Java – Aula 05

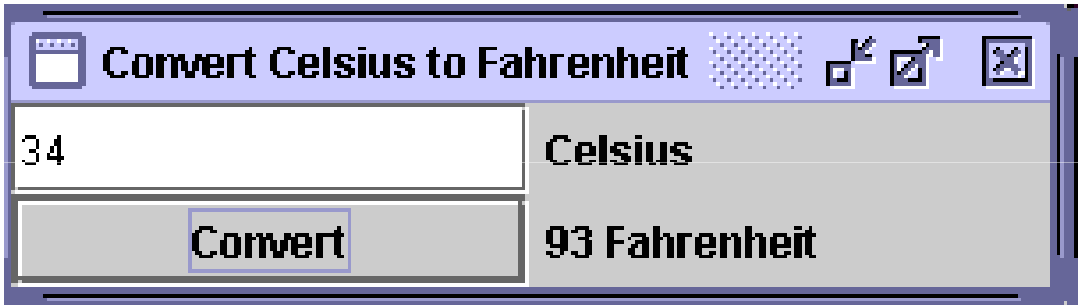
JDBC

12/09/2012

Celso Olivete Júnior

olivete@fct.unesp.br

Exercício para aquecimento...utilize o pacote Swing



$$C = \frac{(F-32)*5}{9}$$

Acesso a Bancos de Dados via JDBC

- Softwares utilizados:
- Java Development Kit (J2SDK)
- Ambiente de Desenvolvimento (JBuilder, NetBeans, Gel, JCreator, etc)
- Sistema Gerenciador de Bancos de Dados (MySQL, Interbase, Oracle, etc)
- Driver de Acesso (Específico do SGBD utilizado) → MySql JDBC Driver

Acesso a Bancos de Dados via JDBC

- Passos necessários:
  - Instalação SGBD
  - Uso do MySQL
  - Criação da Base de Dados a partir de uma modelagem e um script SQL ou diretamente no "front" → PhpMyAdmin
  - Definição de uma Fonte de dados

Acesso a Bancos de Dados via JDBC

- Biblioteca específica:
  - java.sql

Acesso a Bancos de Dados via JDBC

Conexão JDBC:

```
import java.sql.*;
...
private Connection connect;
...
try {
        String url = "jdbc:mysql://localhost/bd_aula5?user=root&password=";
        Class.forName( "com.mysql.jdbc.Driver” );
        connect = DriverManager.getConnection( url );
    }
catch ( ClassNotFoundException cnfex )
{System.err.println("Falha ao ler o driver JDBC/ODBC“+ cnfex );}
catch ( SQLException sqlex )
{System.out.println("Impossível conectar“+ sqlex );}
catch ( Exception ex )
{System.out.println(“Outro erro“+ ex );}
```

Acesso a Bancos [illegible] via JDBC

Biblioteca SQL

Conexão JDBC:

```
import java.sql.*;
...
private Connection connect;
...
try {
        String url = "jdbc:mysql://localhost/bd_aula5?user=root&password=";
        Class.forName( "com.mysql.jdbc.Driver” );
        connect = DriverManager.getConnection( url );
     }
catch ( ClassNotFoundException cnfex )
{System.err.println("Falha ao ler o driver JDBC/ODBC“+ cnfex );}
catch ( SQLException sqlex )
{System.out.println("Impossível conectar“+ sqlex );}
catch ( Exception ex )
{System.out.println(“Outro erro“+ ex );}
```

Acesso a Bancos [illegible] via JDBC

Conexão JDBC:

Biblioteca SQL

Objeto para a Conexão

```
import java.sql.*;
...
private Connection connect;
...
try {
        String url = "jdbc:mysql://localhost/bd_aula5?user=root&password=";
        Class.forName( "com.mysql.jdbc.Driver” );
        connect = DriverManager.getConnection( url );
     }
catch ( ClassNotFoundException cnfex )
{System.err.println("Falha ao ler o driver JDBC/ODBC“+ cnfex );}
catch ( SQLException sqlex )
{System.out.println("Impossível conectar“+ sqlex );}
catch ( Exception ex )
{System.out.println(“Outro erro“+ ex );}
```

Acesso a Bancos [illegible] via JDBC

Conexão JDBC:

Biblioteca SQL

Objeto para a Conexão

Caminho de conexão apontando para o nome da fonte JDBC. Neste caso o BD chama **bd_aula5;** o user = **root** e password = ***vazio***

```
import java.sql.*;
...
private Connection connect;
...
try {
        String url = "jdbc:mysql://localhost/bd_aula5?user=root&password=";
        Class.forName( "com.mysql.jdbc.Driver" );
        connect = DriverManager.getConnection( url );
     }
catch ( ClassNotFoundException cnfex )
{System.err.println("Falha ao ler o driver JDBC/ODBC"+ cnfex );}
catch ( SQLException sqlex )
{System.out.println("Impossível conectar"+ sqlex );}
catch ( Exception ex )
{System.out.println("Outro erro"+ ex );}
```

Acesso a Bancos [illegible] via JDBC

Conexão JDBC:

Biblioteca SQL

Objeto para a Conexão

```
import java.sql.*;
...
private Connection connect;
...
try {
        String url = "jdbc:mysql://localhost/bd_aula5?user=root&password=";
        Class.forName( "com.mysql.jdbc.Driver" );
        connect = DriverManager.getConnection( url );
    }
catch ( ClassNotFoundException cnfex )
{System.err.println("Falha ao ler o driver JDBC/ODBC"+ cnfex );}
catch ( SQLException sqlex )
{System.out.println("Impossível conectar"+ sqlex );}
catch ( Exception ex )
{System.out.println("Outro erro"+ ex );}
```

Efetua a conexão

Acesso a Bancos [illegible] via JDBC

Conexão JDBC:

Biblioteca SQL

Objeto para a Conexão

```
import java.sql.*;
...
private Connection connect;
...
try {
        String url = "jdbc:mysql://localhost/bd_aula5?user=root&password=";
        Class.forName( "com.mysql.jdbc.Driver" );
        connect = DriverManager.getConnection( url );
    }
catch ( ClassNotFoundException cnfex )
{System.err.println("Falha ao ler o driver JDBC/ODBC"+ cnfex );}
catch ( SQLException sqlex )
{System.out.println("Impossível conectar"+ sqlex );}
catch ( Exception ex )
{System.out.println("Outro erro"+ ex );}
```

Três tipos possíveis de erro

Acesso a Bancos de Dados via JDBC

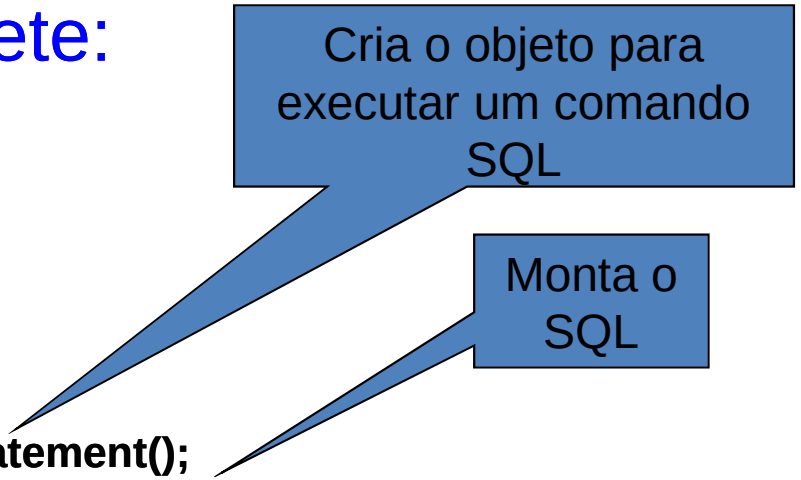
Insert / Select / Update / Delete:

Cria o objeto para executar um comando SQL

```
import java.sql.*;
...
private Connection connect;
...
try {   Statement statement = connect.createStatement();
        String query = "UPDATE pessoa SET nome = 'Joaquim' "+
                       " WHERE id = 1";
        int result = statement.executeUpdate( query );
        statement.close();
}
catch ( SQLException sqlex ) {...}
```

Acesso a Bancos de Dados via JDBC

Insert / Select / Update / Delete:




```
import java.sql.*;
...
private Connection connect;
...
try {   Statement statement = connect.createStatement();
        String query = "UPDATE pessoa SET nome = 'Joaquim' "+
                       " WHERE id = 1";
        int result = statement.executeUpdate( query );
        statement.close();
}
catch ( SQLException sqlex ) {...}
```

Acesso a Bancos de Dados via JDBC

Insert / Select / Update / Delete:

```
import java.sql.*;
...
private Connection connect;
...
try {   Statement statement = connect.createStatement();
        String query = "UPDATE pessoa SET nome = 'Joaquim' "+
                       " WHERE id = 1";
        int result = statement.executeUpdate( query );
        statement.close();
}
catch ( SQLException sqlex ) {...}
```

Cria o objeto para executar um comando SQL

Monta o SQL

Executa o SQL retornando o número de registros afetados

Acesso a Bancos de Dados via JDBC

Insert / Select / Update / Delete:

```
import java.sql.*;
...
private Connection connect;
...
try {   Statement statement = connect.createStatement();
        String query = "UPDATE pessoa SET nome = 'Joaquim' "+
                       " WHERE id = 1";
        int result = statement.executeUpdate( query );
        statement.close();
}
catch ( SQLException sqlex ) {...}
```

Cria o objeto para executar um comando SQL

Monta o SQL

Executa o SQL retornando o número de registros afetados

Finaliza o objeto que executou o SQL

Acesso a Bancos de Dados via JDBC

Insert / Select / Update / Delete:

```
import java.sql.*;
...
private Connection connect;
...
try {   Statement statement = connect.createStatement();
        String query = "UPDATE pessoa SET nome = 'Joaquim' "+
                       " WHERE id = 1";
        int result = statement.executeUpdate( query );
        statement.close();
}
catch ( SQLException sqlex ) {...}
```

Cria o objeto para executar um comando SQL

Monta o SQL

Executa o SQL retornando o número de registros afetados

Finaliza o objeto que executou o SQL

Erro de SQL

Acesso a Bancos de Dados via JDBC

Select:

```
import java.sql.*;
...
private Connection connect;
...
try {
        Statement statement = connect.createStatement();
        String query = “SELECT * FROM addresses”;
        ResultSet rs = statement.executeQuery( query );
        statement.close();
}
catch ( SQLException sqlex ) {...}
```

Acesso a Bancos de Dados via JDBC

Select:

```
import java.sql.*;
...
private Connection connect;
...
try {
        Statement statement = connect.createStatement();
       String query = “SELECT * FROM addresses”;
        ResultSet rs = statement.executeQuery( query );
        statement.close();
}
catch ( SQLException sqlex ) {...}
```

Cria o objeto para executar um comando SQL

Acesso a Bancos de Dados via JDBC

Select:

```
import java.sql.*;
...
private Connection connect;
...
        Statement statement = connect.createStatement();
         String query = "SELECT * FROM addresses";
        ResultSet rs = statement.executeQuery( query );
            statement.close();
}
catch ( SQLException sqlex ) {...}
```

Cria o objeto para executar um comando SQL

Monta SQL

Acesso a Bancos de Dados via JDBC

Select:

```
import java.sql.*;
...
private Connection connect;
...
        Statement statement = connect.createStatement();
        String query = "SELECT * FROM addresses";
        ResultSet rs = statement.executeQuery( query );
        statement.close();
}
catch ( SQLException sqlex ) {...}
```

Cria o objeto para executar um comando SQL

Monta SQL

Executa o SQL retornando um conjunto de dados

Acesso a Bancos de Dados via JDBC

Select:

```
import java.sql.*;
...
private Connection connect;
...
        Statement statement = connect.createStatement();
        String query = "SELECT * FROM addresses";
        ResultSet rs = statement.executeQuery( query );
        statement.close();
}
catch ( SQLException sqlex ) {...}
```

Cria o objeto para executar um comando SQL

Monta SQL

Executa o SQL retornando um conjunto de dados

Finaliza o objeto que executou o SQL

Acesso a Bancos de Dados via JDBC

- Utilizando os dados retornados:
  - Posicionamento de registros:

```
ResultSet rs;
...
rs.first();   rs.last();   rs.next();   rs.previous();
```

  - Escolhendo um campo:

```
Int Codigo = rs. getInt( "ID" );
String Nome = rs. getString( "Nome" );
Date Nascimento = rs. getDate( "DtNasc" );
```

Exemplo:

```
String info;
while(rs.next())
{  info=rs.getString("sobrenome");
   System.out.println(info);
}
```

Insert: ➔ Exemplo 1 (setando valores fora do sql)

```
import java.sql.*;
...
private Connection connect;
...
try {
        PreparedStatement insere_dados;
        insere_dados = connect.prepareStatement("insert into tb_dados values (?,?,?)");
        insere_dados.setString(1, null);
        insere_dados.setString(2, "xxxxx");
        insere_dados.setString(3, "yyyy");
        insere_dados.executeUpdate();
        System.out.print("Dados inseridos...");
      }
      catch (SQLException sqlex )
      {System.err.println("Erro no SQL!"+sqlex); }
```

Essa mesma forma pode ser feita para as instruções SELECT (no caso de uma busca), UPDATE e DELETE

Insert: ➔ Exemplo 2 (setando valores no sql)

```
import java.sql.*;
...
private Connection connect;
...
try {
        Statement insere_dados = connect.createStatement();
        String query = "INSERT INTO tb_dados (item,descricao) VALUES ('xxxxx','yyyyyy')";
        insere_dados.execute(query);
        System.out.print("Inserido...");
        }
catch (SQLException sqlex )
        {System.err.println("Erro no SQL!"+sqlex); }
```

Delete:

```
import java.sql.*;
...
private Connection connect;
...
 try {
        String query = "DELETE FROM tb_dados WHERE tb_cod = ?";
        PreparedStatement statement = connect.prepareStatement(query);
        statement.setInt(1, 44);
        statement.executeUpdate();  ;
        System.out.print("Dados excluídos...");
        }
        catch (SQLException sqlex )
        {System.err.println("Erro no SQL!"+sqlex); }
```

Exercício

- Desenvolver um BD e uma interface para sua respectiva manipulação (insert, select, update e delete)

Acesso a Bancos de Dados via JDBC

• Utilizando os dados retornados:

• Recuperando os campos de uma tabela

```
ResultSet rs;
                ...
ResultSetMetaData rsm = rs.getMetaData();

for( int i=1;i<=rsm.getColumnCount(); i++)
    System.out.println(rsm.getColumnName(i));
```

Recupera os campos de uma tabela ou query

Número de campos

• getColumnTypeName – retorna o tipo do dado;

→ java.sql.Types

```
for (int i=1; i<=rsmd.getColumnCount(); i++)
    {
     info=rsmd.getColumnTypeName(i);
     info2=rsmd.getColumnName(i);
     System.out.println(info2 + " --> " + info + "\n");
     }
```

Os campos

Retorna o tipo

Acesso a Bancos de Dados via JDBC

Proprietário

cod	nome	cpf
1	Jose Anibal	312121212
2	Cleide Alvares	121212323
3	Celina Dilon	121323423

Veículo

cod	codp	modelo	valor
1	1	Fusca 1600	1320,00
2	2	Brasilia	1200,00
3	1	Gordine II	4500,00

SQL:

```
SELECT modelo FROM Veiculo WHERE valor > 2000;
SELECT * FROM Proprietario WHERE name LIKE 'Jose%';
SELECT v.* FROM Proprietario p, veiculo v WHERE p.cod=v.codp;
INSERT INTO Veiculo VALUES(5,3,'DKW',3000.50);
UPDATE veiculo SET valor=valor*1.1 WHERE modelo="Fusca 1600";
```

Acesso a Bancos de Dados via JDBC

Inserindo um novo Registro (Tabela Pessoa)

```
//insere novo registro
rs.moveToInsertRow();
// move para a linha a ser inserida (depois da última tupla)
rs.updateString(2, "Heitor Villa Lobos");
// atualiza a coluna nome
rs.updateString("endereco", "Rua 7 de Setembro, 200");
rs.updateString("cidade","Presidente Prudente");
rs.updateString("estado","SP");
//rs.updateInt(3,1);
//rs.updateBoolean(3, true);
rs.insertRow();
rs.moveToCurrentRow();
```

Acesso a Bancos de Dados via JDBC

Inserindo um novo Registro (Tabela Pessoa)

```
//inserção usando um statement
Statement statement = connect.createStatement();

String query = "INSERT INTO pessoa
                (nome,endereco,cidade,estado)
                VALUES
                ('José de Alencar','Av. Copacabana, 222', 'Rio
                de Janeiro', 'RJ')";

boolean res = statement.execute(query);
```

Acesso a Bancos de Dados via JDBC

Alterando um Registro (Tabela Pessoa)

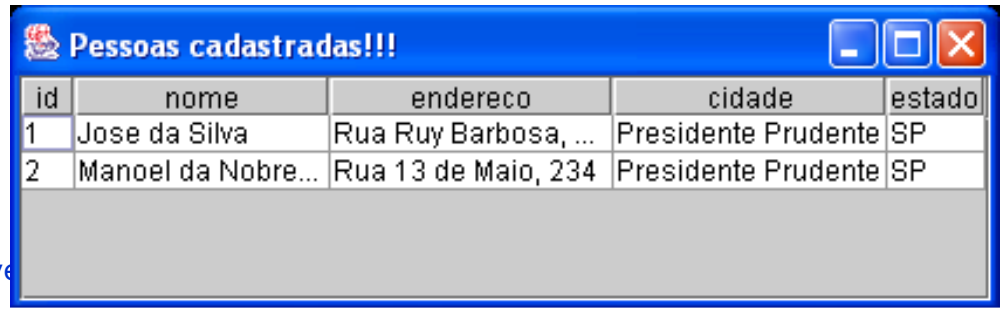
```
//alteração usando um statement
Statement statement = connect.createStatement();

String query = "UPDATE pessoa SET nome = 'Clara Nunes'
                WHERE id = 5";

int res = statement.executeUpdate(query);
```

Acesso a Bancos de Dados via JDBC

- Transações
- O método setAutoCommit() do objeto Connection que especifica se
  - cada instrução SQL executada deve ser confirmada (committed) individualmente setAutoCommit(true)
  - ou se várias instruções devem ser agrupadas como uma transação setAutoCommit(false)
- Se for usado setAutoCommit(false) a execução do bloco deve ser finalizada como o método commit() do objeto Connection
- Ou então, pelo método rollback() para voltar ao estado anterior.
- O método getAutoCommit() retorna o estado do AutoCommit (true ou false)

Acesso a Bancos de Dados via JDBC: exemplo com JTable

```
import java.sql.*;
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
import java.util.*; //por causa do Vector
public class TesteTable extends JFrame{
      private Connection connect;
       private JTable table;
public TesteTable()   //construtor
{
 try {
         String url = "jdbc:mysql://localhost/bd_aula5?user=root&password=";
         Class.forName( "com.mysql.jdbc.Driver” );
         connect = DriverManager.getConnection( url );
      }
 catch ( ClassNotFoundException cnfex )
 {System.err.println("Falha ao ler o driver JDBC/ODBC");}
 catch ( SQLException sqlex )
 {System.out.println("Impossível conectar");}
 catch ( Exception ex )
 {System.out.println("Outro erro");
 }
 getTable();
 show();
}
```



Acesso a Bancos de Dados via JDBC: exemplo com JTable

```
private void getTable(){
   Statement st;
   ResultSet rs;
   try{
       String query = "SELECT * FROM pessoa";
       st = connect.createStatement();
       rs = st.executeQuery(query);
       mostraResultado(rs);
       st.close();
   }catch(SQLException sqlex){
       sqlex.printStackTrace();
      }
 }
```

Acesso a Bancos de Dados via JDBC: exemplo com JTable

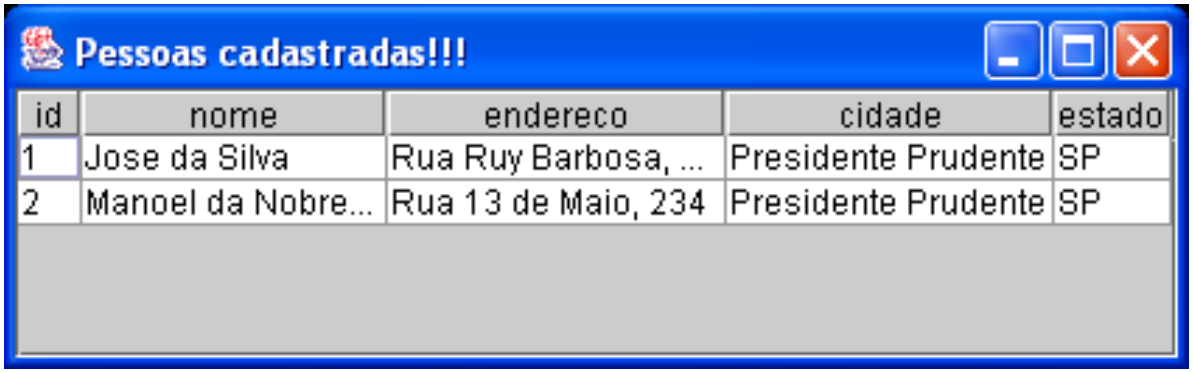
```
private void mostraResultado(ResultSet rs) throws SQLException {
   boolean verificaDados = rs.next();
   if (!verificaDados){
     JOptionPane.showMessageDialog(null, "Nenhum registro!");
     setTitle("Nenhum registro!!!");
     return;
   }
   setTitle("Pessoas cadastradas!!!");
   Vector cabecalho = new Vector();
   Vector tuplas = new Vector();
   try {
        ResultSetMetaData rsmd = rs.getMetaData();
        for (int i = 1; i<=rsmd.getColumnCount(); i++)
          cabecalho.addElement(rsmd.getColumnName(i));
       do{
         tuplas.addElement(getTupla(rs,rsmd));
       }while(rs.next());
       table = new JTable(tuplas, cabecalho);
       JScrollPane scroll = new JScrollPane(table);
       getContentPane().add(scroll, BorderLayout.CENTER);
       validate(); //Atualiza desenho da tela
    }catch(SQLException sqlex){
        sqlex.printStackTrace();    }
}
```

Acesso a Bancos de Dados via JDBC: exemplo com JTable

```
private Vector getTupla(ResultSet rs, ResultSetMetaData rsmd) throws SQLException{
    Vector tupla = new Vector();

    for (int i=1; i<=rsmd.getColumnCount(); i++)
    {
          switch(rsmd.getColumnType(i)){
                    case Types.VARCHAR:
                      tupla.addElement(rs.getString(i));
                      break;
                    case Types.INTEGER:
                      tupla.addElement(new Long(rs.getLong(i)));
                      break;
                    case Types.SMALLINT:
                      tupla.addElement(new Long(rs.getLong(i)));
                      break;
                    case Types.CHAR:
                      tupla.addElement(rs.getString(i));
                      break;
                    default:
                       System.out.println("O Tipo era: " + rsmd.getColumnTypeName(i));
          }
    }
    return tupla;
 }
```

Acesso a Bancos de Dados via JDBC: exemplo com JTable

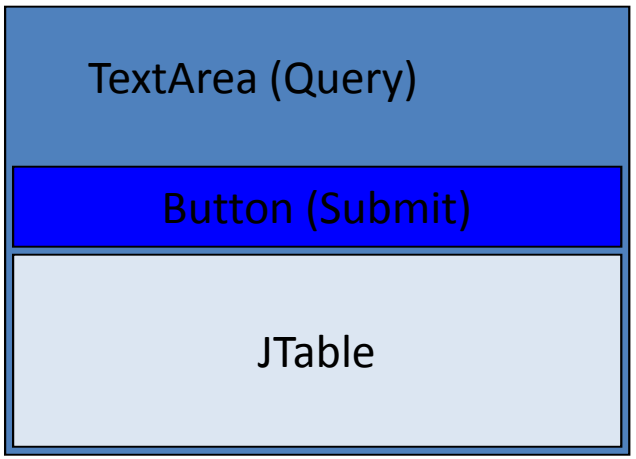
```
public void shutDown() {

   try{
     connect.close();
   }catch(SQLException sqlex){
   System.err.println("Não foi possível desconetar!!!");
   sqlex.printStackTrace();
   }
}


public static void main(String[] args){

  final TesteTable app = new TesteTable();

  app.addWindowListener( new WindowAdapter() {
   public void windowClosing(WindowEvent e){
          app.shutDown();
          System.exit(0);
   }
  });

}
```



Exercício 1

- Implementar um ambiente para consulta usando JTable, para qualquer consulta SQL efetuada pelo usuário;
- O JTable será recriado de acordo com o resultado da consulta;
- Dicas:
  - Para cada consulta o JTable deve ser removido:

```
Container c = getContentPane();
c.remove(1);
c.add(scroll, BorderLayout.CENTER);
c.validate();
```



Exercício 1

```
tEntrada = new JTextArea("SELECT * FROM pessoa", 4, 30);
 bConsulta = new JButton("Consultar");
 bConsulta.addActionListener( new ActionListener() {
        public void actionPerformed (ActionEvent e){
                if (e.getSource() == bConsulta)
                  getTable();
    } });
 JPanel topPanel = new JPanel();
 topPanel.setLayout( new BorderLayout());
 topPanel.add(new JScrollPane(tEntrada), BorderLayout.CENTER);
 topPanel.add(bConsulta,BorderLayout.SOUTH);

 table = new JTable(4,4);

 Container c = getContentPane();
 c.setLayout(new Borderlayout());
 c.add(topPanel,BorderLayout.NORTH);
 c.add(table,BorderLayout.CENTER);
```