CS1400 Lab #13: Implementing the Token Machine Class

Introduction

Many of the programs that you will create from now on will contain two objects that work together to complete the work of the application:

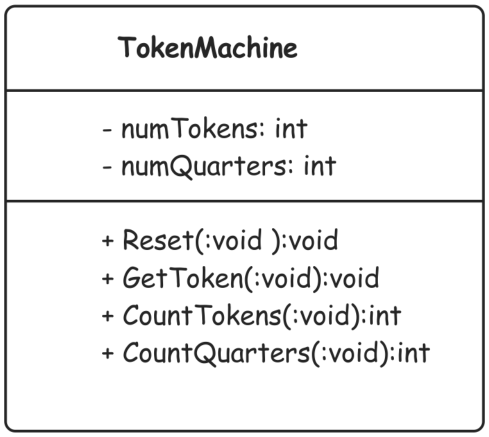
1. An Application object
2. A Form object

It is important to note that each object has an important role to play, and the roles must not be confused. The Application object declares the application's data and provides methods to work with that data. Application objects know nothing about the user interface. On the other hand, the Form object contains the GUI components (Buttons, TextBoxes, MenuItems, etc., and event handlers. It knows nothing about the application's data or how the application does its work. The responsibility of the Form is to show the user the graphical user interface and respond to events that are generated at the interface.

Before you proceeed, carefully study the code in the example provided here to make sure that you understand this important relationship.

Adding an Application Class

In this lab you will learn how to add an application class to a WindowsForms Application. Remember that the code in the Form class is responsible for managing the user interface. All of the application logic in your program will reside in the application class you add to your program. Sometimes we call this class a domain class. Take careful notes on how you write this code as any programs you write from now on that use 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. These are represented by the following methods:

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.

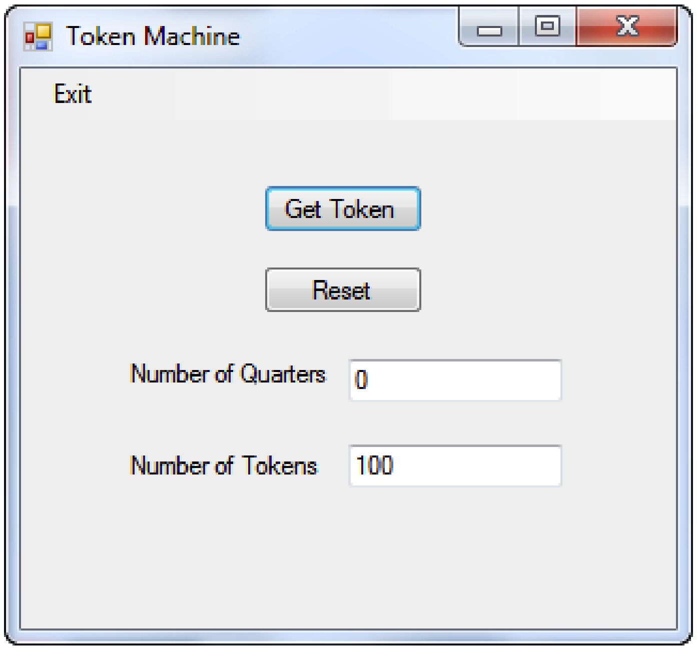
CountTokens( ): Returns the number of tokens that are in the machine.

CountQuarters( ): Returns the number of quarters that are in the machine.

Reset( ): The reset operation removes 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

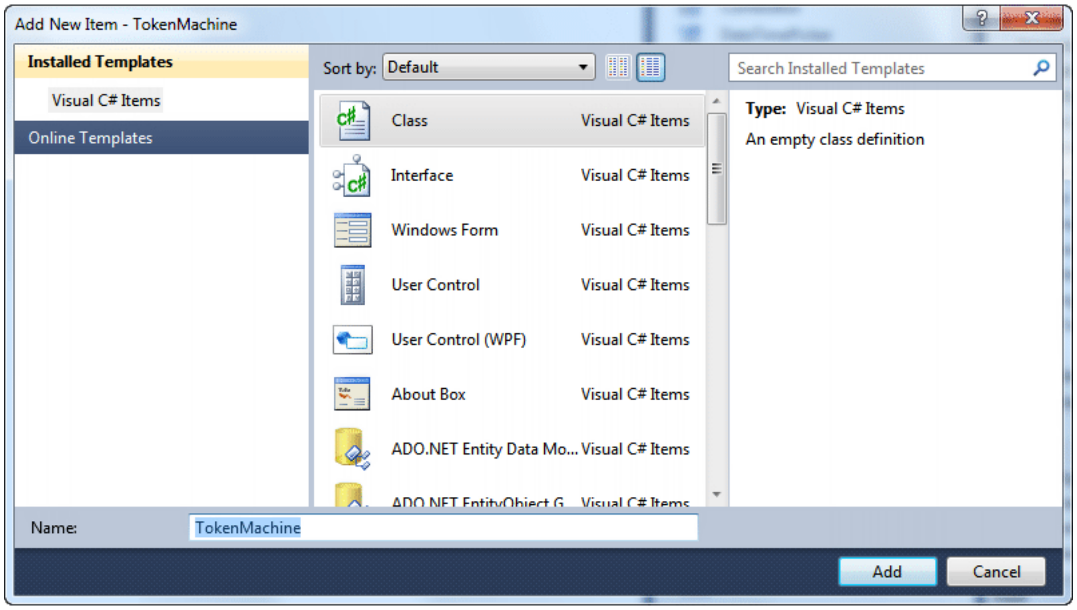
Start Visual Studio and create a new Windows Form Application. Name your project TokenMachine. Lay out a user interface for your project. It need not look like the example shown below, but it should have the same functionality and by intuitive to use.



Writing the Code

In this project we are going to create a domain class that represents a TokenMachine and connect it to the graphical user interface. To do this, carefully follow these steps:

1. On the Visual Studio MenuBar, click on Project->Add Class. You should see a DialogBox like this:


Add New Item - TokenMachine
Installed Templates
Visual C# Items
Online Templates
Sort by: Default
Search Installed Templates
Class Visual C# Items
Interface Visual C# Items
Windows Form Visual C# Items
User Control Visual C# Items
User Control (WPF) Visual C# Items
About Box Visual C# Items
ADO.NET Entity Data Mo... Visual C# Items
ADO.NET EntityObject G... Visual C# Items
Type: Visual C# Items
An empty class definition
Name: TokenMachine
Add
Cancel

2. Click on Class (the top item in the List)
3. Name your class TokenMachine.cs and click on the Add Button in the bottom right corner of the dialog box.
4. A new class file will be added to your project.
5. Using the UML Class Diagram as your guide, fill in the code for the Token Machine class.
6. Now look at the code for the Form. This file should be named Form1.cs.
   a. Just inside the Form1 class declaration add a line of code to create a reference to a TokenMachine object. The code will look like this:

   ```
   public partial class Form1 : Form
   {
       // a class level reference to a token machine
       private TokenMachine tm;
   ```

   b. In the constructor for the Form, create a new Token machine object and call its Reset method. The code will look like this:

   ```
   public Form1()
   {
       InitializeComponent();

       // create a token machine object
       tm = new TokenMachine();
       tm.Reset();
   ```

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 call the GetToken method in your token machine class and then update the TextBoxes to show the new 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 here. 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_13_your-initials_V1.0.zip. For example, I would name my file lab_13_RKD_V1.0.zip. Submit this assignment as Lab #13 on Canvas.

Grading Guidelines

Description	**Points possible**
Assignment meets grading guidelines: o Class diagram contains a declaration that this is your own work. o Assignment has been properly submitted to Canvas	2
Your program works like the sample program provided.	3
Total	5