```
 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: 12 Mar 2015
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 ValidAlphaNumericCharacter( Key: char) : char;
27: function ValidCallsignCharacter( Key: char) : char;
28: function ValidDigitCharacter( Key: char) : char;
29: function ValidFloatCharacter( Key: char) : char;
30: function ValidFilenameCharacter (Key: char) : char;
31: function ValidDirectoryCharacter (Key: char) : char;
32:
33: // Message Boxes
34: function ErrorMessageDlgOk(vstrCaption, vstrMsg  : string) : TModalResult;
35: function InfoMessageDlgOk(vstrCaption, vstrMsg  : string) : TModalResult;
36: function ConfirmationMessageDlg(vstrCaption, vstrMsg  : string) : TModalResult;
37: // Registration Routines
38: function CalculateRegistrationKey (vstrInputString : string) : string;
39: // Debugging Routines
40: procedure ShowInt(vstrTitle : string; vintValue : integer );
41: procedure ShowFloat(vstrTitle : string; vfltValue : double );
42:
43: const
44:   keyNull = #0;
45:
46:   keyBS = #8;
47:   keyDecimalPoint = #46;
48:   keyDEL = #127;
49:   keyFSlash = #47;
50:   keyPeriod = #46;
51:   keySingleQuote = #39;
52:   keySpace = #32;
53:   keyUScore = #95;
54:
55:   key0 = #48;
56:   key1 = #49;
57:   key2 = #50;
58:   key3 = #51;
59:   key4 = #52;
60:   key5 = #53;
```

```
 61:   key6 = #54;
 62:   key7 = #55;
 63:   key8 = #56;
 64:   key9 = #57;
 65:
 66:   keyA = #65;
 67:   keyB = #66;
 68:   keyC = #67;
 69:   keyD = #68;
 70:   keyE = #69;
 71:   keyF = #70;
 72:   keyG = #71;
 73:   keyH = #72;
 74:   keyI = #73;
 75:   keyJ = #74;
 76:   keyK = #75;
 77:   keyL = #76;
 78:   keyM = #77;
 79:   keyN = #78;
 80:   keyO = #79;
 81:   keyP = #80;
 82:   keyQ = #81;
 83:   keyR = #82;
 84:   keyS = #83;
 85:   keyT = #84;
 86:   keyU = #85;
 87:   keyV = #86;
 88:   keyW = #87;
 89:   keyX = #88;
 90:   keyY = #89;
 91:   keyZ = #90;
 92:
 93:   key_a = #97;
 94:   key_b = #98;
 95:   key_c = #99;
 96:   key_d = #100;
 97:   key_e = #101;
 98:   key_f = #102;
 99:   key_g = #103;
100:   key_h = #104;
101:   key_i = #105;
102:   key_j = #106;
103:   key_k = #107;
104:   key_l = #108;
105:   key_m = #109;
106:   key_n = #110;
107:   key_o = #111;
108:   key_p = #112;
109:   key_q = #113;
110:   key_r = #114;
111:   key_s = #115;
112:   key_t = #116;
113:   key_u = #117;
114:   key_v = #118;
115:   key_w = #119;
116:   key_x = #120;
117:   key_y = #121;
118:   key_z = #122;
119:
120: implementation
```

```
121:
122: //=============================================================================================
123: //          CHARACTER VALIDATION ROUTINES
124: //=============================================================================================
125: function ValidAlphaCharacter( Key: char) : char;
126: begin
127:     // Returns only Valid Alphabetic Characters. Non-valid characters are converted
128:     // into Null (#0) characters.
129:     //Valid Alpha C haracters are:
130:     // <BS>
131:     // <SP>
132:     // [A..Z]
133:     // [a..z]
134:     Result := Key;
135:     case Key of
136:       keyBS : Exit; // <BS>
137:       keySpace : Exit; // <SP>
138:       keyA..keyZ : Exit; // [A..Z]
139:       key_a..key_z : Exit; // [a..z]
140:     else
141:       Result := keyNull;
142:     end;// case Key of
143: end;// function ValidAlphaCharacter(var Key: char);
144:
145: //=============================================================================================
146: function ValidAlphaNumericCharacter( Key: char) : char;
147: begin
148:     // Returns only Valid Alphabetic Characters. Non-valid characters are converted
149:     // into Null (#0) characters.
150:     //Valid Alpha C haracters are:
151:     // <BS>
152:     // <SP>
153:     // [A..Z]
154:     // [a..z]
155:     // [0..9]
156:     Result := Key;
157:     case Key of
158:       keyBS : Exit; // <BS>
159:       keySpace : Exit; // <SP>
160:       key0..key9 : Exit; // [0..9]
161:       keyA..keyZ : Exit; // [A..Z]
162:       key_a..key_z : Exit; // [a..z]
163:     else
164:       Result := keyNull;
165:     end;// case Key of
166: end;// function ValidAlphaNumericCharacter(var Key: char);
167:
168: //=============================================================================================
169: function ValidCallsignCharacter( Key: char) : char;
170: begin
171:     // Returns only Valid Callsign Characters. Non-valid characters are converted
172:     // into Null (#0) characters.
173:     //Valid Alpha C haracters are:
174:     // <BS>
175:     // </>
176:     // [0..9]
177:     // [A..Z]
178:     // [a..z] Converted to Uppercase
179:     Result := Key;
180:     case Key of
```

```
181:       keyBS : Exit; // <BS>
182:       keyFSlash : Exit;  // </>
183:       key0..key9 : Exit; // [0..9]
184:       keyA..keyZ : Exit; // [A..Z]
185:       key_a..key_z : begin
186:                       Result := UpCase(Key);
187:                       Exit; // [a..z]
188:                     end;
189:     else
190:       Result := keyNull;
191:     end;// case Key of
192: end;// function ValidCallsignCharacter(var Key: char);
193:
194: //============================================================================================
195: function ValidDigitCharacter( Key: char) : char;
196: begin
197:   // Returns only Valid Digits. Non-valid characters are converted
198:   // into Null (#0) characters.
199:   //Valid Digit Characters are:
200:   // <BS>
201:   // <DEL>
202:   // [0..9]
203:   Result := Key;
204:   case Key of
205:     keyBS : Exit; // <BS>
206:     key0..key9 : Exit; // [0..9]
207:   else
208:     Result := keyNull;
209:   end;// case Key of
210: end;// function ValidDigitCharacter(var Key: char);
211:
212: //============================================================================================
213: Function ValidFloatCharacter( Key: char) : char;
214: begin
215:   // Returns only Valid Digits and the Decimal Point. Non-valid characters are converted
216:   // into Null (keyNull) characters.
217:   // Valid Digit Characters are:
218:   // <BS>
219:   // [.]
220:   // <DEL>
221:   // [0..9]
222:   Result := Key;
223:   case Key of
224:     keyBS : Exit; // <BS>
225:     keyDecimalPoint : Exit; // <.>
226:     keyDEL : Exit; // <DEL>
227:     key0..key9 : Exit; // [0..9]
228:   else
229:     Result := keyNull;
230:   end;// case Key of
231: end;// Function ValidFloatCharacter( Key: char)
232:
233: //============================================================================================
234: function ValidFilenameCharacter(Key: char) : char;
235: begin
236:     // Returns only Valid Filename Characters. Non-valid characters are converted
237:     // into Null (#0) characters.
238:     //Valid Alpha C haracters are:
239:     // <BS>
240:     // <SP>
```

```
241:     // [A..Z]
242:     // [a..z]
243:     // [0..9]
244:     // <_>
245:     Result := Key;
246:     case Key of
247:       keyBS : Exit; // <BS>
248:       keySpace : Exit; // <SP>
249:       key0..key9 : Exit; // [0..9]
250:       keyA..keyZ : Exit; // [A..Z]
251:       keyUScore : Exit; // <_>
252:       key_a..key_z : Exit; // [a..z]
253:     else
254:       Result := keyNull;
255:     end;// case Key of
256: end;// function ValidFilenameCharacter
257:
258: //============================================================================
259: function ValidDirectoryCharacter (Key: char) : char;
260: begin
261:
262:     // Returns only Valid Directory Characters. Non-valid characters are converted
263:     // into Null (#0) characters.
264:     //Valid Alpha Characters are:
265:     // <BS>
266:     // <SP>
267:     // [A..Z]
268:     // [a..z]
269:     // [0..9]
270:     // <_>
271:     Result := Key;
272:     case Key of
273:       keyBS : Exit; // <BS>
274:       keySpace : Exit; // <SP>
275:       key0..key9 : Exit; // [0..9]
276:       keyA..keyZ : Exit; // [A..Z]
277:       keyUScore : Exit; // <_>
278:       key_a..key_z : Exit; // [a..z]
279:     else
280:       Result := keyNull;
281:     end;// case Key of
282:
283: end;// function ValidDirectoryCharacter
284:
285: //============================================================================
286: //          MESSAGES
287: //============================================================================
288: function ErrorMessageDlgOk(vstrCaption, vstrMsg  : string) : TModalResult;
289: begin
290:   Result := MessageDlg(vstrCaption, vstrMsg, mtError, [mbOk], 0);
291: end;// function ErrorMessageDlgOk
292:
293: //============================================================================
294: function InfoMessageDlgOk(vstrCaption, vstrMsg  : string) : TModalResult;
295: begin
296:   Result := MessageDlg(vstrCaption, vstrMsg, mtInformation, [mbOk], 0);
297: end;// function InfoMessageDlgOk
298:
299: //============================================================================
300: function ConfirmationMessageDlg(vstrCaption, vstrMsg  : string) : TModalResult;
```

```
301: begin
302:   Result := MessageDlg(vstrCaption, vstrMsg, mtConfirmation, [mbYes, mbNo], 0);
303: end;// function ConfirmationMessageDlg
304:
305: //=============================================================================================
306: //          REGISTRATION ROUTINES
307: //=============================================================================================
308:  function CalculateRegistrationKey (vstrInputString : string) : string;
309:
310:  var
311:    vintVal1 : Longint;
312:    vintVal2 : Longint;
313:    vintVal3 : Longint;
314:    vintVal4 : Longint;
315:    vstrTStr : string;
316:
317:  begin
318:    // A Registration Key is based on the Ordinal value of the first and last characters
319:    // of a string passed in the vstrInputString variable. These values are multiplied
320:    // five times to obtain at least a 10 digit integer. That integer is converted
321:    // into a string and the firt eight characters are returned as a calculated "Key"
322:    // value for that specific Input String.
323:    //
324:    // For Testing purposes, an input value of 'HU' will produce a Key of '93636000'
325:    // 'HS' will produce a Key of '89281440' and VU wil produce a Key of '13359025'.
326:
327:    if Length(vstrInputString) < 2 then
328:    begin
329:      Result := '';
330:      Exit;
331:    end;// if Length() < 2
332:
333:    vstrTStr := UpperCase(vstrInputString);
334:    vintVal1 := Ord(vstrTStr[1]);
335:    if vintVal1 < 32 then
336:      vintVal1 := 32;
337:    if vintVal1 > 90 then
338:      vintVal1 := 90;
339:    vintVal2 := Ord(vstrTStr[Length(vstrTStr)]);
340:    if vintVal2 < 32 then
341:      vintVal2 := 32;
342:    if vintVal2 > 90 then
343:      vintVal2 := 90;
344:
345:    vintVal3 := vintVal1*vintVal2*vintVal1*vintVal2;
346:    vintVal4 := vintVal1*vintVal2*vintVal1*vintVal2;
347:    Result := Copy(IntToStr(vintVal3*25),1,8);
348:
349:  end;// function ValidRegistration
350:
351: //=============================================================================================
352: //          DEBUGGING ROUTINES
353: //=============================================================================================
354: procedure ShowInt(vstrTitle : string; vintValue : integer);
355: begin
356:   ShowMessage(' ' +
357:               vstrTitle +
358:               ' = ' +
359:               IntToStr(vintValue));
360: end;// procedure ShowInt
```

```
361:
362: //=============================================================================================
363: procedure ShowFloat(vstrTitle : string; vfltValue : double);
364: begin
365:   ShowMessage(' ' +
366:               vstrTitle +
367:               ' = ' +
368:               FloatToStrF(vfltValue, ffFixed, 5, 3));
369: end;// procedure ShowFloat
370:
371: //=============================================================================================
372: end.// unit HUtils;
373:
```