# Objetos em Clipper

22/01/2021
Autor: Pedro Luis Kantek Garcia Navarro

Clipper ainda não é uma linguagem completamente orientada a objeto. Na verdade, as classes em Clipper já estão definidas, e não podemos criar novas. Esta abordagem tem as suas vantagens. Como a POO é uma revolução no método tradicional de programação, nada melhor do que começar aos poucos, usando classes que gerem objetos prontos e depurados.

As classes são na verdade 3: ERROR (tratamento de erros em tempo de execução), GET (operações em tela) e TBROWSE (edição de banco de dados na forma tabular). Dentro de TBROWSE, existe a classe TBCOLUMN, que é a classe de cada umas das colunas que serão manuseadas no TBROWSE.

Inicialmente, definamos a terminologia:

Um objeto é uma entidade de software que leva às últimas conseqüências a idéia de encapsulamento. Um objeto é um conjunto de funções e de dados usados pelas funções. A única maneira de conversar com um objeto é mandando-lhe mensagens, e recebendo-as de volta. O programa (nenhum programa) tem acesso às variáveis internas do objeto e muito menos ao código que está sendo executado.

Define-se CLASSE como sendo o modelo original do qual se originarão os objetos de trabalho do meu programa. Assim, diz-se classe TBROWSE, quando nos referirmos genericamente a ela, e objeto TBROWSE, quando estivermos falando do TBROWSE do meu programa que acabou de receber dados e está produzindo resultados (ou está em wait dentro do programa, mas enfim, é uma materialização da classe).

As funções que o objeto executa são chamadas MÉTODOS.

A passagem e recepção de dados denomina-se MENSAGEM. Isto é, passar um dado para um objeto, é mandar-lhe uma mensagem, e receber um resultado é aceitar uma mensagem.

Para cada classe oferecida, existe uma função especial denominada criadora,

que gera uma nova instância da classe, ou em outras palavras CRIA UM OBJETO. Este só começa a existir após a chamada à função criadora.

O objeto que acabou de ser criado HERDA todos os comportamentos (métodos) e variáveis internas da classe, e passa a ser formalmente conhecido como OCORRÊNCIA ou INSTÂNCIA, ou informalmente como objeto.

Objetos em Clipper são manipulados por sua referência. O manuseio das suas variáveis é feito através do operador SEND (:). Referências a objetos podem ser atribuídas, passadas a funções, e recebidas de volta.

A função criadora sempre devolve uma referência ao objeto criado, que deve ser guardada em uma variável. Todo o objeto passará a ser acessado via esta variável.

VARIÁVEIS PARTICULARES

Algumas são invisíveis, outras são "read-only" e finalmente outras podem ser lidas e alteradas. Estas últimas duas são conhecidas como variáveis EXPORTADAS. De qualquer maneira, a atribuição só pode ser feita através do operador SEND (:).

MENSAGENS

Mandar uma mensagem para um objeto (o que seria executar uma função do objeto, na linguagem convencional), obedece à seguinte sintaxe:

variável-referência-ao-objeto:
método chamada ([parâmetros]).

Exemplo: Suponhamos que eu tenha criado uma instância de TBROWSE na variável TELA. Para mandar fazer um avanço de página, farei TELA:PAGEUP().

Muitas vezes, há um retorno de valor, como se fosse uma chamada à função.

As variáveis exportadas podem ser lidas como:

meu-nome:=
objeto:variável-exportada

E as que podem ser alteradas, o são na forma:

objeto:variável-exportada:=

novo-valor.

Atende que a sintaxe de variável ou método é a mesma, com exceção do ().

TBROWSE

O Browse apresenta dados recuperados de arquivos, e apresenta-os na tela na forma de uma tabela, onde as colunas são os diversos campos e as linhas os diversos registros do arquivo. Os registros são recuperados através de blocos de código. Existem dois métodos de criação: NEW e DB. Veremos apenas o DB.

FUNÇÃO CRIADORA DE TBROWSE

TBROWSEDB (Li, Ci, Lf, Cf) - devolve uma referência ao objeto recém-criado.

Exemplo:

BR1:= TBROWSEDB (10, 2, 23, 78)

Só a criação de um objeto TBROWSE não resolve nada. Precisamos, na seqüência, informar quais colunas farão parte do objeto; quem faz isto é a classe TBCOLUMN, cujas instâncias devem ser criadas através de chamadas à sua função criadora TBCOLUMNNEW.

A classe TBCOLUMN não possui métodos, apenas variáveis exportadas.

FUNÇÃO CRIADORA DE TBCOLUMN

TBCOLUMNNEW (< cabeçalho da coluna>, <bloco de código>) -> retorna a instância da coluna.

O resultado da função TBCOLUMNNEW deve ser adicionado ao objeto TBROWSE, através do método ADDCOLUMN (coluna-recém-criada).

Existe um método do objeto TBROWSE, denominado stabilize() que devolve .T. quando toda a tela já foi desenhada e .F. enquanto isso não ocorrer. A finalidade é o programa poder interferir durante o preenchimento da tela (embora talvez não pareça, isto é um grande avanço).

Feito isso, vamos ao nosso primeiro programa.

```
#include “inkey.ch” // insere modulo de constantes de teclas
Funct ion main( )
local meubrowse, minhacoluna, tecla
CA( ) // cria os arquivos, se estes não existirem
set deleted on
set key k_F2 to deuctrlf2 () // associa F2 a CTRL-W
clear
@ 2,2 to 21,78 doub

use cad index cad

// montagem do objeto browse
meubrowse:=Tbrowsedb(3,3,21,77) // crio o objeto browse
meubrowse:=headsep := chr(205)+chr(203)+(205) // cabeçalhos de colunas
meubrowse:=colsep := “ “ + chr (186) + “ “ // separadores de colunas
meubrowse:=colorspec := “BG/B,GR+/R,W/N,N,GR+/W,G/B,R+/B”

minhacoluna:=Tbcolumnnew(“Codigo”,fieldblock(“codi”)) / crio coluna
minhacoluna:=defcolor := (1,2) // defino cores para ela
minhacoluna=footing := “>” // adiciono um rodape
meubrowse:addcolumn(minhacoluna) // associo esta coluna ao objeto browse
minhacoluna:=Tbcolumnnew(Nome do fregues”, fieldblock(“nome”))
minhacoluna:=defcolor := (1,3)
meubrowse:addcolumn(minhacoluna)

minhacoluna:=Tbcolumnnew(“Salário”, fieldblock(“sala”))
meubrowse:=addcolumn(minhacoluna)

minhacoluna:= Tbcolumnnew(“Memo”, {|| “Obs”})
minhacoluna:footing := “<<<”
meubrowse:addcolumn(minhacoluna)

meubrowse:freeze := 1 // a primeira coluna ficara congelada

while .T. // ciclo perpetuo

@ 23,20 say “INS=Inserir / ENTER=alterar / DELL=excluir /”
@ 24,20 say “F10=imprimir / F2=encerrar edicao”

if lastrec() == 0 // existem registro no arquivo ?
@ 10,10 say “Nao existem registros a incluir. Incluiremos agora”
lixo := inkey(0)
```

```
inclue(meubrowse)
endif

if meubrowse:colpos <=meubrowse:freze // tentou-se jogar o cursor
// coluna congelada
meubrowse=colpos := meubrowse : freeze + 1 // se sim, deslocado
endif

while (imeubrowse:stabilize())
tecla := inkey () // se algo for teclado, interrompa o enchimento da if (TECLA !=0)
// tela e processa o que foi digitado
Exit
endif
end

if (meubrowse:stable) // tudo parado
if (meubrowse:hittop .or. meubrowse.hitbottom) // tentou-se BOF () ou // EOF ()
tone (87,1) // faz barulho
tone(40,4)
endif

tecla := inkey(0) // agora que estabilizou, entro emwaitaguardando
endif // uma tecla (qualquer uma)

if (tecla == K_DOWN)
meubrowse:down() //abaixa uma linha
endif

if (tecla == K_UP) // levanta uma linha

meu browse:up()
endif

if (tecla == K+PGDN) // avanca uma tela
meubrowse:pageDown
endif

If (tecla == K_PGUP ) // volta uma tela
meubrowse:pageup()
endif

If ( tecla == K_CTRL_PGUP ) // Inicio do arquivo
meubrowse:gotop()
```

```
endif

If (tecla == K_CTRL_PGDN ) //fim do arquivo
meubrowse:gobottom()
endif

If ( tecla == K_RIGHT ) // 1 coluna a direita
meubrowse:right()
endif

... // vide demais metodos no manual

if (tecla == K_ESC ) // Se escape, então sai.
Exit
endif

If (tecla == K_ENTER ) // se ENTER
geteia (meubrowse) // vamos alterar o campo
endif
If (tecla == K_INS) // se INS
Inclua (meu browse) // vamos incluir um registro
endif
If (tecla == K_DEL) // se DEL
endif

If (tecla == K_F10) //se F10
imprimir () // imprimir o arquivo inteiro
endif
enddo

Function GETEIA (bro) // processa o ENTER (alteração)
local coluna, meuget
setcursor(1) // desenha o cursor
while(!bro:stabilize()) // espera estabilizar
end

if (bro:colpos == 6) // este é o campo memo do arquivo
clear
@ 1,1 to 22,77
repl memo with memoedit(memo, 2,2,23,78) edita o campo memo
clear
@ 2,2 to 21,78 doub
```

```
else
coluna : = brco:getcolumn(bro:colpos)
meuget :=getnew(row(), col(), coluna:block)
readmodal({meuget}) faz um get/read direto do arquivo
endif

bro:refreshcurrent() // redesenha a tela
setcursor(0) // apaga o cursor
return nil

Function INCLUE(bro)
setcursor(1)
clear
wcodi := 0
wnome : = space (30)
wsala : = 0.0
wende := space (25)
wcida:= space (15)
wmemo : = space (512)
@ 5, 10 say “Codigo ..... “ get wcodi pict “999”
@ 6, 10 say “Nome ..... “get wnome pict “@!”
@ 7, 10 say “Salario .....”get wsala pict “99,999,999,99”
@8, 10 say “Endereço .....”get wende pict “@!”
@9,10 say “Cidade .....” get wcida pict “@!”
@11,5 to 22,75 doub
read
wmemo : = memoedit(wmemo, 12,7,21,73)
appe blank
repl codi with wcodi, nome with wnome, sala with wsala, :
ende with, wende, cida with wcida
bro: refreshall
setcursor (0)
return nil

Function EXCLUDE (bro)
delete
if lastrec () == 0
@10,10 say “não existem registros a incluir. Incluiremos agora”
lixo : = inkey(0)
inclue(meubrowse)
```

```
endif
bro: refreshall()
return nil

Function IMPRIMIR
public agora, clin, tudo
aqui := savescreen (7,5,13,74) //preparando o terreno para o termometro
tudo:= 0
dbeval ({|| tudo := tudo : = tudo + 1}) // contando quantos registros tem no
arquivo
agora : = 0
clin:=80
dbeval ({||listar ()}) // execute listar () sobre cada um dos registros
set device to print
@ 0,0 say “ “ // pule de folha (otimo para impressoras laser)
set device to screen
restcreen (7,5,13,74, aqui) // redesenha o que havia por baixo termometro
return nil

Function LISTAR ()
local aa
aa := inkey()
if aa 0 .and. lastkey() == K_ESC // se se apertou ESC - caia fora
return
endif
@ clin, 3 say codif
@ clin, 10 say nome
clin ++
agora ++
set device to screen
termometro (agora, tudo, “Impressão solicitada”) // chama termometro
set device to print
return nil

Function deuctrlf2()
Keyboard chr(K_CTRL_W) // Cada vez que pressionar F2, manda um CTRL-W
return nil

Function CA()
If .not. file(“CAD.DBF”) // arquivo não existe ?
```

```
ad : = {}
aadd(ad,{“CODI”, “N”,3,0})
aadd(ad,{“NOME”, “C”, 30,0})
aadd(ad,{”SALA”,”N”,13,2})
aadd(ad,{“ENDE”,”C”,25,0})
aadd(ad,{“CIDA”,”C”,15,0})
aadd(ad,{“MEMO”,”M”,10,0})
dbcreate(“CAD.DBF”,ad) // crie-o
endif
USE CAD
if .not. file(“CAD.NTX”) // índice não existe ?
index on codi to cad // crie - o
endif
USE CAD INDEX CAD
return

**********TERMOMETRO**********
* Indica a quantas anda uma determinada a tarefa *
**********

Function termometro (xxonde, xxtudo, mensag)
local tama
@ 7,5 clear to 13,74
@ 7,5 to 13,74 doub
tama := int((xxonde/xxtudo)*60)
@ 8,30 say mensag
@ 10,9 say repl (chr (4), tama)
@ 11,9 say “ ------+-----+-----+-----+------+------+-----+-----+-----+------+”
@ 12,9 say “ 1 2 3 4 5 6 7 8 9 0
return nil
```