CSC 342 Project 3

Loan Calculator - Windows Form Application

The purpose of this project is to demonstrate the process of numeric I/O through text boxes and to practice the design and implementation of a simple Windows Form application.

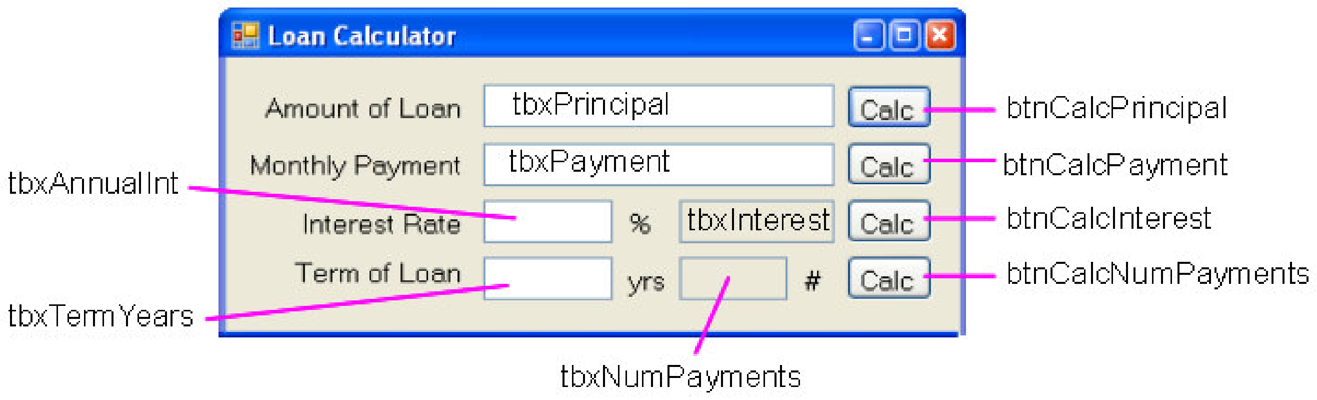
Project Overview: This project builds on the Loan Amortization project by giving the user the option of computing any one of the parameters having provided values for the others. The original form of the expression is given by,

$$M = P \frac{[\,I(1+I)^N\,]}{[\,(1+I)^N - 1\,]}$$

M = Monthly Payment
P = Mortgage Principal
I = Monthly Interest
N = Number of Months

The expression for computing the principal P given a monthly payment amount M, interest I and number of payments N can be obtained algebraically. To calculate the Interest or number of payments given values for the other parameters may require iteration.

Form Design: Typically the first step in the development of a Windows application is the graphical design and layout of the form. For this project we will use a form similar to the one shown below. Variation on this theme is encouraged so long as your design provides an equivalent or greater functionality. This form shows the names of the text boxes and buttons used in this guide. Of course, you may choose any names you like, just make them relate to their purpose.



Note that *tbxInterest* and *tbxNumPayments* should be "ReadOnly" which is a property that can be set in the Properties panel. We will permit the user to enter only annual interest rate (as a percentage) and the term of the loan (in years).

As you create the Calc buttons be sure to give them names that make their intended purpose clear. Also, you might want to go ahead and create button-click event methods for each of them by double-clicking the button on the designer form. Be sure you have re-named each button before creating its button-click event.

Data Entry: Now that we have text boxes to accept inputs for our parameters we need to define variables to hold the actual numeric values. These variable can be made to be accessible from all parts of the project (global) by declaring them at the top of the Form class.

```
    public partial class Form1 : Form
    {
        public static double principal;
        public static double monthlypayment;
        public static double termyears;
        public static double numpayments;
        public static double annualinterest;
        public static double monthlyinterest;
```

The first action that needs to be taken when a *Calc* button is pressed is that the numbers in the text boxes need to be captured, converted into numeric values and stored in their respective numeric variables. If we assume that the user has appropriately entered all the necessary data, our program would only need to perform conversions such as the following for the needed parameters:

```
        principal = Convert.ToDouble(tbxPrincipal.Text);
```

or

```
        annualinterest = Convert.ToDouble(tbxAnnualInt.Text);
        monthlyinterest = annualinterest / 1200.0;
```

At this point in the design, you have some decisions to make. You could put data entry code for the needed variables at the top of each Calc button-click event method or you could create one data-entry method that uses try-catch blocks to gather all the available data, such as,

```
        public void getdata()
        {
            try
            {
                principal = Convert.ToDouble(tbxPrincipal.Text);
            }
            catch
            {
                tbxPrincipal.Text = "";
                principal = -1.0;
            }
```

and call getdata( ); at the top of each Calc button-click event method. Since the user may not have entered a valid number in the text box corresponding to the value to be calculated the non-numeric entry would be dealt-with in the catch portion of the try-catch block (i.e. the Convert.ToDouble( ) method will fail on a non-numeric text string).

In this example code segment, the catch block sets the principal to -1.0 which can be used as a *sentinel value* to alert the program that this value is not available for processing. If the principal is the value being computed then the -1.0 will be ignored and replaced with the computed value. If principal is a value needed for computation but was entered incorrectly then the -1.0 can be used in a test to see if computation may proceed. Note that use of a *sentinel value* is an option in this project.

Processing: The user will enter values into three of the four input text boxes and then click the *Calc* button to find the remaining value. The button-click event methods for computing monthly payment and for principal (loan amount) given a monthly payment and the other parameters must be completed in order to receive full credit for this project. The computation of interest rate or term of loan are optional. For now you may leave these two button-click event methods unimplemented.

The following code is a sample of the button-click method for computing the monthly payment. In this example the method getdata( ) is assumed to read all the text boxes and convert the entries to numeric values if possible. If the value of a parameter cannot be interpreted from its corresponding text box a value of -1.0 is used instead. This allows us to check for the values we need to compute the monthly payment before performing the calculation.

```
private void btnCalcPayment_Click(object sender, EventArgs e)
{
  double onePlusI2N;
  getdata();
  if (principal > 0.0 & monthlyinterest > 0.0 & numpayments > 0.0)
  {
    onePlusI2N = Math.Pow((1.0+monthlyinterest),numpayments);
    monthlypayment = principal*(monthlyinterest*onePlusI2N/(onePlusI2N - 1.0));
    tbxPayment.Text = Convert.ToString(Math.Round(monthlypayment,2));
  }

}
```

A similar method can be built for computing the principal (loan amount) that can be borrowed given a specified monthly payment, interest rate and term of loan. Build this method and include it as part of your project submission.

Optional Work: Computing the interest rate for a given monthly payment, principal and term of loan is a bit more involved, since you cannot solve for this value algebraically. Computing the term of the loan given the other parameters creates similar issues. These values are best computed using some form of iteration such as the bisection method. Have the program choose a starting value and the increase or decrease it as necessary to force one of the other parameters close to its specified values. For example you could use the form of the expression for computation of the monthly payment given above.

If you choose to do this optional work it is recommended that you use a while loop and set some small value greater than zero as a threshold for terminating the iteration. For example, if you wanted to compute the annual interest rate given the other parameters you could end the iteration when the interest rate used resulted in a difference between the specified monthly payment and the computed monthly payment less than 0.01. That is, you compute the value to the nearest penny.