Linguagem de Programação II
IMD0040
Aula 23 – Interface Gráfica e Controle de Janelas

INSTITUTO METRÓPOLE DIGITAL

UFRN UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

Controle de Janelas

- Uma **janela** é uma área visual contendo algum tipo de interface do usuário, permitindo a saída do sistema ou permitindo a entrada de dados.

- As janelas são geralmente apresentadas como objetos **bidimensionais** e **retangulares**, organizados em uma área de trabalho.

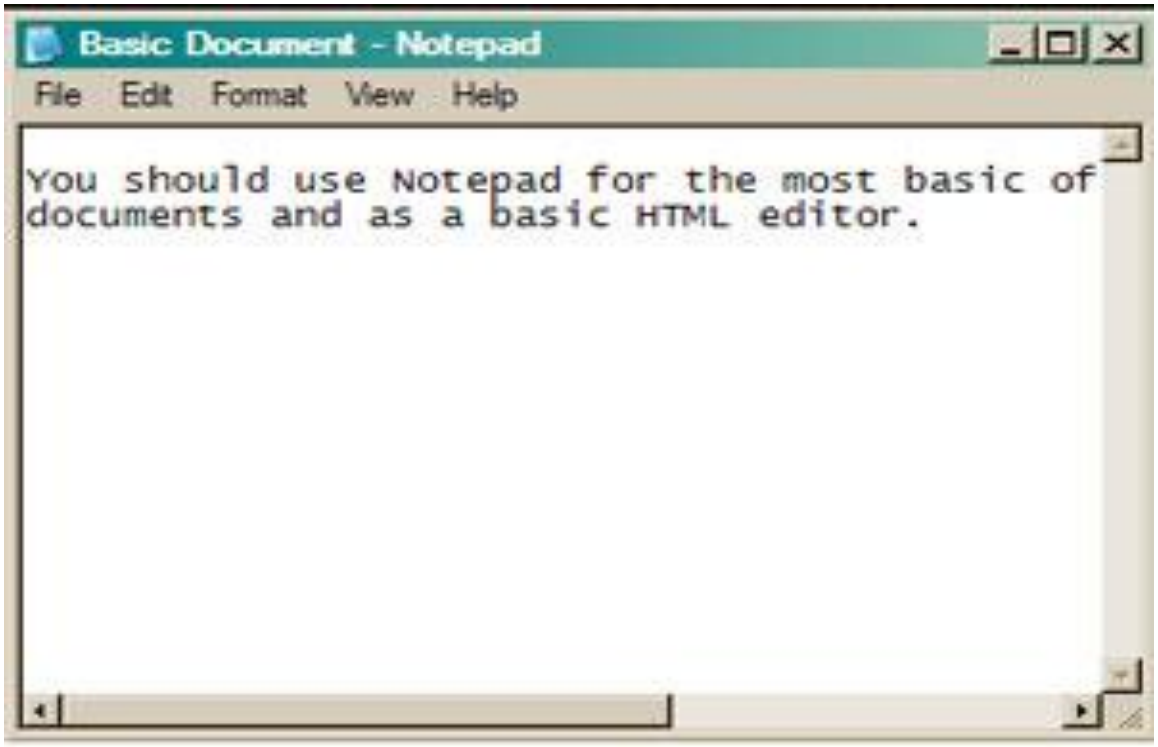
- Existem dois métodos de manipulação de janelas, são eles: SDI e MDI:

Single Document Interface

- Single Document Interface (SDI): uma SDI abre cada documento em sua própria janela principal.
- Cada janela tem seu próprio menu, barra de ferramentas e entrada na barra de tarefas.
- SDI não é restrito a uma janela pai.
- Isso torna mais fácil para o usuário visualizar o conteúdo das várias janelas.

Single Document Interface

- Exemplo de SDI:

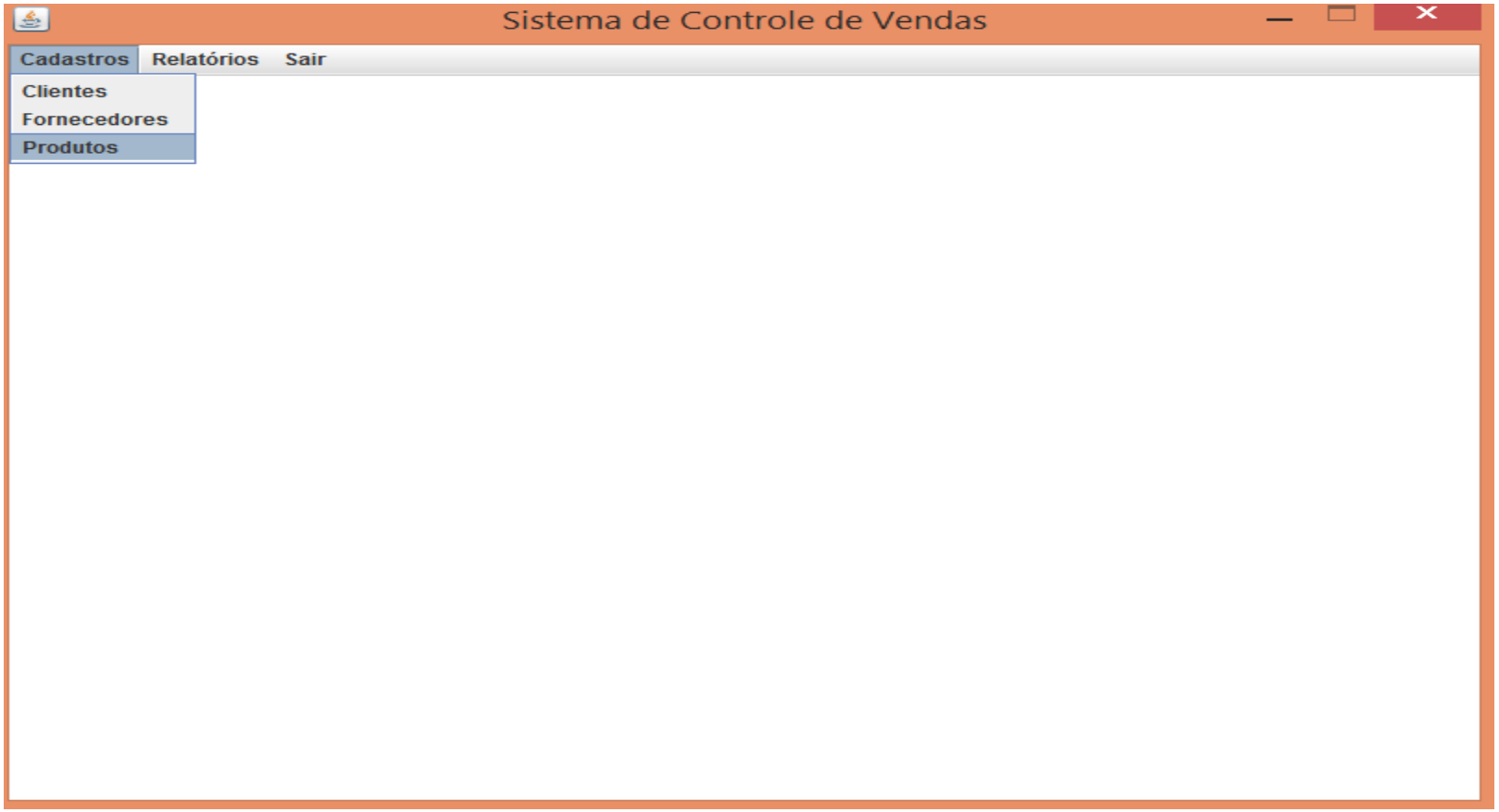


Multiple Document Interface

- Multiple Document Interface (MDI): Um MDI permite abrir mais de um documento ao mesmo tempo.
- O MDI tem uma janela pai e qualquer número de janelas filho.
- As janelas filhas geralmente compartilham várias partes da interface da janela pai, incluindo a barra de menus, a barra de ferramentas e a barra de status.
- Portanto, um MDI é restrito à janela pai.

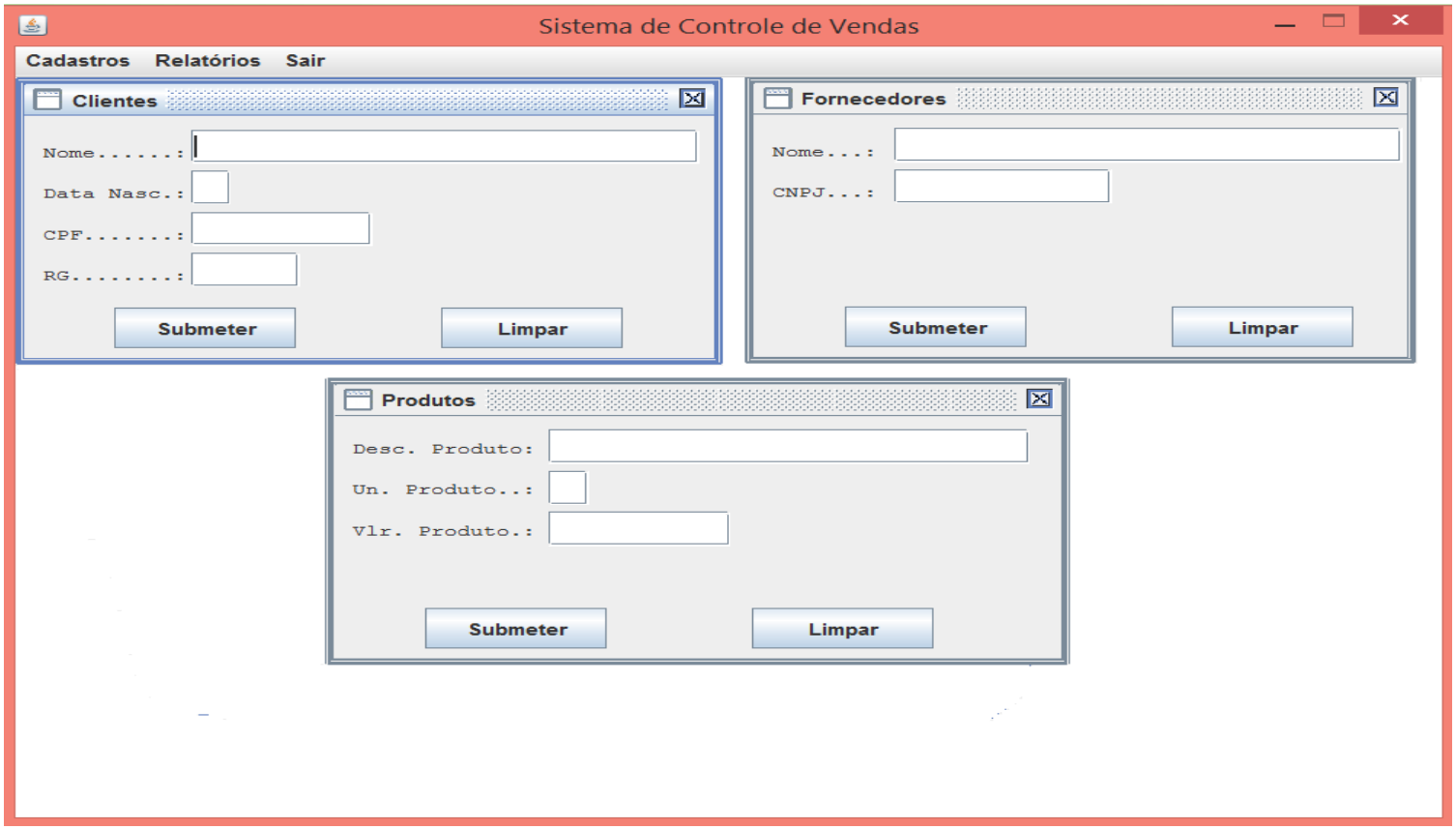
Multiple Document Interface

- Exemplo de MDI:



Multiple Document Interface

- Exemplo de MDI:



Multiple Document Interface

- **Swing** disponibiliza duas classes para tal tarefa:
  - **`JDesktopPane`**;
  - **`JInternalFrame`**.

Multiple Document Interface

- Classe **`TelaPrincipal`**:

```
public class TelaPrincipal extends JFrame implements ActionListener{

    JDesktopPane dtp = new JDesktopPane();

    JMenuBar mnbar    = new JMenuBar();
    /* ... */

    public TelaPrincipal(){
        Container ct = this.getContentPane();
        ct.setLayout(new BorderLayout());

        setJMenuBar(mnbar);
        /* ... */

        ct.add(BorderLayout.CENTER, dtp);

        setSize(800,600);
        setResizable(false);
        setTitle("Sistema de Controle de Vendas");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
```

Multiple Document Interface

- Classe **`TelaCliente`**:

```
public class TelaCliente extends JInternalFrame implements ActionListener{

    // rótulos
    JLabel lnome =  new JLabel("Nome.:");
    // campos
    JTextField tnome  = new JTextField();
    // botões
    JButton b1 = new JButton("Submeter");

    public TelaCliente(String str)  {
        super(str,false,true);

        Container ct = this.getContentPane();
        ct.setLayout(null);

        // coordenadas
        lnome.setBounds(10,10,100,30);
        tnome.setBounds(55,10,280,25);
        // idem
        b1.setBounds(10 ,140,100,30);
        // adicionando componentes
        ct.add(lnome);
        ct.add(tnome);
        // evento dos botões
        b1.addActionListener(this);
        // especificações do formulário
        setSize(350,230);
        setTitle(str);
    }
```

Multiple Document Interface

- Classe **`TelaFornecedor`**:

```
public class TelaFornecedor extends JInternalFrame implements ActionListener {
    private static final long serialVersionUID = 1L;

    Banco bc;

    private Font f  = new Font("Courier", Font.PLAIN, 12);

    // rótulos
    JLabel lnome    = new JLabel("Nome...:");
    JLabel lcnpj    = new JLabel("CNPJ...:");

    // campos
    JTextField tnome   = new JTextField();
    JTextField tcnpj    = new JTextField();

    // botões
    JButton btSubmeter = new JButton("Submeter");
    JButton btLimpar = new JButton("Limpar");

    public TelaFornecedor(String str){
        super(str,false,true);

        Container ct = this.getContentPane();
        ct.setLayout(null);

        // setando a fonte
        lnome.setFont(f);
        lcnpj.setFont(f);

        // coordenadas
        lnome.setBounds(10,10,100,30);
        tnome.setBounds(77,10,280,25);
```

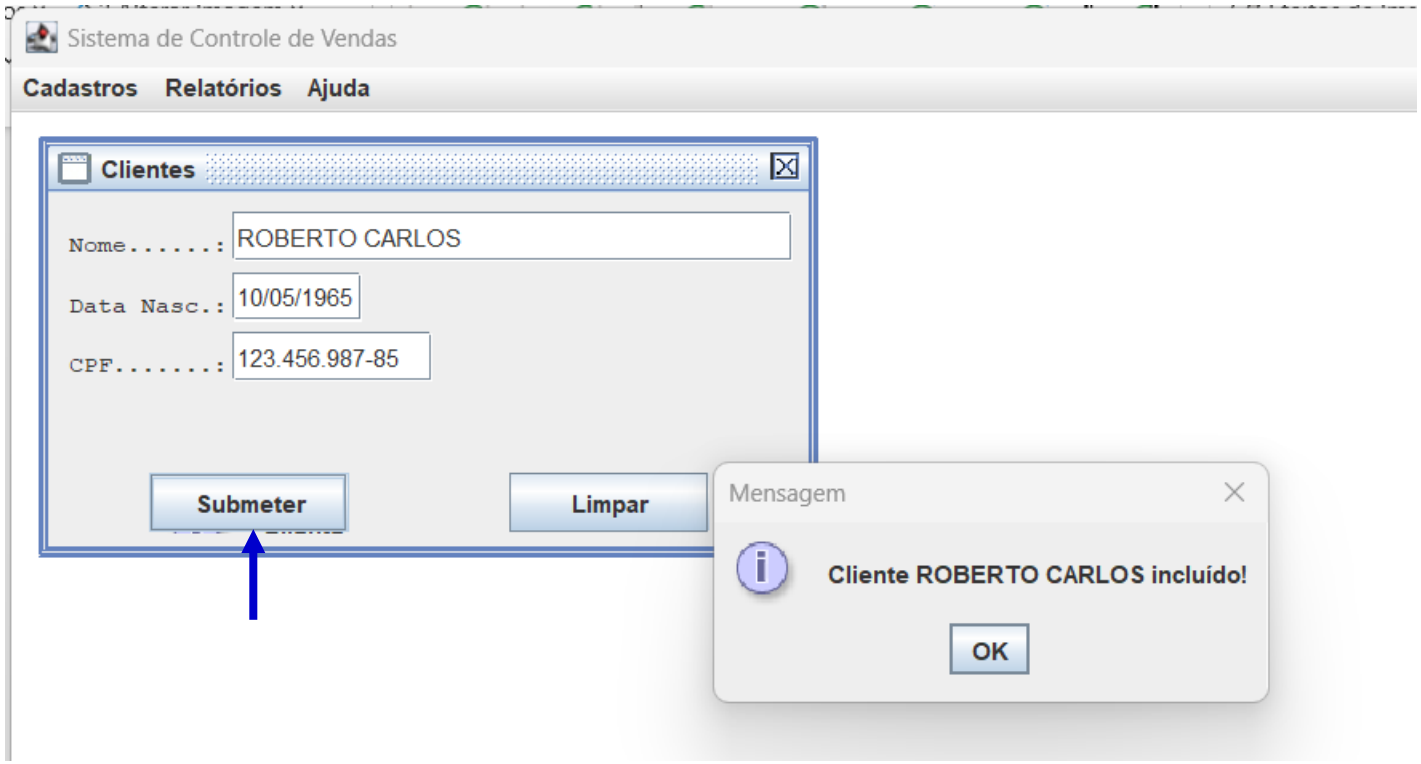
Multiple Document Interface

- Classe **`TelaProduto`**:

```
public class TelaProduto extends JInternalFrame implements ActionListener{
    private static final long serialVersionUID = 1L;

    Banco bc;

    // rótulos
    JLabel ldescProduto = new JLabel("Desc. Produto:");
    JLabel luniProduto  = new JLabel("Un. Produto..:");
    JLabel lvlrProduto  = new JLabel("Vlr. Produto.:");

    private Font f  = new Font("Courier", Font.PLAIN, 12);

    // campos
    JTextField tdesProduto = new JTextField();
    JTextField tuniProduto = new JTextField();
    JTextField tvlrProduto = new JTextField();

    // botões
    JButton btSubmeter = new JButton("Submeter");
    JButton btLimpar = new JButton("Limpar");

    public TelaProduto(String str){
        super(str,false,true);

        Container ct = this.getContentPane();
        ct.setLayout(null);

        // setando a fonte
        ldescProduto.setFont(f);
        luniProduto.setFont(f);
        lvlrProduto.setFont(f);
```

Multiple Document Interface

- Visão **`TelaCliente`**:

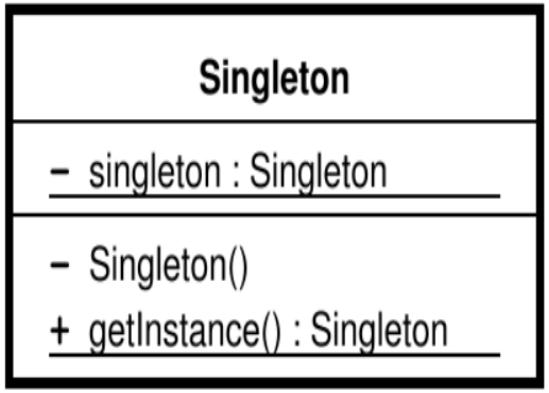


Perguntas...

Padrão de Projeto: Singleton

- Objetivo: permitir que uma classe tenha **somente uma instância** no projeto e que essa instância seja de acesso global.
- Padrão muito utilizado para trabalhar com instâncias de classes que não alteram seus estados ou quando há necessidade de utilizar algum método especifico da classe, várias vezes.
  - Algo que seria um problema sem o Singleton.
  - Muitas instâncias seriam criadas somente para utilizar alguns métodos, esse sendo independentes do estado do objeto.

Padrão de Projeto: Singleton

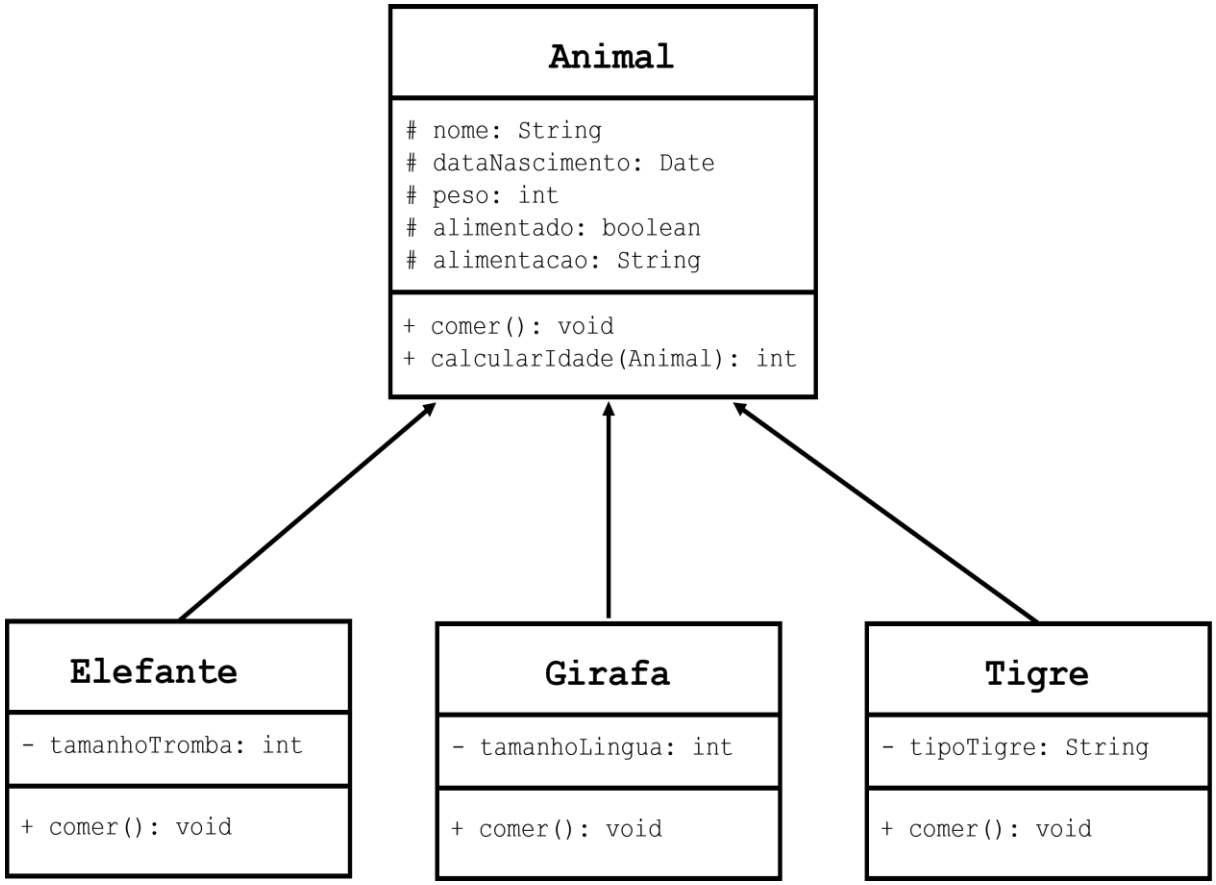



```
public class BancoDeDados {

   private static BancoDeDados instance;
   ...

   private BancoDeDados(){
      ...
   }

   public static BancoDeDados getInstance(){

      if( instance == null ){

         instance = new BancoDeDados();
      }

      return instance;
   }
   ...
}
```

Atividade

- Especificação:
  - Crie um novo projeto ZooNatal;
  - Crie três pacotes “br.ufrn.imd.modelo”, “br.ufrn.imd.visao” e “br.ufrn.imd.dao”;
  - Dentro do pacote “br.ufrn.imd.modelo”, crie quatro classes:
    - Animal com os seguintes atributos (nome, dataNascimento, peso, tipoAlimentacao e alimentado)
    - Elefante com o seguinte atributo (tamanhoTromba);
    - Girafa com o seguinte atributo (tamanhoLingua);
    - Tigre com o seguinte atributo (tipoTigre).

Atividade


Animal
nome: String
dataNascimento: Date
peso: int
alimentado: boolean
alimentacao: String
+ comer(): void
+ calcularIdade(Animal): int
Elefante
- tamanhoTromba: int
+ comer(): void
Girafa
- tamanhoLingua: int
+ comer(): void
Tigre
- tipoTigre: String
+ comer(): void

Atividade

- Especificação dos métodos:
  - Note que cada bicho possui um cálculo específico para a quantidade de alimento diário. No geral, cada bicho consome 5% de seu peso em comida por dia. Contudo, as Girafas podem consumir cerca de 10%, os elefantes cerca de 15% e os Tigres cerca de 4% de seu peso.
  - O veterinário responsável por alimentar os animais precisa atualizar a tag de controle de cada animal, assim que o mesmo for alimentado. Inicialmente, todo e qualquer animal é criado com o atributo alimentado igual a false.
  - Além de alimentar os bichos, é importante manter os animais em jaulas (como se fosse uma grande coleção de animais - utilize um ArrayList). Para facilitar a identificação, é importante imprimir uma lista com todos os atributos dos bichos, incluindo a idade de cada um.

Atividade

- Especificação dos métodos:
  - Como os animais herbívoros se alimentam de frutas e legumes é muito importante para o zoológico saber qual a quantidade total de alimento gasto com elefantes e girafas depois da alimentação. Isso também vale para os tigres que se alimentam de carne.
  - Todos os animais precisarão passar por uma consulta de acordo com suas idades. Para os tigres, a consulta sempre ocorre em períodos de 3 anos. Já para as girafas a consulta ocorre em períodos de 5 anos. E por último, os elefantes sempre passam por essa consulta a cada 7 anos.
  - Os veterinários sempre gostam de consultar uma lista de animais que mostra quais são os animais mais novos e mais velhos por espécie de acordo com suas idades.