|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CS1400 Lab #11**  **Implementing the Token Machine Class**  **Version 1.0**  **Introduction**  A complete Graphical User Interface (GUI) program normally consists of the following objects:   * An Singleton Application object * A Form object * One or more Control objects, and * One or more Domain objects   In this lab you will learn how to add a ***Domain class*** to a Windows Forms Application. Remember that the code in the Form class is responsible for managing the user interface. All of the business logic in your program will reside in the ***Domain class*** you add to your program. Take careful notes on how you write this code as any programs you write from now on, that uses a graphical user interface will follow this pattern.    Here is a design for a token dispensing machine class. You designed a similar class in your last lab. This class only has two data members; the number of tokens it contains and the number of quarters it contains. There are five operations you can do on this machine:  Figure 1 UML Diagram for TokenMachine class  ***TokenMachine( )*:**The constructor that is used to initialize the data members of a TokenMachine object.  ***GetToken( )*:** You get a token by putting a quarter in the machine. The number of tokens in the machine is reduced by one, and the number of quarters in the machine is increased by one. One token is returned by the method.  ***CountTokens( )*:** Returns the number of tokens that are in the machine.  ***CountQuarters( )*:** Returns the number of quarters that are in the machine.  ***Reset( )*:** The reset operation removes (returns) all of the quarters and fills the machine with tokens. For this class we will assume that the machine holds 100 tokens.  **Designing the GUI**  Start Visual Studio and create a new Windows Form Application. Name your Project Lab\_11 or the TokenMachine. Change the name of your Form to “FrmTokenMachine.” Lay out a user interface for your Project. It need not look like the example shown below, but it should have the same functionality and be intuitive to use.    **Writing the Code**  In this project we are going to create a ***Domain class*** named ***TokenMachine***, that represents the actual machine. Now connect it to the graphical user interface by instantiating an object with the new operator. To do this, carefully follow these steps:   * On the Visual Studio ***MenuBar***, click on ***Project->Add Class***. You should see a ***DialogBox*** like this:   http://debryro.tc.uvu.edu/1400/labs/lab11/addClass.png   * Click on Class (the top item in the List) * Name your class ***TokenMachine*** and click on the ***Add Button*** in the bottom right corner of the dialog box. A new class file will be added to your project. * Using the ***UML Class Diagram*** as your guide, fill in the code for the ***TokenMachine*** class. * Now look at the code for the Form. This file should be named Form1.cs. * Just inside the FrmTokenMachine class declaration add a line of code to create a reference to a ***TokenMachine*** object. The code will look like this:   public partial class FrmTokenMachine : Form {    // a class level reference to a token machine    private TokenMachine tm;   * 1. In the ***FrmTokenMachine\_Load*** event handler, create a ***new TokenMachine*** object and call its ***Reset*** method and display the number of quarters and tokens in the ***TextBoxes***. The code will look like this:   public void FrmTokenMachine\_Load(…) {    // create a token machine object    ***tm = new TokenMachine();***   * Add event handlers for the ***Exit*** ***Menu*** item and the ***About*** ***Menu*** item. * Add an event handler for the ***Get Token Button***. This Button will add one to the number of quarters in the machine and subtract one from the number of tokens. Update the ***TextBoxes*** to show the current values. * Add an event handler for the ***Reset*** Button. The ***Reset*** Button will call the Reset method in your token machine class and update the ***TextBoxes*** to show the current values. * **Sample Executable** * A sample executable file can be found on Canvas. Your program should work in a similar manner. * **File(s) to Submit:** * Place your complete Project folder in a zip file and name the zip file Lab\_11\_your-initials\_V1.0.zip. For example, I would name my file Lab\_11\_DAF\_V1.0.zip. Submit this assignment as Lab #11 on Canvas.   **Grading Checklist**   |  |  |  |  | | --- | --- | --- | --- | | # | Program | 1st Submission | 2nd Submission | | 1 | Meets & works to specifications | Correct | Xnot | 6 points | | 2 | Error Free, elegant & efficient | C | X | 4 points | | 3 | Pseudo-Code | C | X | -3 points | | 4 | Style Guidelines | C | X | -2 points | | 6 | Source Files(s) & Formatting | C | X | -2 points | | 7 | Project Prolog | C | X | -1 points | | 8 | Function Prologs | C | X | -1 points | | 9 | Zip Filename | C | X | -1 points | | 10 | Lab & Project Names | C | X | -1 points | | 11 | Zip File is invalid or will not unzip | Lab = 0 pts | Lab = 0 pts | |  | Total Points | 10 | 0 | 10 | 0-9 | |