```
 1: unit HUtils;
 2:
 3: //=============================================================================================
 4: //
 5: //  HUtils.pas
 6: //
 7: //  Calls:
 8: //
 9: //  Called By:
10: //
11: //  Ver: 1.0.0
12: //
13: //  Date: 21 Dec 2013
14: //
15: //=============================================================================================
16:
17: {$mode objfpc}{$H+}
18:
19: interface
20:
21: uses
22:   Classes, Dialogs, Forms, SysUtils;
23:
24: // Character Validation Routines
25: function ValidAlphaCharacter( Key: char) : char;
26: function ValidCallsignCharacter( Key: char) : char;
27: function ValidDigitCharacter( Key: char) : char;
28: // Message Boxes
29: function ErrorMessageDlgOk(vstrCaption, vstrMsg  : string) : TModalResult;
30: function InfoMessageDlgOk(vstrCaption, vstrMsg  : string) : TModalResult;
31: function ConfirmationMessageDlg(vstrCaption, vstrMsg  : string) : TModalResult;
32: // Registration Routines
33: function CalculateRegistrationKey (vstrInputString : string) : string;
34:
35: implementation
36:
37: //=============================================================================================
38: //          CHARACTER VALIDATION ROUTINES
39: //=============================================================================================
40: function ValidAlphaCharacter( Key: char) : char;
41: begin
42:     // Returns only Valid Alphabetic Characters. Non-valid characters are converted
43:     // into Null (#0) characters.
44:     //Valid Alpha C haracters are:
45:     // <BS>
46:     // <SP>
47:     // [A..Z]
48:     // [a..z]
49:     Result := Key;
50:     case Key of
51:       #8 : Exit; // <BS>
52:       #32 : Exit; // <SP>
53:       #65..#90 : Exit; // [A..Z]
54:       #97..#122 : Exit; // [a..z]
55:     else
56:       Result := #0;
57:     end;// case Key of
58: end;// function ValidAlphaCharacter(var Key: char);
59:
60: //=============================================================================================
```

```
 61:  function ValidCallsignCharacter( Key: char) : char;
 62: begin
 63:     // Returns only Valid Callsign Characters. Non-valid characters are converted
 64:     // into Null (#0) characters.
 65:     //Valid Alpha C haracters are:
 66:     // <BS>
 67:     // </>
 68:     // [0..9]
 69:     // [A..Z]
 70:     // [a..z] Converted to Uppercase
 71:     Result := Key;
 72:     case Key of
 73:       #8 : Exit; // <BS>
 74:       #47 : Exit;  // </>
 75:       #48..#57 : Exit; // [0..9]
 76:       #65..#90 : Exit; // [A..Z]
 77:       #97..#122 : begin
 78:                     Result := UpCase(Key);
 79:                     Exit; // [a..z]
 80:                   end;
 81:     else
 82:       Result := #0;
 83:     end;// case Key of
 84: end;// function ValidCallsignCharacter(var Key: char);
 85:
 86: //============================================================================================
 87:  function ValidDigitCharacter( Key: char) : char;
 88:  begin
 89:      // Returns only Valid Digits. Non-valid characters are converted
 90:      // into Null (#0) characters.
 91:      //Valid Digit Characters are:
 92:      // <BS>
 93:      // [0..9]
 94:      Result := Key;
 95:      case Key of
 96:        #8 : Exit; // <BS>
 97:        #48..#57 : Exit; // [0..9]
 98:      else
 99:        Result := #0;
100:      end;// case Key of
101:  end;// function ValidDigitCharacter(var Key: char);
102:
103: //============================================================================================
104: //          MESSAGES
105: //============================================================================================
106: function ErrorMessageDlgOk(vstrCaption, vstrMsg  : string) : TModalResult;
107: begin
108:   Result := MessageDlg(vstrCaption, vstrMsg, mtError, [mbOk], 0);
109: end;// function ErrorMessageDlgOk
110:
111: //============================================================================================
112: function InfoMessageDlgOk(vstrCaption, vstrMsg  : string) : TModalResult;
113: begin
114:   Result := MessageDlg(vstrCaption, vstrMsg, mtInformation, [mbOk], 0);
115: end;// function InfoMessageDlgOk
116:
117: //============================================================================================
118: function ConfirmationMessageDlg(vstrCaption, vstrMsg  : string) : TModalResult;
119: begin
120:   Result := MessageDlg(vstrCaption, vstrMsg, mtConfirmation, [mbYes, mbNo], 0);
```

```
121: end;// function ConfirmationMessageDlg
122:
123: //=============================================================================================
124: //          REGISTRATION ROUTINES
125: //=============================================================================================
126:  function CalculateRegistrationKey (vstrInputString : string) : string;
127:
128:  var
129:    vintVal1 : Longint;
130:    vintVal2 : Longint;
131:    vintVal3 : Longint;
132:    vintVal4 : Longint;
133:    vstrTStr : string;
134:    vchrChar1 : Char;
135:    vchrChar2 : Char;
136:
137:  begin
138:    // A Registration Key is based on the Ordinal value of the first and last characters
139:    // of a string passed in the vstrInputString variable. These values are multiplied
140:    // five times to obtain at least a 10 digit integer. That integer is converted
141:    // into a string and the firt eight characters are returned as a calculated "Key"
142:    // value for that specific Input String.
143:    //
144:    // For Testing purposes, an input value of 'HU' will produce a Key of '93636000'
145:    // 'HS' will produce a Key of '89281440' and VU wil produce a Key of '13359025'.
146:
147:    if Length(vstrInputString) < 2 then
148:    begin
149:      Result := '';
150:      Exit;
151:    end;// if Length() < 2
152:
153:    vstrTStr := UpperCase(vstrInputString);
154:    vintVal1 := Ord(vstrTStr[1]);
155:    if vintVal1 < 32 then
156:      vintVal1 := 32;
157:    if vintVal1 > 90 then
158:      vintVal1 := 90;
159:    vintVal2 := Ord(vstrTStr[Length(vstrTStr)]);
160:    if vintVal2 < 32 then
161:      vintVal2 := 32;
162:    if vintVal2 > 90 then
163:      vintVal2 := 90;
164:
165:    vintVal3 := vintVal1*vintVal2*vintVal1*vintVal2;
166:    vintVal4 := vintVal1*vintVal2*vintVal1*vintVal2;
167:    Result := Copy(IntToStr(vintVal3*25),1,8);
168:
169:  end;// function ValidRegistration
170:
171: //=============================================================================================
172: end.// unit HUtils;
173:
```