Introdução

Nome: Dia2Harbour
Objetivo : Agilizar a criação de classes no Harbour através de diagramas feitos no software DIA.
O que a ferramenta faz : Usando o software DIA, você cria um diagrama de classes. Após gerar o diagrama, você executa o software para poder gerar o(s) arquivo(s) fonte das classes que foram criadas no DIA. Caso você venha a alterar o seu arquivo PRG e alterar o diagrama no DIA, o que você alterou não é sobreposto, desde que você informe que tal trecho é de sua responsabilidade.

Obtendo o software DIA

O software DIA utilizado nesse tutorial foi o disponibilizado pelo site https://portableapps.com/.

O processo de trabalho

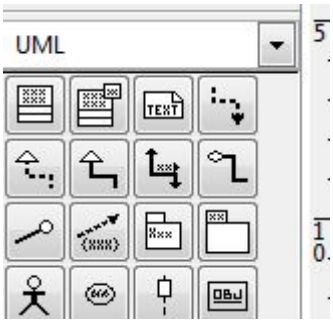
A interface do DIA é bem simples, caso você não esteja familiarizado com ela, dê uma olhada em alguns tutoriais que existem na internet. Basicamente os passos são :

1. Gerar o seu template usando o dia2harbour usando o parâmetro --create.
   Por exemplo: *dia2harbour --create teste* irá criar o arquivo teste.xml
2. Abrir o arquivo teste.xml usando o software DIA.
2. Criar e editar o seu diagrama de classes.
3. Salvar o seu diagrama de classes.
4. Executar o programa dia2harbour para gerar os PRG's usando o parâmetro --file.
   Por exemplo: *dia2harbour --file teste.xml* irá gerar um PRG para cada classe.
5. Você poderá editar os arquivos PRGs gerados. Veremos mais adiante como fazer isso sem correr o risco de perder o que você fez caso você execute novamente o dia2harbour.

O básico do DIA : Criando o seu primeiro diagrama de classes

Existem vários tutoriais disponíveis na internet ensinando a gerar diagramas no DIA, mas basicamente o que você deve saber será listado a seguir :

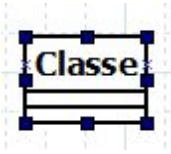
1. Você usará o menu arquivo para **abrir o template gerado pelo Dia2Harbour**.
2. Depois você deve usar a barra lateral com os elementos do seu diagrama de classes



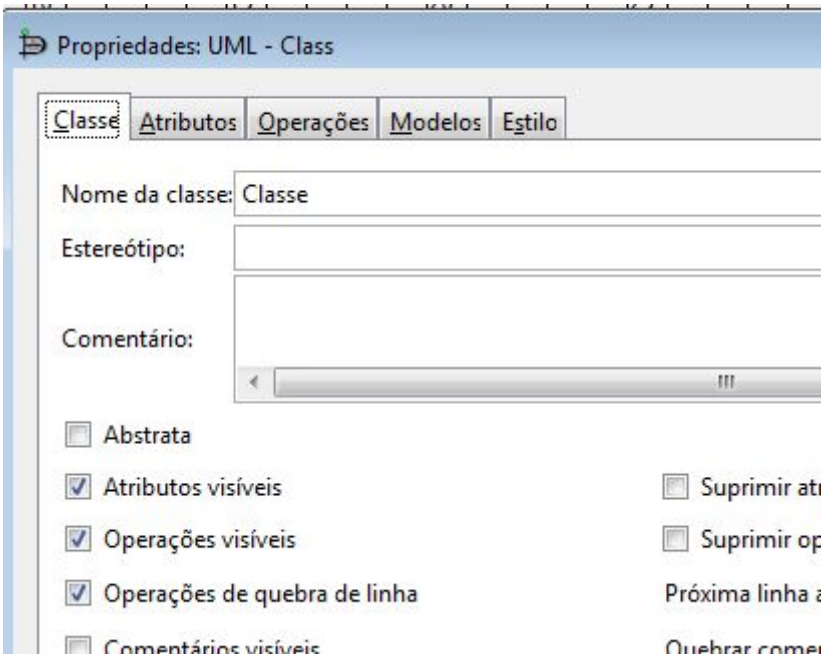
Nota: existem diversos modelos de diagramas, mas você deve usar. Você deve selecionar a opção UML.

3. Para criar uma classe, clique no símbolo abaixo e depois clique em algum local na folha quadriculada a sua direita.

4. Uma classe em branco aparecerá



5. Clique duas vezes sobre a classe, uma janela irá surgir conforme a figura abaixo

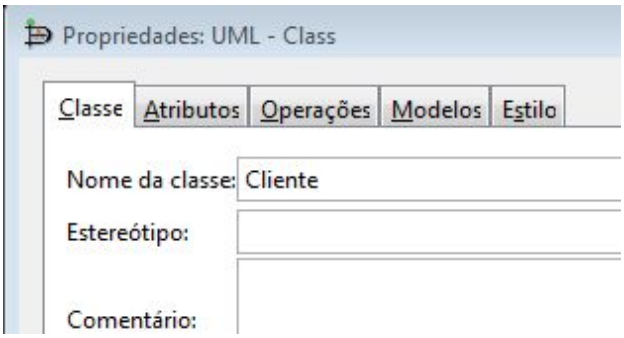


6. Você só vai precisar usar :
6.1. A aba “Classe” para informar o nome da classe.
6.2. A aba “Atributos” para informar os atributos da classe.
6.3. A aba “Operações” para informar os métodos da classe.

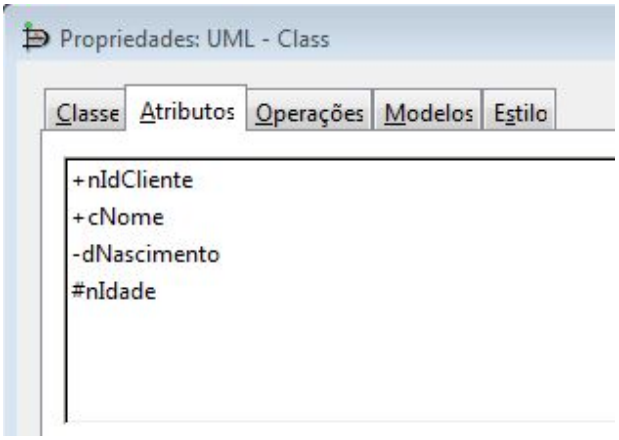
Exemplo 01 : Criando uma classe simples somente com quatro atributos.

Abra a pasta “samples”, dentro dela temos as seguintes pastas : ex01 ... ex05. Vamos treinar um pouco.

1. Abra a pasta ex01.
2. Usando o DIA, abra o arquivo cliente.xml
3. Temos ums classe “Cliente” contendo apenas três propriedades.
4. Clique duas vezes sobre a classe “Cliente” e veja como ela foi preenchida :
4.1. Na aba “Classe” temos o nome da classe.



4.2. Na aba atributos temos 4 atributos, conforme a figura abaixo :



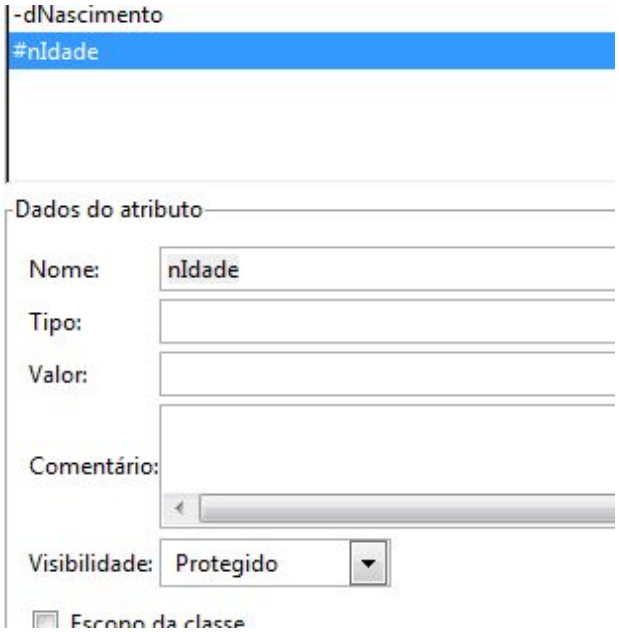
Note que, do lado direito temos 3 tipos de símbolos: + , - e #

+ Atributo público : Significa que o atributo é visível de fora da classe (evite isso)

- Atributo privado : Significa que o atributo só é visível de dentro da classe mas não é visível pelas classes filhas (herança).

Atributo protegido : Significa a mesma coisa do “-”, com a diferença que o atributo é visível pelas classes filhas (herança).

Para alterar qualquer desses atributos clique sobre ele e note que o formulário na parte inferior será preenchido com os dados do atributo, conforme a figura a seguir.



5. Para gerar o código execute o comando a seguir : *dia2harbour --file cliente.xml*

Note que um arquivo com o mesmo nome da classe será criado : *cliente.prg*

6. Analisando o arquivo *cliente.prg* note que ele é composto por subdivisões, onde cada uma delas iniciam com os símbolos *@ e terminam com o símbolos *@@. Vamos tomar como exemplo a primeira subdivisão do programa :

```
*@Includes
*Auto=Yes
#include "hbclass.ch"

*@@Includes
```

Cada uma dessas subdivisões possuem características comuns.

6.1. A abertura com o título da divisão. Por exemplo **@Includes*
6.2. Uma segunda linha com **Auto=Yes*
6.3. O código gerado, que pode ter várias linhas, depende da divisão.
6.4. O final da divisão *@@ seguido do título da divisão. No caso **@@Includes*

7. Vamos agora fazer um pequeno teste. Edite o arquivo fonte *cliente.prg* e inclua dois arquivos cabeçalhos a mais, conforme a figura abaixo :

```
*@Includes
 *Auto=Yes
 #include "hbclass.ch"
 #include "meucabecalho.ch"
 #include "meucabecalho2.ch"

*@@Includes
```

Salve o arquivo e execute no prompt de comando *dia2harbour --file cliente.xml* para regerar o código fonte a partir do diagrama de classes.

Note que as duas linhas que você inseriu sumiram.

8. Agora, inclua as duas linhas e altere a linha **Auto=Yes* para **Auto=No*, conforme abaixo :

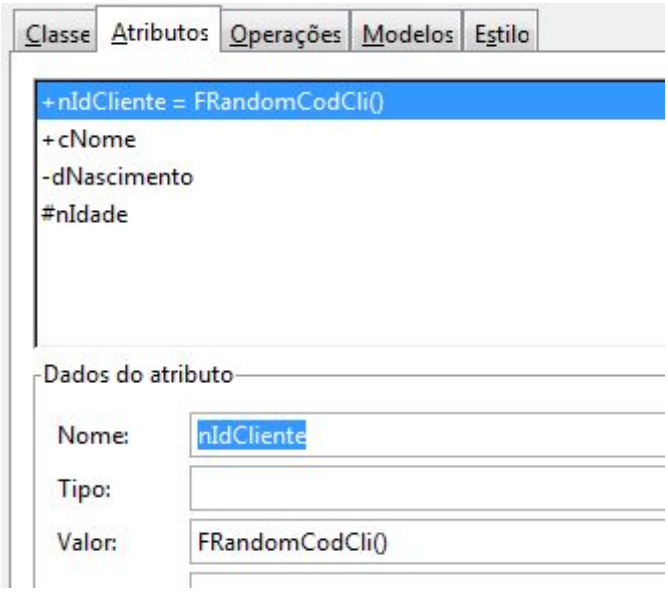
```
*@Includes
 *Auto=No
 #include "hbclass.ch"
 #include "meucabecalho.ch"
 #include "meucabecalho2.ch"
*@@Includes
```

Salve o arquivo e execute de novo *dia2harbour --file cliente.xml*

Agora sim, o programa não vai regerar o código da divisão alterada.

9. Vamos agora atribuir um valor de inicialização ao atributo *nIdCliente.* Siga o roteiro abaixo:

9.1. No software DIA clique duas vezes sobre a classe *Cliente.*
9.2. Clique na aba Atributos e inclua um valor inicial para nIdCliente (uma suposta função FRandomCli)



9.3. Salve o arquivo e execute de novo *dia2harbour --file cliente.xml*

9.4. Abra o arquivo gerado *cliente.prg* e veja que o código foi alterado na definição do atributo *nIdCliente*.

```
*@BodyClass
*Auto=Yes
    * Attributes and methods
    EXPORTED:
    DATA nIdCliente INIT FRandomCodCli()
```

Dica.

Sempre que você gerar o seu código fonte a partir do diagrama de classes uma cópia backup do código anterior ficará armazenado na pasta *backup*

A nomenclatura dos backups obedece ao seguinte critério:

nome do arquivo + data + hora + minuto + segundo

Detalhe importante!!

Procure sempre prefixar o nome do seu atributo com o tipo de dado correspondente.
Conforme os exemplos abaixo :

- cNome (nome do cliente, tipo caractere)
- nIdade (idade do cliente, tipo numérico)
- dDataNascimento (data de nascimento, tipo data)
- hItens (itens de uma lista, tipo hash)

- aItens (itens de uma lista, tipo array)

  O dia2harbour irá se utilizar dessas nomenclaturas quando for usar o método get (veja no exemplo 2)

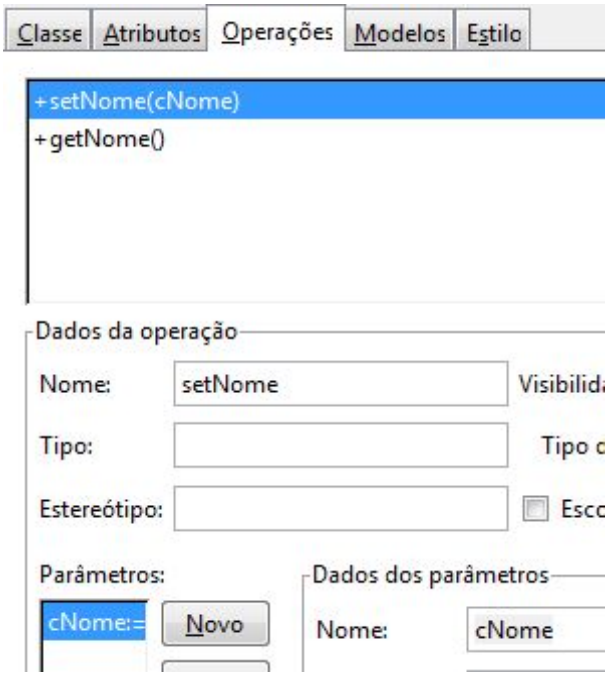
Exemplo 02 : Os métodos get e set

Se você não sabe o que são métodos get e set, faça uma busca no Google pelos termos.

Primeiro vamos aprender o método set

1. Abra a pasta ex02.
2. Usando o DIA, abra o arquivo cliente.xml
3. Temos uma classe "Cliente" contendo apenas dois atributos e duas propriedades.
4. As propriedades (na aba *Operações*) referem-se aos métodos get e set do atributo *cNome*.
5. Clique na aba "*Operações*" e depois clique sobre o método *setNome*

Note que o método *setNome* possui um parâmetro chamado *cNome*, que irá preencher o atributo *cNome* da classe.

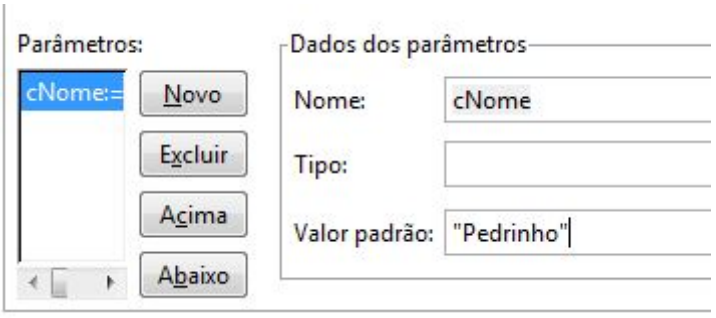


6. Salve o arquivo e execute *dia2harbour --file cliente.xml*
7. O sistema irá gerar os métodos e suas respectivas definições na classe correspondente
8. Note que o método setNome já faz a atribuição do parâmetro ao atributo da classe.

```
METHOD setNome(cNome) CLASS Cliente

*@@setNome_MethodHead

*@setNome_MethodDeclareVar
*Auto=Yes
    /* LOCAL variables */
*@@setNome_MethodDeclareVar

*@setNome_MethodDefaultValue
*Auto=Yes
    /* Default values */
*@@setNome_MethodDefaultValue


*@setNome_SetAttr
*Auto=Yes
    /* Set attribute value */
    ::cNome := cNome
    RETURN NIL
```

9. Vamos agora aprender a atribuir um valor padrão ao parâmetro *cNome* do método *setNome*.

9.1. No software DIA clique duas vezes sobre a classe cliente
9.2. Selecione o método setNome (aba Operações)
9.3. Nos dados do parâmetro digite o valor padrão desejado no campo "Valor Padrão"



9.4. Salve o diagrama e gere o seu código fonte com *dia2harbour --file cliente.xml*
9.5. Abra o código fonte *cliente.prg* e note que o método *setNome* foi alterado. Veja na figura abaixo o trecho que sofreu a inclusão do código referente ao valor padrão.

```
*@setNome_MethodDefaultValue
*Auto=Yes
    /* Default values */
    hb_Default( @cNome , "Pedrinho" )
*@@setNome_MethodDefaultValue
```

Importante !!

Para que o método set consiga atribuir corretamente o valor ao atributo da classe, o nome do parâmetro deve ser **exatamente igual** ao nome do atributo. No nosso caso *cNome*.

Agora vamos aprender o método get

1. Ainda no nosso diagrama, clique sobre o método getNome.
2. Veja que basta isso para o dia2harbour identificar que refere-se a um método get do atributo cNome.

```
METHOD getNome() CLASS Cliente

*@@getNome_MethodHead

 *@getNome_MethodDeclareVar
 *Auto=Yes
     /* LOCAL variables */
 *@@getNome_MethodDeclareVar


 *@getNome_GetAttr
 *Auto=Yes
     /* Get attribute value */
     RETURN ::cNome
 /************/
```

Detalhe técnico

O dia2harbour sempre irá pressupor que você prefixou a variável com o seu tipo. E irá comparar apenas a partir da segunda posição para achar a correspondencia entre o método get/set e o atributo.

Exemplo :

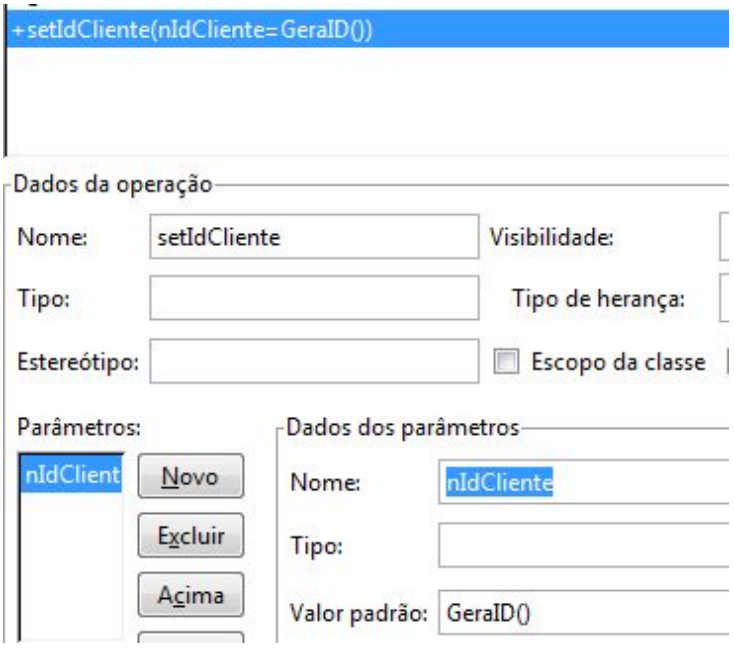
Atributo : *cNome*
Método get : *getNome()*
Método set : *setNome()*

Se você criar o atributo e não prefixar o tipo. Por exemplo: *Nome* ao invés de *cNome* o dia2harbour não vai conseguir estabelecer uma relação. O código fonte será gerado mas uma mensagem de observação será inserida dentro do seu código. Conforme a figura abaixo :

```
*@getNome_GetAttr
*Auto=Yes
   /* Get attribute value */
   // ATTRIBUTE VALUE NOT FOUND : Cliente / getNome
   RETURN NIL
*************/
```

Exemplo 03 : Atribuindo um valor padrão a um método qualquer

1. Abra a pasta ex03.
2. Usando o DIA, abra o arquivo cliente.xml e vá para a aba *Operações*
3. Para atribuir um valor padrão a um parâmetro de um método basta digitar o valor no campo correspondente ao formulário, conforme a figura abaixo.



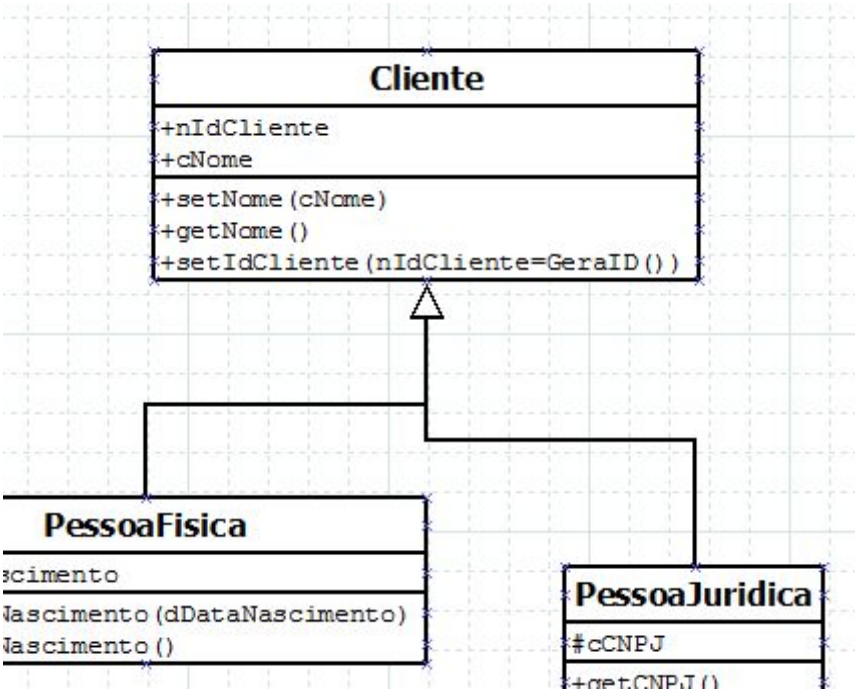
4. Salve o arquivo e execute *dia2harbour --file cliente.xml*
5. Note que, no arquivo resultante cliente.prg o trecho do código que atribui um valor padrão foi inserido dentro do método correspondente.

```
METHOD setIdCliente(nIdCliente) CLASS Cliente

*@@setIdCliente_MethodHead

*@setIdCliente_MethodDeclareVar
*Auto=Yes
    /* LOCAL variables */
*@@setIdCliente_MethodDeclareVar

*@setIdCliente_MethodDefaultValue
*Auto=Yes
    /* Default values */
    hb_Default( @nIdCliente , GeraID() )
*@@setIdCliente_MethodDefaultValue
```

Exemplo 04 : Herança

1. Abra a pasta ex04.
2. Esse diagrama é apenas para mostrar o mecanismo de herança. Para informar que uma classe é "pai" de outra você deve:
    2.1. Clique no símbolo "Generalização: herança de classe"

    2.2. Clique na classe "pai" e arraste a seta até a classe filho.
    2.3. A ponta da seta deve apontar sempre para a classe pai.



3. Salve o arquivo e execute *dia2harbour --file cliente.xml*
4. Note que serão gerados três arquivos (cada um correspondendo a uma classe), e nas classes filhas o código de herança foi adicionado, conforme abaixo:

```
CREATE CLASS PESSOAFISICA INHERIT Cliente
*@@HeadClass
```