Linguagem de Programação II - Entrega 4 - Pedro Barbosa de Souza

controle.ControladorCadastroPersonagens

```
package controle;

import interfaces.JanelaCadastrarPersonagens;
import entidade.Personagem;

public class ControladorCadastroPersonagens {

    public ControladorCadastroPersonagens(){
        new JanelaCadastrarPersonagens(this).setVisible(true);
    }

    public String inserirPersonagem (Personagem personagem) {
        if (!Personagem.existePersonagem
(personagem.getBolsa().getSequencial(),personagem.getSerMagico().getNome())) {
            return Personagem.inserirPersonagem (personagem);
        } else return "Personagem já cadastrado";
    }

    public String alterarPersonagem(Personagem personagem_informada) {
        Personagem personagens_cadastradas =
Personagem.buscarPersonagem(personagem_informada.getSequencial() + 1);


        if
((personagem_informada.getSerMagico().getNome().equals(personagens_cadastradas.get
SerMagico().getNome()))
```

```
        && (personagem_informada.getBolsa().getSequencial() ==
personagens_cadastradas.getBolsa().getSequencial())) {

        return Personagem.alterarPersonagem (personagem_informada);

        } else return "Alteração não permitida : chave do Ser e/ou da bolsa foram
alteradas";

    }


    public String removerPersonagem (int sequencial) {

        if (Personagem.existePersonagem(sequencial)) return
Personagem.removerPersonagem (sequencial);

        else return "Sequencial não corresponde a nenhuma reserva cadastrada";

    }

}
```

controle.ControladorCadastroBolsas

```
package controle;

import entidade.Bolsa;

import interfaces.JanelaCadastrarBolsas;


public class ControladorCadastroBolsas {


   public ControladorCadastroBolsas(){

       System.out.println("Ele entra aqui");

      new JanelaCadastrarBolsas(this).setVisible(true);

   }
```

```
    public String inserirBolsa(Bolsa bolsa) {
        if (!Bolsa.existeBolsaMesmosAtributos(bolsa)) {
            return Bolsa.inserirBolsa(bolsa);
        } else {
            return "Já existe uma bolsa com os mesmos atributos";
        }
    }

    public String alterarBolsa(Bolsa bolsa) {
        Bolsa bolsa1 = Bolsa.buscarBolsa(bolsa.getSequencial());
        if (bolsa1 != null) {
            return Bolsa.alterarBolsa(bolsa);
        } else {
            return "Bolsa não cadastrada";}
    }

    public String removerBolsa(int sequencial) {
        Bolsa bolsa1 = Bolsa.buscarBolsa(sequencial);
        if (bolsa1 != null) {
            return Bolsa.removerBolsa(sequencial);
        } else {
            return "bolsa não cadastrado";
        }
    }
}
```

controle.ControladorCadastroItens

```
package controle;
```

```
import entidade.Item;
import interfaces.JanelaCadastrarItens;

public class ControladorCadastroItens {

   public ControladorCadastroItens(){
      new JanelaCadastrarItens(this).setVisible(true);
   }



   public String inserirItem (Item item) {
      Item item1 = Item.buscarItem(item.getNome());
      if (item1 == null) return Item.inserirItem(item);
      else return "Nome de item já cadastrado";
   }

   public String alterarItem(Item item) {
      Item item1 = Item.buscarItem(item.getNome());
      if (item1 != null) return Item.alterarItem(item);
      else return "Nome de item não cadastrado";
   }

   public String removerItem (String nome) {
      Item item1 = Item.buscarItem(nome);
      if (item1 != null) return Item.removerItem(nome);
      else return "Nome de item não cadastrado";
   }
}
  controle.ControladorCadastroPossuirBolsas
```

```
package controle;


import entidade.Possuir;
import interfaces.JanelaCadastrarPossuir;
import interfaces.JanelaCadastrarBolsas;

public class ControladorCadastroPossuirBolsas {

   public ControladorCadastroPossuirBolsas(
         JanelaCadastrarBolsas janela_cadastro_possuir, int sequencial_bolsa) {
      new JanelaCadastrarPossuir(this,
            janela_cadastro_possuir, sequencial_bolsa).setVisible(true);
   }

   public String removerPossuir(int sequencial) {
      boolean existe_possuir = Possuir.existePossuir(sequencial);
      if (existe_possuir) {
         return Possuir.removerPossuir(sequencial);
      } else {
         return "Posse não cadastrada";
      }
   }

   public String inserirPossuir(Possuir possuir) {
      boolean existe_possuir = Possuir.existePossuir(possuir.getBolsa().getSequencial(),
            possuir.getItem().getNome());
      if (!existe_possuir) {
         return Possuir.inserirPossuir(possuir);
```

```
        } else {
            return "Sequencial de posse já cadastrado";
        }
    }
}
```

controle.ControladorCadastroSeresMágicos

```
package controle;
import entidade.SerMágico;
import interfaces.JanelaCadastrarSeresMágicos;

public class ControladorCadastroSeresMágicos {

    public ControladorCadastroSeresMágicos(){
        new JanelaCadastrarSeresMágicos(this).setVisible(true);
    }

    public String inserirSerMágico (SerMágico ser_mágico) {
        SerMágico ser_mágico1 = SerMágico.buscarSerMágico(ser_mágico.getNome());
        if (ser_mágico1 == null) return SerMágico.inserirSerMágico(ser_mágico);
        else return "Nome de ser mágico já cadastrado";
    }

    public String alterarSerMágico(SerMágico ser_mágico) {
        SerMágico ser_mágico1 = SerMágico.buscarSerMágico(ser_mágico.getNome());
        if (ser_mágico1 != null) return SerMágico.alterarSerMágico(ser_mágico);
        else return "Nome de ser mágico não cadastrado";
    }
```

```
    public String removerSerMágico (SerMágico ser_mágico) {
        SerMágico ser_mágico1 = SerMágico.buscarSerMágico(ser_mágico.getNome());
        if (ser_mágico1 != null) return SerMágico.removerSerMágico(ser_mágico1);
        else return "Nome de ser mágico não cadastrado";
    }
}
```

entidade.Bolsa

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package entidade;

import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import persistência.BD;


public class Bolsa {
    public enum Qualidade {Ruim, Regular, Bom, Ótimo, Prestigio, Lendário, Herói};

    private int sequencial;
    private String peso_max;
```

```
private Qualidade qualidade;

public Bolsa(int sequencial, String peso_max, Qualidade qualidade){
   this.sequencial = sequencial;
   this.peso_max = peso_max;
   this.qualidade = qualidade;

}

   public Bolsa(int sequencial, String peso_max) {
   this.sequencial = sequencial;
   this.peso_max = peso_max;
}

public static int últimoSequencial() {
   String sql = "SELECT MAX(sequencial) FROM Bolsa";
   ResultSet lista_resultados = null;
   int sequencial = 0;
   try {
      PreparedStatement comando = BD.conexão.prepareStatement(sql);
      lista_resultados = comando.executeQuery();
      while (lista_resultados.next()) {
         sequencial = lista_resultados.getInt(1);
      }
      lista_resultados.close();
      comando.close();
   } catch (SQLException exceção_sql) {
      exceção_sql.printStackTrace();
   }
```

```
        return sequencial;
    }

    public static Bolsa[] getVisões() {
        String sql = "SELECT Sequencial, Peso_Max FROM Bolsas";
        ResultSet lista_resultados = null;
        ArrayList<Bolsa> visões = new ArrayList();
        try {
            PreparedStatement comando = BD.conexão.prepareStatement(sql);
            lista_resultados = comando.executeQuery();
            while (lista_resultados.next()) {
                visões.add(new Bolsa(lista_resultados.getInt("Sequencial"),
lista_resultados.getString("Peso_Max")));
            }
            lista_resultados.close();
            comando.close();
        } catch (SQLException exceção_sql) {
            exceção_sql.printStackTrace();
        }
        return visões.toArray(new Bolsa[visões.size()]);
    }


    public static Boolean existeBolsaMesmosAtributos(Bolsa bolsa) {
        String sql = "SELECT COUNT(Sequencial) FROM Bolsas WHERE Peso_Max = ? AND
Qualidade = ?";
        ResultSet lista_resultados = null;
        int n_bolsas_mesmos_atributos = 0;
        try {
            PreparedStatement comando = BD.conexão.prepareStatement(sql);
```

```
        comando.setString(1, bolsa.peso_max);
        comando.setString(2, bolsa.qualidade.toString());
        lista_resultados = comando.executeQuery();
        while (lista_resultados.next()) {
           n_bolsas_mesmos_atributos = lista_resultados.getInt(1);
        }
        lista_resultados.close();
        comando.close();
     } catch (SQLException exceção_sql) {
        exceção_sql.printStackTrace();
     }
     if (n_bolsas_mesmos_atributos > 0) {
        return true;
     } else {
        return false;
     }
  }

   public static String inserirBolsa(Bolsa bolsa) {
     String sql = "INSERT INTO Bolsas (sequencial, Peso_Max, Qualidade) VALUES (?, ?,?)";
     try {
        PreparedStatement comando = BD.conexão.prepareStatement(sql);
        comando.setInt(1, bolsa.getSequencial());
        comando.setString(2, bolsa.getPesoMax());
        comando.setInt(3, bolsa.getQualidade().ordinal());
        comando.execute();
        comando.close();
        return null;
     } catch (SQLException exceção) {
```

```
            exceção.printStackTrace();
            return "Erro na inserção da bolsa no BD";
        }
    }

    public static String alterarBolsa(Bolsa bolsa) {
    String sql = "UPDATE Bolsas SET Peso_Max = ?, Qualidade = ? WHERE sequencial = ?";
    try {
        PreparedStatement comando = BD.conexão.prepareStatement(sql);
        comando.setString(1, bolsa.getPesoMax());
        comando.setInt(2, bolsa.getQualidade().ordinal());
        comando.setInt(3, bolsa.getSequencial());
        comando.executeUpdate();
        comando.close();
        return null;
    } catch (SQLException exceção) {
        exceção.printStackTrace();
        return "Erro na alteração da Bolsa no BD";
    }
    }

    public static String removerBolsa(int sequencial) {
    String sqlDeletePossuir = "DELETE FROM Possuir WHERE BolsaId = ?";
    String sqlDeleteBolsa = "DELETE FROM Bolsas WHERE sequencial = ?";
    try {
        PreparedStatement comando = BD.conexão.prepareStatement(sqlDeletePossuir);
        comando.setInt(1, sequencial);
        comando.executeUpdate();
```

```
            comando.close();

            comando = BD.conexão.prepareStatement(sqlDeleteBolsa);
            comando.setInt(1, sequencial);
            comando.executeUpdate();
            comando.close();

            return null;
        } catch (SQLException exceção) {
            exceção.printStackTrace();
            return "Erro na exclusão da bolsa e seus itens no BD";
        }
    }


     public static Bolsa buscarBolsa(int sequencial) {
        String sql = "SELECT * FROM Bolsas WHERE sequencial = ?";
        ResultSet lista_resultados = null;
        Bolsa bolsa = null;

        try {
            PreparedStatement comando = BD.conexão.prepareStatement(sql);
            comando.setInt(1, sequencial);
            lista_resultados = comando.executeQuery();
            while (lista_resultados.next()) {
                bolsa = new Bolsa(lista_resultados.getInt("sequencial"),
                        lista_resultados.getString("Peso_Max"),
                        Qualidade.values()[lista_resultados.getInt("Qualidade")]);
            }
```

```
            lista_resultados.close();
            comando.close();
        } catch (SQLException exceção) {
            exceção.printStackTrace();
            bolsa = null;
        }
        return bolsa;
    }

    public Bolsa getVisão() {
        return new Bolsa(sequencial, peso_max);
    }

    public String toString() {
        return "[" + getSequencial() + "] " + getPesoMax() + "Kg";
    }
  public String toStringFull() {
        return  "[" + sequencial + "] - peso Máximo: " + peso_max+ " Kg ";
    }


    public int getSequencial() {
        return sequencial;
    }

    public void setSequencial(int sequencial) {
        this.sequencial = sequencial;
    }
```

```
    public String getPesoMax() {
        return peso_max;
    }

    public Qualidade getQualidade(){
        return qualidade;
    }

    public void setNome(String peso_max) {
        this.peso_max = peso_max;
    }

}
```

entidade.Item

```
package entidade;

import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import persistência.BD;
```

```
public class Item {
   private String nome, peso, valor, quantidade;
   private Boolean consumivel;



   public Item(String nome, String peso, String valor, String quantidade, Boolean
consumivel) {
      this.nome = nome;
      this.peso = peso;
      this.valor = valor;
      this.quantidade = quantidade;
      this.consumivel = consumivel;
   }

      public Item(String nome, String peso) {
      this.nome = nome;
      this.peso = peso;
   }

     public static Item[] getVisões() {
      String sql = "SELECT Nome, Peso FROM Itens";
      ResultSet lista_resultados = null;
      ArrayList<Item> visões = new ArrayList();
      try {
         PreparedStatement comando = BD.conexão.prepareStatement(sql);
         lista_resultados = comando.executeQuery();
         while (lista_resultados.next()) {
            String nome = lista_resultados.getString("Nome");
            String peso = lista_resultados.getString("Peso");
            visões.add(new Item(nome,peso));
```

```
        }
        lista_resultados.close();
        comando.close();
    } catch (SQLException exceção_sql) {
        exceção_sql.printStackTrace ();
    }
    return visões.toArray(new Item[visões.size()]);
    }

public static Item buscarItem (String nome) {
    String sql = "SELECT * FROM Itens WHERE Nome = ?";
    ResultSet lista_resultados = null;
    Item item = null;
    try {
        PreparedStatement comando = BD.conexão.prepareStatement(sql);
        comando.setString(1, nome);
        lista_resultados = comando.executeQuery();
        while (lista_resultados.next()) {
            item = new Item (lista_resultados.getString("nome"),
            lista_resultados.getString("Valor"),
            lista_resultados.getString("Peso"),
            lista_resultados.getString("Quantidade"),
            lista_resultados.getBoolean("Consumivel"));
        }
        lista_resultados.close();
        comando.close();
    } catch (SQLException exceção_sql) {
        exceção_sql.printStackTrace ();
        item = null;
```

```
        }
        return item;
    }

    public static String inserirItem(Item item) {
        String sql = "INSERT INTO Itens (Nome, Peso, Valor, Quantidade,Consumivel) VALUES
(?,?,?,?,?)";
        try {
        PreparedStatement comando = BD.conexão.prepareStatement(sql);
        comando.setString(1, item.getNome());
        comando.setString(2, item.getPeso());
        comando.setString(3, item.getValor());
        comando.setString(4, item.getQuantidade());
        comando.setBoolean(5, item.getConsumivel());
        comando.executeUpdate();
        comando.close();
        return null;
        } catch (SQLException exceção_sql) {
        exceção_sql.printStackTrace ();
        return "Erro na Inserção do Item no BD";
        }
    }

    public static String alterarItem (Item item) {
        String sql = "UPDATE Itens SET  Peso = ?, Valor = ?, Quantidade = ?, Consumivel = ?
WHERE Nome = ?";
        try {
            PreparedStatement comando = BD.conexão.prepareStatement(sql);

            comando.setString(1, item.getPeso());
```

```
        comando.setString(2, item.getValor());
        comando.setString(3, item.getQuantidade());
        comando.setBoolean(4, item.getConsumivel());
        comando.setString(5, item.getNome());
        comando.executeUpdate();
        comando.close();
        return null;
    } catch (SQLException exceção) {
        exceção.printStackTrace();
        return "Erro na Alteração do Item no BD";
    }
}

public static String removerItem (String nome) {
    String sql = "DELETE FROM Itens WHERE Nome = ?";
    try {
    PreparedStatement comando = BD.conexão.prepareStatement(sql);
    comando.setString(1, nome);
    comando.executeUpdate();
    comando.close();
    return null;
    } catch (SQLException exceção_sql) {
    exceção_sql.printStackTrace ();
    return "Erro na Remoção do Item no BD";
    }
}
```

```
    public String toString() {
        return "[" + getNome() + "] " + getPeso() + "Kg";
    }
    public String toStringFull() {
       return   " [" + nome + "] - Peso: " + peso  + " Kg  -";
    }


    public Item getVisão() {
       return new Item(nome,peso);
    }

     public String getNome(){
       return nome;
    }
      public String getPeso(){
       return peso;
    }
       public String getValor(){
       return valor;
    }
       public String getQuantidade(){
       return quantidade;
    }
       public Boolean getConsumivel(){
       return consumivel;
    }

}
```

entidade.SerMágico

```
package entidade;

import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import persistência.BD;
import entidade.SerMágicoRaçaInteligente.Habitat;
import entidade.SerMágicoRaçaInteligente.Tendência;



public class SerMágico {

   public enum HabilidadeMágica {Proeficiência, Fortalecimento, Sobrecarga, Voo};

   protected String nome;
   protected HabilidadeMágica habilidade_mágica;

   public SerMágico(String nome, HabilidadeMágica habilidade_mágica){
      this.nome = nome;
      this.habilidade_mágica = habilidade_mágica;
```

```
    }
    public SerMágico(String nome){
       this.nome = nome;
    }

    public SerMágico getVisão() {
       return new SerMágico (nome);
    }

    public static SerMágico[] getVisões() {
       System.out.println("vapo 2");
       String sql = "SELECT Nome, HabilidadeMagica FROM SeresMagicos";
       ResultSet lista_resultados = null;
       ArrayList<SerMágico> visões = new ArrayList();
        HabilidadeMágica habilidademágica = null;
       try {
          PreparedStatement comando = BD.conexão.prepareStatement(sql);
          lista_resultados = comando.executeQuery();
          while (lista_resultados.next()) {
             String nome = lista_resultados.getString("Nome");
             habilidademágica =
HabilidadeMágica.values()[lista_resultados.getInt("HabilidadeMagica")];

             visões.add(new SerMágico(nome,habilidademágica));
          }
          lista_resultados.close();
          comando.close();
       } catch (SQLException exceção_sql) {
          exceção_sql.printStackTrace ();
       }
```

```
    return visões.toArray(new SerMágico[visões.size()]);
    }

  public static SerMágico buscarSerMágico (String nome) {
    System.out.println("vapo 1");
    String sql = "SELECT * FROM SeresMagicos WHERE Nome = ?";
    ResultSet lista_resultados = null;
    HabilidadeMágica habilidademágica = null;
    try {
      PreparedStatement comando = BD.conexão.prepareStatement(sql);
      comando.setString(1, nome);
      lista_resultados = comando.executeQuery();
      while (lista_resultados.next()) {
        nome = lista_resultados.getString("Nome");
        habilidademágica =
HabilidadeMágica.values()[lista_resultados.getInt("HabilidadeMagica")];
      }
      lista_resultados.close();
      comando.close();
    } catch (SQLException exceção_sql) {
      exceção_sql.printStackTrace ();
    }
    if (nome == null) return null;

    sql = "SELECT Hostil, DanoAdicional FROM SeresMagicosMonstros " + " WHERE
SerMágicoId = ?";
    lista_resultados = null;
    try {
      PreparedStatement comando = BD.conexão.prepareStatement(sql);
      comando.setString(1, nome);
```

```
            lista_resultados = comando.executeQuery();
            while (lista_resultados.next()) {
               return new SerMágicoMonstro (nome, habilidademágica,
               lista_resultados.getBoolean("Hostil"),
               lista_resultados.getInt("DanoAdicional"));
            }
            lista_resultados.close();
            comando.close();
         } catch (SQLException exceção_sql) { exceção_sql.printStackTrace (); }
         sql = "SELECT Tendência, Habitat FROM SeresMagicosRacasInteligente" + " WHERE
SerMágicoId = ?";
         lista_resultados = null;
         try {
            PreparedStatement comando = BD.conexão.prepareStatement(sql);
            comando.setString(1, nome);
            lista_resultados = comando.executeQuery();
            while (lista_resultados.next()) {
               return new SerMágicoRaçaInteligente (nome, habilidademágica,
               Tendência.values()[lista_resultados.getInt("Tendência")],
               Habitat.values()[lista_resultados.getInt("Habitat")]);
         }
         lista_resultados.close();
         comando.close();
         } catch (SQLException exceção_sql) { exceção_sql.printStackTrace (); }

         return null;
      }

      public static String inserirSerMágico(SerMágico ser_magico) {
         String sql = "INSERT INTO SeresMagicos (Nome, HabilidadeMagica) VALUES (?,?)";
```

```
try {
   PreparedStatement comando = BD.conexão.prepareStatement(sql);
   comando.setString(1, ser_magico.getNome());
   comando.setInt(2, ser_magico.getHabilidadeMagica().ordinal());

   comando.executeUpdate();
   comando.close();
} catch (SQLException exceção_sql) {
   exceção_sql.printStackTrace ();
   return "Erro na Inserção do Ser mágico no BD";
}
   String nome = últimoNome();
   if(ser_magico instanceof SerMágicoMonstro) {
   SerMágicoMonstro serMágico_monstro = (SerMágicoMonstro) ser_magico;
   sql = "INSERT INTO SeresMagicosMonstros"
   + " (Hostil, DanoAdicional,"
   + " SerMágicoId) VALUES (?, ?, ?)";
   try {
      PreparedStatement comando = BD.conexão.prepareStatement(sql);
      comando.setBoolean(1, serMágico_monstro.getHostil());
      comando.setInt(2, serMágico_monstro.getDanoAdicional());
      comando.setString(3, nome);
      comando.executeUpdate();
      comando.close();
   } catch (SQLException exceção_sql) {
      exceção_sql.printStackTrace ();
      return "Erro na Inserção do SerMágicoMonstro no BD";
   }
} else if(ser_magico instanceof SerMágicoRaçaInteligente) {
```

```
        SerMágicoRaçaInteligente serMágico_raçaInteligente = (SerMágicoRaçaInteligente)
ser_magico;
        sql = "INSERT INTO SeresMagicosRacasInteligente"
        + " (Tendência, Habitat,"
        + " SerMágicoId) VALUES (?, ?, ?)";
        try {
           PreparedStatement comando = BD.conexão.prepareStatement(sql);
           comando.setInt(1, serMágico_raçaInteligente.getTendência().ordinal());
           comando.setInt(2, serMágico_raçaInteligente.getHabitat().ordinal());
           comando.setString(3, nome);
           comando.executeUpdate();
           comando.close();
        } catch (SQLException exceção_sql) {
           exceção_sql.printStackTrace ();
           return "Erro na Inserção do SerMágicoRaçaInteligente no BD";
         }
     }
     return null;
  }

  public static String alterarSerMágico (SerMágico ser_magico) {
     String sql = "UPDATE SeresMagicos SET HabilidadeMagica = ? WHERE Nome = ?";
     try {
        PreparedStatement comando = BD.conexão.prepareStatement(sql);
        comando.setInt(1, ser_magico.getHabilidadeMagica().ordinal());
        comando.setString(2, ser_magico.getNome());

        comando.executeUpdate();
        comando.close();
     } catch (SQLException exceção_sql) {
```

```
            exceção_sql.printStackTrace ();
            return "Erro na Alteração do Ser mágico no BD";
        }
            if(ser_magico instanceof SerMágicoMonstro) {
            SerMágicoMonstro serMagico_monstro = (SerMágicoMonstro) ser_magico;
            sql = "UPDATE SeresMagicosMonstros"
            + " SET Hostil = ?, DanoAdicional = ?"
            + " WHERE SerMágicoId = ?";
            try {
               PreparedStatement comando = BD.conexão.prepareStatement(sql);
               comando.setBoolean(1, serMagico_monstro.getHostil());
               comando.setInt(2, serMagico_monstro.getDanoAdicional());
               comando.setString(3, serMagico_monstro.getNome());
               comando.executeUpdate();
               comando.close();
            } catch (SQLException exceção_sql) {
               exceção_sql.printStackTrace ();
               return "Erro na Inserção do SerMágicoMonstro no BD";
            }
        } else if(ser_magico instanceof SerMágicoRaçaInteligente) {
            SerMágicoRaçaInteligente serMagico_raça = (SerMágicoRaçaInteligente)
ser_magico;
            sql = "UPDATE SeresMagicosRacasInteligente"
            + " SET Tendência = ?, Habitat = ?"
            + " WHERE SerMágicoId = ?";
            try {
               PreparedStatement comando = BD.conexão.prepareStatement(sql);
               comando.setInt(1, serMagico_raça.getTendência().ordinal());
               comando.setInt(2, serMagico_raça.getHabitat().ordinal());
               comando.setString(3, serMagico_raça.getNome());
```

```
            comando.executeUpdate();
            comando.close();
        } catch (SQLException exceção_sql) {
            exceção_sql.printStackTrace ();
            return "Erro na Inserção do SerMágicoRaçaInteligente no BD";
        }
    }
    return null;

}

public static String removerSerMágico (SerMágico ser_mágico) {
    String nome = ser_mágico.getNome();
    if (ser_mágico instanceof SerMágicoMonstro) {
        String sql = "DELETE FROM SeresMagicosMonstros WHERE SerMágicoId = ?";
        try {
            PreparedStatement comando = BD.conexão.prepareStatement(sql);
            comando.setString(1, nome);
            comando.executeUpdate();
            comando.close();
        } catch (SQLException exceção_sql) {
            exceção_sql.printStackTrace ();
            return "Erro na Remoção do SerMágicoMonstro do BD";
        }
    } else if (ser_mágico instanceof SerMágicoRaçaInteligente) {
        String sql = "DELETE FROM SeresMagicosRacasInteligente WHERE SerMágicoId = ?";
        try {
            PreparedStatement comando = BD.conexão.prepareStatement(sql);
            comando.setString(1, nome);
```

```
            comando.executeUpdate();
            comando.close();
        } catch (SQLException exceção_sql) {
            exceção_sql.printStackTrace ();
            return "Erro na Remoção do FilmeOriginal do BD";
        }
    }

    String sql = "DELETE FROM SeresMagicos WHERE Nome = ?";
    try {
        PreparedStatement comando = BD.conexão.prepareStatement(sql);
        comando.setString(1, nome);
        comando.executeUpdate();
        comando.close();
        return null;
    } catch (SQLException exceção_sql) {
        exceção_sql.printStackTrace ();
        return "Erro na Remoção do Ser mágico no BD";
    }

}

 public static String últimoNome() {
    String sql = "SELECT MAX(Nome) FROM SeresMagicos";
    ResultSet lista_resultados = null;
    String nome = null;
    try {
        PreparedStatement comando = BD.conexão.prepareStatement(sql);
        lista_resultados = comando.executeQuery();
```

```
        while (lista_resultados.next()) {
            nome = lista_resultados.getString(1);
        }
        lista_resultados.close();
        comando.close();
    } catch (SQLException exceção_sql) {
        exceção_sql.printStackTrace();
    }
    return nome;
}



public String getNome(){
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public HabilidadeMágica getHabilidadeMagica(){
    return habilidade_mágica;
}

public String toString(){
    return "[" + nome + "] habilidade: " + habilidade_mágica + " - ";
}
```

```
    public String toStringFull() {
        return "["+getNome()+"] Habilidade: " + habilidade_mágica;
    }


}
```

entidade.SerMágicoMonstro

```
package entidade;

public class SerMágicoMonstro extends SerMágico{
    private boolean hostil;
    private String dano_adicional;

    public SerMágicoMonstro(String nome, HabilidadeMágica habilidade_mágica, boolean
hostil, String dano_adicional) {
        super(nome, habilidade_mágica);
        this.hostil = hostil;
        this.dano_adicional = dano_adicional;
    }

    public SerMágicoMonstro getVisão () {
          return new SerMágicoMonstro (nome, habilidade_mágica, hostil, dano_adicional);
    }

    public String toString() {
```

```
        String str = "[" + nome + "] " + habilidade_mágica;
        if (hostil) str += " - Monstro hostil:"  ;
        else str += " - Monstro não hostil:"  ;
        return str;
    }

  public String toString() {
        String str = "[" + nome + "] " + habilidade_mágica;
        if (hostil) str += " - Monstro hostil:"  ;
        else str += " - Monstro não hostil:"  ;
        return str;
    }



        public Boolean getHostil(){
        return hostil;
    }

       public void setHostilMonstro(boolean hostil) {
        this.hostil = hostil;
    }
       public String getDanoAdicional(){
        return dano_adicional;
    }


}
```

Entidade.SerMágicoRaçaInteligente

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package entidade;
import entidade.SerMágico;
import entidade.SerMágico.HabilidadeMágica;



public class SerMágicoRaçaInteligente extends SerMágico{
   public enum Tendência {Caótico, Neutro, Bom};
   public enum Habitat {Montanha, Desfiladeiro, Savana, Deserto, Oceano, Floresta};

   private Tendência tendência;
   private Habitat habitat;

   public SerMágicoRaçaInteligente(String nome, HabilidadeMágica habilidade_mágica, Tendência tendência, Habitat habitat) {
      super(nome, habilidade_mágica);
      this.tendência = tendência;
      this.habitat = habitat;
   }

   public SerMágicoRaçaInteligente getVisão () {
```

```
        return new SerMágicoRaçaInteligente(nome, habilidade_mágica, tendência,
habitat);
   }

   public String toString() {
      String str = "[" + nome + "] " + habilidade_mágica;
      if (tendência != null) str += " - Tendência: "+tendência;
      return str;
   }

public String toStringFull() {
         return " " + super.toStringFull() + " - Tendência: " + tendência+ "  ";

   }

     public Tendência getTendência(){
      return tendência;
   }
       public Habitat getHabitat(){
      return habitat;
   }

    public void setTendência(Tendência tendência) {
       this.tendência = tendência;
    }
    public void setHabitat(Habitat habitat) {
       this.habitat = habitat;
    }
```

```
}
```

entidade.Possuir

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package entidade;


import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;

import persistência.BD;

public class Possuir {
   private int sequencial;
   private Bolsa bolsa;
   private Item item;

   public Possuir(int sequencial,  Bolsa bolsa,  Item item) {
      this.sequencial = sequencial;
```

```
    this.bolsa = bolsa;
    this.item = item;
}

public Possuir(int sequencial) {
    this.sequencial = sequencial;
}

public Possuir getVisão() {
    return new Possuir(sequencial, bolsa, item);
}

public static int últimoSequencial() {
    String sql = "SELECT MAX(sequencial) FROM Possuir";
    ResultSet lista_resultados = null;
    int sequencial = 0;
    try {
        PreparedStatement comando = BD.conexão.prepareStatement(sql);
        lista_resultados = comando.executeQuery();
        while (lista_resultados.next()) {
            sequencial = lista_resultados.getInt(1);
        }
        lista_resultados.close();
        comando.close();
    } catch (SQLException exceção_sql) {
        exceção_sql.printStackTrace();
    }
    return sequencial;
}
```

```
public static Possuir[] getVisões() {
   String sql = "SELECT sequencial, BolsaId, ItemId from Possuir";
   ResultSet lista_resultados = null;
   ArrayList<Possuir> visões = new ArrayList();
   try {
      PreparedStatement comando = BD.conexão.prepareStatement(sql);
      lista_resultados = comando.executeQuery();
      while (lista_resultados.next()) {
         visões.add(new Possuir(lista_resultados.getInt("sequencial"),
               Bolsa.buscarBolsa(lista_resultados.getInt("bolsaId")).getVisão(),
               Item.buscarItem(lista_resultados.getString("itemId")).getVisão()));
      }
      lista_resultados.close();
      comando.close();
   } catch (SQLException exceção_sql) {
      exceção_sql.printStackTrace();
   }
   return visões.toArray(new Possuir[visões.size()]);
}

 public static boolean existePossuir(int chave_bolsa, String chave_item) {
   String sql = "SELECT Sequencial from Possuir WHERE bolsaId = ? AND itemId = ?";
   ResultSet lista_resultados = null;
   boolean existe = false;
   try {
      PreparedStatement comando = BD.conexão.prepareStatement(sql);
      comando.setInt(1, chave_bolsa);
```

```
      comando.setString(2, chave_item);
      lista_resultados = comando.executeQuery();
      while (lista_resultados.next()) {
         existe = true;
      }
      lista_resultados.close();
      comando.close();
   } catch (SQLException exceção_sql) {
      exceção_sql.printStackTrace();
   }
   return existe;
}


  public static boolean existePossuir(int sequencial) {
   String sql = "SELECT COUNT(Sequencial) from Possuir WHERE sequencial = ?";
   ResultSet lista_resultados = null;
   boolean existe = false;
   try {
      PreparedStatement comando = BD.conexão.prepareStatement(sql);
      comando.setInt(1, sequencial);
      lista_resultados = comando.executeQuery();
      while (lista_resultados.next()) {
         existe = true;
      }
      lista_resultados.close();
      comando.close();
   } catch (SQLException exceção_sql) {
      exceção_sql.printStackTrace();
```

```
    }
    return existe;
}

   public static String inserirPossuir(Possuir possuir) {
    String sql = "INSERT INTO Possuir VALUES(?, ?, ?)";
    try {
        PreparedStatement comando = BD.conexão.prepareStatement(sql);
        comando.setInt(1, possuir.getSequencial());
        comando.setInt(2, possuir.bolsa.getSequencial());
        comando.setString(3, possuir.item.getNome());
        comando.executeUpdate();
        comando.close();
    } catch (SQLException exceção_sql) {
        exceção_sql.printStackTrace();
        return "Erro ao adicionar Possuir";
    }
    return null;
}


 public static String removerPossuir(int sequencial) {
    System.out.println("Sequencial passado para remoção: " + sequencial);
    String sql = "DELETE FROM Possuir WHERE sequencial = ?";
    try {
        PreparedStatement comando = BD.conexão.prepareStatement(sql);
        comando.setInt(1, sequencial);
        comando.executeUpdate();
        comando.close();
```

```
            return null;
        } catch (SQLException exceção_sql) {
            exceção_sql.printStackTrace();
            return "Erro ao remover possuir";
        }
    }

    public static Possuir[] buscarPossuirBolsa(int sequencial_bolsa) {
        String sql = "SELECT * FROM Possuir WHERE bolsaId = ?";
        ArrayList<Possuir> possuis = new ArrayList<>();

        try (PreparedStatement comando = BD.conexão.prepareStatement(sql)) {
            comando.setInt(1, sequencial_bolsa);
            try (ResultSet lista_resultados = comando.executeQuery()) {
                while (lista_resultados.next()) {
                    Possuir possuir = new Possuir(
                            lista_resultados.getInt("Sequencial"),
                            Bolsa.buscarBolsa(lista_resultados.getInt("bolsaId")),
                            Item.buscarItem(lista_resultados.getString("itemId"))
                    );
                    possuis.add(possuir);
                }
            }
        } catch (SQLException exceção_sql) {
            exceção_sql.printStackTrace();
            return new Possuir[0];
        }

        return possuis.toArray(new Possuir[0]);
```

```
    }


     public int getSequencial() {
        return sequencial;
    }

    public void setSequencial(int sequencial) {
        this.sequencial = sequencial;
    }

    public Bolsa getBolsa() {
        return bolsa;
    }

    public Item getItem() {
        return item;
    }

    public String toString() {
        return item.getNome() + " [" + item.getPeso() + "]";
    }
}
```

Interfaces.JanelaCadastrarBolsas

```
package interfaces;

import controle.ControladorCadastroPossuirBolsas;
import javax.swing.DefaultListModel;
```

```
import javax.swing.JOptionPane;
import controle.ControladorCadastroBolsas;
import entidade.Bolsa;
import entidade.Possuir;
import javax.swing.DefaultComboBoxModel;


public class JanelaCadastrarBolsas extends javax.swing.JFrame {
   ControladorCadastroBolsas controlador;
   Bolsa[] bolsas_cadastrados;
   DefaultListModel item_possuir_bolsa;



   public JanelaCadastrarBolsas(ControladorCadastroBolsas controlador) {
      this.controlador = controlador;
      bolsas_cadastrados = Bolsa.getVisões();

      initComponents();
      item_possuir_bolsa = (DefaultListModel) itensList.getModel();
      limparCampos();

   }
 private void limparCampos() {
      sequencialTextField.setText("");
      pesoMáximoTextField.setText("");
      item_possuir_bolsa.clear();
   }
```

```
    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")


    private Bolsa obtémBolsaInformado() {
        String sequencial_str = sequencialTextField.getText();
       int sequencial = 0;
       if (!sequencial_str.isEmpty()) {
          sequencial = Integer.parseInt(sequencial_str);
       }
       String peso = pesoMáximoTextField.getText();
       if (peso.isEmpty()) {
          return null;
       }

       return new Bolsa(sequencial, peso);
    }

     private void informarErro(String mensagem) {
       JOptionPane.showMessageDialog(this, mensagem, "Erro",
JOptionPane.ERROR_MESSAGE);
    }

    private void informarSucesso(String mensagem) {
```

```
        JOptionPane.showMessageDialog(this, mensagem, "Sucesso",
JOptionPane.INFORMATION_MESSAGE);
    }

   private Bolsa getVisãoAlterada(int sequencial) {
        for (Bolsa visão : bolsas_cadastrados) {
            if (visão.getSequencial() == sequencial) {
                return visão;
            }
        }
        return null;
    }

    public void atualizarListaPossuirBolsas(int sequencial) {
        item_possuir_bolsa.clear();
        Possuir[] possuir_bolsas = Possuir.buscarPossuirBolsa(sequencial);
        for (Possuir possuir : possuir_bolsas) {
            item_possuir_bolsa.addElement(possuir);
        }
    }

    private void pesoMáximoTextFieldActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void consultarBolsa(java.awt.event.ActionEvent evt) {
      Bolsa visão = (Bolsa) bolsas_cadastradosComboBox.getSelectedItem();
        Bolsa bolsa = null;
        String mensagem_erro = null;
        int sequencial = -1;
```

```
    if (visão != null) {
        sequencial = visão.getSequencial();
        bolsa = Bolsa.buscarBolsa(sequencial);
        if (bolsa == null) {
            mensagem_erro = "Bolsa não encontrada";
        }
    } else {
        mensagem_erro = "Nenhuma bolsa selecionada";
    }
    if (mensagem_erro == null) {
        sequencialTextField.setText(sequencial + "");
        pesoMáximoTextField.setText(String.valueOf(bolsa.getPesoMax()));

        atualizarListaPossuirBolsas(sequencial);
    } else {
        informarErro(mensagem_erro);
    }
}

private void inserirBolsa(java.awt.event.ActionEvent evt) {
        Bolsa bolsa = obtémBolsaInformado();
    String mensagem_erro = null;
    if (bolsa != null) {
        mensagem_erro = controlador.inserirBolsa(bolsa);
    } else {
        mensagem_erro = "Preencha todos os campos";
    }
    if (mensagem_erro == null) {
        int sequencial = Bolsa.últimoSequencial();
```

```
        bolsa.setSequencial(sequencial);
        Bolsa visão = bolsa.getVisão();
        bolsas_cadastradosComboBox.addItem(visão);
        bolsas_cadastradosComboBox.setSelectedItem(visão);
        sequencialTextField.setText("" + sequencial);
    } else {
        informarErro(mensagem_erro);
    }
}

private void removerBolsa(java.awt.event.ActionEvent evt) {
    Bolsa visão = (Bolsa) bolsas_cadastradosComboBox.getSelectedItem ();
    String mensagem_erro = null;
    if (visão != null) mensagem_erro = controlador.removerBolsa(visão.getSequencial());
    else mensagem_erro = "Nenhum Bolsa selecionada";
    if (mensagem_erro == null) {
        bolsas_cadastradosComboBox.removeItem(visão);
        limparCampos();
    } else informarErro (mensagem_erro);;
}

private void limparCampos(java.awt.event.ActionEvent evt) {
    limparCampos();
}

private void alterarBolsa(java.awt.event.ActionEvent evt) {
    Bolsa bolsa = obtémBolsaInformado();
    String mensagem_erro = null;
    if (bolsa != null) mensagem_erro = controlador.alterarBolsa(bolsa);
```

```
        else mensagem_erro = "Algum atributo do Bolsa não foi informado";
        if (mensagem_erro == null) {
            Bolsa visão = getVisãoAlterada(bolsa.getSequencial());
            if (visão != null) {
                bolsas_cadastradosComboBox.updateUI();
                bolsas_cadastradosComboBox.setSelectedItem(visão);
            }
        } else informarErro (mensagem_erro);
    }

    private void bolsas_cadastradosComboBoxActionPerformed(java.awt.event.ActionEvent
evt) {
        // TODO add your handling code here:
    }

    private void sequencialTextFieldActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void cadastrarPossuir(java.awt.event.ActionEvent evt) {
         // TODO add your handling code here:
        String sequencial_str = sequencialTextField.getText();
        int sequencial = 0;
        if (!sequencial_str.isEmpty()) {
            sequencial = Integer.parseInt(sequencial_str);
        }
        if (sequencial > 0){
              new ControladorCadastroPossuirBolsas(this, sequencial);}
        else
            informarErro("Nenhuma loja selecionada");
```

```
    }


    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional)
">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and
feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(JanelaCadastrarBolsas.class.getName()).log(java.util.log
ging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(JanelaCadastrarBolsas.class.getName()).log(java.util.log
ging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
```

```
java.util.logging.Logger.getLogger(JanelaCadastrarBolsas.class.getName()).log(java.util.log
ging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(JanelaCadastrarBolsas.class.getName()).log(java.util.log
ging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {


            }
        });
    }
interfaces.JanelaCadastrarItens
package interfaces;
import javax.swing.JOptionPane;
import controle.ControladorCadastroItens;
import entidade.Item;
import javax.swing.DefaultComboBoxModel;

public class JanelaCadastrarItens extends javax.swing.JFrame {

    ControladorCadastroItens controlador;
    Item[] itens_cadastrados;
```

```
public JanelaCadastrarItens(ControladorCadastroItens controlador) {
    this.controlador = controlador;
    itens_cadastrados = Item.getVisões();

    initComponents();
    limparCampos();
}

@SuppressWarnings("unchecked")

  private Item obterItemInformado() {
    String nome = nomeTextField.getText();
    if (nome.isEmpty()){
        return null;
   }
    String peso = pesoTextField.getText();

    if (peso.isEmpty()) peso = null;

    String valor = valorTextField.getText();
    if (valor.isEmpty()) {
        return null;
    }

    String quantidade = quantidadeTextField.getText();
    if (quantidade.isEmpty()){
        return null;
    }
    boolean consumivel = consumivelCheckBox.isSelected();
```

```
        return new Item(nome, peso, valor, quantidade, consumivel);
    }

    private void limparCampos() {
        nomeTextField.setText("");
        pesoTextField.setText("");
        valorTextField.setText("");
        quantidadeTextField.setText("");
        consumivelCheckBox.setSelected(false);
    }

    private void informarErro (String mensagem) {
        JOptionPane.showMessageDialog (this, mensagem, "Erro",
JOptionPane.ERROR_MESSAGE);
    }

    private void informarSucesso(String mensagem) {
        JOptionPane.showMessageDialog(this, mensagem, "Sucesso",
JOptionPane.INFORMATION_MESSAGE);
    }

    private void atualizarComboBox() {
        itens_cadastrados = Item.getVisões();
        itens_cadastradosComboBox.setModel(new
DefaultComboBoxModel(itens_cadastrados));
    }

    private Item getVisãoAlterada(String nome) {
```

```
    for(Item visão : itens_cadastrados){
        if(visão.getNome().equals(nome)){
            return visão;
        }
    }
    return null;
}


private void inserirItem(java.awt.event.ActionEvent evt) {
    Item item = obterItemInformado();
    String mensagem_erro = null;
    if (item != null) {
        mensagem_erro = controlador.inserirItem(item);
        informarSucesso("Item inserido com sucesso");
    } else {
        mensagem_erro = "Algum atributo do item não foi informado";
    }

    if (mensagem_erro == null) {
        Item visão = item.getVisão();
        itens_cadastradosComboBox.addItem(visão);
        itens_cadastradosComboBox.setSelectedItem(visão);
    } else
        informarErro(mensagem_erro);
}

private void consultarItem(java.awt.event.ActionEvent evt) {
    Item visão = (Item) itens_cadastradosComboBox.getSelectedItem();
```

```
    Item item = null;
    String mensagem_erro = null;
    if (visão != null) {
        item = Item.buscarItem(visão.getNome());
        if (item == null) {
            mensagem_erro = "item não cadastrado";
        }
    } else {
        mensagem_erro = "Nenhum item selecionado";
    }
    if (mensagem_erro == null) {
        nomeTextField.setText(item.getNome());
        String peso = item.getPeso();
        if (peso == null) {
            peso = "";
        }
        pesoTextField.setText(peso);
        valorTextField.setText(item.getValor());
        quantidadeTextField.setText(item.getQuantidade());
        consumivelCheckBox.setSelected(item.getConsumivel());
    } else
        informarErro(mensagem_erro);
}

private void alterarItem(java.awt.event.ActionEvent evt) {
    Item item = obterItemInformado();
    String mensagem_erro = null;
    if (item != null) mensagem_erro = controlador.alterarItem(item);
    else mensagem_erro = "Algum atributo do item não foi informado";
```

```
        if (mensagem_erro == null) {
            Item visão = getVisãoAlterada(item.getNome());
            if (visão != null) {
                itens_cadastradosComboBox.updateUI();
                itens_cadastradosComboBox.setSelectedItem(visão);
            }
        } else informarErro (mensagem_erro);
    }

    private void removerItem(java.awt.event.ActionEvent evt) {
        Item visão = (Item) itens_cadastradosComboBox.getSelectedItem ();
        String mensagem_erro = null;
        if (visão != null) mensagem_erro = controlador.removerItem(visão.getNome());
        else mensagem_erro = "Nenhum celular selecionado";
        if (mensagem_erro == null) {
            itens_cadastradosComboBox.removeItem(visão);
            limparCampos();
        } else informarErro (mensagem_erro);;
    }

    private void limparCampos(java.awt.event.ActionEvent evt) {
        limparCampos();
    }

    private void consumivelCheckBoxActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void itens_cadastradosComboBoxActionPerformed(java.awt.event.ActionEvent
evt) {
```

```
        // TODO add your handling code here:
    }

    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
            }
        });
    }
```

interfaces.JanelaCadastrarPossuir

```
package interfaces;

import controle.ControladorCadastroPossuirBolsas;
import entidade.Item;
import entidade.Bolsa;
import entidade.Possuir;
import javax.swing.DefaultComboBoxModel;

import javax.swing.DefaultListModel;
import javax.swing.JOptionPane;


public class JanelaCadastrarPossuir extends javax.swing.JFrame {

    ControladorCadastroPossuirBolsas controlador;
    JanelaCadastrarBolsas janela_mãe;
    int sequencial_bolsa;
```

```
Item[] itens_cadastrados;
DefaultListModel itens_lista_possuir;


public JanelaCadastrarPossuir(ControladorCadastroPossuirBolsas controlador,
      JanelaCadastrarBolsas janela_mãe, int sequencial_bolsa) {

   this.controlador = controlador;
   this.janela_mãe = janela_mãe;
   this.sequencial_bolsa = sequencial_bolsa;
   itens_cadastrados = Item.getVisões();
   initComponents();
   atualizarPesoBolsa();
   atualizarListaPossuirBolsas();
}

private void atualizarPesoBolsa() {
   Bolsa bolsa = Bolsa.buscarBolsa(sequencial_bolsa);
   bolsaLabel.setText("Bolsa: " + bolsa.getPesoMax());

}


 private void atualizarListaPossuirBolsas() {
   itens_lista_possuir = (DefaultListModel) itens_bolsasList.getModel();
   Possuir[] possuir_bolsa = Possuir.buscarPossuirBolsa(sequencial_bolsa);
   for (Possuir possuir : possuir_bolsa) {
      itens_lista_possuir.addElement(possuir);
   }
```

```
    }


     private void informarErro(String mensagem) {
       JOptionPane.showMessageDialog(this, mensagem, "Erro",
JOptionPane.ERROR_MESSAGE);
    }


    private void informarSucesso(String mensagem) {
       JOptionPane.showMessageDialog(this, mensagem, "Sucesso",
JOptionPane.INFORMATION_MESSAGE);
    }


    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
  private void InserirPossuir(java.awt.event.ActionEvent evt) {
       Item visão_item = (Item) itens_cadastradosComboBox.getSelectedItem();
       String mensagem_erro = null;
       Possuir possuir = null;
       if (visão_item != null) {
          Bolsa visão_bolsa = Bolsa.buscarBolsa(sequencial_bolsa);
          possuir = new Possuir(0, visão_bolsa, visão_item);
          mensagem_erro = controlador.inserirPossuir(possuir);
          if (mensagem_erro == null) {
             int sequencial = Possuir.últimoSequencial();
             possuir.setSequencial(sequencial);
             itens_lista_possuir.addElement(possuir);
```

```
        } else {
            informarErro(mensagem_erro);
        }
    }
}

private void removerPossuir(java.awt.event.ActionEvent evt) {
    Possuir visão = (Possuir) itens_bolsasList.getSelectedValue();
    if (visão != null) {
        int sequencial = visão.getSequencial();
        String mensagem_erro = controlador.removerPossuir(sequencial);
        if (mensagem_erro == null) {
            itens_lista_possuir.removeElement(visão);
        } else {
            informarErro(mensagem_erro);
        }
    } else {
        informarErro("Nenhum Item foi selecionado");
    }
}

private void none(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void atualizarJanelaCadastroPossuir(java.awt.event.WindowEvent evt) {
     janela_mãe.atualizarListaPossuirBolsas(sequencial_bolsa);
}
```

interfaces.JanelaCadastrarSeresMágicos

```
package interfaces;
import javax.swing.JOptionPane;
import controle.ControladorCadastroSeresMágicos;
import entidade.SerMágico;
import entidade.SerMágico.HabilidadeMágica;
import entidade.SerMágicoMonstro;
import entidade.SerMágicoRaçaInteligente;
import entidade.SerMágicoRaçaInteligente.Habitat;
import entidade.SerMágicoRaçaInteligente.Tendência;


import javax.swing.DefaultComboBoxModel;

public class JanelaCadastrarSeresMágicos extends javax.swing.JFrame {

   ControladorCadastroSeresMágicos controlador;
   SerMágico[] seres_mágicos_cadastrados;
   PainelSerMágicoMonstro serMágico_monstroPainel;
   PainelSerMágicoRaçaInteligente serMágico_raçaInteligentePainel;

   public JanelaCadastrarSeresMágicos(ControladorCadastroSeresMágicos controlador) {
      this.controlador = controlador;
      seres_mágicos_cadastrados = SerMágico.getVisões();
      initComponents();
      serMágico_monstroPainel = new PainelSerMágicoMonstro();
```

```
      serMágico_raçaInteligentePainel = new PainelSerMágicoRaçaInteligente();
      especialização_serMagicoTabbedPane.addTab("Ser Mágico
Monstro",serMágico_monstroPainel);
      especialização_serMagicoTabbedPane.addTab("Ser Mágico Raça
Inteligente",serMágico_raçaInteligentePainel);
      limparCampos(null);
   }

   @SuppressWarnings("unchecked")


   private SerMágico obterSerMágicoInformado() {
      String nome = nomeTextField.getText();
      if (nome.isEmpty()){
         return null;
     }
      HabilidadeMágica habilidade = null;
      if (habilidadeButtonGroup.getSelection() != null)
      habilidade =
HabilidadeMágica.values()[habilidadeButtonGroup.getSelection().getMnemonic()];
      else return null;


      SerMágico sermágico = null;
      int índice_aba_secionada = especialização_serMagicoTabbedPane.getSelectedIndex();
      switch (índice_aba_secionada) {
         case 0:
            boolean hostilSerMágico = serMágico_monstroPainel.isHostilMonstro();
            String danoAdicionalSerMágico =
serMágico_monstroPainel.getDanoAdicionalMonstro();
            sermágico = new
SerMágicoMonstro(nome,habilidade,hostilSerMágico,danoAdicionalSerMágico);
```

```
            break;
        case 1:
            Tendência tendênciaRaça =
serMágico_raçaInteligentePainel.getSelectedTendencia();
            Habitat habitatRaça = serMágico_raçaInteligentePainel.getSelectedhabitat();
            sermágico = new
SerMágicoRaçaInteligente(nome,habilidade,tendênciaRaça,habitatRaça);
    }
      return sermágico;
    }

    private void informarErro (String mensagem){
      JOptionPane.showMessageDialog (this, mensagem, "Erro",
JOptionPane.ERROR_MESSAGE);
    }

    private SerMágico getVisãoAlterada(String nome) {
       for(SerMágico visão : seres_mágicos_cadastrados){
          if(visão.getNome().equals(nome)){
             return visão;
          }
       }
       return null;
    }

    private void limparCampos() {
      nomeTextField.setText("");
      habilidadeButtonGroup.clearSelection();
      serMágico_monstroPainel.limparCampos();
      serMágico_raçaInteligentePainel.limparCampos();
```

```
    }

    private void selecionarHabilidadeRadioButton(int índice_habilidade_mágica) {
        switch(índice_habilidade_mágica) {
            case 0: proeficiênciaRadioButton.setSelected(true);
            break;
            case 1: fortalecimentoRadioButton.setSelected(true);
            break;
            case 2: sobreCargaRadioButton.setSelected(true);
            break;
            case 3: vooRadioButton.setSelected(true);
        }
    }


    private void removerSerMágico(java.awt.event.ActionEvent evt) {
        SerMágico visão = (SerMágico)
seres_magicos_cadastradosComboBox.getSelectedItem ();
        String mensagem_erro = null;
        if (visão != null) mensagem_erro = controlador.removerSerMágico(visão);
        else mensagem_erro = "Nenhum Ser Mágico selecionado";
        if (mensagem_erro == null) {
            seres_magicos_cadastradosComboBox.removeItem(visão);
            limparCampos();
        } else informarErro (mensagem_erro);;
    }

    private void alterarSerMágico(java.awt.event.ActionEvent evt) {
        SerMágico ser_mágico = obterSerMágicoInformado();
        String mensagem_erro = null;
```

```
    if (ser_mágico != null) mensagem_erro = controlador.alterarSerMágico(ser_mágico);
    else mensagem_erro = "Algum atributo do Ser Mágico não foi informado";
    if (mensagem_erro == null) {
       SerMágico visão = getVisãoAlterada(ser_mágico.getNome());
       if (visão != null) {
          visão.setNome(ser_mágico.getNome());
          if (ser_mágico instanceof SerMágicoMonstro) {
             SerMágicoMonstro serMágico_monstro = (SerMágicoMonstro) ser_mágico;
             SerMágicoMonstro visão_monstro   = (SerMágicoMonstro) visão;

             visão_monstro.setHostilMonstro(serMágico_monstro.getHostil());

          } else if(ser_mágico instanceof SerMágicoRaçaInteligente) {
             SerMágicoRaçaInteligente serMágico_raça = (SerMágicoRaçaInteligente)
ser_mágico;
             SerMágicoRaçaInteligente visão_raça = (SerMágicoRaçaInteligente) visão;

             visão_raça.setTendência(serMágico_raça.getTendência());
             visão_raça.setHabitat(serMágico_raça.getHabitat());
          }

          seres_magicos_cadastradosComboBox.updateUI();


       }
    } else informarErro (mensagem_erro);
  }

  private void consultarSerMágico(java.awt.event.ActionEvent evt) {
```

```
      SerMágico visão = (SerMágico)
seres_magicos_cadastradosComboBox.getSelectedItem ();
      SerMágico ser_mágico = null;
      String mensagem_erro = null;
      if (visão != null) {
         ser_mágico = SerMágico.buscarSerMágico (visão.getNome());
         if (ser_mágico == null) mensagem_erro = "Ser Mágico não cadastrado";
      } else mensagem_erro = "Nenhum Ser Mágico selecionado";
      if (mensagem_erro == null) {
         nomeTextField.setText(ser_mágico.getNome());
         selecionarHabilidadeRadioButton(ser_mágico.getHabilidadeMagica().ordinal());

         if(ser_mágico instanceof SerMágicoMonstro) {
            especialização_serMagicoTabbedPane.setSelectedIndex(0);
            SerMágicoMonstro serMágico_monstro = (SerMágicoMonstro) ser_mágico;

            serMágico_monstroPainel.setHostilMonstro
            (serMágico_monstro.getHostil());

            serMágico_monstroPainel.setDanoAdicionalMonstro
            (serMágico_monstro.getDanoAdicional());

         } else if (ser_mágico instanceof SerMágicoRaçaInteligente) {
            especialização_serMagicoTabbedPane.setSelectedIndex(1);
            SerMágicoRaçaInteligente serMágico_raça = (SerMágicoRaçaInteligente)
ser_mágico;

            serMágico_raçaInteligentePainel.setSelectedTendencia
            (serMágico_raça.getTendência().ordinal());
```

```
            serMágico_raçaInteligentePainel.setSelectedgetSelectedhabitat
            (serMágico_raça.getHabitat());
        }
    } else informarErro (mensagem_erro);


}

private void inserirSerMágico(java.awt.event.ActionEvent evt) {
    SerMágico ser_mágico = obterSerMágicoInformado();
    String mensagem_erro = null;
    if (ser_mágico != null) mensagem_erro = controlador.inserirSerMágico(ser_mágico);
    else mensagem_erro = "Algum atributo do Ser Mágico não foi informado";

    if (mensagem_erro == null) {
        SerMágico visão = ser_mágico.getVisão();
        seres_magicos_cadastradosComboBox.addItem(visão);
        seres_magicos_cadastradosComboBox.setSelectedItem(visão);
    } else informarErro (mensagem_erro);
}

private void nomeTextFieldActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void
seres_magicos_cadastradosComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}
```

```
    private void proeficiênciaRadioButtonActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
    }

    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
            }
        });
    }
```

interfaces.PainelSerMágicoMonstro

```
package interfaces;



public class PainelSerMágicoMonstro extends javax.swing.JPanel {

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

public boolean isHostilMonstro() { return hostilCheckBox.isSelected(); }

public void setHostilMonstro(boolean hostil_monstro) {
```

```
    hostilCheckBox.setSelected(hostil_monstro);
}

public String getDanoAdicionalMonstro() {
    String dano = danoAdicionalTextField.getText();
    if (dano.isEmpty()) return null;
    else return dano;
}
public void setDanoAdicionalMonstro(String dano) {
danoAdicionalTextField.setText(dano); }



    public void limparCampos() {
        hostilCheckBox.setSelected(false);
        danoAdicionalTextField.setText("");
    }
    @SuppressWarnings("unchecked")



    private void hostilCheckBox(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void danoAdicionalTextField(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

interfaces.PainelSerMágicoRaçaInteligente

package interfaces;
```

```
import entidade.SerMágicoRaçaInteligente.Habitat;
import entidade.SerMágicoRaçaInteligente.Tendência;
import javax.swing.DefaultComboBoxModel;


public class PainelSerMágicoRaçaInteligente extends javax.swing.JPanel {

    /**
     * Creates new form PainelSerMágicoRaçaInteligente
     */
    public PainelSerMágicoRaçaInteligente() {
        initComponents();
    }
    public Habitat getSelectedhabitat() {
        Object habitat = habitatComboBox.getSelectedItem();
        if (habitat != null) return (Habitat) habitat;
        else return null;
    }
    public void setSelectedgetSelectedhabitat(Habitat habitat) {
    habitatComboBox.setSelectedItem(habitat);
    }

    public Tendência getSelectedTendencia() {
        Tendência tendencia = null;
        if (tendenciaButtonGroup.getSelection() != null)
            tendencia =
Tendência.values()[tendenciaButtonGroup.getSelection().getMnemonic()];
        return tendencia;
    }
```

```
public void setSelectedTendencia(int índice_tendencia) {
    switch(índice_tendencia) {
        case 0: caóticoRadioButton.setSelected(true); break;
        case 1: neutroRadioButton.setSelected(true);
        case 2: bomRadioButton.setSelected(true);
    }
}

public void limparCampos() {
    habitatComboBox.setSelectedIndex(-1);
    tendenciaButtonGroup.clearSelection();
}

@SuppressWarnings("unchecked")
private void caóticoRadioButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void neutroRadioButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void habitatComboBox(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void bomRadioButton(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}
```

```
interfaces.JanelaPersonagensRpg
 */
package interfaces;

import persistência.BD;
import javax.swing.JOptionPane;
import controle.ControladorCadastroItens;
import controle.ControladorCadastroSeresMágicos;
import controle.ControladorCadastroBolsas;
/**
 *
 * @author alvaro.olazar
 */
public class JanelaPersonagensRpg extends javax.swing.JFrame {

    /**
     * Creates new form JanelaNovosCelulares
     */
    public JanelaPersonagensRpg() {
        BD.criaConexão();
        initComponents();
    }

    @SuppressWarnings("unchecked")
    private void terminarSistema(java.awt.event.WindowEvent evt) {
```

```
    BD.fechaConexão();
    System.exit(0);
}

private void informarServiçoIndisponível() {
  JOptionPane.showMessageDialog (this, "Serviço Indisponível", "Informação",
  JOptionPane.INFORMATION_MESSAGE);
}

private void pesquisarPersonagens(java.awt.event.ActionEvent evt) {
    informarServiçoIndisponível();
}

private void cadastrarItem(java.awt.event.ActionEvent evt) {
    new ControladorCadastroItens();
}

private void cadastrarBolsa(java.awt.event.ActionEvent evt) {
    new ControladorCadastroBolsas();
}

private void cadastrarPersonagem(java.awt.event.ActionEvent evt) {
    informarServiçoIndisponível();
}

private void cadastrarSerMágico(java.awt.event.ActionEvent evt) {
    new ControladorCadastroSeresMágicos();
}
```

```
    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional)
">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and
feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(JanelaPersonagensRpg.class.getName()).log(java.util.log
ging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(JanelaPersonagensRpg.class.getName()).log(java.util.log
ging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(JanelaPersonagensRpg.class.getName()).log(java.util.log
ging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
```

```
java.util.logging.Logger.getLogger(JanelaPersonagensRpg.class.getName()).log(java.util.log
ging.Level.SEVERE, null, ex);
        }
        //</editor-fold>
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new JanelaPersonagensRpg().setVisible(true);
            }
        });
    }
```

persistência.BD

```
package persistência;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class BD {
    static final String URL_BD = "jdbc:mysql://localhost/ENTREGA";
    static final String USUÁRIO = "root";
    static final String SENHA = "admin";
    public static Connection conexão = null;

    public static void criaConexão () {
        try {
```

```
      conexão = DriverManager.getConnection (URL_BD, USUÁRIO, SENHA);
      } catch (SQLException exceção_sql) {exceção_sql.printStackTrace ();}
   }
   public static void fechaConexão () {
      try {
      conexão.close();
      } catch (SQLException exceção_sql) {exceção_sql.printStackTrace ();}
   }
}
```

entidade.Personagem

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package entidade;

import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import persistência.BD;

import entidade.SerMágico.HabilidadeMágica;
import entidade.SerMágicoRaçaInteligente.Tendência;
```

```
public class Personagem {

   public enum Classe {Barbaro,Guerreiro,Mago,Monge, Sarcedote};

   private String nome;
   private int sequencial, nivel, pontos_vida, pontos_forca;
   private Classe classe;
   private Bolsa bolsa;
   private SerMágico serMágico;

   public Personagem(int sequencial, String nome, int nivel, int pontos_vida, int
pontos_forca, Classe classe) {
      this.sequencial = sequencial;
      this.nome = nome;
      this.nivel = nivel;
      this.pontos_vida = pontos_vida;
      this.pontos_forca = pontos_forca;
      this.classe = classe;
   }

public Personagem(int sequencial, Bolsa bolsa, SerMágico serMágico, int nivel) {
   this.sequencial = sequencial;
   this.bolsa = bolsa;
   this.serMágico = serMágico;
   this.nivel= nivel;

}
```

```
public Personagem(int sequencial, String nome, int nivel, int pontos_vida, int
pontos_forca,
      Classe classe, Bolsa bolsa, SerMágico serMágico) {
   this.nome = nome;
   this.nivel = nivel;
   this.pontos_vida = pontos_vida;
   this.pontos_forca = pontos_forca;
   this.classe = classe;
   this.bolsa = bolsa;
   this.serMágico = serMágico;
}

public Personagem(int sequencial, String nome, int nivel) {
   this.sequencial = sequencial;
   this.nome = nome;
   this.nivel = nivel;

}

   public String toString() {
      return "["+sequencial+ "] LV: " + nivel + " - Bolsa: " + bolsa.getVisão() + " | Ser
Mágico: " + serMágico.getVisão();
   }

  public String toStringFull() {
    String str = "[" + nome + "] Nivel: " + nivel
          + " --- Ser "+ serMágico.toStringFull()
          + " -- Bolsa: " + bolsa.toStringFull() + "\n    Itens:";

     Item[] itens_bolsa = Possuir.buscarItensBolsa(bolsa.getSequencial());
```

```
    for (Item item : itens_bolsa) {
        str += item.toStringFull();
    }
    return str;
}

public Personagem getVisão() {
    return new Personagem(sequencial,nome,nivel);
}

public int getSequencial(){
    return sequencial;
}

public void setSequencial(int sequencial) {
    this.sequencial = sequencial;
}

public String getNome(){
    return nome;
}

public int getNivel(){
    return nivel;
}

public int getPontosVida(){
    return pontos_vida;
}
```

```
public int getPontosForça(){
   return pontos_forca;
}

public Classe getClasse(){
   return classe;
}

 public Bolsa getBolsa(){
   return bolsa;
}


   public SerMágico getSerMagico(){
   return serMágico;
}



public static Personagem[] getVisões () {
   String sql = "SELECT Sequencial, BolsaId, SerMágicoId,Nivel FROM Personagens";
   ResultSet lista_resultados = null;
   ArrayList<Personagem> visões = new ArrayList();
   try {
      PreparedStatement comando = BD.conexão.prepareStatement(sql);
      lista_resultados = comando.executeQuery();
      while (lista_resultados.next()) {
         visões.add(new Personagem (lista_resultados.getInt("Sequencial"),
```

```
            Bolsa.buscarBolsa(lista_resultados.getInt("BolsaId")).getVisão(),

SerMágico.buscarSerMágico(lista_resultados.getString("SerMágicoId")).getVisão(),
            lista_resultados.getInt("Nivel")));
         }
         lista_resultados.close();
         comando.close();
      } catch (SQLException exceção_sql) {exceção_sql.printStackTrace ();}
      return visões.toArray(new Personagem[visões.size()]);
   }

   public static boolean existePersonagem (int chave_bolsa, String chave_ser) {
      String sql = "SELECT Sequencial FROM Personagens WHERE BolsaId = ? AND
SerMágicoId = ?";
      ResultSet lista_resultados = null;
      boolean existe = false;
      try {
         PreparedStatement comando = BD.conexão.prepareStatement(sql);
         comando.setInt(1, chave_bolsa);
         comando.setString(2, chave_ser);
         lista_resultados = comando.executeQuery();
         while (lista_resultados.next()) {
            existe = true;
         }
         lista_resultados.close();
         comando.close();
      } catch (SQLException exceção_sql) {
      exceção_sql.printStackTrace ();
      }
      return existe;
```

```
    }

    public static String inserirPersonagem (Personagem personagem) {
        String sql = "INSERT INTO Personagens (BolsaId, SerMágicoId, Nome,"
        + " Nivel, Pontos_vida, Pontos_força, Classe)"
        + " VALUES (?, ?, ?, ?, ?, ?, ?)";
        try {
            PreparedStatement comando = BD.conexão.prepareStatement(sql);
            comando.setInt(1, personagem.getBolsa().getSequencial());
            comando.setString(2, personagem.getSerMagico().getNome());
            comando.setString(3, personagem.getNome());
            comando.setInt(4, personagem.getNivel());
            comando.setInt(5, personagem.getPontosVida());
            comando.setInt(6, personagem.getPontosForça());
            comando.setInt(7, personagem.getClasse().ordinal());
            comando.executeUpdate();
            comando.close();
            return null;
        } catch (SQLException exceção_sql) {
            exceção_sql.printStackTrace ();
            return "Erro na Inserção da Personagem no BD";
        }
    }

    public static String alterarPersonagem (Personagem personagem) {
        String sql = "UPDATE Personagens SET Nivel = ?, Pontos_vida = ?, Pontos_força = ?,
Classe = ?, BolsaId = ?, SerMágicoID = ? WHERE Sequencial = ?";
        System.out.println("Nivel: " + personagem.getNivel());
        System.out.println("Pontos de vida: " + personagem.getPontosVida());
        System.out.println("Pontos de força: " + personagem.getPontosForça());
```

```
    System.out.println("Classe: " + personagem.getClasse());
    System.out.println("Sequencial: " + personagem.getSequencial());

    try {
        PreparedStatement comando = BD.conexão.prepareStatement(sql);
        comando.setInt(1, personagem.getNivel());
        comando.setInt(2, personagem.getPontosVida());
        comando.setInt(3, personagem.getPontosForça());
        comando.setInt(4, personagem.getClasse().ordinal());
        comando.setInt(5, personagem.bolsa.getSequencial());
        comando.setString(6, personagem.serMágico.getNome());
        comando.setInt(7, personagem.getSequencial());

        comando.executeUpdate();
        comando.close();
        return null;
    } catch (SQLException exceção_sql) {
        exceção_sql.printStackTrace ();
        return "Erro na Alteração da Personagem no BD";
        }
}

public static Personagem buscarPersonagem (int sequencial) {
    String sql = "SELECT * FROM Personagens where Sequencial = ?";
    ResultSet lista_resultados = null;
    Personagem personagem = null;
    try {
        PreparedStatement comando = BD.conexão.prepareStatement(sql);
        comando.setInt(1, sequencial);
```

```
        lista_resultados = comando.executeQuery();
        while (lista_resultados.next()) {
              personagem = new Personagem (
                sequencial,
                lista_resultados.getString("Nome"),
                lista_resultados.getInt("Nivel"),
                lista_resultados.getInt("Pontos_vida"),
                lista_resultados.getInt("Pontos_força"),
                Classe.values()[lista_resultados.getInt("Classe")],
                Bolsa.buscarBolsa(lista_resultados.getInt("BolsaId")),
                SerMágico.buscarSerMágico(lista_resultados.getString("SerMágicoId"))
              );
        }
        lista_resultados.close();
        comando.close();
    } catch (SQLException exceção_sql) {
    exceção_sql.printStackTrace ();
    personagem = null;
    }
    return personagem;
}

public static boolean existePersonagem (int sequencial) {
    String sql = "SELECT COUNT(Sequencial) FROM Personagens WHERE Sequencial = ?";
    ResultSet lista_resultados = null;
    boolean existe = false;
    try {
        PreparedStatement comando = BD.conexão.prepareStatement(sql);
        comando.setInt(1, sequencial);
```

```
        lista_resultados = comando.executeQuery();
        while (lista_resultados.next()) {
           existe = true;
        }
        lista_resultados.close();
        comando.close();
    } catch (SQLException exceção_sql) {
         exceção_sql.printStackTrace ();
    }
    return existe;
}

    public static String removerPersonagem (int sequencial) {
    String sql = "DELETE FROM Personagens WHERE Sequencial = ?";
    try {
    PreparedStatement comando = BD.conexão.prepareStatement(sql);
    comando.setInt(1, sequencial);
    comando.executeUpdate();
    comando.close();
    return null;
    } catch (SQLException exceção_sql) {
    exceção_sql.printStackTrace ();
    return "Erro na Remoção de Personagem no BD";
    }
}


public static int últimoSequencial() {
    String sql = "SELECT MAX(sequencial) FROM Personagens";
```

```
    ResultSet lista_resultados = null;
    int sequencial = 0;
    try {
       PreparedStatement comando = BD.conexão.prepareStatement(sql);
       lista_resultados = comando.executeQuery();
       while (lista_resultados.next()) {
          sequencial = lista_resultados.getInt(1);
       }
       lista_resultados.close();
       comando.close();
    } catch (SQLException exceção_sql) {
       exceção_sql.printStackTrace();
    }
    return sequencial;
  }

  public static ArrayList<Personagem> pesquisarPersonagens
    (int nivel_personagem, int peso_item, HabilidadeMágica habilidade_ser,
    char hostil_monstro, Tendência tendencia_rac, Boolean todos_itens_bolsa,int
peso_bolsa) {
       String sql = "SELECT Per.Sequencial, Per.Nivel, Ser.nome, Ser.HabilidadeMagica,
Bol.Sequencial, Bol.Peso_Max"
       + " FROM Personagens Per, Bolsas Bol, SeresMagicos Ser"
       + " WHERE Per.BolsaId = Bol.Sequencial AND Per.SerMágicoId = Ser.Nome";

       if (nivel_personagem > -1) sql += " AND Per.Nivel <= ?";
       if (peso_bolsa > -1) sql += " AND Bol.Peso_Max <= ?";
       if (habilidade_ser != null) sql += " AND Ser.HabilidadeMagica = ?";
       sql += " ORDER BY Per.Sequencial";
```

```
        ResultSet lista_resultados = null;
        ArrayList<Personagem> personagens_selecionadas = new ArrayList();
        int index = 0;
        int sequencial_bolsa = -1;
        String chave_ser = null;
        try {
           PreparedStatement comando = BD.conexão.prepareStatement(sql);
           if (nivel_personagem > 0) comando.setInt(++index, nivel_personagem);
           if (peso_bolsa > 0) comando.setInt(++index, peso_bolsa);
           if (habilidade_ser != null) comando.setInt(++index, habilidade_ser.ordinal());
           lista_resultados = comando.executeQuery();

           while (lista_resultados.next()) {
              Personagem personagem_pesquisada =
Personagem.buscarPersonagem(lista_resultados.getInt(1));
              chave_ser = lista_resultados.getString(3);
              sequencial_bolsa = lista_resultados.getInt(5);

              if (!itensAtendemFiltros(sequencial_bolsa, peso_item, todos_itens_bolsa)) {
              continue;
              }
              if (hostil_monstro != 'X') {
                 if (isOkPesquisaEmSerMágicoMonstro(chave_ser, hostil_monstro))
                    personagens_selecionadas.add(personagem_pesquisada);
              } else if (tendencia_rac != null) {
                 if (isOkPesquisaEmSerMágicoRaça(chave_ser, tendencia_rac))
                    personagens_selecionadas.add(personagem_pesquisada);
              } else personagens_selecionadas.add(personagem_pesquisada);
           }
           lista_resultados.close();
```

```
            comando.close();
        } catch (SQLException exceção_sql) {
            exceção_sql.printStackTrace ();
        }
        return personagens_selecionadas;
    }

    private static boolean itensAtendemFiltros(int sequencial_bolsa, int peso_item,
Boolean todos_itens_bolsa){
            Item[] itens_bolsa = Possuir.buscarItensBolsa(sequencial_bolsa);
            int total_itens_não_atendem_filtros = 0;
            for (Item item : itens_bolsa) {
               if (((peso_item != -1) && (item.getPeso() >= peso_item))) {
                  total_itens_não_atendem_filtros++;
               if (todos_itens_bolsa) return false;
               }
            }
            if (total_itens_não_atendem_filtros == 0) return true;
            if ((todos_itens_bolsa) || (total_itens_não_atendem_filtros ==
itens_bolsa.length))return false;
            return true;
        }


    private static boolean isOkPesquisaEmSerMágicoMonstro(String chave_ser, char
hostil_monstro) {
        boolean pesquisa_ok = false;
        String sql = "SELECT * FROM SeresMagicosMonstros WHERE SerMágicoId = ?";
        if (hostil_monstro != 'X') {
            sql += " AND Hostil = ?";
```

```
        }
        ResultSet lista_resultados = null;
        int index = 1;
        try {
            PreparedStatement comando = BD.conexão.prepareStatement(sql);
            comando.setString(1, chave_ser);

           switch(hostil_monstro) {
               case 'T': comando.setBoolean(++index, true); break;
               case 'F': comando.setBoolean(++index, false);
            }

            lista_resultados = comando.executeQuery();
            while (lista_resultados.next()) {
               pesquisa_ok = true;
            }
            lista_resultados.close();
            comando.close();
        } catch (SQLException exceção_sql) {
            exceção_sql.printStackTrace();
        }
        return pesquisa_ok;
    }

        private static boolean isOkPesquisaEmSerMágicoRaça(String chave_ser,Tendência
tendencia) {
        boolean pesquisa_ok = false;
        String sql = "SELECT * FROM SeresMagicosRacasInteligente WHERE SerMágicoId = ?";
        if (tendencia != null) {
            sql += " AND Tendência = ?";
```

```
        }
        ResultSet lista_resultados = null;
        try {
            PreparedStatement comando = BD.conexão.prepareStatement(sql);
            comando.setString(1, chave_ser);
            if (tendencia != null) {
                comando.setInt(2, tendencia.ordinal());
            }
            lista_resultados = comando.executeQuery();
            while (lista_resultados.next())
                pesquisa_ok = true;
            lista_resultados.close();
            comando.close();
        } catch (SQLException exceção_sql) {
            exceção_sql.printStackTrace();
        }
        return pesquisa_ok;
    }
}
```

interfaces.JanelaCadastrarPersonagens

```
package interfaces;

import javax.swing.JOptionPane;
import entidade.Personagem;
import entidade.Personagem.Classe;
import entidade.Bolsa;
import entidade.SerMágico;
```

```
import javax.swing.DefaultComboBoxModel;
import javax.swing.DefaultListModel;

import controle.ControladorCadastroPersonagens;




public class JanelaCadastrarPersonagens extends javax.swing.JFrame {
   ControladorCadastroPersonagens controlador;
   DefaultListModel modelo_lista_personagens;
   Bolsa[] bolsas_cadastrados;
   SerMágico[] seresMágicos_cadastrados;
  //modelo_lista_personagens = (DefaultListModel) personagensList.getModel();

   public JanelaCadastrarPersonagens(ControladorCadastroPersonagens controlador) {
      this.controlador = controlador;
      bolsas_cadastrados = Bolsa.getVisões();
      seresMágicos_cadastrados = SerMágico.getVisões();
      initComponents();
      inicializarListaPersonagens();
      limparCampos();
   }
     private void inicializarListaPersonagens(){
      modelo_lista_personagens = (DefaultListModel) personagensList.getModel();
      Personagem[] visões = Personagem.getVisões();
      for(Personagem visão : visões){
         modelo_lista_personagens.addElement(visão);
      }
```

```
    }

    private void informarErro (String mensagem) {
        JOptionPane.showMessageDialog (this, mensagem, "Erro",
JOptionPane.ERROR_MESSAGE);
    }


    private SerMágico getVisãoSerSelecionado(Personagem personagem) {
        String chave_ser = personagem.getSerMagico().getNome();
        for (SerMágico visão_ser : seresMágicos_cadastrados) {
            if (visão_ser.getNome().equals(chave_ser)) {
                return visão_ser;
            }
        }
        return null;
    }

    private Bolsa getVisãoBolsaSelecionado(Personagem personagem ) {
        int chave_bolsa = personagem.getBolsa().getSequencial();
        for (Bolsa visão_bolsa : bolsas_cadastrados) {
            if (visão_bolsa.getSequencial() == chave_bolsa) {
                return visão_bolsa;
            }
        }
        return null;
    }

    private void limparCampos() {
            sequencialTextField.setText("");
```

```
        bolsas_cadastradosComboBox.setSelectedIndex(-1);
        seres_cadastradosComboBox.setSelectedIndex(-1);
        classeComboBox.setSelectedIndex(-1);
        nomeTextField.setText("");
        nivelTextField.setText("");
        pontosVidaTextField.setText("");
        pontosForçaTextField.setText("");
}
  /**
   * This method is called from within the constructor to initialize the form.
   * WARNING: Do NOT modify this code. The content of this method is always
   * regenerated by the Form Editor.
   */
  @SuppressWarnings("unchecked")

  private Personagem obtémPersonagemInformada() {
     String sequencial_str = sequencialTextField.getText();
     int sequencial = 0;
     if (!sequencial_str.isEmpty()) sequencial = Integer.parseInt(sequencial_str);

     Bolsa visão_bolsa = (Bolsa) bolsas_cadastradosComboBox.getSelectedItem();
     if (visão_bolsa == null) return null;

     SerMágico visão_ser = (SerMágico) seres_cadastradosComboBox.getSelectedItem();
     if (visão_ser == null) return null;

     String nome = nomeTextField.getText();
     if (nome.isEmpty()){
        return null;
```

```
        }

        String nivel_str = nivelTextField.getText();
        int nivel = 0;
        if (!nivel_str.isEmpty()) nivel = Integer.parseInt(nivel_str);
        else return null;

        String pontos_vida_str = pontosVidaTextField.getText();
        int pontos_vida = 0;
        if (!pontos_vida_str.isEmpty()) pontos_vida = Integer.parseInt(pontos_vida_str);
        else return null;

        String pontos_força_str = pontosForçaTextField.getText();
        int pontos_força =0;
        if (!pontos_força_str.isEmpty()) pontos_força = Integer.parseInt(pontos_força_str);
        else return null;



        Classe classe = null;
        if (classeComboBox.getSelectedItem() != null)
        classe = (Classe) classeComboBox.getSelectedItem();
        else return null;

        return new Personagem(sequencial, nome, nivel, pontos_vida,pontos_força, classe,
visão_bolsa, visão_ser);
    }

  private void inserirPersonagem(java.awt.event.ActionEvent evt) {
          Personagem personagem = obtémPersonagemInformada();
          String mensagem_erro = null;
```

```
        if (personagem != null) mensagem_erro =
controlador.inserirPersonagem(personagem);
        else mensagem_erro = "Algum atributo do Personagem não foi informado";
        if (mensagem_erro == null) {
          int sequencial = Personagem.últimoSequencial();
          personagem.setSequencial(sequencial);
          modelo_lista_personagens.addElement(personagem.getVisão());
          personagensList.setSelectedIndex(modelo_lista_personagens.size() - 1);
          sequencialTextField.setText("" + sequencial);
        } else informarErro (mensagem_erro);
    }

    private void consultarPersonagem(java.awt.event.ActionEvent evt) {
        Personagem visão_personagem = (Personagem) personagensList.getSelectedValue();
        Personagem personagem = null;
        String mensagem_erro = null;
        if (visão_personagem != null) {
        personagem = Personagem.buscarPersonagem(visão_personagem.getSequencial());
        if (personagem == null) mensagem_erro = "Personagem não cadastrada";
        } else mensagem_erro = "Nenhuma personagem selecionada";
        if (mensagem_erro == null) {
        sequencialTextField.setText(personagem.getSequencial() + "");

bolsas_cadastradosComboBox.setSelectedItem(getVisãoBolsaSelecionado(personagem));
        seres_cadastradosComboBox.setSelectedItem(getVisãoSerSelecionado(personagem));
        classeComboBox.setSelectedItem(personagem.getClasse());
        nomeTextField.setText(personagem.getNome());
        nivelTextField.setText(personagem.getNivel()+ "");
        pontosVidaTextField.setText(personagem.getPontosVida() + "");
        pontosForçaTextField.setText(personagem.getPontosForça() + "");
```

```
        } else informarErro (mensagem_erro);
    }

    private void alterarPersonagem(java.awt.event.ActionEvent evt) {
        Personagem personagem = obtémPersonagemInformada();
        String mensagem_erro = null;
        if (personagem != null)
            mensagem_erro = controlador.alterarPersonagem(personagem);
        else
            mensagem_erro = "Algum atributo de personagem não foi informado";
        if (mensagem_erro != null) informarErro (mensagem_erro);
    }

    private void removerPersonagem(java.awt.event.ActionEvent evt) {
        Personagem visão = (Personagem) personagensList.getSelectedValue();
        String mensagem_erro = null;
        if (visão != null) mensagem_erro =
controlador.removerPersonagem(visão.getSequencial());
        else mensagem_erro = "Nenhum Personagem selecionado";
        if (mensagem_erro == null) {
            modelo_lista_personagens.removeElement(visão);
        } else informarErro (mensagem_erro);;
    }

    private void LimparCampos(java.awt.event.ActionEvent evt) {
        limparCampos();
    }
```

```
    private void sequencialTextFieldActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void classeComboBox(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional)
">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and
feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
```

```
java.util.logging.Logger.getLogger(JanelaCadastrarPersonagens.class.getName()).log(java.u
til.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(JanelaCadastrarPersonagens.class.getName()).log(java.u
til.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(JanelaCadastrarPersonagens.class.getName()).log(java.u
til.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(JanelaCadastrarPersonagens.class.getName()).log(java.u
til.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>
        //</editor-fold>

        /* Create and display the form */

    }
```

interfaces.JanelaPesquisarPersonagens

```
*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change
this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this
template
 */
package interfaces;
import java.util.ArrayList;
import entidade.SerMágicoRaçaInteligente.Tendência;
```

```
import entidade.SerMágico.HabilidadeMágica;
import entidade.Personagem;
import javax.swing.DefaultComboBoxModel;

/**
 *
 * @author Familia
 */
public class JanelaPesquisarPersonagens extends javax.swing.JFrame {
   PainelFiltrosSerMágicoMonstro filtro_serMágico_monstroPainel;
   PainelFiltrosSerMágicoRaçaInteligente filtro_serMágico_raçaInteligentePainel;
   /**
    * Creates new form JanelaPesquisarPersonagens
    */
   public JanelaPesquisarPersonagens() {
      initComponents();
      filtro_serMágico_monstroPainel = new PainelFiltrosSerMágicoMonstro();
      filtro_serMágico_raçaInteligentePainel = new
PainelFiltrosSerMágicoRaçaInteligente();
      especialização_serMagicoTabbedPane.addTab("
Monstro",filtro_serMágico_monstroPainel);
      especialização_serMagicoTabbedPane.addTab("Raça
Inteligente",filtro_serMágico_raçaInteligentePainel);
      limparFiltros(null);
   }

   private void mostrarPersonagensSelecionadas(ArrayList<Personagem> personagens) {
      boolean primeira_personagem = true;
      for (Personagem personagem : personagens) {
         if (primeira_personagem) {
```

```
            pesquisasTextArea.append(personagem.toStringFull());
            primeira_personagem = false;
        } else pesquisasTextArea.append("\n" + personagem.toStringFull());
        if(personagem.toStringFull() == null) System.out.println("tem nada?");
    }
    System.out.println("mostrou?");
}

/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */

private void nivelMáximoTextFieldActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void pesoMáximoTextFieldActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void todosCheckBoxActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void pesquisarPersonagens(java.awt.event.ActionEvent evt) {
    int nivel_personagem = -1;
    if (!nivelMáximoTextField.getText().isEmpty())
```

```
        nivel_personagem = Integer.parseInt(nivelMáximoTextField.getText());
        int peso_bolsa = -1;
        if (!pesoMáximoBolsaTextField.getText().isEmpty())
        peso_bolsa = Integer.parseInt(pesoMáximoBolsaTextField.getText());
        int peso_item = -1;
        if (!pesoMáximoTextField.getText().isEmpty())
        peso_item = Integer.parseInt(pesoMáximoTextField.getText());
        HabilidadeMágica habilidade_ser = null;
        if (habilidadeComboBox.getSelectedItem() != null)
        habilidade_ser = (HabilidadeMágica) habilidadeComboBox.getSelectedItem();
        boolean todos_itens_bolsa = todosCheckBox.isSelected();
        char hostil_monstro= 'X';
        Tendência tendencia_ser = null;
       int indice_aba_selecionada =
especialização_serMagicoTabbedPane.getSelectedIndex();
       if (indice_aba_selecionada == 0) {
            hostil_monstro = filtro_serMágico_monstroPainel.getHostil();
        } else if (indice_aba_selecionada == 1) {
            tendencia_ser = filtro_serMágico_raçaInteligentePainel.getSelectedTendencia();
        }


       ArrayList<Personagem> personagens =
Personagem.pesquisarPersonagens(nivel_personagem,
         peso_item,
        habilidade_ser,
        hostil_monstro,
        tendencia_ser,
        todos_itens_bolsa,
        peso_bolsa
        );
```

```
        mostrarPersonagensSelecionadas(personagens);
    }

    private void habilidadeComboBox(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void limparFiltros(java.awt.event.ActionEvent evt) {
    nivelMáximoTextField.setText("");
    pesoMáximoBolsaTextField.setText("");
    pesoMáximoTextField.setText("");
    todosCheckBox.setSelected(false);
    habilidadeComboBox.setSelectedIndex(-1);

    }

    private void limparPesquisa(java.awt.event.ActionEvent evt) {
        pesquisasTextArea.setText("");
    }

    private void pesoMáximoBolsaTextFieldActionPerformed(java.awt.event.ActionEvent
evt) {
        // TODO add your handling code here:
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
```

```
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional)
">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and
feel.
     * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
     */
    try {
       for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
          if ("Nimbus".equals(info.getName())) {
             javax.swing.UIManager.setLookAndFeel(info.getClassName());
             break;
          }
       }
    } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(JanelaPesquisarPersonagens.class.getName()).log(java.u
til.logging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(JanelaPesquisarPersonagens.class.getName()).log(java.u
til.logging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(JanelaPesquisarPersonagens.class.getName()).log(java.u
til.logging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(JanelaPesquisarPersonagens.class.getName()).log(java.u
til.logging.Level.SEVERE, null, ex);
    }
    //</editor-fold>
```

```
        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new JanelaPesquisarPersonagens().setVisible(true);
            }
        });
    }
```

interfaces.PainelFiltrosSerMágicoMonstro

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JPanel.java to edit this template
 */
package interfaces;

/**
 *
 * @author Familia
 */
public class PainelFiltrosSerMágicoMonstro extends javax.swing.JPanel {

    /**
     * Creates new form PainelFiltrosSerMágicoMonstro
     */
    public PainelFiltrosSerMágicoMonstro() {
        initComponents();
    }
```

```
    public char getHostil(){
     char muito_conhecida = 'X';
     if(HostilButtonGroup.getSelection() != null){
       switch (HostilButtonGroup.getSelection().getMnemonic()) {
           case 0: muito_conhecida = 'T'; break;
           case 1: muito_conhecida = 'F'; break;
       }
     }
     return muito_conhecida;
   }

      public void limparCampos() {
       HostilButtonGroup.clearSelection();
      }
   /**
    * This method is called from within the constructor to initialize the form.
    * WARNING: Do NOT modify this code. The content of this method is always
    * regenerated by the Form Editor.
    */
   @SuppressWarnings("unchecked")
```

interfaces.PainelFiltrosSerMágicoRaçaInteligente

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JPanel.java to edit this template
 */
```

```
package interfaces;
import entidade.SerMágicoRaçaInteligente.Tendência;
/**
 *
 * @author Familia
 */
public class PainelFiltrosSerMágicoRaçaInteligente extends javax.swing.JPanel {

    /**
     * Creates new form PainelFiltrosSerMágicoRaçaInteligente
     */
    public PainelFiltrosSerMágicoRaçaInteligente() {
        initComponents();
    }
    public Tendência getSelectedTendencia() {
        Tendência tendencia = null;
        if (tendenciaButtonGroup.getSelection() != null)
            tendencia =
Tendência.values()[tendenciaButtonGroup.getSelection().getMnemonic()];
        return tendencia;
    }
    public void setSelectedTendencia(int índice_tendencia) {
        switch(índice_tendencia) {
            case 0: caóticoRadioButton.setSelected(true); break;
            case 1: neutroRadioButton.setSelected(true);
            case 2: bomRadioButton.setSelected(true);
        }
    }
     public void limparCampos() {
        tendenciaButtonGroup.clearSelection();
```

```
}
/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
```

_____

Dourados 27/11/2024