

FCT/Unesp – Presidente Prudente
Departamento de Matemática e Computação

Programação Orientada a Objetos

Programação em Camadas

Prof. Danilo Medeiros Eler
danilo.eler@unesp.br

Programação em Camadas

- Até o momento, a programação que fizemos não separou as camadas do sistema
- A interface e as regras de negócio estavam todas em uma mesma classe de interface
- A única separação que tínhamos era a parte dos dados

Programação em Camadas

- Exemplo de programação na interface

```
public static void main(String args[]) {
    int opcao;
    String num, entrada;
    double saldo, limite, valor;
    Conta c = null;

    int cont = 0;
    Conta contas[] = new Conta[MAX];

    do {
        opcao = processarMenu();
        switch (opcao) {
            case 1: //Cadastrar Conta
                num = JOptionPane.showInputDialog("Número da Conta")
                entrada = JOptionPane.showInputDialog("Saldo Inicial")
                saldo = Double.parseDouble(entrada);
```

Programação em Camadas

- Exemplo de programação na interface

```
case 1: //Cadastrar Conta
```

```
    num = JOptionPane.showInputDialog("Número da Conta");  
    entrada = JOptionPane.showInputDialog("Saldo Inicial");  
    saldo = Double.parseDouble(entrada);  
    contas[cont] = new Conta(num, saldo);  
    cont++;  
    break;
```

```
case 2: //Cadastrar Conta Especial
```

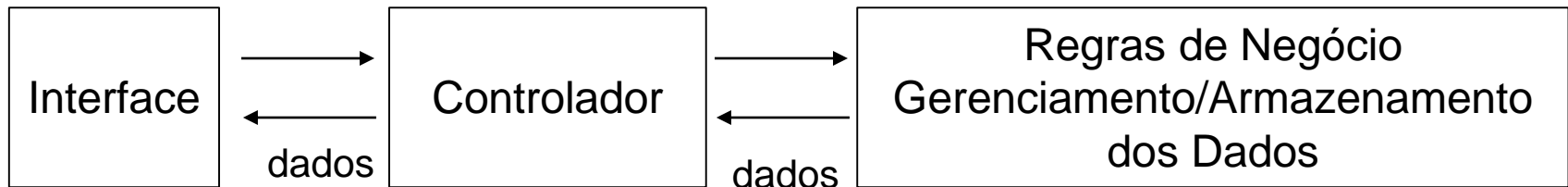
```
    num = JOptionPane.showInputDialog("Número da Conta");  
    entrada = JOptionPane.showInputDialog("Saldo Inicial");  
    saldo = Double.parseDouble(entrada);  
    entrada = JOptionPane.showInputDialog("Limite da Conta");  
    limite = Double.parseDouble(entrada);  
    contas[cont] = new ContaEspecial(num, saldo, limite);  
    cont++;  
    break;
```

Programação em Camadas

- Programação na Interface



- Programação em Camadas

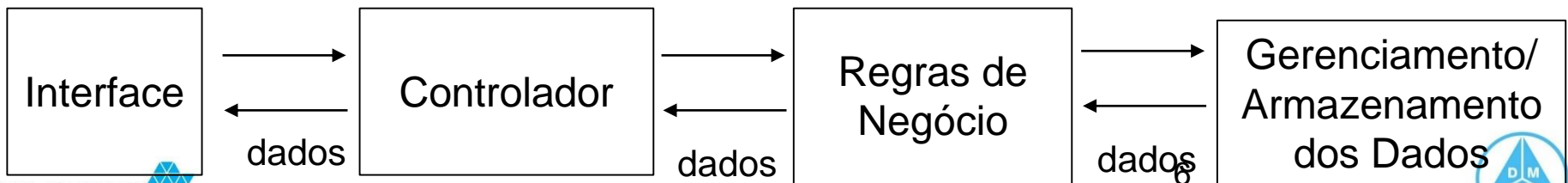
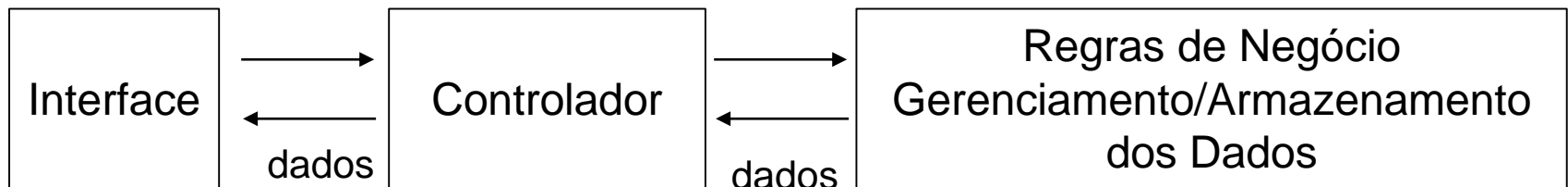


Programação em Camadas

- Programação na Interface



- Programação em Camadas



Padrão Controlador

- A ideia do padrão Controlador é livrar a interface de ter a responsabilidade de trabalhar com os processos e as regras de negócio
 - Para isso, uma classe é criada com essa responsabilidade
 - Os dados são concentrados nas outras classes do modelo
 - A interface cuida apenas da interação com o usuário e entrada e saída de informações

Padrão Controlador

- Interface utilizando o controlador

```
ControladorBanco control = new ControladorBanco();
do {
    opcao = processarMenu();
    switch (opcao) {
        case 1: //Cadastrar Conta
            num = JOptionPane.showInputDialog("Número da Conta");
            entrada = JOptionPane.showInputDialog("Saldo Inicial");
            valor = Double.parseDouble(entrada);
            control.adicionarConta(num, valor);
            break;
        case 2: //Cadastrar Conta Especial
            num = JOptionPane.showInputDialog("Número da Conta");
            entrada = JOptionPane.showInputDialog("Saldo Inicial");
            valor = Double.parseDouble(entrada);
            entrada = JOptionPane.showInputDialog("Limite de Conta");
            limite = Double.parseDouble(entrada);
            control.adicionarContaEspecial(num, valor, limite);
            break;
```


Padrão Controlador

```
public class Banco {  
    private String nome;  
    private Conta contas[];  
    private int cont;  
    private int max;
```

```
public class ControladorBanco {  
    private static Banco banco = new Banco("Banco Teste", 1000);  
    private Conta contaAtual = null;  
    public void adicionarConta(String numero, double saldo) {  
        banco.addConta(numero, saldo);  
    }  
    public void adicionarContaEspecial(String numero, double saldo,  
        banco.addContaEspecial(numero, saldo, limite);  
    }  
    public Conta buscarConta(String numero) {  
        return banco.buscar(numero);  
    }  
    public void setContaAtual(Conta conta) {...3 linhas }  
    public void resetContaAtual() {...3 linhas }  
    public void sacar(double valor) {...5 linhas }  
    public void depositar(double valor) {...5 linhas }
```

Padrão Controlador

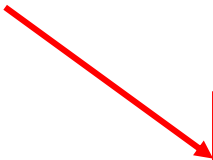
- Interface utilizando o controlador
 - Exemplo de relatório
 - Saída de dados

```
ControladorBanco control = new ControladorBanco();  
String dados = control.getDadosContas();  
System.out.println("RELATÓRIO DE CONTAS");  
System.out.println(dados);
```

Padrão Controlador

- Interface utilizando o controlador
 - Exemplo de Operação
 - Saque

```
Conta c = control.buscarConta(num);  
if (c != null){  
    control.sacar(num, valor);  
}
```



```
public void sacar(String num, double valor){  
    Conta c = buscarConta(num);  
    if (c != null) c.sacar(valor);  
}
```

Referências

- Padrões de Projeto
 - **Livro:** Padrões de Projeto: Soluções reutilizáveis de software orientado a objetos
 - **Autores:** RICHARD HELM, JOHN VLISSIDES, RALPH JOHNSON, ERICH GAMMA
 - **ISBN:** 9788573076103

Bibliografia

BIBLIOGRAFIA BÁSICA

1. SINTES, A., Aprenda programação orientada a objetos em 21 dias, Pearson Education do Brasil, 2002.
2. VAREJÃO, F., Linguagens de programação : Java, C e C++ e outras : conceitos e técnicas, Campus, 2004.
3. DEITEL, H. M., DEITEL, P. J., **Java:** como programar, São Paulo: Pearson Education do Brasil, 2010. 1144p.
4. DEITEL, H. M., DEITEL, P. J., **Java:** como programar, Porto Alegre: Bookman, 2003. 1386p.
5. SAVITCH, W. J., C++ absoluto, Pearson Education : Addison Wesley, 2004.

BIBLIOGRAFIA COMPLEMENTAR

1. BERMAN, A. M. *Data Structures via C++: Objects by Evolution*, Oxford University Press Inc., 1997.
2. BARNES, D.J. & KÖLLING, M., Programação orientada a objetos com Java, Pearson Education : Prentice Hall, 2004.
3. DEITEL, H. M. e DEITEL, P. J. *C++: Como Programar*, Bookman, 2001.
4. GILBERT, R. F. e FOROUZAN, B. A. *Data Structures: A Pseudo Approach with C++*, Brooks/Cole Thomson Learning, 2001.
5. MUSSER, D. R. e SAINI, A. *STL Tutorial and Reference Guide: Programming with the Standard Template Library*, Addison-Wesley, 1996.
6. SEBESTA, R. W. *Conceitos de Linguagem de Programação*, 4ª Ed., Bookman, 2003.
7. SEDGEWICK, R. *Algorithms in C++*, Addison-Wesley, 2002.
8. STROUSTRUP, B. *A Linguagem de Programação C++*, 3ª Ed., Bookman, 2000.

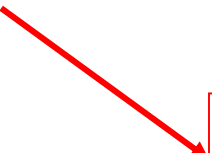
Padrão Controlador

- Nesse exemplo de contas bancárias, haverá um momento em que será necessário sacar ou depositar a partir de uma conta específica
- Portanto, será necessário realizar uma busca para, em seguida, sacar ou depositar na conta recuperada a partir da busca

Padrão Controlador

- Dependendo da implementação, a busca é realizada duas vezes
 - A primeira é para verificar se a Conta existe e a segunda é para recuperar a Conta para efetuar o saque ou depósito

```
Conta c = control.buscarConta(num);  
if (c != null){  
    control.sacar(num, valor);  
}
```



```
public void sacar(String num, double valor){  
    Conta c = buscarConta(num);  
    if (c != null) c.sacar(valor);  
}
```

Padrão Controlador

- Nesse caso, é interessante diminuir o número de buscas para minimizar o tempo de processamento dos métodos
 - Uma solução é especificar a Conta atual para a operação de saque, no caso
 - Uma alternativa comum é utilizar um atributo na classe controladora para especificar a conta atual

Padrão Controlador

```
public class ControladorBanco {  
    private static Banco banco = new Banco("Banco Teste", 1000);  
    private Conta contaAtual = null;  
    public void setContaAtual(Conta conta) {  
        this.contaAtual = conta;  
    }  
    public void resetContaAtual() {  
        contaAtual = null;  
    }  
    public void sacar(double valor) {  
        if (contaAtual != null) {  
            contaAtual.sacar(valor);  
        }  
    }  
    public void depositar(double valor) {  
        if (contaAtual != null) {  
            contaAtual.depositar(valor);  
        }  
    }  
}
```

Padrão Controlador

```
public class ControladorBanco {  
    private static Banco banco = new Banco("Banco Teste", 1000);  
    private Conta contaAtual = null;  
    public void setContaAtual(Conta conta) {  
        this.contaAtual = conta;  
    }  
    public void resetContaAtual() {  
        contaAtual = null;  
    }  
    public void sacar(double valor) {  
        if (contaAtual != null) {  
            contaAtual.sacar(valor);  
        }  
    }  
    public void depositar(double va  
        if (contaAtual != null) {  
            contaAtual.depositar(va  
        }  
    }  
}
```

```
c = control.buscarConta(num);  
if (c != null) {  
    control.setContaAtual(c);  
    control.sacar(valor);  
    control.resetContaAtual();  
}
```

FCT/Unesp – Presidente Prudente
Departamento de Matemática e Computação

Programação Orientada a Objetos

Programação em Camadas

Prof. Danilo Medeiros Eler
danilo.eler@unesp.br