HW07 Problem Set

CS-3020

Assignment
Deitel & Deitel Exercises 14.7, 14.8

HW07-1: (Deitel & Deitel Exercise 14.7)

14.7 ***(Enhanced Painter)*** Extend the program of Fig. 14.38 to include options for changing the size and color of the lines drawn. Create a GUI similar to Fig. 14.43. The user should be able to draw on the app's `Panel`. To retrieve a `Graphics` object for drawing, call method *panelName*`.CreateGraphics()`, substituting in the name of your `Panel`.

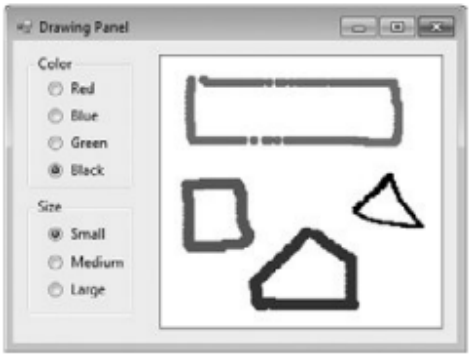

Fig. 14.43 | Drawing Panel GUI.

HW07-2: (Deitel & Deitel Exercise 14.8)

14.8 ***(Guess the Number Game)*** Write a program that plays "guess the number" as follows: Your program chooses the number to be guessed by selecting an `int` at random in the range 1–1000. The program then displays the following text in a label:

```
I have a number between 1 and 1000--can you guess my number?
Please enter your first guess.
```

A `TextBox` should be used to input the guess. As each guess is input, the background color should change to red or blue. Red indicates that the user is getting "warmer," blue that the user is getting "colder." A `Label` should display either "`Too High`" or "`Too Low`," to help the user zero in on the correct answer. When the user guesses the correct answer, display "`Correct!`" in a message box, change the `Form`'s background color to green and disable the `TextBox`. Recall that a `TextBox` (like other controls) can be disabled by setting the control's `Enabled` property to `false`. Provide a `Button` that allows the user to play the game again. When the `Button` is clicked, generate a new random number, change the background to the default color and enable the `TextBox`.

```
1  // Fig. 14.38: PainterForm.cs
2  // Using the mouse to draw on a Form.
3  using System;
4  using System.Drawing;
5  using System.Windows.Forms;
6
7  namespace Painter
8  {
9     // creates a Form that's a drawing surface
10    public partial class PainterForm : Form
11    {
12       bool shouldPaint = false; // determines whether to paint
13
14       // default constructor
15       public PainterForm()
16       {
17          InitializeComponent();
18       } // end constructor
19
20       // should paint when mouse button is pressed down
21       private void PainterForm_MouseDown(
22          object sender, MouseEventArgs e )
23       {
24          // indicate that user is dragging the mouse
25          shouldPaint = true;
26       } // end method PainterForm_MouseDown
27
28       // stop painting when mouse button is released
29       private void PainterForm_MouseUp( object sender, MouseEventArgs e )
30       {
31          // indicate that user released the mouse button
32          shouldPaint = false;
33       } // end method PainterForm_MouseUp
34
35       // draw circle whenever mouse moves with its button held down
36       private void PainterForm_MouseMove(
37          object sender, MouseEventArgs e )
38       {
39          if ( shouldPaint ) // check if mouse button is being pressed
40          {
41             // draw a circle where the mouse pointer is present
42             using ( Graphics graphics = CreateGraphics() )
43             {
44                graphics.FillEllipse(
45                   new SolidBrush( Color.BlueViolet ), e.X, e.Y, 4, 4 );
46             } // end using; calls graphics.Dispose()
47          } // end if
48       } // end method PainterForm_MouseMove
49    } // end class PainterForm
50 } // end namespace Painter
```

Fig. 14.38 | Using the mouse to draw on a `Form`. (Part 2 of 2.)

HW07 Problem Set

CS-3020

Grading Rubric

Each problem is worth 10 pts (score will be recorded as a percentage of that amount)

10% Properly submitted
10% Properly named
20% Adequate comments
10% Runs
20% Produces correct output
30% Effort evidenced by the submitted work