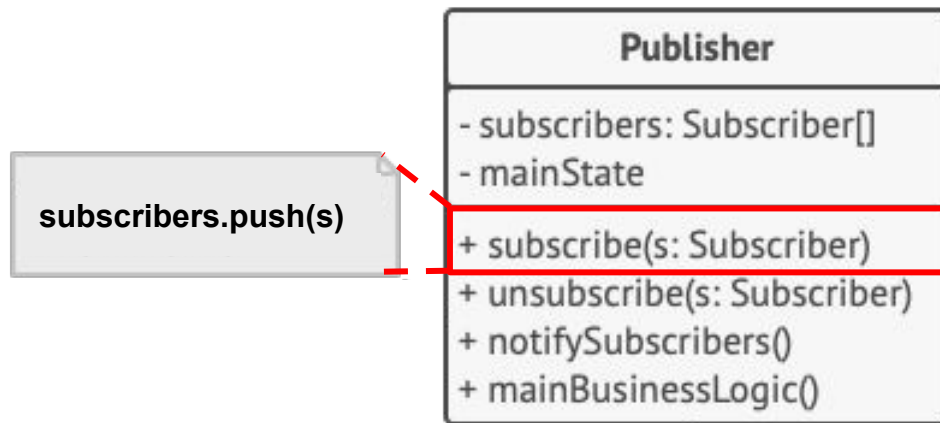


# Padrão Observer - Descrição da Solução

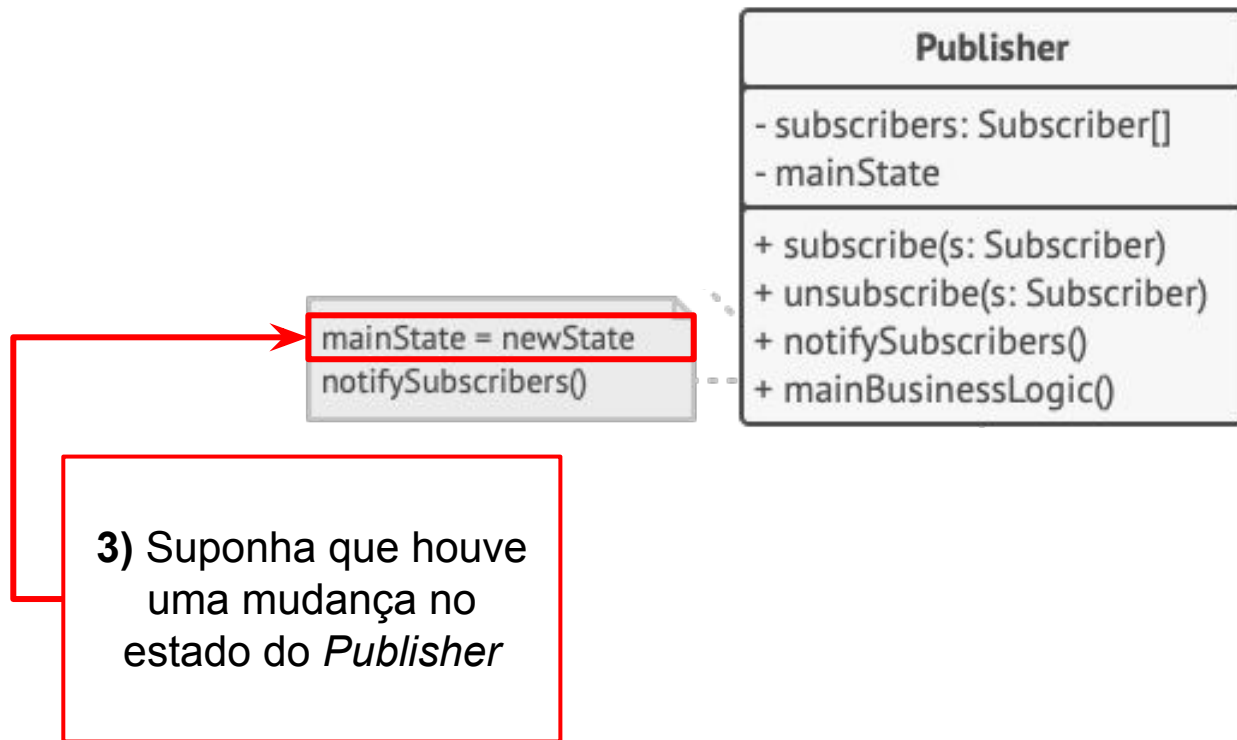


# Padrão Observer - Descrição da Solução

---



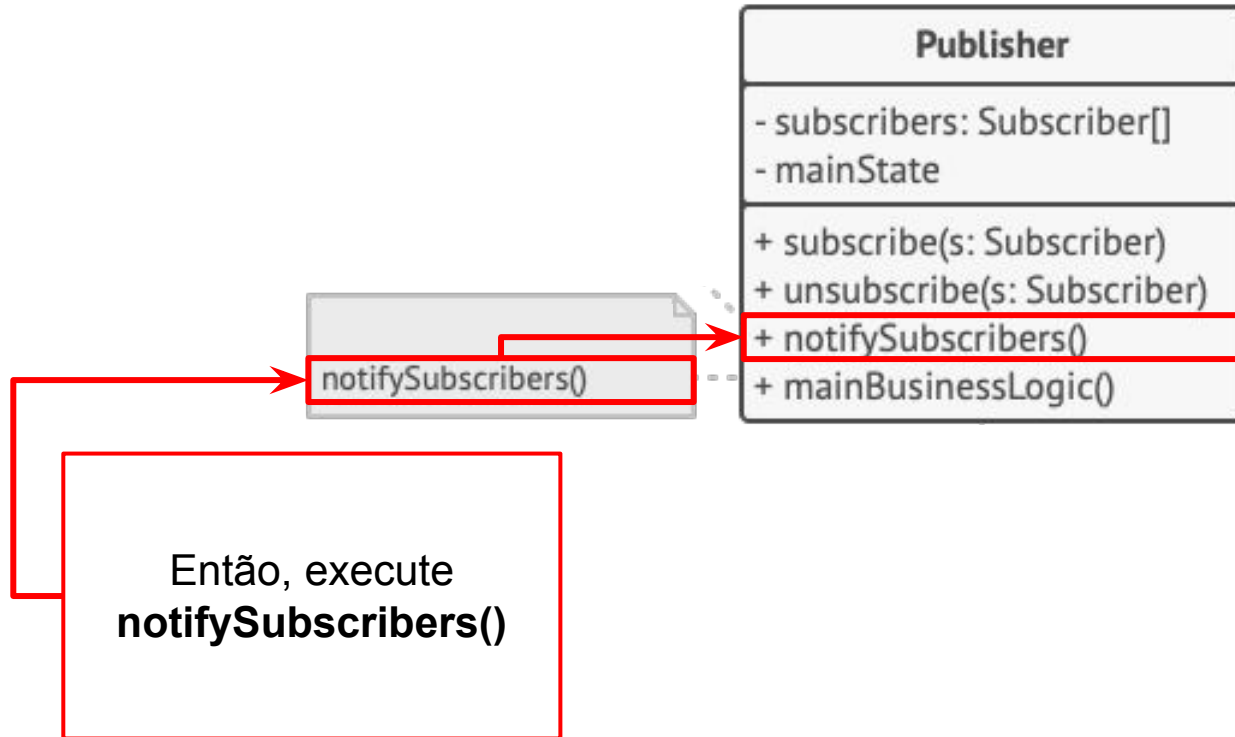
# Padrão Observer - Descrição da Solução



# Padrão Observer - Descrição da Solução



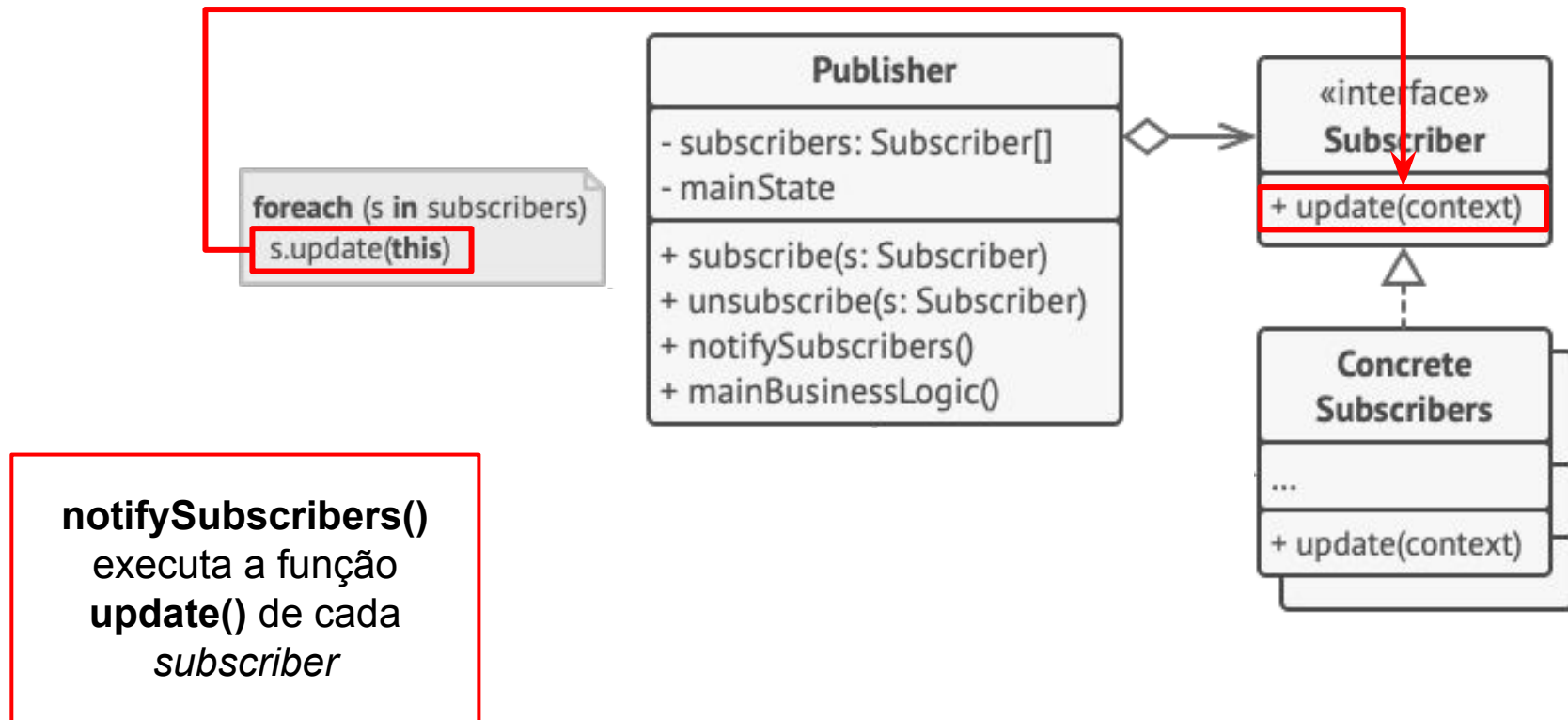
# Padrão Observer - Descrição da Solução



# Padrão Observer - Descrição da Solução



# Padrão Observer - Descrição da Solução



# Exercício

---

Considerando os padrões de projeto, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Um dos objetivos principais de se adotar o padrão Facade em projetos é facilitar a utilização e manutenção de códigos de terceiros, separando a lógica de negócio dos detalhes de implementação desses softwares.

II - O padrão Facade oculta detalhes de implementação e complexidades de subsistemas, criando uma interface simplificada de interação com esses programas.

III - O padrão Facade aumenta a flexibilidade no uso dos subsistemas que ele abstrai.

IV - O padrão Observer permite monitorar o estado de um objeto.

- ☐ Somente I.
- ☐ Somente I, II e IV.
- ☐ Somente II, III e IV.
- ☐ Somente IV.
- ☐ Nenhuma das alternativas anteriores.



# Exercício

---

Considerando os padrões de projeto, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Um dos objetivos principais de se adotar o padrão Facade em projetos é facilitar a utilização e manutenção de códigos de terceiros, separando a lógica de negócio dos detalhes de implementação desses softwares. **V**

II - O padrão Facade oculta detalhes de implementação e complexidades de subsistemas, criando uma interface simplificada de interação com esses programas.

III - O padrão Facade aumenta a flexibilidade no uso dos subsistemas que ele abstrai.

IV - O padrão Observer permite monitorar o estado de um objeto.

- ☐ Somente I.
- ☐ Somente I, II e IV.
- ☐ Somente II, III e IV.
- ☐ Somente IV.
- ☐ Nenhuma das alternativas anteriores.

# Exercício

Considerando os padrões de projeto, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Um dos objetivos principais de se adotar o padrão Facade em projetos é facilitar a utilização e manutenção de códigos de terceiros, separando a lógica de negócio dos detalhes de implementação desses softwares. **V**

II - O padrão Facade oculta detalhes de implementação e complexidades de subsistemas, criando uma interface simplificada de interação com esses programas. **V**

III - O padrão Facade aumenta a flexibilidade no uso dos subsistemas que ele abstrai.

IV - O padrão Observer permite monitorar o estado de um objeto.

- ☐ Somente I.
- ☐ Somente I, II e IV.
- ☐ Somente II, III e IV.
- ☐ Somente IV.
- ☐ Nenhuma das alternativas anteriores.

# Exercício

Considerando os padrões de projeto, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Um dos objetivos principais de se adotar o padrão Facade em projetos é facilitar a utilização e manutenção de códigos de terceiros, separando a lógica de negócio dos detalhes de implementação desses softwares. **V**

II - O padrão Facade oculta detalhes de implementação e complexidades de subsistemas, criando uma interface simplificada de interação com esses programas. **V**

III - O padrão Facade aumenta a flexibilidade no uso dos subsistemas que ele abstrai. **F**

IV - O padrão Observer permite monitorar o estado de um objeto.

- ☐ Somente I.
- ☐ Somente I, II e IV.
- ☐ Somente II, III e IV.
- ☐ Somente IV.
- ☐ Nenhuma das alternativas anteriores.

# Exercício

Considerando os padrões de projeto, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Um dos objetivos principais de se adotar o padrão Facade em projetos é facilitar a utilização e manutenção de códigos de terceiros, separando a lógica de negócio dos detalhes de implementação desses softwares. **V**

II - O padrão Facade oculta detalhes de implementação e complexidades de subsistemas, criando uma interface simplificada de interação com esses programas. **V**

III - O padrão Facade aumenta a flexibilidade no uso dos subsistemas que ele abstrai. **F**

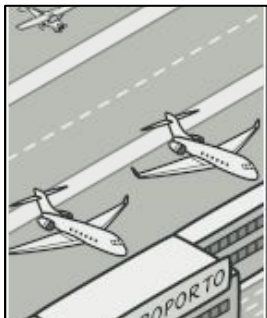
IV - O padrão Observer permite monitorar o estado de um objeto. **V**

- ☐ Somente I.
- ☒ Somente I, II e IV.
- ☐ Somente II, III e IV.
- ☐ Somente IV.
- ☐ Nenhuma das alternativas anteriores.

# Padrão Strategy

## Aplicabilidade:

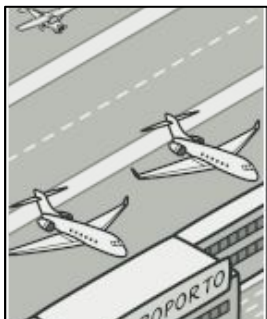
Permite definir uma família de algoritmos, cada um implementado em uma classe separada, todas intercambiáveis (**polimorfismo**)



# Padrão Strategy

## Aplicabilidade:

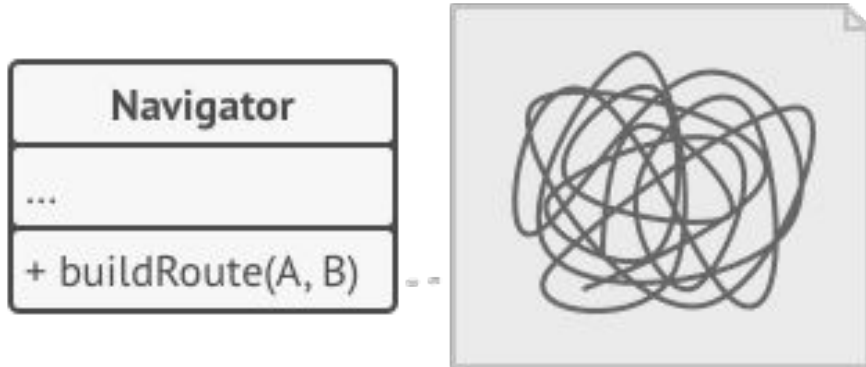
Permite definir uma família de algoritmos, cada um implementado em uma classe separada, todas intercambiáveis (**polimorfismo**)



Isto é, o padrão Strategy é uma implementação do conceito de **Polimorfismo de Orientação a Objetos**

# Porque usar o padrão Strategy?

**Permite isolar comportamentos e algoritmos do restante da classe**  
(reduzindo a chance de causar erros no código da classe ao modificar o algoritmo)



# Padrão Strategy - Descrição da Solução

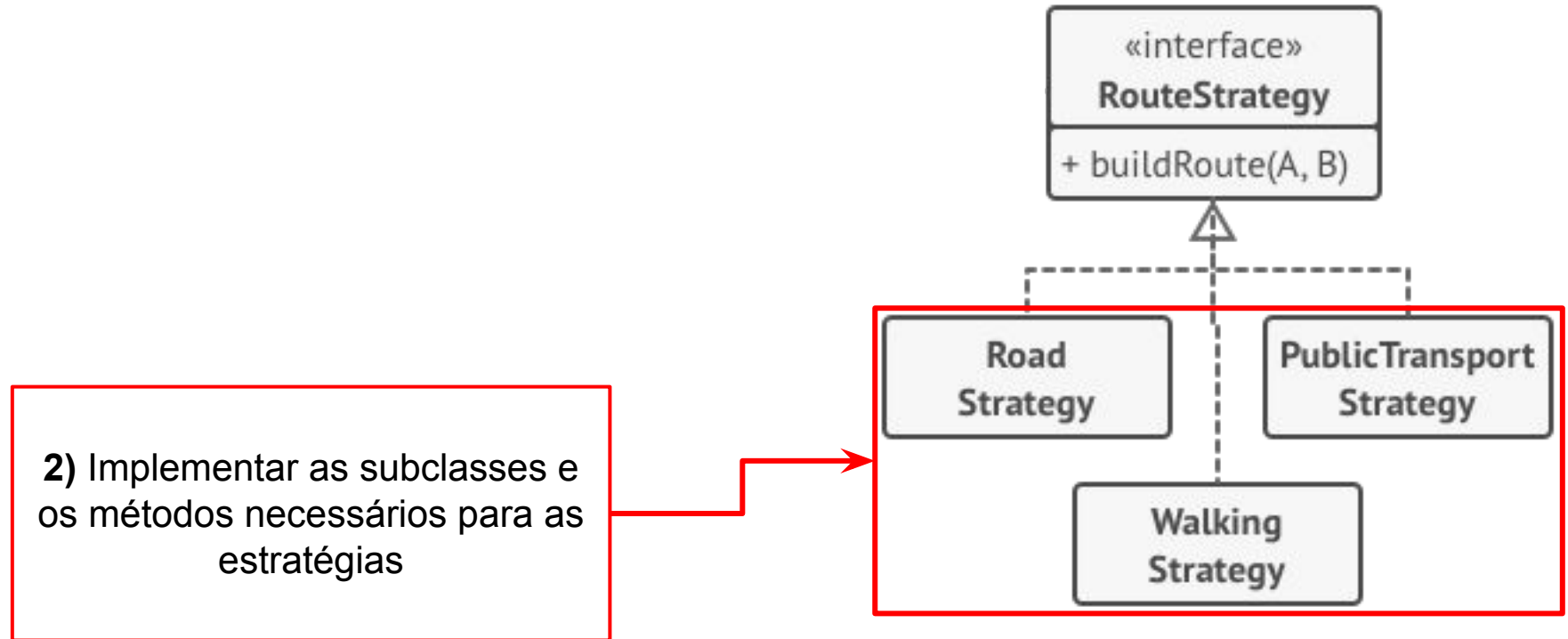
---



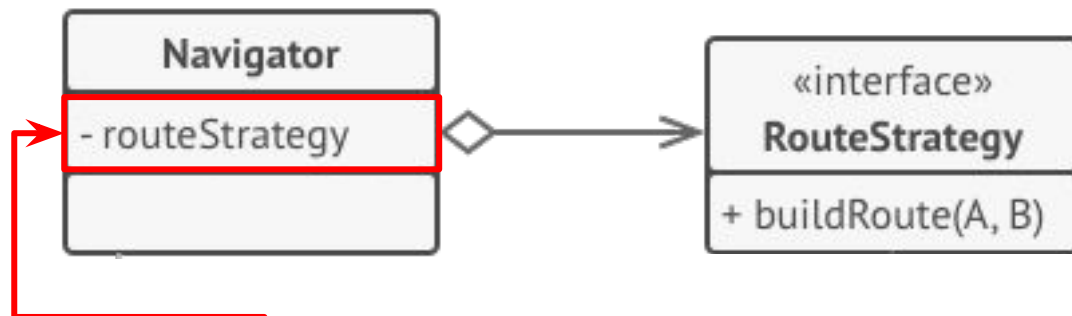
1) Criar a classe base (chamada de **contexto**) que contem os métodos que devem ser implementados nas subclasses



# Padrão Strategy - Descrição da Solução

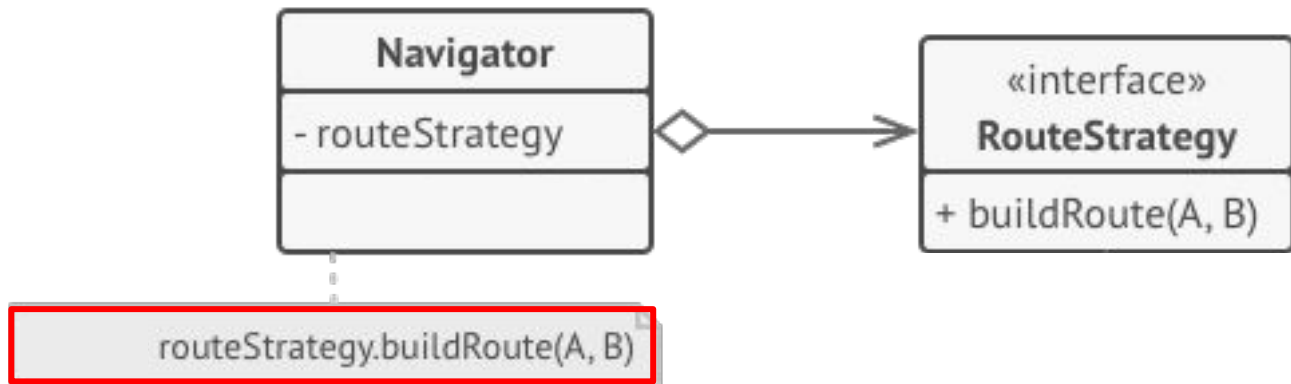


# Padrão Strategy - Descrição da Solução



3) Armazenar uma instância das subclasses para poder usar os métodos implementados nela

# Padrão Strategy - Descrição da Solução



Para usar os métodos implementados pela subclasse, basta chama-los usando a instancia

# Atividade em sala

---

- Em grupo, avaliem quais padrões de projeto podem e devem ser utilizados no projeto
  - Vocês podem utilizar múltiplos padrões simultaneamente, para diferentes subsistemas e componentes do sistema
  - Considerem as vantagens, e limitações de cada padrão
  - Investiguem os demais padrões de projeto que não discutimos em sala
    - Há algum que seja interessante para ser adotado no projeto?

# Referencial Bibliográfico

---

- SOMMERVILLE, Ian. **Engenharia de Software**. 6. ed. São Paulo: Addison-Wesley, 2003.
- PRESSMAN, Roger S. **Engenharia de Software**. São Paulo: Makron Books, 1995.
- JUNIOR, H. E. **Engenharia de Software na Prática**. Novatec, 2010.