CIS 345 – Business Information Systems Development II – Fall 2014

In-class Exercise 12: Painting Program

Due on Blackboard: Today, Monday, November 10 by 10 PM

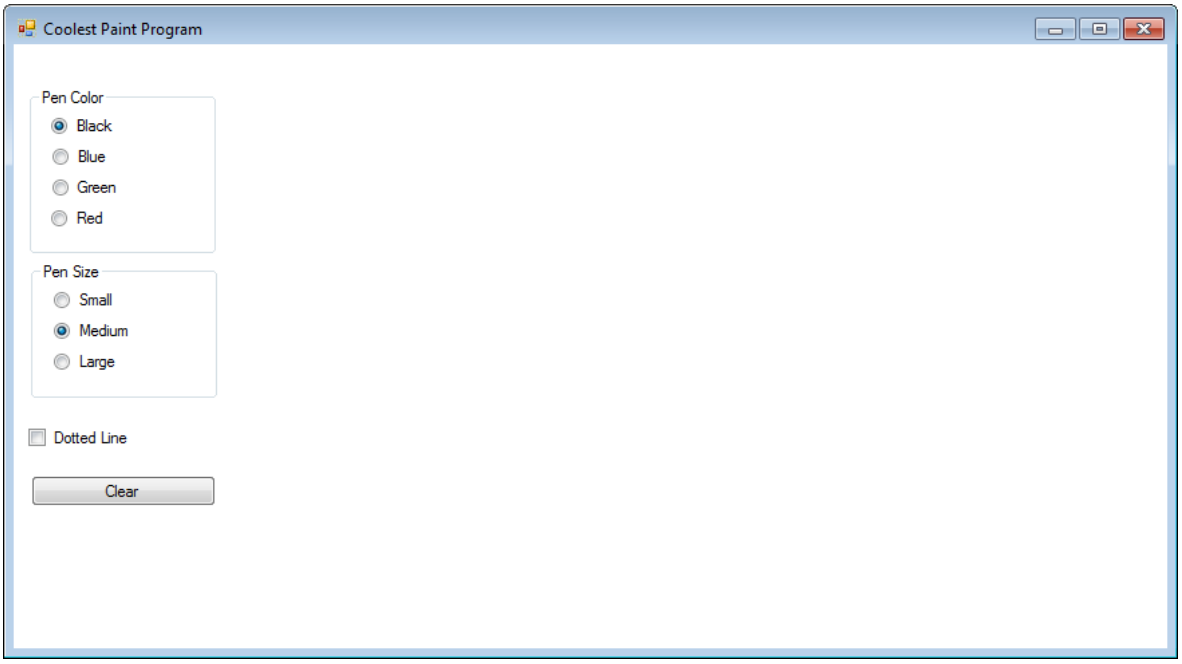
Create a simple paint application that allows a user to select a color and line width. The user should be able to draw directly on the Form. This requires the implementation of writing logic as well as responding to mouse events. Use the RadioButton and CheckBox controls and respond to their events as well.

Project Details and Requirements

A sample executable file is uploaded and available on Blackboard for you to test program behavior.

Project Name: Paint
Form Name: Paint

Form Paint [Design]



Properties: Since Paint inherits from Form, a host of inherited properties are already available. The following properties will need to be set at design time using the IDE. Modify your form to look like that in the sample.

- Text (this property sets the form title): "Paint Program"
- FormBorderStyle: FixedSingle
- MaximizeBox: False
- MinimizeBox: False
- StartPosition: CenterScreen
- BackColor: White (use Web Colors)

1) Set up the Paint Form as show in the figure above. Use suffixes within the control names so that when you work with in code, you can easily distinguish a TextBox from a Label. e.g. “” should easily be interpreted as the TextBox which contains a guess, while “guessLabel” should easily suggest that is the label accompanying a guess.

Using the IDE, create the following controls. Set properties as specified.

- Two GroupBox controls, for housing radio buttons
    - Naming convention: suffix group boxes with “GroupBox”, e.g. colorGroupBox
- Radio Buttons, for selecting a color
    - Naming convention: suffix name with “RadioButton”, e.g., redRadioButton
      Note: All Radio Buttons should be put *inside* a GroupBox.
      For the *Black* radio button, set the Checked property to true.

- Radio Buttons, for selecting a pen width
    - Naming convention: suffix name with "RadioButton", e.g., smallRadioButton
      Note: <u>All Radio Buttons</u> should be put *inside* a GroupBox.
      For the *Small* radio button, set the Checked property to true.
- CheckBox, to select a dashed line
    - Naming convention: suffix name with "CheckBox"
- Button, to clear the screen.
    - Naming convention: suffix name with "Button"

<u>Form Paint [Paint.cs Partial Class]</u>

Do not add any code in the Designer.cs or Program.cs classes.

1) **Instance variables**
   Within the Paint.cs partial class, add private ***instance*** variables for the following.

   1 Color variable, penColor
   1 Pen variable, myPen.
   2 Point variables, previousPoint and currentPoint
   A point object represents a point on the form. It will eventually have a set of X and Y axes values. The two points will store where the mouse has been located on the Form for the last two times.

   1 Graphics variable, myGraphics.
   1 boolean, keepDrawing

METHODS:

Methods should go inside the class as with all objects. The precise location (top, bottom, etc.) does not matter so long as methods are located within the class block.

Create event handler methods using the Properties window.

2) Load event handler.

Create a Load event handler for the form.

Purpose: The purpose of this event handler is to initialize the instance variables, objects, and anything else that needs to be done before the user starts using the program. Load is one of the first events that happens when a Windows Form is "loaded" and displayed to the user.

General Method Logic:

- Instantiate previousPoint and currentPoint by creating a new instance of Point, using the point default/parameterless constructor. *This is to keep track of the mouse pointer.*
- Assign the Black color to the penColor variable. To do this, access the static Black property of the Color class and assign it to penColor. *Setting default color to black.*
- Set keepDrawing to false. *Assuming user doesn't want to draw anything.*
- Instantiate a new Pen object by calling the constructor. Store the reference in myPen. Provide the color (it's stored in your Color variable!) as the argument. *We are making sure we have pens ready!*
- Access the CreateGraphics method of the Form (use "this") – it will return to you an object of type Graphics. Assign it to your Graphics instance variable. *This will store a reference to the Graphics object of the form. We will later be able to use it to draw on the Form.*

3) CheckedChanged event handlers

Purpose: The CheckedChanged event handler method is called every time the radio button is checked OR unchecked.

Create a CheckedChanged event handler for <u>every</u> RadioButton that you have (seven!)

NOTE: All radio buttons have identical logic – just customize the logic for each button based on what that radio button is being used for.

*General Method logic for **Color-related** radio buttons.*

- If the radio button is checked (use the *Checked* property, which returns a boolean value)
    - Set your Color instance variable to the color that the user selected.
      e.g. if greenRadioButton is checked, then penColor should store the color Green.
      Access the static property Green of the Color class.

*General Method logic for **Width-related** radio buttons.*

- If the radio button is checked (use the *Checked* property, which returns a boolean value)
    - Set the Width property of myPen according to what the user selected.
      Use widths 2, 4, and 6 for small, medium, and large sizes.

4) MouseDown event handler for the Form

Purpose: The MouseDown event handler method is called when the user presses (but has not released!) the Mouse button (usually the left button on multi-button mice).

General Method logic:

- Set keepDrawing to true. *User has pressed the mouse button to draw! So, set the variable to true.*
- *Keep a track of where the user clicked on the form.* Create a new Point using the Point constructor. The point constructor will want X and Y coordinates. Get X and Y coordinates from the parameter "e" which is of type MouseEventArgs. You should be able to access the X and Y properties of "e". Store the reference to the Point object in currentPoint. *It's the new "current" point!*

5) MouseUp event handler for the Form

Purpose: The MouseUp event handler method is called when the user releases the Mouse button.
General Method logic:

- Set keepDrawing to false. *User has released the mouse button! She doesn't want to draw anymore!*

6) MouseMove event handler for the Form

Purpose: The MouseMove event handler method is called when the user moves the Mouse button. It is therefore called very frequently and any code that goes in here must be very carefully managed so that it executes only whenever you really want it executed since the mouse could potentially be moving all the time.

General Method logic:

- If the user wants to draw (use your boolean draw variable to test this)
    - Assign the "current" point variable to the "previous" one. *There were two Point objects – previous and current. This will store in the "previous" variable the value of the "currently" variable. The "current" point is thereby now the "previous" one.*
    - *Now that the "current" Point location is stored in the "previous" variable, the "current" variable needs a new Point location. The MouseEvent handler can tell you where the user clicked on the screen.* Call the Point constructor and create a new Point. Access the X and Y properties of the parameter "e", which is of type MouseEventArg, and pass those as arguments to the constructor. Store the reference to the newly created Point object in your "current" Point variable.
    - Set the Color property of the myPen object to penColor. *This is done in case the user has changed the color of the pen. If the user did change it, the CheckChanged event handlers will have stored the new color in the penColor variable.*
    - Call the DrawLine method of the Graphics instance variable that you have. It will require three arguments. Give it your pen variable, your "previous" point, and your "current" point. *So, it will use the pen, which has in it the color and the width, to draw a line from the previous point to the current point on the screen.*

6) CheckedChanged event handler for the CheckBox

Purpose: The CheckedChanged event handler method is called whenever the checkbox is selected or unselected.

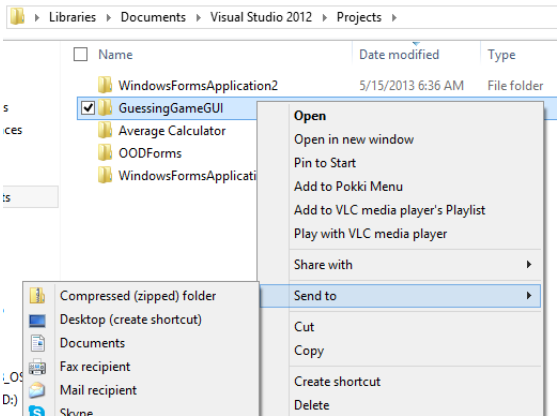
General Method logic:

- If the checkbox is checked (use the Checked property which returns a boolean)
    - Set the DashStyle property of myPen to System.Drawing.Drawing2D.DashStyle.Dot
- Else
    - Set the DashStyle property of myPen to System.Drawing.Drawing2D.DashStyle.Solid

Submission Instructions

Submission should be made using a zip file that contains all of the Visual Studio C# project files. You will need to ***zip the entire project folder*** along with the .sln and .suo files. The folder will automatically contain the class source files as well as the executable file that is generated in \ProjectName\bin\Debug folder. Upload file to the Blackboard assignment drop box.

Zip the entire top-level folder by right-clicking the folder and selecting Send to | Compressed (zipped) folder.



Using built-in windows zip tools: http://windows.microsoft.com/en-us/windows/compress-uncompress-files-zip-files

Make sure you check the following. Your grade is dependent on all these criteria being met.

- You have included your name as a comment within your class.
    - e.g. " // In-class 11, Jane C. Smith, CIS 345, Tuesday 9:00 AM"
    - Your project is called Paint
- Zip filename is: FirstNameLastName_Inclass11.zip

Verify your zip file before you submit

- Check for actual class files being present in the folder before you zip it.
- Check your zip file size after zipping – if it is 1K, it likely contains only a shortcut.
- Uncompress your zip file before submitting and verify that files are present.
- Download your zip file after submitting, uncompress, and again verify that your files are present. Test your files in Visual Studio after uncompressing.
- Make sure you have submitted your file and not just saved a draft on Blackboard. A blue clock indicates a submission in progress, i.e. a draft, not a submission.