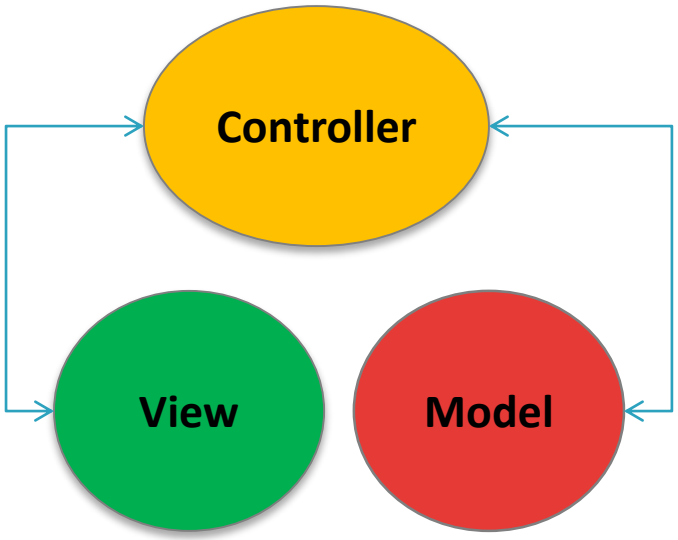
Desenvolvimento Rápido de Aplicações

Padrão MVC

Profa. Joyce Miranda

Padrão MVC

- MVC
  - Fundamentos
    - Padrão Arquitetural de Software
      - Não é um padrão de projeto
    - Dividir a aplicação em camadas com responsabilidades específicas



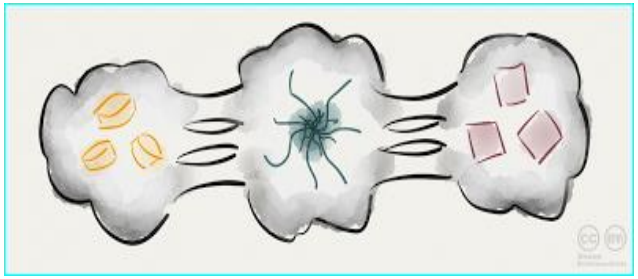
  - Vantagens
    - Legibilidade
    - Facilidade de manutenção
    - Independência maior entre as camadas

Padrão MVC

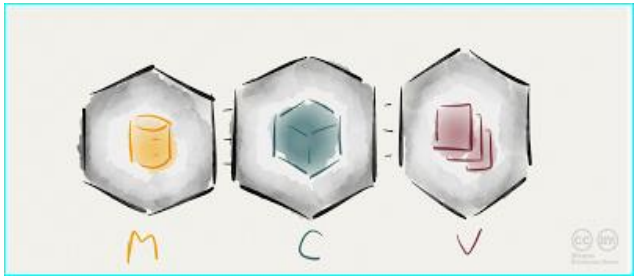
- MVC
  - Fundamentos

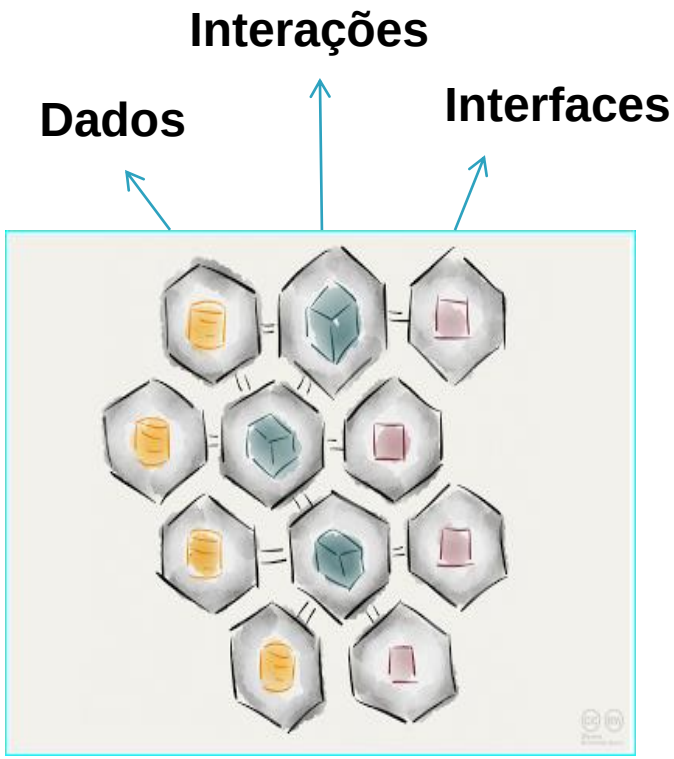
1º

2º



3º

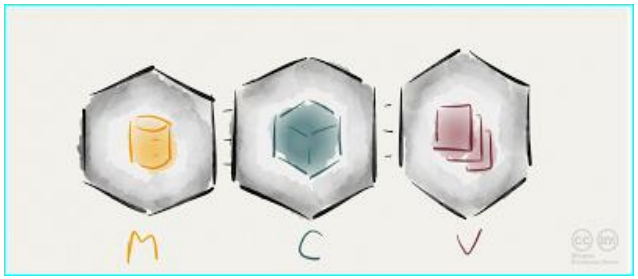




N camadas M, V, C
* existindo regras de interação entre elas

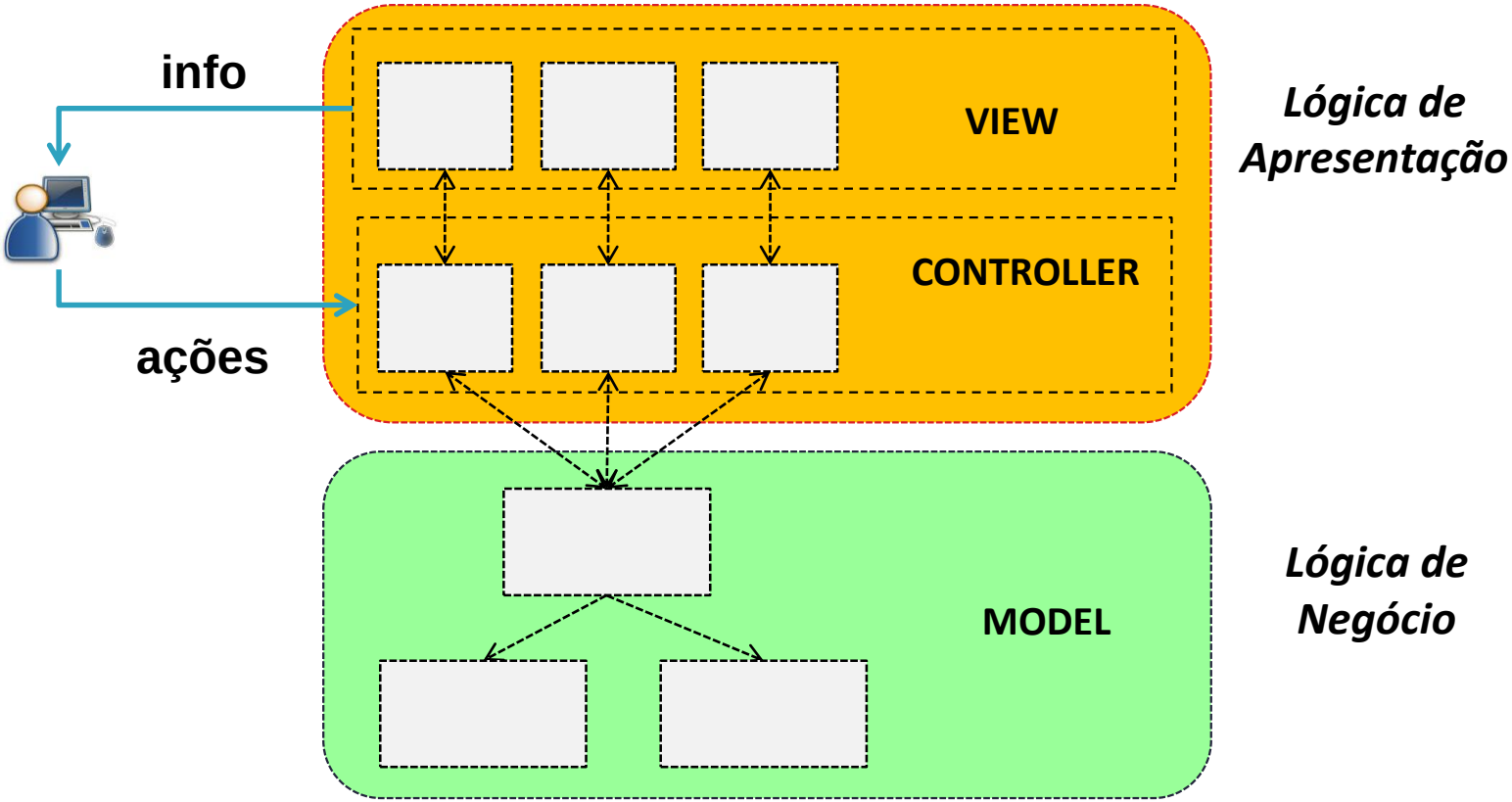
Padrão MVC

- MVC
  - Fundamentos
    - **Regras**
      - A comunicação entre camadas deve ser sempre intermediada pela camada *Controller*
      - Controladores não devem se comunicar entre si.



Padrão MVC

- MVC
  - Fundamentos – Separação de Responsabilidades



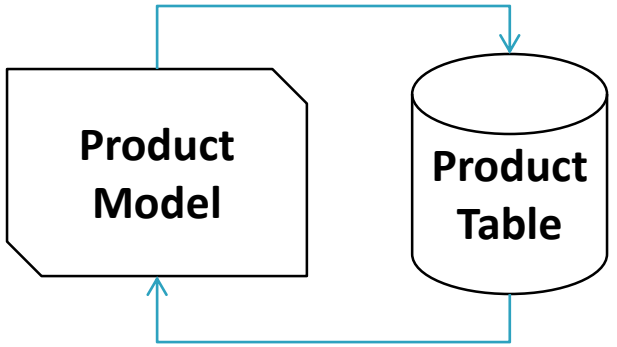
Padrão MVC

- MVC
  - Fundamentos
    - **MODEL**
      - Composta por classes que representam o domínio da aplicação

Regras de Negócio

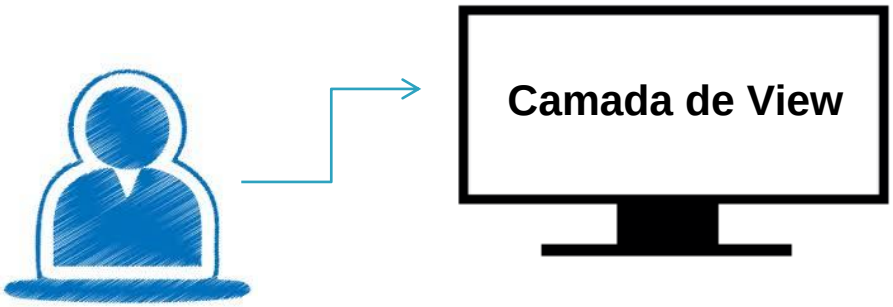
Representação dos Dados

Camada de Acesso a Dados



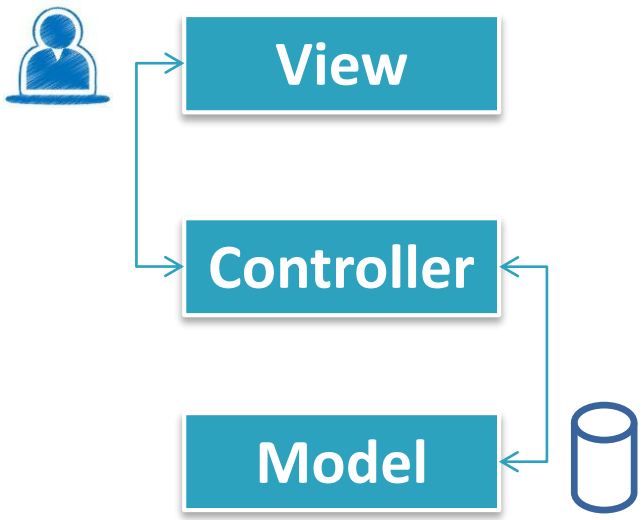
Padrão MVC

- MVC
  - Fundamentos
    - **VIEW**
      - Representa a camada de interface com o usuário
      - Invoca métodos do *Model* por meio do *Controller*
      - Apresenta o resultado dado como resposta a uma requisição
      - Monitora mudanças do *Model* e o apresenta atualizado



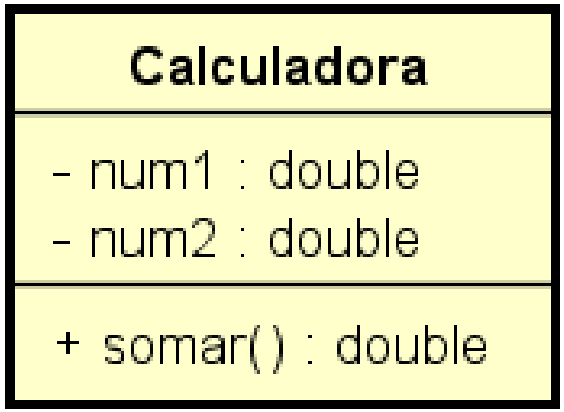
Padrão MVC

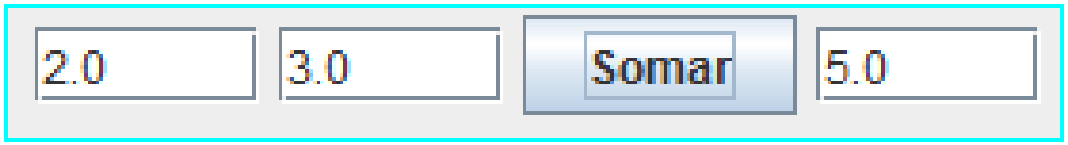
- MVC
  - Fundamentos
    - **CONTROLLER**
      - Processa ações do usuário (capturadas pela View)
      - Apresenta novas 'Views'
      - Atualiza modelo



Padrão MVC

- MVC na prática
  - Estudo de caso: Calculadora

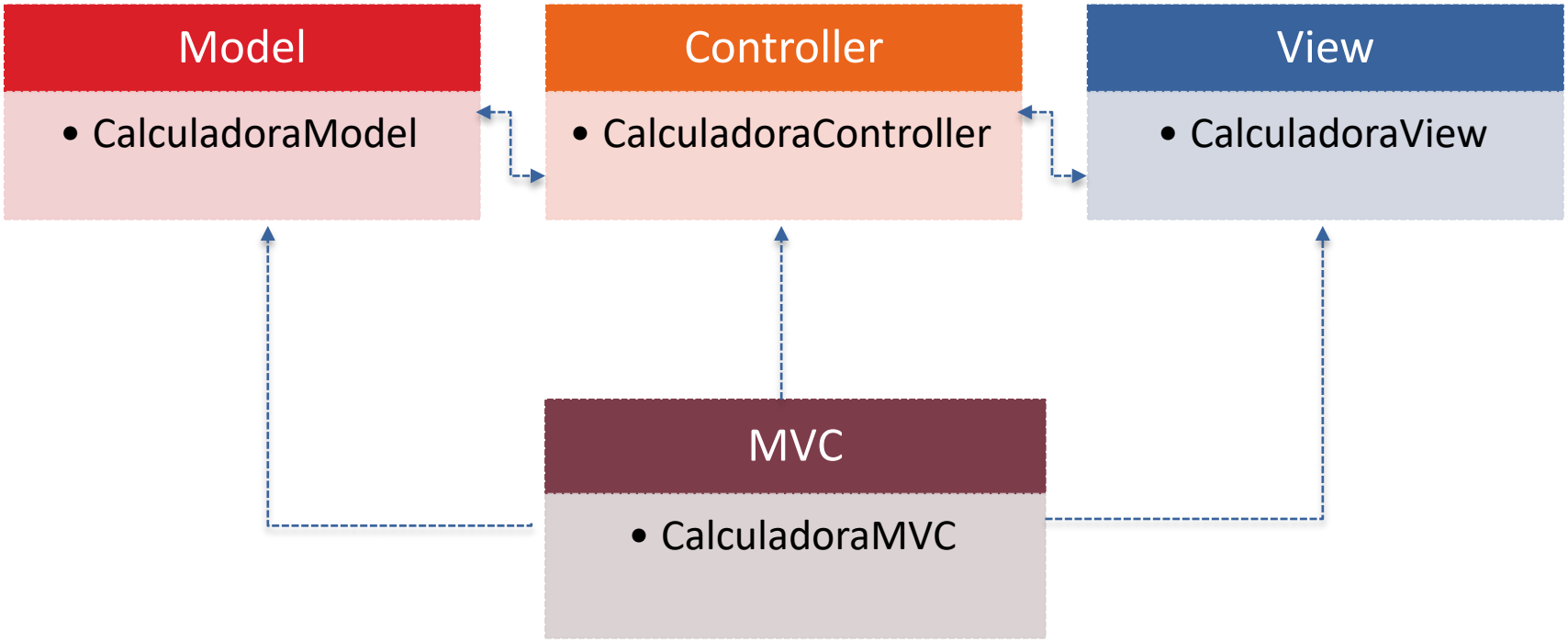




Código disponível em:
https://github.com/joyceMiranda/classCodes/tree/master/DRA_PROJETO_MVC_SWING

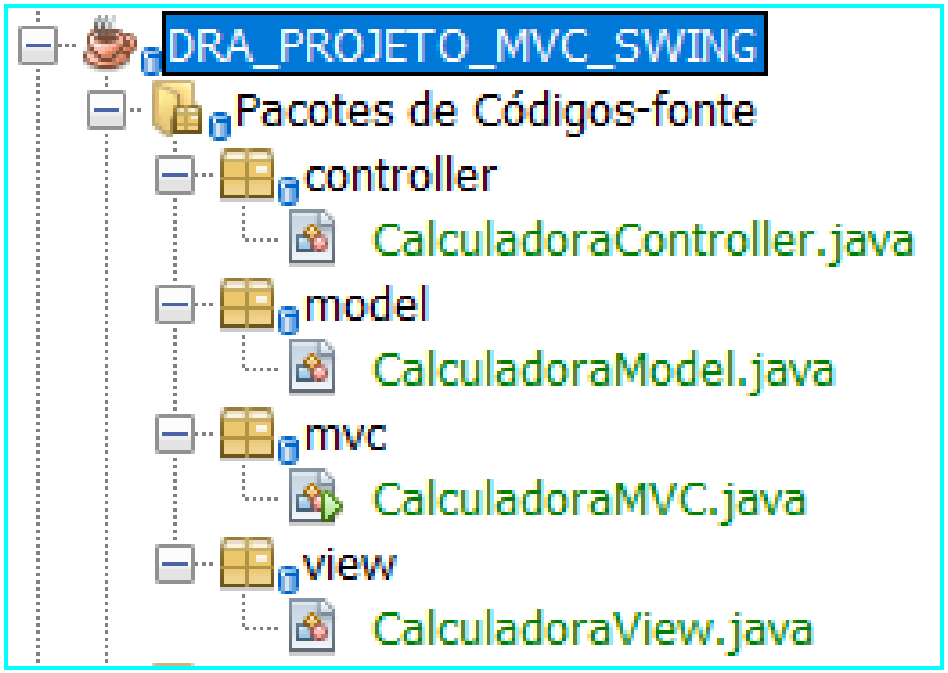
Padrão MVC

- MVC na prática
  - Estudo de caso: Calculadora
  - Estrutura de Classes



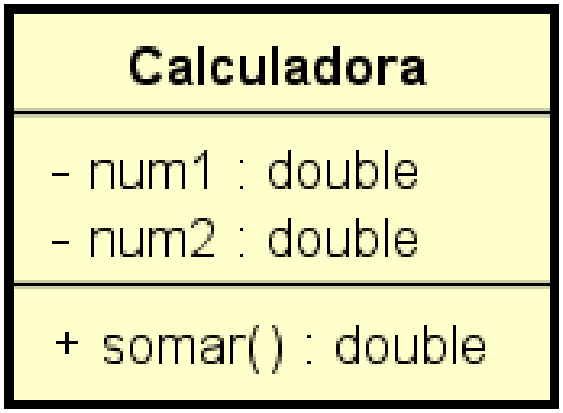
Padrão MVC

- MVC na prática
  - Estudo de caso: Calculadora
  - Estrutura de Pacotes



Padrão MVC

- MVC na prática
  - Estudo de caso: Calculadora
  - Modelo



Implementando elementos do modelo

```
public class CalculadoraModel {

    private double num1;
    private double num2;

    public double getNum1() {
        return this.num1;
    }

    public double getNum2() {
        return this.num2;
    }

    public void setNum1(double num1){
        this.num1 = num1;
    }

    public void setNum2(double num2){
        this.num2 = num2;
    }

    public double somar(){
        return num1 + num2;
    }

}
```

Padrão MVC

- MVC na prática
  - Estudo de caso: Calculadora
  - View

Definindo elemento de interface

```
public class CalculadoraView extends JFrame {

    JPanel painel = new JPanel();
    JTextField txtNum1 = new JTextField(5);
    JTextField txtNum2 = new JTextField(5);
    JTextField txtResultado = new JTextField(5);
    JButton btnSomar = new JButton("Somar");

    public CalculadoraView(){
        this.setSize(400, 200);
        painel.add(txtNum1);
        painel.add(txtNum2);
        painel.add(btnSomar);
        painel.add(txtResultado);
        this.add(painel);
    }
```

Padrão MVC

- MVC na prática
  - Estudo de caso: Calculadora
  - View (cont.)

Gets para campos de entrada
Sets para campos de saída

```
public String getTxtNum1(){
    return txtNum1.getText();
}

public String getTxtNum2(){
    return txtNum2.getText();
}

public void setResultado(String resultado){
    txtResultado.setText(resultado);
}
```

Padrão MVC

- MVC na prática
  - Estudo de caso: Calculadora
  - View (cont.)

Adicionando ação aos elementos da tela

```
public void addBtnSomarListener(ActionListener listenForBtnSomar){
    btnSomar.addActionListener(listenForBtnSomar);
}

public void displayMensagemDeErro(String msg){
    JOptionPane.showMessageDialog(this, msg);
}
```

Padrão MVC

- MVC na prática
  - Estudo de caso: Calculadora
  - Controller

Linkando ação do botão

```
public class CalculadoraController {

    private CalculadoraView theView;
    private CalculadoraModel theModel;

    public CalculadoraController(CalculadoraView theView,
                                 CalculadoraModel theModel) {
        this.theView = theView;
        this.theModel = theModel;

        this.theView.addBtnSomarListener(new SomarListener());
    }
```

Padrão MVC

Implementando ação do botão

- MVC na prática
  - Controller (cont.)

```
class SomarListener implements ActionListener{

    @Override
    public void actionPerformed(ActionEvent e) {
        try{

            double num1 = Double.parseDouble(theView.getTxtNum1());
            double num2 = Double.parseDouble(theView.getTxtNum2());

            theModel.setNum1(num1);
            theModel.setNum2(num2);
            double resultado = theModel.somar();

            theView.setResultado(Double.toString(resultado));

        }catch(NumberFormatException ex){
            theView.displayMensagemDeErro(
                    "Entre com dos valores numéricos!!");
        }
    }
}
```

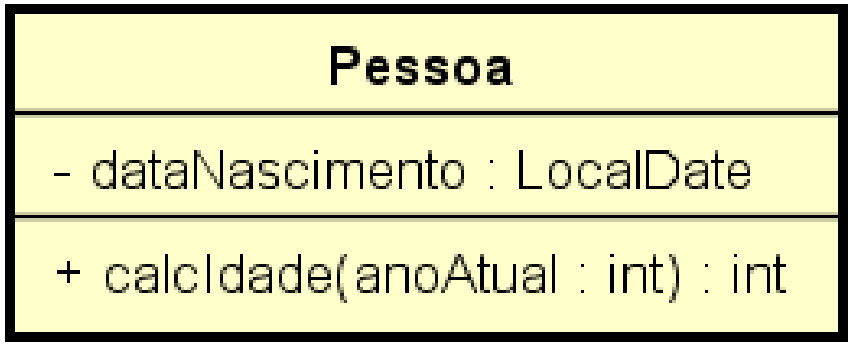
Padrão MVC

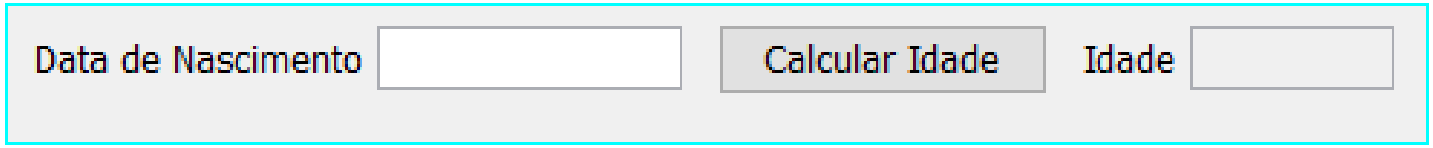
- MVC na prática
  - Estudo de caso: Calculadora
  - Classe MVC - Sincronizadora

```
public class CalculadoraMVC {

    public static void main(String args[]){

        CalculadoraView theView = new CalculadoraView();
        CalculadoraModel theModel = new CalculadoraModel();
        CalculadoraController theController =
                new CalculadoraController(theView, theModel);

        theView.setVisible(true);
    }

}
```

Padrão MVC

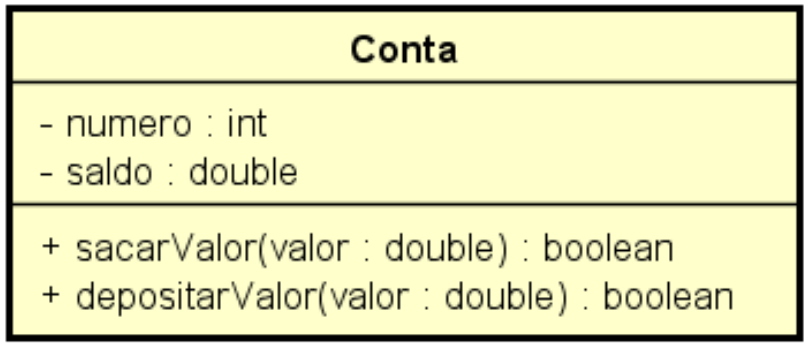
- Praticando...
  - Utilizando o padrão MVC
    - Crie uma interface responsável por receber a data de nascimento de uma pessoa e exibir sua idade.

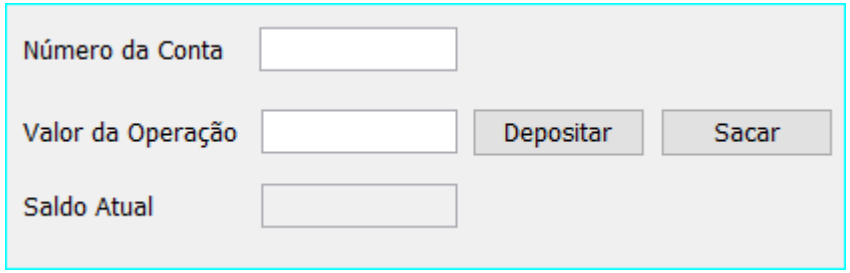




Padrão MVC

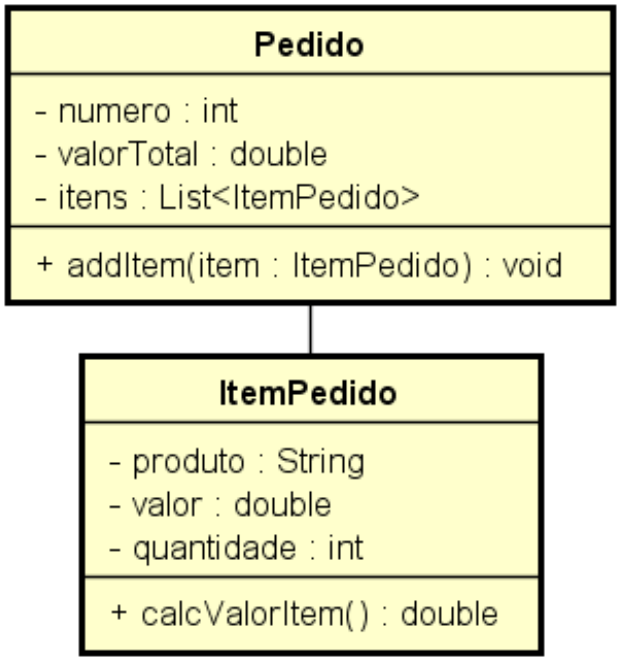
- Praticando...
  - Utilizando o padrão MVC
    - Crie uma interface responsável por receber o número de uma conta e fornecer as funcionalidades de depósito e saque.
      - Regra de Negócio: O saldo só poderá ser modificado mediante as operações de saque e depósito.

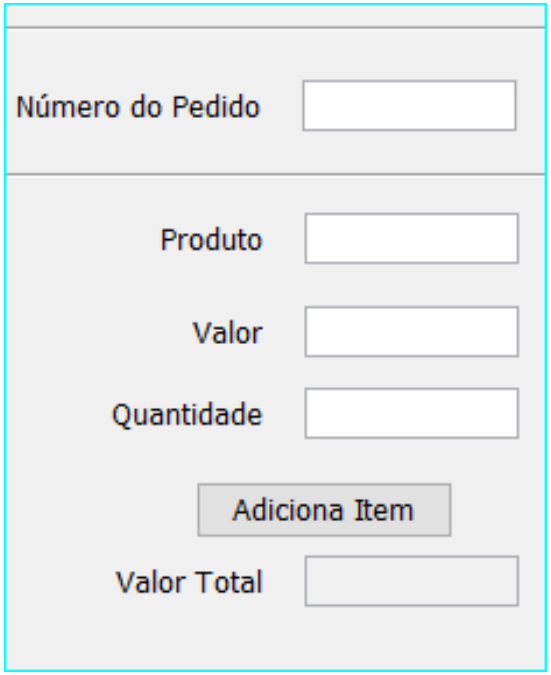




Padrão MVC

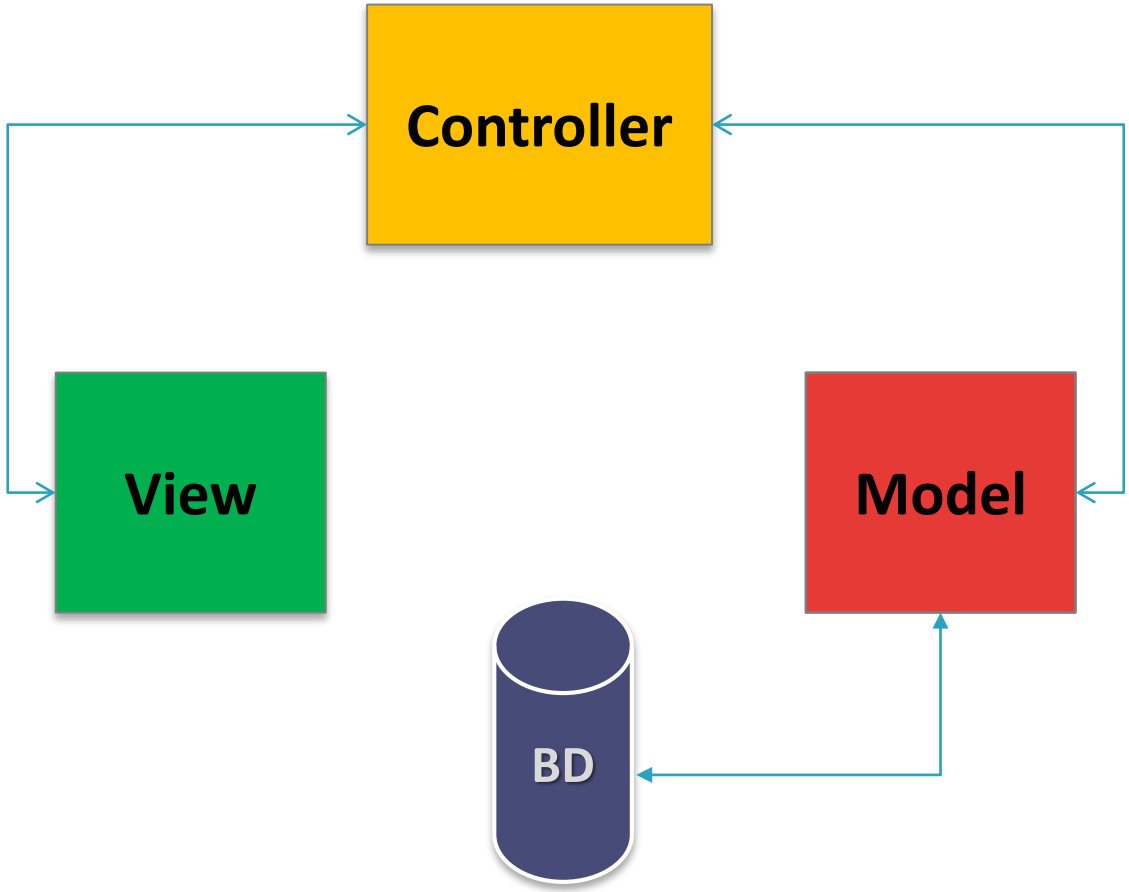
- Tarefa de Implementação
  - Utilizando o padrão MVC
    - Crie uma interface responsável por registrar os itens de um pedido e atualizar o valor total do pedido.





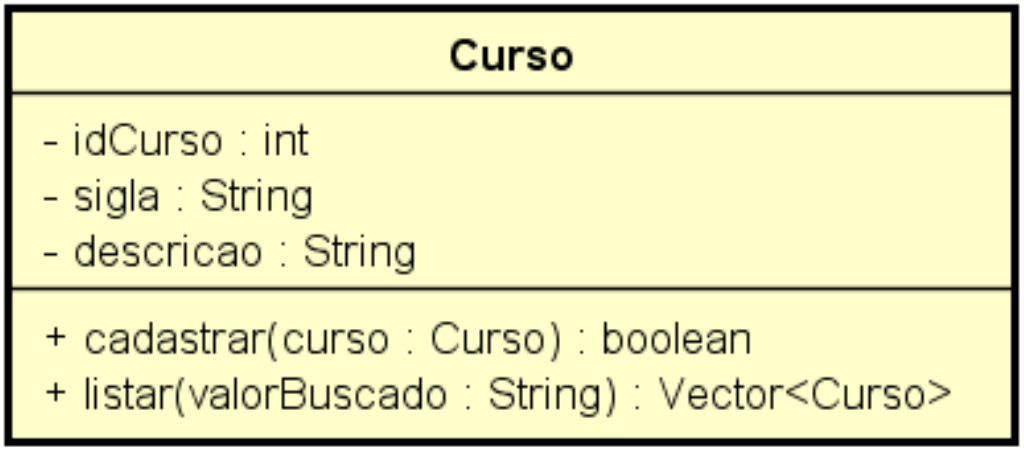
Padrão MVC

- MVC com acesso ao BD



Padrão MVC

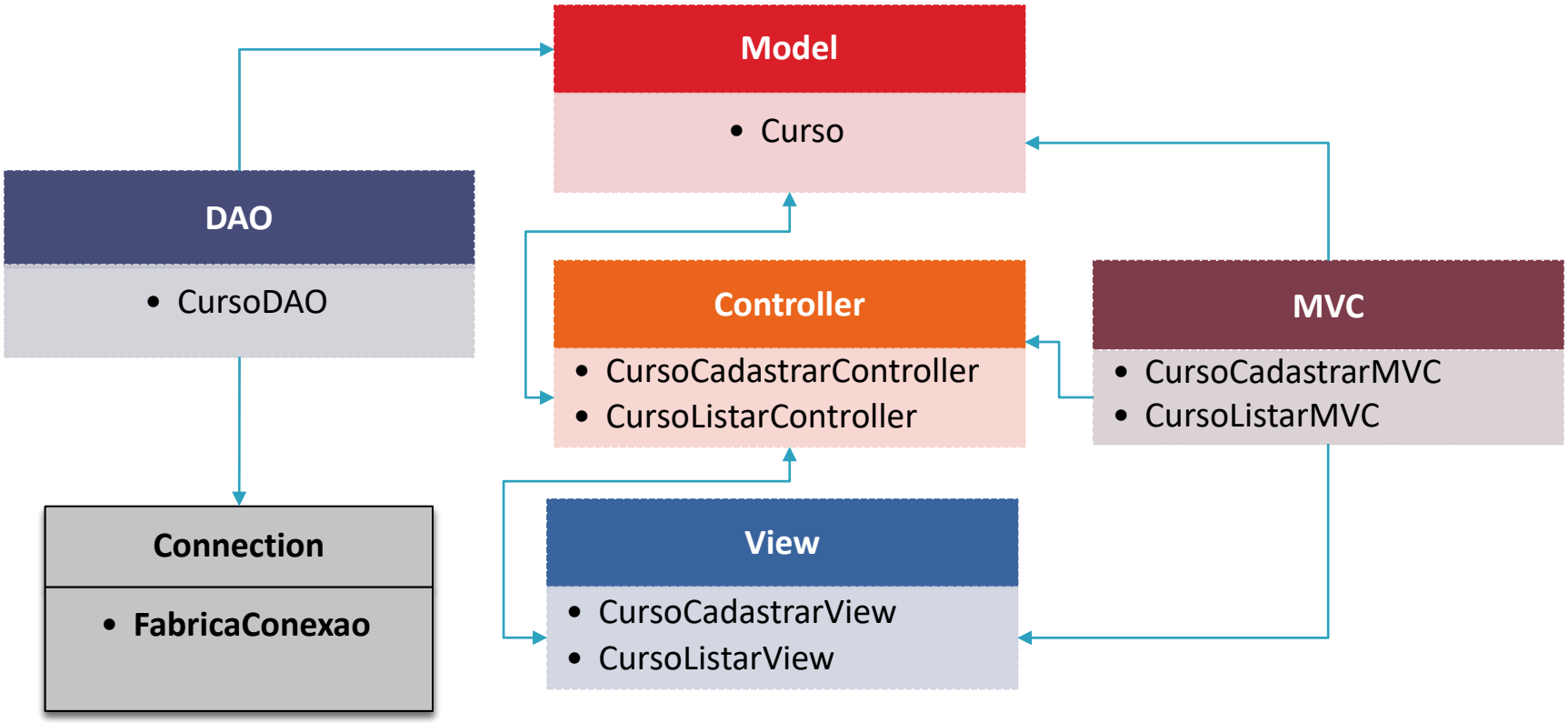
- MVC na prática
  - Estudo de caso: Curso



Código disponível em:
https://github.com/joyceMiranda/classCodes/tree/master/DRA_PROJETO_MVC_SWING

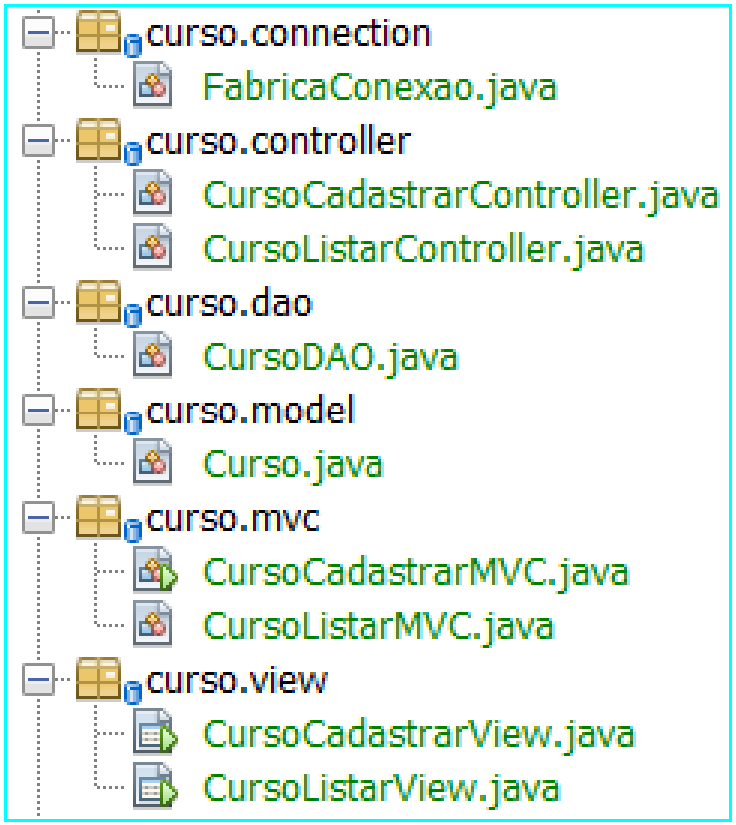
Padrão MVC

- MVC na prática
  - Estrutura de Classes

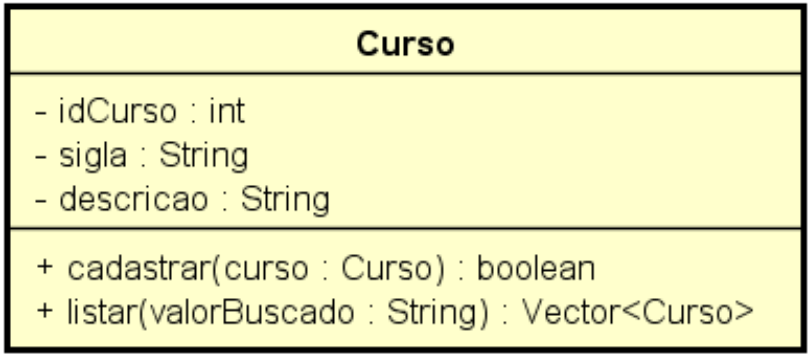


Padrão MVC

- MVC na prática
  - Estrutura de Pacotes



Padrão MVC



- MVC na prática
  - Base de Dados
    - MySQL
    - Workbench

```
--
-- Database: 'sysControleAcademico'
--
CREATE DATABASE sysControleAcademico;

use sysControleAcademico;
-- --------------------------------------


--
-- Table structure for table 'curso'
--
CREATE TABLE "curso" (
  "idcurso" int(11) NOT NULL AUTO_INCREMENT,
  "sigla" varchar(15) NOT NULL,
  "descricao" varchar(200) NOT NULL,
  PRIMARY KEY ("idcurso")
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Padrão MVC

- MVC na prática
  - FabricaConexao – Conexão com JDBC
    - Necessário incluir o driver de conexão ao projeto.

```
public class FabricaConexao {

    public static Connection getConnection(){
        try{
            String host = "jdbc:mysql://localhost/sysControleAcademico";
            String user = "root";
            String password = "";
            return DriverManager.getConnection(
                    host, user, password);

        }catch(SQLException e){
            throw new RuntimeException(e);
        }
    }
}
```

Padrão MVC

Curso
- idCurso : int - sigla : String - descricao : String
+ cadastrar(curso : Curso) : boolean + listar(valorBuscado : String) : Vector<Curso>

- MVC na prática
  - Curso (Model)

```
public class Curso {

    private int idCurso;
    private String sigla;
    private String descricao;

    public Curso() {...2 linhas }
    public Curso(int idCurso, String sigla, String descricao) {...5 linhas }

    public int getIdCurso() {...3 linhas }
    public void setIdCurso(int idCurso) {...3 linhas }

    public String getSigla() {...3 linhas }
    public void setSigla(String sigla) {...3 linhas }

    public String getDescricao() {...3 linhas }
    public void setDescricao(String descricao) {...3 linhas }
}
```

Padrão MVC

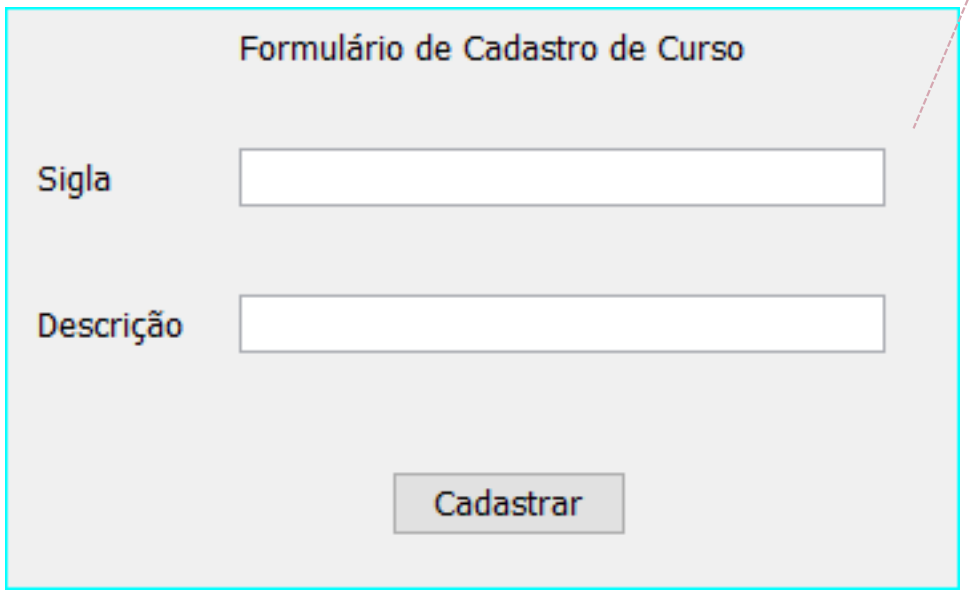
Curso
- idCurso : int - sigla : String - descricao : String
+ cadastrar(curso : Curso) : boolean + listar(valorBuscado : String) : Vector<Curso>

- MVC na prática
  - CursoDAO – Trecho de Código

```
public class CursoDAO {

    private Connection connection;

    public CursoDAO() {
        this.connection = FabricaConexao.getConnection();
    }

    public boolean cadastrar(Curso curso){
        String sql = "INSERT INTO CURSO VALUES (0, ?, ?)";
        try {
            PreparedStatement ps = connection.prepareStatement(sql);
            ps.setString(1, curso.getSigla());
            ps.setString(2, curso.getDescricao());
            ps.execute();
            connection.close();
            return true;
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
    }
```

Padrão MVC

- MVC na prática
  - CursoCadastrarView



Implementar *gets* para campos de entrada e métodos para adicionar ação na interface

Padrão MVC

- MVC na prática
  - CursoCadastrarController – Trecho de código

```
public CursoCadastrarController(Curso theModel, CursoCadastrarView theView){
    this.theModel = theModel;
    this.theView = theView;
    theView.addBtnCadastrarEventListener(new CadastrarCursoListener());
}
```

Definindo a ação do botão cadastrar

Padrão MVC

- MVC na prática
  - CursoCadastrarController – Trecho de código

```
class CadastrarCursoListener implements ActionListener{
    @Override
    public void actionPerformed(ActionEvent e) {
        String  sigla = theView.getTxtSigla();
        String  descricao = theView.getTxtDescricao();

        theModel.setSigla(sigla);
        theModel.setDescricao(descricao);

        CursoDAO dao = new CursoDAO();
        boolean cadastrou = dao.cadastrar(theModel);

        if(cadastrou){
            theView.showMessage("Cadastro realizado com sucesso");
        }else{
            theView.showMessage("Cadastro não realizado!!");
        }
    }
}
```

Ação do botão cadastrar

Padrão MVC

Curso
- idCurso : int - sigla : String - descricao : String
+ cadastrar(curso : Curso) : boolean + listar(valorBuscado : String) : Vector<Curso>

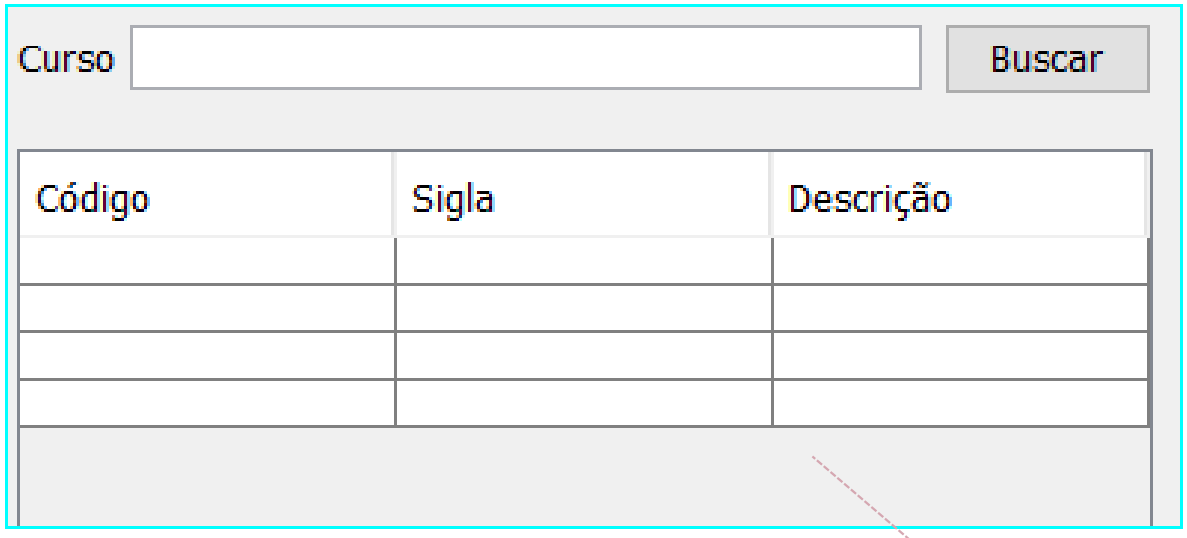
- MVC na prática
  - CursoDAO – Trecho de Código (Listar)

```
public Vector<Curso> listar(String valorBuscado){
    String sql = "SELECT * FROM CURSO c WHERE c.descricao LIKE ? ";
    try{
        PreparedStatement ps = connection.prepareStatement(sql);
        ps.setString(1, '%' + valorBuscado + '%');
        ResultSet rs = ps.executeQuery();
        Vector<Curso> listaCursos = new Vector();
        while(rs.next()){
            int codigo = rs.getInt("idCurso"); /** nome do campo no BD **/
            String sigla = rs.getString("sigla");
            String descricao = rs.getString("descricao");
            Curso curso = new Curso(codigo, sigla, descricao);
            listaCursos.add(curso);
        }
        ps.close();
        connection.close();
        return listaCursos;
    }catch (SQLException e) {
        throw new RuntimeException(e);
    }
}
```

Padrão MVC

Curso
- idCurso : int - sigla : String - descricao : String
+ cadastrar(curso : Curso) : boolean + listar(valorBuscado : String) : Vector<Curso>

- MVC na prática
  - CursoListarView



Implementar *gets* e *sets* para campos de entrada e métodos para adicionar ação na interface

Padrão MVC

- MVC na prática
  - CursoListarController – Trecho de código

```
class ListarCursoListener implements ActionListener{
    @Override
    public void actionPerformed(ActionEvent e) {

        String  valorBuscado = theView.getTxtValorBuscado();

        CursoDAO dao = new CursoDAO();
        listaCursos = dao.listar(valorBuscado);

        Vector conjuntoLinhas = new Vector();

        for(Curso curso: listaCursos){
            Vector linha = new Vector();

            linha.add(curso.getIdCurso());
            linha.add(curso.getSigla());
            linha.add(curso.getDescricao());

            conjuntoLinhas.add(linha);
        }
```

Ação do botão buscar

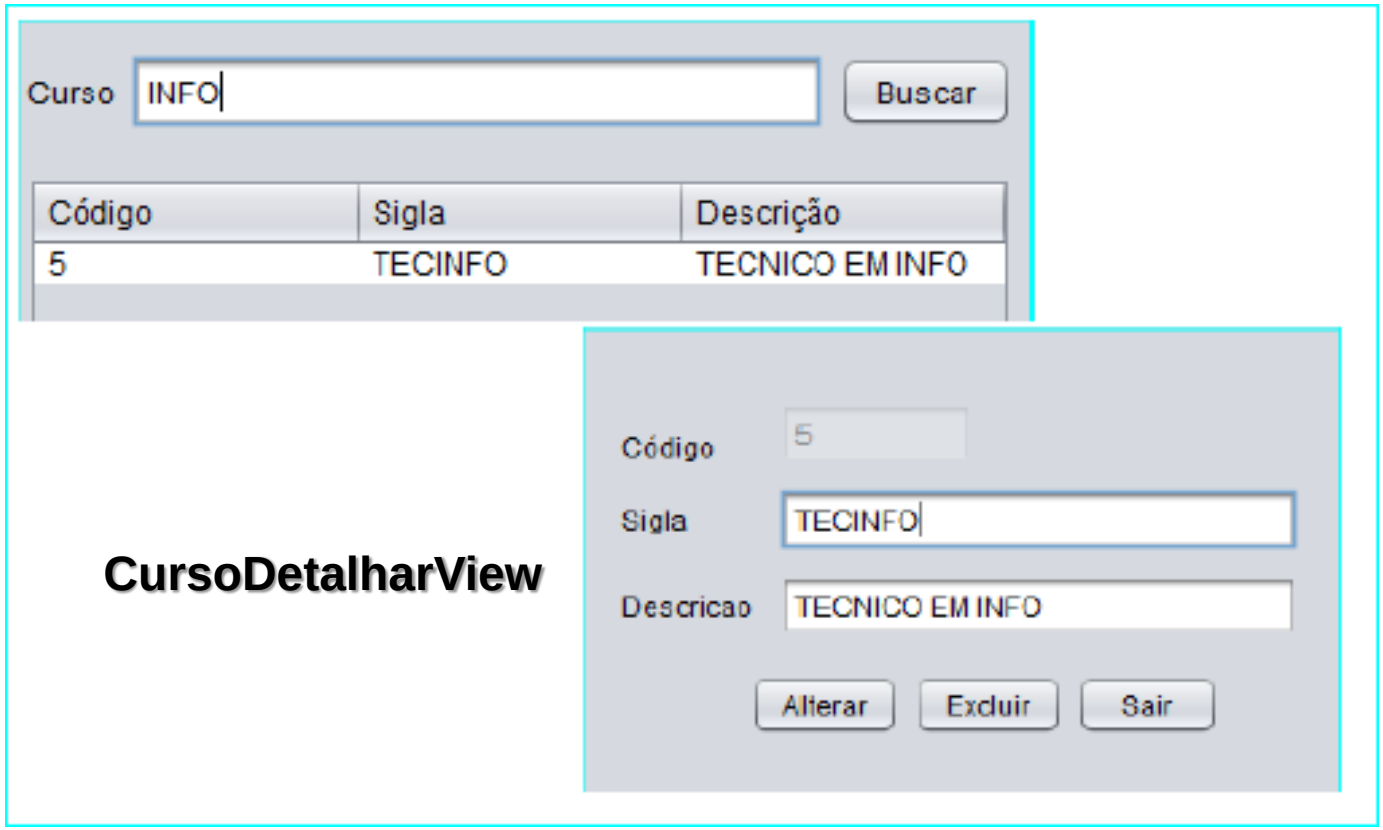
Padrão MVC

- MVC na prática
  - CursoListarController – Trecho de código (cont.)

```
        Vector conjuntoColunas = new Vector();
        conjuntoColunas.add("Código");
        conjuntoColunas.add("Sigla");
        conjuntoColunas.add("Descrição");

        DefaultTableModel modeloTabela =
                new DefaultTableModel(conjuntoLinhas, conjuntoColunas);
        theView.setTblCursos(modeloTabela);
    }
}
```

Padrão MVC

- MVC na prática



Padrão MVC

- MVC na prática
  - CursoListarController – Trecho de código

Ação do Click da Tabela

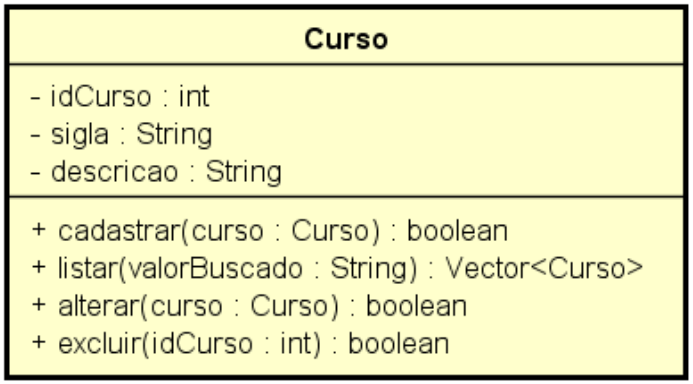
```
class DetalharCursoListener implements MouseListener{

    @Override
    public void mouseClicked(MouseEvent e) {
        int indice = theView.getTblCursos();
        Curso curso = listaCursos.get(indice);
        new CursoDetalharView(curso).setVisible(true);
    }
```

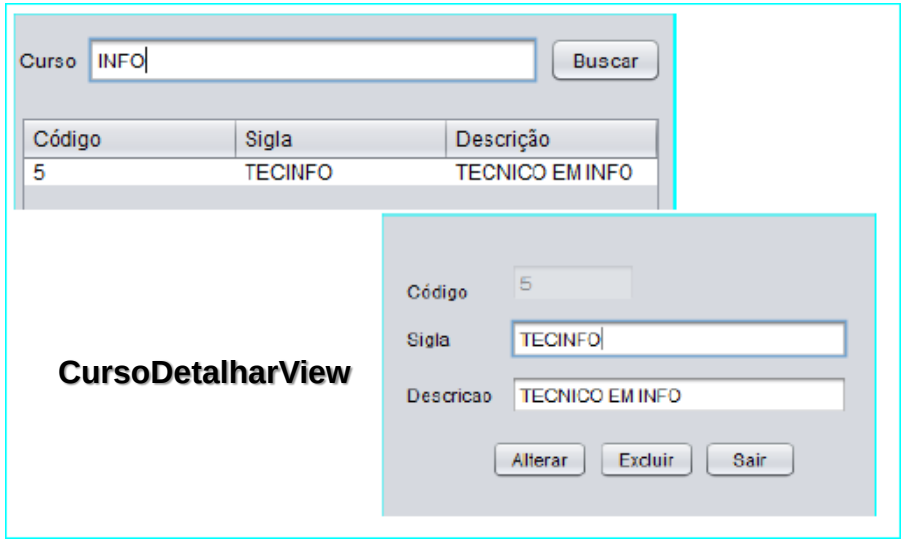
  - CursoListarView – Trecho de código

```
public int getTblCursos(){
    return this.tblCursos.getSelectedRow();
}
```

Padrão MVC

- Tarefa de Implementação
  - Utilizando o padrão MVC
    - Crie uma interface responsável por alterar e excluir cursos de acordo com as especificações abaixo.





Padrão MVC

- Tarefa de Implementação
  - Utilizando o padrão MVC
    - Implemente as seguintes funcionalidades:
      - Registrar uma compra com seus respectivos itens.
      - Consultar uma compra e exibir os seus itens.