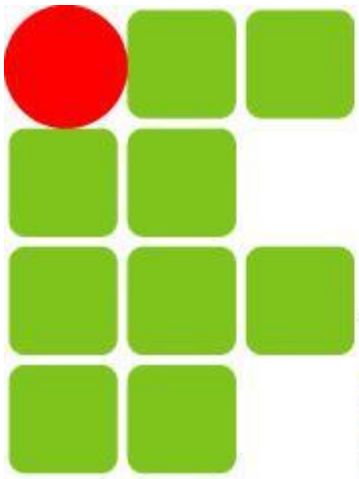


INSTITUTO FEDERAL
ESPÍRITO SANTO
Campus Cachoeiro de Itapemirim

Padrões de Projeto
Design Patterns

A adoção dos padrões terá um efeito profundo e duradouro sobre a forma de escrevermos programas
Ward Cunningham e Ralph Johnson

Programação Orientada a Objetos II

Rafael Vargas Mesquita

http://www.ci.ifes.edu.br
ftp://ftp.ci.ifes.edu.br/informatica/mesquita/

Design Patterns

- Conhecer os princípios OO não faz de você um bom projetista OO
- Bons projetos OO são reutilizáveis, extensíveis e fáceis de manter
- Os padrões mostram como construir um projeto OO com estas qualidades

Design Patterns

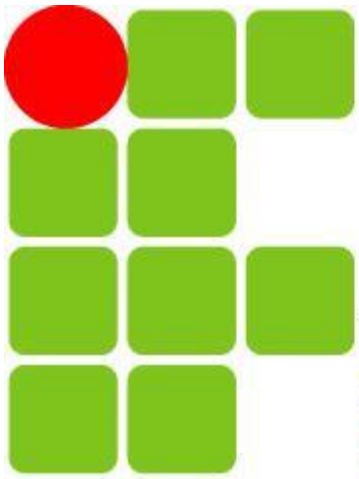
- Os padrões não são uma biblioteca de código. Eles fornecem soluções genéricas para problemas de projeto. Você tem de aplicá-los a seu específico sistema.

- Os padrões não são inventados, eles são descobertos

- A maioria dos padrões permite que parte do sistema varie independentemente de todas as outras partes

Design Patterns

- *Controlador*
- *Dao*
- *Factory*



INSTITUTO FEDERAL
ESPÍRITO SANTO
Campus Cachoeiro de Itapemirim

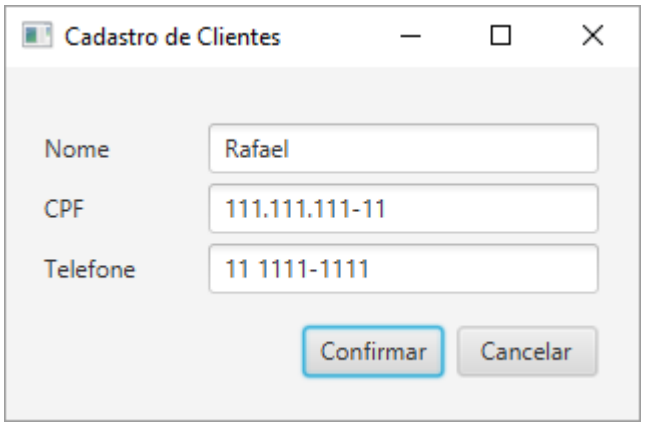
O Controlador

Rafael Vargas Mesquita

http://www.ci.ifes.edu.br
ftp://ftp.ci.ifes.edu.br/informatica/mesquita/

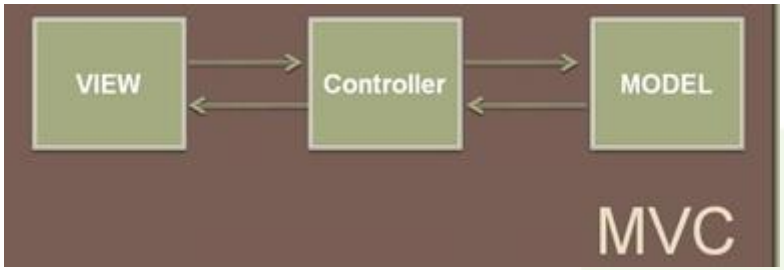
O Controlador

- **Problema**: Que objeto, fora da camada de apresentação, deve receber e coordenar a solicitação da execução de uma operação?



O princípio da separação MVC

- O princípio da separação MVC (Model View Controller) pode ser enunciado em duas partes:
  - Não conecte diretamente objetos pertencentes ao modelo (Model) com objetos pertencentes à interface com o usuário (View).
  - Não coloque lógica da aplicação (tal como o cálculo de impostos) nos métodos dos objetos da interface com o usuário (View)



O princípio da separação MVC

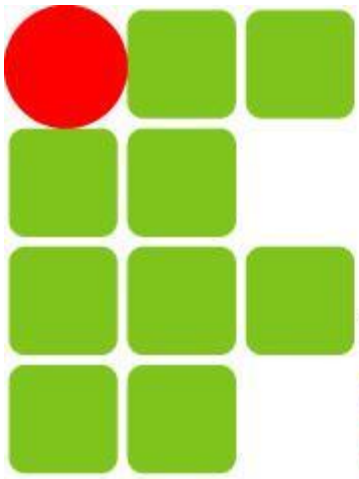
- A motivação para a separação MVC inclui:
  - Permitir o desenvolvimento separado das camadas de apresentação e negócio.
  - Minimizar o impacto na camada de negócio das alterações nos requisitos da interface com o usuário.
    - Exemplo: Migração de um Sistema Desktop para Web

O Objeto Controlador

- O controlador é o primeiro objeto fora da camada de interface com o usuário a receber ou tratar uma mensagem para o sistema.
- Existem duas alternativas possíveis para o objeto controlador:
  - Um objeto Controlador para todo o sistema
  - Um objeto Controlador por Caso de Uso

Resumo das Camadas

COMPONENTES (MVC)	FAZER	CONHECER
Objetos de Fronteira *(View)*	A comunicação homem sistema. Apresentando o modelo num formato adequado ao utilizador, na saída de dados.	As necessidades dos usuários do sistema através da interface homem sistema.
Objetos de Controle *(Controller)*	Recebimento da entrada de dados e iniciar a resposta ao utilizador ao invocar objetos do modelo. Ele também é responsável pela validação e filtragem da entrada de dados, bem como por gerenciar conexão com banco de dados.	Os valores acumulados, temporários ou derivados durante a realização de um caso de uso.
Objetos do Modelo *(Modelo)*	A representação de como o dado é armazenado ou acessado.	Os objetos persistentes que poderão ser utilizados por qualquer sistema.



INSTITUTO FEDERAL
ESPÍRITO SANTO
Campus Cachoeiro de Itapemirim

DAO

Rafael Vargas Mesquita

http://www.ci.ifes.edu.br
ftp://ftp.ci.ifes.edu.br/informatica/mesquita/

O DAO

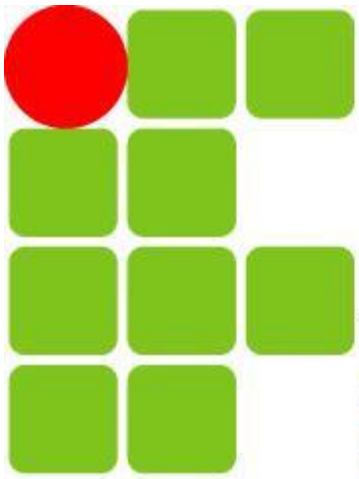
- Objeto de acesso a dados (ou simplesmente DAO, acrônimo de Data Access Object), é um padrão para persistência de dados que permite separar regras de negócio das regras de acesso a banco de dados.
- Numa aplicação que utilize a arquitetura MVC, todas as funcionalidades de bancos de dados, tais como executar comandos SQL, devem ser feitas por classes DAO.

- **Propósito**: prover isolamento da tecnologia de persistência.

Cliente x ClienteDao

```java
public class Cliente {
    private String nome;
    private String cpf;
    private String telefone;

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

    public String getCpf() {
        return cpf;
    }
    public void setCpf(String cpf) {
        this.cpf = cpf;
    }

    public String getTelefone() {
        return telefone;
    }
    public void setTelefone(String telefone) {
        this.telefone = telefone;
    }

}
```

```java
public class ClienteDAO {

    private Connection connection;

    public boolean inserir(Cliente cliente) {
        String sql = "INSERT INTO clientes(nome, cpf, telefone) VALUES(?,?,?)";
        ...
    }

    public boolean alterar(Cliente cliente) {
        String sql = "UPDATE clientes SET nome=?,
        ...           cpf=?, telefone=? WHERE cdCliente=?";
    }

    public boolean remover(Cliente cliente) {
        String sql = "DELETE FROM clientes WHERE cdCliente=?";
        ...
    }

    public List<Cliente> listar() {
        String sql = "SELECT * FROM clientes";
        ...
    }

    public Cliente buscar(Cliente cliente) {
        String sql = "SELECT * FROM clientes WHERE cdCliente=?";
        ...
    }
}
```



INSTITUTO FEDERAL
ESPÍRITO SANTO
Campus Cachoeiro de Itapemirim

Factory

Rafael Vargas Mesquita

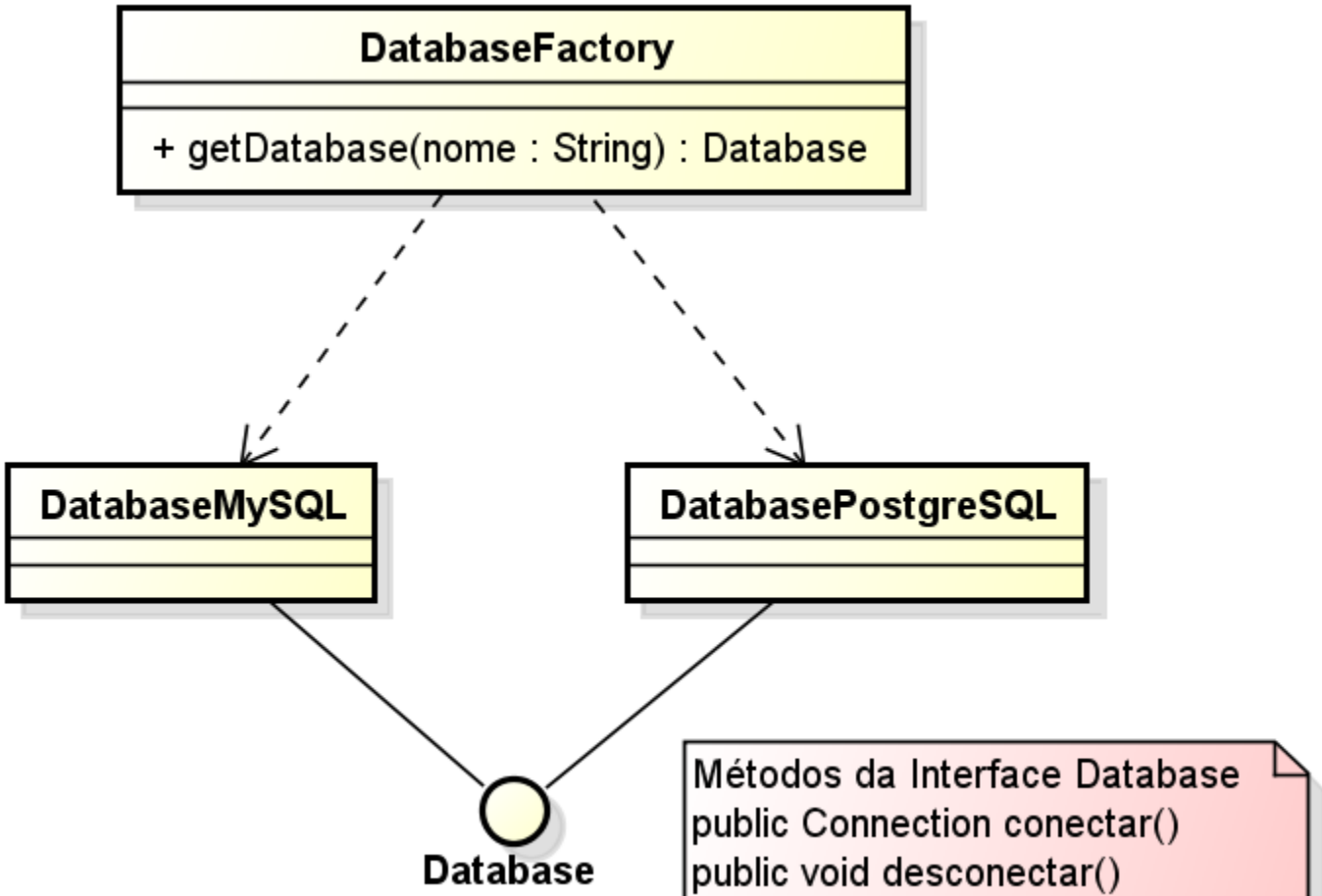
http://www.ci.ifes.edu.br
ftp://ftp.ci.ifes.edu.br/informatica/mesquita/

O Padrão Factory

- **Problema:** suponha que devemos trabalhar em um sistema com diferentes possibilidades de persistência de dados (SGBD).

- **Propósito**: retornar uma instância dentre muitas possíveis classes, dependendo dos dados providos.

Exemplo do Padrão Factory Method



Resumo dos Padrões

PADRÃO	UTILIZAÇÕES NESTA DISCIPLINA
Controlador	Implementar validações Implementar regras de negócio Gerenciar conexão com Banco de Dados
DAO	Implementar scripts de acesso ao Banco de Dados
Factory	Instanciar específica classe de persistência de dados

Visão dos Padrões
Diagrama de Sequência