Chapter 9

Visual Basic Program: Infusion Maintenance Application

In this chapter, you will learn how to use the following Visual Basic Application functions to World Class standards:

- **Open the Visual Basic Editor and**
- **Beginning a New Visual Basic Application**
- **Laying Out a User Input Form in Visual Basic**
- **Common Controls in Visual Basic**
- **Inserting a Labels and Textboxes in the Form**
- **Inserting a Group Box in the Form**
- **Inserting a Radio Button in the Form**
- **Inserting a Checkbox in the Form**
- **Inserting a Command Buttons in the Form**
- **Inserting a Picture in the Form**
- **Adding a Copyright Statement to a Form**
- **Adding a Copyright Statement**
- **Declaring Variables in the Program**
- **Setting Variables in the Program**
- **Computing Answers in the Program**
- **Outputting the Answers in the Program**
- **Programming for the Radio Button**
- **Programming for the Check Box**
- **Clearing the Data and Exiting the Program**
- **Adding a Browse and Open File Button**
- **Programming a Help Hyperlink and a Help Button**
- **Programming a Print Button**
- **Adding Code to Check for Blank Textboxes**
- **Controlling the Input in the Textbox**
- **Running the Program**

The Infusion Application

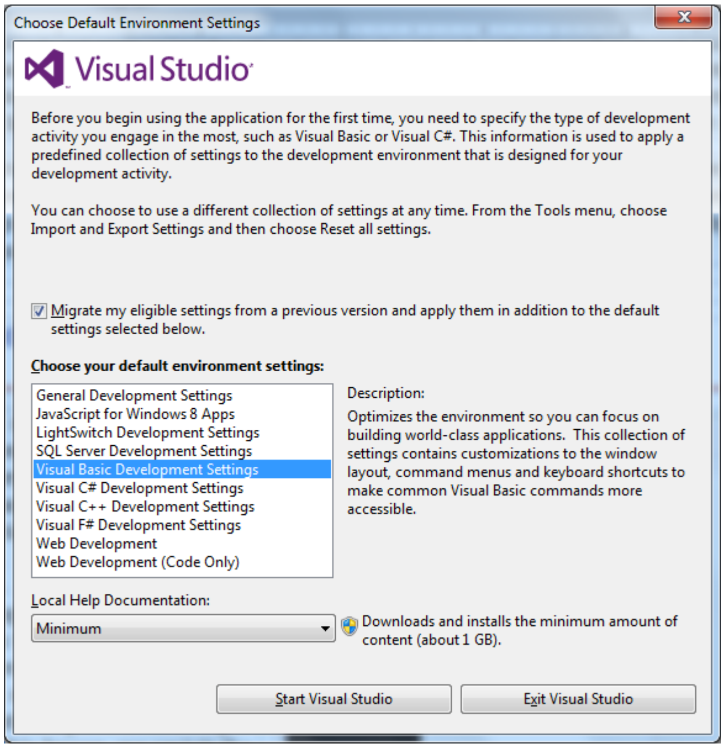
One semester ago, we were working with Veterinarian Technicians as they learned how to calculate the amount of fluid to use to treat dehydrated animals and the flow rate to set when administering it. After several weeks, we took the project to the computer department and we developed an application that the professionals could use to quickly help the animals. After several adaptations by the staff, a solution for the program was published. This chapter will take us through the process again.

In this lesson, we need to utilize textboxes, labels, option buttons, check boxes and command buttons as we have in previous training sessions. We will also utilize message boxes for help information. We will also check our input text boxes for missing or incorrect information. We will also have information appear on the form if the default for the drip rate is not selected.

In our tutorial, we will again use condition statements to make decisions and we will continue to concatenate text strings to construct information that is useful to the computer user. Although this program is larger than some done before, many components of the code are rhythmic and we can quickly spot the changes to each segment of the software.

Open the Visual Basic Editor

In this session, we will step through each procedure in adding labels, textboxes and command buttons and we will integrate into the tutorial the methods to add, subtract, multiply and divide numbers. We will also include formatting the answers as they are shown in the answer labels. As in every project, we will create variables, set their values, execute mathematical equations and output data. In this lesson, we revisit the procedure to add the computer date and time to the form. We open Visual Studio and we will choose the default environment as Visual Basic. Then we start Visual Studio.



Before you begin using the application for the first time, you need to specify the type of development activity you engage in the most, such as Visual Basic or Visual C#. This information is used to apply a predefined collection of settings to the development environment that is designed for your development activity.

You can choose to use a different collection of settings at any time. From the Tools menu, choose Import and Export Settings and then choose Reset all settings.

☑ Migrate my eligible settings from a previous version and apply them in addition to the default settings selected below.

Choose your default environment settings:

- General Development Settings
- JavaScript for Windows 8 Apps
- LightSwitch Development Settings
- SQL Server Development Settings
- Visual Basic Development Settings
- Visual C# Development Settings
- Visual C++ Development Settings
- Visual F# Development Settings
- Web Development
- Web Development (Code Only)

Description:
Optimizes the environment so you can focus on building world-class applications. This collection of settings contains customizations to the window layout, command menus and keyboard shortcuts to make common Visual Basic commands more accessible.

Local Help Documentation: Minimum

Downloads and installs the minimum amount of content (about 1 GB).

Figure 9.1 – Default Environment Setting

The next window we see is the Start Page.

To open a new application, we select New Project on the left pane of the Start menu. The New Project window will appear.

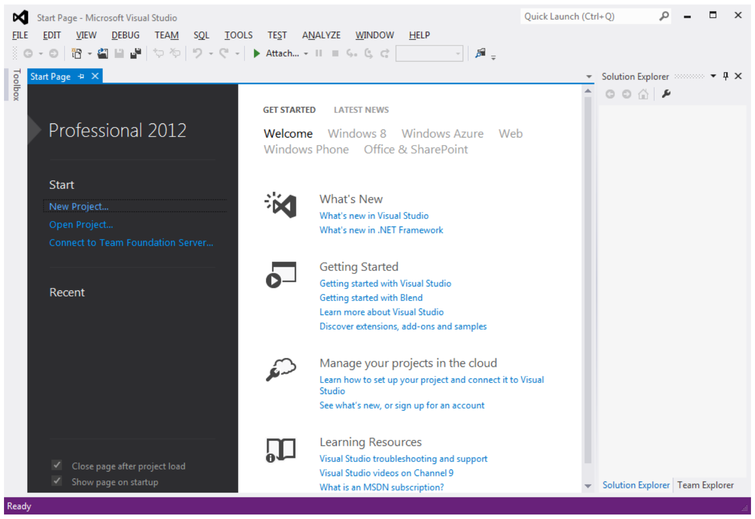
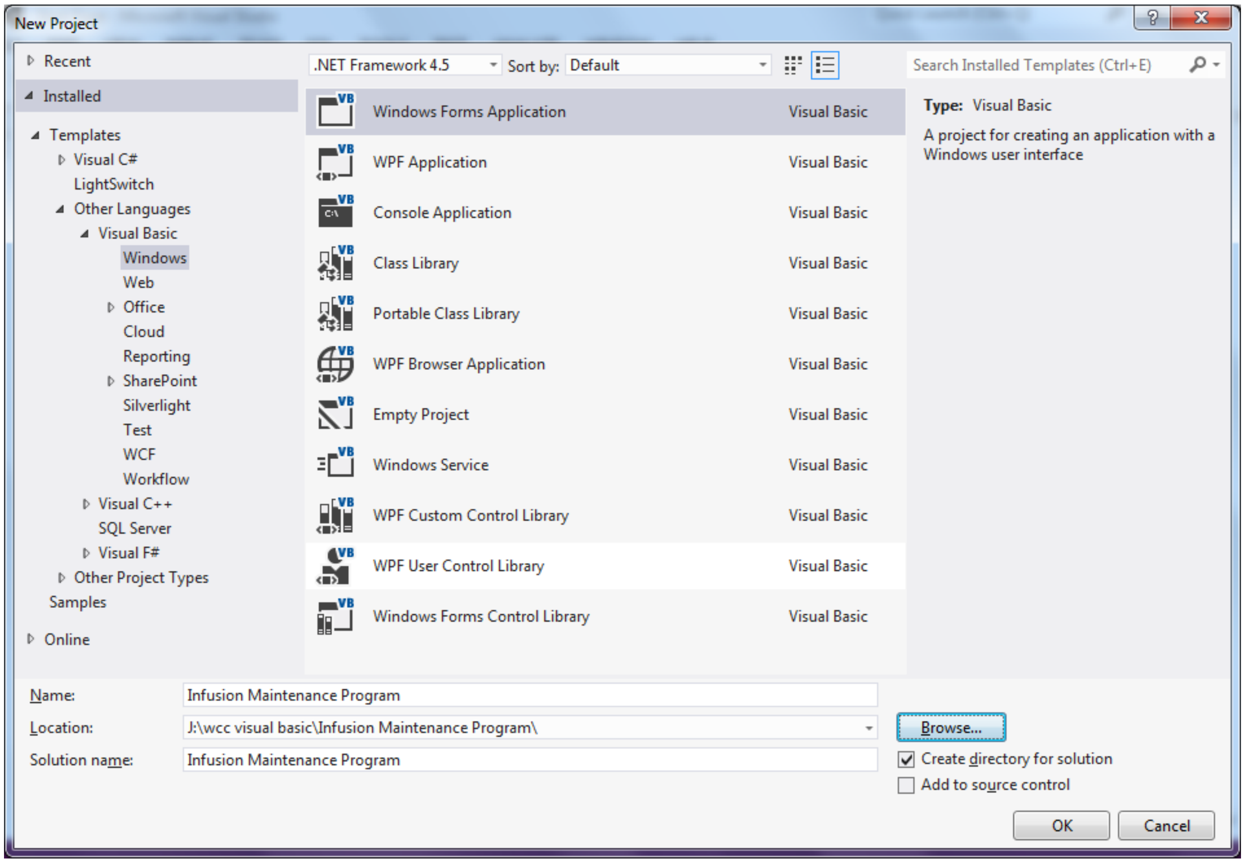

Start Page - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
FILE EDIT VIEW DEBUG TEAM SQL TOOLS TEST ANALYZE WINDOW HELP
Attach...
Toolbox
Start Page
Solution Explorer
Professional 2012
Start
New Project...
Open Project...
Connect to Team Foundation Server...
Recent
Close page after project load
Show page on startup
GET STARTED
LATEST NEWS
Welcome
Windows 8
Windows Azure
Web
Windows Phone
Office & SharePoint
What's New
What's new in Visual Studio
What's new in .NET Framework
Getting Started
Getting started with Visual Studio
Getting started with Blend
Learn more about Visual Studio
Discover extensions, add-ons and samples
Manage your projects in the cloud
Learn how to set up your project and connect it to Visual Studio
See what's new, or sign up for an account
Learning Resources
Visual Studio troubleshooting and support
Visual Studio videos on Channel 9
What is an MSDN subscription?
Solution Explorer
Team Explorer
Ready

Figure 9.2 – Microsoft Visual Studio Start Page



Windows Forms Application	Visual Basic
WPF Application	Visual Basic
Console Application	Visual Basic
Class Library	Visual Basic
Portable Class Library	Visual Basic
WPF Browser Application	Visual Basic
Empty Project	Visual Basic
Windows Service	Visual Basic
WPF Custom Control Library	Visual Basic
WPF User Control Library	Visual Basic
Windows Forms Control Library	Visual Basic

Type: Visual Basic

A project for creating an application with a Windows user interface

Name: Infusion Maintenance Program

Location: J:\wcc visual basic\Infusion Maintenance Program\

Solution name: Infusion Maintenance Program

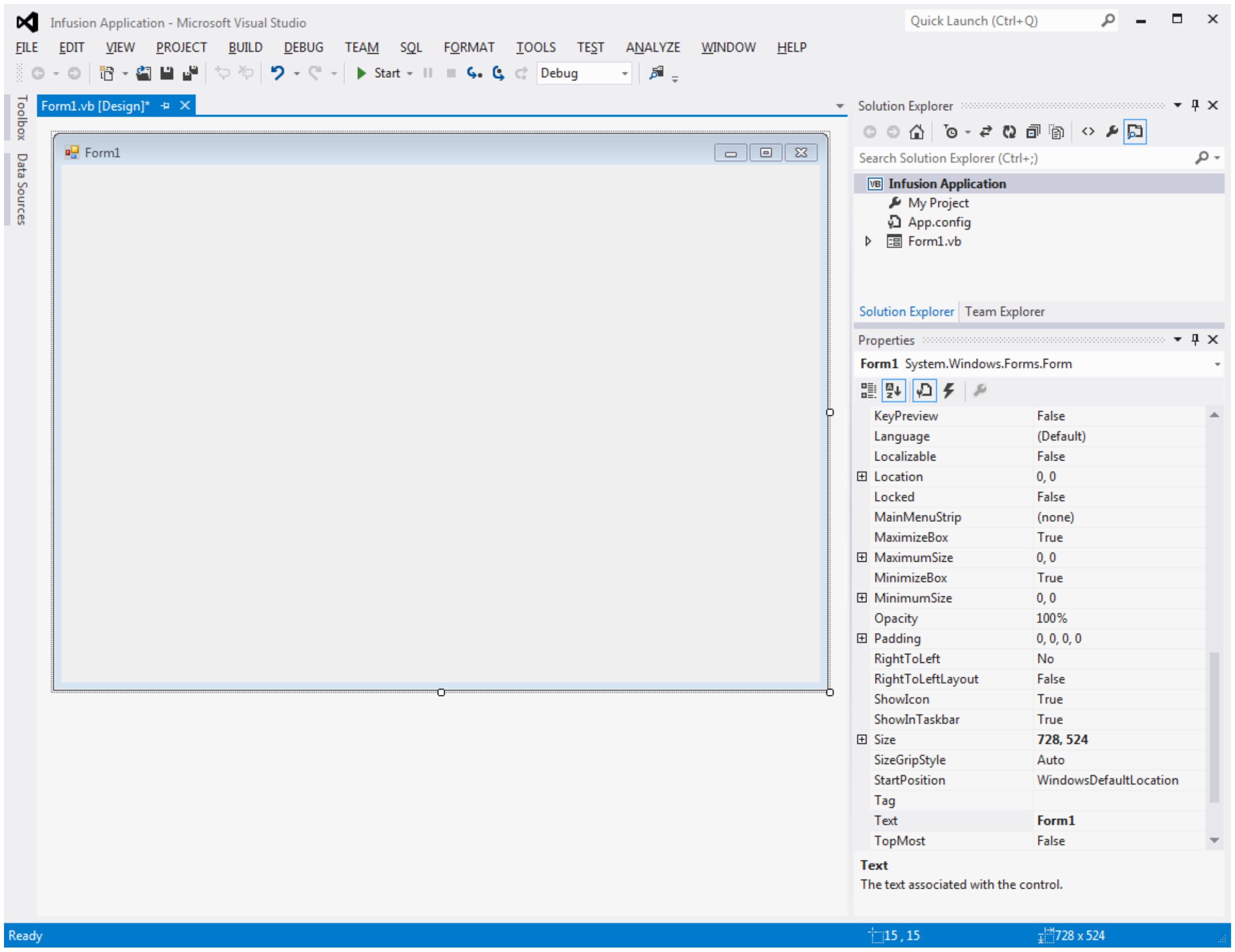
☑ Create directory for solution

☐ Add to source control

Figure 9.3 – New Project

We start a new Windows Application by picking the Windows Forms Application icon from the installed templates list on the New Project window.

With the Visual Basic Editor open, select **File** on the Menu Bar and select **Save All**. For the location, we will browse to the folder “Visual Basic Projects” that we made in previous chapters. We will name this project "Infusion Maintenance Application". A folder called "Infusion Maintenance Application" will be made and all the files for the program will be located in the folder. We press the OK button to begin.



Form1 System.Windows.Forms.Form

KeyPreview	False
Language	(Default)
Localizable	False
⊞ Location	0, 0
Locked	False
MainMenuStrip	(none)
MaximizeBox	True
⊞ MaximumSize	0, 0
MinimizeBox	True
⊞ MinimumSize	0, 0
Opacity	100%
⊞ Padding	0, 0, 0, 0
RightToLeft	No
RightToLeftLayout	False
ShowIcon	True
ShowInTaskbar	True
⊞ Size	**728, 524**
SizeGripStyle	Auto
StartPosition	WindowsDefaultLocation
Tag	
Text	**Form1**
TopMost	False

Text
The text associated with the control.

Figure 9.4 – Starting the Application

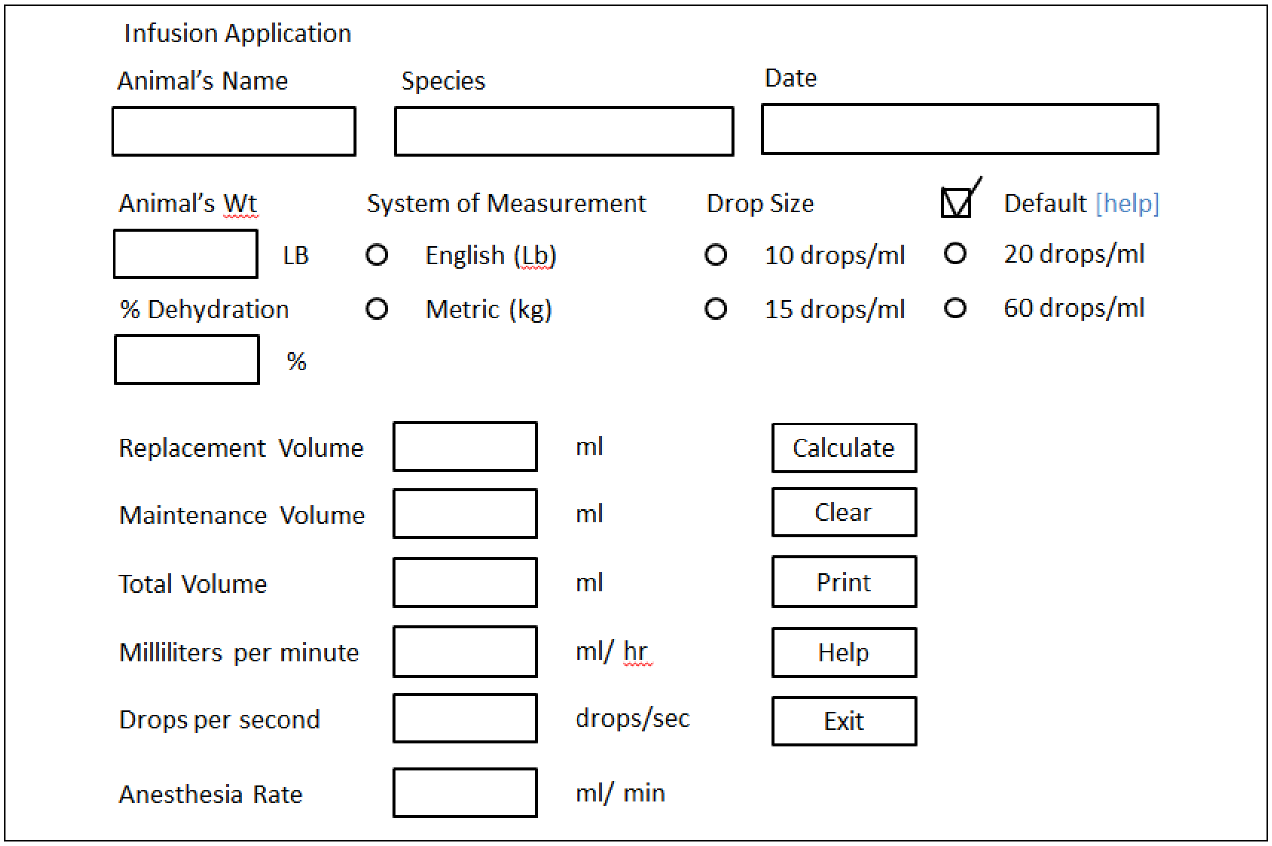
Beginning a New Visual Basic Application

Our project starts with a sketch where we ask for the animal's name, species name, weight and percent of dehydration and we place the data in textboxes. The animal's weight could be either in pounds or kilograms. The date and time will come from the computer system. This will be placed in a label. We will set the pounds as default but someone else could easily opt for the metric mass. The default drops will be 15 drops per milliliter for animals weighing 20 pounds or greater and will be 60 drops per milliliter for animals weighing 20 pounds or less.

When we calculate the answers, we will compute the replacement volume in milliliters, the maintenance volume in milliliters, and the total volume in milliliters. Then we will convert that volume given over a twenty four hour period to milliliters per hour. By using a condition statement for the 15 or 60 drops per milliliter, we will determine the drops per second. Finally, we will give the anesthesia rate in milliliters per minute based upon the animals weight and at 4 milliliters per pound and per hour.

We also found that professionals could pick tools to set the flow rate at 10 drops per milliliter and 60 drops per milliliter. So we made an option group that was controlled be the default checkbox.

We added a Clear button to wipe all of the input textboxes and output labels clean and a Print button to send the from to the printer so these calculations can be saved in a file. We added an Exit button to close the application.



Infusion Application

Animal's Name	Species	Date

Animal's Wt		System of Measurement		Drop Size		✓	Default [help]
	LB	○	English (Lb)	○	10 drops/ml	○	20 drops/ml
% Dehydration		○	Metric (kg)	○	15 drops/ml	○	60 drops/ml
	%						

Replacement Volume		ml	Calculate
Maintenance Volume		ml	Clear
Total Volume		ml	Print
Milliliters per minute		ml/ hr	Help
Drops per second		drops/sec	Exit
Anesthesia Rate		ml/ min	

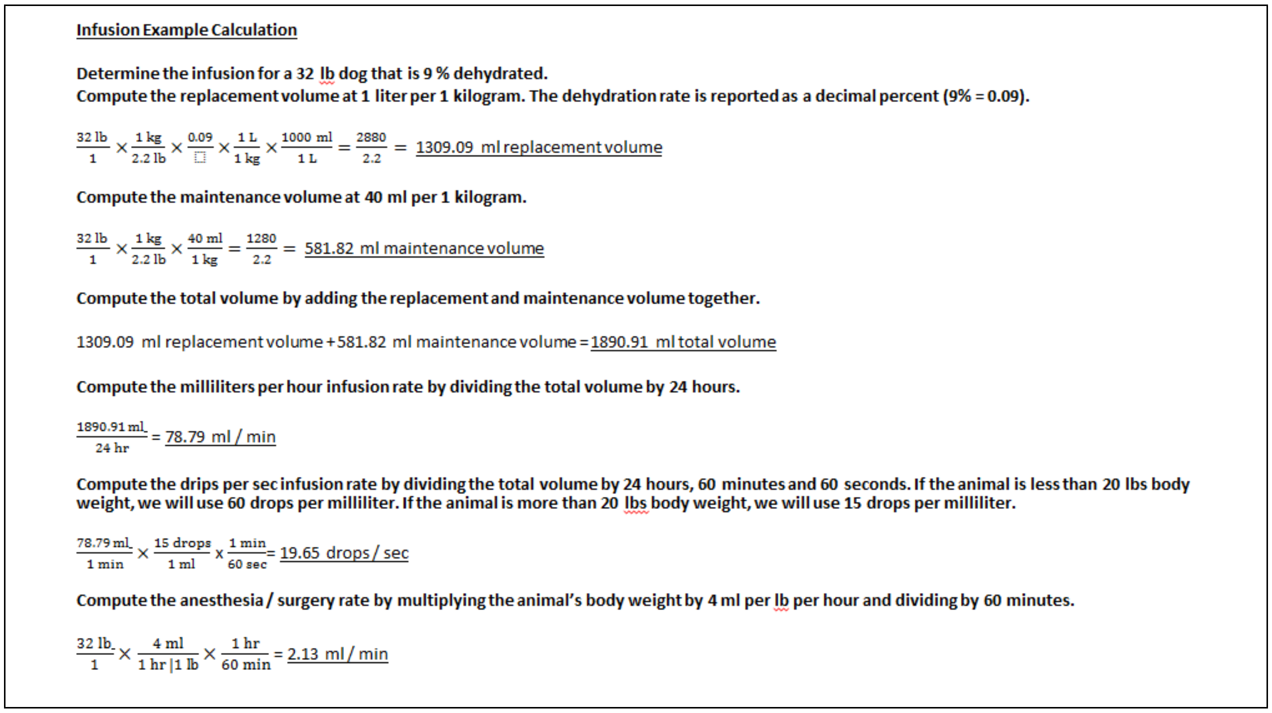
Figure 9.5 – Sketch of the Resistor Sizing Form

We have created two more diagrams showing the answers to the help hyperlink and the Help command button. The first one gives the explanation of when to uncheck the default checkbox.

Default Help for Drop Size
If the animal is less than or equal to 20 lbs body weight, we will use 60 drops per milliliter. If the animal is more than 20 lbs body weight, we will use 15 drops per milliliter. Uncheck the default box and any drop size can be chosen.

Figure 9.6 –Information for the Help Hyperlink

Another diagram will show an example of the infusion calculation.



Infusion Example Calculation

Determine the infusion for a 32 lb dog that is 9 % dehydrated.
Compute the replacement volume at 1 liter per 1 kilogram. The dehydration rate is reported as a decimal percent (9% = 0.09).

$$\frac{32\ \text{lb}}{1} \times \frac{1\ \text{kg}}{2.2\ \text{lb}} \times \frac{0.09}{\square} \times \frac{1\ \text{L}}{1\ \text{kg}} \times \frac{1000\ \text{ml}}{1\ \text{L}} = \frac{2880}{2.2} = \underline{1309.09\ \text{ml replacement volume}}$$

Compute the maintenance volume at 40 ml per 1 kilogram.

$$\frac{32\ \text{lb}}{1} \times \frac{1\ \text{kg}}{2.2\ \text{lb}} \times \frac{40\ \text{ml}}{1\ \text{kg}} = \frac{1280}{2.2} = \underline{581.82\ \text{ml maintenance volume}}$$

Compute the total volume by adding the replacement and maintenance volume together.

$1309.09\ \text{ml replacement volume} + 581.82\ \text{ml maintenance volume} = \underline{1890.91\ \text{ml total volume}}$

Compute the milliliters per hour infusion rate by dividing the total volume by 24 hours.

$$\frac{1890.91\ \text{ml}}{24\ \text{hr}} = \underline{78.79\ \text{ml / min}}$$

Compute the drips per sec infusion rate by dividing the total volume by 24 hours, 60 minutes and 60 seconds. If the animal is less than 20 lbs body weight, we will use 60 drops per milliliter. If the animal is more than 20 lbs body weight, we will use 15 drops per milliliter.

$$\frac{78.79\ \text{ml}}{1\ \text{min}} \times \frac{15\ \text{drops}}{1\ \text{ml}} \times \frac{1\ \text{min}}{60\ \text{sec}} = \underline{19.65\ \text{drops / sec}}$$

Compute the anesthesia / surgery rate by multiplying the animal's body weight by 4 ml per lb per hour and dividing by 60 minutes.

$$\frac{32\ \text{lb}}{1} \times \frac{4\ \text{ml}}{1\ \text{hr}\,|\,1\ \text{lb}} \times \frac{1\ \text{hr}}{60\ \text{min}} = \underline{2.13\ \text{ml / min}}$$

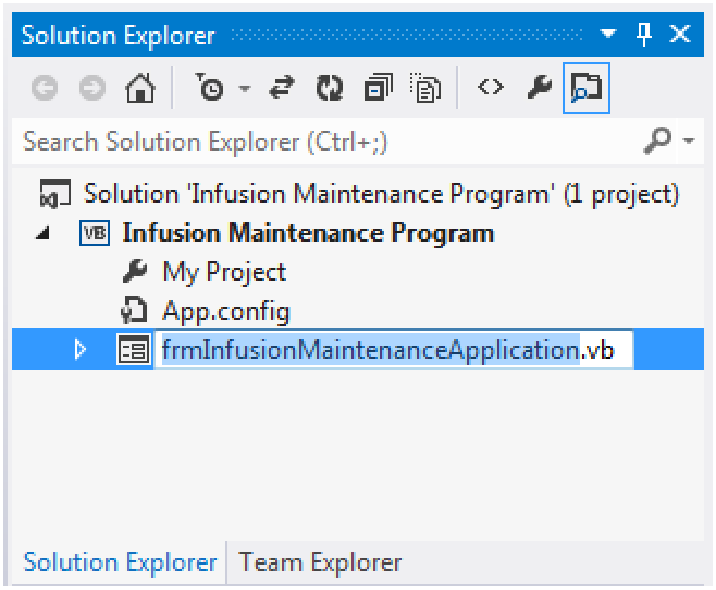
Figure 9.7 –Information for the Help Command Button

Laying Out a User Input Form in Visual Basic

We will change the **Text** in the Properties pane to Infusion Application to agree with the sketch in Figure 9.5. Go ahead and change the form in two other aspects, BackColor and Size.

Alphabetic	
(Name)	frmInfusionMaintenance Application
Size	575, 475
Text	Infusion Maintenance Application

The first number is the width and the second number is the height. The form will change in shape to the size measurement.



Solution Explorer

Search Solution Explorer (Ctrl+;)

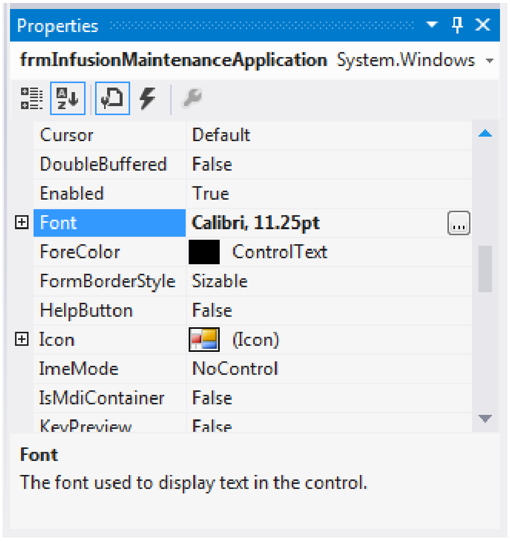
Solution 'Infusion Maintenance Program' (1 project)
- Infusion Maintenance Program
  - My Project
  - App.config
  - frmInfusionMaintenanceApplication.vb

Figure 9.8 – Setting the Form Properties

The background color is already white. There are many more attributes in the Properties pane that we will use on future projects.

In this project, we will select the font in the form. By selecting the font, font style and size for the form, each label, textbox and command button we insert will have these settings for their font.

When highlighting the row for Font, a small command button with three small dots appears to the right of the default font name of Microsoft San Serif. Click on the three dotted button to open the Visual Basic Font window.

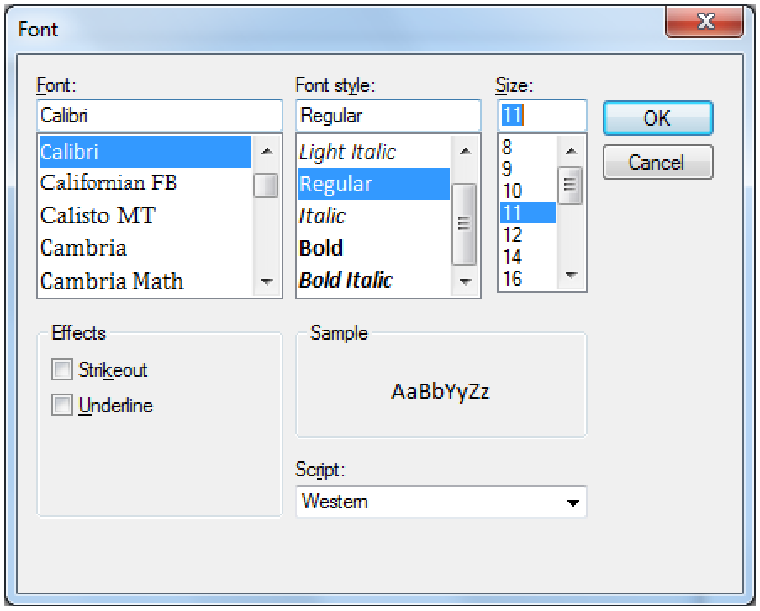


Cursor	Default
DoubleBuffered	False
Enabled	True
Font	Calibri, 11.25pt
ForeColor	ControlText
FormBorderStyle	Sizable
HelpButton	False
Icon	(Icon)
ImeMode	NoControl
IsMdiContainer	False
KeyPreview	False

Font
The font used to display text in the control.

Figure 9.9 – The Font Window in Visual Basic

We will select the Calibri font, Regular font style and 11 size for this project to agree with the initial sketch if the user input form. If we wish to underline the text or phrase in the label, add a check to the Underline checkbox in the Effects section of the Font window. When we finish making changes to the font property, select the OK command button to return to the work area.



Font: Calibri
Calibri
Californian FB
Calisto MT
Cambria
Cambria Math

Font style: Regular
Light Italic
Regular
Italic
Bold
Bold Italic

Size: 11
8
9
10
11
12
14
16

Strikeout
Underline

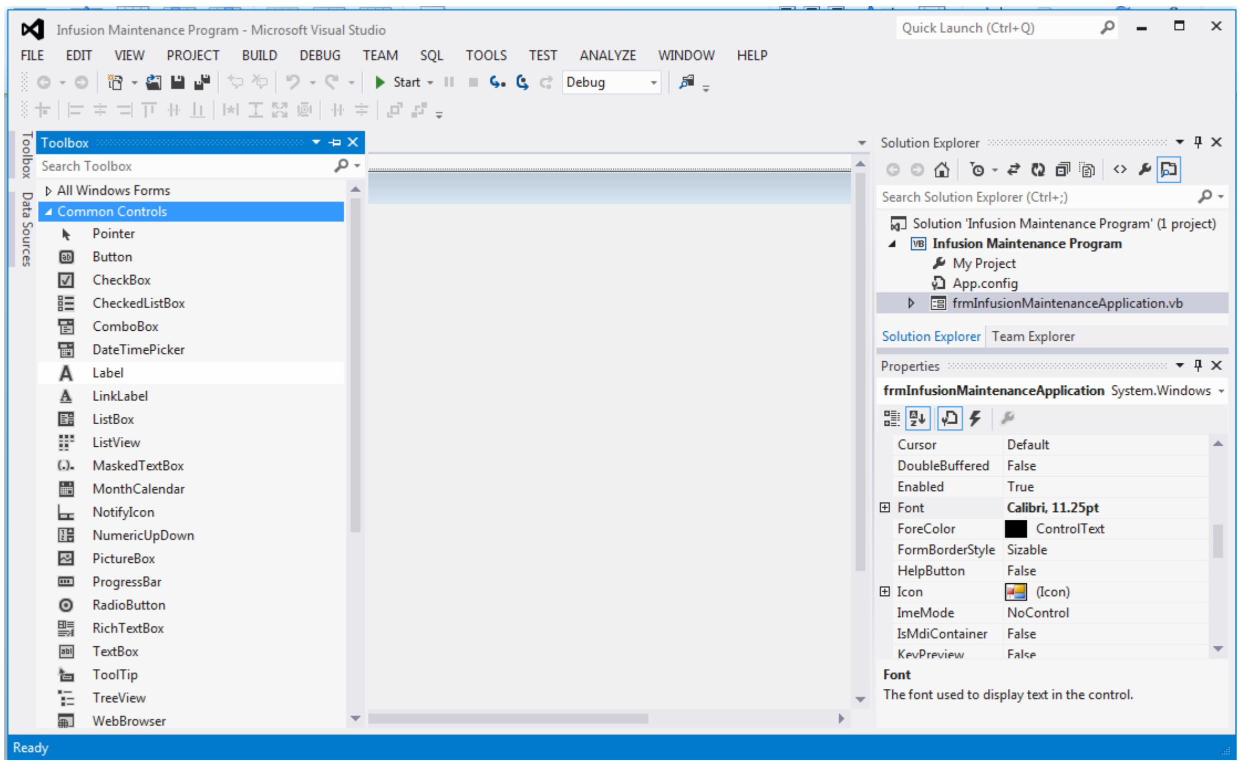
Script: Western

Figure 9.10 – Changing the Font to Calibri

Common Controls in Visual Basic

On the left side of the programming window, we can see the words Toolbox and Data Sources written vertically. We will choose Toolbox and we will expand our selection to see the Common Tools to use when making a user form.

The common tools we would use are labels, text boxes, checkboxes, picture box, radio buttons and buttons.



frmInfusionMaintenanceApplication System.Windows

Cursor	Default
DoubleBuffered	False
Enabled	True
Font	Calibri, 11.25pt
ForeColor	■ ControlText
FormBorderStyle	Sizable
HelpButton	False
Icon	(Icon)
ImeMode	NoControl
IsMdiContainer	False
KeyPreview	False

Font
The font used to display text in the control.

Figure 9.11 – Common Tools on the Toolbox

Inserting a Label into a Form

A good form is easy to figure out by the user, so when we are attempting to provide information on the window that will run in Windows; we add labels to textboxes to explain our intent. Press the Label (A) button on the Control Toolbar to add a label. To size the label area, click on the upper left area of the form and hold down on the left mouse button, draw the dotted label box.

When the first label is done, the background color of the label matches the background color of the form. In many cases that effect is visually pleasing to the eye, versus introducing another color. Both color and shape will direct the user in completing the form along with the explanation we place on the window to guide the designer in using the automated programs. Use colors and shape strategically to communicate well.

We will insert our first Label on the upper left corner of the form and call the entity **lblAnimalName**.

Alphabetic	
(Name)	lblAnimalName
Text	Animal's Name

Since the font is already set, we just type "Animal's Name" at the text attribute.

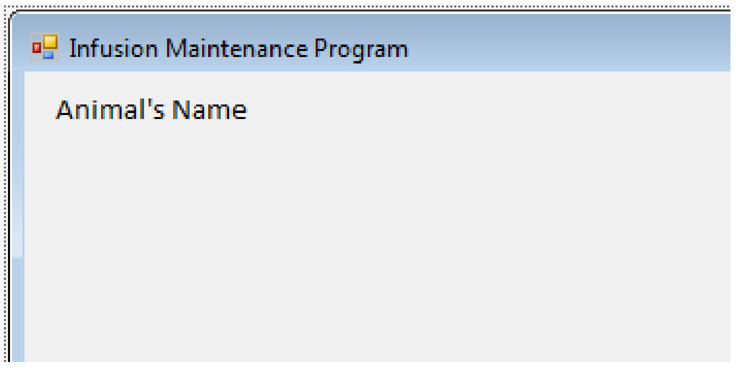


Figure 9.12 – The Finished Label on the Form

Inserting a Textbox into a Form

A textbox is used so that a user of the computer program can input data in the form of words, numbers or a mixture of both. Press the TextBox (ab) button on the Control Toolbar to add a textbox. To size the label area, click on the upper left area of the form and hold down on the left mouse button, draw the dotted textbox.

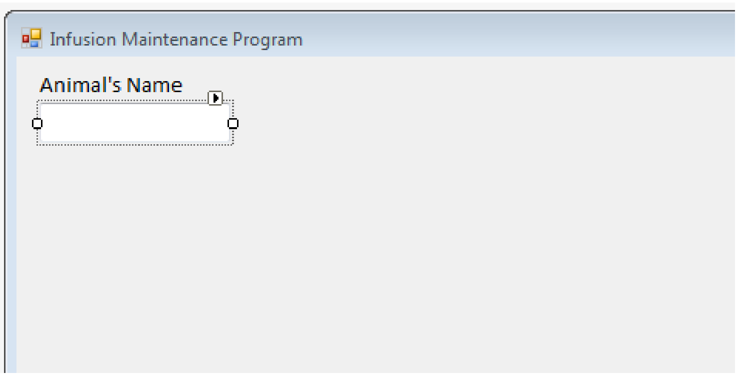
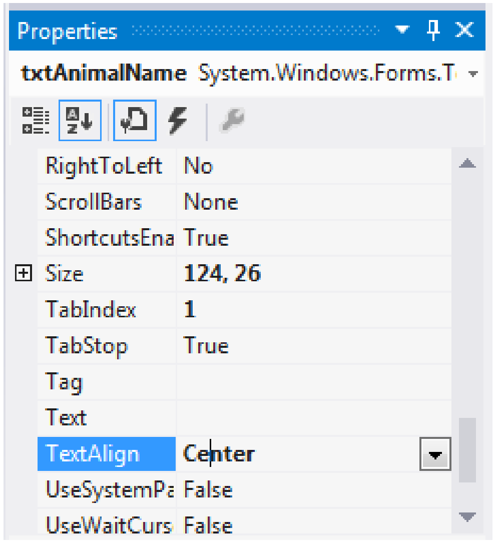


Figure 9.13 – Placing a TextBox on the Form

We will name the TextBox using the three letter prefix txt followed by the name or phrase of the tool. For our first textbox, the name is **txtAnimalName.**

Alphabetic	
(Name)	txtAnimalName
Size	124, 26
TextAlign	Center

The size of the textbox will be 124 wide and 26 tall and the characters inside the textbox will be center aligned.



txtAnimalName System.Windows.Forms.T

RightToLeft	No
ScrollBars	None
ShortcutsEna	True
Size	124, 26
TabIndex	1
TabStop	True
Tag	
Text	
TextAlign	Center
UseSystemPa	False
UseWaitCurs	False

Figure 9.14 – Setting the Size of the Textbox

We will insert three labels named **lblSpecies**, **lblSpecies** and **lblPercentDehydration** that display the names shown in Figure 9.15.. We will insert three more textboxes named **txtSpecies, txtAnimalWeight** and **txtPercentDeHydration** as shown in Figure 9.15.

We will modify the properties for the textboxes as shown below.

Alphabetic	
(Name)	txtSpecies
Size	124, 26
TextAlign	Center

Alphabetic	
(Name)	txtAnimalWeight
Size	124, 26
TextAlign	Center

Alphabetic	
(Name)	txtPercentDehydration
Size	124, 26
TextAlign	Center

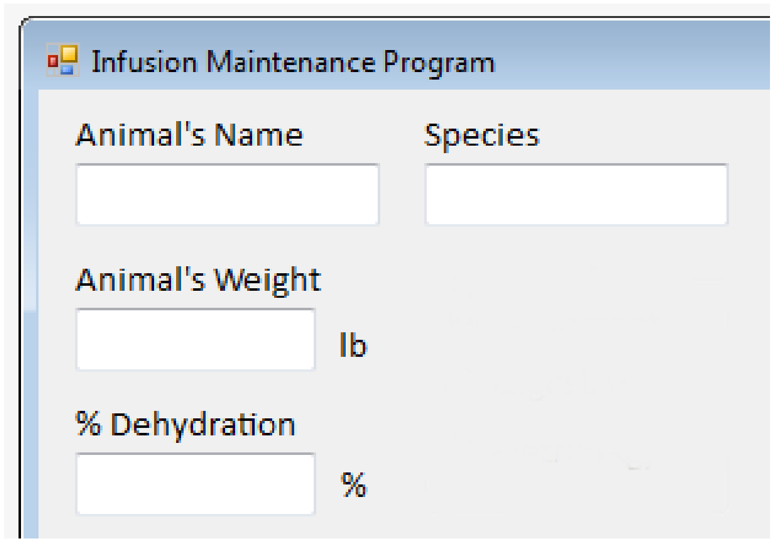


Figure 9.15 – Adding another Label

We will also add a label for pounds after the animal's weight textbox. We will eventually add a radio button group that will change the lb to kg if the users chooses the metric option.

Alphabetic	
(Name)	lblSystemOfMeasurementUnit
Text	lb

We will add a label for a percent sign after the percent dehydration textbox.

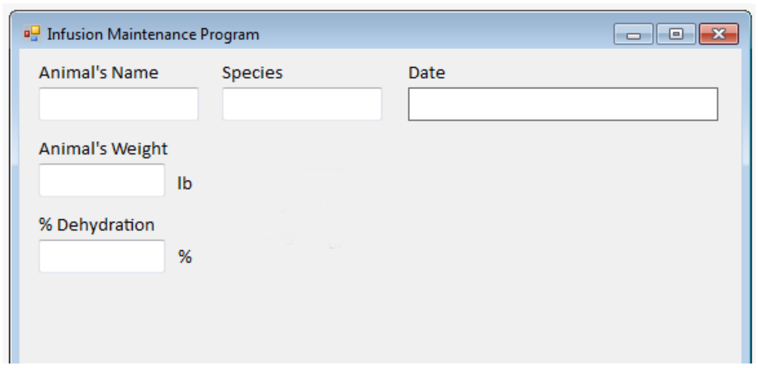
Alphabetic	
(Name)	lblPercentDehydrationUnit
Text	%

Inserting a Label into a Form to Post the Output

Some labels on a form are in a position to display an answer after the user inputs data and they press the command button to execute the application. To add this label, press the Label (A) button on the Control Toolbar to add a label. To size the label area, click on the upper left area of the form and hold down on the left mouse button, draw the dotted label box.

We will place a label to the right of the **lblSpecies** label and call it **lblDate.** We will make the label text Date. The key attributes for the label are:

Alphabetic	
(Name)	lblDate
Text	Date


Infusion Maintenance Program

Animal's Name

Species

Date

Animal's Weight

lb

% Dehydration

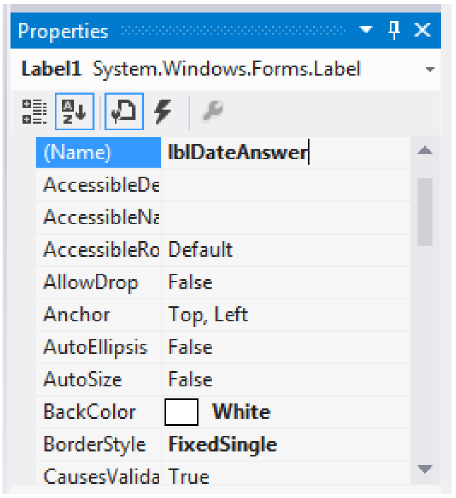
%

Figure 9.16 – Placing a Label for the Answer

We will insert the label for the answer to the bottom of **lblDate** label and name the label **lblDateAnswer.**

Alphabetic	
(Name)	lblSubnetMaskAnswer
BackColor	White
BorderStyle	FixedSingle
Size	240,26
TextAlign	Middle left

We will make the borderstyle FixedSingle to place a line around the answer.


Properties

Label1 System.Windows.Forms.Label

(Name)	lblDateAnswer
AccessibleDe	
AccessibleNa	
AccessibleRo	Default
AllowDrop	False
Anchor	Top, Left
AutoEllipsis	False
AutoSize	False
BackColor	⬜ White
BorderStyle	FixedSingle
CausesValida	True

Figure 9.17 – BackColor is White

Inserting a Group into a Form

When we want two radial or option buttons to toggle in an either or situation, then we will insert a group box into the form. In this case, we want to toggle between English (pounds) and Metric (kilograms). To add the GroupBox, choose the entity from the Container list on the Toolbox. To size the GroupBox area, click on the upper left area of the form and hold down on the left mouse button, draw the dotted Groupbox.

We will insert the groupbox for the system of measurement below the **txtSpecies** textbox and name the it **grpSystemMeasurement.**

Alphabetic	
(Name)	grpSystemMeasurement

The groupbox will enclose the English and Metric radio buttons.

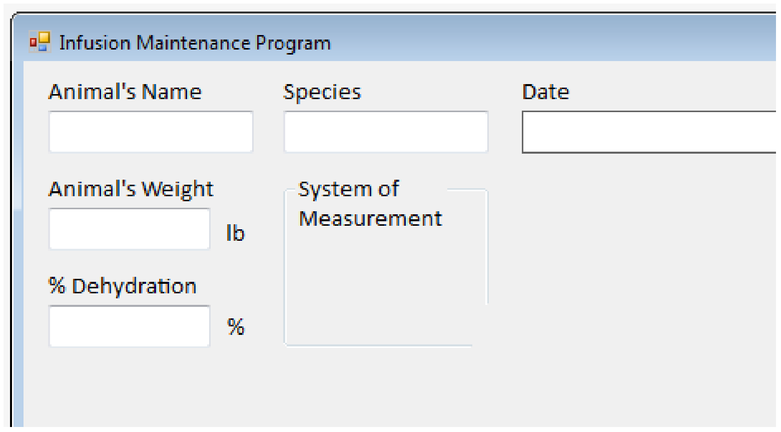


Figure 9.18 – GroupBox

Inserting a Radio Buttons into a Form

To add the first option button to place inside the GroupBox, choose the Radial Button from the list on the Toolbox. To size the Radial Button area, click inside the GroupBox and make a rectangle. We will name the entity **optEnglish**.

Alphabetic	
(Name)	optEnglish
Checked	True
Text	English (lb)

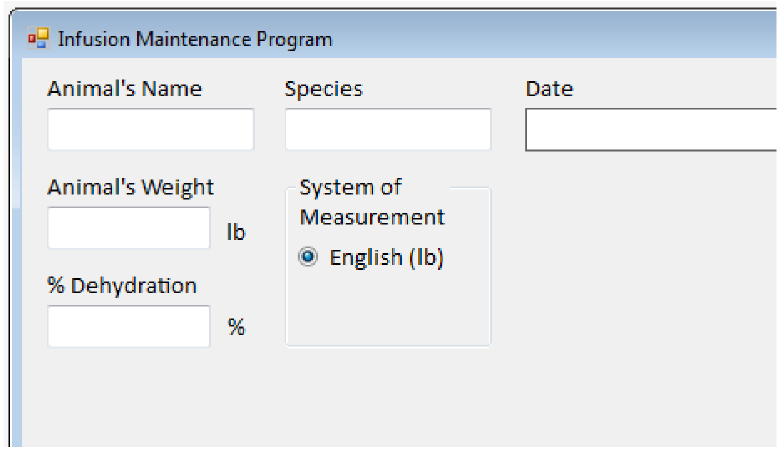


Figure 9.19 – English Radio Button

To add the second option button to place inside the GroupBox, choose the Radial Button from the list on the Toolbox. To size the Radial Button area, click inside the GroupBox and make a rectangle. We will name the entity **optMetric**.

Alphabetic	
(Name)	optMetric
Text	Metric (kg)

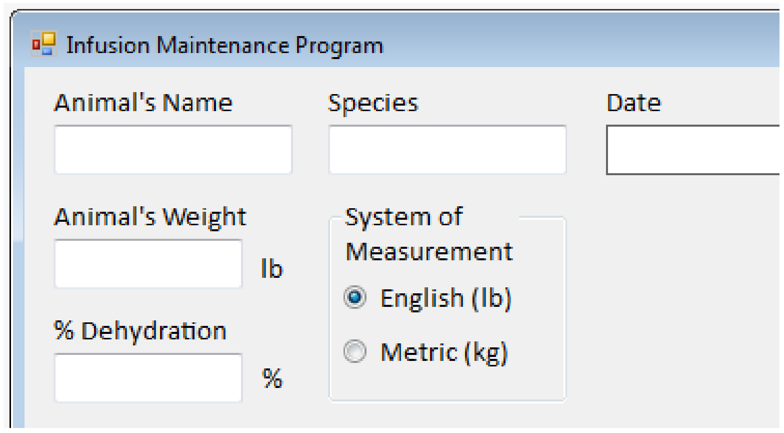


Figure 9.20 – Metric Radio Button

Inserting a Checkbox into a Form

In the next part of the form, we will insert a checkbox that will use the 15 drops per milliliter for animals weighing 20 pounds or more and the 60 drops per milliliter for animals weighing less than 20 pounds. When the default checkbox is annotated, we will make the drop size group invisible. Likewise, when the default box is unchecked, the four different size drops per milliliter can be selected.

We insert a checkbox and place it below the **lblDateAnswer** label. We will name the entity **chkDefault**.

Alphabetic	
(Name)	chkDefault
Checked	False
Text	Default

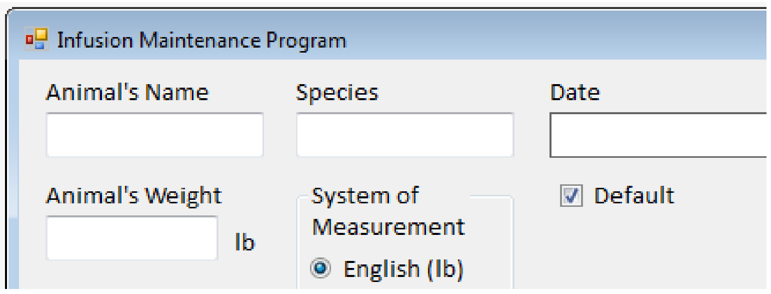


Figure 9.21 – Default Checkbox

Inserting another Groupbox and Radio Buttons into a Form

We will insert the groupbox for the drop size below the **lblDateAnswer** label and name the it **grpDropSize.**

Alphabetic	
(Name)	grpDropSize

Insert the four radio button in the Drop Size group.

Alphabetic	
(Name)	opt10DropsPerMilliliter
Text	10 drops / ml

Alphabetic	
(Name)	opt15DropsPerMilliliter
Text	15 drops / ml

Alphabetic	
(Name)	opt20DropsPerMilliliter
Text	20 drops / ml

Alphabetic	
(Name)	Opt60DropsPerMilliliter
Text	60 drops / ml

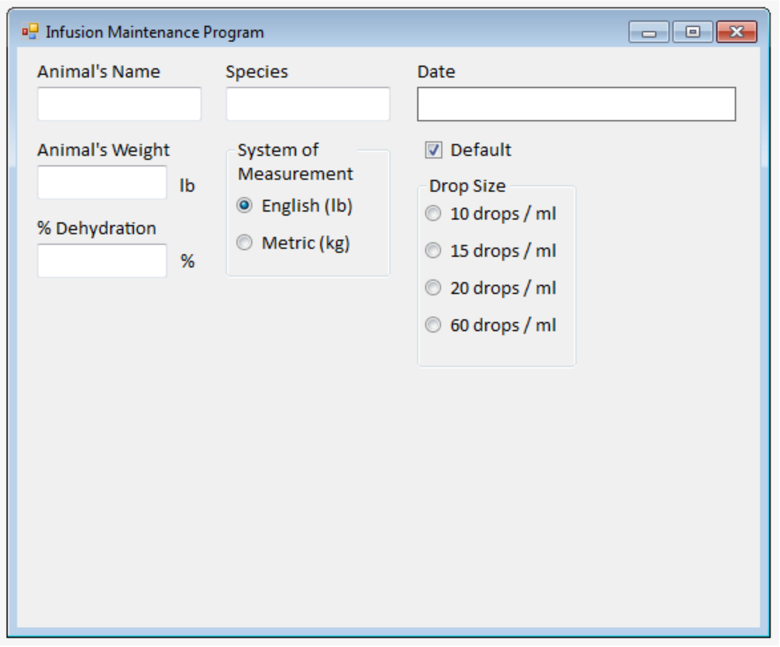


Figure 9.22 – Drop Size Group and Radio Buttons

Inserting More Labels into a Form to Post the Output

There are six calculations that the Vet makes to determine the amount of fluid to give a dehydrated animal and the rate which it should flow. These are replacement volume, maintenance volume, total volume, milliliters per hour, drops per second and the anesthesia rate. We will use labels to report the information, so for each one there are three common controls to name. We use variations of the first label name on the second one in the row by adding "answer" to the end and on the third label we concatenate "unit". It is always best to control every entity on the form so we can change properties on the control if needed.

Here are the definitions of each label.

Alphabetic	1st Label	2nd Label	3rd Label
(Name)	lblReplacementVolume	lblReplacementVolumeAnswer	lblReplacementVolumeUnit
BackColor	Control	White	Control
BorderStyle	None	FixedSingle	None
Size	141,18	94,26	24,18
Text	Replacement Volume	None	ml
TextAlign	Middle left	Middle left	Middle left

Alphabetic	1st Label	2nd Label	3rd Label
(Name)	lblMaintenanceVolume	lblMaintenanceVolumeAnswer	lblMaintenanceVolumeUnit
BackColor	Control	White	Control
BorderStyle	None	FixedSingle	None
Size	141,18	94,26	24,18
Text	Maintenance Volume	None	ml
TextAlign	Middle left	Middle left	Middle left

Alphabetic	1st Label	2nd Label	3rd Label
(Name)	lblTotalVolume	lblTotalVolumeAnswer	lblTotalVolumeUnit
BackColor	Control	White	Control
BorderStyle	None	FixedSingle	None
Size	141,18	94,26	24,18
Text	Total Volume	None	ml
TextAlign	Middle left	Middle left	Middle left

Alphabetic	1st Label	2nd Label	3rd Label
(Name)	lblMillilitersPerHour	lblMillilitersPerHourAnswer	lblMillilitersPerHourUnit
BackColor	Control	White	Control
BorderStyle	None	FixedSingle	None
Size	141,18	94,26	24,18
Text	Milliliters per Hour	None	ml / hr
TextAlign	Middle left	Middle left	Middle left

Alphabetic	1st Label	2nd Label	3rd Label
(Name)	lblDropsPerSecond	lblDropsPerSecondAnswer	lblDropsPerSecondUnit
BackColor	Control	White	Control
BorderStyle	None	FixedSingle	None
Size	141,18	94,26	24,18
Text	Drops per Second	None	Drops /sec
TextAlign	Middle left	Middle left	Middle left

Alphabetic	1st Label	2nd Label	3rd Label
(Name)	lblAnesthesiaRate	lblAnesthesiaRateAnswer	lblAnesthesiaRateUnit
BackColor	Control	White	Control
BorderStyle	None	FixedSingle	None
Size	141,18	94,26	24,18
Text	Anesthesia Rate	None	ml / min
TextAlign	Middle left	Middle left	Middle left

The Infusion Maintenance form is starting to take shape. We now need to add command buttons.

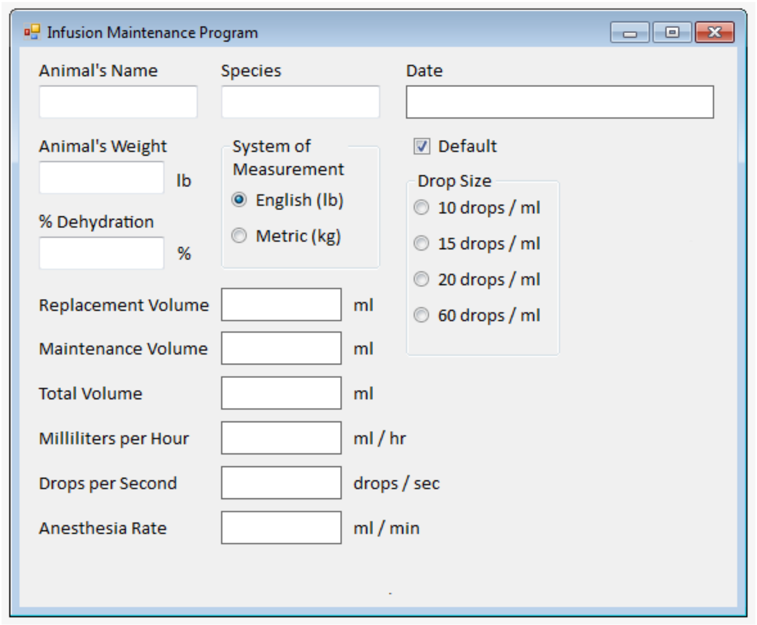


Figure 9.23 – Program Outputs

Inserting a Command Buttons into a Form

A command button is used so that a user will execute the application. Press the Button on the Common Control menu to add a command button. To size the label area, click on the upper left area of the form and hold down on the left mouse button, draw the command button as shown in Figure 9.5.

We will name the command button using the name is **cmdCalculate.**

Alphabetic	
(Name)	cmdCalculate
Caption	Calculate
Size	104,30

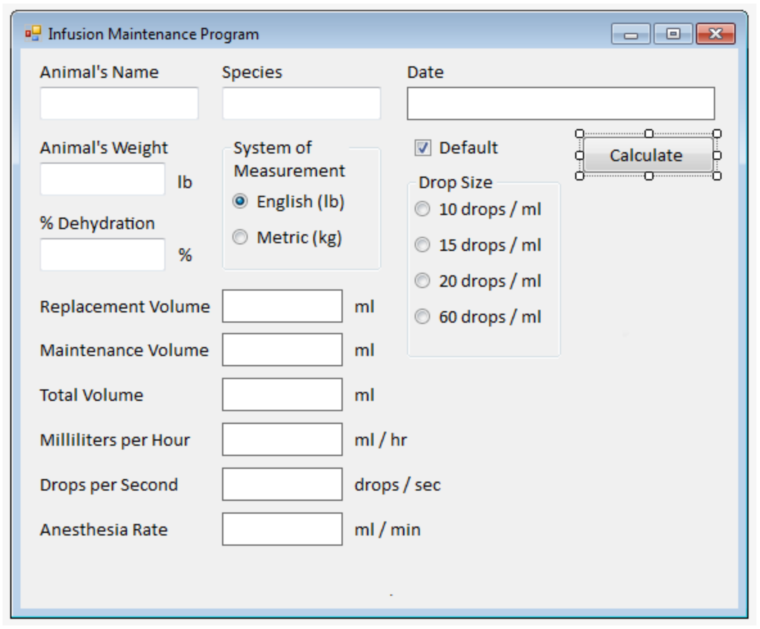


Figure 9.24 – The cmdCalculate Button

Add four more command buttons, named **cmdClear**, **cmdPrint**, **cmdHelp** and **cmdExit**. The Clear button will clear text boxes, output labels, option button and checkboxes. The Print button will send the form to the printer. The Help button will open a message box that will show a complete sample calculation with a full explanation. The Exit button will close the program. Notice the equal spacing between the command buttons gives a visually friendly appearance.

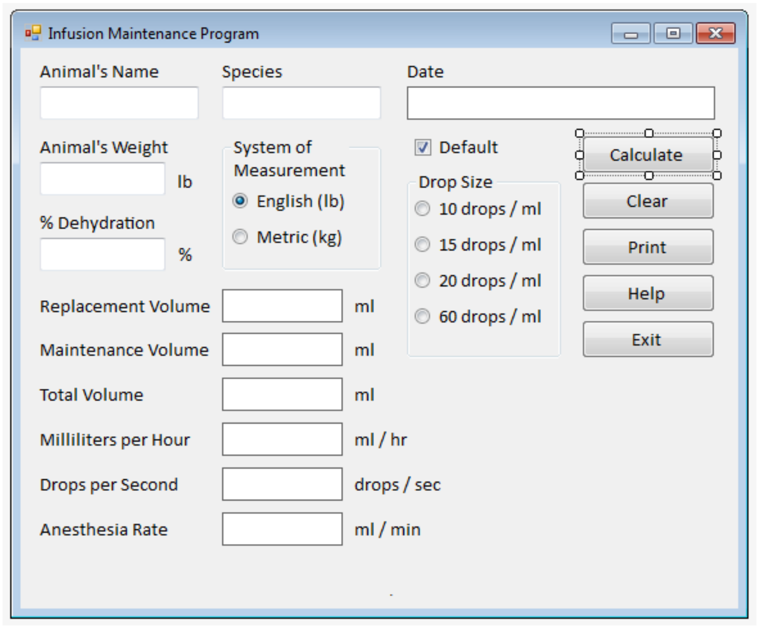


Figure 9.25 – Insert Four More Command Buttons

We will add another label that will launch a message box about the default drop size.

Alphabetic	
(Name)	lblHelp
ForeColor	Blue
Text	Help

Although this is a label, it will act like a button or hyperlink when the user presses on it.

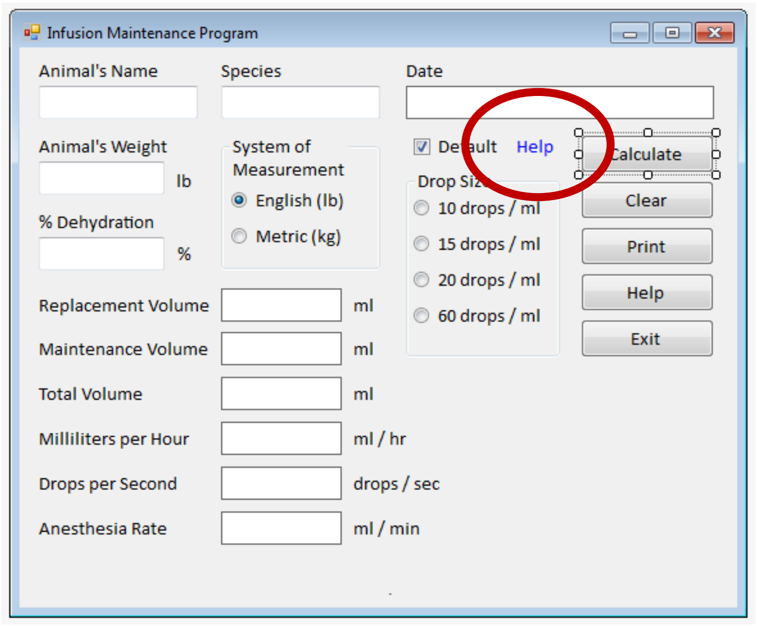


Figure 9.26 – Help Hyperlink

Adding a Picture to a Form

We select the Common toolbox and Picturebox and we draw a box to the bottom of the buttons. We name the picturebox **imgIV**. We scroll down on the properties window and select the three dots button at the Image property and a Select Resource window will appear. We then will import the graphic of our IV bag which we have in Microsoft Paint and saved as a bitmap image. We then press the OK button and the image will appear in the picture box.

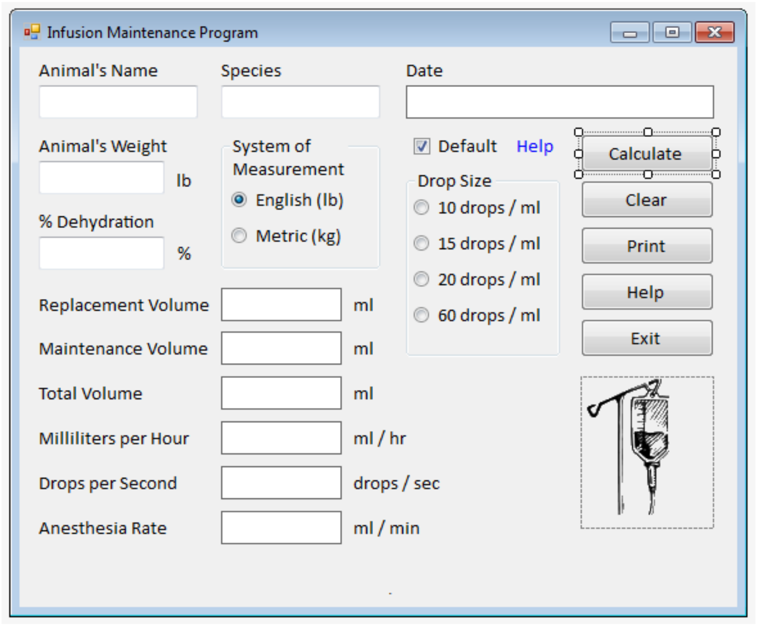


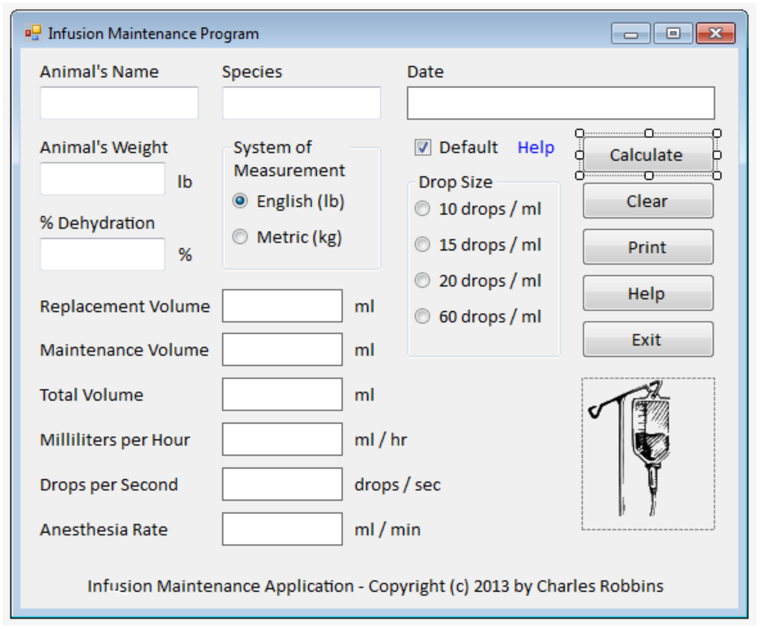
Figure 9.27 – IV Image

Adding a Copyright Statement to a Form

At the beginning of a new program, we will expect to see an explanation or any special instructions in the form of comments such as copyright, permissions or other legal notices to inform programmers what are the rules dealing with running the code. Comments at the opening of the code could help an individual determine whether the program is right for their application or is legal to use. The message box is a great tool when properly utilized to inform someone if they are breaking a copyright law when running the code.

We insert a label called lblCopyright and we type this in for text.

Infusion Maintenance Application - Copyright (c) 2013 by Charles Robbins



Infusion Maintenance Application - Copyright (c) 2013 by Charles Robbins

Figure 9.28 – Adding a Copyright Statement

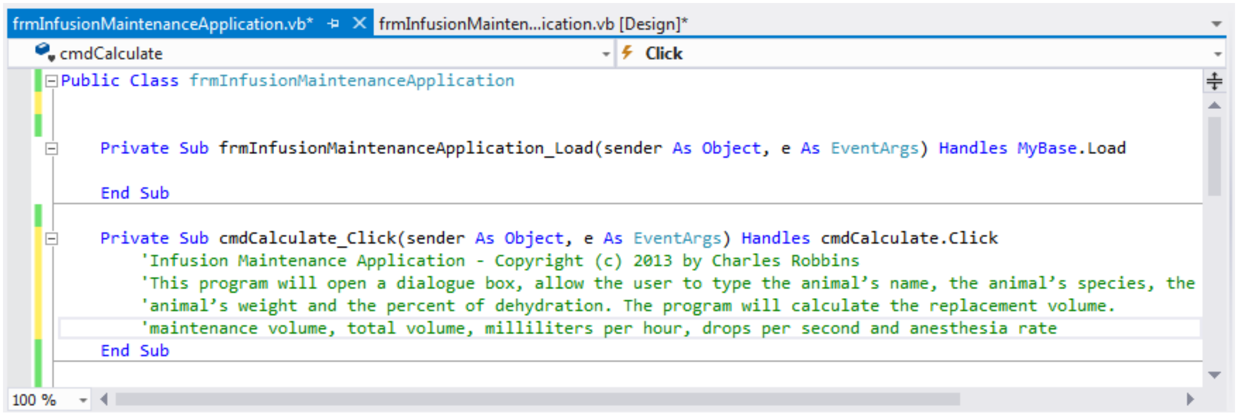
Adding Comments in Visual Basic to Communicate the Copyright

The comments we placed in the first four lines of the program will inform the individual opening and reading the code, but those user that may run the application without checking, the label on the bottom of the form with the copyright information is a great tool to alert the client to the rules of the program and what will the application do.

To begin the actual coding of the program, double click on the Calculate command button. At the top of the program and before the line of code with Private Sub cmdCalculate_Click (), place the following comments with the single quote (‘) character. Remember, the single quote character (‘) will precede a comment and when the code is compiled, comments are ignored.

Type the following line of code:

```
'Infusion Maintenance Application - Copyright (c) 2013 by Charles Robbins
'This program will open a dialogue box, allow the user to type the animal's name, the animal's species, the
'animal's weight and the percent of dehydration. The program will calculate the replacement volume.
'maintenance volume, total volume, milliliters per hour, drops per second and anesthesia rate
```


frmInfusionMaintenanceApplication.vb*
frmInfusionMainten...ication.vb [Design]*
cmdCalculate
Click
100 %

```
Public Class frmInfusionMaintenanceApplication


    Private Sub frmInfusionMaintenanceApplication_Load(sender As Object, e As EventArgs) Handles MyBase.Load

    End Sub

    Private Sub cmdCalculate_Click(sender As Object, e As EventArgs) Handles cmdCalculate.Click
        'Infusion Maintenance Application - Copyright (c) 2013 by Charles Robbins
        'This program will open a dialogue box, allow the user to type the animal's name, the animal's species, the
        'animal's weight and the percent of dehydration. The program will calculate the replacement volume.
        'maintenance volume, total volume, milliliters per hour, drops per second and anesthesia rate
    End Sub
```

Figure 9.29 – Adding a Copyright Statement

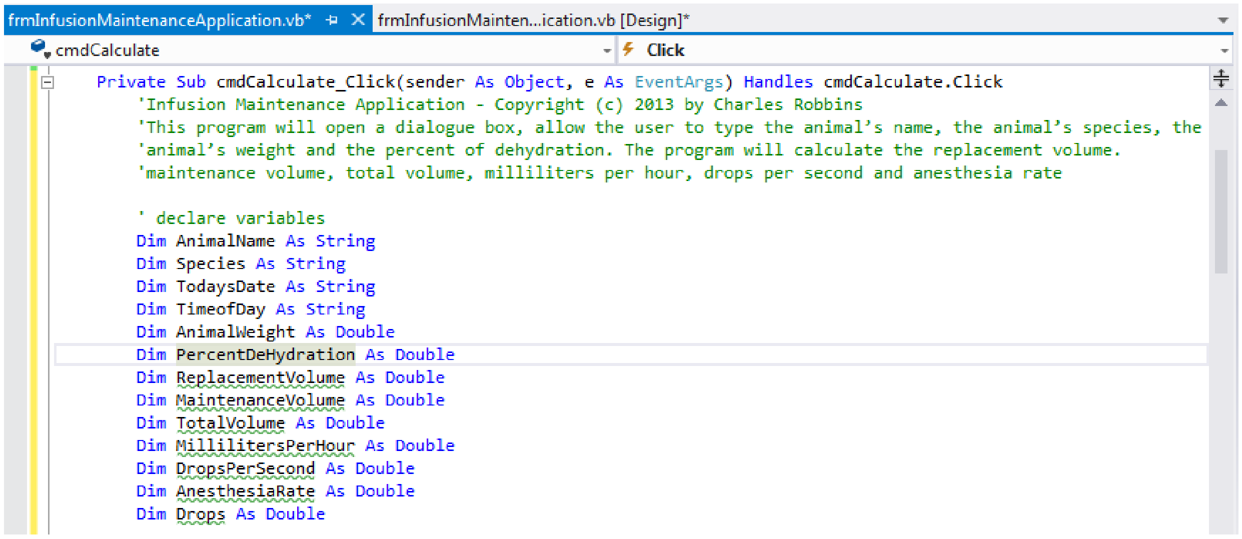
Declaring Variables in a Program with the Dimension Statement

When we are going to use a number, text string or object that may change throughout the life of the code, we create a variable to hold the value of that changing entity. In Visual Basic, the dimension statement is one of the ways to declare a variable at the procedure level. The other two ways are the Private and Public statements.

In this program, we will declare four variables as strings because they will hold names or data that represent days and time. The other nine variables will be double integers since they will be used in calculations.

Type the following code under the cmdCalculate subroutine of the program.

```
' declare variables
Dim AnimalName As String
Dim Species As String
Dim TodaysDate As String
Dim TimeofDay As String
Dim AnimalWeight As Double
Dim PercentDeHydration As Double
Dim ReplacementVolume As Double
Dim MaintenanceVolume As Double
Dim TotalVolume As Double
Dim MillilitersPerHour As Double
Dim DropsPerSecond As Double
Dim AnesthesiaRate As Double
Dim Drops As Double
```


frmInfusionMaintenanceApplication.vb*
frmInfusionMainten...ication.vb [Design]*
cmdCalculate
Click

```
    Private Sub cmdCalculate_Click(sender As Object, e As EventArgs) Handles cmdCalculate.Click
        'Infusion Maintenance Application - Copyright (c) 2013 by Charles Robbins
        'This program will open a dialogue box, allow the user to type the animal's name, the animal's species, the
        'animal's weight and the percent of dehydration. The program will calculate the replacement volume.
        'maintenance volume, total volume, milliliters per hour, drops per second and anesthesia rate

        ' declare variables
        Dim AnimalName As String
        Dim Species As String
        Dim TodaysDate As String
        Dim TimeofDay As String
        Dim AnimalWeight As Double
        Dim PercentDeHydration As Double
        Dim ReplacementVolume As Double
        Dim MaintenanceVolume As Double
        Dim TotalVolume As Double
        Dim MillilitersPerHour As Double
        Dim DropsPerSecond As Double
        Dim AnesthesiaRate As Double
        Dim Drops As Double
```

Figure 9.30 – Declaring Variables with Dim Statements

Notice that the variable name should be a word or a phrase without spaces that represents the value that the variable contains. If we want to hold a value of one's date of birth, we can call the variable, DateofBirth. The keywords Date and Birth are in sentence case with the first letter capitalized. There are no spaces in the name. Some programmers use the underscore character (_) to separate words in phrases. This is acceptable, but a double underscore (__) can cause errors if we do not detect the repeated character.

Setting Variables in a Program

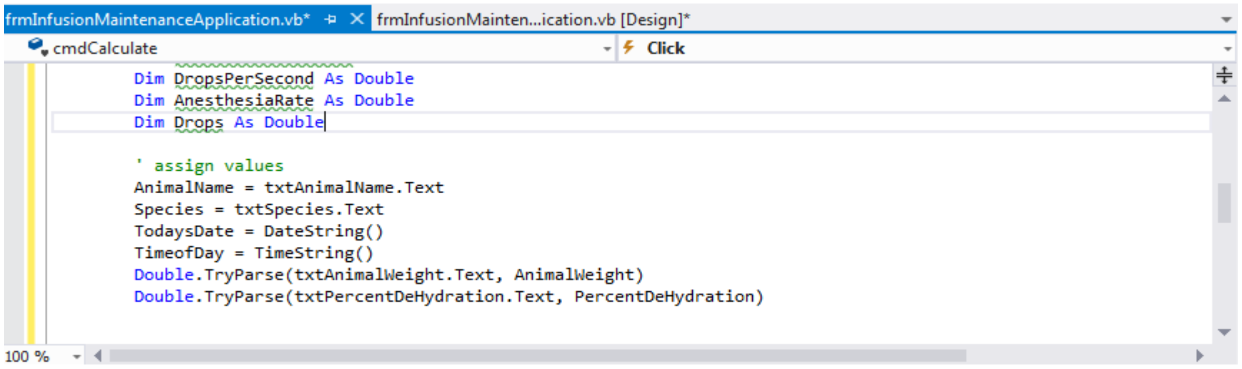
Next, we will set the variables using the equal and the TryParse function. We will set the animal name and species variables using the equal sign. Both of these entities are strinsg so we do not need to convert them. TodaysDate will equal the DateString on the calendar in the personal computer that is running the program. TimeofDay will equal the TimeString on the clock on the same personal computer.

The next expressions that capture the animal's weight and percent of dehydration use the Double.TryParse function. In this code, we type the name of the textbox and the text field that holds the data followed by the name of the variable that will hold the number.

Type the following code under the "set variable" section of the cmdCalculate subroutine of the program.

```
' assign values

AnimalName = txtAnimalName.Text
Species = txtSpecies.Text
TodaysDate = DateString()
TimeofDay = TimeString()
```

```
Double.TryParse(txtAnimalWeight.Text, AnimalWeight)
Double.TryParse(txtPercentDehydration.Text, PercentDeHydration)
```


frmInfusionMaintenanceApplication.vb*
frmInfusionMainten...ication.vb [Design]*
cmdCalculate
Click
100 %

```
        Dim DropsPerSecond As Double
        Dim AnesthesiaRate As Double
        Dim Drops As Double

        ' assign values
        AnimalName = txtAnimalName.Text
        Species = txtSpecies.Text
        TodaysDate = DateString()
        TimeofDay = TimeString()
        Double.TryParse(txtAnimalWeight.Text, AnimalWeight)
        Double.TryParse(txtPercentDeHydration.Text, PercentDeHydration)
```

Figure 9.31 – Setting the Variables in the VBA Code

Compute the Answers

Our calculations will be based on a premise that the animal's weight will be in kilograms, so we will write a condition statement that asks if the English radio button is checked then we will divide the animal's weight by 2.2 (there are 2.2 pounds in a kilogram). Else if the Metric radio button is checked then we state that the animal's weight will stay the same.

After that we will calculate the replacement volume as the animal weight times the percent of dehydration divided by one hundred and then times one thousand milliliters.

Then we will compute the maintenance volume as the animal weight times forty milliliters and the total volume as the sum of the replacement volume and the replacement volume. We next find the milliliters per hour as the total volume divided by 24 hours.

We will write another condition statement that asks if the animal is less or equal to 9.09008 kilogram then the drops will be 60 or else it will be 15.

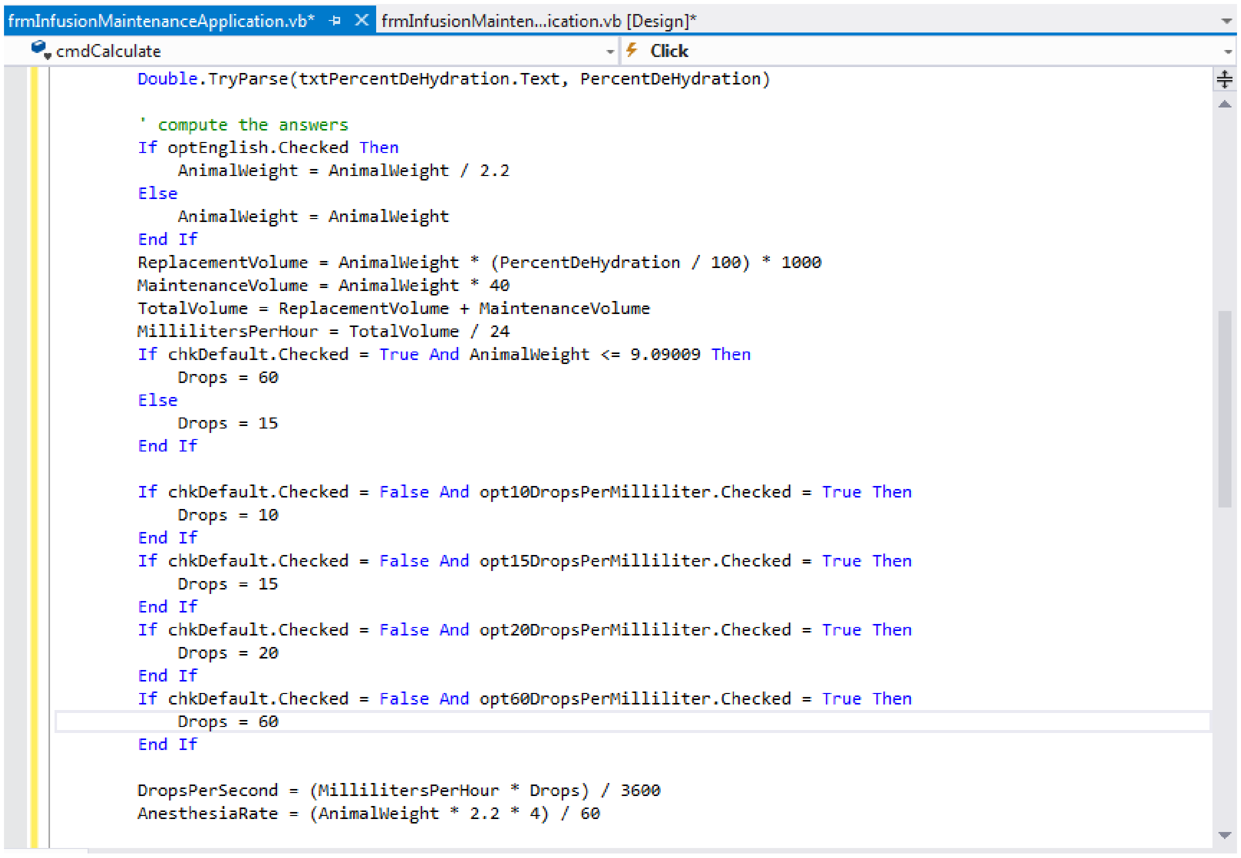
```
' compute the answers
If optEnglish.Checked Then
   AnimalWeight = AnimalWeight / 2.2
Else
   AnimalWeight = AnimalWeight
End If
ReplacementVolume = AnimalWeight * (PercentDeHydration / 100) * 1000
MaintenanceVolume = AnimalWeight * 40
TotalVolume = ReplacementVolume + MaintenanceVolume
MillilitersPerHour = TotalVolume / 24
If chkDefault.Checked = True And AnimalWeight <= 9.09009 Then
   Drops = 60
Else
   Drops = 15
End If
```

The following four condition statements asks if the default checkbox is annotated and if one of the radio button is checked and then sets the drops to 10, 15, 20 or 60.

```
If chkDefault.Checked = False And opt10DropsPerMilliliter.Checked = True Then
    Drops = 10
End If
If chkDefault.Checked = False And opt15DropsPerMilliliter.Checked = True Then
    Drops = 15
End If
If chkDefault.Checked = False And opt20DropsPerMilliliter.Checked = True Then
    Drops = 20
End If
If chkDefault.Checked = False And opt60DropsPerMilliliter.Checked = True Then
    Drops = 60
End If
```

The subsequent calculation of drops per second is milliliters per hour times drops divided by 3600. We will also determine the anesthesia as the animal weight times 2.2 pounds times 4 milliliters per hour per pound and divided by 60 minutes per hour.

```
DropsPerSecond = (MillilitersPerHour * Drops) / 3600
AnesthesiaRate = (AnimalWeight * 2.2 * 4) / 60
```


frmInfusionMaintenanceApplication.vb* frmInfusionMainten...ication.vb [Design]*
cmdCalculate Click

```
        Double.TryParse(txtPercentDeHydration.Text, PercentDeHydration)

        ' compute the answers
        If optEnglish.Checked Then
            AnimalWeight = AnimalWeight / 2.2
        Else
            AnimalWeight = AnimalWeight
        End If
        ReplacementVolume = AnimalWeight * (PercentDeHydration / 100) * 1000
        MaintenanceVolume = AnimalWeight * 40
        TotalVolume = ReplacementVolume + MaintenanceVolume
        MillilitersPerHour = TotalVolume / 24
        If chkDefault.Checked = True And AnimalWeight <= 9.09009 Then
            Drops = 60
        Else
            Drops = 15
        End If

        If chkDefault.Checked = False And opt10DropsPerMilliliter.Checked = True Then
            Drops = 10
        End If
        If chkDefault.Checked = False And opt15DropsPerMilliliter.Checked = True Then
            Drops = 15
        End If
        If chkDefault.Checked = False And opt20DropsPerMilliliter.Checked = True Then
            Drops = 20
        End If
        If chkDefault.Checked = False And opt60DropsPerMilliliter.Checked = True Then
            Drops = 60
        End If

        DropsPerSecond = (MillilitersPerHour * Drops) / 3600
        AnesthesiaRate = (AnimalWeight * 2.2 * 4) / 60
```

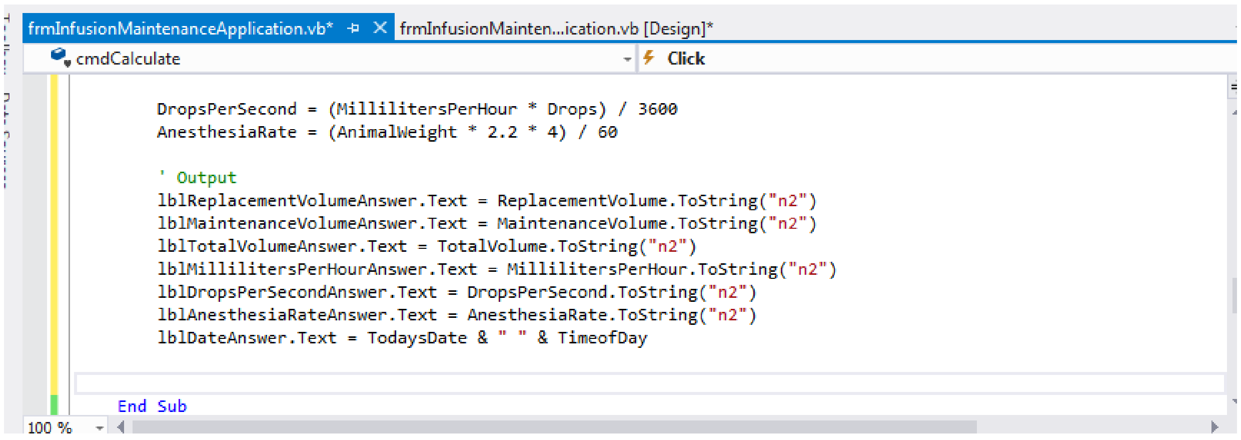
Figure 9.32 – Computing the Answers

Output the Data

The next section of code is the expressions that will take the six calculations and the computer date and time and send that information to seven labels. For the six double integers, we will use the “ToString” property to convert the double integer to string text and we will set the answer to a two decimal number using “n2”.

For the lblDateAnswer.text we will concatenate the variable called TodaysDate and a space and the variable TimeofDay.

```
' Output
lblReplacementVolumeAnswer.Text = ReplacementVolume.ToString("n2")
lblMaintenanceVolumeAnswer.Text = MaintenanceVolume.ToString("n2")
lblTotalVolumeAnswer.Text = TotalVolume.ToString("n2")
lblMillilitersPerHourAnswer.Text = MillilitersPerHour.ToString("n2")
lblDropsPerSecondAnswer.Text = DropsPerSecond.ToString("n2")
lblAnesthesiaRateAnswer.Text = AnesthesiaRate.ToString("n2")
lblDateAnswer.Text = TodaysDate & " " & TimeofDay
```


frmInfusionMaintenanceApplication.vb*
frmInfusionMainten...ication.vb [Design]*
cmdCalculate
Click
100 %

```
        DropsPerSecond = (MillilitersPerHour * Drops) / 3600
        AnesthesiaRate = (AnimalWeight * 2.2 * 4) / 60

        ' Output
        lblReplacementVolumeAnswer.Text = ReplacementVolume.ToString("n2")
        lblMaintenanceVolumeAnswer.Text = MaintenanceVolume.ToString("n2")
        lblTotalVolumeAnswer.Text = TotalVolume.ToString("n2")
        lblMillilitersPerHourAnswer.Text = MillilitersPerHour.ToString("n2")
        lblDropsPerSecondAnswer.Text = DropsPerSecond.ToString("n2")
        lblAnesthesiaRateAnswer.Text = AnesthesiaRate.ToString("n2")
        lblDateAnswer.Text = TodaysDate & " " & TimeofDay

    End Sub
```

Figure 9.33 – Output the Data

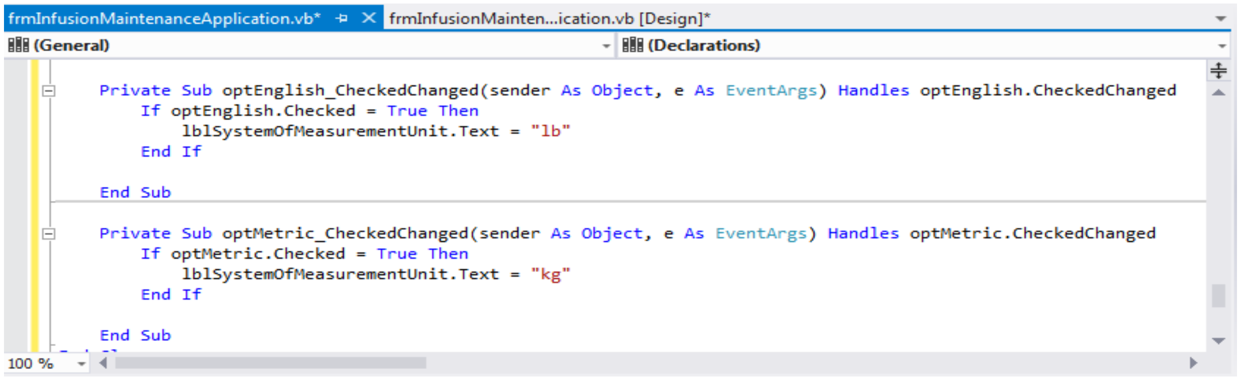
Programming for the Unit of Measurement Label

On the form, we double click on the optEnglish radio button. Then we type a condition statement that asks if the radio button is checked then the lblSystemOfMeasurementUnit.Text will equal "lb".

```
Private Sub optEnglish_CheckedChanged(sender As Object, e As EventArgs) Handles
optEnglish.CheckedChanged
    If optEnglish.Checked = True Then
        lblSystemOfMeasurementUnit.Text = "lb"
    End If
End Sub
```

Again back on the form, we double click on the optMetric radio button. Then we type a condition statement that asks if the radio button is checked then the lblSystemOfMeasurementUnit.Text will equal "kg".

```
Private Sub optMetric_CheckedChanged(sender As Object, e As EventArgs) Handles
optMetric.CheckedChanged
     If optMetric.Checked = True Then
        lblSystemOfMeasurementUnit.Text = "kg"
     End If
End Sub
```



frmInfusionMaintenanceApplication.vb* frmInfusionMainten...ication.vb [Design]*

(General) (Declarations)

```
    Private Sub optEnglish_CheckedChanged(sender As Object, e As EventArgs) Handles optEnglish.CheckedChanged
        If optEnglish.Checked = True Then
            lblSystemOfMeasurementUnit.Text = "lb"
        End If

    End Sub

    Private Sub optMetric_CheckedChanged(sender As Object, e As EventArgs) Handles optMetric.CheckedChanged
        If optMetric.Checked = True Then
            lblSystemOfMeasurementUnit.Text = "kg"
        End If

    End Sub
```

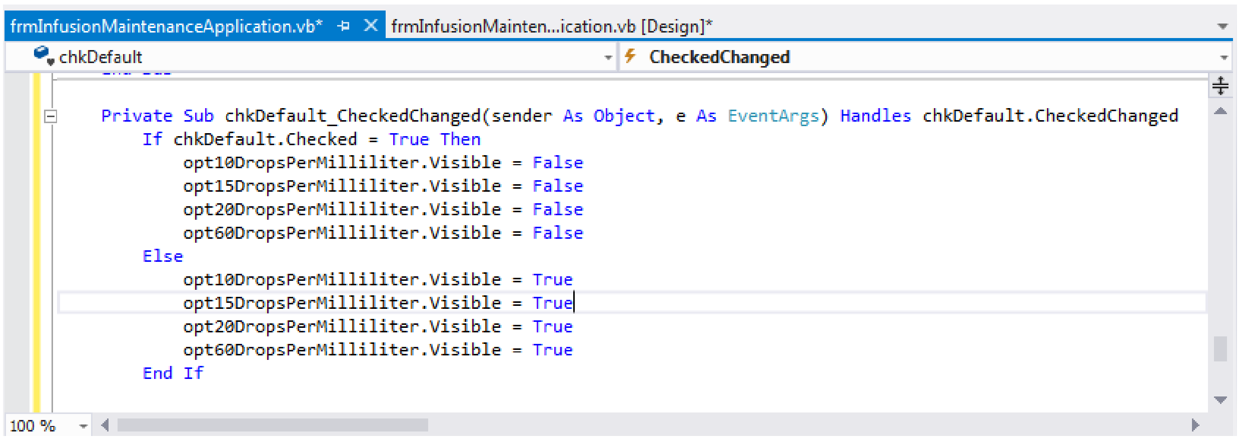
100 %

Figure 9.34 – If Then Statement for the System of Measurement Label

Programming for the Default Checkbox

Back to the form, we double click on the chkDefault checkbox. Then we type a condition statement that asks if the check box is checked then the opt10DropsPerMilliliter.Visible, opt15DropsPerMilliliter.Visible, opt20DropsPerMilliliter.Visible and opt60DropsPerMilliliter.Visible will equal "false". That means these option buttons will not be visible. Else if the check box is not checked then the opt10DropsPerMilliliter.Visible, opt15DropsPerMilliliter.Visible, opt20DropsPerMilliliter.Visible and opt60DropsPerMilliliter.Visible will equal "true". That means these option buttons will be visible.

```
Private Sub chkDefault_CheckedChanged(sender As Object, e As EventArgs) Handles
chkDefault.CheckedChanged
     If chkDefault.Checked = True Then
        opt10DropsPerMilliliter.Visible = False
        opt15DropsPerMilliliter.Visible = False
        opt20DropsPerMilliliter.Visible = False
        opt60DropsPerMilliliter.Visible = False
     Else
        opt10DropsPerMilliliter.Visible = True
        opt15DropsPerMilliliter.Visible = True
        opt20DropsPerMilliliter.Visible = True
        opt60DropsPerMilliliter.Visible = True
End If
```


frmInfusionMaintenanceApplication.vb*
frmInfusionMainten...ication.vb [Design]*
chkDefault
CheckedChanged
100 %

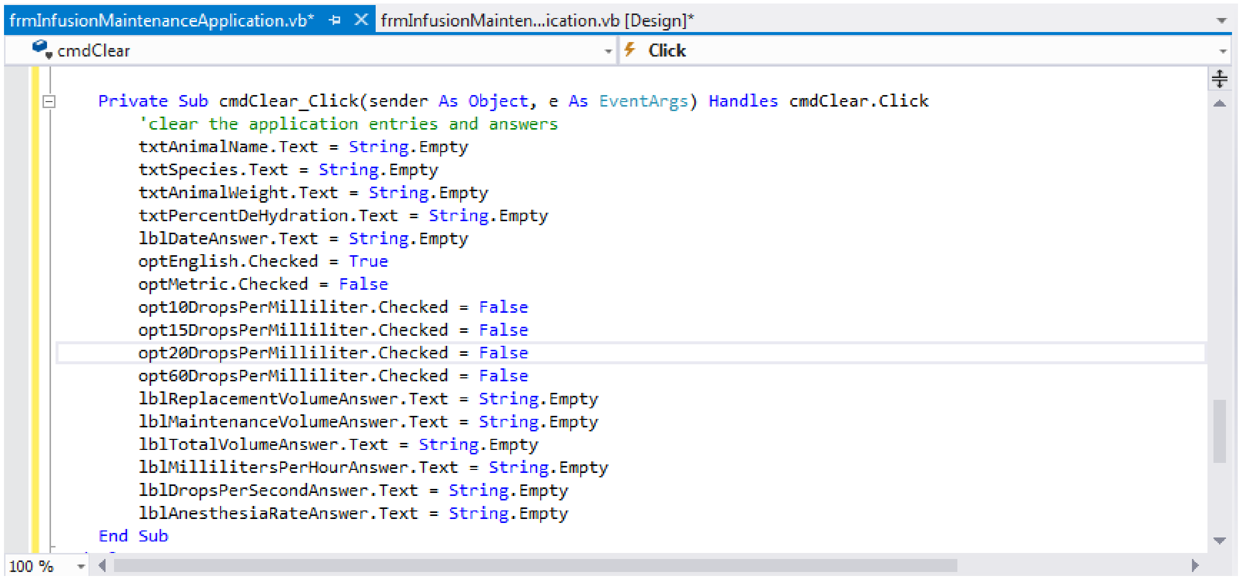
```
Private Sub chkDefault_CheckedChanged(sender As Object, e As EventArgs) Handles chkDefault.CheckedChanged
    If chkDefault.Checked = True Then
        opt10DropsPerMilliliter.Visible = False
        opt15DropsPerMilliliter.Visible = False
        opt20DropsPerMilliliter.Visible = False
        opt60DropsPerMilliliter.Visible = False
    Else
        opt10DropsPerMilliliter.Visible = True
        opt15DropsPerMilliliter.Visible = True
        opt20DropsPerMilliliter.Visible = True
        opt60DropsPerMilliliter.Visible = True
    End If
```

Figure 9.35 – If Then Statement for the Default Checkbox

Clearing the Data

To clear the textboxes or labels containing the data, we will clear the four input textboxes, the seven output labels, and the six option buttons. Type the following code under the cmdClear subroutine of the program

```
Private Sub cmdClear_Click(sender As Object, e As EventArgs) Handles cmdClear.Click
    'clear the application entries and answers
    txtAnimalName.Text = String.Empty
    txtSpecies.Text = String.Empty
    txtAnimalWeight.Text = String.Empty
    txtPercentDeHydration.Text = String.Empty
    lblDateAnswer.Text = String.Empty
    optEnglish.Checked = True
    optMetric.Checked = False
    opt10DropsPerMilliliter.Checked = False
    opt15DropsPerMilliliter.Checked = False
    opt20DropsPerMilliliter.Checked = False
    opt60DropsPerMilliliter.Checked = False
    lblReplacementVolumeAnswer.Text = String.Empty
    lblMaintenanceVolumeAnswer.Text = String.Empty
    lblTotalVolumeAnswer.Text = String.Empty
    lblMillilitersPerHourAnswer.Text = String.Empty
    lblDropsPerSecondAnswer.Text = String.Empty
    lblAnesthesiaRateAnswer.Text = String.Empty
End Sub
```

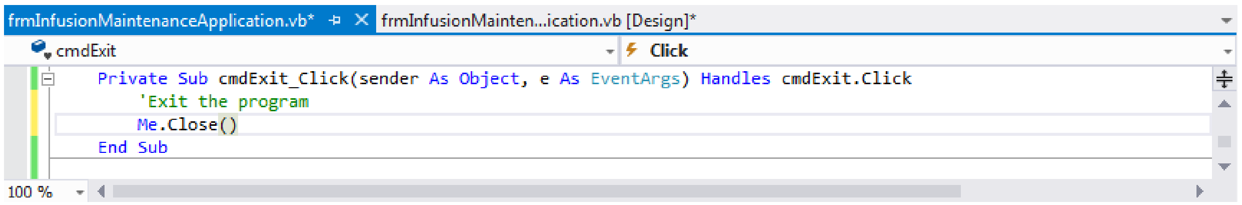

frmInfusionMaintenanceApplication.vb* frmInfusionMainten...ication.vb [Design]*
cmdClear Click

```
Private Sub cmdClear_Click(sender As Object, e As EventArgs) Handles cmdClear.Click
    'clear the application entries and answers
    txtAnimalName.Text = String.Empty
    txtSpecies.Text = String.Empty
    txtAnimalWeight.Text = String.Empty
    txtPercentDeHydration.Text = String.Empty
    lblDateAnswer.Text = String.Empty
    optEnglish.Checked = True
    optMetric.Checked = False
    opt10DropsPerMilliliter.Checked = False
    opt15DropsPerMilliliter.Checked = False
    opt20DropsPerMilliliter.Checked = False
    opt60DropsPerMilliliter.Checked = False
    lblReplacementVolumeAnswer.Text = String.Empty
    lblMaintenanceVolumeAnswer.Text = String.Empty
    lblTotalVolumeAnswer.Text = String.Empty
    lblMillilitersPerHourAnswer.Text = String.Empty
    lblDropsPerSecondAnswer.Text = String.Empty
    lblAnesthesiaRateAnswer.Text = String.Empty
End Sub
```

100 %

Figure 9.36 – Computing the Clear Button by Clearing a Textbox and Label Caption

Exiting the Program


frmInfusionMaintenanceApplication.vb* frmInfusionMainten...ication.vb [Design]*
cmdExit Click

```
Private Sub cmdExit_Click(sender As Object, e As EventArgs) Handles cmdExit.Click
    'Exit the program
    Me.Close()
End Sub
```

100 %

Figure 9.37 – Exiting the Program

To exit this program, we will unload the application and end the program.
Type the following code:

```
'Exit the program
Me.Close()
```

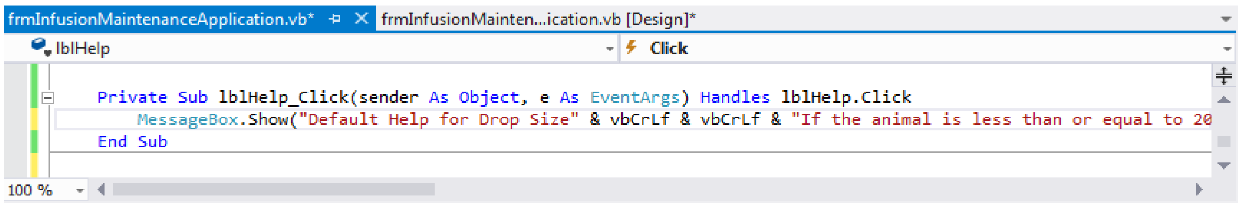
Programming a Help Hyperlink

To create a link to a label, we should double click on the blue Help label on the form. A private sub will be created in the application called lblHelp_Click. We will want to add our code that will launch a message box in this subroutine.

```
Private Sub lblHelp_Click(sender As Object, e As EventArgs) Handles lblHelp.Click

    MessageBox.Show("Default Help for Drop Size" & vbCrLf & vbCrLf & "If the animal is less than or
```

equal to 20 lbs body weight, we will " & vbCrLf & "use 60 drops per milliliter. If the animal is more than 20 lbs body " & vbCrLf & "weight, we will use 15 drops per milliliter. Uncheck the default" & vbCrLf & "box and any drop size can be chosen.")

End Sub


frmInfusionMaintenanceApplication.vb*
frmInfusionMainten...ication.vb [Design]*
lblHelp
Click
100 %

```
    Private Sub lblHelp_Click(sender As Object, e As EventArgs) Handles lblHelp.Click
        MessageBox.Show("Default Help for Drop Size" & vbCrLf & vbCrLf & "If the animal is less than or equal to 20
    End Sub
```

Figure 9.38 – Help Label

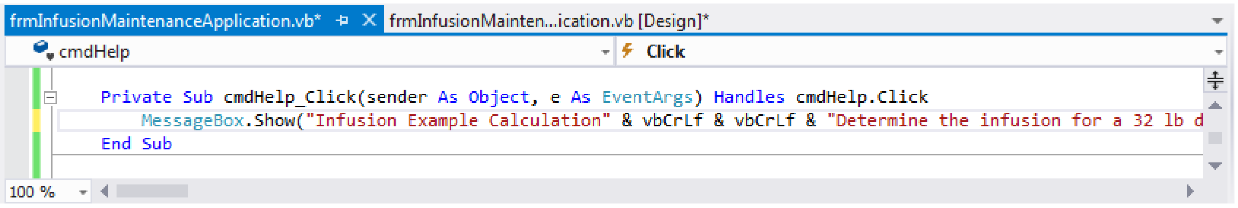
Programming a Help Button

On the form, we will double click on the Help command button. We will program a message box using the MessgeBox.Show ( ) expression. We use our storyboard information to construct the message. We use the **& vbCrLf & vbCrLf &** to add lines to the document.

Private Sub cmdHelp_Click(sender As Object, e As EventArgs) Handles cmdHelp.Click

MessageBox.Show("Infusion Example Calculation" & vbCrLf & vbCrLf & "Determine the infusion for a 32 lb dog that is 9 % dehydrated." & vbCrLf & vbCrLf & "Compute the replacement volume at 1 liter per 1 kilogram. The dehydration rate is reported as a decimal percent (9% = 0.09)." & vbCrLf & vbCrLf & "(32 lb)/1×(1 kg)/(2.2 lb)×0.09/×(1 L)/(1 kg)×(1000 ml)/(1 L)=2880/2.2= 1309.09 ml replacement volume" & vbCrLf & vbCrLf & "Compute the maintenance volume at 40 ml per 1 kilogram." & vbCrLf & vbCrLf & "(32 lb)/1×(1 kg)/(2.2 lb)×(40 ml)/(1 kg)=1280/2.2= 581.82 ml maintenance volume" & vbCrLf & vbCrLf & "Compute the total volume by adding the replacement and maintenance volume together." & vbCrLf & vbCrLf & "1309.09 ml replacement volume + 581.82 ml maintenance volume = 1890.91 ml total volume" & vbCrLf & vbCrLf & "Compute the milliliters per hour infusion rate by dividing the total volume by 24 hours." & vbCrLf & vbCrLf & "(1890.91 ml)/24 hr = 78.79 ml / hr " & vbCrLf & vbCrLf & "Compute the drips per second infusion rate by dividing the total volume by 24 hours and 60 minutes and 60 seconds. If the animal is less than 20 lbs body weight, we will use 60 drops per milliliter. If the animal is more than 20 lbs body weight, we will use 15 drops per milliliter." & vbCrLf & vbCrLf & "(78.79 ml)/(1 hour)×(15 drops)/(1 ml)x(1 hour)/(3600 sec) = 0.33 drops / sec" & vbCrLf & vbCrLf & "Compute the anesthesia / surgery rate by multiplying the animal's body weight by 4 ml per lb per hour and dividing by 60 minutes." & vbCrLf & vbCrLf & "(32 lb)/1×(4 ml)/(1 hr |1 lb)×(1 hr)/(60 min) = 2.13 ml / min")

End Sub


frmInfusionMaintenanceApplication.vb*
frmInfusionMainten...ication.vb [Design]*
cmdHelp
Click
100 %

```
    Private Sub cmdHelp_Click(sender As Object, e As EventArgs) Handles cmdHelp.Click
        MessageBox.Show("Infusion Example Calculation" & vbCrLf & vbCrLf & "Determine the infusion for a 32 lb d
    End Sub
```

Figure 9.39 – Help Button Code

Programming a Print Button

When we want the completed form printed we will need to do some extra work. Back on the form, we will open the Toolbox and at the bottom of the menu, we will expand the Visual Basic PowerPacks and double click on PrintForm.

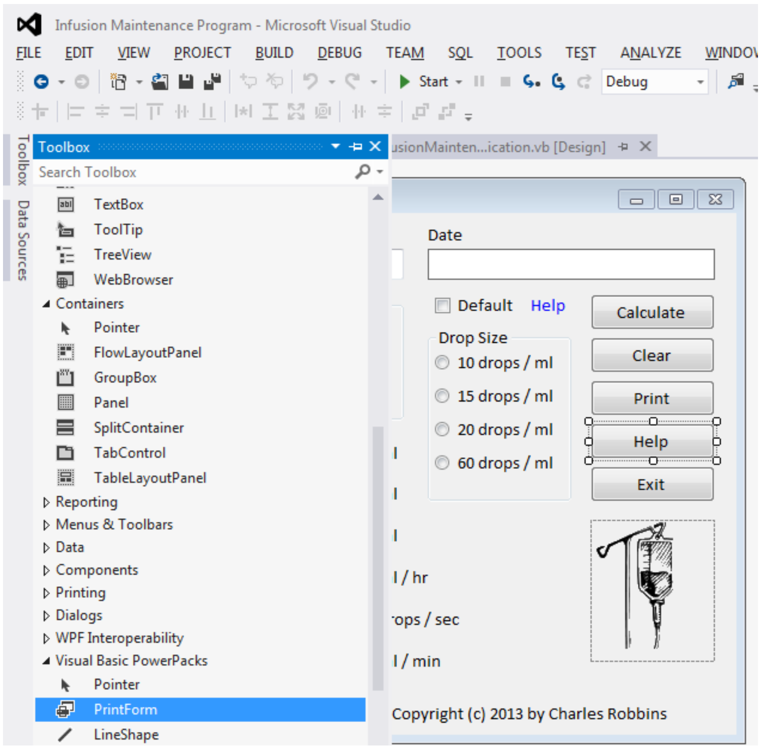


Figure 9.40 – Visual Basic PowerPacks and PrintForm

We double click on the Print command button and we type the following.

```
PrintForm1.PrintAction = Printing.PrintAction.PrintToPreview
PrintForm1.Print()
```

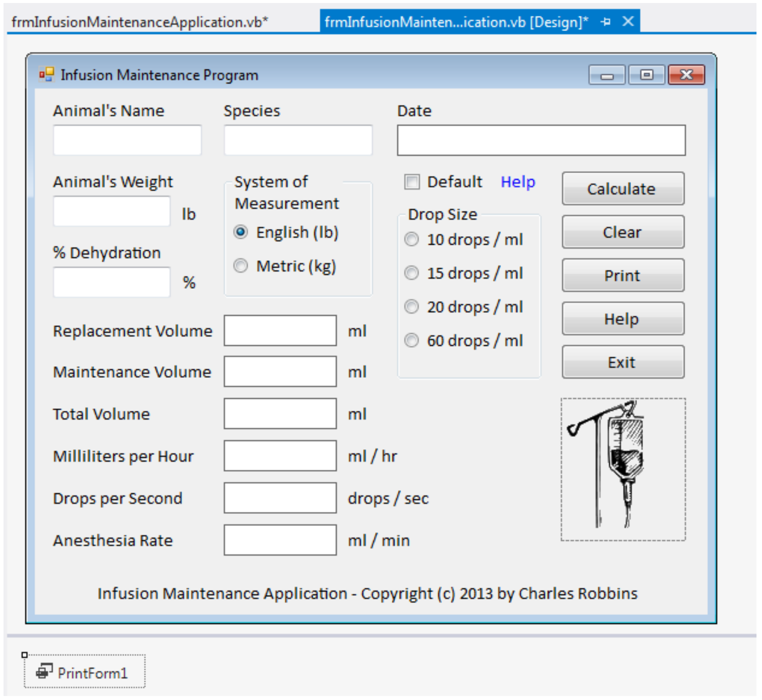
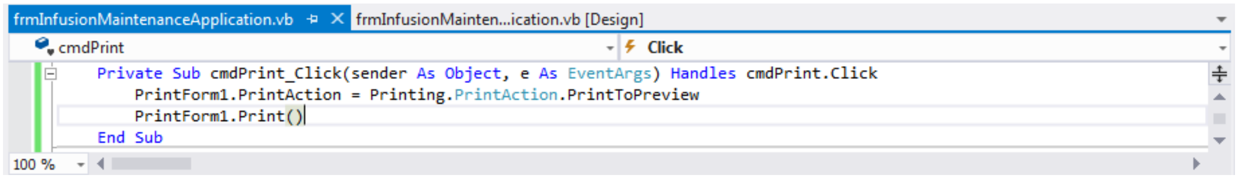


Figure 9.41 – PrintForm1 on the Program


frmInfusionMaintenanceApplication.vb
frmInfusionMainten...ication.vb [Design]
cmdPrint
Click
100 %

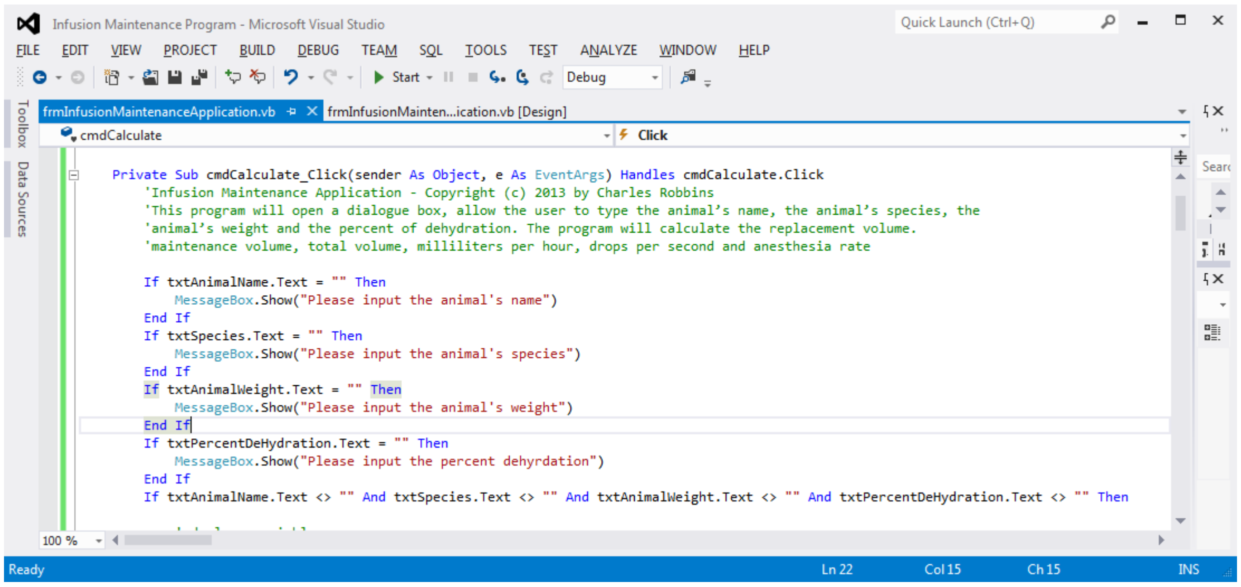
```
Private Sub cmdPrint_Click(sender As Object, e As EventArgs) Handles cmdPrint.Click
    PrintForm1.PrintAction = Printing.PrintAction.PrintToPreview
    PrintForm1.Print()
End Sub
```

Figure 9.42 – Print Code

Adding Code to Check for Blank Textboxes

Still again, we will add four more condition statements on the code under the cmdCalculate subroutine. We will check to see if the Animal's Name textbox, the Species textbox, the Animal's Weight textbox and the Percent Dehydration textbox are blank. If one or all are blank, then we will be prompted to fix the problem. If they all are not in the next condition statement, then the program will run, so we add the conclusion to the second if statement to the end of calculate subroutine.

```
If txtAnimalName.Text = "" Then
        MessageBox.Show("Please input the animal's name")
    End If
    If txtSpecies.Text = "" Then
        MessageBox.Show("Please input the animal's species")
    End If
    If txtAnimalWeight.Text = "" Then
        MessageBox.Show("Please input the animal's weight")
    End If
    If txtPercentDehydration.Text = "" Then
        MessageBox.Show("Please input the percent dehyrdation")
    End If
    If txtAnimalName.Text <> "" And txtSpecies.Text <> "" And txtAnimalWeight.Text <> "" And
txtPercentDeHydration.Text <> "" Then

        ' declare variables
        Dim AnimalName As String

         ................. The Calculate Subroutine ...............

        lblAnesthesiaRateAnswer.Text = AnesthesiaRate.ToString("n2")
        lblDateAnswer.Text = TodaysDate & " " & TimeofDay

     End If

End Sub
```


Infusion Maintenance Program - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ANALYZE WINDOW HELP
Start
Debug
frmInfusionMaintenanceApplication.vb
frmInfusionMainten...ication.vb [Design]
cmdCalculate
Click
Toolbox
Data Sources
100 %
Ready
Ln 22
Col 15
Ch 15
INS

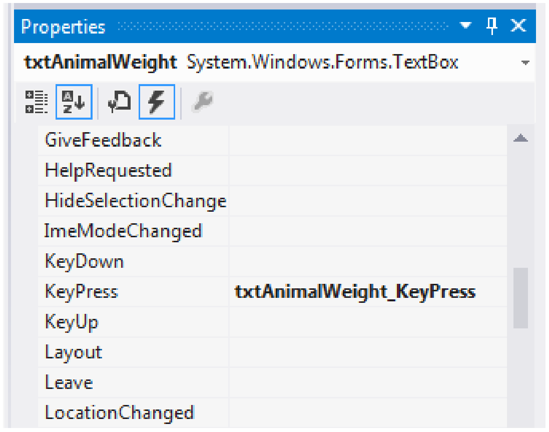
```
    Private Sub cmdCalculate_Click(sender As Object, e As EventArgs) Handles cmdCalculate.Click
        'Infusion Maintenance Application - Copyright (c) 2013 by Charles Robbins
        'This program will open a dialogue box, allow the user to type the animal’s name, the animal’s species, the
        'animal’s weight and the percent of dehydration. The program will calculate the replacement volume.
        'maintenance volume, total volume, milliliters per hour, drops per second and anesthesia rate

        If txtAnimalName.Text = "" Then
            MessageBox.Show("Please input the animal's name")
        End If
        If txtSpecies.Text = "" Then
            MessageBox.Show("Please input the animal's species")
        End If
        If txtAnimalWeight.Text = "" Then
            MessageBox.Show("Please input the animal's weight")
        End If
        If txtPercentDeHydration.Text = "" Then
            MessageBox.Show("Please input the percent dehyrdation")
        End If
        If txtAnimalName.Text <> "" And txtSpecies.Text <> "" And txtAnimalWeight.Text <> "" And txtPercentDeHydration.Text <> "" Then
```

Figure 9.43 – Condition Statements Checking for Blank Text Boxes

Controlling the Input in the Textbox

Back at the form, we highlight the Animal Weight textbox. On the Properties window, we will press the Events button that has a lightning bolt on it. Go to KeyPress on the list and double click it.

Do the same for the Percent Dehydration textbox.



GiveFeedback	
HelpRequested	
HideSelectionChange	
ImeModeChanged	
KeyDown	
KeyPress	txtAnimalWeight_KeyPress
KeyUp	
Layout	
Leave	
LocationChanged	

Figure 9.44 – KeyPress Event

We will then add yet another condition statement that checks for characters less than 46 or more than 57 or equal to 47. If this is true then we will launch as message box asking for numbers only. We use the ASCII chart to find these numbers.

```
Private Sub txtAnimalWeight_KeyPress(sender As Object, e As KeyPressEventArgs) Handles
txtAnimalWeight.KeyPress
    If Asc(e.KeyChar) < 46 Or Asc(e.KeyChar) > 57 Or Asc(e.KeyChar) = 47 Then
      e.Handled = True
      MessageBox.Show("Please, only input numbers")
    End If
  End Sub
```

```
   Private Sub txtPercentDeHydration_KeyPress(sender As Object, e As KeyPressEventArgs) Handles
txtPercentDeHydration.KeyPress
       'Allows Only Numbers
       If Asc(e.KeyChar) < 46 Or Asc(e.KeyChar) > 57 Or Asc(e.KeyChar) = 47 Then
           e.Handled = True
           MessageBox.Show("Please, only input numbers")
       End If
   End Sub
```

Running the Program

After noting that the program is saved, press the F5 to run the Infusion Maintenance Application. The Infusion Maintenance Application window will appear on the graphical display as shown in Figure 9.45. Notice the professional appearance and presentation of information in a clean dialogue box.

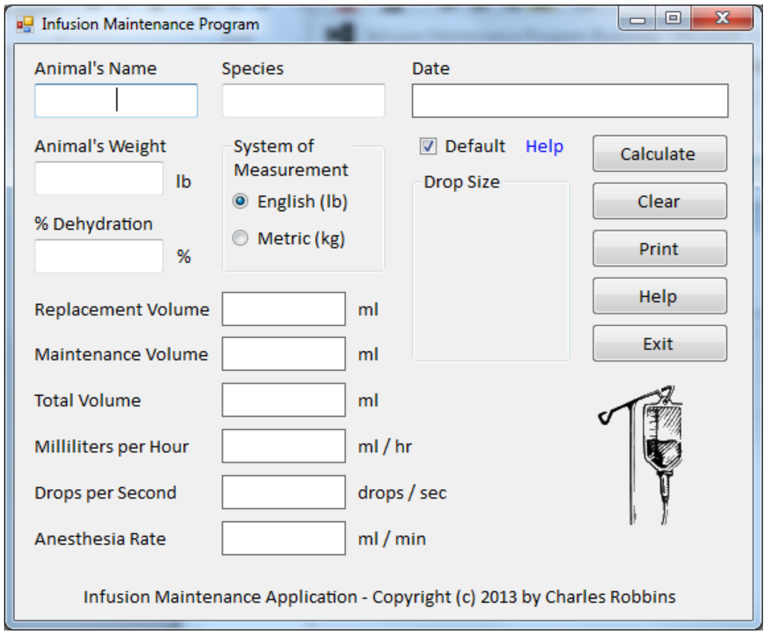
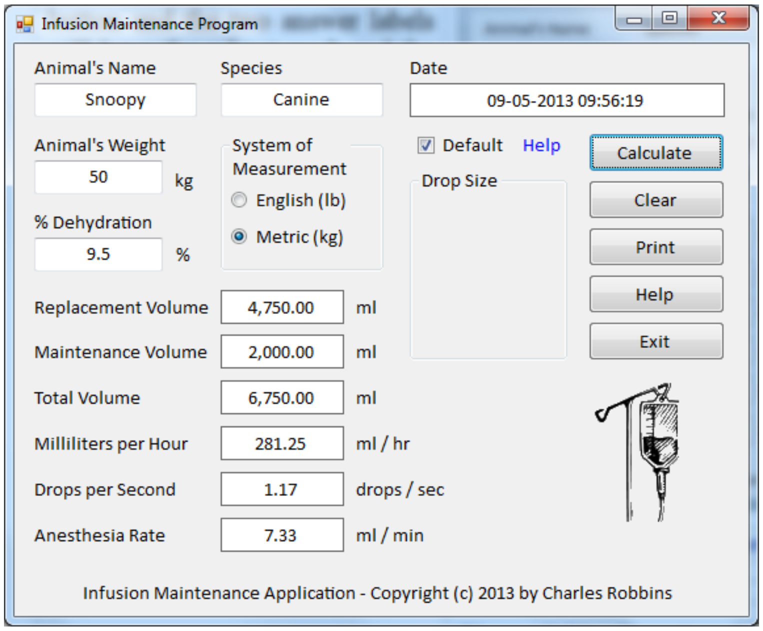


Figure 9.45 – Running the Program using Pounds

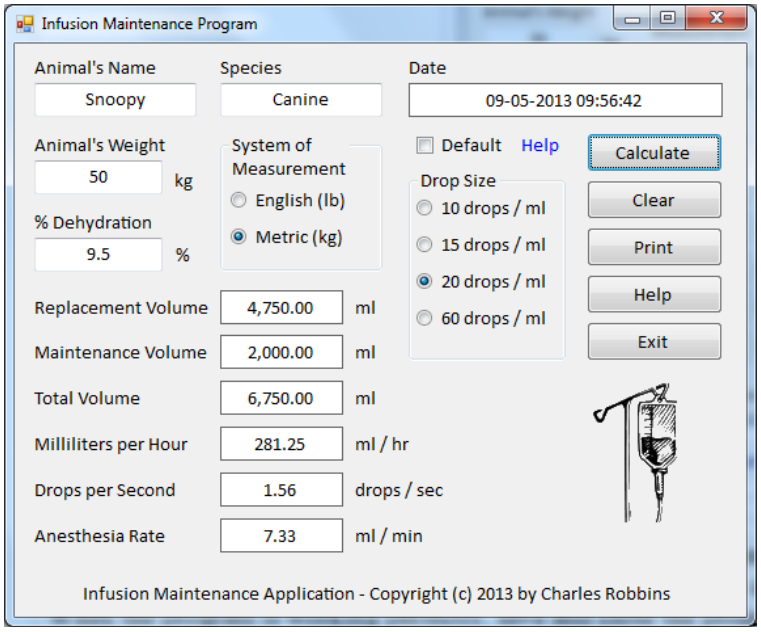
Type in the Animal's Name, the Species type, the Animal's Weight and the Percent Dehydration as shown in figure 9.46. Press the Calculate command button and the six answer labels will have the infusion maintenance answers and one label will have the date and time.



Animal's Name: Snoopy
Species: Canine
Date: 09-05-2013 09:56:19
Animal's Weight: 50 kg
% Dehydration: 9.5 %
System of Measurement: English (lb) [✓ Metric (kg)]
[✓ Default] Help
Drop Size
Replacement Volume: 4,750.00 ml
Maintenance Volume: 2,000.00 ml
Total Volume: 6,750.00 ml
Milliliters per Hour: 281.25 ml / hr
Drops per Second: 1.17 drops / sec
Anesthesia Rate: 7.33 ml / min
Infusion Maintenance Application - Copyright (c) 2013 by Charles Robbins

Figure 9.46 – Running the Program using Kilograms

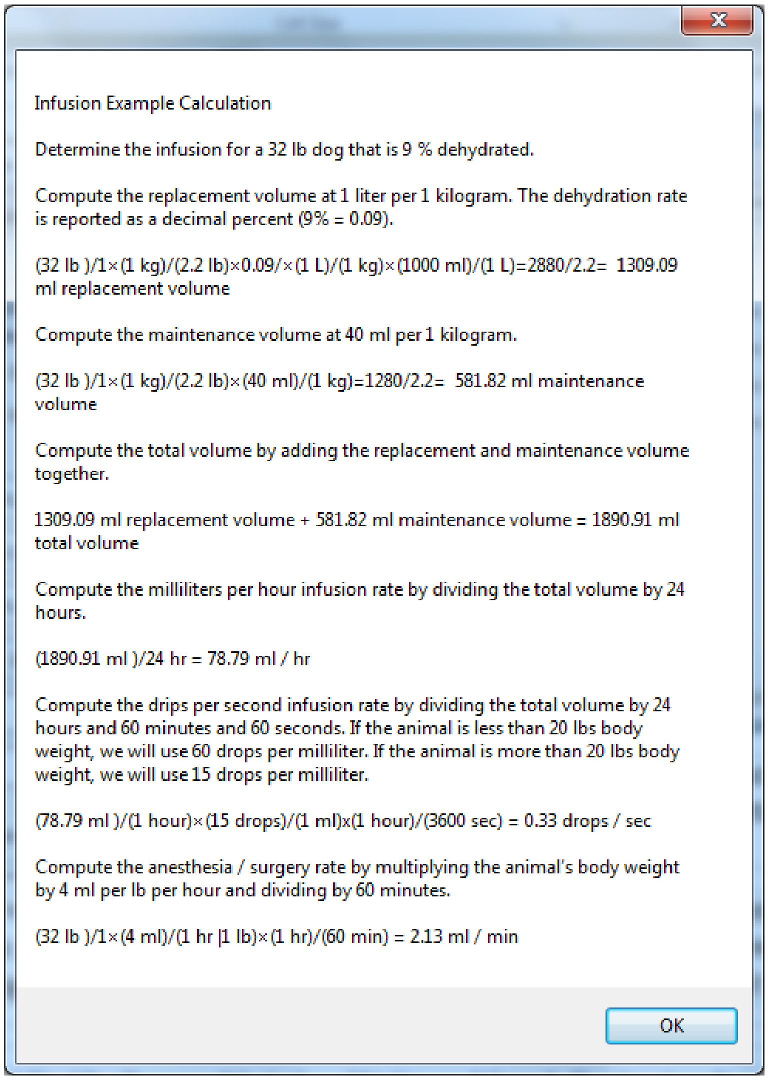
To check the rest of the program, we will uncheck the default box, then annotate each of the drop size radio buttons and calculate the answers.



Animal's Name: Snoopy
Species: Canine
Date: 09-05-2013 09:56:42
Animal's Weight: 50 kg
% Dehydration: 9.5 %
System of Measurement: English (lb) [✓ Metric (kg)]
Default Help
Drop Size: 10 drops / ml, 15 drops / ml, [✓ 20 drops / ml], 60 drops / ml
Replacement Volume: 4,750.00 ml
Maintenance Volume: 2,000.00 ml
Total Volume: 6,750.00 ml
Milliliters per Hour: 281.25 ml / hr
Drops per Second: 1.56 drops / sec
Anesthesia Rate: 7.33 ml / min
Infusion Maintenance Application - Copyright (c) 2013 by Charles Robbins

Figure 9.47 – Running the Program using Drop Size

We will press the Help button and the large message box will appear. We press the OK button to close it.



Infusion Example Calculation

Determine the infusion for a 32 lb dog that is 9 % dehydrated.

Compute the replacement volume at 1 liter per 1 kilogram. The dehydration rate is reported as a decimal percent (9% = 0.09).

(32 lb)/1×(1 kg)/(2.2 lb)×0.09/×(1 L)/(1 kg)×(1000 ml)/(1 L)=2880/2.2= 1309.09 ml replacement volume

Compute the maintenance volume at 40 ml per 1 kilogram.

(32 lb)/1×(1 kg)/(2.2 lb)×(40 ml)/(1 kg)=1280/2.2= 581.82 ml maintenance volume

Compute the total volume by adding the replacement and maintenance volume together.

1309.09 ml replacement volume + 581.82 ml maintenance volume = 1890.91 ml total volume

Compute the milliliters per hour infusion rate by dividing the total volume by 24 hours.

(1890.91 ml)/24 hr = 78.79 ml / hr

Compute the drips per second infusion rate by dividing the total volume by 24 hours and 60 minutes and 60 seconds. If the animal is less than 20 lbs body weight, we will use 60 drops per milliliter. If the animal is more than 20 lbs body weight, we will use 15 drops per milliliter.

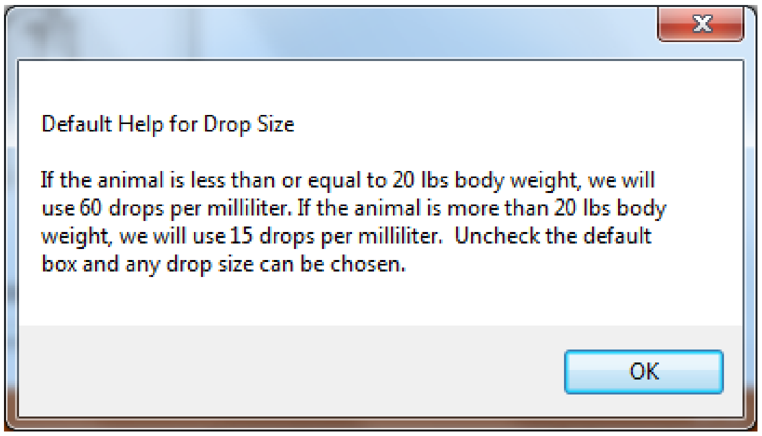
(78.79 ml)/(1 hour)×(15 drops)/(1 ml)x(1 hour)/(3600 sec) = 0.33 drops / sec

Compute the anesthesia / surgery rate by multiplying the animal's body weight by 4 ml per lb per hour and dividing by 60 minutes.

(32 lb)/1×(4 ml)/(1 hr |1 lb)×(1 hr)/(60 min) = 2.13 ml / min

Figure 9.48 – The Help Button Message Box

We will press the blue Help label and the small message box will appear. We press the OK button to close it.



Default Help for Drop Size

If the animal is less than or equal to 20 lbs body weight, we will use 60 drops per milliliter. If the animal is more than 20 lbs body weight, we will use 15 drops per milliliter. Uncheck the default box and any drop size can be chosen.

Figure 9.49 – The Help Label Message Box

We will press the Print button and we will print the form.

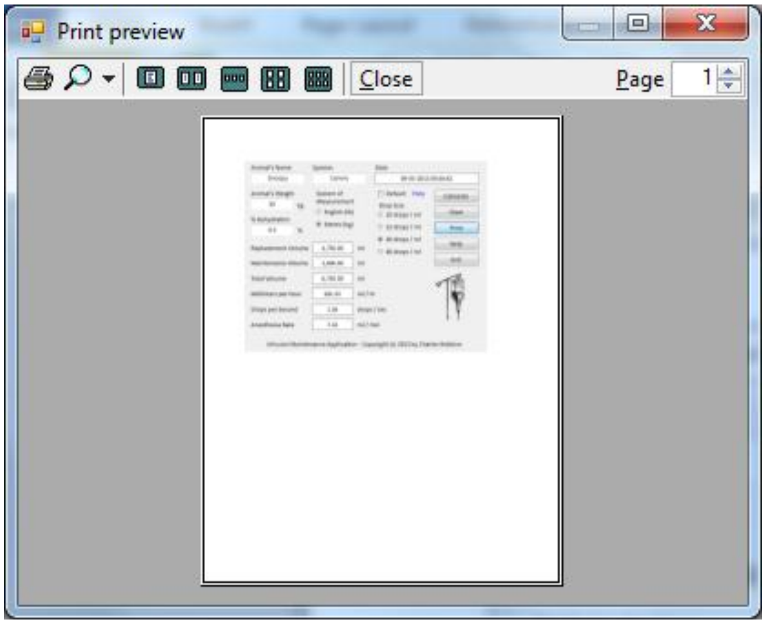
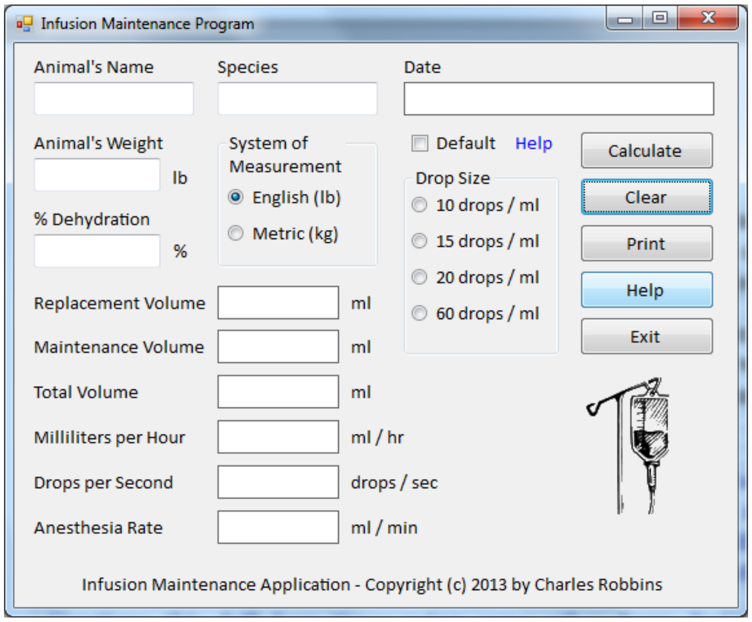


Figure 9.50 – The Print Button

We will press the Clear button to make sure the textboxes and output labels are emptied.


Infusion Maintenance Program

Animal's Name	Species	Date
Animal's Weight lb	System of Measurement: [✓ English (lb)] Metric (kg)	Default Help
% Dehydration %		Drop Size: 10 drops / ml, 15 drops / ml, 20 drops / ml, 60 drops / ml
Replacement Volume	ml	
Maintenance Volume	ml	
Total Volume	ml	
Milliliters per Hour	ml / hr	
Drops per Second	drops / sec	
Anesthesia Rate	ml / min	

Infusion Maintenance Application - Copyright (c) 2013 by Charles Robbins

Figure 9.51 – The Clear Button

If our program does not function correctly, go back to the code and check the syntax against the program shown in previous sections. Repeat any processes to check or Beta test the program. When the program is working perfectly, save and close the project.

There are many variations of this Visual Basic Application we can practice and obtain information from a personal computer. While we are practicing with forms, we can learn how to use variables, strings and comments. These are skills that we want to commit to memory.

*** World Class CAD Challenge 90-9 * - Write a Visual Basic Application that writes a program that calculates the amount and rate of flow of fluids for dehydrated animals. Complete the assignment in two hours to maintain your World Class CAD ranking.**

Continue this drill four times using some other form designs, each time completing the Visual Basic Project in less than 2 hour to maintain your World Class ranking.