PUC Minas

Programação Modular

PUC Minas
Bacharelado em Engenharia de Software
Prof. Daniel Kansaon

PUC Minas

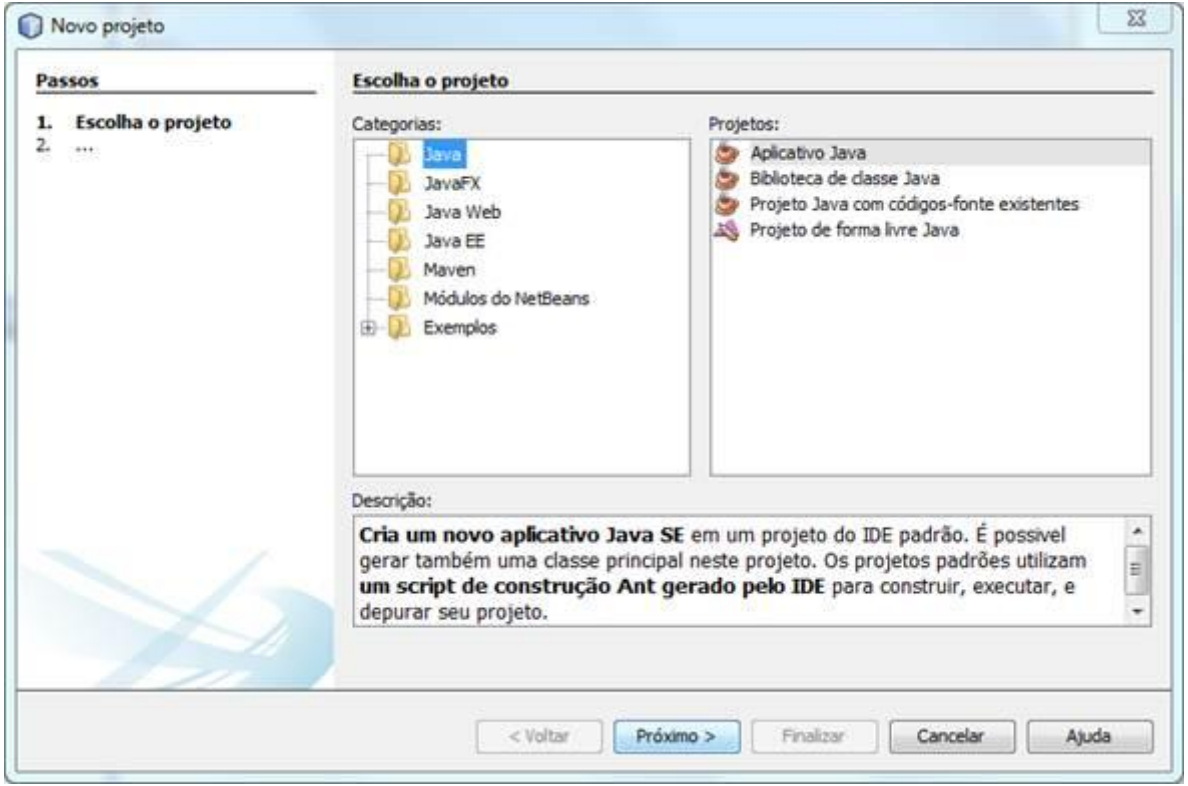
MVC + GUI + Classes de Coleção

JAVA SWING

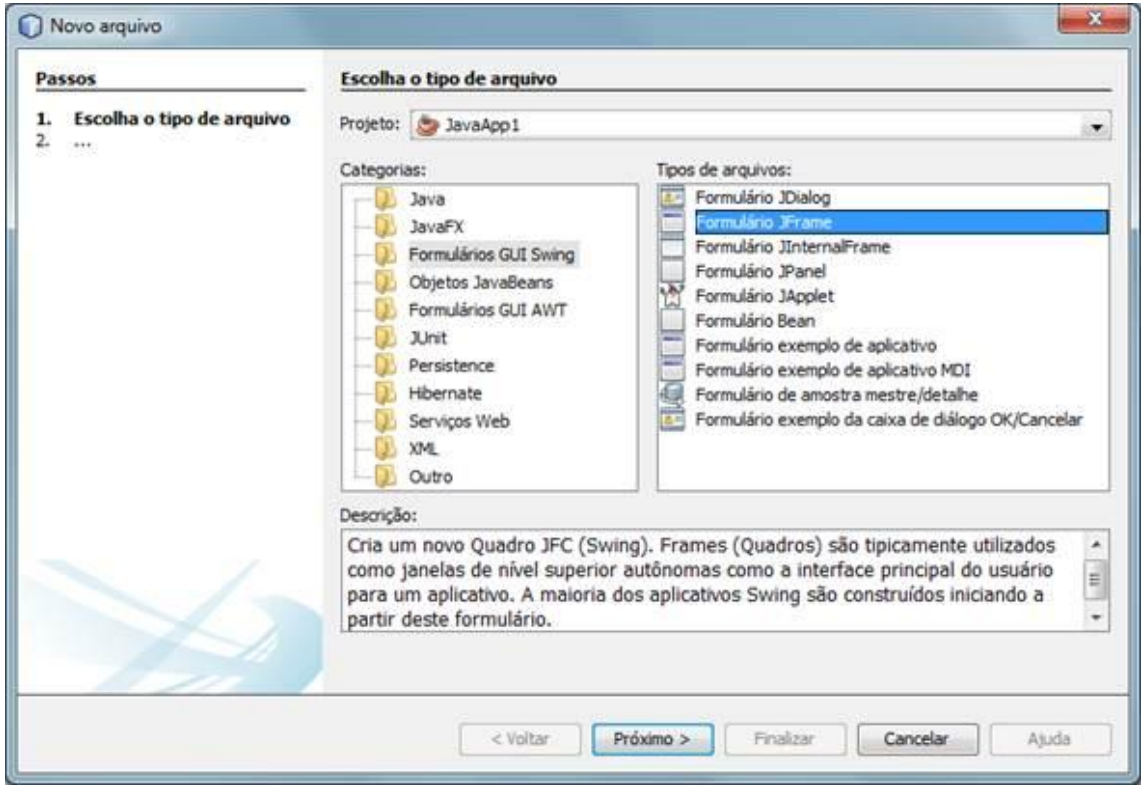
GUI significa **Graphical User Interface**

O **Java Swing** integra as JFC (Java Foundation Classes) e reúne componentes para a construção de uma **GUI**

CRIANDO PROJETO

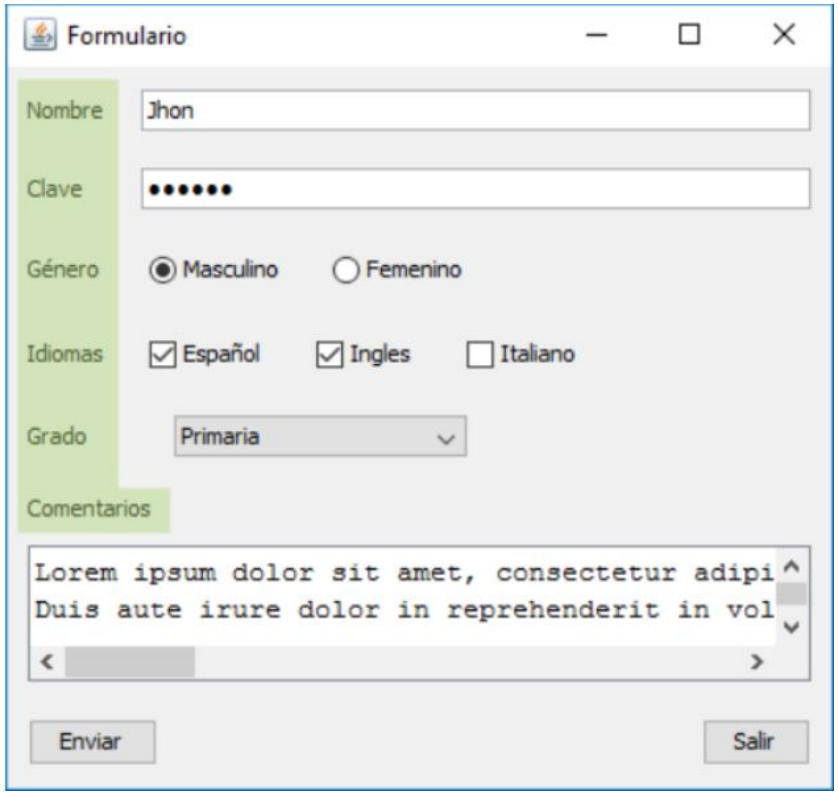


CRIANDO JFRAME



CRIANDO JFRAME

CRIANDO JFRAME



COMPONENTES

- ***JButton***: botão para realizar uma ação
- ***JLabelGenericName***: texto não editável
- **JtextFieldName**: campo de texto de uma linha
- **JTextAreaName**: área de texto de múltiplas linhas
- **JRadioButton**: Botão para escolha de uma opção (pessoa física, jurídica)
- ***JOptionPane***: Janela de diálogo para exibir mensagens

NOMEAR COMPONENTES

- ***JButton***: btnSalvar, btnCancelar
- ***JLabelGenericName***: lblNome, lblEmail
- **JtextFieldName**: txtNome, txtEmail
- **JTextAreaName**: área de texto de múltiplas linhas
- **JRadioButton**:rdbPessoa

ACESSAR VALORES


```
txtUsuario.getText();

JOptionPane.showMessage
Dialog(this, “bem
vindo”);
```

PUC Minas

MVC

MVC (Model, View, Controller)

É um **padrão de projeto** de software focado no reuso de código e a separação de responsabilidades em **três camadas**:

- Model
- View
- Controller

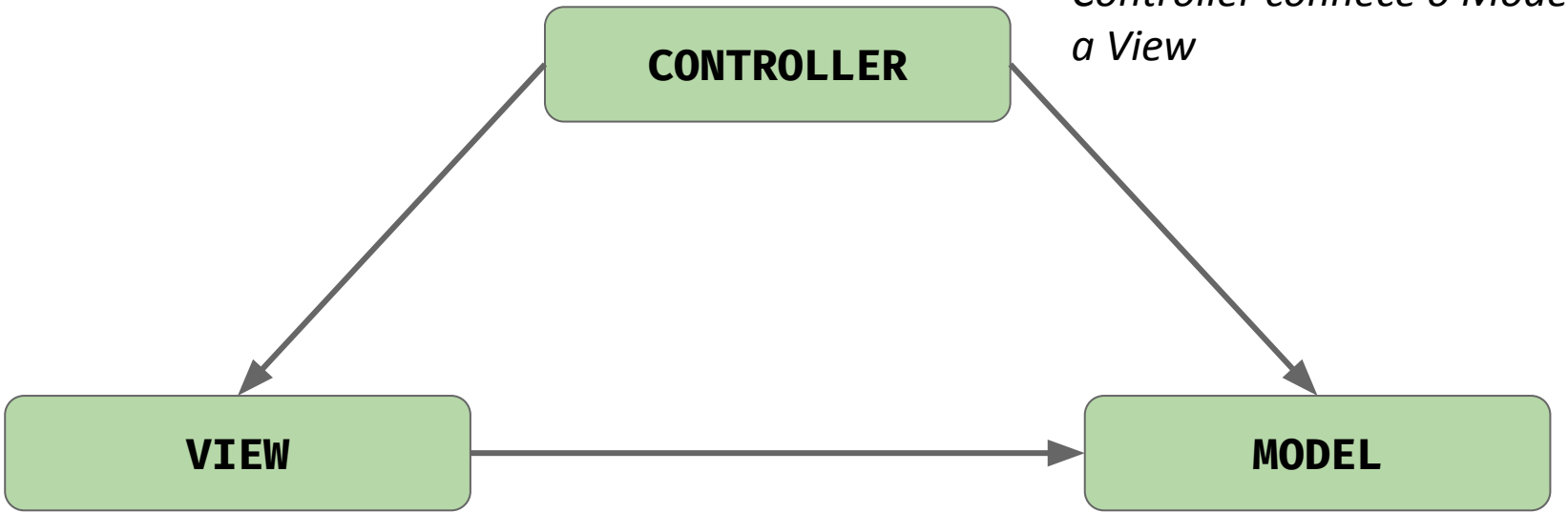
MVC (Model, View, Controller)

- **Model**: Lógica de negócio e acesso aos dados (?)

- **View**: Apresentação visual ao usuário

- **Controller**: Coordena as ações entre **Model** e **View**

MVC (Model, View, Controller)

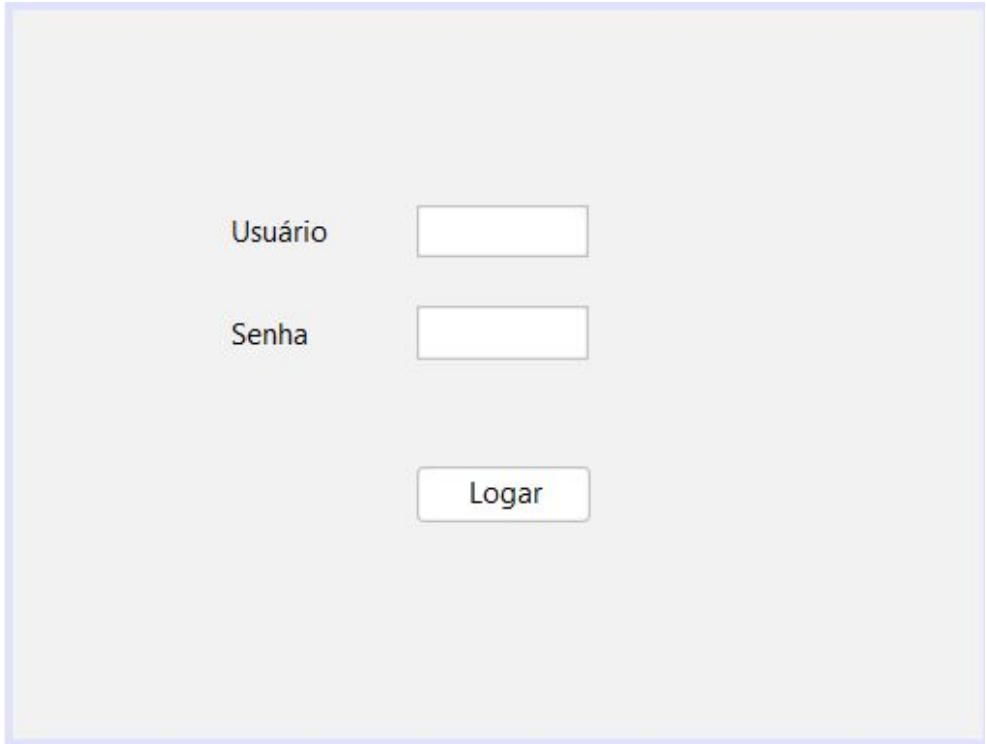


Controller conhece o Model e a View

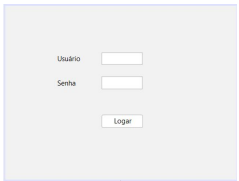
View não conhece diretamente o Controller

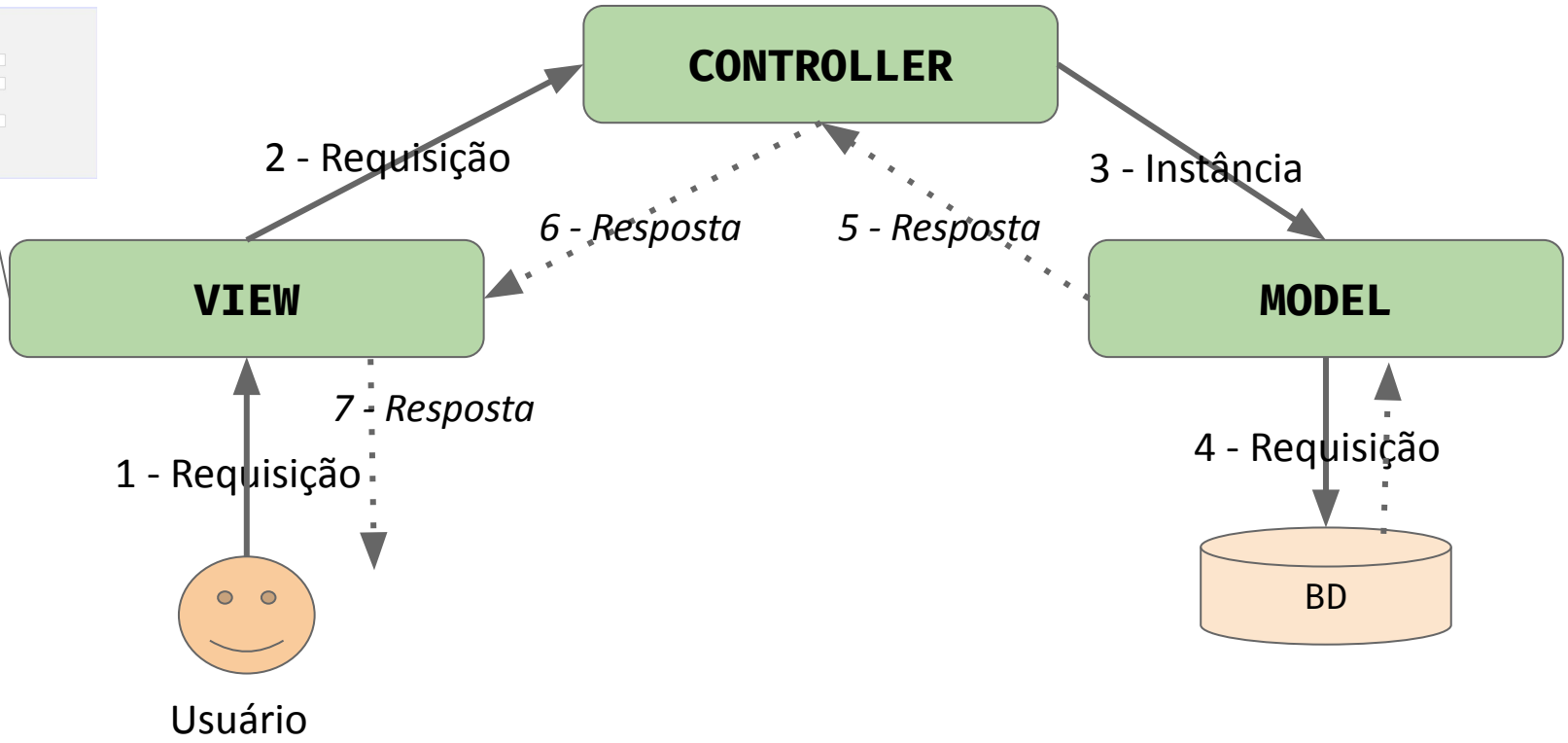
Model não conhece ninguém

COMO AUTENTICAR USANDO MVC?



COMO AUTENTICAR USANDO MVC?

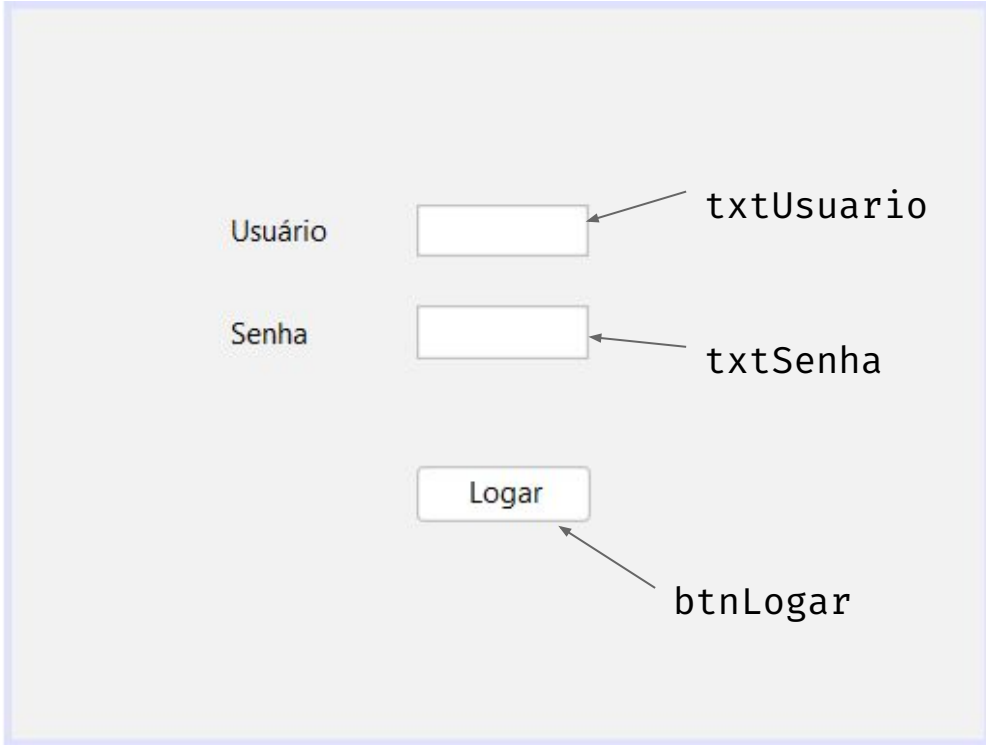




QUAL A VANTAGEM?

- Separação de responsabilidades
- Reutilização de código
- Facilidade de testes
- Organização

EXEMPLO - VIEW - Designer



EXEMPLO - VIEW - Source

```
public class AutenticarView extends javax.swing.JFrame {

    /**
     * Creates new form AutenticacaoView
     */
    public AutenticarView() {
        initComponents();
    }

     public JButton getBtnLogar() {
        return this.btnLogar;
    }

     public JTextField getTxtUsuario() {
        return this.txtUsuario;
    }

      public JTextField getTxtSenha() {
        return this.txtSenha;
    }
```

EXEMPLO - CONTROLLER

```java
public class AutenticarController {

    private AutenticarView view;

    public AutenticarController(){
        this.view = new AutenticarView();

        //Adiciona o evento ao botão logar
        this.view.getBtnLogar().addActionListener((e) -> {
            logar();
        });

        this.view.setVisible(true); //Carrega a view para o usuário
    }

    public void logar(){
        String usuario = this.view.getTxtUsuario().getText();
        String senha = this.view.getTxtSenha().getText();

        //Adicionar lógica
    }
}
```

NOTAÇÕES

letras minúsculas!

Camel Case!

controller

- AutenticarController
- ClientesController

view

- AutenticarView
- ClientesView

model

- Cliente

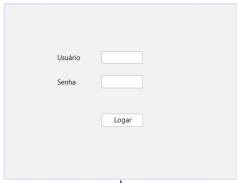
Vamos acrescentar… (DAO)

Data Access Object - DAO

- O DAO (Data Access Object) é um padrão de projeto que encapsula o acesso aos dados em uma camada separada

- A lógica de negócios não tem conhecimento direto das operações de acesso aos dados

COMO AUTENTICAR USANDO MVC?



CONTROLLER

2 - Requisição

3 - Instância

6 - Resposta

5 - Resposta

VIEW

MODEL

7 - Resposta

1 - Requisição

4 - Requisição

DAO

BD

Usuário

NOTAÇÕES

letras minúsculas!

Camel Case!

controller

- AutenticarController
- ClientesController

view

- AutenticarView
- ClientesView

model

- Cliente

dao

- ClienteDAO

QUAL A VANTAGEM?

- Separação de responsabilidades

    - Se você trocar um arquivo .txt por um banco MySQL, só o DAO precisa mudar

ESTUDAR EXEMPLO

https://github.com/pucmg-aulas/ProjCarroJFrame

ArrayList<>

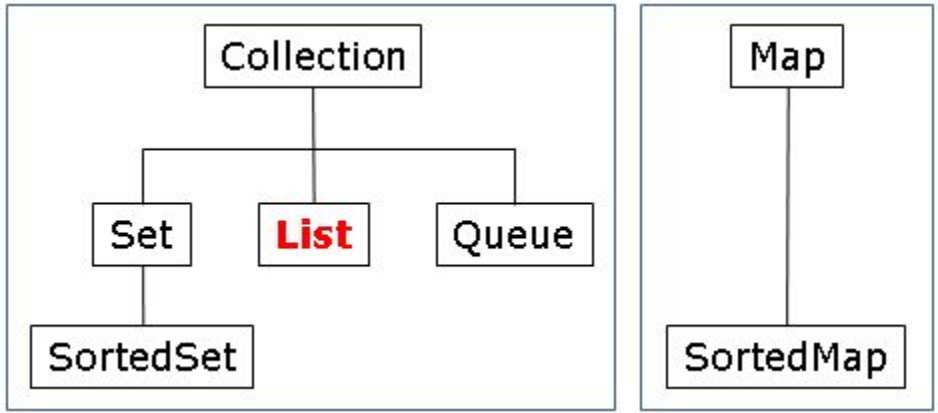
ARRAYLIST<>

- ArrayList é uma classe para coleções

- É uma estrutura de dados que funciona como um vetor (array)

- Tamanho dinâmico

ARRAYLIST<>



ARRAYLIST<>

- Mais flexível que um array tradicional (int[], String[])

```
ArrayList<String> nomes = new ArrayList<String>();
```

ARRAYLIST<>

- Mais fácil de manipular: adicionar, remover, buscar, etc

- Muito usado para guardar listas de objetos

```
ArrayList<Cliente> clientes = new ArrayList<Cliente>();
```

MÉTODOS

https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
boolean	`add(E e)`	Appends the specified element to the end of this list.
void	`add(int index, E element)`	Inserts the specified element at the specified position in this list.
boolean	`addAll(Collection<? extends E> c)`	Appends all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's Iterator.
boolean	`addAll(int index, Collection<? extends E> c)`	Inserts all of the elements in the specified collection into this list, starting at the specified position.
void	`clear()`	Removes all of the elements from this list.
Object	`clone()`	Returns a shallow copy of this `ArrayList` instance.
boolean	`contains(Object o)`	Returns `true` if this list contains the specified element.
void	`ensureCapacity(int minCapacity)`	Increases the capacity of this `ArrayList` instance, if necessary, to ensure that it can hold at least the number of elements specified by the minimum capacity argument.
void	`forEach(Consumer<? super E> action)`	Performs the given action for each element of the `Iterable` until all elements have been processed or the action throws an exception.
E	`get(int index)`	Returns the element at the specified position in this list.
int	`indexOf(Object o)`	Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.
boolean	`isEmpty()`	Returns `true` if this list contains no elements.
Iterator<E>	`iterator()`	Returns an iterator over the elements in this list in proper sequence.
int	`lastIndexOf(Object o)`	Returns the index of the last occurrence of the specified element in this list, or -1 if this list does

ARRAYLIST<>

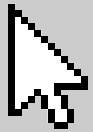
```
import java.util.ArrayList;

import java.util.List;
```

```
List<Cliente> listaClientes = new ArrayList<>();
listaClientes.add(new Cliente("João", "joao@gmail.com"));
listaClientes.forEach(cli -> System.out.print(cli.nome + " - " + cli.email));
```

PUC Minas

Obrigado!



PUC Minas
Bacharelado em Engenharia de Software
Prof. Daniel Kansaon