Programming Techniques

Project 1 Part 2: Computerized Gallery Systems (CGS)

INTRODUCTION

This project is the second part of the last in a series of four, which you will complete in the Phase 1 courses. In the projects' scenario, you are one member of a software development team responsible for the design and development of a business solution for Computerized Gallery Systems (CGS).

The CGS is a proposed computerized system for keeping track of the works of art shown and sold in a private art gallery. It includes facilities for keeping track of each work of art from the time it enters the system. This system runs on a single computer.

The current application was designed and developed using structured programming techniques. The company for which you work has decided to update existing programs using object-oriented and structured programming techniques.

YOUR ROLE

You are assigned to create an application with a graphical interface using Windows forms and components and an object-oriented programming language.

OBJECTIVES

The objectives of this project are:

- Create an application with a graphical interface
- Write an event-driven program using an object-oriented language
- Use a custom class library to demonstrate reusability
- Read from a file
- Write to a file
- Use the three structured theorem programming constructs where applicable
- Debug and handle errors to produce an error-free application

TIME REQUIRED

You will require 15 hours to complete this project.

MATERIALS REQUIRED

To complete this project, you require:

Hardware

- One PC per student with an Internet connection and access to a printer
- Processor 600 MHz (1G MHz recommended)
- Windows 2000 (with SP 4), XP Professional (with SP 2) or Windows Server 2003 (with SP 1) operating system

- 192 MB RAM (256 MB RAM recommended)
- 2G hard disk space (minimum) for the operating system and applications
- DVD-ROM
- 800 x 600 (1024 x 768 recommended) SVGA monitor
- 1 blank diskette

Software

- Microsoft Office Professional Edition
- Microsoft Visual Studio 2005 Professional
- AVG Anti-virus (or similar antivirus software)
- Microsoft Internet Explorer version 5 or later

ASSIGNMENT

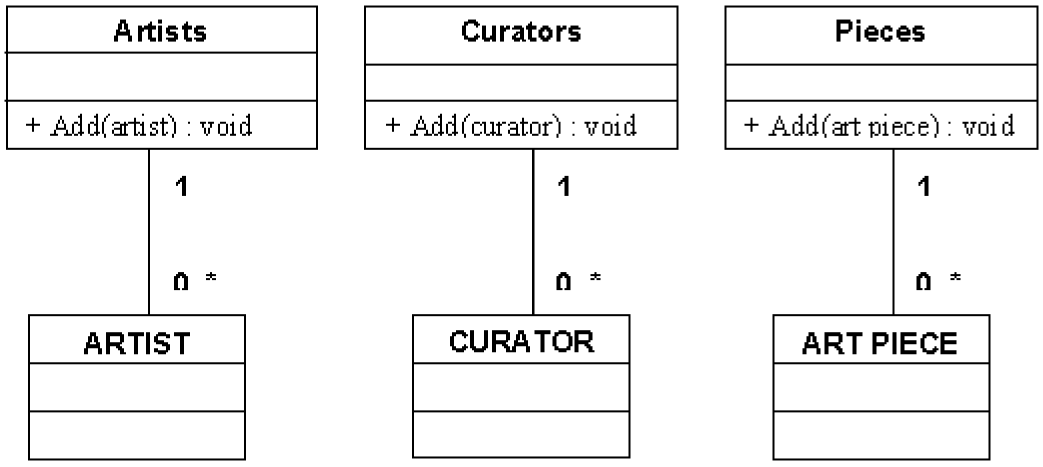
You are one member of the team assigned to the CGS project. Most of the systems analysis and design is done. For this project, you must supply the final application.

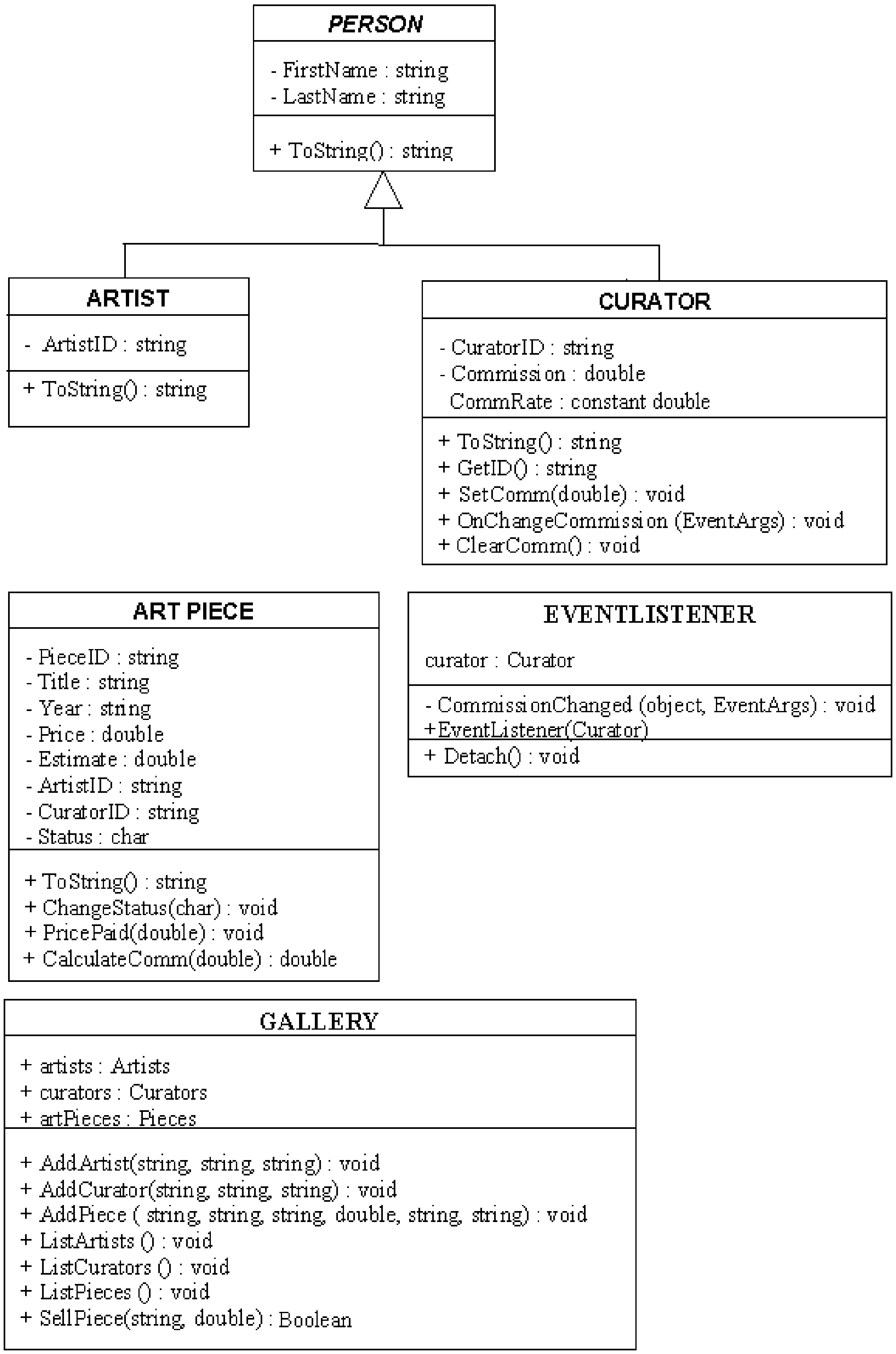
The company was impressed with the console prototype you created and has given you the opportunity to develop the graphical application prototype. This application will use the CGS class library you created plus include the ability to read curators from and save curators to a file.

You should review the CGS class library and the ArtGallery client you created in the previous project. In this project, you will replace the ArtGallery client with a Windows-based interface.

The CGS class library diagrams are included for reference:

Assume each parameter is preceded by the UML keyword ‘in’ (removed for clarity).




PERSON
- FirstName : string
- LastName : string
+ ToString() : string
ARTIST
- ArtistID : string
+ ToString() : string
CURATOR
- CuratorID : string
- Commission : double
CommRate : constant double
+ ToString() : string
+ GetID() : string
+ SetComm(double) : void
+ OnChangeCommission (EventArgs) : void
+ ClearComm() : void
ART PIECE
- PieceID : string
- Title : string
- Year : string
- Price : double
- Estimate : double
- ArtistID : string
- CuratorID : string
- Status : char
+ ToString() : string
+ ChangeStatus(char) : void
+ PricePaid(double) : void
+ CalculateComm(double) : double
EVENTLISTENER
curator : Curator
- CommissionChanged (object, EventArgs) : void
+EventListener(Curator)
+ Detach() : void
GALLERY
+ artists : Artists
+ curators : Curators
+ artPieces : Pieces
+ AddArtist(string, string, string) : void
+ AddCurator(string, string, string) : void
+ AddPiece (string, string, string, double, string, string) : void
+ ListArtists () : void
+ ListCurators () : void
+ ListPieces () : void
+ SellPiece(string, double) : Boolean

PROCEDURE

Sketch a flowchart or write pseudocode for all structured logic. You need to hand in your sketches. Make sure they are legible.

STEP 1:

First you will make some changes to the CGS class library. You will add a read and write method to the CGS class library. The file ‘curators.txt’ is supplied. Study it to determine its structure. You will also make a change to art piece to enhance its function.

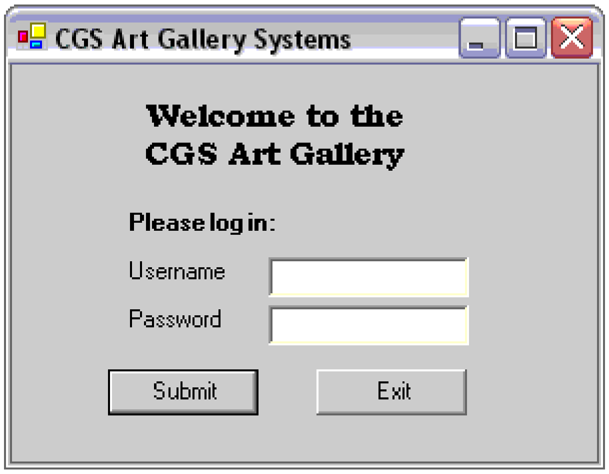
1. In Gallery, add a WriteCurators() method that returns a Boolean value indicating success or failure.
    a. Create a new file, or truncate and open an existing file.
    b. Iterate through the Curators collection to add each property for each curator to a string.
        i. Be sure to include the commas to separate the properties.
        ii. Add a new line character to the end of each curator. Do not add the new line character to the last curator.
2. In Curators, add a ReadCurators() method.
    a. The method should return a Boolean value indicating success or failure.
    b. The method will read the file contents into an array. From the array, the method adds each curator to the Curators collection.
    c. Assume the curators file will be located in the same folder as the project.
3. In Gallery, add a GetCurators() method that calls the ReadCurators() method. This method should return a Boolean value indicating success or failure.
4. In Art Piece, revise the ChangeStatus method to receive a character.
    a. You will need to revise the SellPiece method: pass the character S in the call to ChangeStatus.
    b. In Gallery, add a method SetStatus with a character parameter. If the art piece status is not sold, this method calls ChangeStatus to set the status to display or in storage. SetStatus should return a Boolean value to indicate success or failure.

STEP 2:

1. Create a Windows application named ArtGalleryWin. The form will welcome the users.
2. Add a second form named CGSArt. Add a reference to the CGS class library that you created in the previous project. Notice the second form does not

contain the static void Main() method. The application cannot start from this form.

3. Create the Welcome form to look similar to the following:



   a. In the first textbox, users will enter their username, and in the other, the password. The textbox for password should display * characters when the user enters the password (type the character in the textbox's Password property).

   b. Add two buttons. The first will check if both textboxes contain text; if not the system will display a message box to inform the user a username and password is required. If text exists in both, and if the username is 'CGS', and the password is 'admin', open the CGSArt form using the following code:

      ```
      CGSArt cgsArt = new CGSArt();
      cgsArt.Visible = True;
      cgsArt.Activate();
      ```

      If the username and password are incorrect, allow the user to try a second time. If on the third try, if the user does not enter the correct username and password, end the application using the following:

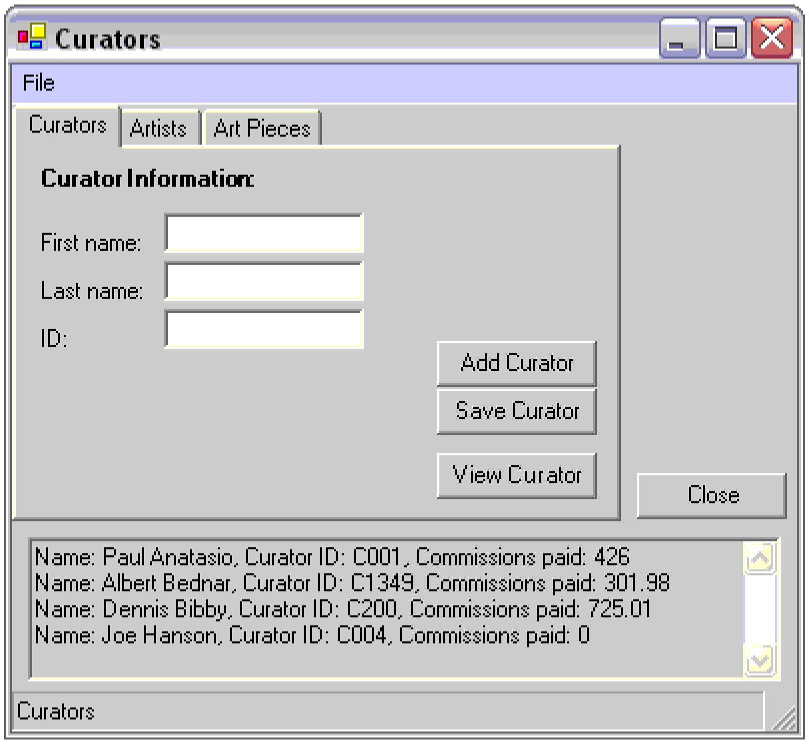
      ```
      Application.Exit();
      ```

   c. The second button should end the application.

4. Build, correct and run the application. Make any required changes.

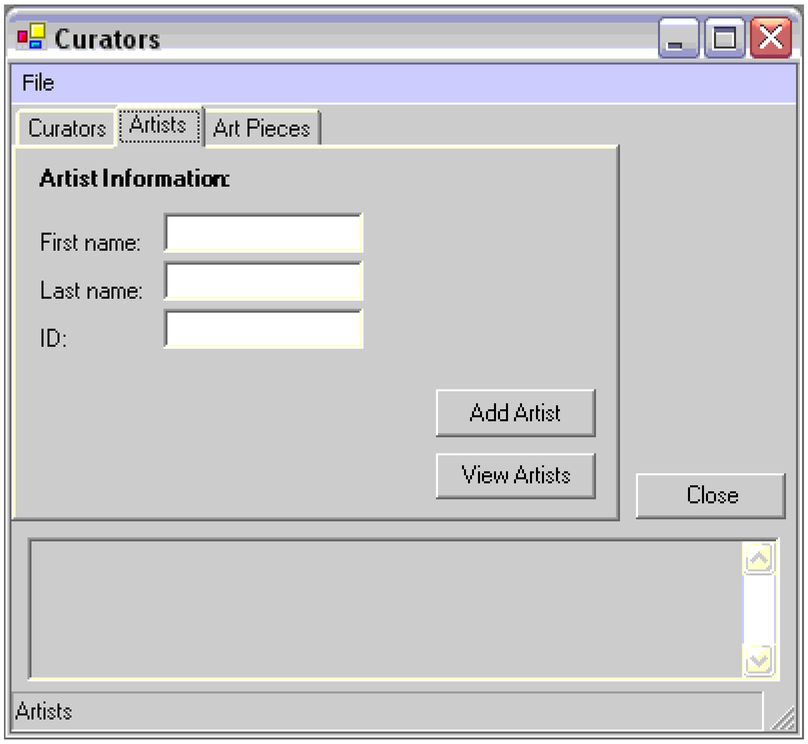
STEP 3:

1. The application allows the user to add curators, artists, and art pieces. It also allows the user to sell pieces. You will separate these functions by using tabs. When finished the cgsArt form might look something like the following (For now, ignore the textbox at the bottom of each tab. You will use a message box instead):

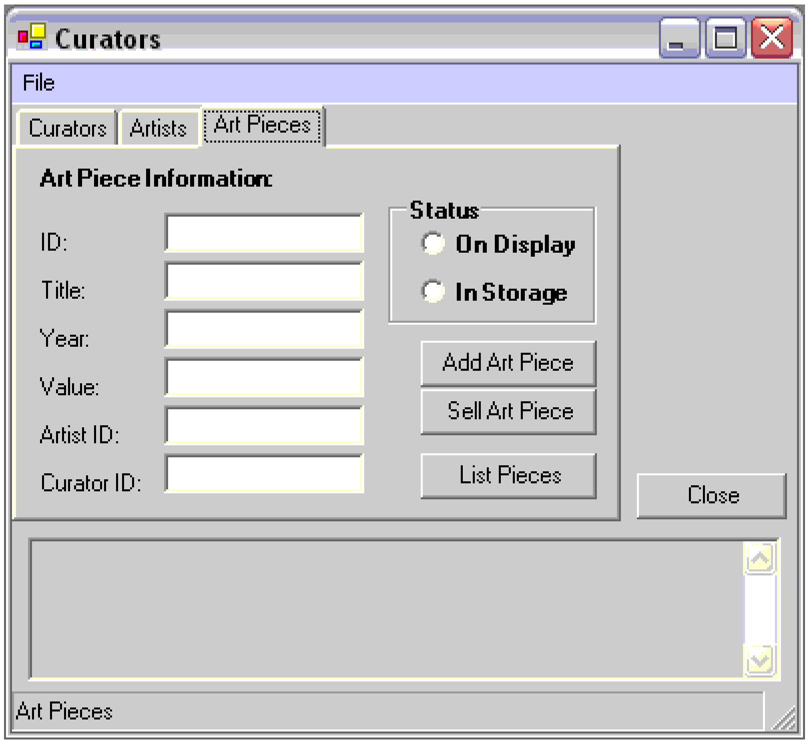


```
Name: Paul Anatasio, Curator ID: C001, Commissions paid: 426
Name: Albert Bednar, Curator ID: C1349, Commissions paid: 301.98
Name: Dennis Bibby, Curator ID: C200, Commissions paid: 725.01
Name: Joe Hanson, Curator ID: C004, Commissions paid: 0
```

Curator tab



Artist tab



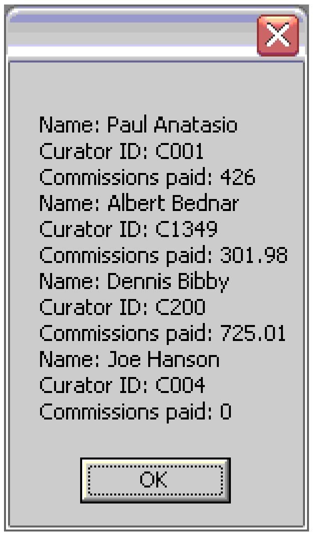
Art Piece tab

Enhancement with your instructor's permission: you can use additional forms for artists and art pieces. If you do, you will need to ensure that any variable you want to use on more than one form is available to other forms.

2. As you continue, make sure you periodically build, correct and run the application. Make adjustments where required.
3. Declare variables for:
   - curator's first name, last name, ID and commission.
   - artist's first name, last name and ID.
   - art piece' ID, title, year, value, artist, curator, price, status.
4. Add labels, textboxes and buttons for the user to input the information to:
   - Add an artist. Call Gallery's AddArtist.
   - Add a curator. Call Gallery's AddCurator.
   - Add an art piece. Call Gallery's AddPiece. The radio buttons allow the user to select if the piece is on display or in storage. In the button which calls AddPiece, add the method SetStatus and pass the value determined by the radio button. When testing, be sure to enter an existing curator ID or sell piece will not work.

- Sell an art piece. This button opens a small form. The form contains two text boxes, two identifying labels and an Okay button. The user enters the art piece ID in the first text box, and the sales price in the other text box. The user clicks the Okay button to call Gallery's SellPiece. If the SellPiece method returns true, a message box informs the user that the art piece was sold and the form closes. If it returns false, a message box informs the user that the art piece could not be sold and the form closes.

5. The application should display the artists, curators, and art pieces. You can call the ListArtists, ListCurators, and ListPieces methods in Gallery. Output each list to a message box, for example:



```
Name: Paul Anatasio
Curator ID: C001
Commissions paid: 426
Name: Albert Bednar
Curator ID: C1349
Commissions paid: 301.98
Name: Dennis Bibby
Curator ID: C200
Commissions paid: 725.01
Name: Joe Hanson
Curator ID: C004
Commissions paid: 0
```

STEP 5:

1. Add a menu with the options File, Open, Save and Exit.
   a. Under Open, call the CGS class method GetCurators() you created in Step 1. Copy the file curators.txt to the ArtGalleryWin folder before testing.
   b. Under Save, call the CGS class method WriteCurators() you created in Step 1.
   c. Under Exit, use the following code to return to the calling form:

   ```
   this.Close();
   ```

2. Add a status bar. The status bar should reflect whether the user is working with curators, artists or art pieces.

STEP 6:

1. Build, run and test the application. Make sure it runs without error.

   You may notice that when you sell an art piece the message indicating that the commission was paid to the curator is missing. It appeared in the console application. The problem is the event outputs a message to a console not a graphical application. You need to make a change to cause the message to appear in the Windows application.

2. In Windows Explorer, copy the Event Listener class from the CGS class library to the ArtGalleryWin folder.
3. In the ArtGalleryWin application, add the Event Listener class to the solution. Open the class and edit the message so that it outputs to a message box instead of the console.
4. Build the application. The result may indicate that two there are two Event Listener classes and that the application will use the one included with ArtGalleryWin. For this application's purpose, this is fine and there is no need to make a change.
5. Run and test the application. Be sure to enter an existing curator ID when adding an art piece.
6. Add appropriate error handling.

Congratulations!

You have created your first object-oriented, event-driven, graphical interface application that uses a custom class library.

If you have time

1. Use forms instead of tabs (described above).
2. Output lists to a textbox instead of message boxes (described above). The tabs in the cgsArt forms illustrated above include a textbox at the bottom. For the lists to display properly, you will need to edit the strings returned by the list methods before displaying the string in the textbox. Each returned string includes the new line escape character. You need to preface each new line character with the character return escape character to avoid a rectangular symbol displaying in place of the new line character.
3. Add a toolbar with buttons that allow the user to open and save the curators file, and sell an art piece. Use bitmaps available with Visual Studio .NET.

If you added some of the enhancements in the previous project, you can add to the ArtGalleryWin application's functionality. Make sure you keep a copy of your completed thus far before you attempt any additional work.

4. Add one or more of the following features to the application:

   Find Curator

   Add a button which prompts the user for a curator ID. If the curator is found, display the information in the textboxes.

   Delete Curator

   Add a button which deletes the curator displayed (if the curator exists).

   Find Artist

   Add a button which prompts the user for an artist ID. If the artist is found, display the information in the textboxes.

   Delete Artist

   Add a button which deletes the artist displayed (if the artist exists).