

Cadastro de Animais Para Zoológico

By: Victor Dallagnol Bento

11 de Dezembro de 2017, Santa Maria - RS

- ⇒ O principal objetivo do programa é efetuar o cadastro de animais que podem ou não estar em extinção.
- ⇒ O programa possui um total de 7 arquivos . Java.



- ↳ *Animal: Classe para o cadastro de animais.*
- ↳ *AnimalExtinto: Estende de Animal, possui atributos específicos para extinção.*
- ↳ *Listagem: Classe com a implementação para as listas de ambos os tipos, Animal e AnimalExtinto.*
- ↳ *AnimalNaoEncontrado, BaseDeDadosVazia, CamposInvalidos: Exceções que são tratadas no decorrer do código.*
- ↳ *Cadastro: Frame para efetuar o cadastro de animais no zoológico.*

⇒ *Animal.java*

```
package cadastrazoológico;
/**
 *
 * @author bent@victor
 */
public class Animal {
    private String raca;
    private float peso;
    private short jaula;
    private boolean restricao;

    /* Construtor para classe Animal */
    public Animal(String raca, float peso, short jaula, boolean restricao){
        this.raca = raca;
        this.peso = peso;
        this.jaula = jaula;
        this.restricao = restricao;
    }
}
```

↳ Declaração dos atributos para raça, peso, jaula, e se o animal possui ou não uma restrição. Declaração do construtor para a classe.

```
/* Métodos Get's */
public String getRaca(){
    return raca;
}
public float getPeso(){
    return peso;
}
public short getJaula(){
    return jaula;
}
public boolean getRestricao(){
    return restricao;
}

/* Métodos Set's */
public void setRaca(String raca){
    this.raca = raca;
}
public void setPeso(float peso){
    this.peso = peso;
}
public void setJaula(short jaula){
    this.jaula = jaula;
}
public void setRestricao(boolean restricao){
    this.restricao = restricao;
}
}
```

↳ Métodos Get's e Set's para acessar atributos da classe.

⇒ *AnimalExtinto.java*

```
package cadastrozoologico;

/**
 *
 * @author bent@victor
 */
public class AnimalExtinto extends Animal{
    private boolean riscoExtincao;
    private int especiesRestantes;
    private int estimativaExtincao;

    /* Construtor para classe Extinção e para superclasse Animal */
    public AnimalExtinto(String raca, float peso, short jaula, boolean restricao,
        boolean riscoExtincao, int especiesRestantes, int estimativaExtincao){
        super(raca, peso, jaula, restricao);
        this.riscoExtincao = riscoExtincao;
        this.especiesRestantes = especiesRestantes;
        this.estimativaExtincao = estimativaExtincao;
    }

    /* Métodos Get's */
    public boolean getRiscoExtincao(){
        return riscoExtincao;
    }
    public int getEspeciesRestantes(){
        return especiesRestantes;
    }
    public int getEstimativaExtincao(){
        return estimativaExtincao;
    }

    /* Métodos Set's */
    public void setRiscoExtincao(boolean riscoExtincao){
        this.riscoExtincao = riscoExtincao;
    }
    public void setEspeciesRestantes(int especiesRestantes){
        this.especiesRestantes = especiesRestantes;
    }
    public void setEstimativaExtincao(int estimativaExtincao){
        this.estimativaExtincao = estimativaExtincao;
    }
}
```

↳ Declaração dos atributos para raça, peso, jaula, e se o animal possui ou não uma restrição. Declaração do construtor para a classe.

→ Criação dos métodos Get's e Set's para ter acesso aos atributos da classe AnimalExtinto.

⇒ Listagem.java

```
package cadastrozoologico;
import java.util.ArrayList;
/**
 *
 * @author bentovictor
 */
public class Listagem {
    public ArrayList<Animal> ListAnimal = new ArrayList();
    public ArrayList<AnimalExtinto> ListAnimalExt = new ArrayList();

    /**Métodos para cadastrar um Animal */
    public void cadastrarAnimal(Animal a){
        ListAnimal.add(a); /*Insere na lista um contato*/
    }
    /**Métodos para cadastrar um Animal Extinto*/
    public void cadastrarAnimalExt(AnimalExtinto aex){
        ListAnimalExt.add(aex); /*Insere na lista um contato*/
    }

    /* Tamanho das listas */
    public int numeroDeAnimais(){
        return ListAnimal.size();
    }
    public int numeroDeAnimaisExt(){
        return ListAnimalExt.size();
    }
}
```

→ Função para criação das listas para Animal e AnimalExtinto.

→ Criação do método para adicionar Animal e AnimalExtinto para suas respectivas Listas.

→ Criação dos métodos que retornam o tamanho das listas.

⇒ Exceções: *AnimalNaoEncontrado.java*, *BaseDeDadosVazia.java* e *CamposInvalidos.java*

```
package cadastrozoologico;
import java.io.IOException;
/**
 *
 * @author bentovictor
 */
public class AnimalNaoEncontrado extends Exception {
    public AnimalNaoEncontrado(String aviso){ /*Const
        super(aviso);
    }
}
```

→ Classe para exceção caso o Animal não esteja no banco de dados do Zoológico.

```
package cadastrozoologico;
/**
 *
 * @author bentovictor
 */
public class BaseDeDadosVazia extends Exception {
    public BaseDeDadosVazia(String aviso){ /*Const
        super(aviso);
    }
}
```

↳ Classe para exceção quando busca é feita e banco de dados está vazio.

```
package cadastrozoologico;
/**
 *
 * @author bentovictor
 */
public class CamposInvalidos extends Exception{
    public CamposInvalidos(String aviso){ /*Const
        super(aviso);
    }
}
```

↳ Classe para exceção quando há uma tentativa de cadastro mas algum(s) campo(s) não foi preenchido.

⇒ Cadastro.java

```
package cadastrozoologico;
import java.awt.Component;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author bentovictor
 */
public class Cadastro extends javax.swing.JFrame {

    Listagem list = new Listagem(); // Declaração da Classe que contém as Listas

    // Tabela para Animais
    public void LoadTableAnimais(){
        // Estrutura da Tabela (linhas e colunas)
        DefaultTableModel animais = new DefaultTableModel(new Object[] {"Raça", "Peso", "Jaula", "Restrição"}
        for(int i = 0; i < list.numeroDeAnimais(); ++i){
            animais.addRow(new Object[] {list.ListAnimal.get(i).getRaca(), list.ListAnimal.get(i).getPeso(),
        }
        tbl_animais.setModel(animais);
    }

    // Tabela para Animais Extintos
    public void LoadTableAnimaisExt(){
        // Estrutura da Tabela (linhas e colunas)
        DefaultTableModel animaisext = new DefaultTableModel(new Object[] {"Raça", "Peso", "Jaula", "Restriç
        for(int i = 0; i < list.numeroDeAnimaisExt(); ++i){
            animaisext.addRow(new Object[] {list.ListAnimalExt.get(i).getRaca(), list.ListAnimalExt.get(i).get
        }
        tbl_animaisext.setModel(animaisext);
    }
}
```

→ Instância e criação de uma nova classe para as listas.

→ Criação e configuração das linhas e colunas para as tabelas de Animais e Animais Extintos.


```

public Cadastro() {
    initComponents();
    setLocationRelativeTo(null);    // Centraliza Tela

    /* Inicializa Campos de Textos e Botões como desabilitados
       Até que o botão NOVO seja clicado */
    c_raca.setEditable(false);
    c_peso.setEditable(false);
    c_jaula.setEditable(false);
    c_especiesrestantes.setEditable(false);
    c_estimativaextincao.setEditable(false);
    c_buscar.setEditable(false);
    c_buscar2.setEditable(false);

    btn_apagar.setEnabled(false);
    btn_apagar2.setEnabled(false);
    btn_restricao_sim.setEnabled(false);
    btn_restricao_nao.setEnabled(false);
    btn_riscoextincao_nao.setEnabled(false);
    btn_riscoextincao_sim.setEnabled(false);
    btn_cadastrar.setEnabled(false);
    btn_cancelar.setEnabled(false);
    btn_buscar.setEnabled(false);
    btn_buscar2.setEnabled(false);
}

```

Quando botão NOVO é pressionado os campos de texto são liberados para escrita e os demais botões também.

O botão NOVO fica desabilitado.

→ Na inicialização do frame ele é centralizado no meio da tela.

→ Os botões (com exceção do novo) e os campos de texto são desabilitados.

```

/***** NOVO *****/
private void btn_novoActionPerformed(java.awt.event.ActionEvent evt) {

    // Seta os botões e campos após clicar no botão NOVO
    btn_novo.setEnabled(false); // Como foi clicado, não pode ser clicado de

    c_raca.setEditable(true);
    c_peso.setEditable(true);
    c_jaula.setEditable(true);
    c_especiesrestantes.setEditable(false);
    c_estimativaextincao.setEditable(false);
    c_buscar.setEditable(true);
    c_buscar2.setEditable(true);

    btn_apagar.setEnabled(true);
    btn_apagar2.setEnabled(true);
    btn_restricao_nao.setEnabled(true);
    btn_restricao_sim.setEnabled(true);
    btn_riscoextincao_nao.setEnabled(true);
    btn_riscoextincao_sim.setEnabled(true);
    btn_cadastrar.setEnabled(true);
    btn_cancelar.setEnabled(true);
    btn_buscar.setEnabled(true);
    btn_buscar2.setEnabled(true);
}

```



```

// CANCELAR
private void btn_cancelarActionPerformed(java.awt.event.ActionEvent evt) {

    // Limpa campos de Texto
    c_raca.setText("");
    c_estimativaextincao.setText("");
    c_buscar.setText("");
    c_buscar2.setText("");
    c_especiesrestantes.setText("");
    c_jaula.setText("");
    c_peso.setText("");
    // Seta botões e campos de textos para não serem acessados até que NOVO seja clicado
    c_raca.setEditable(false);
    c_peso.setEditable(false);
    c_jaula.setEditable(false);
    c_especiesrestantes.setEditable(false);
    c_estimativaextincao.setEditable(false);

    btn_novo.setEnabled(true);
    btn_restricao_nao.setEnabled(false);
    btn_restricao_sim.setEnabled(false);
    btn_riscoextincao_nao.setEnabled(false);
    btn_riscoextincao_sim.setEnabled(false);
    btn_cadastrar.setEnabled(false);
    btn_cancelar.setEnabled(false);
    // Limpa Seleções dos botões
    buttonGroup1.clearSelection();
    buttonGroup2.clearSelection();
}

```

→ Ao ser clicado o botão CANCELAR limpa as caixas de texto, os botões de seleção e desabilita campos e botões (com exceção do NOVO).

```

private void btn_riscoextincao_naoActionPerformed(java.awt.event.ActionEvent evt) {

    // SE animal não estiver em risco de Extinção
    // desabilita Campos de texto referentes a Extinção
    if(btn_riscoextincao_nao.isSelected()){
        c_especiesrestantes.setEditable(false);
        c_estimativaextincao.setEditable(false);
    }else{
        // CASO CONTRÁRIO habilita Campos de texto referentes a Extinção
        c_especiesrestantes.setEditable(true);
        c_estimativaextincao.setEditable(true);
    }
}

```

→ Se o animal não estiver em Extinção os campos de preenchimento deste requisito são desabilitados.

```

private void btn_riscoextincao_simActionPerformed(java.awt.event.ActionEvent evt) {

    // SE animal estiver em risco de Extinção
    // habilita Campos de texto referentes a Extinção
    if(btn_riscoextincao_sim.isSelected()){
        c_especiesrestantes.setEditable(true);
        c_estimativaextincao.setEditable(true);
    }else{// CASO CONTRÁRIO desabilita Campos de texto referentes a Extinção
        c_especiesrestantes.setEditable(false);
        c_estimativaextincao.setEditable(false);
    }
}

```

→ Caso contrário, se o animal estiver em Extinção, os campos são habilitados.

```

// ***** CADASTRAR *****
private void btn_cadastrarActionPerformed(java.awt.event.ActionEvent evt) {
    int flagCadastro = 0; // FLAG para controlar preenchimento dos campos

    // Exceção para Campos inválidos
    if(btn_riscoextincao_sim.isSelected()){
        try{
            if(c_raca.getText().trim().equals("") || c_peso.getText().trim().equals("") || c_jaula.getText().trim().equals("") || c_jaula.getText().trim().equals("")){
                flagCadastro = 1; // Campos não foram corretamente preenchidos
                throw new CamposInvalidos("avisar");
            }
        }catch(CamposInvalidos e){
            Component ERRO = null;
            JOptionPane.showMessageDialog(ERRO, "Preencha todos os campos antes de Cadastrar.");
        }
    }else if(btn_riscoextincao_nao.isSelected()){
        try{
            if(c_raca.getText().trim().equals("") || c_peso.getText().trim().equals("") || c_jaula.getText().trim().equals("") || c_jaula.getText().trim().equals("")){
                flagCadastro = 1; // Campos não foram corretamente preenchidos
                throw new CamposInvalidos("avisar");
            }
        }catch(CamposInvalidos e){
            Component ERRO = null;
            JOptionPane.showMessageDialog(ERRO, "Preencha todos os campos antes de Cadastrar.");
        }
    }

    // Se Campos foram preenchidos corretamente e o Animal não está em extinção
    if(flagCadastro == 0 && btn_riscoextincao_nao.isSelected()){
        boolean c = false;

        if (btn_restricao_nao.isSelected()){ // If para pegar o Radio Button para restrição c
            c = true;
        }else if (btn_restricao_sim.isSelected()){
            c = false;
        }

        // Utiliza construtor da classe Animal
        Animal a = new Animal(c_raca.getText(), Float.parseFloat(c_peso.getText()), Short.parseShort(c_jaula.getText()), c_jaula.getText());
        list.cadastrarAnimal(a); // Adiciona na lista de animais
        loadTableAnimais(); // carrega na tabela

        // Caso o animal esteja em Extinção
    }else if(flagCadastro == 0 && btn_riscoextincao_sim.isSelected()){
        boolean c, d;
        c = false;

        if (btn_restricao_nao.isSelected()){ // If para pegar o Radio Button para restrição c
            c = true;
        }else if (btn_restricao_sim.isSelected()){
            c = false;
        }

        d = true; // btn_riscoextincao_sim esta selecionado
    }
}

```

→ A FLAG "flagCadastro" controla o preenchimento dos campos caso não tenha erro.

→ Exceção para quando campos não são devidamente preenchidos e o animal não está em extinção (desconsidera campos para a mesma).

→ Exceção para quando campos não são devidamente preenchidos e o animal está em extinção (considera campos para a mesma).

→ Variável "c" recebe "true" caso o animal possua alguma restrição caso contrário recebe "false".

→ Se os campos foram preenchidos corretamente e o animal não está em Extinção, utiliza a classe Animal e seu construtor.

→ Adiciona instancia na lista e na tabela.


```

// Utiliza o construtor da Classe AnimalExtinto
AnimalExtinto aExt = new AnimalExtinto(c_raca.getText(), Float.parseFloat(c_peso.getText()), she
list.cadastrarAnimalExt(aExt); // Adiciona na lista de animais extintos
LoadTableAnimaisExt(); // Carrega na tabela
}
// Limpa os campos de textos
c_raca.setText("");
c_estimativaextincao.setText("");
c_especiesrestantes.setText("");
c_jaula.setText("");
c_peso.setText("");
// Desabilita campos de textos
c_raca.setEditable(false);
c_peso.setEditable(false);
c_jaula.setEditable(false);
c_especiesrestantes.setEditable(false);
c_estimativaextincao.setEditable(false);

btn_novo.setEnabled(true); // Como já foi cadastrado, precisa ser criado um novo cadastro

// Desabilita botões
btn_restricao_nao.setEnabled(false);
btn_restricao_sim.setEnabled(false);
btn_riscoextincao_nao.setEnabled(false);
btn_riscoextincao_sim.setEnabled(false);
btn_cadastrar.setEnabled(false);
btn_cancelar.setEnabled(false);
// Desmarca caixas de seleção dos botões
buttonGroup1.clearSelection();
buttonGroup2.clearSelection();

```

→ Variável "c" recebe "true" caso o animal possua alguma restrição caso contrário recebe "false".

→ Variável "d" recebe "true" já que o animal está em extinção.

→ Se os campos foram preenchidos corretamente e o animal esta em Extinção, utiliza a classe AnimalExtinto e seu construtor, que utiliza o construtor da classe Animal(super).

→ Adiciona instancia na lista e na tabela.

↳ Quando animal é cadastrado os campos de texto são limpos e desabilitados. O mesmo acontece com os botões que são desabilitados e desmarcados.

↳ Apenas o botão NOVO é habilitado, para que possa ser feito um novo cadastro

```

/***** APAGAR (ANIMAL EXTINTO) *****/
private void btn_apagarActionPerformed(java.awt.event.ActionEvent evt) {
    int index = tbl_animaisext.getSelectedRow(); // index recebe linha da tabela
    if(index >= 0 && index<list.numeroDeAnimaisExt()){ // Se index estiver entre inicio da
        list.ListAnimalExt.remove(index); // Remove da lista
    }
    LoadTableAnimaisExt(); // Carrega na Tabela
}

/***** APAGAR (ANIMAL) *****/
private void btn_apagar2ActionPerformed(java.awt.event.ActionEvent evt) {
    int index = tbl_animais.getSelectedRow(); // index recebe linha da tabela
    if(index >= 0 && index<list.numeroDeAnimais()){ // Se index estiver entre inicio da tab
        list.ListAnimal.remove(index); // Remove da lista
    }
    LoadTableAnimais(); // Carrega na Tabela
}

```

- ↳ Método para apagar um cadastro de Animal ou Animal Extinto.
- ↳ FLAG "index" recebe a linha selecionada da tabela que será apagada.
- ↳ Se a linha for maior ou igual a linha 0 e menor do que o tamanho máximo de linhas, o animal é removido da sua respectiva lista e a tabela é atualizada.


```

/***** BUSCAR *****/
private void btn_buscarActionPerformed(java.awt.event.ActionEvent evt) {

    boolean achou = false;        // Flag para quando acha na Lista
    short achouTabela;            // Flag para receber o campo da BUSCA

    achouTabela = Short.parseShort(c_buscar.getText());    // Flag recebe

    // Percorre Lista do AnimalExtinto
    for(AnimalExtinto aEx: list.ListAnimalExt){
        // SE o campo de BUSCA for igual a jaula do Animal contido na lista
        if (aEx.getJaula() == Short.parseShort(c_buscar.getText())){
            achou = true;        // Achou o Animal na Lista
            achouTabela = Short.parseShort(c_buscar.getText());    // Flag
        }
    }
    if(achou){
        // Encontrou elemento na lista, procura ele na tabela
        for(int i = 0; i < tbl_animaisext.getRowCount(); ++i){    // Varre
            if(achouTabela == (short)tbl_animaisext.getValueAt(i, 2)){
                tbl_animaisext.setRowSelectionInterval(i,i);    // Seleção
                break;
            }
        }
    }

    // Exceção para Animal não encontrado na Lista
    try{
        if(achou == false){
            throw new AnimalNaoEncontrado("Não encontrado");
        }

    }catch(AnimalNaoEncontrado e){
        Component ERRO = null;
        JOptionPane.showMessageDialog(ERRO, "Jaula Vazia.");
    }

    // Exceção para lista de animais vazia
    try{
        if(list.numeroDeAnimaisExt() == 0){
            throw new BaseDeDadosVazia("avisar");
        }
    }catch(BaseDeDadosVazia e){
        Component ERRO = null;
        JOptionPane.showMessageDialog(ERRO, "Base de dados do Zoológico está vazio.");
    }
}

```

→ Metodo para BUSCAR Animal Extinto através de sua jaula. Duas FLAGS são criadas, uma para receber o campo da busca e uma para validar se animal foi encontrado na lista.

→ Percorre a Lista de Animais Extintos, se o campo da BUSCA for igual a Jaula do animal contido na lista, FLAG é setada para "true" e a outra FLAG recebe o campo da busca.

→ SE o elemento foi encontrado na lista ele é procurado na tabela.

→ Um for para percorrer todas as linhas da coluna 2 (jaula) até encontrar a jaula solicitada.

→ Quando a jaula é encontrada, a linha da tabela fica selecionada e o laço é encerrado.

→ Exceções caso Animal não seja encontrado em sua lista e caso a lista esteja vazia.

```

private void btn_buscar2ActionPerformed(java.awt.event.ActionEvent evt) {

    boolean achou = false;        // Flag para quando acha na Lista
    short achouTabela;            // Flag para receber o campo da BUSCA

    achouTabela = Short.parseShort(c_buscar2.getText());    // Flag recebe

    // Percorre Lista do AnimalExtinto
    for(Animal a: list.ListAnimal){
        // SE o campo de BUSCA for igual a jaula do Animal contido na lista
        if (a.getJaula() == Short.parseShort(c_buscar2.getText())){
            achou = true;        // Achou o Animal na Lista
            achouTabela = Short.parseShort(c_buscar2.getText());    // Flag
        }
    }
    if(achou){
        // Encontrou elemento na lista, procura ele na tabela
        for(int i = 0; i < tbl_animais.getRowCount(); ++i){    // Varredura
            if(achouTabela == (short)tbl_animais.getValueAt(i, 2)){
                tbl_animais.setRowSelectionInterval(i,i);    // Seleciona
                break;
            }
        }
    }

    // Exceção para Animal não encontrado na Lista
    try{
        if(achou == false){
            throw new AnimalNaoEncontrado("Não encontrado");
        }

    }catch(AnimalNaoEncontrado e){
        Component ERRO = null;
        JOptionPane.showMessageDialog(ERRO, "Jaula vazia.");
    }

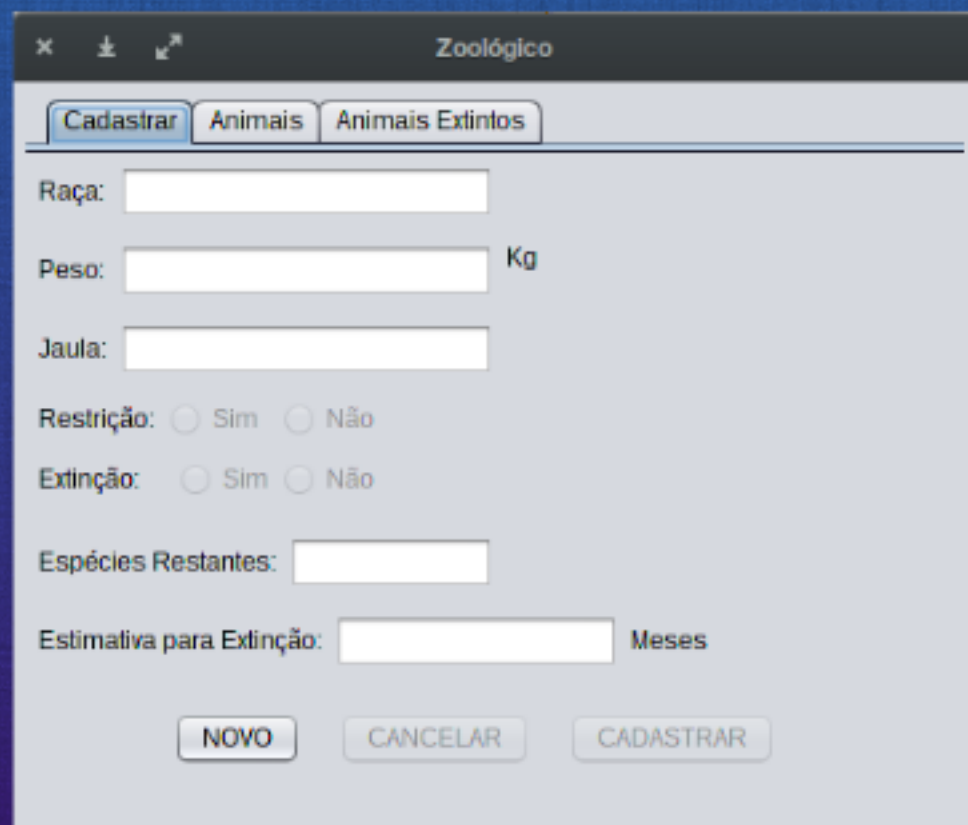
    // Exceção para lista de animais vazia
    try{
        if(list.numeroDeAnimais() == 0){
            throw new BaseDeDadosVazia("avisar");
        }
    }catch(BaseDeDadosVazia e){
        Component ERRO = null;
        JOptionPane.showMessageDialog(ERRO, "Base de dados do Zoológico está vazio.");
    }
}

```

→ O Método para buscar um Animal que não está em extinção é semelhante ao do animal que está em Extinção.

→ A única diferença é que a busca é feita na lista "ListAnimal" ao invés de "ListAnimalExt" e a tabela utilizada é "tbl_animais" ao invés de "tbl_animaisext".

⇒ Demonstração do Trabalho:



A screenshot of a software application window titled "Zoológico". The window has a dark header bar with standard window controls (close, maximize, and a zoom icon) on the left. Below the header, there is a tabbed interface with three tabs: "Cadastrar" (highlighted in blue), "Animais", and "Animais Extintos". The "Cadastrar" tab contains a form with the following fields and controls:

- Raça:** A text input field.
- Peso:** A text input field followed by the unit "Kg".
- Jaula:** A text input field.
- Restrição:** Two radio buttons labeled "Sim" and "Não".
- Extinção:** Two radio buttons labeled "Sim" and "Não".
- Espécies Restantes:** A text input field.
- Estimativa para Extinção:** A text input field followed by the unit "Meses".

At the bottom of the form, there are three buttons: "NOVO", "CANCELAR", and "CADASTRAR".