**Atividade 7.1 – Implementando Herança usando Base e classes derivadas**

Nesta atividade, você irá se familiarizar com o seguinte:

* Criando uma classe base e classes derivadas que herda estes métodos.
* Usando o modificador de acesso protected (protegido) para uso restrito do método da classe base.
* Criando uma classe base abstrata.

**Criando uma Classe base e classes derivadas**

Para criar a classe Account (Conta), siga esses passos:

1. Iniciei o Visual Studio. Selecione *File* -> *Open* -> *Project*.
2. Navegamos para a pasta Activity7\_1, clique no arquivo *Activity7\_1.sln*, e então clique *Open*. Quando abrimos o projeto, iremos desenhar um formulário chamado teller (caixa). Você irá usar este formulário mais tarde para testar as classes que você criou.
3. Na janela *Solution Explorer*, clique com botão direito no nó do Project e selecione *Add* -> *Class*.
4. No *Dialog Box* no *Add New Item*, renomeie o arquivo da classe com o nome *Account.cs*e clique *Open*. O arquivo *Account.cs*é adicionado para o projeto, e a classe *Account*, na definição do código é adicionado o arquivo.
5. Adicionamos o seguinte código para o arquivo de definição da classe para criar a variável de instância privada. (private é o modificador padrão para variáveis de instância.)

*int \_accountNumber;*

1. Adicionamos o seguinte método *GetBalance* para a definição da classe

*public double GetBalance(int accountNumber)*

*{*

*\_accountNumber = accountNumber;*

*//Data normally retrieved from database.*

*if (\_accountNumber == 1)*

*{*

*return 1000;*

*}*

*else if (\_accountNumber == 2)*

*{*

*return 2000;*

*}*

*else*

*{*

*return -1; //Account number is incorrect*

*}*

*}*

1. Depois de criar a classe *Account* (Conta), adicionamos o seguinte código para criar as classes derivadas *CheckingAccount* e *SavingAccount*:

*class CheckingAccount : Account*

*{*

*}*

*class SavingsAccount : Account*

*{*

*}*

1. Selecione *Build* -> *Build Solution.* Tenha certeza que não erros na janela *Error List*. Se tiver, então conserte-os, e então rode novamente.

**Testando as classes**

Para testar as classes, siga estes passos:

1. Abra o formulário Teller (Caixa) no editor de código e localize o botão *btnGetBalance* e no código *click event*.
2. Dentro do procedimento *event,* antes de fazer o bloco *try*, declaramos e instanciamos uma variável do tipo *CheckingAccount* chamado *oCheckingAccount,* uma variável chamada *oSavingAccount*, e uma variável do tipo *Account* chamado *oAccount*:

*CheckingAccount oCheckingAccount = new CheckingAccount();*

*SavingsAccount oSavingsAccount = new SavingAccount();*

*Account oAccount = new Account();*

1. Dependendo em que a ferramenta *radiobutton* é selecionado, chame o método *GetBalance* do objeto apropriado e passe o valor do número da conta para a ferramenta *textbox* *AccountNumber*. Localize o seguinte código no bloco *try* antes de fazer a declaração *catch:*

*if (rdbChecking.Checked)*

*{*

*txtBalance.Text =*

*oCheckingAccount.GetBalance(int.Parse(txtAccountNumber.Text)).ToString();*

*}*

*else if (rdbSavings.Checked)*

*{*

*txtBalance.Text =*

*oSavingAccount.GetBalance(int.Parse(txtAccountNumber.Text)).ToString();*

*}*

*else if (rdbGeneral.Checked)*

*{*

*txtBalance.Text =*

*oAccount.GetBalance(int.Parse(txtAccountNumber.Text)).ToString();*

*}*

1. Selecione *Build* -> *Build Solution.* Tenha certeza que não erros na janela *Error List*. Se tiver, então conserte-os, e então rode novamente.
2. Selecione *Debug* -> *Start* para rodar o projeto. Digite um número da conta de 1 e clique no botão *GetBalance* para o tipo *CheckingAccount*. Você poderá obter um balanço de 1000. Teste os outros tipos de números de contas. Você poderá obter o mesmo resultado, desde que todas as classes são usadas a mesma função *GetBalance* definidas na classe base.
3. Depois de testar, feche o formulário, com isto irá encerrar o depurador.

**Restringindo o uso de um método da classe base para estas classes derivadas**

Neste ponto, o método *GetBalance* da classe base é *public* (pública), isto significa que isto pode ser acessado pela classe derivada e estes clientes. Vamos alterar isto assim que o método *GetBalance* pode ser acessado apenas pelas classes derivadas sozinhas, e não por estes estes clientes. Para proteger o método *GetBalance* nesta maneira, siga estes passos:

1. Localize o método *GetBalance* da classe *Account*.
2. Troque o modificador de acesso do método *GetBalance* de *público* para *protegido*.
3. Mude para o editor de código frmTeller e localize o botão *btnGetBalance* no evento *Click* no código.
4. Paire o cursor no final da chamada para o método *GetBalance* do objeto *oCheckingAccount*. Você irá ver um aviso afirmando que isto é uma função protegida e neste contexto não é acessível.
5. Comente o código entre o *try* e a declaração *catch*.
6. Mude para a classe *Account.cs* no editor do código.
7. Adicione o seguinte código para criar a seguinte variável de instância privada para o arquivo da definição da classe *SavingsAccount*:

*double \_dblBalance;*

1. Adicione o seguinte método *Withdraw* para a classe *SavingsAccount.* Está função chama o método *protected* (protegido) da classe base *Account*:

*public double Withdraw(int accountNumber, double amount)*

*{*

*\_dblBalance = GetBalance(accountNumber);*

*if(\_dblBalance >= amount)*

*{*

*\_dblBalance -= amount;*

*return \_dblBalance;*

*}*

*else*

*{*

*//Not enough funds.*

*return -1;*

*}*

*}*

1. Selecione *Build* -> *Build Solution.* Tenha certeza que não erros na janela *Error List*. Se tiver, então conserte-os, e então rode novamente.

**Testando o método da classe base Protected (Protegida)**

Para testar o método *Withdraw*, siga estes passos:

1. Abra o form (formulário) *frmTeller* no editor de código e localize o botão *btnWithdraw* e dentro do código evento *Click*.
2. Dentro do procedimento *event*, antes do bloco *try*, declaramos e instanciamos uma variável do tipo *SavingsAccount* chamado *oSavingsAccount*.

*SavingsAccount oSavingsAccount = new SavingsAccount();*

1. Dentro do método *Withdraw* do objeto *oSavingsAccount.* Passe o valor número da conta do *textbox AccountNumber* e o total de retirada no *textbox Amount*. Mostra o valor de retorno no *textbox Balance*. Localize o seguinte código no bloco *try* antes de fazer para a declaração *catch*:

*txtBalance.Text = oSavingsAccount.Withdraw(int.Parse(txtAccountNumber.Text),double.Parse(txtAmount.Text)).ToString();*

1. Selecione *Build* -> *Build Solution.* Tenha certeza que não erros na janela *Error List*. Se tiver, então conserte-os, e então rode novamente.
2. Selecione *Debug* -> *Start* para rodar o projeto.
3. Teste o método *Withdraw* da classe *SavingsAccount* entrando um número de conta de 1 e total de retirada de 200. Clique no botão *Withdraw*. Você poderá obter resultanto um balanço de 800.
4. Digitando um número de conta de 1 e um total de quantidade de 2000. Clique no botão Withdraw. Você deverá obter um valor -1 indicando que terá fundos insuficiente.
5. Depois de testar o método *Withdraw,* feche o formulário (form), com isso irá fechar o depurador.

**Restringindo o uso de todos os membros de uma classe base para estas classes derivadas.**

Por causa classe base *Account* é pública, isto pode ser instanciando pelos clientes das classes derivadas. Você pode alterar estas para alterar esta fabricação da classe base *Account* uma classe abstrata. Uma classe abstrata pode ser acessada apenas por estas classes derivadas e não pode ser instanciada e acessadas por estes clientes. Para criar e testar a acessibilidade das classes abstratas, sigas estes exemplos:

1. Localize a definição da classe *Account* no código *Account.cs*.
2. Adicione a palavra-chave *abstract* para o código da definição da classe:

*abstract class Account*

1. Selecione *Build* -> *Build Solution*. Você deverá receber um erro na janela *Error List.*

*Localize a linha do código causando o erro.*

*Account oAccount = new Account();*

1. Comentando a linha do código, e selecione *Build* -> *Build Solution* novamente. Isto poderá agora rodar sem nenhum erro.
2. Salve e feche o projeto.