Building Your Own Classes and Objects

In this paper you will learn to create and use your own classes (sometimes known as a programmer defined classes or type). Creating your own classes is the key part of object oriented programming (OOP). As in procedures, classes can be reused. These classes contain properties, methods, and events which are abstracts that gather all the data and behavior into an enclosed package with well-defined interfaces for the outside code to use. Those interfaces determine exactly how code outside of the class can interact with the class. A class determines which data values are visible to the outside and which are hidden. It also determines the routines (methods) that the class supports and their availability (visible or hidden).

KEY TERMS:

- Accessors: A method like code units that handle the details of modifying and returning data.
- Class: A definition of a complete object, which may include one or more interfaces. This is the place where you declare INSTANCE variables, and write your code for the enclosed METHOD and PROPERTY procedures. While some object implementations allow only a single interface per class, PowerBASIC objects (and COM objects in general) support the idea of optional multiple interfaces. Still, remember that a CLASS is the complete definition of an object. It defines all of the code and all of the data which will be found in a particular object. For this reason, there is only one copy of a CLASS.
- Method: A subroutine, very similar to a user-defined Sub/Function. A method has the special attribute that it can access the variables stored in the object. A method can return a value like a Function, or return nothing, like a Sub
- Property: This is a METHOD, but in a specific form, for a specific purpose. A PROPERTY has all the attributes of a standard METHOD. It has a special syntax, and is specifically used to read or write private data to/from the internal variables in an object. This helps to maintain the principle of "encapsulation". Properties are usually created in pairs, a GET PROPERTY to read a variable, and a SET PROPERTY to write to a variable. Paired properties use the same name for both, since PowerBASIC will choose the correct one based upon the usage in your source code.
- Get/End Get Keywords: Reserved words that define a property's GET accessor.
- Instance Data: Each CLASS defines some INSTANCE variables which are present in every object. When you create multiple objects (of the same class), each object gets its own unique copy of them. These variables are called INSTANCE variables because a new set of them is created for each instance of the object.
- Interface: A definition of a set of methods and properties which are implemented on an object. You might think of it as a list of DECLARE statements where the sequence of the Declares must be preserved. Remember, the interface is just the definition, not the actual code. Every interface is associated with a GUID (a 128-bit number or string) which uniquely identifies this particular interface from all other interfaces, anywhere in the world.
- Set/End Set keywords: Reserved words that define a property's SET accessor.

A Microwave Oven Application:

You will create a microwave oven simulator where the user will enter an amount of time for the microwave to cook food. To handle the time data, you will create a class called cTime. This class stores the minutes and seconds (which your **Microwave Oven** application will use to keep track of the remaining cook time) and provides properties so clients of this class can change the number of minutes and seconds.

Application Requirements

A junior high school wants to train students to use microwave ovens in their home economics class. The school has asked you to develop an application that simulates a microwave oven. The oven contains a keypad that allows the student to specify the microwave cook time, which is displayed for the student. Once a time is entered, the student clicks the Start button to begin the cooking process. The microwave's glass window changes color (from gray to yellow) to simulate the oven's light that remains on while the food is cooking, and a timer counts down one second at a time. Once the time expires, the color of the microwave's glass window returns to gray (indicating that the microwave's light is now off) and the microwave displays the text "Done!" The student can click the Clear button at any time to stop the microwave and enter a new time. The student should be able to enter a number of minutes no larger than 59 and a number of seconds no larger than 59,' otherwise, the invalid portion of the cook time is set to zero. A beep is sounded whenever a Button is clicked and when the microwave oven has finished a countdown.

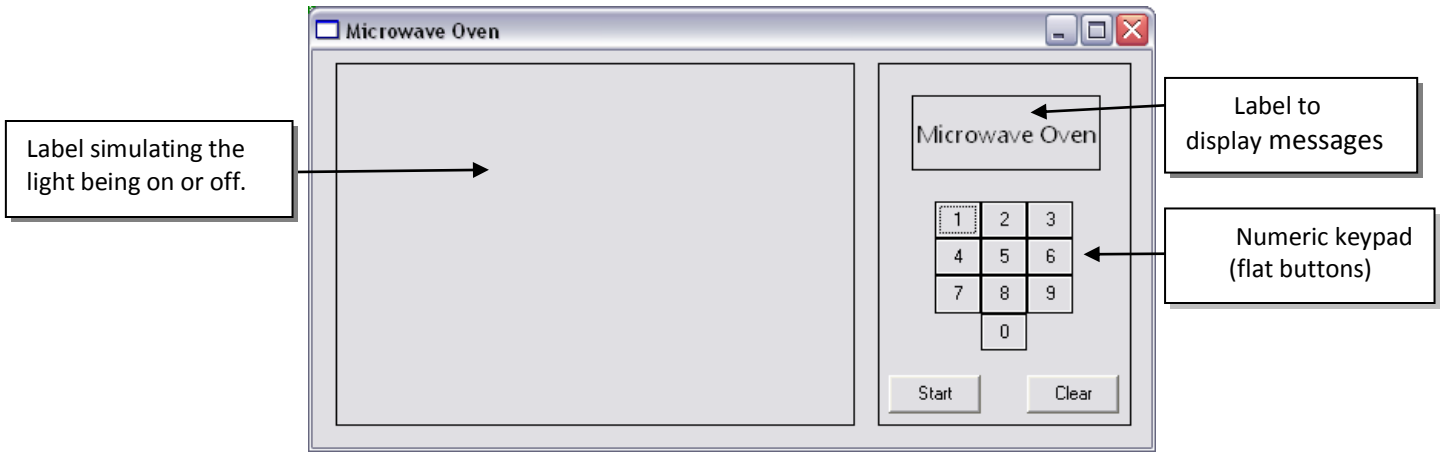

Figure 1 Microwave Oven Application at startup

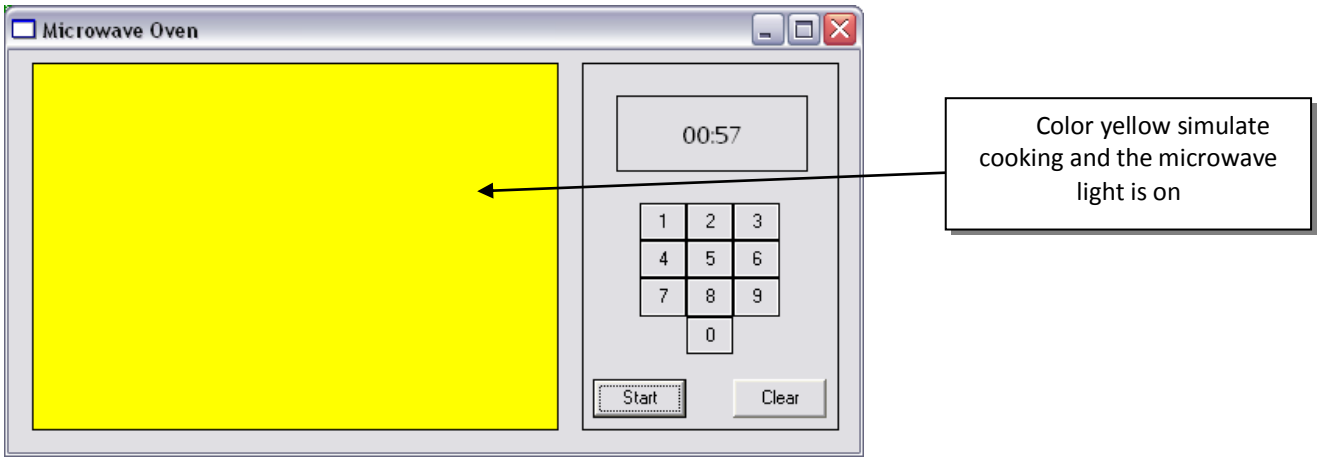

Figure 2 Microwave Oven Application with the inside light turned on(simulated cooking)

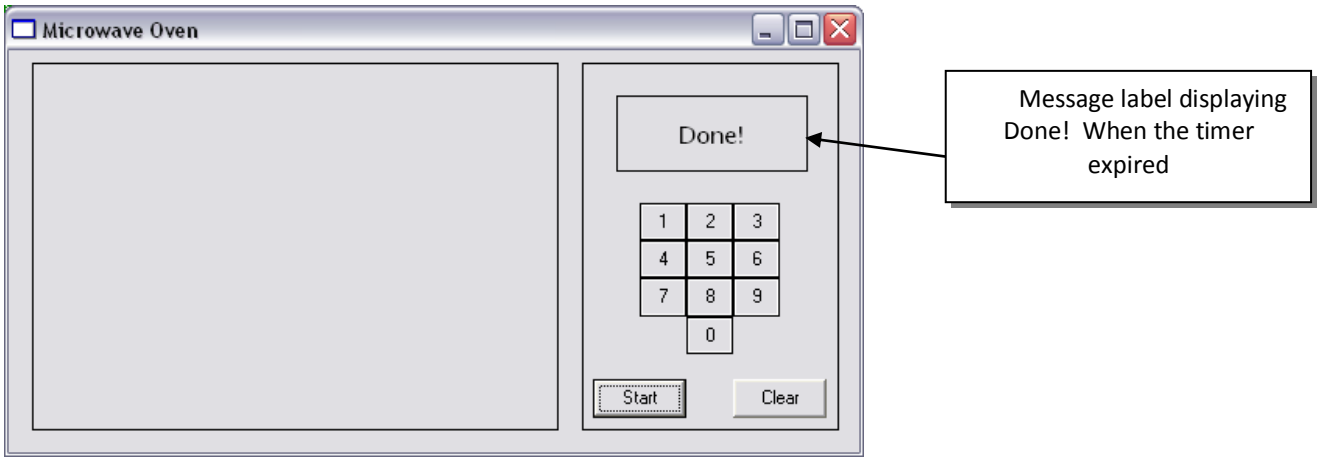

Figure 3 Microwave Oven Application after the cooking time has expired.

When the time has expired the cooking light returns to the default color and the message label shows the word "Done!" The application is then closed by clicking the close box (X on the top right of the form).

The coding

The Microwave Oven Application contains a class (called cTime) whose objects store the cook time in minutes and seconds. You will start by creating pseudo code for both the main application and the cTime class. The following pseudo code describes the basic operation of class cTime:

```
When the time object is created:
    Assign input to variables for number of minutes and number of seconds

When setting the number of minutes:
    If the number of minutes is less than 60
        Set the number of minutes to specified value
    Else
        Set the number of minutes to 0

When setting the number of seconds:
    If the number of seconds is less than 60
        Set the number of seconds to specified value
    Else
        Set the number of seconds to O
```

When an object of class cTime is created, the number of minutes and number of seconds are initialized. Any invalid number (minutes or seconds) is set to 0. The following pseudo code describes the basic operation of your Microwave Oven program:

```
When the user clicks a numeric Button:
        Sound beep
        Display the formatted time

When the user clicks the Start Button:
        Store the minutes and seconds
        Display the formatted time
        Begin countdown—Start timer
        Turn the microwave light on

When the timer ticks (once per second):
        Decrease time by one second
        Display new time
        If new time is zero
        Stop the countdown
        Sound beep
        Display text "Done!"
        Turn the microwave light off
        Delete the object

When the user clicks the Clear Button:
        Display the text "Microwave Oven"
        Clear input, time data and delete the object
        Stop the countdown
        Turn the microwave light off
```

The user enters input by clicking the numeric Buttons. Each time a numeric Button is clicked, the number on that Button is appended to the end of the cook time displayed in the GUI's Label. At most, four digits can be displayed. After entering the cook time, the user can click the Start Button to begin the cooking process or click the Clear Button and enter a new time. Each Button makes a beeping sound when clicked. If the Start Button is clicked, a countdown using a Windows Timer begins, and the microwave oven's window changes to yellow, indicating that the oven's light is on (so that the user can watch the food cook). Each second, the display is updated to show the remaining cooking time. When the countdown finishes, another beep is sounded, the display label displays the text 'Done!'

Creating the cTime class: Following the pseudo code construct a class called cTime and save it to a file called cTime.inc. Your code should look like the one in Fig. 4. The class has two properties, each with two accessors which handle the details of modifying and returning data. The class also has a method that is a constructor that initializes the properties for first use. The class keeps two variables hidden by using the INSTANCE statement. This one class has just one Interface called iTime.

```
CLASS cTime

    ' declare Instance variables for minute and second
    INSTANCE m_MinuteValue AS INTEGER
    INSTANCE m_secondValue AS INTEGER

    'definition of a set of methods and properties which are implemented on an object
    INTERFACE Itime

        ' offers both direct access and Dispatch access to the interface methods
        INHERIT IDISPATCH

        ' Time constructor, minute and second supplied
        METHOD NEW(BYVAL mm AS INTEGER, BYVAL ss AS INTEGER)
            ME.Minute = mm ' invokes Minute accessor
            ME.Second = ss ' invokes Second accessor
        END METHOD ' NEW

        ' property Minute
        PROPERTY GET Minute() AS INTEGER
            PROPERTY = m_MinuteValue
        END PROPERTY ' end of Get accessor

        PROPERTY SET Minute(BYVAL value AS INTEGER)
            IF (value < 60) THEN 'if minute value entered is valid
                m_MinuteValue = value
            ELSE
                m_MinuteValue = 0 ' set invalid input to 0
            END IF
        END PROPERTY ' end of Set accessor

        ' property Second
        PROPERTY GET Second() AS INTEGER
            PROPERTY = m_secondValue
        END PROPERTY ' end of Get accessor

        PROPERTY SET Second(BYVAL value AS INTEGER)
            IF (value < 60) THEN 'if second value entered is valid
                m_secondValue = value
            ELSE
                m_secondValue = 0 ' set invalid input to 0
            END IF
        END PROPERTY ' end of Set accessor
    END INTERFACE ' Itime
END CLASS ' cTime
```

Figure 4 cTime Class

Note: buttons are all numbered concurrently

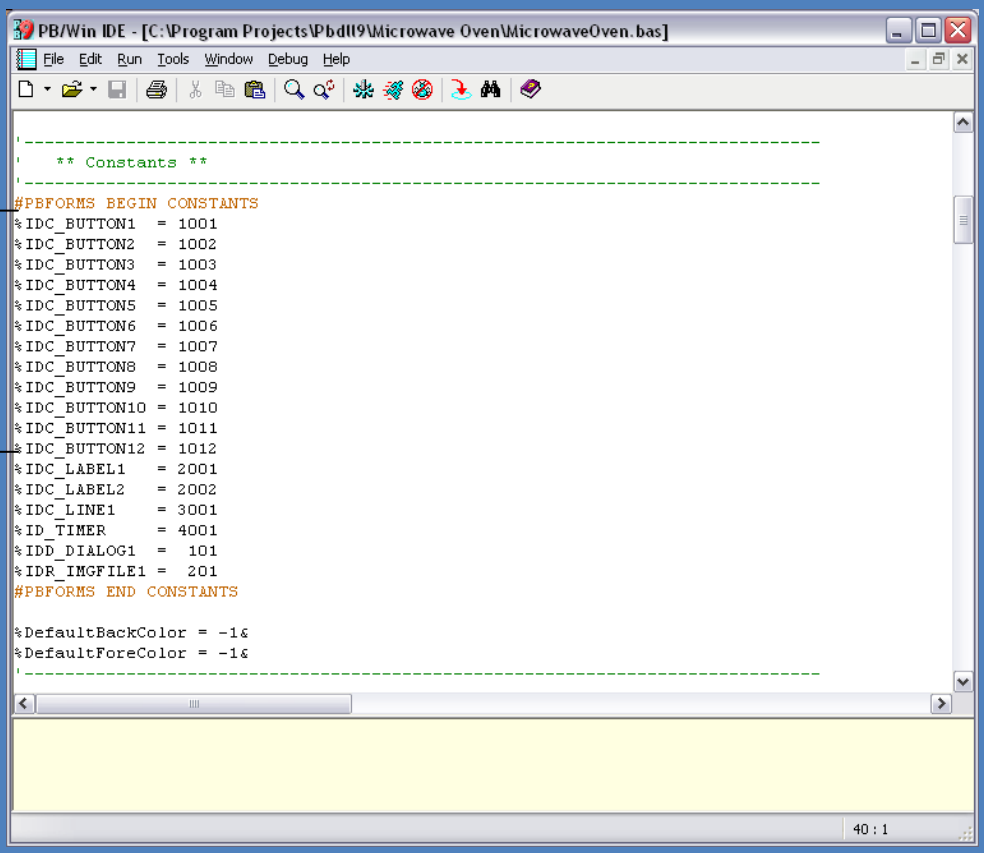

Figure 5 Constants for the dialog and controls

This is the function that initializes the cTime class in figure 6

The only place in the program where the timeObject object is created

```
'-------------------------------------------------------------------------------
'   ** Supporting Procedures **
'-------------------------------------------------------------------------------
FUNCTION StartCooking() AS STRING

    LOCAL second AS INTEGER
    LOCAL minute AS INTEGER

    timeis = RIGHT$(("0000" & timeis), 4) ' pad left with 4 zeros

    ' extract seconds and minutes
    second = VAL(RIGHT$(timeis, 2))
    minute = VAL(LEFT$(timeis, 2))

    'Create and initialize the cTime Class
    timeObject = CLASS "cTime"
    timeObject.new(minute, second)

    FUNCTION = FormatTime(timeObject.minute, timeObject.second)

END FUNCTION ' StartCooking

'-------------------------------------------------------------------------------
```

Figure 6 Start Cooking function

Code to build the form:

PB/Win IDE - [C:\Program Projects\Pbdll9\Microwave Oven\MicrowaveOven.bas]

File Edit Run Tools Window Debug Help

```
    DIALOG NEW hParent, "Microwave Oven", 221, 137, 364, 159, %WS_POPUP OR _
        %WS_BORDER OR %WS_DLGFRAME OR %WS_SYSMENU OR %WS_MINIMIZEBOX OR _
        %WS_MAXIMIZEBOX OR %WS_CLIPSIBLINGS OR %WS_VISIBLE OR %DS_MODALFRAME _
        OR %DS_3DLOOK OR %DS_NOFAILCREATE OR %DS_SETFONT, _
        %WS_EX_CONTROLPARENT OR %WS_EX_LEFT OR %WS_EX_LTRREADING OR _
        %WS_EX_RIGHTSCROLLBAR, TO hDlg
    DIALOG  SET ICON    hDlg, "#" + FORMAT$(%IDR_IMGFILE1)
    CONTROL ADD LABEL,  hDlg, %IDC_LABEL1, "", 10, 5, 225, 145, %WS_CHILD OR _
        %WS_VISIBLE OR %WS_BORDER OR %SS_LEFT OR %SS_NOWORDWRAP OR _
        %SS_SIMPLE, %WS_EX_LEFT OR %WS_EX_LTRREADING
    CONTROL ADD LINE,   hDlg, %IDC_LINE1, "", 245, 5, 110, 145, _
        %WS_CHILD OR %WS_VISIBLE OR %WS_BORDER
    CONTROL ADD LABEL,  hDlg, %IDC_LABEL2, "Microwave Oven", 260, 18, 82, 30, _
        %WS_CHILD OR %WS_VISIBLE OR %WS_BORDER OR %SS_CENTERIMAGE OR _
        %SS_CENTER, %WS_EX_LEFT OR %WS_EX_LTRREADING
    CONTROL ADD BUTTON, hDlg, %IDC_BUTTON1, "1", 270, 60, 20, 15, %WS_CHILD _
        OR %WS_VISIBLE OR %WS_TABSTOP OR %BS_TEXT OR %BS_PUSHBUTTON OR _
        %BS_FLAT OR %BS_CENTER OR %BS_VCENTER, %WS_EX_LEFT OR _
        %WS_EX_LTRREADING
    CONTROL ADD BUTTON, hDlg, %IDC_BUTTON2, "2", 290, 60, 20, 15, %WS_CHILD _
        OR %WS_VISIBLE OR %WS_TABSTOP OR %BS_TEXT OR %BS_PUSHBUTTON OR _
        %BS_FLAT OR %BS_CENTER OR %BS_VCENTER, %WS_EX_LEFT OR _
        %WS_EX_LTRREADING
    CONTROL ADD BUTTON, hDlg, %IDC_BUTTON3, "3", 310, 60, 20, 15, %WS_CHILD _
        OR %WS_VISIBLE OR %WS_TABSTOP OR %BS_TEXT OR %BS_PUSHBUTTON OR _
        %BS_FLAT OR %BS_CENTER OR %BS_VCENTER, %WS_EX_LEFT OR _
        %WS_EX_LTRREADING
    CONTROL ADD BUTTON, hDlg, %IDC_BUTTON4, "4", 270, 75, 20, 15, %WS_CHILD _
        OR %WS_VISIBLE OR %WS_TABSTOP OR %BS_TEXT OR %BS_PUSHBUTTON OR _
        %BS_FLAT OR %BS_CENTER OR %BS_VCENTER, %WS_EX_LEFT OR _
        %WS_EX_LTRREADING
    CONTROL ADD BUTTON, hDlg, %IDC_BUTTON5, "5", 290, 75, 20, 15, %WS_CHILD _
        OR %WS_VISIBLE OR %WS_TABSTOP OR %BS_TEXT OR %BS_PUSHBUTTON OR _
        %BS_FLAT OR %BS_CENTER OR %BS_VCENTER, %WS_EX_LEFT OR _
        %WS_EX_LTRREADING
```

198 : 48

Figure 7

PB/Win IDE - [C:\Program Projects\Pbdll9\Microwave Oven\MicrowaveOven.bas]

File Edit Run Tools Window Debug Help

```
        %WS_EX_LTRREADING
    CONTROL ADD BUTTON, hDlg, %IDC_BUTTON5, "5", 290, 75, 20, 15, %WS_CHILD _
        OR %WS_VISIBLE OR %WS_TABSTOP OR %BS_TEXT OR %BS_PUSHBUTTON OR _
        %BS_FLAT OR %BS_CENTER OR %BS_VCENTER, %WS_EX_LEFT OR _
        %WS_EX_LTRREADING
    CONTROL ADD BUTTON, hDlg, %IDC_BUTTON6, "6", 310, 75, 20, 15, %WS_CHILD _
        OR %WS_VISIBLE OR %WS_TABSTOP OR %BS_TEXT OR %BS_PUSHBUTTON OR _
        %BS_FLAT OR %BS_CENTER OR %BS_VCENTER, %WS_EX_LEFT OR _
        %WS_EX_LTRREADING
    CONTROL ADD BUTTON, hDlg, %IDC_BUTTON7, "7", 270, 90, 20, 15, %WS_CHILD _
        OR %WS_VISIBLE OR %WS_TABSTOP OR %BS_TEXT OR %BS_PUSHBUTTON OR _
        %BS_FLAT OR %BS_CENTER OR %BS_VCENTER, %WS_EX_LEFT OR _
        %WS_EX_LTRREADING
    CONTROL ADD BUTTON, hDlg, %IDC_BUTTON8, "8", 290, 90, 20, 15, %WS_CHILD _
        OR %WS_VISIBLE OR %WS_TABSTOP OR %BS_TEXT OR %BS_PUSHBUTTON OR _
        %BS_FLAT OR %BS_CENTER OR %BS_VCENTER, %WS_EX_LEFT OR _
        %WS_EX_LTRREADING
    CONTROL ADD BUTTON, hDlg, %IDC_BUTTON9, "9", 310, 90, 20, 15, %WS_CHILD _
        OR %WS_VISIBLE OR %WS_TABSTOP OR %BS_TEXT OR %BS_PUSHBUTTON OR _
        %BS_FLAT OR %BS_CENTER OR %BS_VCENTER, %WS_EX_LEFT OR _
        %WS_EX_LTRREADING
    CONTROL ADD BUTTON, hDlg, %IDC_BUTTON10, "0", 290, 105, 20, 15, %WS_CHILD _
        OR %WS_VISIBLE OR %WS_TABSTOP OR %BS_TEXT OR %BS_PUSHBUTTON OR _
        %BS_FLAT OR %BS_CENTER OR %BS_VCENTER, %WS_EX_LEFT OR _
        %WS_EX_LTRREADING
    CONTROL ADD BUTTON, hDlg, %IDC_BUTTON11, "Start", 250, 130, 40, 15
    CONTROL ADD BUTTON, hDlg, %IDC_BUTTON12, "Clear", 310, 130, 40, 15

        hFont1 = PBFormsMakeFont("Segoe UI", 12, 400, %FALSE, %FALSE, %FALSE, _
        %ANSI_CHARSET)

    CONTROL SEND hDlg, %IDC_LABEL2, %WM_SETFONT, hFont1, 0
#PBFORMS END DIALOG

    DIALOG SHOW MODAL hDlg, CALL ShowDIALOG1Proc TO lRslt
```

216 : 66

Setup a new Font for the Label.

Figure 8

Display time function and format the time function:

```
FUNCTION DisplayTime() AS STRING

    LOCAL second AS INTEGER
    LOCAL minute AS INTEGER
    LOCAL MyDisplay AS STRING ' String displays current input

    ' if too much input entered just use the left 4 places
    IF LEN(timeIs) > 4 THEN
        timeis = LEFT$(timeis, 4)
    END IF

    MyDisplay = RIGHT$(("0000" & timeis), 4) ' pad left with 4 zeros

    ' extract seconds and minutes
    second = VAL(RIGHT$(MyDisplay, 2))
    minute = VAL(LEFT$(MyDisplay, 2))

    FUNCTION = FormatTime(minute, second)

END FUNCTION ' DisplayTime

'-------------------------------------------------------------------------------

FUNCTION FormatTime(BYVAL mm AS INTEGER, BYVAL ss AS INTEGER) AS STRING
    FUNCTION = FORMAT$(mm, "0#") & ":" & FORMAT$(ss, "0#")
END FUNCTION ' FormatTime
'-------------------------------------------------------------------------------
```

Figure 9

Timer event handler:

PB/Win IDE - [C:\Program Projects\Pbdll9\Microwave Oven\MicrowaveOven.bas]

File Edit Run Tools Window Debug Help

```
        CASE %WM_NCACTIVATE

            STATIC hWndSaveFocus AS DWORD
            IF ISFALSE CB.WPARAM THEN
                ' Save control focus
                hWndSaveFocus = GetFocus()
            ELSEIF hWndSaveFocus THEN
                ' Restore control focus
                SetFocus(hWndSaveFocus)
                hWndSaveFocus = 0
            END IF

        CASE %WM_TIMER ' Process timer ticks

            IF timeObject.Second > 0 THEN
                timeObject.Second = timeObject.Second - 1
                CONTROL SET TEXT CB.HNDL, %IDC_LABEL2, _
                    FormatTime(timeObject.minute, timeObject.second)
            ELSEIF timeObject.Minute > 0 THEN
                timeObject.Minute = timeObject.Minute - 1
                timeObject.Second = 59
                CONTROL SET TEXT CB.HNDL, %IDC_LABEL2, _
                    FormatTime(timeObject.minute, timeObject.second)
            ELSE ' countdown finished
                KillTimer CB.HNDL, %ID_TIMER
                BEEP
                timeis = ""
                CONTROL SET TEXT CB.HNDL, %IDC_LABEL2, "Done!"
                CONTROL SET COLOR CB.HNDL, %IDC_LABEL1, %DefaultForeColor, %DefaultBackColor
                CONTROL REDRAW CB.HNDL, %IDC_LABEL1
                timeObject = NOTHING
            END IF

        CASE %WM_COMMAND
```

138 : 27

Return the Label's back color to the default and redraw the label.

Figure 10

Button press handler code.

Select Case statement for all ten buttons.

Get the text from the button to add to display

Set the label's back color to yellow and redraw the label to simulate cooking

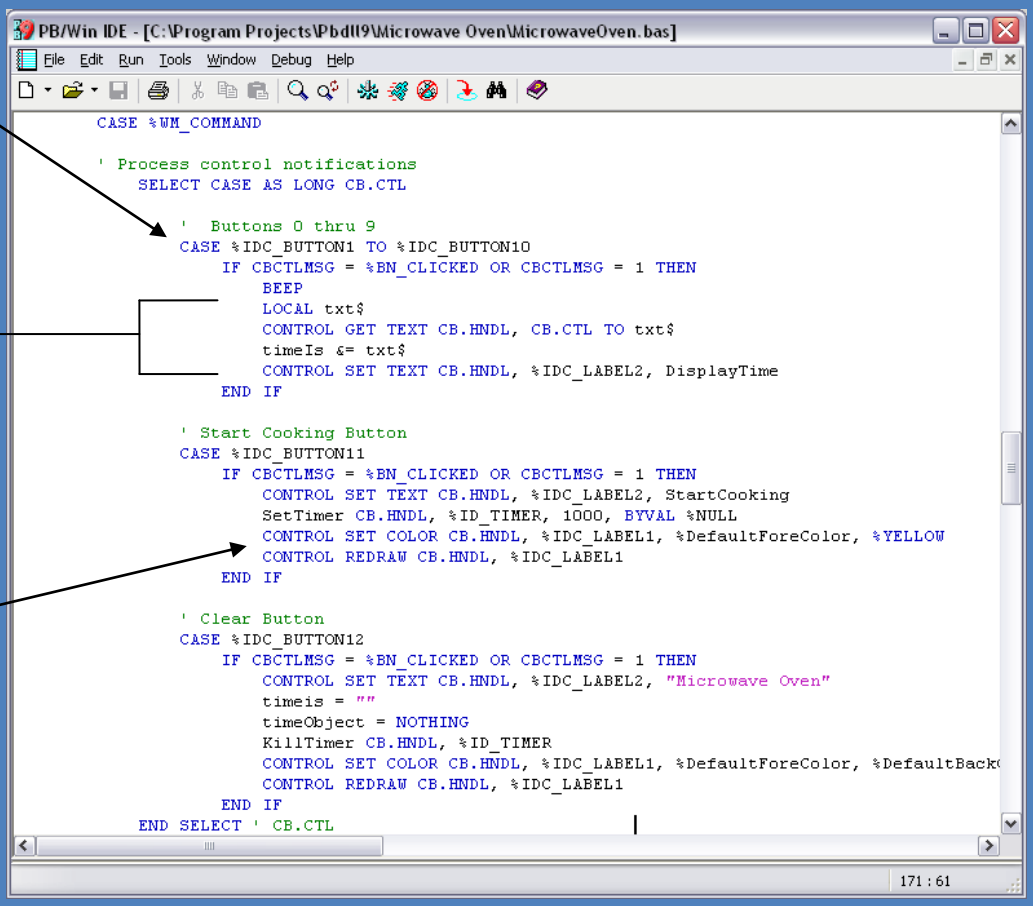

Figure 11

Declarations and global variables used :

Need these throughout the program.

```
'------------------------------------------------------------------------------
'    ** Declarations **
'------------------------------------------------------------------------------
#PBFORMS BEGIN DECLARATIONS
DECLARE CALLBACK FUNCTION ShowDIALOG1Proc()
DECLARE FUNCTION ShowDIALOG1(BYVAL hParent AS DWORD) AS LONG
#PBFORMS END DECLARATIONS

DECLARE FUNCTION StartCooking() AS STRING
DECLARE FUNCTION DisplayTime() AS STRING
DECLARE FUNCTION FormatTime(BYVAL mm AS INTEGER, BYVAL ss AS INTEGER) AS STRING
'------------------------------------------------------------------------------

'------------------------------------------------------------------------------
'    ** Globals **
'------------------------------------------------------------------------------
GLOBAL timeIs          AS STRING
GLOBAL timeObject      AS Itime
'------------------------------------------------------------------------------

'------------------------------------------------------------------------------
'    ** Main Application Entry Point **
'------------------------------------------------------------------------------
FUNCTION PBMAIN()
    ShowDIALOG1 %HWND_DESKTOP
END FUNCTION
'------------------------------------------------------------------------------
```

Figure 12

cTime Class:

```
CLASS cTime

    ' declare Instance variables for minute and second
    INSTANCE m_MinuteValue AS INTEGER
    INSTANCE m_secondValue AS INTEGER

    'definition of a set of methods and properties which are implemented on an object
    INTERFACE Itime

        ' offers both direct access and Dispatch access to the interface methods
        INHERIT IDISPATCH

        ' Time constructor, minute and second supplied
        METHOD NEW(BYVAL mm AS INTEGER, BYVAL ss AS INTEGER)
            me.Minute = mm ' invokes Minute accessor
            me.Second = ss ' invokes Second accessor
        END METHOD ' NEW

        ' property Minute
        PROPERTY GET Minute() AS INTEGER
            PROPERTY = m_MinuteValue
        END PROPERTY ' end of Get accessor

        PROPERTY SET Minute(BYVAL value AS INTEGER)
            IF (value < 60) THEN 'if minute value entered is valid
                m_MinuteValue = value
            ELSE
                m_MinuteValue = 0 ' set invalid input to 0
            END IF
        END PROPERTY ' end of Set accessor

        ' property Second
        PROPERTY GET Second() AS INTEGER
            PROPERTY = m_secondValue
        END PROPERTY ' end of Get accessor

        PROPERTY SET Second(BYVAL value AS INTEGER)
            IF (value < 60) THEN 'if second value entered is valid
                m_secondValue = value
            ELSE
                m_secondValue = 0 ' set invalid input to 0
            END IF
        END PROPERTY ' end of Set accessor
    END INTERFACE ' Itime
END CLASS ' cTime
```

MicrowaveOven.bas:

```
#PBFORMS CREATED V1.51
'-------------------------------------------------------------------------------
' The first line in this file is a PB/Forms metastatement.
' It should ALWAYS be the first line of the file.
' PB/Forms metastatements are placed at the beginning and
' end of "Named Blocks" of code that should be edited
' with PBForms only. Do not manually edit or delete these
' metastatements or PB/Forms will not be able to reread
' the file correctly.  See the PB/Forms documentation for
' more information.
' Named blocks begin like this:    #PBFORMS BEGIN ...
' Named blocks end like this:      #PBFORMS END ...
' Other PB/Forms metastatements such as:
'     #PBFORMS DECLARATIONS
' are used by PB/Forms to insert additional code.
' Feel free to make changes anywhere else in the file.
'-------------------------------------------------------------------------------

#COMPILE EXE
#DIM ALL

'-------------------------------------------------------------------------------
'   ** Includes **
'-------------------------------------------------------------------------------
#PBFORMS BEGIN INCLUDES
#RESOURCE "MicrowaveOven.pbr"
#INCLUDE ONCE "WIN32API.INC"
#INCLUDE ONCE "PBForms.INC"
#PBFORMS END INCLUDES
'-------------------------------------------------------------------------------

'-------------------------------------------------------------------------------
'   ** Classes Includes**
'-------------------------------------------------------------------------------
#INCLUDE "cTime.inc"
'-------------------------------------------------------------------------------

'-------------------------------------------------------------------------------
'   ** Constants **
'-------------------------------------------------------------------------------
#PBFORMS BEGIN CONSTANTS
%IDC_BUTTON1  = 1001
%IDC_BUTTON2  = 1002
%IDC_BUTTON3  = 1003
%IDC_BUTTON4  = 1004
%IDC_BUTTON5  = 1005
%IDC_BUTTON6  = 1006
%IDC_BUTTON7  = 1007
%IDC_BUTTON8  = 1008
%IDC_BUTTON9  = 1009
%IDC_BUTTON10 = 1010
%IDC_BUTTON11 = 1011
%IDC_BUTTON12 = 1012
%IDC_LABEL1   = 2001
%IDC_LABEL2   = 2002
%IDC_LINE1    = 3001
%ID_TIMER     = 4001
%IDD_DIALOG1  =  101
%IDR_IMGFILE1 =  201
#PBFORMS END CONSTANTS
```

```
%DefaultBackColor = -1&
%DefaultForeColor = -1&
'--------------------------------------------------------------------------------

'--------------------------------------------------------------------------------
'   ** Declarations **
'--------------------------------------------------------------------------------
#PBFORMS BEGIN DECLARATIONS
DECLARE CALLBACK FUNCTION ShowDIALOG1Proc()
DECLARE FUNCTION ShowDIALOG1(BYVAL hParent AS DWORD) AS LONG
#PBFORMS END DECLARATIONS

DECLARE FUNCTION StartCooking() AS STRING
DECLARE FUNCTION DisplayTime() AS STRING
DECLARE FUNCTION FormatTime(BYVAL mm AS INTEGER, BYVAL ss AS INTEGER) AS STRING
'--------------------------------------------------------------------------------

'--------------------------------------------------------------------------------
'   ** Globals **
'--------------------------------------------------------------------------------
GLOBAL timeIs          AS STRING
GLOBAL timeObject      AS Itime
'--------------------------------------------------------------------------------

'--------------------------------------------------------------------------------
'   ** Main Application Entry Point **
'--------------------------------------------------------------------------------
FUNCTION PBMAIN()
    ShowDIALOG1 %HWND_DESKTOP
END FUNCTION
'--------------------------------------------------------------------------------

'--------------------------------------------------------------------------------
'   ** CallBacks **
'--------------------------------------------------------------------------------
CALLBACK FUNCTION ShowDIALOG1Proc()

    SELECT CASE AS LONG CB.MSG

        CASE %WM_INITDIALOG
            ' Initialization handler

        CASE %WM_NCACTIVATE

            STATIC hWndSaveFocus AS DWORD
            IF ISFALSE CB.WPARAM THEN
                ' Save control focus
                hWndSaveFocus = GetFocus()
            ELSEIF hWndSaveFocus THEN
                ' Restore control focus
                SetFocus(hWndSaveFocus)
                hWndSaveFocus = 0
            END IF

        CASE %WM_TIMER ' Process timer ticks

            IF timeObject.Second > 0 THEN
                timeObject.Second = timeObject.Second - 1
                CONTROL SET TEXT CB.HNDL, %IDC_LABEL2, _
                    FormatTime(timeObject.minute, timeObject.second)
            ELSEIF timeObject.Minute > 0 THEN
                timeObject.Minute = timeObject.Minute - 1
```

```
                timeObject.Second = 59
                CONTROL SET TEXT CB.HNDL, %IDC_LABEL2, _
                    FormatTime(timeObject.minute, timeObject.second)
            ELSE ' countdown finished
                KillTimer CB.HNDL, %ID_TIMER
                BEEP
                timeis = ""
                CONTROL SET TEXT CB.HNDL, %IDC_LABEL2, "Done!"
                CONTROL SET COLOR CB.HNDL, %IDC_LABEL1, %DefaultForeColor, _
                %DefaultBackColor
                CONTROL REDRAW CB.HNDL, %IDC_LABEL1
                timeObject = NOTHING
            END IF

        CASE %WM_COMMAND

        ' Process control notifications
            SELECT CASE AS LONG CB.CTL

                '  Buttons 0 thru 9
                CASE %IDC_BUTTON1 TO %IDC_BUTTON10
                    IF CBCTLMSG = %BN_CLICKED OR CBCTLMSG = 1 THEN
                        BEEP
                        LOCAL txt$
                        CONTROL GET TEXT CB.HNDL, CB.CTL TO txt$
                        timeIs &= txt$
                        CONTROL SET TEXT CB.HNDL, %IDC_LABEL2, DisplayTime
                    END IF

                ' Start Cooking Button
                CASE %IDC_BUTTON11
                    IF CBCTLMSG = %BN_CLICKED OR CBCTLMSG = 1 THEN
                        CONTROL SET TEXT CB.HNDL, %IDC_LABEL2, StartCooking
                        SetTimer CB.HNDL, %ID_TIMER, 1000, BYVAL %NULL
                        CONTROL SET COLOR CB.HNDL, %IDC_LABEL1, %DefaultForeColor, _
                        %YELLOW
                        CONTROL REDRAW CB.HNDL, %IDC_LABEL1
                    END IF

                ' Clear Button
                CASE %IDC_BUTTON12
                    IF CBCTLMSG = %BN_CLICKED OR CBCTLMSG = 1 THEN
                        CONTROL SET TEXT CB.HNDL, %IDC_LABEL2, "Microwave Oven"
                        timeis = ""
                        timeObject = NOTHING
                        KillTimer CB.HNDL, %ID_TIMER
                        CONTROL SET COLOR CB.HNDL, %IDC_LABEL1, %DefaultForeColor, _
                        %DefaultBackColor
                        CONTROL REDRAW CB.HNDL, %IDC_LABEL1
                    END IF
            END SELECT ' CB.CTL
    END SELECT ' CB.MSG
END FUNCTION ' ShowDIALOG1Proc
'------------------------------------------------------------------------------

'------------------------------------------------------------------------------
'   ** Dialogs **
'------------------------------------------------------------------------------
FUNCTION ShowDIALOG1(BYVAL hParent AS DWORD) AS LONG
    LOCAL lRslt  AS LONG

#PBFORMS BEGIN DIALOG %IDD_DIALOG1->->
    LOCAL hDlg   AS DWORD
```

```
LOCAL hFont1 AS DWORD

DIALOG NEW hParent, "Microwave Oven", 221, 137, 364, 159, %WS_POPUP OR _
    %WS_BORDER OR %WS_DLGFRAME OR %WS_SYSMENU OR %WS_MINIMIZEBOX OR _
    %WS_MAXIMIZEBOX OR %WS_CLIPSIBLINGS OR %WS_VISIBLE OR %DS_MODALFRAME _
    OR %DS_3DLOOK OR %DS_NOFAILCREATE OR %DS_SETFONT, _
    %WS_EX_CONTROLPARENT OR %WS_EX_LEFT OR %WS_EX_LTRREADING OR _
    %WS_EX_RIGHTSCROLLBAR, TO hDlg
DIALOG SET ICON    hDlg, "#" + FORMAT$(%IDR_IMGFILE1)
CONTROL ADD LABEL,  hDlg, %IDC_LABEL1, "", 10, 5, 225, 145, %WS_CHILD OR _
    %WS_VISIBLE OR %WS_BORDER OR %SS_LEFT OR %SS_NOWORDWRAP OR _
    %SS_SIMPLE, %WS_EX_LEFT OR %WS_EX_LTRREADING
CONTROL ADD LINE,   hDlg, %IDC_LINE1, "", 245, 5, 110, 145, _
    %WS_CHILD OR %WS_VISIBLE OR %WS_BORDER
CONTROL ADD LABEL,  hDlg, %IDC_LABEL2, "Microwave Oven", 260, 18, 82, 30, _
    %WS_CHILD OR %WS_VISIBLE OR %WS_BORDER OR %SS_CENTERIMAGE OR _
    %SS_CENTER, %WS_EX_LEFT OR %WS_EX_LTRREADING
CONTROL ADD BUTTON, hDlg, %IDC_BUTTON1, "1", 270, 60, 20, 15, %WS_CHILD _
    OR %WS_VISIBLE OR %WS_TABSTOP OR %BS_TEXT OR %BS_PUSHBUTTON OR _
    %BS_FLAT OR %BS_CENTER OR %BS_VCENTER, %WS_EX_LEFT OR _
    %WS_EX_LTRREADING
CONTROL ADD BUTTON, hDlg, %IDC_BUTTON2, "2", 290, 60, 20, 15, %WS_CHILD _
    OR %WS_VISIBLE OR %WS_TABSTOP OR %BS_TEXT OR %BS_PUSHBUTTON OR _
    %BS_FLAT OR %BS_CENTER OR %BS_VCENTER, %WS_EX_LEFT OR _
    %WS_EX_LTRREADING
CONTROL ADD BUTTON, hDlg, %IDC_BUTTON3, "3", 310, 60, 20, 15, %WS_CHILD _
    OR %WS_VISIBLE OR %WS_TABSTOP OR %BS_TEXT OR %BS_PUSHBUTTON OR _
    %BS_FLAT OR %BS_CENTER OR %BS_VCENTER, %WS_EX_LEFT OR _
    %WS_EX_LTRREADING
CONTROL ADD BUTTON, hDlg, %IDC_BUTTON4, "4", 270, 75, 20, 15, %WS_CHILD _
    OR %WS_VISIBLE OR %WS_TABSTOP OR %BS_TEXT OR %BS_PUSHBUTTON OR _
    %BS_FLAT OR %BS_CENTER OR %BS_VCENTER, %WS_EX_LEFT OR _
    %WS_EX_LTRREADING
CONTROL ADD BUTTON, hDlg, %IDC_BUTTON5, "5", 290, 75, 20, 15, %WS_CHILD _
    OR %WS_VISIBLE OR %WS_TABSTOP OR %BS_TEXT OR %BS_PUSHBUTTON OR _
    %BS_FLAT OR %BS_CENTER OR %BS_VCENTER, %WS_EX_LEFT OR _
    %WS_EX_LTRREADING
CONTROL ADD BUTTON, hDlg, %IDC_BUTTON6, "6", 310, 75, 20, 15, %WS_CHILD _
    OR %WS_VISIBLE OR %WS_TABSTOP OR %BS_TEXT OR %BS_PUSHBUTTON OR _
    %BS_FLAT OR %BS_CENTER OR %BS_VCENTER, %WS_EX_LEFT OR _
    %WS_EX_LTRREADING
CONTROL ADD BUTTON, hDlg, %IDC_BUTTON7, "7", 270, 90, 20, 15, %WS_CHILD _
    OR %WS_VISIBLE OR %WS_TABSTOP OR %BS_TEXT OR %BS_PUSHBUTTON OR _
    %BS_FLAT OR %BS_CENTER OR %BS_VCENTER, %WS_EX_LEFT OR _
    %WS_EX_LTRREADING
CONTROL ADD BUTTON, hDlg, %IDC_BUTTON8, "8", 290, 90, 20, 15, %WS_CHILD _
    OR %WS_VISIBLE OR %WS_TABSTOP OR %BS_TEXT OR %BS_PUSHBUTTON OR _
    %BS_FLAT OR %BS_CENTER OR %BS_VCENTER, %WS_EX_LEFT OR _
    %WS_EX_LTRREADING
CONTROL ADD BUTTON, hDlg, %IDC_BUTTON9, "9", 310, 90, 20, 15, %WS_CHILD _
    OR %WS_VISIBLE OR %WS_TABSTOP OR %BS_TEXT OR %BS_PUSHBUTTON OR _
    %BS_FLAT OR %BS_CENTER OR %BS_VCENTER, %WS_EX_LEFT OR _
    %WS_EX_LTRREADING
CONTROL ADD BUTTON, hDlg, %IDC_BUTTON10, "0", 290, 105, 20, 15, %WS_CHILD _
    OR %WS_VISIBLE OR %WS_TABSTOP OR %BS_TEXT OR %BS_PUSHBUTTON OR _
    %BS_FLAT OR %BS_CENTER OR %BS_VCENTER, %WS_EX_LEFT OR _
    %WS_EX_LTRREADING
CONTROL ADD BUTTON, hDlg, %IDC_BUTTON11, "Start", 250, 130, 40, 15
CONTROL ADD BUTTON, hDlg, %IDC_BUTTON12, "Clear", 310, 130, 40, 15

    hFont1 = PBFormsMakeFont("Segoe UI", 12, 400, %FALSE, %FALSE, %FALSE, _
    %ANSI_CHARSET)
```

```
    CONTROL SEND hDlg, %IDC_LABEL2, %WM_SETFONT, hFont1, 0
#PBFORMS END DIALOG

    DIALOG SHOW MODAL hDlg, CALL ShowDIALOG1Proc TO lRslt

#PBFORMS BEGIN CLEANUP %IDD_DIALOG1
    DeleteObject hFont1
#PBFORMS END CLEANUP

    FUNCTION = lRslt
END FUNCTION ' ShowDIALOG1
'------------------------------------------------------------------------------

'------------------------------------------------------------------------------
'   ** Supporting Procedures **
'------------------------------------------------------------------------------
FUNCTION StartCooking() AS STRING

    LOCAL second AS INTEGER
    LOCAL minute AS INTEGER

    timeis = RIGHT$(("0000" & timeis), 4) ' pad left with 4 zeros

    ' extract seconds and minutes
    second = VAL(RIGHT$(timeis, 2))
    minute = VAL(LEFT$(timeis, 2))

    'Create and initialize the cTime Class
    timeObject = CLASS "cTime"
    timeObject.new(minute, second)

    FUNCTION = FormatTime(timeObject.minute, timeObject.second)

END FUNCTION ' StartCooking

'------------------------------------------------------------------------------

FUNCTION DisplayTime() AS STRING

    LOCAL second AS INTEGER
    LOCAL minute AS INTEGER
    LOCAL MyDisplay AS STRING ' String displays current input

    ' if too much input entered just use the left 4 places
    IF LEN(timeIs) > 4 THEN
        timeis = LEFT$(timeis, 4)
    END IF

    MyDisplay = RIGHT$(("0000" & timeis), 4) ' pad left with 4 zeros

    ' extract seconds and minutes
    second = VAL(RIGHT$(MyDisplay, 2))
    minute = VAL(LEFT$(MyDisplay, 2))

    FUNCTION = FormatTime(minute, second)

END FUNCTION ' DisplayTime

'------------------------------------------------------------------------------
FUNCTION FormatTime(BYVAL mm AS INTEGER, BYVAL ss AS INTEGER) AS STRING
    FUNCTION = FORMAT$(mm, "0#") & ":" & FORMAT$(ss, "0#")
END FUNCTION ' FormatTime
'------------------------------------------------------------------------------
```

Use what you have learned programming challenge:

DVD Burner Application Create an application that simulates a DVD burner. Users create a DVD with their choice of title and bonus material. Design the GUI like the one in Figure 13. Create a class (cDVD) to represent the DVD object and another class (cBonus) to represent the bonus material for the DVD object.

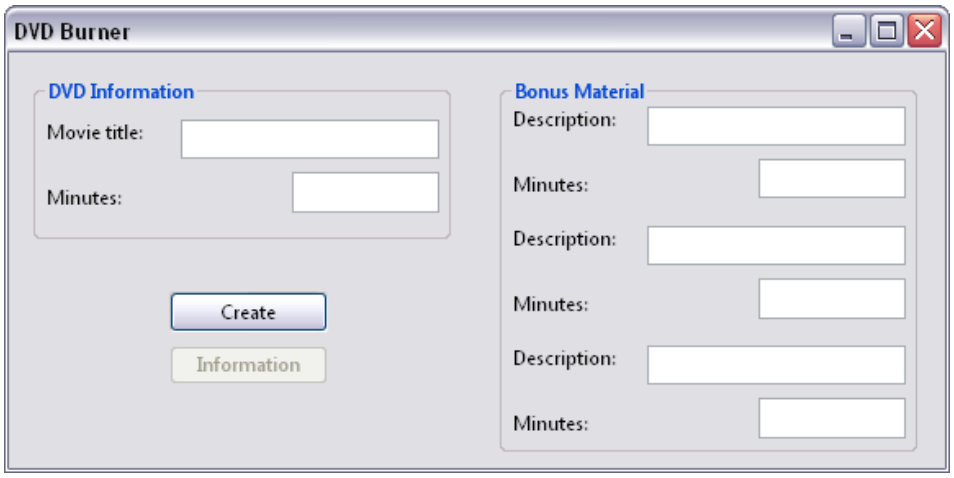

Figure 13 **DVD Burner** application's GUI

1. ***Creating the bonus material object.*** Create a class, and name it cBonus. The class's objects each represent one bonus-material item on the DVD. Each Bonus object should have a name (description) and a length (in minutes).
2. ***Creating the DVD class.*** Create a class, and name it cDVD. This class contains the movie title and the length of the movie. Create properties that allow clients to get and set the movie's title and length. Create a Read Only property to get the movie's bonus material as a String containing each bonus item's name and length. This is also the place to create an array of Bonus objects to store the bonus material.
3. ***Creating the necessary variables.*** Before you define the Create Button's event handler, create a DVD class instance variable that is STATIC in the Dialog's callback function. Inside the Create Button's event, create the necessary variables to store the information from the TextBoxes on the GUI.
4. ***Adding bonus-material information.*** Add the description and length of each specified bonus item to the Bonus array you created in the DVD class using a method called NEW.
5. ***Creating a DVD object.*** Use information about the movie, its title, length and the array of bonus materials to make your DVD object.
6. ***Displaying the output.*** Locate the Information Button's event, add a String containing the complete information on the DVD object and Bonus Materials that you created earlier and display this String to a Message Box. The Information button is not enabled until a DVD is created.
7. ***Running the application.*** Select Run > Compile and Execute to run your application. Enter information for several DVDs. After the information is entered for each, click the Create Button. Then click the Information Button and verify that the information being displayed is correct for your newly created DVD.