

 **Class : Protected App**

1 **Inherits** Application

 **Window : Public MainWindow** **Constants of MainWindow**

2 **Private Const** kDefaultBoardSize=8

 **Properties of MainWindow**

3 **Private** mEightQueens **As** EightQueensThread

 **Events for MainWindow**

```
4 Function CancelClose(appQuitting as Boolean) As Boolean
5     #If TargetWin32 Then
6         // Want to make sure the thread has stopped before
7         // quitting.
8         If mEightQueens <> Nil Then
9             If mEightQueens.State <> Thread.NotRunning Then
10                // Tell the thread to stop itself
11                // The timer will attempt to close again once
12                // it restarts the thread if it was suspended.
13                mEightQueens.Stop = True
14            End If
15            Return True
16        End If
17    End If
18 #Endif
19 End Function
```

```
20 Sub Open()
21     App.AutoQuit = True
22
23     BoardSizeField.Text = Str(kDefaultBoardSize)
24
25     ChessBoard.CreateChessboard(kDefaultBoardSize)
26 End Sub
```

 **Events for SolveButton**




```
27 Sub Action()
28     mEightQueens = New EightQueensThread
29     mEightQueens.Priority = Thread.LowestPriority
30
31     Dim boardSize As Integer
32     boardSize = Val(BoardSizeField.Text)
33
34     ChessBoard.CreateChessboard(boardSize)
35
36     mEightQueens.WatchSolutions = PauseCheck.Value
37     mEightQueens.BoardSize = boardSize
38     mEightQueens.Run
39
40     ChessBoardUpdateTimer.Mode = Timer.ModeMultiple
41     ChessBoardUpdateTimer.Enabled = True
42 End Sub
```

 Events for ChessBoardUpdateTimer

```

43 Sub Action()
44   #If TargetWin32 Then
45     If mEightQueens.Stop Then
46       If mEightQueens.State = Thread.Suspended Then
47         // In order to allow the thread to
48         // terminate itself, it has to be
49         // resumed.
50         mEightQueens.Resume
51       End If
52
53       Self.Close
54
55       Return
56     End If
57   #Endif
58
59   If mEightQueens.State = Thread.Suspended Then
60     // Display a solution
61     ChessBoard.DrawBoard(mEightQueens.ChessBoard)
62     ChessBoard.Text = "Solution #" + Str(mEightQueens.SolutionNumber)
63
64     // If that was the last solution then stop and display totals and times
65     If mEightQueens.Finished Then
66       ChessBoard.Text = (Str(mEightQueens.SolutionNumber) + " solutions found.") // in "
67         + Format((Ticks-start)/60, "##0.000") + " seconds"
68
69       Me.Enabled = False
70
71       mEightQueens.Resume // Thread will now end normally
72     Else
73       // Otherwise, restart thread after a short delay
74       App.SleepCurrentThread(500)
75       mEightQueens.Resume
76     End If
77   Else
78     // Update the chess board
79     ChessBoard.DrawBoard(mEightQueens.ChessBoard)
80   End If
81 End Sub

```

 Containercontrol : Public c2
 Containercontrol : Public c3
 Containercontrol : Public ChessBoardContainer

 Properties of ChessBoardContainer

```

81 Public Board(-1,-1) As String
82 Private mBoardSize As Integer
83 Private mChessBoardSquareCount As Integer

```

 Events for ChessBoard

```

84 Sub Paint(g As Graphics, areas() As REALbasic.Rect)
85   // Draw background
86   Dim rowWidth As Integer = g.Height \ mBoardSize
87   Dim colWidth As Integer = g.Width \ mBoardSize

```

```

88
89   g.ForeColor = &c000000
90   For row As Integer = 1 To mBoardSize
91     For col As Integer = 1 To mBoardSize
92       g.ForeColor = &ffffff
93       If row Mod 2 = 0 Then
94         If col Mod 2 = 0 Then
95           g.ForeColor = &cccccc
96         End If
97       Else
98         If col Mod 2 <> 0 Then
99           g.ForeColor = &cccccc
100        End If
101      End If
102      g.FillRect((col-1)*rowWidth, (row-1)*colWidth, rowWidth, colWidth)
103    Next
104  Next
105
106  For col As Integer = 1 To Ubound(Board, 1)
107    For row As Integer = 1 To Ubound(Board, 2)
108      If Board(row, col) = "Q" Then
109        g.DrawPicture(star, (col-1)*rowWidth, (row-1)*colWidth, rowWidth, colWidth, 0,
110          0, star.Width, star.Height)
111      End If
112    Next
113  Next
114 End Sub

```

Methods/Procedures

```

114 Public Sub CreateChessboard(boardSize As Integer)
115   mBoardSize = boardSize
116
117   ChessBoard.Refresh(False)
118 End Sub

```

```

119 Public Sub DrawBoard(b(,) As String)
120   Board = b
121
122   ChessBoard.Refresh(False)
123 End Sub

```

```

124 Public Sub Text(assigns t As String)
125   SolutionLabel.Text = t
126   SolutionLabel.Refresh
127 End Sub

```

ContainerControl : Public ContainerControl1

Thread : Protected EightQueensThtf

```
128 Inherits Thread
```

About the 8 Queens Problem

```

129 The 8 Queens Problem is a common challenge given to first year computer science students.
130
131 On a chessboard (which is an 8x8 grid) there are a finite number of ways that 8 queens
132 can be placed. If you don't know chess, a queen can move any number of spaces in
133 any direction, horizontal, vertical and diagonal. There are 92 possible ways

```

```

134 to place 8 queens on a standard 8x8 chessboard chessboard.
135
136 This program uses a brute-force algorithm to repeatedly try solutions until it finds all 92
.

```

✓ Properties of EightQueensThtf

```

137 Public ChessBoard(8,8) As String
138 Private mBoardSize As Integer
139 Private mDiagonalArrayOffset As Integer
140 Private mDiagonalDifference(16) As Boolean
141 The DiagonalDifference array tracks squares on the board from a upper left to lower right
diagonal.
142
143 It works the same way as the DiagonalSum, but we instead take the difference of the column
and row.
144 Note that this results in a negative number for many squares and since REALbasic does not
allow
145 negative values for array indices, we simple offset it by 7.
146 End Property

```

```

147 Private mDiagonalSum(16) As Boolean
148 The DiagonalSum array identifes all the squares on the board that are on a lower-left to to
right diagonal.
149
150 For example, look at column 2, row 4. It's sum (column+row) is 6. Now also look at column
5, row 1. It is on the diagonal
151 for column 2, row 4. And it's sum is also 6. So when we find a place to put a Queen, we
also set
152 the DiagonalSum value to True. This allows us to quickly find diagonal "hits" for future
possible positions.
153
154 1 2 3 4 5 6 7 8
155 1 + + + + + + + +
156 2 + + + + + + + +
157 3 + + + + + + + +
158 4 + + + + + + + +
159 5 + + + + + + + +
160 6 + + + + + + + +
161 7 + + + + + + + +
162 8 + + + + + + + +
163 End Property

```

```

164 Private mDisplay As Boolean
165 Indicates whether solutions should be displayed.
166 End Property

```

```

167 Private mFinished As Boolean
168 Private mPicked(8) As Integer
169 The Picked array identifies the row containing a Queen for each column on the board.
170
171 Picked(2) = 6 indicates the the Queen is on row 6 in column 2.
172 End Property

```

```

173 Private mRowCheck(8) As Boolean
174 The RowCheck array contains a value of True if a Queen was placed in the row.
175 Otherwise it is False.

```

```

176
177 If a Queen is in a row then we can immediately move to the next row.
178 End Property

```

```

179 Private mSolutionNumber As Integer
180 Public Stop As Boolean
181 Public WatchSolutions As Boolean

```

Computed Properties of EightQueensThtf

```

182 Public BoardSize As Integer
183 Get
184     Return mBoardSize
185 End Get
186 Set
187     mBoardSize = value
188 End Set
189 End Property

```

```

190 Public Finished As Boolean
191 Get
192     Return mFinished
193 End Get
194 notes on finished
195 End Property

```

```

196 Public SolutionNumber As Integer
197 Get
198     Return mSolutionNumber
199 End Get
200 End Property

```

Events for EightQueensThtf

```

201 Sub Run()
202     Solve
203 End Sub

```

Methods/Procedures

```

204 Private Sub Backup(ByRef row As Integer, ByRef column As Integer)
205     // Go back one column and try the next row
206     // If the next row is higher than the board size
207     // then we go back another column.
208     // If we reach column 0 then we've found all the solutions.
209     Do
210         column = column - 1
211
212         If column > 0 Then
213             // Clear the current column settings since we've just
214             // removed the queen.
215             row = mPicked(column)
216             ChessBoard(row, column) = ""
217             mPicked(column) = 0
218             mRowCheck(row) = False
219             mDiagonalSum(column+row) = False
220             mDiagonalDifference(column-row+mDiagonalArrayOffset) = False
221
222             // Try the next row

```

```
223     row = row + 1
224   End If
225 Loop Until row <= mBoardSize Or column = 0
226
227   msgbox if(column>99,"true","false")
228
229   select case 1
230   case 1
231     msgbox "a"
232   case 2
233     msgbox "b"
234   case else
235     msgbox "c"
236   end select
237
238
239   select case 1
240   case 1
241     msgbox "a"
242   case 2
243     msgbox "b"
244   else
245     msgbox "c"
246   end select
247
248
249   passwordInsert.sqlexecute(EncodeHex(generatorandomstring(bytecount,1111,2222,3333)),
EncodeHex(generatorandomstring(bytecount,4444,5555,6666)),EncodeHex(generatorandomstring
(bytecount,7777,8888,9999)),EncodeHex(generatorandomstring(bytecount,1111,2222,3333)),
EncodeHex(generatorandomstring(bytecount,4444,5555,6666)),EncodeHex(generatorandomstring
(bytecount,7777,8888,9999)),EncodeHex(generatorandomstring(bytecount,1111,2222,3333)),
EncodeHex(generatorandomstring(bytecount,4444,5555,6666)),EncodeHex(generatorandomstring
(bytecount,7777,8888,9999)),EncodeHex(generatorandomstring(bytecount,1111,2222,3333)),
EncodeHex(generatorandomstring(bytecount,4444,5555,6666)),EncodeHex(generatorandomstring
(bytecount,7777,8888,9999)),EncodeHex(generatorandomstring(bytecount,1111,2222,3333)),
EncodeHex(generatorandomstring(bytecount,4444,5555,6666)),EncodeHex(generatorandomstring
(bytecount,7777,8888,9999))
250 End Sub

```

```
251 Private Sub ControlBackups(ByRef row As Integer, ByRef column As Integer, ByRef isFound As
Boolean)
252   // If we've reached the size of the board and have a solution
253   // then show it
254   If column = mBoardSize And isFound Then
255     DisplaySolution
256     isFound = False
257     column = mBoardSize + 1
258     Backup(row, column)
259   Else
260     // We've reached the end of the board and have no more
261     // room for queens. It's time to back up and
262     // try again.
263     If row > mBoardSize Then
264       Backup(row, column)
265     End If
266   End If

```

```
267 End Sub
268 Private Sub DisplaySolution()
269     mSolutionNumber = mSolutionNumber + 1
270
271     If WatchSolutions Then Self.Suspend
272 End Sub
273 Private Sub FindSolutions()
274     Dim start As Integer = 1
275     Dim solutionNumber As Integer
276     Dim row As Integer
277     Dim column As Integer
278
279     Dim isFound As Boolean
280     Dim allDone As Boolean
281
282     // Start with the upper left corner of the chessboard
283     mPicked(1) = start
284     mRowCheck(start) = True
285     mDiagonalSum(1+start) = True
286     mDiagonalDifference(1-start+mDiagonalArrayOffset) = True
287
288     // We can immediately start with column 2 since we've already placed
289     // queen in column 1
290     column = 2
291
292     Do
293         row = 1
294         Do
295             If WatchSolutions Then
296                 // Uncomment to watch as it works through solutions
297                 'App.SleepCurrentThread(1)
298             End If
299
300             isFound = False
301
302             If column > 0 Then
303                 // If there is no queen on the same row or diagonal for this column
304                 // and row then we can try a queen here.
305                 If Not mRowCheck(row) And Not mDiagonalSum(column+row) And _
306                     Not mDiagonalDifference(column-row+mDiagonalArrayOffset) Then
307
308                     isFound = True
309                     mRowCheck(row) = True
310                     ChessBoard(row, column) = "Q"
311                     mDiagonalSum(column+row) = True
312                     mDiagonalDifference(column-row+mDiagonalArrayOffset) = True
313                     mPicked(column) = row
314                 Else
315                     // Try the next row in the column
316                     row = row + 1
317                 End If
318
319                 // We need to backup if we've reached beyond the last row
320                 ControlBackups(row, column, isFound)
```

```
321 |         }Else
322 |         |         isFound = True
323 |         |         allDone = True
324 |         |     End If
325 |     Loop Until isFound
326 |
327 |         column = column + 1
328 |
329 | Loop Until allDone Or Stop
330 |
331 | mFinished = True
332 | End Sub
```


```
333 | Public Sub Solve()
334 |     mBoardSize = boardSize
335 |     mDiagonalArrayOffset = mBoardSize-1
336 |
337 |     ReDim mPicked(mBoardSize)
338 |     ReDim mRowCheck(mBoardSize)
339 |     ReDim mDiagonalDifference(mBoardSize*2)
340 |     ReDim mDiagonalSum(mBoardSize*2)
341 |     ReDim ChessBoard(mBoardSize, mBoardSize)
342 |
343 |     ChessBoard(1, 1) = "Q"
344 |
345 |     FindSolutions
346 | End Sub
```

 **Thread : Protected t1**

347 | Inherits Thread

 **Class : Protected abc**

 **Class : Protected def**

 **Interface : Protected intfce**

348 | Implements DataSource

 **Report : Global rpt**

Properties of rpt

349 | Public Shared Untitled As Integer

 **Menu Handler(s) for rpt**

```
350 | Function FileQuit() As Boolean Handles FileQuit.Action
351 |     msgbox "quit"
352 |     Return True
353 | End Function
354 |
```


A

abc.....8
 Action.....1-2,8
 allDone.....7-8
 App.....1-2
 Application.....1
 appQuitting.....1
 areas.....2

B

Backup.....5-6
 Board.....2-3
 boardSize.....1,3,5,8
 BoardSizeField.....1
 bytecount.....6

C

c2.....2
 c3.....2
 CancelClose.....1
 ChessBoard.....1-5,7-8
 ChessBoardContainer.....2
 ChessBoardUpdateTimer.....1

Class(s)

abc.....8
 App.....1
 def.....8
 col.....3
 column.....5-8
 colWidth.....2-3

Containercontrol(s)

c2.....2
 c3.....2
 ChessBoardContainer.....2
 ContainerControl1.....3
 ContainerControl1.....3
 ControlBackups.....6-7
 CreateChessboard.....1,3

D

DataSource.....8
 def.....8
 DisplaySolution.....6-7
 DrawBoard.....2-3

E

EightQueensThread.....1
 EightQueensThtf.....3

F

FileQuit.....8
 FindSolutions.....7-8
 Finished.....2,5
Function(s)

CancelClose.....1

F

FileQuit.....8

G

g.....2-3
 generaterandomstring.....6

IInterface(s)

intfce.....8
 intfce.....8
 isFound.....6-8

K

kDefaultBoardSize.....1

M

MainWindow.....1
 mBoardSize.....2-6,8
 mChessBoardSquareCount.....2
 mDiagonalArrayOffset...4-5,7-8
 mDiagonalDifference...4-5,7-8
 mDiagonalSum.....4-5,7-8
 mDisplay.....4
 mEightQueens.....1-2
 mFinished.....4-5,8
 mPicked.....4-5,7-8
 mRowCheck.....4-5,7-8
 mSolutionNumber.....5,7

O

Open.....1

P

Paint.....2
 passwordInsert.....6
 PauseCheck.....1

R

REALbasic.....2

Report(s)

rpt.....8
 row.....3,5-7
 rowWidth.....2-3
 rpt.....8
 Run.....1,5

S

SolutionLabel.....3
 SolutionNumber.....2,5,7
 Solve.....5,8
 star.....3
 start.....7

S

Stop.....1-2,5,8

Sub(s)

Action.....1-2
 Backup.....5
 ControlBackups.....6
 CreateChessboard.....3
 DisplaySolution.....7
 DrawBoard.....3
 FindSolutions.....7
 Open.....1
 Paint.....2
 Run.....5
 Solve.....8
 Text.....3

T

t.....3
 t1.....8
 Text.....1-3

Thread(s)

EightQueensThtf.....3
 t1.....8
 Timer.....1

U

Untitled.....8

V

Value.....1,5

W

WatchSolutions.....1,5,7

Window(s)

MainWindow.....1