6 – Acessos a banco de dados

Vamos agora ver como podemos utilizar objetos trabalhando em conjunto com banco de dados. Podemos utilizar qualquer banco de dados. Vamos utilizar, por praticidade, um bando de dados criado no MSAccess.

Utilizaremos o ADO .Net (Active Data Object para .Net), que permite, através de uma string de conexão, utilizar uma grande variedade de SGDBs, por exemplo

Para acessar Oracle

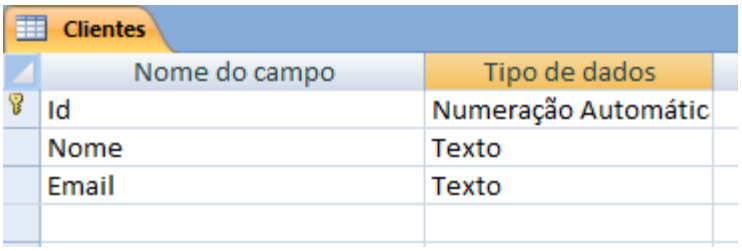
```
"Provider=MSDAORA; Data Source=ORACLE8i7;Persist Security Info=False;Integrated
Security=Yes"
```

```
Para acessar MSAccess(até 2007)
"Provider=Microsoft.Jet.OLEDB.4.0; Data Source=c:\LocalAccess40.mdb"

Para acessar MSAccess(até 2013)
"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=c:\LocalAccess40.accdb"


Para acessar MSSQLServer
"Provider=SQLOLEDB;Data Source=(local);Integrated Security=SSPI"
```

Antes de iniciarmos, então, precisamos invocar o OleDB para podermos realizar os acessos ao Banco de Dados. Então, nas chamadas de bibliotecas, iremos adicionar:

```
using System.Data.OleDb;
```

Será necessário termos um banco de dados criado no MsAccess. Então, vamos criar um novo banco de dados e inserir os seguintes campos (no MsAccess):



A tabela chama Clientes. O Banco de dados (arquivo) foi criado como Teste.MDB e os campos como descritos acima.
Vamos inserir alguns registros para podermos utilizar nos exemplos iniciais:

Clientes

Id	Nome	Email
1	Portari	portari@gmail
2	João	jao@bol.com.b
3	Jessica	jesslang@htor
4	Janice	janjan@uol.co
5	Miriam	mimi@uol.con
(Novo)		

Com o arquivo criado, é necessário colocarmos este arquivo em uma pasta conhecida. Para esta aula, foi criada uma pasta chamada c:\sergio e colocado o arquivo dentro dela. Você pode criar com os nomes que desejar, bastando depois adequar estes nome no código mostrado como exemplo.

Então vamos ao que interessa:

Para podermos chamar os métodos de acesso e manipulação ao banco de dados, inicialmente iremos criar uma classe estática:

```
public static class AcessoAoAccess
```

Esta classe conterá as instruções de Acesso ao nosso banco de dados Access e, por ser uma classe estática, não precisa ser instanciada por um objeto para podermos acessar seus métodos.

Em seguida, iremos criar um método dentro desta classe estática para realizar a conexão com o banco sempre que um outro método precisar acessar o banco:

```
public static OleDbConnection AbrirConexao()
{
  return new OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=c:\sergio\teste.mdb");
}
```

Este método retorna um objeto do tipo OleDbConnection, que será uma conexão aberta com o banco de dados destacado no fim da instrução. Neste destaque, deve-se colocar a pasta e o nome do banco de dados caso você tenha alterado algo.

Para podemos testar se tudo está correto na conexão, vamos criar um método para mostrar os dados que inserimos manualmente no banco no início da aula.

```
public static void Mostrar()
{
    //cria a conexão com o banco de dados
    OleDbConnection aConnection = AbrirConexao();

    //cria o objeto command and armazena a consulta SQL
    OleDbCommand aCommand = new OleDbCommand("select * from Clientes", aConnection);
    try  //tentativa de conexão com o banco. Se der certo continua, qualquer erro vai para o catch
    {
         aConnection.Open();
         //cria o objeto datareader para fazer a conexao com a tabela
         OleDbDataReader aReader = aCommand.ExecuteReader();

         //Faz a interação com o banco de dados lendo os dados da tabela
         Console.WriteLine("Os valores retornados da tabela são : ");
         Console.WriteLine("Id\tNome\t\t\tE-mail");
         while(aReader.Read()) //faz um While para ler, uma a uma, as linhas que retornaram da select
```

```
        {
            Console.WriteLine("{0}\t{1}\t\t\t{2}", aReader[0].ToString(), aReader.GetString(1),
aReader.GetString(2));
        }

        //fecha o reader
        aReader.Close();
        //fecha a conexao
        aConnection.Close();

    }

    //Trata a exceção (erro)
    catch(OleDbException e)
    {
          Console.WriteLine("Erro: {0}", e.Errors[0].Message);
    }
}
```

Se criarmos o nosso arquivo Main() e inserirmos a chamada a esse método criado e tudo estiver correto, conseguiremos ver os dados de nossa tabela. A chamada no Main() ficaria assim:

```
public static void Main()
{

  AcessoAoAccess.Mostrar();
  Console.ReadKey();
}
```

Para nosso exemplo ficar mais interessante, vamos criar uma classe chamada Cliente. Esta classe terão os atributos iguais aos campos do banco Access e poderemos utilizá-los para carregar dados ou gravar dados no banco utilizando um objeto cliente. Também criaremos alguns métodos para facilitar alguns procedimentos. Vamos definir a classe cliente (lembrando de colocá-la no início da classe Programa, antes da classe abstrata:

```
        public class cliente
        {
            public int Id { get; set; }
            public string Nome { get; set; }
            public string Email { get; set; }


            public void NovoCliente()
            {
                Console.Write("Digite o nome: ");
                this.Nome=Console.ReadLine();
                Console.Write("Digite o email: ");
                this.Email = Console.ReadLine();
                this.Id = 0;
            }

            public void  PreencheCliente(OleDbDataReader registro) //jajá falaremos sobre esse método
            {

                this.Id = registro.GetInt32(0);
                this.Nome = registro.GetString(1);
                this.Email = registro.GetString(2);

            }

            public void Mostrar()
            {
```

```
            Console.WriteLine("Id\tNome\t\t\tE-mail");
            Console.WriteLine("{0}\t{1}\t\t\t{2}", this.Id.ToString(), this.Nome, this.Email);
        }
    }
```

Iremos mudar então o nosso método Mostrar() da classe abstrata. Insira bem no início do método a instanciação de um objeto da classe cliente:

```
cliente lido = new cliente();
```

e vamos alterar, dentro deste mesmo método, o laço While que criamos agora a pouco, que foi responsável por mostrar todos os registros que estavam no banco de dados:

```
while(aReader.Read()) //faz um While para ler de um por um as linhas que retornaram da select
{
      lido.Id = aReader.GetInt32(0);
      lido.Nome = aReader.GetString(1);
      lido.Email = aReader.GetString(2);
      Console.WriteLine("{0}\t{1}\t\t\t{2}", lido.Id.ToString(), lido.Nome, lido.Email);
      //Console.WriteLine("{0}\t{1}\t\t\t{2}", aReader[0].ToString(), aReader.GetString(1),
aReader.GetString(2));
}
```

Agora vamos executar novamente e ver que não houve nenhuma diferença aparente para o usuário, porém estamos inserindo os dados que estavam no banco em um objeto do tipo cliente. Isso será útil mais pra frente. É possível criarmos um vetor com todos os dados, mas iremos abordar isso futuramente.

Agora vamos ver como incluir dados em nosso banco de dados. Iremos utilizar o método NovoCliente criado na classe cliente que irá ser responsável por ler os dados de um novo cliente a ser inserido no banco, armazenando seu conteúdo em um objeto que, depois, será utilizado para fornecer os dados para serem transmitidos para o banco. Vamos ver o código:

```
public static void Incluir(cliente novo) //O método incluir recebe um cliente já com os dados
{

     //cria a conexão com o banco de dados
     OleDbConnection cnn = AbrirConexao();

     //cria o objeto command and armazena a consulta SQL
     OleDbCommand cmd = new OleDbCommand();
     cmd.CommandText = "Insert into Clientes (nome, email) values (\"" + novo.Nome + "\",\"" +
novo.Email + "\")";
     cmd.Connection = cnn;
     try
     {
         cnn.Open();
         //executa a inserção no banco
         cmd.ExecuteNonQuery();//como não é uma query, não retorna nada, o comando é outro
         cnn.Close();//fecha a conexão com o banco

      }

      //Trata a exceção
      catch (OleDbException e)
      {
           Console.WriteLine("Error: {0}", e.Errors[0].Message);
      }
}
```

Por estarmos utilizando uma tabela cujo campo Id é autonumerado, não utilizamos ele na string de inserção. Ele terá o conteúdo preenchido automaticamente. Fiquem atento às concatenações dos valores com a string. Neste ponto, `values (\"" + novo.Nome + "\"` encontramos a inserção de aspas (indicada por `\"`) e a junção do conteúdo (indicada por `" + novo.Nome + "`). Se o campo não for string, não necessitaríamos de colocar as aspas. Se o campo for data ou hora, ele utiliza # e para cada tipo de dados diferente, teremos uma notação específica. Esta notação muda de acordo com o banco de dados utilizado.

Vamos agora fazer a chamada no Main() do método para incluir:

```
public static void Main()
{

     cliente teste = new cliente();
     AcessoAoAccess.Mostrar();
     teste.NovoCliente();//vamos preencher os dados para enviá-los para inclusão
     AcessoAoAccess.Incluir(teste); //passa o cliente já preenchido em teste para ser inserido
     AcessoAoAccess.Mostrar();
     Console.ReadKey();

}
```

Vamos criar então um método que realiza a procura de um nome específico no banco de dados. Este método receberá como parâmetro uma string para a busca, chamada QualNome. Deveremos ler esta string no Main() e enviar para o método na hora de chamá-lo.

```
public static void Buscar(string QualNome)
        {

            cliente achado = new cliente();
            //cria a conexão com o banco de dados
            OleDbConnection cnn = AbrirConexao();

            //cria o objeto command and armazena a consulta SQL
            OleDbCommand cmd = new OleDbCommand();
            cmd.CommandText = "Select * from Clientes where Nome = \"" + QualNome + "\"";
            Console.WriteLine(cmd.CommandText);
            cmd.Connection = cnn;
            try
            {
                cnn.Open();
                //executa a procura no banco
                OleDbDataReader resultado = cmd.ExecuteReader();//traz o resultado para o DataReader
                resultado.Read();//DataReader (resultado) tem uma cópia dos dados resultantes da sql
                if (resultado.HasRows) //verifica se retornou alguma linha da SQL (se achou o nome)
                {
                    achado.PreencheCliente(resultado); //se achou o nome, preenche o cliente achado
                    achado.Mostrar();//para depois poder mostrar estes dados
                }
                else
                {
                    Console.WriteLine("Nome não encontrado");//Emite aviso caso não ache o nome
                }
                cnn.Close();//fecha a conexão com o banco

            }

             //Trata a exceção
```

```
            catch (OleDbException e)
            {
                Console.WriteLine("Error: {0}", e.Errors[0].Message);
            }

        }
```

Quando chamamos o método PreencheCliente() temos que enviar como parâmetro uma cópia dos dados resultantes da SQL executada. Para não termos que enviar um por um dos dados, podemos passar todo o resultado como parâmetro. Então, se verificar o método criado na classe Cliente, verá que ele recebe um OleDbDataReader, que é um objeto que tem estes dados.

Vamos adicionar, então, a chamada no Main() para realizarmos a busca. Lembre-se que precisamos ler o nome a ser procurado. Podemos utilizar uma variável e passá-la como parâmetro, ou simplesmente, mandar como parâmetro o próprio retorno do Console.ReadLine(), desta forma:

```
  Console.Write("Qual o nome do cliente deseja procurar:");
  AcessoAoAccess.Buscar(Console.ReadLine());//O usuário irá digitar e, o que for digitado, é enviado
a Buscar.
```

Agora que já sabemos como procurar e mostrar um cliente, podemos então criar um método que poderá dar a opção de apagar um registro procurado. Este método precisa realizar uma busca, mostrar o resultado desta busca, perguntar se deseja apagar o cliente e, somente depois, realizaremos a exclusão do registro.

Então vamos criar o método Apagar():

```
public static void Apagar(string QualNome)
{
     cliente achado = new cliente();
     string resp;
     //cria a conexão com o banco de dados
     OleDbConnection cnn = AbrirConexao();
     //cria o objeto command and armazena a consulta SQL
     OleDbCommand cmd = new OleDbCommand();
     cmd.CommandText = "Select * from Clientes where Nome = \"" + QualNome + "\"";
     Console.WriteLine(cmd.CommandText);
     cmd.Connection = cnn;
     try
     {
         cnn.Open();
         //executa a procura no banco
         OleDbDataReader resultado = cmd.ExecuteReader();
         resultado.Read();
         if (resultado.HasRows)
         {
             achado.PreencheCliente(resultado);
             achado.Mostrar();
             Console.Write("Deseja mesmo apagar este registro? (S/N)");
             resp = Console.ReadLine();//vamos perguntar de quer apagar o cliente encontrado
             if ((resp == "S") || (resp == "s"))//vamos colocar para aceitar S ou s como SIM
             {
                 resultado.Close();//se aceitou apagar, vamos fechar o Reader, para liberar o cliene
                 cmd.CommandText = "Delete from Clientes where Id = " + achado.Id; //e apagá-lo
                 cmd.ExecuteNonQuery();//mais uma vez não retorna nada, por isso NonQuery
                 Console.WriteLine("Registro apagado!");
             }
             else
             {
                 Console.WriteLine("Nome não apagado");//Se escolheu qualquer coisa que não s ou S
```

```
            }
        }
        else
        {
            Console.WriteLine("Nome não encontrado");//Se não achou o nome
        }
        cnn.Close();
    }
    //Trata a exceção
    catch (OleDbException e)
    {
        Console.WriteLine("Error: {0}", e.Errors[0].Message);
    }

}
```

Então iremos colocar no Main() a chamada para realizar a exclusão. Lembre-se que temos que fazer a solicitação do nome, tal qual na busca, antes de chamar o método para excluir.

```
            Console.Write("Qual o nome do cliente deseja apagar:");
            AcessoAoAccess.Apagar(Console.ReadLine());
            AcessoAoAccess.Mostrar();
```

Por fim, iremos agora realizar a última tarefa. A Alteração de um registro. Como não estamos utilizando um sistema GUI, esta alteração é um pouco complicada para o usuário final entender. Então vamos fazer de uma forma um pouco mais fácil. Iremos realizar uma busca inicial, tal qual para exclusão. Em seguida, iremos permitir que o usuário possa redigitar o nome e o e-mail. Caso ele não deseje alterar qualquer um dos campos, irá apertar um Enter sem digitar nada, mantendo desta forma aquele campo específico sem alteração.
Vamos criar o método de alteração:

```
  public static void Alterar(string QualNome)
        {

            cliente achado = new cliente();
            //cria a conexão com o banco de dados
            OleDbConnection cnn = AbrirConexao();

            //cria o objeto command and armazena a consulta SQL
            OleDbCommand cmd = new OleDbCommand();
            cmd.CommandText = "Select * from Clientes where Nome = \"" + QualNome + "\"";
            Console.WriteLine(cmd.CommandText);
            cmd.Connection = cnn;
            try
            {
                cnn.Open();
                //executa a busca no banco
                OleDbDataReader resultado = cmd.ExecuteReader();
                resultado.Read();
                if (resultado.HasRows) //Caso encontre o registro
                {
                    string nome, email; //criamos duas variáveis para auxiliar na alteração
                    achado.PreencheCliente(resultado);
                    achado.Mostrar();//vamos mostrar os dados encontrados
                    Console.Write("Digite o novo nome (ou enter para não alterar nada): ");
                    nome = Console.ReadLine();//e solicitar o novo nome ou enter vazio para não mudar
                    Console.Write("Digite o novo email (ou enter para não alterar nada): ");
                    email = Console.ReadLine();//e a mesma coisa para o e-mail
                    if (nome != "")//se foi digitado um novo nome, vamos trocar este nome no objeto
                        achado.Nome = nome;
                    if (email != "")//a mesma coisa para o e-mail
```

```
                        achado.Email = email;
                    resultado.Close();//agora iremos fechar o Reader para liberar o cliente e
                    cmd.CommandText = "Update Clientes set nome = \"" + achado.Nome + "\", email=\""
+ achado.Email + "\" where Id = " + achado.Id; //preencher a string que iremos usar para alterar
                    cmd.ExecuteNonQuery();//como não retorna nada, executamos com um NonQuery
                    Console.WriteLine("Registro alterado!");
                }
                else
                {
                    Console.WriteLine("Nome não encontrado");
                }
                cnn.Close();

            }

             //Trata a exceção
            catch (OleDbException e)
            {
                Console.WriteLine("Error: {0}", e.Errors[0].Message);
            }

        }
```

Por fim, vamos colocar em nosso Main() a instrução para realizar a alteração, tal qual a busca e a exclusão de registros.

```
     Console.Write("Qual o nome do cliente deseja alterar:");
     AcessoAoAccess.Alterar(Console.ReadLine());
     AcessoAoAccess.Mostrar();
```

Com estas operações, conseguimos realizar todas as operações básicas em nosso banco de dados. Existem muitas oportunidades ainda a serem exploradas.

Vamos então colocar o código completo:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data.OleDb; //necessário para realizar acesso a banco de dados do access


    class Program
    {
        public class cliente
        {
            public int Id { get; set; }
            public string Nome { get; set; }
            public string Email { get; set; }


            public void NovoCliente()
            {
                Console.Write("Digite o nome: ");
                this.Nome=Console.ReadLine();
                Console.Write("Digite o email: ");
                this.Email = Console.ReadLine();
                this.Id = 0;
            }

            public void  PreencheCliente(OleDbDataReader registro)
```

```
            {

                this.Id = registro.GetInt32(0);
                this.Nome = registro.GetString(1);
                this.Email = registro.GetString(2);

            }

            public void Mostrar()
            {
                Console.WriteLine("Id\tNome\t\t\tE-mail");
                Console.WriteLine("{0}\t{1}\t\t\t{2}", this.Id.ToString(), this.Nome, this.Email);
            }
        }

    public static class AcessoAoAccess
    {
        public static OleDbConnection AbrirConexao() //como vários métodos precisam conextar, cria um
só para facilitar esta ação
        {
            return new OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=c:\sergio\teste.mdb");
        }

        public static void Mostrar()
        {

            cliente lido = new cliente();

            //cria a conexão com o banco de dados
           OleDbConnection aConnection = AbrirConexao();

           //cria o objeto command and armazena a consulta SQL
           OleDbCommand aCommand = new OleDbCommand("select * from Clientes", aConnection);
           try
           {
                aConnection.Open();
                //cria o objeto datareader para fazer a conexao com a tabela
                OleDbDataReader aReader = aCommand.ExecuteReader();


                //Faz a interação com o banco de dados lendo os dados da tabela
                Console.WriteLine("Os valores retornados da tabela são : ");
                Console.WriteLine("Id\tNome\t\t\tE-mail");
                while(aReader.Read()) //faz um While para ler de um por um as linhas que retornaram
da select
                {
                    lido.Id = aReader.GetInt32(0);
                    lido.Nome = aReader.GetString(1);
                    lido.Email = aReader.GetString(2);

                    Console.WriteLine("{0}\t{1}\t\t\t{2}", lido.Id.ToString(), lido.Nome,
lido.Email);
                    //Console.WriteLine("{0}\t{1}\t\t\t{2}", aReader[0].ToString(),
aReader.GetString(1), aReader.GetString(2));
                }

                //fecha o reader
                aReader.Close();
                //fecha a conexao
                aConnection.Close();
```

```
            }

            //Trata a exceção
            catch(OleDbException e)
            {
                  Console.WriteLine("Error: {0}", e.Errors[0].Message);
            }
         }


        public static void Incluir(cliente novo) //O método incluir recebe um cliente já com os dados
{

     //cria a conexão com o banco de dados
     OleDbConnection cnn = AbrirConexao();

     //cria o objeto command and armazena a consulta SQL
     OleDbCommand cmd = new OleDbCommand();
     cmd.CommandText = "Insert into Clientes (nome, email) values (\"" + novo.Nome + "\",\"" +
novo.Email + "\")";
     cmd.Connection = cnn;
     try
     {
         cnn.Open();
         //executa a inserção no banco
         cmd.ExecuteNonQuery();//como não é uma query, não retorna nada, o comando é outro
         cnn.Close();//fecha a conexão com o banco

      }

      //Trata a exceção
      catch (OleDbException e)
      {
           Console.WriteLine("Error: {0}", e.Errors[0].Message);
      }
}


        public static void Buscar(string QualNome)
        {

            cliente achado = new cliente();
            //cria a conexão com o banco de dados
            OleDbConnection cnn = AbrirConexao();

            //cria o objeto command and armazena a consulta SQL
            OleDbCommand cmd = new OleDbCommand();
            cmd.CommandText = "Select * from Clientes where Nome = \"" + QualNome + "\"";
            Console.WriteLine(cmd.CommandText);
            cmd.Connection = cnn;
            try
            {
                cnn.Open();
                //executa a procura no banco
                OleDbDataReader resultado = cmd.ExecuteReader();//traz o resultado para o DataReader
                resultado.Read();//DataReader (resultado) tem uma cópia dos dados resultantes da sql
                if (resultado.HasRows) //verifica se retornou alguma linha da SQL (se achou o nome)
                {
                    achado.PreencheCliente(resultado); //se achou o nome, preenche o cliente achado
                    achado.Mostrar();//para depois poder mostrar estes dados
                }
                else
```

```
                {
                    Console.WriteLine("Nome não encontrado");//Emite aviso caso não ache o nome
                }
                cnn.Close();//fecha a conexão com o banco

            }

             //Trata a exceção
            catch (OleDbException e)
            {
                Console.WriteLine("Error: {0}", e.Errors[0].Message);
            }

        }


public static void Apagar(string QualNome)
{
     cliente achado = new cliente();
     string resp;
     //cria a conexão com o banco de dados
     OleDbConnection cnn = AbrirConexao();
     //cria o objeto command and armazena a consulta SQL
     OleDbCommand cmd = new OleDbCommand();
     cmd.CommandText = "Select * from Clientes where Nome = \"" + QualNome + "\"";
     Console.WriteLine(cmd.CommandText);
     cmd.Connection = cnn;
     try
     {
         cnn.Open();
         //executa a procura no banco
         OleDbDataReader resultado = cmd.ExecuteReader();
         resultado.Read();
         if (resultado.HasRows)
         {
             achado.PreencheCliente(resultado);
             achado.Mostrar();
             Console.Write("Deseja mesmo apagar este registro? (S/N)");
             resp = Console.ReadLine();//vamos perguntar de quer apagar o cliente encontrado
             if ((resp == "S") || (resp == "s"))//vamos colocar para aceitar S ou s como SIM
             {
                 resultado.Close();//se aceitou apagar, vamos fechar o Reader, para liberar o cliene
                 cmd.CommandText = "Delete from Clientes where Id = " + achado.Id; //e apagá-lo
                 cmd.ExecuteNonQuery();//mais uma vez não retorna nada, por isso NonQuery
                 Console.WriteLine("Registro apagado!");
             }
             else
             {
                 Console.WriteLine("Nome não apagado");//Se escolheu qualquer coisa que não s ou S
             }
         }
         else
         {
             Console.WriteLine("Nome não encontrado");//Se não achou o nome
         }
         cnn.Close();
     }
     //Trata a exceção
     catch (OleDbException e)
     {
         Console.WriteLine("Error: {0}", e.Errors[0].Message);
     }
```

```
}

        public static void Alterar(string QualNome)
        {

            cliente achado = new cliente();
            //cria a conexão com o banco de dados
            OleDbConnection cnn = AbrirConexao();

            //cria o objeto command and armazena a consulta SQL
            OleDbCommand cmd = new OleDbCommand();
            cmd.CommandText = "Select * from Clientes where Nome = \"" + QualNome + "\"";
            Console.WriteLine(cmd.CommandText);
            cmd.Connection = cnn;
            try
            {
                cnn.Open();
                //executa a busca no banco
                OleDbDataReader resultado = cmd.ExecuteReader();
                resultado.Read();
                if (resultado.HasRows) //Caso encontre o registro
                {
                    string nome, email; //criamos duas variáveis para auxiliar na alteração
                    achado.PreencheCliente(resultado);
                    achado.Mostrar();//vamos mostrar os dados encontrados
                    Console.Write("Digite o novo nome (ou enter para não alterar nada): ");
                    nome = Console.ReadLine();//e solicitar o novo nome ou enter vazio para não mudar
                    Console.Write("Digite o novo email (ou enter para não alterar nada): ");
                    email = Console.ReadLine();//e a mesma coisa para o e-mail
                    if (nome != "")//se foi digitado um novo nome, vamos trocar este nome no objeto
                        achado.Nome = nome;
                    if (email != "")//a mesma coisa para o e-mail
                        achado.Email = email;
                    resultado.Close();//agora iremos fechar o Reader para liberar o cliente e
                    cmd.CommandText = "Update Clientes set nome = \"" + achado.Nome + "\", email=\""
+ achado.Email + "\" where Id = " + achado.Id; //preencher a string que iremos usar para alterar
                    cmd.ExecuteNonQuery();//como não retorna nada, executamos com um NonQuery
                    Console.WriteLine("Registro alterado!");
                }
                else
                {
                    Console.WriteLine("Nome não encontrado");
                }
                cnn.Close();

            }

             //Trata a exceção
            catch (OleDbException e)
            {
                Console.WriteLine("Error: {0}", e.Errors[0].Message);
            }

        }



}
```

```
        public static void Main()
        {

            cliente teste = new cliente();
            AcessoAoAccess.Mostrar();
            teste.NovoCliente();
            AcessoAoAccess.Incluir(teste);
            AcessoAoAccess.Mostrar();

            Console.Write("Qual o nome do cliente deseja procurar:");
            AcessoAoAccess.Buscar(Console.ReadLine());

            Console.Write("Qual o nome do cliente deseja apagar:");
            AcessoAoAccess.Apagar(Console.ReadLine());
            AcessoAoAccess.Mostrar();

            Console.Write("Qual o nome do cliente deseja alterar:");
            AcessoAoAccess.Alterar(Console.ReadLine());
            AcessoAoAccess.Mostrar();

            Console.ReadKey();
    }

}
```