Padrão Facade, na prática, com MVC

Kalil de Oliveira
Faculdade Senac Criciúma

Parte 1

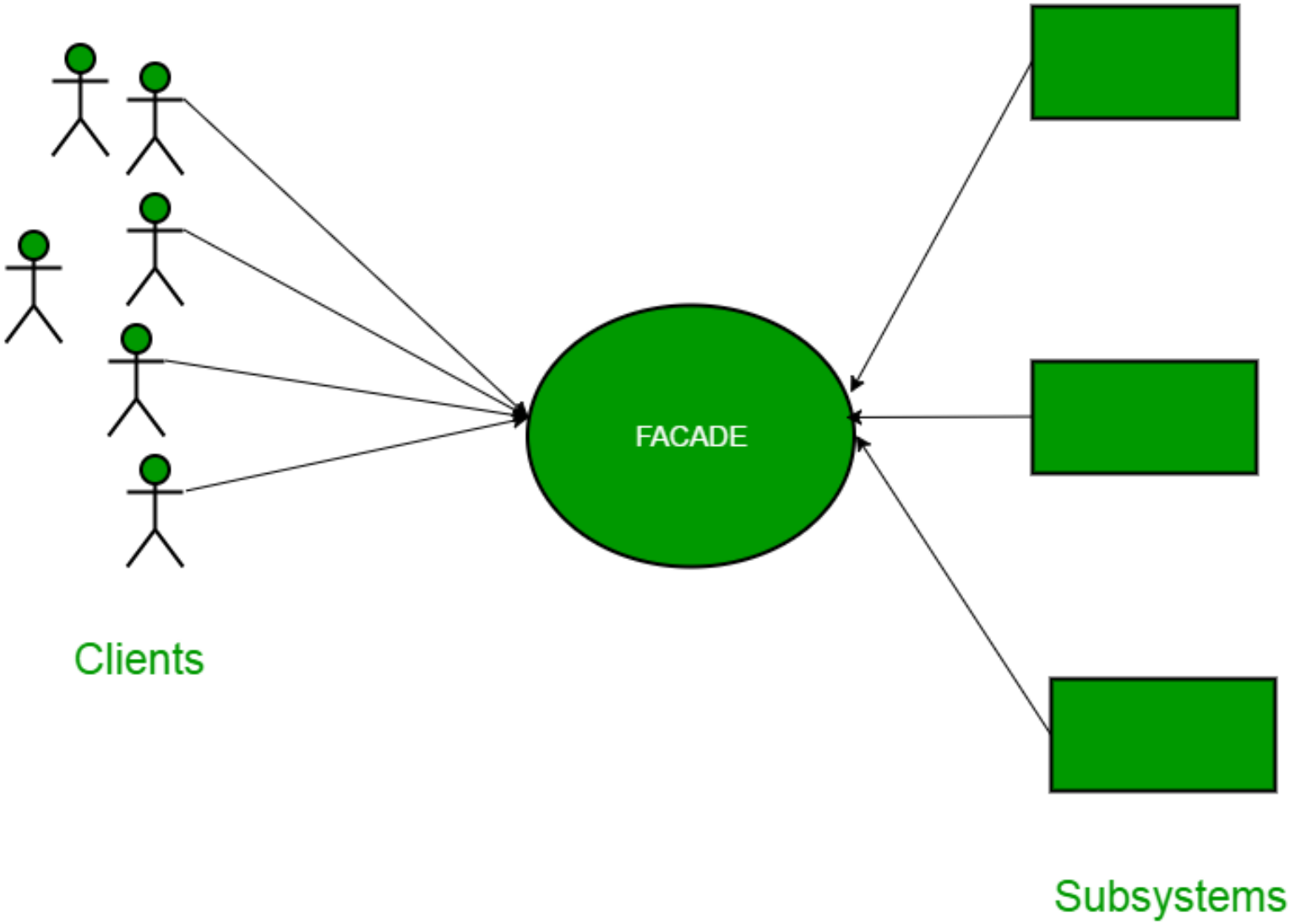
Sobre o Facade

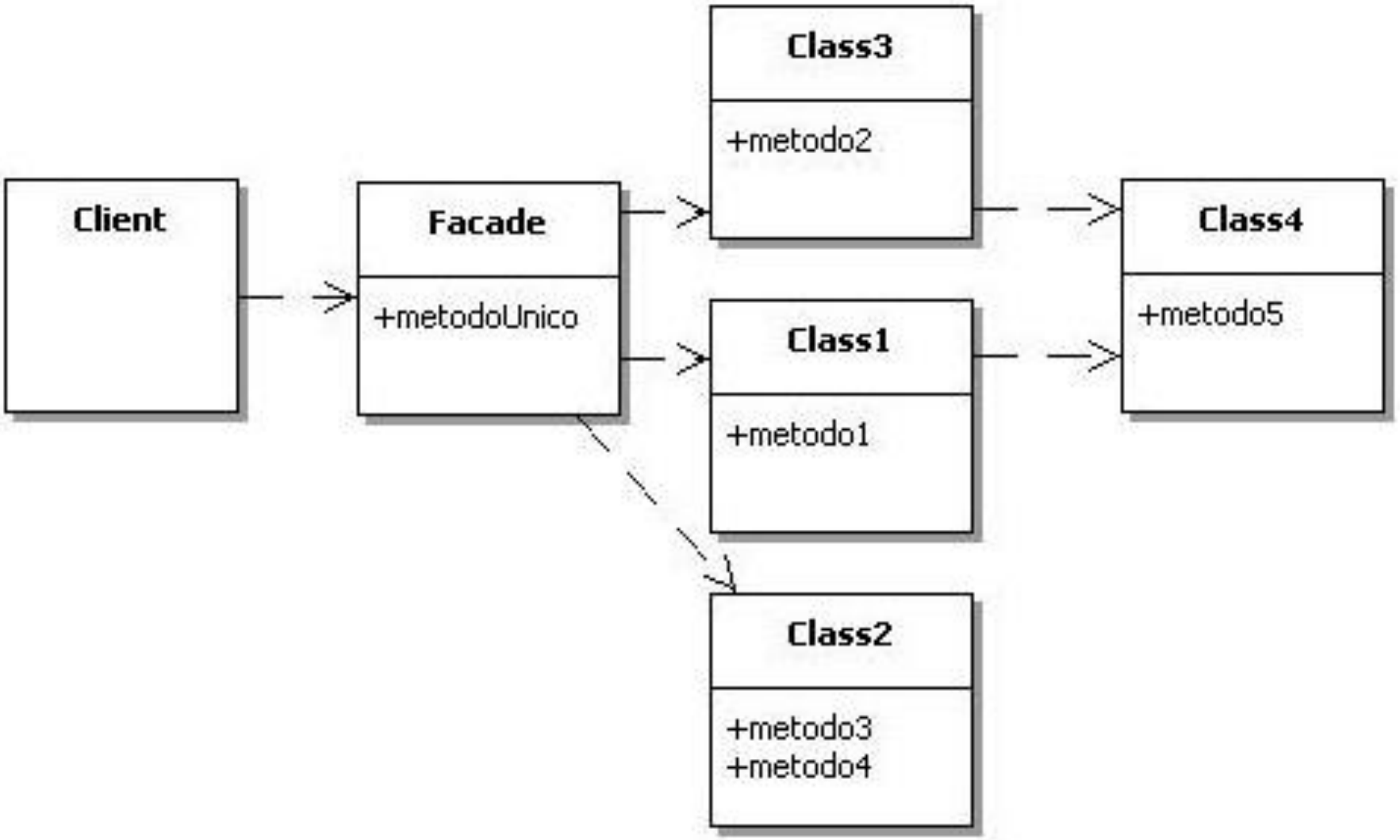
❖ Introdução

- Fornece uma interface unificada para um conjunto de interfaces em um subsistema
- Define uma interface de nível mais alto que torna o subsistema mais fácil de ser usado.

❖ Introdução

- Torna a implementação da classe Client ou Main independente da implementação de vários objetos.
- Pode ser uma alternativa a outros design pattern ou ser usado de maneira integrada, como opção de encapsulamento e organização de código.


FACADE
Clients
Subsystems


Client
Facade
+metodoUnico
Class3
+metodo2
Class1
+metodo1
Class2
+metodo3
+metodo4
Class4
+metodo5

❖ Exemplo clássico: CPU

```
class CPU {
    public void freeze() { ... }
    public void jump(long position) { ... }
    public void execute() { ... }
}

class Memory {
    public void load(long position, byte[] data) { ... }
}

class HardDrive {
    public byte[] read(long lba, int size) { ... }
}
```

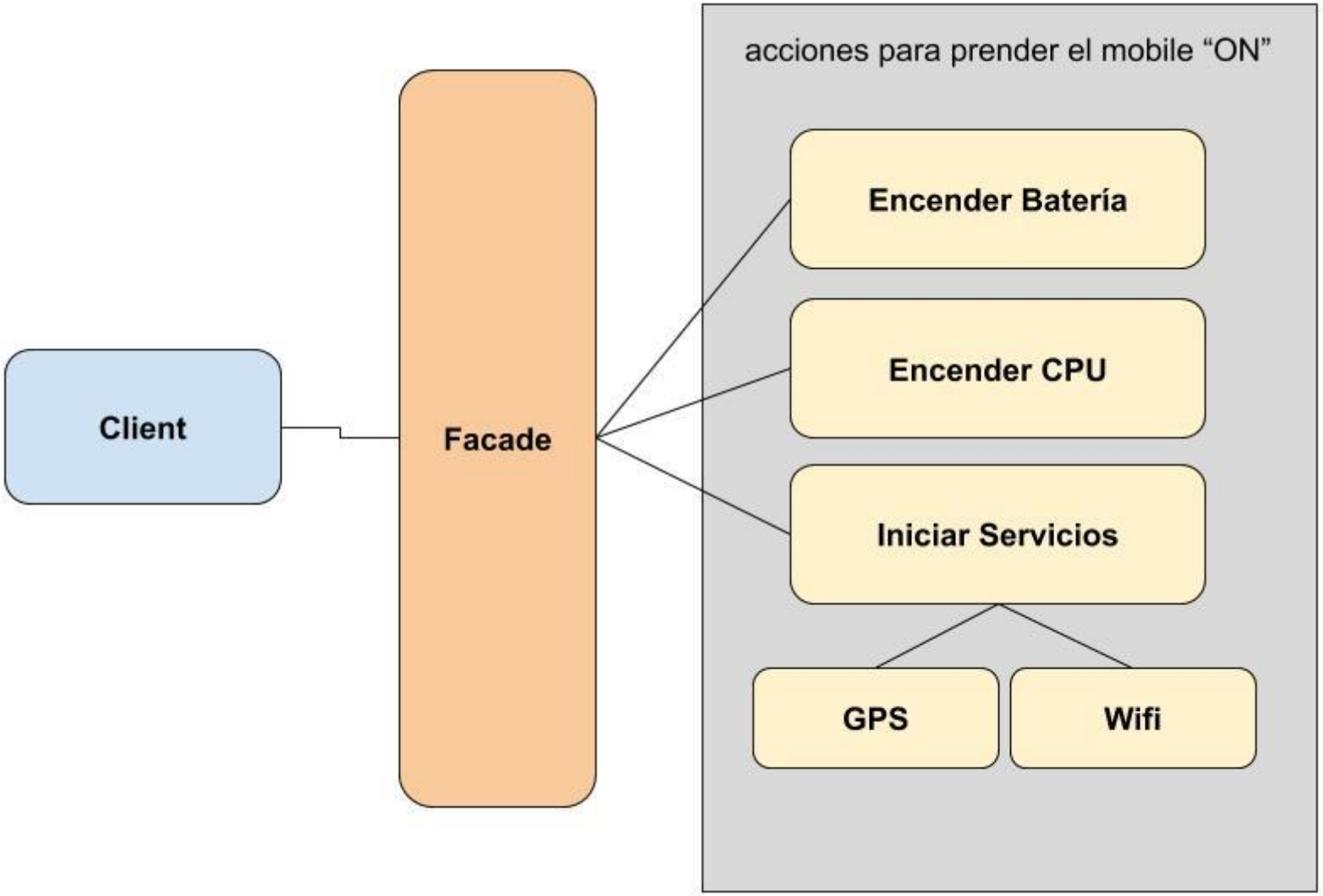
❖ Exemplo clássico: CPU

```
class ComputerFacade {
    private CPU processor;
    private Memory ram;
    private HardDrive hd;

    public ComputerFacade() {
        this.processor = new CPU();
        this.ram = new Memory();
        this.hd = new HardDrive();
    }
```

❖ Exemplo clássico: CPU

```
class You {
     public static void main(String[] args) {
          ComputerFacade computer = new ComputerFacade();
          computer.start();
     }
}
```


acciones para prender el mobile “ON”
Client
Facade
Encender Batería
Encender CPU
Iniciar Servicios
GPS
Wifi

Parte 2

Nosso exemplo

❖ Requisitos

- Em MVC, o sistema será em modo gráfico com duas label, dois textfield e um button para receber nome, sobrenome e enviar a requisição.

❖ A classe NomeModel

```
class NomeModel {

   private String nome, snome;

   public void setNome(String nome) { this.nome = nome; }

  public void setSnome(String snome) { this.snome = snome; }

    public String getCompleto() { return nome + " " + snome;  }}
```

❖ Class NomeView : imports

```
import javax.swing.*;

import java.awt.event.ActionListener;
```

❖ NomeView : atributos

```
class NomeView {

private JTextField txtNome, txtSnome, btnOK;

...
```

❖ NomeView : instanciar a janela

```
public NomeCompletoView() {

JFrame janela = new JFrame(“Bem-vindo");

janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

...
```

❖ NomeView : instanciar um container

```
Jpanel caixa = new Jpanel();

...
```

❖ NomeView : rótulos e caixas de texto

```
JLabel lblNome = new JLabel("Nome: ");

txtNome = new JTextField(20);
```

```
JLabel lblSnome = new JLabel("Sobrenome: ");

txtSnome = new JTextField(20);

...
```

❖ NomeView : botão OK

```
JButton btnOK = new JButton(«OK»);

...
```

❖ NomeView : add elementos ao container

```
caixa.add(lblNome);

caixa.add(txtNome);

caixa.add(lblSnome);

caixa.add(txtSnome);

caixa.add(btnOK);

...
```

❖ NomeView : finalizando a janela

```
janela.add(caixa);

janela.setSize(300, 200);

janela.setVisible(true);

...
```

❖ NomeView : métodos get

```
public String getNome() {
 return txtNome.getText();  }
public String getSnome() {
return txtSnome.getText();   }
...
```

❖ NomeView : evento para o btnOK

```
public void setBtnOKEscuta(ActionListener escuta) {

        btnOk.addActionListener(escuta);

}

...
```

❖ Classe NomeController: atributos

```
class NomeController {

private NomeModel model;

private NomeView view;

...
```

❖ NomeController: construtor

```
public NomeController(NomeModel model, NomeView view)
{    this.model = model;   this.view = view;
        view. setBtnOKEscuta(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
                atualizaView();
                showNome ();            }      });    }
```

❖ NomeController: métodos privados

```
private void atualizaView() {

model.setNome(view.getNome());

model.setSnome(view.getSnome());   }

private void showNome() {

JOptionPane.showMessageDialog(null, “Bem-vindo: " +

model.getCompleto());   }
```

❖ NomeFacade

```
private void atualizaView() {

model.setNome(view.getNome());

model.setSnome(view.getSnome());   }

private void showNome() {

JOptionPane.showMessageDialog(null, “Bem-vindo: " +

model.getCompleto());   }
```

Parte 3

Desafio

❖ Desafio

Imagine que você foi convidado para fazer parte de uma equipe de desenvolvimento de uma nova rede social. Neste caso, basta ampliar o nosso programa e criar novos botões que executam métodos das classes Curtir, Comentar e Compartilhar, retornando um JOptionPane.showMessageDialog(null, “Você curtiu”), conforme o caso. Use o Facade.