Multi-Tier Applications - LaSalle College

Project Category 1: Electronic Telethon Systems (ETS)

INTRODUCTION

This project is part one in a series of three, which you will complete in this semester. In the projects' scenario, you are one member of a software development team responsible for the design and development of a business solution for Electronic Telethon Systems (ETS).

The ETS tracks donations called in by telephone during a telethon. The system keeps track of each donation, each donor, and the allocation of prizes. Every donation is paid in full by credit card at the time it is taken. 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 C# 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 - Minimum Required

- One PC per student with an Internet connection.
- Processor 600 MHz (1G MHz recommended).
- Windows 10 or 11.

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

Software

- Microsoft Office Professional Edition
- Microsoft Visual Studio 2022 Community

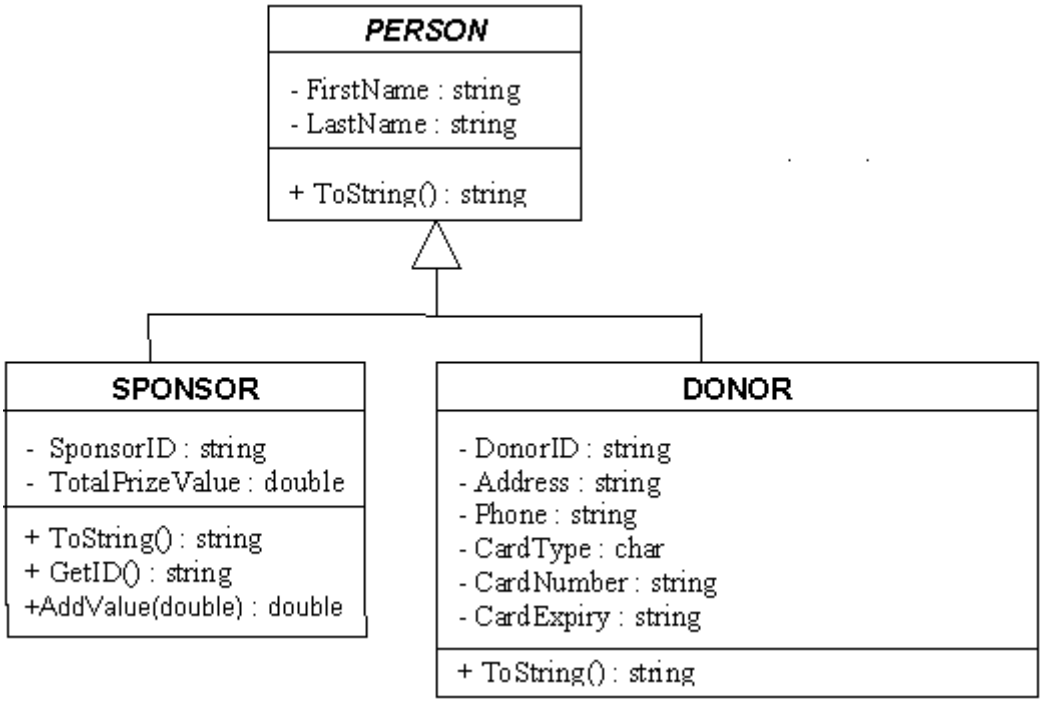
ASSIGNMENT

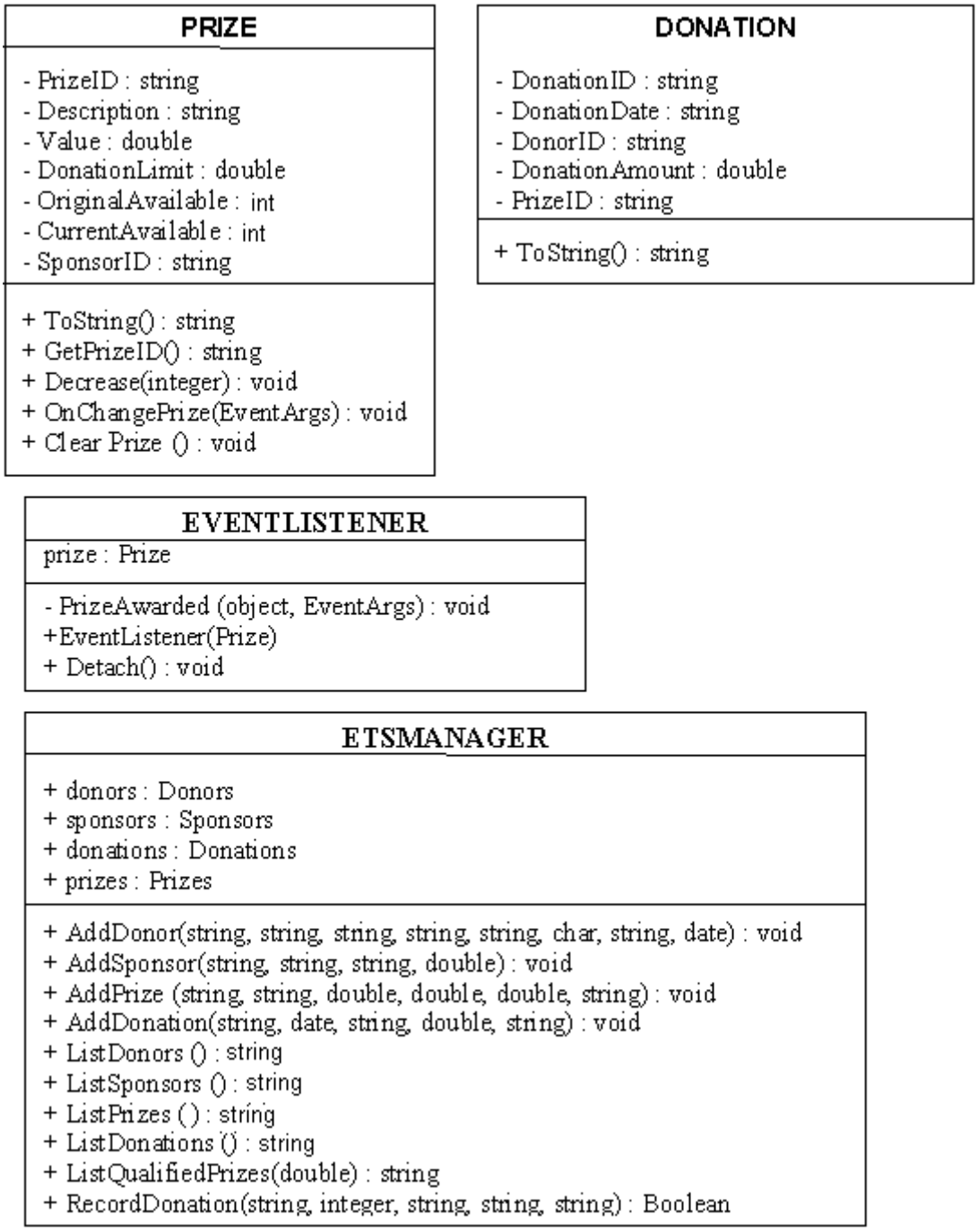
You are one member of the team assigned to the ETS project. Most of the systems analysis and design is done. For this project, you must supply the final application.

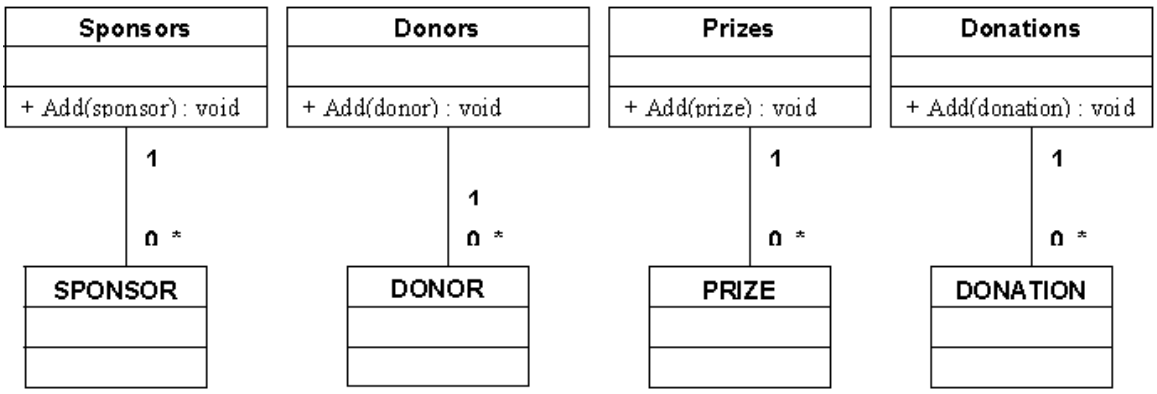
The company was impressed with the console prototype you created and has given you the opportunity to develop the graphical application prototype. This application will use the ETS class library you should create plus include the ability to read donors from and save donors to a file.

You should make the ETS class library and the TelethonSystem client. In this project, you should use a Windows-based interface.

The ETS class library diagrams are included for reference:







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 ETS class library. You will add a read and write method to the ETS class library. The file 'donors.txt' is supplied. Study it to determine its structure. You will also make a change to donors to enhance its function.

1. In ETSManager, add a WriteDonors() 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 Donors collection to add each property for each donor 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 donor. Do not add the new line character to the last donor.
2. In Donors, add a ReadDonors() 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 donor to the Donors collection.
    c. Assume the donors file will be located in the same folder as the project.
3. In ETSManager, add a GetDonors() method that calls the ReadDonors() method. This method should return a Boolean value indicating success or failure.

STEP 2:

1. Create a Windows application named TelethonSystemWin. The form will welcome the users.
2. Add a second form named ETSTelethon. Add a reference to the ETS 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 'ETS', and the password is 'admin', open the ETSTelethon form using the following code:

```
ETSTelethon ETSTelethon = new ETSTelethon ();
ETSTelethon.Visible = True;
ETSTelethon.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:

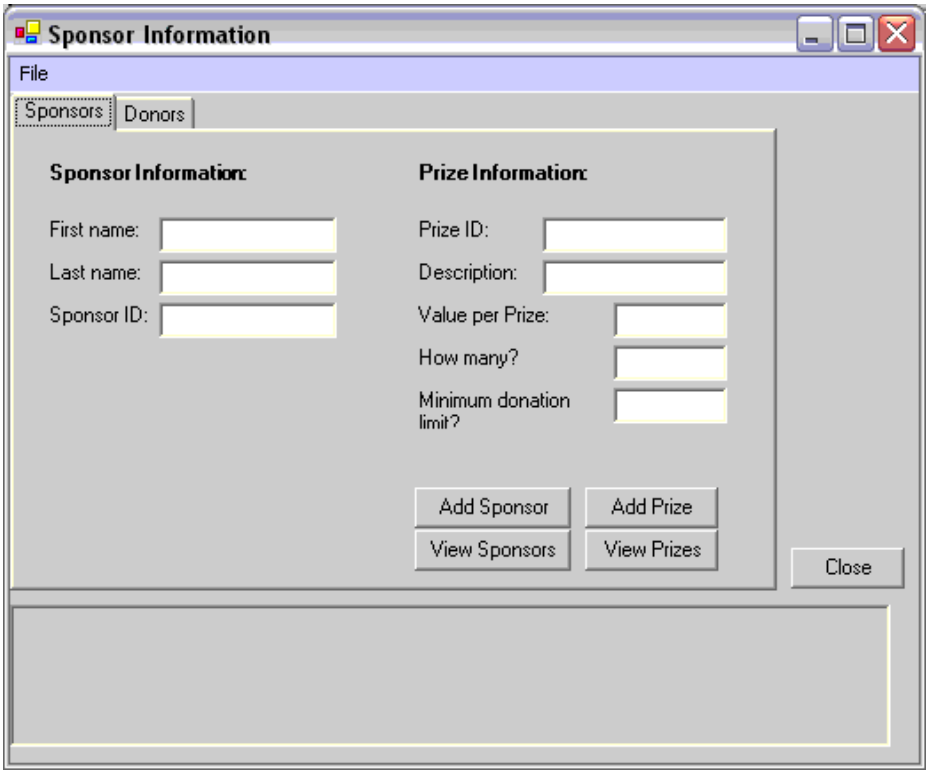
```
Application.Exit();
```

   c. The second button should end the application.

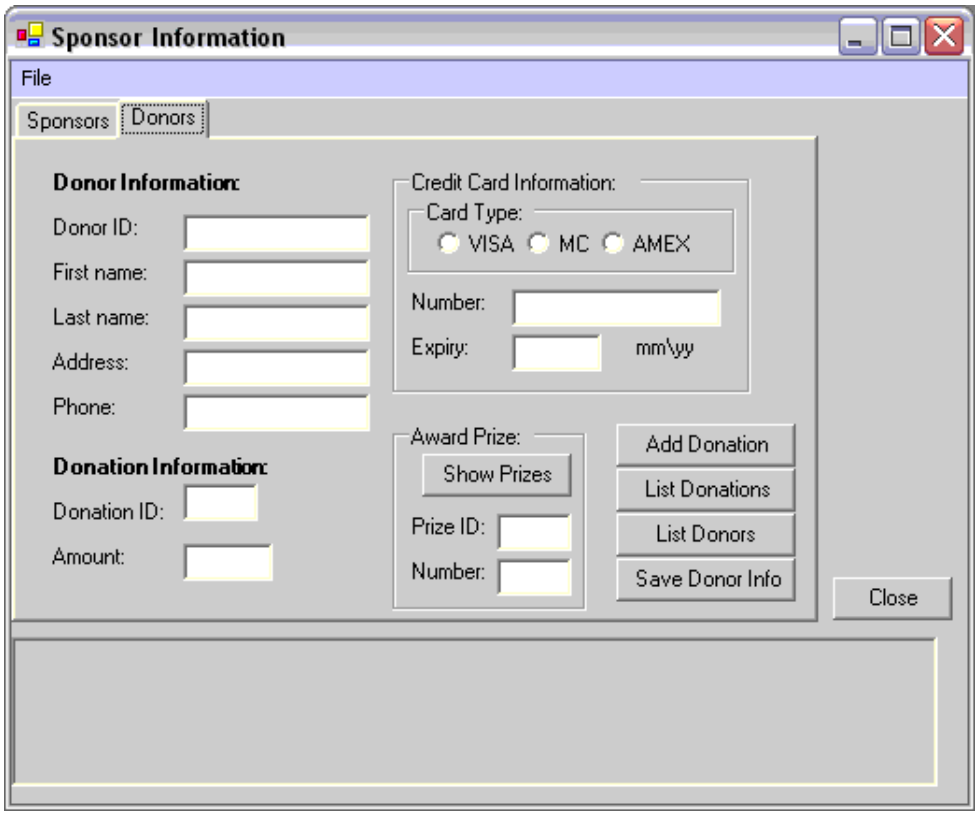
4. Build, correct and run the application. Make any required changes.

STEP 3:

1. The application allows the user to add sponsors and prizes, and donors and donations. You will separate these functions by using tabs. When finished the ETSTelethon form might look something like the following (For now, ignore the textbox at the bottom of each tab. The textbox would hold the lists. You will use a message box instead):



Sponsors tab



Donors tab

Enhancement with Mohammad's permission: you can use additional forms for prizes and donations. 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:
    - sponsor's first name, last name, ID and total prize value.
    - donors's first name, last name, ID, address, phone, card type, card number and card expiry.
    - prizes's ID, description, value, donation limit, original number available, current number available and sponsor ID.
    - donation's ID, date, donor ID, amount and ID of prize awarded.
    - You will also need to add local variables where required.
4. Add labels, textboxes and buttons for the user to input the information to:
    - Add a sponsor. Call ETSManager's AddSponsor.
    - Add a prize. Call ETSManager's AddPrize.

- Add a donor. The radio buttons allow the user to select the credit card type.
- Show Prizes calls ListQualifiedPrizes. The user enters the prize ID and number of prizes to award in the accompanying textboxes.
- Add a donation. Call ETSManager's RecordDonation. All information submitted in the tab is used to add a donation, award a prize (if eligible), and add a donor.

5. The application should display the sponsors, prizes, donors and donations. You can call the ListSponsors, ListPrizes, ListDonors and ListDonations methods in ETSManager. Output each list to a message box, for example:

STEP 5:

1. Add a menu with the options File, Open, Save and Exit.
    a. Under Open, call the ETS class method GetDonors() you created in Step 1. Copy the file donors.txt to the TelethonSystemWin folder before testing.
    b. Under Save, call the ETS class method WriteDonors() 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 sponsor and prizes, or donors and donations.

STEP 6:

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

    You may notice that when ETS awards a prize the message indicating that the prize was awarded 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 ETS class library to the TelethonSystemWin folder.
3. In the TelethonSystemWin 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. Be sure to add the using statement for System.Windows.Form.
4. Build the application. The result may indicate that there are two Event Listener classes and that the application will use the one included with TelethonSystemWin. For this application's purpose, this is fine and there is no need to make a change.
5. Run and test the application.
6. Add appropriate error handling.

Best of Luck!

Bonus Section (Optional)

1. Use forms instead of tabs (described above).
2. Output lists to a textbox instead of message boxes (described above). The tabs in the ETSTelethon form illustrated above includes 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 donors file, and show prizes. Use bitmaps available with Visual Studio .NET.

If you added some of the enhancements in the previous project, you can add to the TelethonSystemWin 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:

   Display Prizes in a Drop-Down List Box

   Instead of using a button and two textboxes to display and receive prize ID selection, display prizes in a drop-down list box. Users would then click the desired prize. Modify ETSManager to always award only 1 prize per donation and thereby do away with both textboxes entirely.

   Display a Sponsor

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

   Allow a sponsor to provide multiple prizes. You need to use the AddValue method described in the previous project to increase the sponsor's total value of prizes provided.

   Display a Donor

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

   To allow a donor to make multiple donations, edit RecordDonation (if you did not do so in the previous project). Move the call to AddDonor outside of RecordDonation so that it may be called independently if RecordDonation is successful.

   Delete Donor

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