

NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

BILLING AND PAYMENT SYSTEM PROGRAM

**Graduation Project
COM 400**

Student: Doğan Bayraktar(20030493)

Supervisor : Mr. Ümit Soyer

Nicosia 2008

ACKNOWLEDGMENTS

"first of all I would like to thank my supervisor Mr. Umit Soyer for his way of treatment with me and believe in my work , he never rejected me when I had any question and, he believed me to do the course of this project.

I don't forget other instructors, Mr. Umit Ilhan was also helped me whenever I had any problem and he gave me ideas. I thank them very much with my all for their advices.

Secondly I would like to thank my family for their molar tooth of graduated me from this university. I know that because of their selfless I was here and I never want to make them disappointed. So we trust each other and I will be graduated.

Thirdly I want to thanks my friends, also Renan Cakirerk , he helped me in my project and gave me ideas and parted his time with me.

Finally I would like to thank all my friends who aid me during my university life, to resist and study."

ABSTRACT

Billing & Payment System Programming is not written to a private company so many companies can use this program by adding a bit of things. The company can control all their users, their customers and their operations, sales and buys in details, payments and debts

Visual Basic programming language and SQL Server is used while doing this program. The program is easy in use; every one can understand the behavior of the program. The user can add company, add customer, record the payments and sold items, calculate the in and out debts.

The main form of the program is MDIform. The menus take place in the MDIform. The user can control the program with this form. All menus record everything that are included in. Also the program can update and delete the records from the database. Sales and Expenses are recorded with this program.

So using like this program makes easy to billing and payment. The company's income and debts are seen. Also the expense debts and payments are shown with this program. You can calculate the profit of the company.

TABLE OF CONTENTS

ACKNOWLEDGMENT.....	i
ABSTRACT.....	ii
TABLE OF CONTENTS.....	iii
LIST OF FIGURES & TABLES.....	vi
INTRODUCTION	viii
CHAPTER I: VISUAL BASIC.....	01
1.1. What is Visual Basic?.....	01
1.2. Significant Language Features	02
1.3. The Advantages and Disadvantages of Visual Basic.....	03
1.3.1. The Advantages of Visual Basic.....	03
1.3.2. The Disadvantages of Visual Basic.....	04
1.4. Areas of Application.....	04
1.5. The Development Environment.....	05
1.6. The Project Explore Window.....	06
1.6.1. Opening an existing Visual Basic Project.....	06
1.6.2. Running and Viewing the project in Detail.....	07
1.6.3. Properties Window.....	09
1.6.4. The Default Layout.....	09
1.6.5. Tool Box.....	10
1.6.6. Opening a New Visual Basic File & Inserting Source code.....	11
1.7. The Events.....	11
1.7.1. What an Event?.....	12
1.7.2. Click Event.....	12
1.7.3. Writing a Code.....	13
1.7.4. UsingThe Event GotFocus.....	14
1.7.5. CheckBoxes.....	14
1.7.6. Option Buttons.....	15
1.7.7. ListBoxes.....	16
1.7.7.1 Add to a Listbox.....	17
1.7.7.2. Removing from Listboxes & Advanced Options.....	18

1.7.8. MsgBoxes.....	18
1.7.9. Opening & Retrieving Information From files.....	21
1.7.10. Storing Information to a file.....	22
1.7.11. Printing Text to the printer.....	22
1.8. Managing Data Types.....	23
1.8.1. Numeric Data Types.....	23
1.8.2. Non-Numeric Data Types.....	23
1.8.3. Declaring Variables.....	24
1.9. VB Functions.....	26
1.9.1. Creating Functions.....	26
1.9.2. Declaring Array.....	26
1.10. Database Application.....	27
CHAPTER II: DATABASE STRUCTURE.....	28
2.1. What is Microsoft SQL Server.....	28
2.2. Creating The Database.....	29
2.2.1. Choosing a Database Platform.....	29
2.2.1. Creating The Database.....	29
2.3. Creating Tables.....	29
2.3.1. Creating Our First Table.....	29
2.2.2. Attributes and Data Types.....	30
2.3.3. NULL Values.....	31
2.3.4. Bulding The Remaining Tables.....	31
2.3.5. Modifying Tables.....	32
2.4. Built in Data Types.....	33
2.5. Microsoft SQL Server Constraints.....	36
2.6. Working With SQL.....	39
2.7. Data Manipulating Language.....	39
2.7.1. INSERT.....	39
2.7.2. SELECT.....	40
2.7.3. UPDATE.....	40
2.7.4. DELETE.....	41
2.8. Tables in The Project.....	41

CHAPTER III: BILLING AND PAYMENT SYSTEM

PROGRAM.....	45
3.1. Login Page.....	45
3.2. Main Page.....	46
3.3. Menu Bar.....	46
3.3.1. Sales Menu.....	46
3.3.2. Expenses Menu.....	47
3.4. Add New Person Page.....	47
3.5. Add New Company Page.....	48
3.6. Invoice Page.....	48
3.7. Payment Page.....	49
3.8. Debt Payment Page.....	50
3.9. Invoiceto Page.....	50
3.10. Paymentto Page.....	51
3.11. Debt Paymentto Page.....	52
3.12. Search Invoice Page... ..	52
3.13. Search Payment Page.....	53
3.14. Search Expenses Page.....	53
3.15. Search Paymentto Page.....	54
3.16. Add Customer Page.....	54
3.17. About Page.....	55
 CONCLUSION.....	 56
REFERENCES.....	57
APPENDICES.....	58

LIST OF FIGURES & TABLES

Figure 1.1: An Empty Form.....	02
Figure 1.2: A Simple Program.....	03
Figure1.3: Development Environment.....	06
Figure1.4: Project of Program in Detail.....	07
Figure 1.5: Properties Window.....	09
Figure1.6: Form & Load.....	12
Figure 1.7: Click Event Form	13
Figure1.8: CheckBox Example.....	14
Figure 1.9: Background Changing Example.....	15
Figure1.10: A list on Form.....	17
Figure 1.10:1. Adding to List.....	17
Figure 1.11: Showing Message.....	20
Figure 3.1: Login Page.....	45
Figure 3.2: Main Page.....	46
Figure 3.3: Menu Bar.....	46
Figure 3.4: Sales Menu.....	46
Figure 3.5: Expenses Menu.....	47
Figure 3.6: Add Person Page.....	47
Figure 3.7: Add Company Page.....	48
Figure 3.8: Invoice Page.....	48
Figure 3.9: Payment Page.....	49
Figure 3.10: Debt Payment Page.....	50
Figure 3.11: Invoiceto Page.....	50
Figure 3.12: Paymentto Page.....	51
Figure 3.13: Debt Paymentto Page.....	52
Figure 3.14: Search Invoice Page.....	52
Figure 3.15: Search Payment Page.....	53
Figure 3.16: Search Expenses Page.....	53
Figure 3.17: Search Paymentto Page.....	54
Figure 3.18: Add Customer Page.....	54
Figure 3.19: About Page.....	55
Table 1.1: Numeric Data Types.....	23

Table 1.2: Nonnumeric Data Types.....	24
Table 2.1: Pperson Table As an Example of SQL Server.....	33
Table 2.2: Data Types.....	35
Table 2.3: Firm Database Table.....	42
Table 2.4: Item Database Table.....	42
Table 2.5: Receipt Database Table.....	42
Table 2.6: Receiptto Database Table.....	43
Table 2.7: Bill Database Table.....	43
Table 2.8: BillNo Database Table.....	43
Table 2.9: Person Database Table.....	43
Table 2.10: Customer Database Table.....	43
Table 2.11: Debtpay Database Table.....	43

INTRODUCTION

Billing and Payment System Program is useful for companies that have items to sell and buy. With this program we can do every process of selling and buying. If this program used in a correct way nothing will be confused while making a process. This program makes everything simple that companies need to process.

With this program you can see the customers and companies that you are selling and buying items see the debtors and your debts, calculate your buying expenses and selling incomes.

Chapter One describes Visual basic, it's properties, components and some examples, we used Visual Basic Program in my project, because we find it easy and we like it's coding system. Visual basic for applications delivers a competitive advantage for ISV seeking to provide full customization and integration capabilities to customer.

Chapter Two presents the Database, SQL Server 2000 is used in my program, Advantages of using SQL Server is to provide exactly the same options for the problems we write as it does for the problems we selected from a data base. Secondly, the process of writing or selecting problems is almost independent of page layout dictions. Also you can see more details about access and SQL.

Chapter Three presents Billing and Payment System Program; it is the most important Chapter of this project, because the codes and design are included there. All the codes are written and a bit help of friends and teachers.

CHAPTER I: VISUAL BASIC

1.1.What Is Visual Basic?

VISUAL BASIC is a high level programming language evolved from the earlier DOS version called BASIC. BASIC means **B**eginners' **A**ll-purpose **S**ymbolic **I**nstruction **C**ode. It is a fairly easy programming language to learn. The codes look a bit like English Language. Different software companies produced different version of BASIC, such as Microsoft QBASIC, QUICKBASIC, GWBASIC ,IBM BASICA and so on. Programmers have undergone a major change in many years of programming various machines. For example what could be created in minutes with Visual Basic could take days in other languages such: as "C" or "Pascal". Visual Basic provides many interesting sets of tools to aid you in building exciting applications. Visual Basic provides these tools to make your life far more easier because all the real hard code is already written for you.

With controls like these you can create many applications which use certain parts of windows. For example, one of the controls could be a button, which we have demonstrated in the "Hello World" program below. First create the control on the screen, then write the code which would be executed once the control button is pressed. With this sort of operation in mind, simple programs would take very little code. Why do it like the poor old "C" programmer who would have to write code to even display a window on the screen, when Visual Basic already has this part written for you.

Even though people tend to say Visual Basic's compiler is far behind the compilers of Pascal and C, it has earned itself the status of a professional programming language, and has almost freed BASIC of the reputation of a children's language. Overall you would class Visual Basic as a Graphics User Interface(GUI). Because as you draw, you write for the program. This must always be remembered in any kind of creation of a Visual Basic program. All in all, VB is the preferred language of many future programmers. If you want to start programming Windows, and don't know how to start, give Visual Basic a shot.

1.2. Significant Language Features.

Visual Basic is not only a programming language, but also a complete graphical development environment. This environment allows users with little programming experience to **quickly** develop useful Microsoft Windows applications which have the ability to use OLE (Object Linking and Embedding) objects, such as an Excel spreadsheet. Visual Basic also has the ability to develop programs that can be used as a front end application to a database system, serving as the user interface which collects user input and displays formatted output in a more appealing and useful form than many SQL versions are capable of.

Visual Basic's main selling point is the ease with which it allows the user to create nice looking, graphical programs with little coding by the programmer, unlike many other languages that may take hundreds of lines of programmer keyed code. As the programmer works in the graphical environment, much of the program code is automatically generated by the Visual Basic program. In order to understand how this happens it is necessary to understand the major concepts, objects and tools used by Visual Basic. The main object in Visual Basic is called a **form**. When you open a new project, you will start with a clear form that looks similar to this :

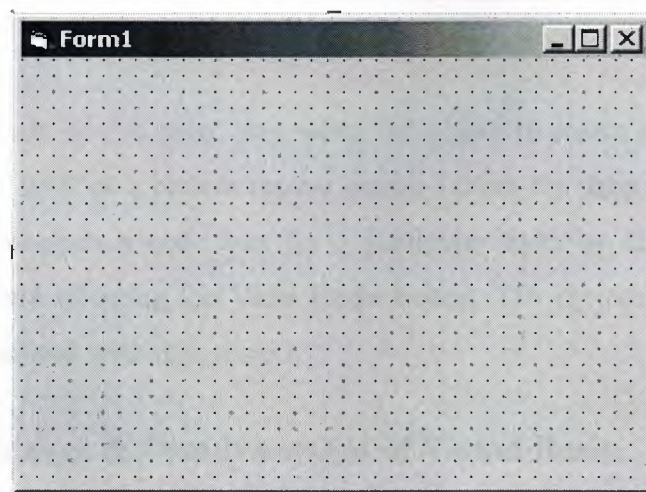


Figure 1.1. An Empty Form

This form will eventually be incorporated into our program as a window. To this form we add **controls**. Controls are things like text boxes, check boxes and command buttons...etc.

We can add **events** to our controls. Events are responses to actions performed on controls. For example, in the "Hello world" program sample on this page, when you click on the command button on our form the event that is triggered is the output of the message "Hello world" to the screen. Code must be written to create an event. we can do this in Visual Basic's **code window**.

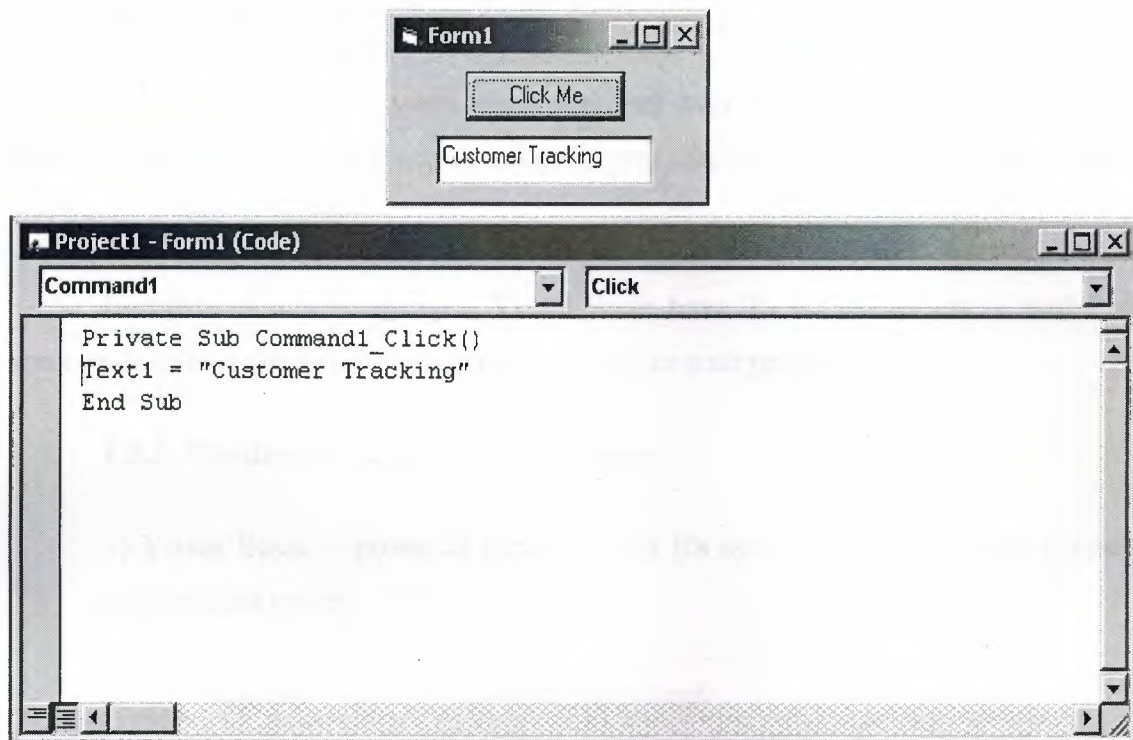


Figure 1.2. A Simple Program

Once the code box is open, you select the object to create an event for and the triggering action (such as a certain mouse action) from the drop down menus in the code box. You can open a code box for a particular form by choosing it from the **project window** and selecting the **View Code** button. The project window contains a list of objects associated with that project.

1.3. The Advantages & Disadvantages of Visual Basic.

1.3.1. The Advantages of Visual Basic:

1) It's simple language. Things that may be difficult to program with other language, can be done in Visual Basic very easily.

2) Because Visual Basic is so popular, There are many good resources (Books, Web sites, News groups and more) that can help you to learn the language. You can find the answers to your programming problems much more easily than other programming languages.

3) You can find many tools (Sharewares and Freewares) on the internet that will spare you some programming time.

For example, if you want to ping a user over the internet in your program, Instead of writing the ping function yourself, you can download a control that does it, and use it in your program.

Compare to other languages, Visual Basic have the widest variety of tools that you can download from the internet and use them in your programs.

1.3.2. The disadvantages of Visual Basic:

1) Visual Basic is powerful language, but it's not suit for programming really sophisticated games.

2) It's much more slower than other languagues.

1.4. Areas of Application

The term "*Personal Programming*" refers to the idea that, wherever we work, whatever we do, we can expand our computer's usefulness by writing applications to use in our own job. Personal Programming is what Visual Basic is all about.

Using Visual Basic's tools, we quickly translate an abstract idea into a program design, we can actually see on the screen. VB encourages us to experiment, revise, correct, and network your design until the new project meets our requirements. However , most of all, it inspires our imagination and creativity.

Visual Basic is ideal for developing applications that run in the new Windows 95 operating system.

VB presents a 3-step approach for creating programs:

1. Design the appearance of application.
2. Assign property settings to the objects of program.
3. Write the code to direct specific tasks at runtime.

Visual Basic can be used in a number of different areas, for example:

- Education
- Research
- Medicine
- Business
- Commerce
- Marketing and Sales
- Accounting
- Consulting
- Law
- Science

1.5. The Development Environment

Learning the ins and outs of the Development Environment before we learn visual basic is somewhat like learning for a test, we must know where all the functions belong and what their purpose is. First we will start with labelling the development environment.

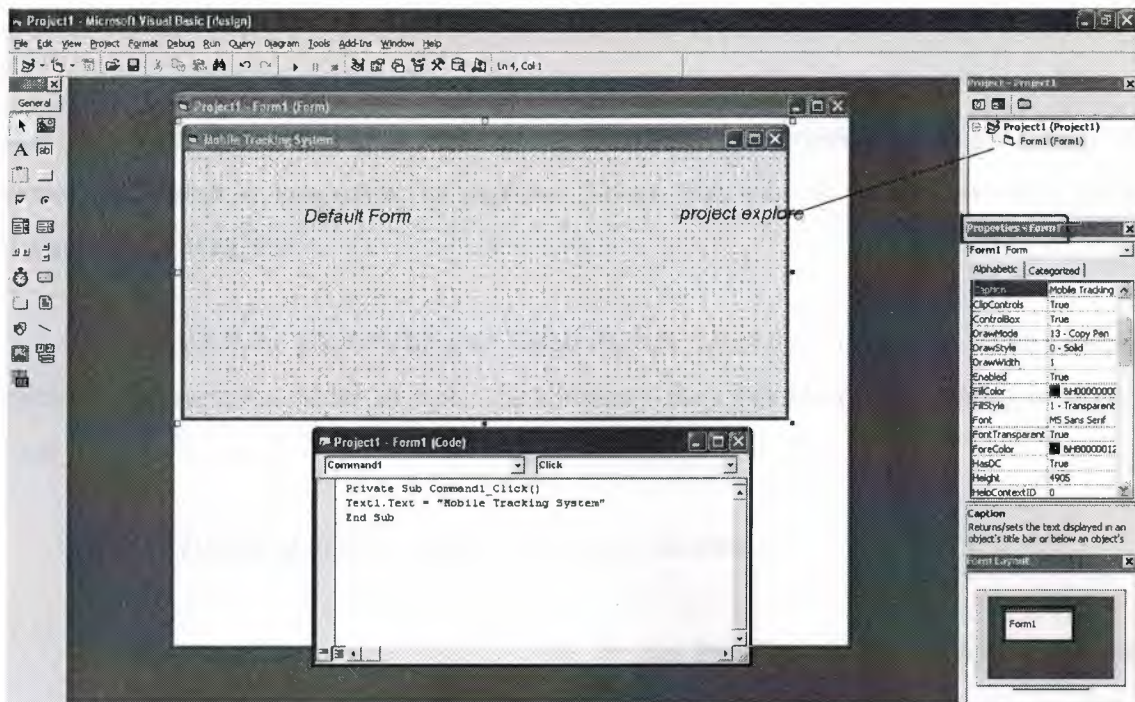


Figure 1.3. Development Environment

The above diagram shows the development environment with all the important points labelled. Many of Visual basic functions work similar to Microsoft word eg the **Tool Bar** and the tool box is similar to other products on the market which work off a single click then drag the width of the object required. The **Tool Box** contains the control we placed on the form window. All of the controls that appear on the **Tool Box** controls on the above picture never runs out of controls as soon as we place one on the form another awaits us on the **Tool Box** ready to be placed as needed.

1.6. The Project Explorer Window




The Project explorer window gives us a tree-structured view of all the files inserted into the application. We can expand these and collapse branches of the views to get more or less detail (**Project explorer**). The project explorer window displays forms, modules or other separators which are supported by the visual basic like class'es and Advanced Modules. If we want to select a form on its own simply double click on the project explorer window for a more detailed look. And it will display it where the **Default form** is located.

1.6.1. Opening an Existing Visual Basic Project

Microsoft have included some freebies with visual basic to show its capabilities and functions. Dismantling or modifying these sample projects is a good way to understand what is happening at runtime. These files can be located at our default directory /SAMPLES/

To Open these projects choose 'Open Project' from the 'File' menu. Then Double click on the samples folder to open the directory then Double click on any project to load it.

1.6.2. Running and Viewing The Project in Detail

Once an application is loaded it can be run by click on the  icon from the toolbar, to pause press  and to terminate use .

Once a project is loaded, the name of the form(s) that it contains is displayed in the project window. To view a form in design mode, we select the form required by clicking with the mouse to highlight its name, then clicking on the view form button.

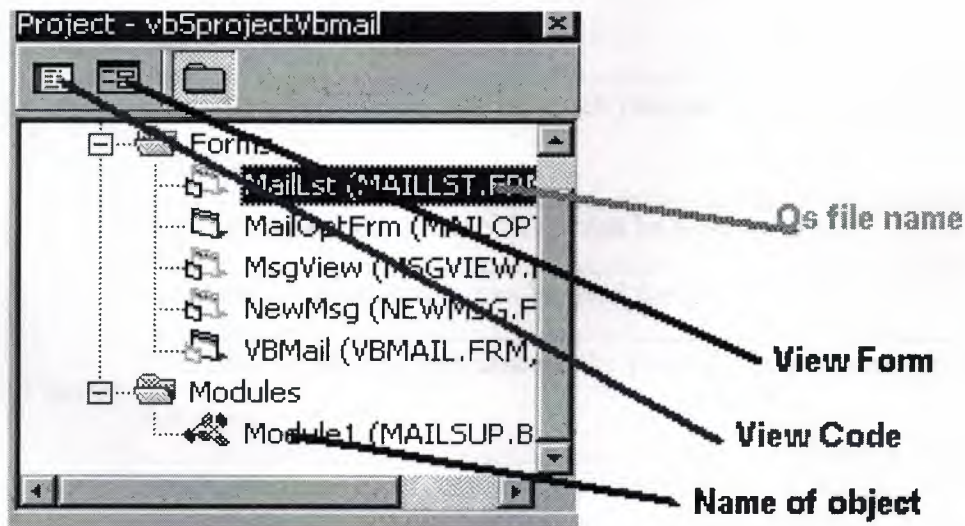


Figure 1.4. Project of the Program in Detail

Button Properties for reference

, Command Button &  labels properties

Property	Description
Name	The name of the object so you can call it at runtime
BackColor	This specifies the command button's background color. Click the <i>BackColor's</i> palette down arrow to see a list of common Windows control colours, you must change this to the style property from 0 - standard to 1 – graphical
Cancel	Determines whether the command button gets a Click event if the user presses escape
Caption	Holds the text that appears on the command button.
Default	Determines if the command button responds to an enter keypress even if another control has the focus
Enable	Determines whether the command button is active. Often, you'll change the enable property at runtime with code to prevent the user pressing the button
Font	Produces a Font dialog box in which you can set the caption's font name, style and size.
Height	Positions the height of the object - can be used for down
Left	Positions the left control - can be used for right
MousePointer	If selected to an icon can change the picture of the mouse pointer over that object
Picture	Hold's the name of an icon graphic image so that it appears as a picture instead of a Button for this option to work the graphical tag must be set to 1
Style	This determines if the Command Button appears as a standard windows dialog box or a graphical image
Tab index	Specifies the order of the command button in tab order
Tab Stop	Whether the object can be tabbed to (this can be used in labels which

	have no other function)
Tool Tip Text	If the mouse is held over the object a brief description can be displayed (for example hold your mouse over one of the above pictures to see this happening
Visible	If you want the user to see the button/label select true other wise just press false
Width	Show the width of the object

1.6.3.Properties Window

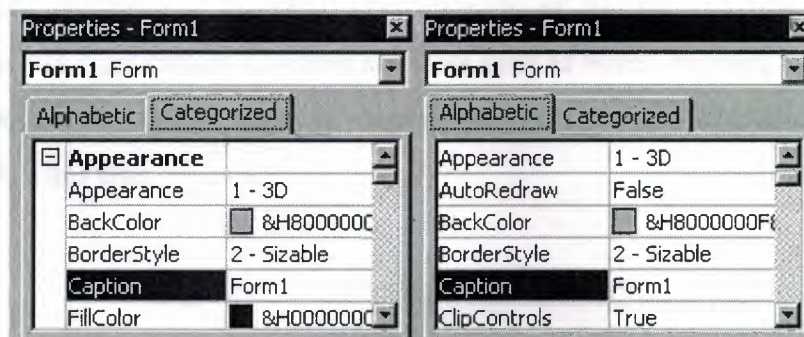


Figure 1.5. Properties Window

Some programmers prefer the Categorizised view of the properties window. By defaulting, the properties window displays its properties alphabetically (with the exception of the name value) when we click on the categorized button the window changes to left picture.

1.6.4.The Default Layout

When we start Visual Basic, we are provided with a VB project. A VB project is a collection of the following modules and files.

- The **global module**(*that contains declaration and procedures*)
- The **form module**(*that contains the graphic elements of the VB application along with the instruction*)

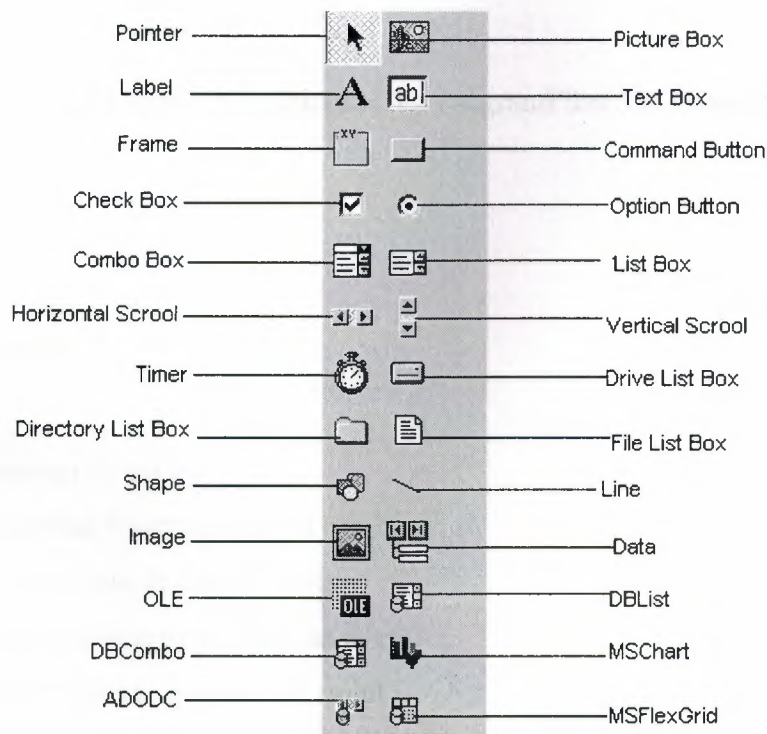
- The **general module** (that generally contains general-purpose instructions not pertaining to anything graphic on-screen)
- The **class module** (that contains the defining characteristics of a class, including its properties and methods)
- The **resource files** (that allows you to collect all of the texts and bitmaps for an application in one place)


On start up, Visual Basic will displays the following windows :

- The **Blank Form** window
- The **Project** window
- The **Properties** window

It also includes a **Toolbox** that consists of all the controls essential for developing a VB Application. Controls are tools such as boxes, buttons, labels and other objects draw on a form to get **input** or **display output**. They also add visual appeal.

1.6.5. Tool Box



When we click on different controls the Properties Window changes slightly this is due to different controls having different functions. Therefore more options are needed, for example if we had a picture then we want to show an image. But if we wanted to open an internet connection we would have to fill in the remote host and other such settings. When we use the command () we will find that a new set of properties come up the following will provide a description and a property.

1.6.6. Opening a New Visual Basic File & Inserting Source Code

When we open the program we choose 'New Project' from the 'File' menu. We use the blank form1 to design a simple interface for an estate agents database, have some textboxes for names and other details. We insert some controls and make it look professional. Textboxes can be used to store their name and other details, we make sure that we put a picture box in for a picture of the house.

Later we insert the following source code for application.

Private Sub

```
Form_Load()Picture1.Picture=LoadPicture("C:\ProgramFiles\VB\Graphics\Icons\Misc\MISC42.ICO")
```


End Sub

1.7. The Events

- Know what an Event is.
- Determine what Events a control can have
- Write code for one or more Events.
- Using optionbuttons to produce an event
- Using checkboxes to produce an event
- Grouping controls using a frame
- Make a simple alteration to the interface, such as changing background colour, at run time.

- Creating a listbox.
- Remove and Add listboxes functions.
- Creating Combo Boxes
- What the different types of combo boxes are.

1.7.1. What an Event ?

The 'look' of a Visual Basic application is determined by what controls are used, but the 'feel' is determined by the events. An event is something which can happen to a control. For example, a user can click on a button, change a text box, or resize a form. As explained in Creating a Visual Basic Application, writing a program is made up of three events: 1) select suitable controls, 2) set the properties, and 3) write the code. It is at the code writing stage when it becomes important to choose appropriate events for each control. To do this double click on the control the event will be used for, or click on the  icon in the project window (usually top right of screen). A code window should now be displayed similar to the one shown below.

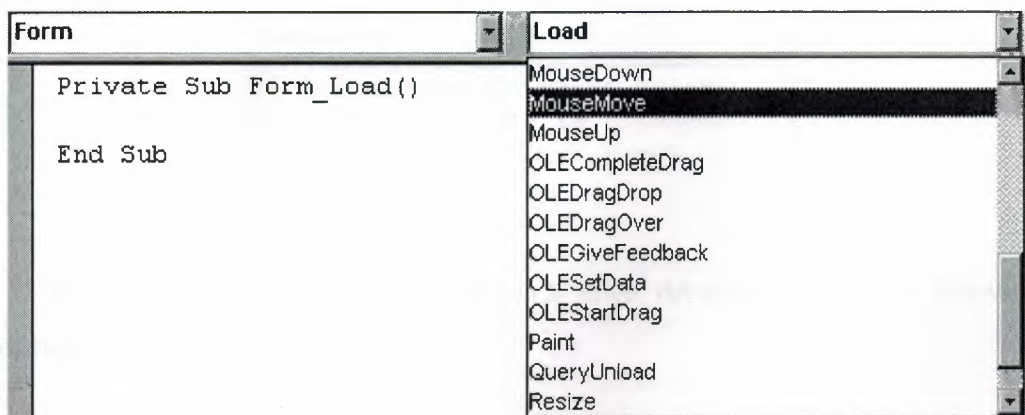


Figure 1.6. Form & Load

The left hand dropdown box provides a list of all controls used by the current form, the form itself, and a special section called General Declarations. The corresponding dropdown box on the right displays a list of all events applicable to the current control (as specified by the left hand dropdown box). Events displayed in bold signify that code has already been written for them, unbold events are unused. To demonstrate that different events can play a significant role in determining the feel of an application, a small example program will be written to add two numbers together and

display the answer. The first solution to this problem will use the click event of a command button, while the second will use the change event of two text boxes.

1.7.2. Click Event

Before any events can be coded it is necessary to design the interface from suitable controls. As shown in the screen shot below use: 2 text boxes to enter the numbers, a label for the '+' sign, a command button for the '=' sign, and another label for the answer.

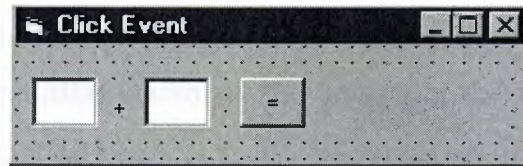
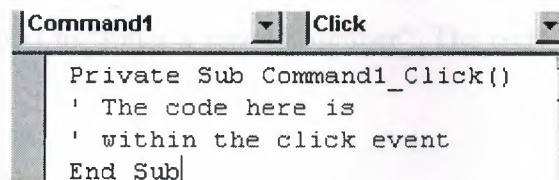


Figure 1.7. Click Event Form

Making the click event is very simple just select the button with the mouse and double click visual basic will generate



You can see on the top right there is a 'click' dropdown list this is known as an event handler.

1.7.3. Writing a Code

In the first example the user enters two numbers and then clicks on the equals button to produce an answer. However, the program can be changed so that the answer will be calculated every time either of the two numbers are changed without requiring an equals button.

Private Sub txtNumber1_Change()

label2.Caption = Str\$(Val(text1.Text) + Val(text2.Text))

End Sub


Private Sub txtNumber2_Change()

label2.Caption = Str\$(Val(text1.Text) + Val(text2.Text))

End Sub

When we run the program, we enter two numbers and observe what happens. Each time a digit changes the answer is recalculated.

1.7.4. Using The Event GotFocus

So far only one event has been used per control, however this does not have to be the case! Add a StatusBar control to the bottom of the form, bring up the code window using , select the first text box (txtNumber1) from the left hand dropdown box, and then select the **GotFocus** event from the right hand dropdown box. Now some basic instructions can be written in the status bar so that when the cursor is in the text box (the text box has focus) the status bar reads "Enter the first number". After completing this change to the second text box and using the same GotFocus event change the statusbar text to "Enter a second number". The code to set the status bar can be seen below.

1.7.5. CheckBoxes

Option bars are used quite often in the windows environment as they can only have two outputs 0 and 1 these get used to process the form. In this example it will be used to change the some text from normal to bold or to italic.

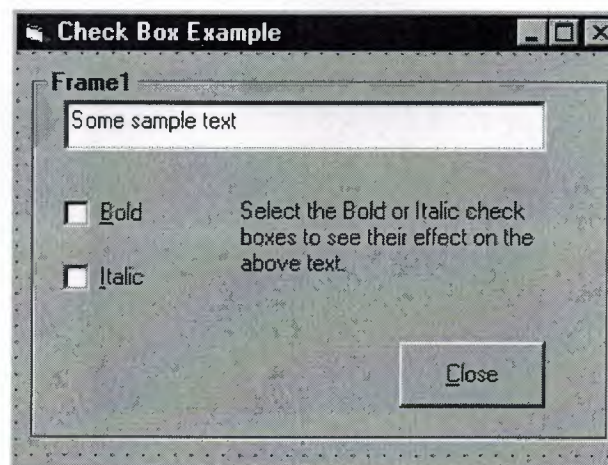


Figure 1.8. Check Box Example

Private Sub chkBold_Click()

If chkBold.Value = 1 Then ' If checked.

txtDisplay.FontBold = True

Else ' If not checked.

txtDisplay.FontBold = False

End If

End Sub

Private Sub chkItalic_Click()

If chkItalic.Value = 1 Then ' If checked.

txtDisplay.FontItalic = True

Else ' If not checked.

txtDisplay.FontItalic = False

End If

End Sub

1.7.6.Option Buttons

Changing the background colour

Changing the background colour gets used mostly by freeware, or the type of programs we generate for use with banners or adverts, anyway it might come in useful sometime. This example shows an ordinary picture of a smiley face then I put in some nice background colours to make it stand out more.

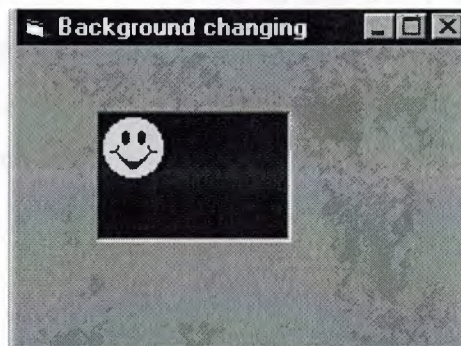


Figure 1.9. Back Ground Changing Example

Private Sub Form_Load()

BackColor = QBColor(Rnd * 15)



ForeColor = QBColor(Rnd * 10)

Picture1.BackColor = QBColor(Rnd * 15)

Picture1.ForeColor = QBColor(Rnd * 10)


End Sub

1.7.7. List Boxes

List boxes and combo boxes are used to supply a list of options to the user. The toolbox icons representing these two controls are  for list box and  for combo box.

A list box is used when the user is to be presented with a fixed set of selections (i.e. a choice must be made only from the items displayed, there is no possibility of the user typing in an alternative).

Examples might be offering a list of the days in a week, the available modules in an elective catalogue, the holiday destinations available from a particular airport or the types of treatment offered by a beauty salon.

To create a list box, double click on the toolbox icon . Drag the resulting box into place on the form and size it according to the data it will hold. The left hand picture below shows a list box that has been created and sized on Form1. In the middle is the code that is required to load a selection of cars into the list. The data has been included in the procedure Sub Form_Load so that it appears when Form1 is loaded. Finally, the picture on the right shows what appears on the form when the application is run. Notice that vertical scroll bars are added automatically if the list box is not deep enough to display all the choices.

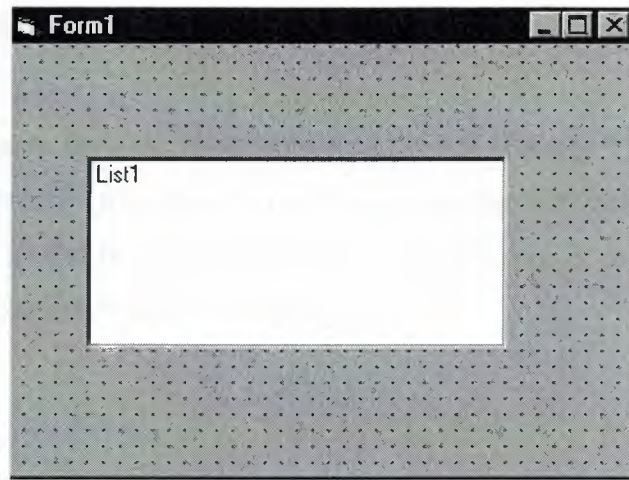


Figure 1.10. A list On Form

If we however add the following source code to this project

1.7.7.1. Add to a Lisbox

Private Sub Form_Load()

```
List1.AddItem "Monday"
```

```
List1.AddItem "Tuesday"
```

```
List1.AddItem "Wednesday"
```

```
List1.AddItem "Thursday"
```

```
List1.AddItem "Friday"
```

```
List1.AddItem "Saturday"
```

```
List1.AddItem "Sunday"
```

End Sub

The source code should look something like this :

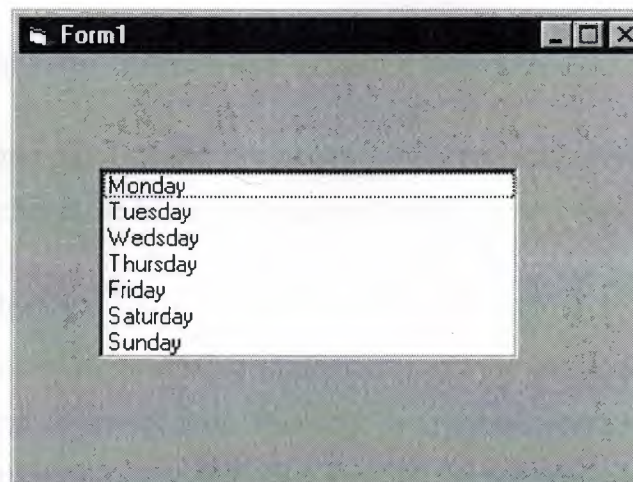


Figure 1.10.1. Adding To List

1.7.7.2.Removing & Advanced Options

They appear in the order they were typed if you changed the properties window by changing sort order = true then they will go into alphabetical order. List items can be added and deleted all the list is counted as the order they are in so for example if you wanted to delete "Tuesday" you would type

list1.RemoveItem 1

And to add to the list in a particular order just add

list1.AddItem "My Day", 5

This will be added after Saturday.

And finally to clear the listbox type

List1.clear This will completely clear the contents of the listbox.

The property ListCount stores the number of items in a list, so list1.ListCount can be used to determine the number of items in list box list1.

The property ListIndex gives the index of the currently selected list item. So the statement list1.RemoveItem List1.ListIndex removes the currently highlighted list item.

Adding an item can be accomplished very neatly using an input dialog box. Try this:

list1.AddItem InputBox("Enter a day", "Add a Day")

This will open a message box and prompt you for a new day to enter this will then be added to the list index at the bottom unless you specify where it should go.

1.7.8. MsgBoxes

Message boxes are used when you want to ask the user a question or display an error message(s) and advise the user. There are six types of message boxes here are their functions and what they do. Here is the listing of all the possible msgbox events

The Buttons displayed in a message here Button Layout	Value	Short Description
vbOKOnly	0	Displays the OK button.
vbOKCancel	1	Displays the ok and cancel button.
vbAbortRetryIgnore	2	Displays the Abort , Retry , Ignore
vbYesNoCancel	3	Displays Yes , No and Cancel button
vbYesNo	4	Displays the Yes / No button
vbRetryCancel	5	Displays the retry and Cancel buttons.

The Icons displayed in the message box are here

Icon on message	Value	Short Description
vbCritical	16	Displays critical message icon
vbQuestion	32	Displays question icon
vbExclamation	48	Displays exclamation icon
vbInformation	64	Displays information icon

The Default button displayed in a message form Default Button	Value	Short Description
vbDefaultButton1	0	Button 1 is default
vbDefaultButton2	256	Button 2 is default
vbDefaultButton3	512	Button 3 is default

Msgbox Return Value Return Value	Value	Short Description
--	-------	-------------------

vbOk	1	The User Clicked OK
vbCancel	2	The User Clicked Cancel
vbAbort	3	The User Clicked Abort
vbRetry	4	The User Clicked Retry
vbIgnore	5	The User Clicked Ignore
VbYes	6	The User Clicked Yes
VbNo	7	The User Clicked No

The syntax for use of the message box in MsgBox "TEXT", VALUE, "TITLE"
If you want to use two or more tables just add the values together. Therefore to print a message box to say "The Device was not Found!" OK & Explanation :

Source code 1

Private Sub Form_Load()

MsgBox "The Device was not Found!", 48, "Header"

End Sub

Source code 2

Private Sub Form_Load()

MsgBox "The Device was not found!", vbExclamation, "Header"

End Sub

we should get the picture shown below whatever source code you used.

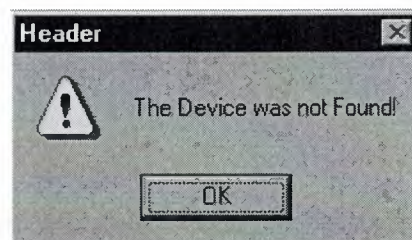


Figure 1.11. Showing message

This is a basic msgbox which in this case has not been processed in any way. The following Source code displays a msgbox that ask you for specific text. For example lets make a password program out of this message box.

Private Sub Form_Load()

```
lngBefore = Timer  
Do  
strAns = InputBox("What is the password Password is Example", "Password Required")  
Loop Until Val(strAns) = Example  
lngAfter = Timer  
msgbox "Correct Password", vbInformation
```

End Sub

Once you copy and paste the source code you should get prompted for a password in a different type of msgbox as it includes text. From looking at this example you should be able to see how the loop function works to generate and then check the value. All of the Return Values work in the same way to return an object input.

1.7.9.Opening & Retriving Information From Files

When applications are loaded they normal get some setting out of the registry or file this section will show you how to retrieve 1 string out of a file.

Private Sub Form_Load()

```
Dim F As Integer, password As String  
F = FreeFile  
Open App.Path & "\password.txt" For Input As F  
Input #F, password  
Close #F  
End Sub
```

As you can see from this source code the password is previously declared as a string. After this is done be sure to close the file otherwise next time you want to store

or read the file the computer will think it is being used by another application and windows will not let you do anything with it. So as you can see it is very important to close the file .

1.7.10. Storing Information to a File

FTP programs often store information to a file such as a username and password or host information in the same way. This following example will put some information into the file.

Private Sub Form_Load()

```
Dim F As Integer, pass As String
F = FreeFile
save = txtNew
Open App.Path & "\password.txt" For Output As F
Write #F, Text1.text
Close #F
```

End Sub

Although this is a bit obvious I think I should include it just incase I think differently to other people.

1.7.11. Printing Text to The Printer

This is rather easy to do and it gets used in notepad etc...

Private Sub Form_Load()

```
Printer.Print " The printer will print this text "
Printer.Print ""
Printer.Print " It will leave a line here"
Printer.Print " It should add the contence of text1.text here : " & Text1.Text & " As you
can see it works"
Printer.Print ""
Printer.EndDoc 'This will tell the printer it has finished.
```

End Sub

Everything that appears in position (A) will get printed by the default printer `printer.print " A "`. The `printer enddoc` is used to tell the printer the job is finished if this is not added the printer can not estimate how near it will be until it has finish and when it has finished it will think it has'nt so be sure to include this to prevent confusion.

1.8. Managing Data Types

There are many types of data we come across in our daily life. For example, we need to handle data such as names, addresses, money, date, stock quotes, statistics and etc everyday. Similarly in Visual Basic, we are also going to deal with these kinds of data. However, to be more systematic, VB divides data into different types.

1.8.1.Numeric Data Types

Numeric data are data that consists of numbers, which can be computed mathematically with various standard operators such as add, minus, multiply, divide and so on. In Visual Basic, the numeric data are divided into 7 types, they are summarized in Table1.1

Type	Storage	Range of Values
Byte	1 byte	0 to 255
Integer	2 bytes	-32,768 to 32,767
Long	4 bytes	-2,147,483,648 to 2,147,483,648
Single	4 bytes	-3.402823E+38 to -1.401298E-45 for negative values 1.401298E-45 to 3.402823E+38 for positive values.
Double	8 bytes	-1.79769313486232e+308 to -4.94065645841247E-324 for negative values 4.94065645841247E-324 to 1.79769313486232e+308 for positive values.

Table 1.1: Numeric Data Types

1.8.2.Non-numeric Data Types

The nonnumeric data types are summarized in Table1.2

Data Type	Storage	Range
String(fixed length)	Length of string	1 to 65,400 characters
String(variable length)	Length + 10 bytes	0 to 2 billion characters
Date	8 bytes	January 1, 100 to December 31, 9999
Boolean	2 bytes	True or False

Table 1.2: Nonnumeric Data Types

1.8.3.Declaring Variables

In Visual Basic, one needs to declare the variables before using them by assigning names and data types. They are normally declared in the general section of the code windows using the **Dim** statement.

The format is as follows:

Dim variableName as DataType

Example1.3

```
Dim password As String
Dim yourName As String
Dim firstnum As Integer
Dim secondnum As Integer
Dim total As Integer
Dim doDate As Date
```


You may also combine them in one line , separating each variable with a comma, as follows: Dim password As String, yourName As String, firstnum As Integer,.....

If.....Then.....Else Statements with Operators

To effectively control the VB program flow, we shall use If...Then...Else statement together with the conditional operators and logical operators. The general format for the if...then...else statement is

If conditions **Then**

VB expressions

Else

VB expressions

End If

Select Case

If you have a lot of conditional statements, using If..Then..Else could be very messy. For multiple conditional statements, it is better to use Select Case. The format is:

Select Case expression

Case value1

Block of one or more VB statements

Case value2

Block of one or more VB Statements

End Select

Looping

Visual Basic allows a procedure to be repeated as many times as long as the processor could support. This is generally called looping .

1. Do

Block of one or more VB statements

Loop While condition

2. Do

Block of one or more VB statements

Loop Until condition

3. For....Next Loop

For counter=startNumber to endNumber (Step increment)

One or more VB statements

Next

1.9.VB Functions

1.9.1.Creating Functions

The general format of a function is as follows:

Public Function **functionName** (Arg As dataType,.....) As dataType

or

Private Function **functionName** (Arg As dataType,.....) As dataType

*Public indicates that the function is applicable to the whole program and

Private indicates that the function is only applicable to a certain module or procedure.

1.9.2.Declaring Array

We could use Public or Dim statement to declare an array just as the way we declare a single variable. The Public statement declares an array that can be used throughout an application while the Dim statement declare an array that could be used only in a local procedure. The general format to declare an array is as follow:

Dim arrayName(subs) as dataType

where subs indicates the last subscript in the array.

Example 13.1

Dim FindName(10) as String

1.10.Database Application

Finally, I want to put end point in that part, with telling about database application. Of course, All topics is not restricted what I emphasize in that part.

Visual basic allows us to manage databases created with different database program such as MS Access, Dbase, Paradox and etc. In this lesson, we are not dealing with how to create database files but we will see how we can access database files in the VB environment.

CHAPTER II: DATABASE STRUCTURE

2.1.What is Microsoft SQL Server?

Microsoft SQL Server 2000 is a full-featured relational database management system (RDBMS) that offers a variety of administrative tools to ease the burdens of database development, maintenance and administration. In this article, we'll cover six of the more frequently used tools: Enterprise Manager, Query Analyzer, SQL Profiler, Service Manager, Data Transformation Services and Books Online. Let's take a brief look at each:

Enterprise Manager is the main administrative console for SQL Server installations. It provides you with a graphical "birds-eye" view of all of the SQL Server installations on your network. You can perform high-level administrative functions that affect one or more servers, schedule common maintenance tasks or create and modify the structure of individual databases.

Query Analyzer offers a quick and dirty method for performing queries against any of your SQL Server databases.

It's a great way to quickly pull information out of a database in response to a user request, test queries before implementing them in other applications, create/modify stored procedures and execute administrative tasks.

SQL Profiler provides a window into the inner workings of your database. You can monitor many different event types and observe database performance in real time. SQL Profiler allows you to capture and replay system "traces" that log various activities. It's a great tool for optimizing databases with performance issues or troubleshooting particular problems.

Service Manager is used to control the MSSQLServer (the main SQL Server process), MSDTC (Microsoft Distributed Transaction Coordinator) and SQLServerAgent processes. An icon for this service normally resides in the system tray of machines running SQL Server. You can use Service Manager to start, stop or pause any one of these services.

Data Transformation Services (DTS) provide an extremely flexible method for importing and exporting data between a Microsoft SQL Server installation and a large variety of other formats. The most commonly used DTS application is the "Import and Export Data" wizard found in the SQL Server program group.

2.2.Creating the Database

2.2.1.Choosing a Database Platform

We've decided to use a database management system (or DBMS) that is built upon the Structured Query Language (SQL). Therefore, all of our database and table creation commands should be written with standard ANSI SQL in mind. As an added benefit, using ANSI-compliant SQL will ensure that these commands will work on any DBMS that supports the SQL standard, including Oracle and Microsoft SQL Server. If you haven't selected a platform for your database yet, the article Database Software Options walks you through the selection process.

2.2.2.Creating the Database

Our first step is to create the database itself. Many database management systems offer a series of options to customize database parameters at this step, but our database only permits the simple creation of a database. As with all of our commands, you may wish to consult the documentation for your DBMS to determine if any advanced parameters supported by your specific system meet your needs. Let's use the CREATE DATABASE command to set up our database:

```
CREATE DATABASE personnel
```

Take special note of the capitalization used in the example above. It's common practice among SQL programmers to use all capital letters for SQL keywords such as "CREATE" and "DATABASE" while using all lowercase letters for user-defined names like the "personnel" database name. These conventions provide for easy readability.

2.3.Creating Tables

Now that we've designed and created our database, we're ready to begin creating the three tables used to store XYZ Corporation's personnel data. We'll be implementing the tables we designed in the previous portion of this tutorial.

2.3.1.Creating Our First Table

Our first table consists of the personal data for each employee of our company. We need to include each employee's name, salary, ID and manager. It's good design practice to separate the last and first names into separate fields to simplify data searching and sorting in the future. Also, we'll keep track of each employee's manager

by inserting a reference to the manager's employee ID in each employee record. Let's first take a look at the desired employee table.

The ReportsTo attribute stores the manager ID for each employee.

From the sample records shown, we can determine that Sue Scampi is the manager of both Tom Kendall and John Smith. However, there is no information in the database on Sue's manager, as indicated by the NULL entry in her row.

Now we can use SQL to create the table in our personnel database. Before we do so, let's ensure that we are in the correct database by issuing a USE command:

USE personnel;

Alternatively, the "DATABASE personnel;" command would perform the same function. Now we can take a look at the SQL command used to create our employees table:

```
CREATE      TABLE      employees
(employeeid INTEGER NOT NULL,
lastname VARCHAR(25) NOT NULL,
firstname VARCHAR(25) NOT NULL,
reportsto INTEGER NULL);
```

As with our previous example, note that programming convention dictates that we use all capital letters for SQL keywords and lowercase letters for user-named columns and tables. The command above may seem confusing at first, but there's actually a simple structure behind it. Here's a generalized view that might clear things up a bit:

```
CREATE TABLE table_name
(attribute_name datatype options,
...,
attribute_name datatype options);
```

2.3.2.Attributes and Data Types

In the previous example, the table name is employees and we include four attributes: employeeid, lastname, firstname, and reportsto. The datatype indicates the type of information we wish to store in each field. The employee ID is a simple integer number, so we'll use the INTEGER datatype for both the employeeid field and the

reportsto field. The employee names will be character strings of variable length and we don't expect any employee to have a first or last name longer than 25 characters. Therefore, we'll use the VARCHAR(25) type for these fields.

2.3.3.NULL Values

We can also specify either NULL or NOT NULL in the options field of the CREATE statement. This simply tells the database whether NULL (or empty) values are allowed for that attribute when adding rows to the database. In our example, the HR department requires that an employee ID and complete name be stored for each employee. However, not every employee has a manager -- the CEO reports to nobody! - so we allow NULL entries in that field. Note that NULL is the default value and omitting this option will implicitly allow NULL values for an attribute.

2.3.4.Building The Remaining Tables

Now let's take a look at the territories table. From a quick look at this data, it appears that we need to store an integer and two variable length strings. As with our previous example, we don't expect the Region ID to consume more than 25 characters. However, some of our territories have longer names, so we'll expand the allowable length of that attribute to 40 characters.

Let's look at the corresponding SQL:

CREATE	TABLE	territories
(territoryid	INTEGER	NOT NULL,
territory Description	VARCHAR(40) NOT NULL,	
regionid	VARCHAR(25) NOT NULL);	

Finally, we'll use the EmployeeTerritories table to store the relationships between employees and territories. Detailed information on each employee and territory is stored in our previous two tables. Therefore, we only need to store the two integer identification numbers in this table. If we need to expand this information we can use a JOIN in our data selection commands to obtain information from multiple tables. This method of storing data reduces redundancy in our database and ensures optimal use of

space on our storage drives. We'll cover the JOIN command in-depth in a future tutorial. Here's the SQL code to implement our final table:

```
CREATE TABLE employeeterritories  
(employeeid INTEGER NOT NULL,  
territoryid INTEGER NOT NULL);
```

2.3.5.Modifying Tables

In the previous two sections of this article we explored the design of a database and the creation of tables to populate that database with information. In this final segment, we'll look at the mechanism SQL provides to alter the structure of a database after creation.

If you're particularly astute today, you might have noticed that we "accidentally" omitted one of the design requirements when implementing our database tables. XYZ Corporation's HR Director requested that the database track employee salary information and we neglected to provide for this in the database tables we created.

However, all is not lost. We can use the ALTER TABLE command to add this attribute to our existing database. We want to store the salary as an integer value. The syntax is quite similar to that of the CREATE TABLE command, here it is:

```
ALTER                TABLE                employees  
ADD salary INTEGER NULL;
```

Notice that we specified that NULL values are permitted for this attribute.

In most cases there is no option when adding a column to an existing table. This is due to the fact that the table already contains rows with no entry for this attribute. Therefore, the DBMS automatically inserts a NULL value to fill the void.

And that wraps up our look at the SQL database and table creation process. Check back often for new installments in our SQL tutorial series. If you'd like an e-mail reminder when new articles are added to the About Databases site, be sure to subscribe to our newsletter!

Column Name	Data Type	Length	Allow Nulls
ID	int	4	
UserID	nvarchar	12	✓
Password	nvarchar	8	✓
Name	char	10	
Surname	char	16	
Address	nvarchar	50	
Tel1	nvarchar	11	
Tel2	nvarchar	11	✓
KimlikNo	nvarchar	11	
Statu	nvarchar	3	✓

Property	Value
Description	
Default Value	
Precision	10
Scale	0
Identity	Yes (Not For Replication)
Identity Seed	1
Identity Increment	1
Is RowGuid	No
Formula	
Collation	

Table 2.1 Person Table As an Eample in SQL Server.

2.4.Built-in Data Types

In Microsoft SQL Server 2000, each object (such as column, variable, or parameter) has a related data type, which is an attribute that specifies the type of data that the object can hold.

SQL Server 2000 ships with 27 built-in (system) data types. They are:

Data Types	Description
bigint	Integer data from -2^{63} through $2^{63}-1$
int	Integer data from -2^{31} through $2^{31} - 1$
smallint	Integer data from -2^{15} through $2^{15} - 1$
tinyint	Integer data from 0 through 255

bit	Integer data with either a 1 or 0 value
decimal	Fixed precision and scale numeric data from $-10^{38} + 1$ through $10^{38} - 1$
numeric	Fixed precision and scale numeric data from $-10^{38} + 1$ through $10^{38} - 1$
money	Monetary data values from -2^{63} through $2^{63} - 1$
smallmoney	Monetary data values from -214,748.3648 through +214,748.3647
float	Floating precision number data from $-1.79E + 308$ through $1.79E + 308$
real	Floating precision number data from $-3.40E + 38$ through $3.40E + 38$
datetime	Date and time data from January 1, 1753, through December 31, 9999, with an accuracy of 3.33 milliseconds
smalldatetime	Date and time data from January 1, 1900, through June 6, 2079, with an accuracy of one minute
char	Fixed-length character data with a maximum length of 8,000 characters
varchar	Variable-length data with a maximum of 8,000 characters
text	Variable-length data with a maximum length of $2^{31} - 1$ characters
nchar	Fixed-length Unicode data with a maximum length of 4,000 characters
nvarchar	Variable-length Unicode data with a maximum length of 4,000 characters
ntext	Variable-length Unicode data with a maximum length of $2^{30} - 1$ characters
binary	Fixed-length binary data with a maximum length of 8,000 bytes
varbinary	Variable-length binary data with a maximum length of 8,000 bytes

image	Variable-length binary data with a maximum length of $2^{31} - 1$ bytes
cursor	A reference to a cursor
sql_variant	A data type that stores values of various data types, except text, ntext, timestamp, and sql_variant
table	A special data type used to store a result set for later processing
timestamp	A database-wide unique number that gets updated every time a row gets updated
uniqueidentifier	A globally unique identifier

Table 2.2 Data Types.

Some of these data types (bigint, sql_variant, and table) are only available in SQL Server 2000, while some were supported under the previous SQL Server versions.

User-defined data types

SQL Server 2000 supports user-defined data types too. User-defined data types provide a mechanism for applying a name to a data type that is more descriptive of the types of values to be held in the object. Using user-defined data type can make it easier for a programmer or database administrator to understand the intended use of any object defined with the data type. The user-defined data types are based on the system data types and can be used to predefine several attributes of a column, such as its data type, length, and whether it supports NULL values. To create a user-defined data type, you can use the **sp_addtype** system stored procedure or you could add one using the Enterprise Manager. When you create a user-defined data type, you should specify the following three properties:

- Data type's name.
- Built-in data type upon which the new data type is based.
- Whether it can contain NULL values.

The following example creates a user-defined data type based on money data type named **cursale** that cannot be NULL:

```
EXEC sp_addtype cursale, money, 'NOT NULL'  
GO
```

Both system and user-defined data types are used to enforce data integrity. It is very important that we put forth a lot of effort while designing tables: the better you design your tables, the more time you can work without any performance problems. In an ideal case, you never will update the structure of your tables.

2.5. Microsoft SQL Server Constraints

A constraint is a property assigned to a column or the set of columns in a table that prevents certain types of inconsistent data values from being placed in the column(s). Constraints are used to enforce the data integrity. This ensures the accuracy and reliability of the data in the database. The following categories of the data integrity exist:

- Entity Integrity
- Domain Integrity
- Referential integrity
- User-Defined Integrity

Entity Integrity ensures that there are no duplicate rows in a table.

Domain Integrity enforces valid entries for a given column by restricting the type, the format, or the range of possible values.

Referential integrity ensures that rows cannot be deleted, which are used by other records (for example, corresponding data values between tables will be vital).

User-Defined Integrity enforces some specific business rules that do not fall into entity, domain, or referential integrity categories.

Each of these categories of the data integrity can be enforced by the appropriate constraints. Microsoft SQL Server supports the following constraints:

- PRIMARY KEY
- UNIQUE
- FOREIGN KEY

- CHECK
- NOT NULL

A **PRIMARY KEY** constraint is a unique identifier for a row within a database table. Every table should have a primary key constraint to uniquely identify each row and only one primary key constraint can be created for each table. The primary key constraints are used to enforce entity integrity.

A **UNIQUE** constraint enforces the uniqueness of the values in a set of columns, so no duplicate values are entered. The unique key constraints are used to enforce entity integrity as the primary key constraints.

A **FOREIGN KEY** constraint prevents any actions that would destroy link between tables with the corresponding data values. A foreign key in one table points to a primary key in another table. Foreign keys prevent actions that would leave rows with foreign key values when there are no primary keys with that value. The foreign key constraints are used to enforce referential integrity.

A **CHECK** constraint is used to limit the values that can be placed in a column. The check constraints are used to enforce domain integrity.

A **NOT NULL** constraint enforces that the column will not accept null values. The not null constraints are used to enforce domain integrity, as the check constraints.

You can create constraints when the table is created, as part of the table definition by using the **CREATE TABLE** statement.

Examples

The following example creates a check_sale CHECK constraint on an employee table:

```
CREATE TABLE employee(  
    EmployeeId INT NOT NULL,  
    LName VARCHAR(30) NOT NULL,  
    FName VARCHAR(30) NOT NULL,
```

```
Address VARCHAR(100) NOT NULL,  
HireDate DATETIME NOT NULL,  
Salary MONEY NOT NULL CONSTRAINT check_sale CHECK (salary > 0)  
)
```

You can add constraints to an existing table by using the ALTER TABLE statement. The following example adds a pk_employee primary key constraint on an employee table:

```
ALTER TABLE employee  
ADD CONSTRAINT pk_employee PRIMARY KEY (EmployeeId)
```

You can add the primary or unique key constraint into an existing table only when there are no duplicate rows in the table. You can drop constraints in an existing table by using the ALTER TABLE statement. The following example drops the pk_employee primary key constraint in the employee table:

```
ALTER TABLE employee  
DROP CONSTRAINT pk_employee
```

Sometimes you need to perform some actions that require the FOREIGN KEY or CHECK constraints be disabled, for example, your company do not hire foreign employees, you made the appropriate constraint, but the situation was changed and your boss need to hire the foreign employee, but only this one. In this case, you need to disable the constraint by using the ALTER TABLE statement. After these actions will be performed, you can re-enable the FOREIGN KEY and CHECK constraints by using the ALTER TABLE statement.

The following example disables the check_sale constraint in the employee table and enables this constraint later:

```
-- disable the check_sale constraint in the employee table  
ALTER TABLE employee NOCHECK CONSTRAINT check_sale
```



```
-- enable the check_sale constraint in the employee table  
ALTER TABLE employee CHECK CONSTRAINT check_sale
```

2.6. Working with SQL

SQL (pronounced “ess-que-el”) stands for structured Query Language. SQL is used to communicate with a database. According to ANSI (American National Standards Institute), it is the standard language for relational database management systems. SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database. Some common relational database management systems that use SQL are: Oracle, Sybase, Microsoft SQL Server, Access, Ingress, etc. although most database systems use SQL, most of them also have their own additional proprietary extensions that are usually only used on their system. However, the standard SQL commands such as “select”, “insert”, “delete”, “create” and “drop” can be used to accomplish almost everything that one needs to do with a database.

2.7. Data Manipulation Language

The Data Manipulation Language (DML) is used to retrieve, insert and modify database information. These commands will be used by all database users during the routine operation of the database. Let's take a brief look at the basic DML commands:

The Data Manipulation Language (DML) is used to retrieve, insert and modify database information. These commands will be used by all database users during the routine operation of the database. Let's take a brief look at the basic DML commands:

2.7.1.INSERT

The INSERT command in SQL is used to add records to an existing table. Returning to the personal_info example from the previous section, let's imagine that our HR department needs to add a new employee to their database. They could use a command similar to the one shown below:


```
INSERT INTO department  
VALUES('c01','computer','engineering',5,6)
```

Note that there are four values specified for the record.

These correspond to the table attributes in the order they were defined: first_name, last_name, employee_id, and salary.

2.7.2.SELECT

The SELECT command is the most commonly used command in SQL. It allows database users to retrieve the specific information they desire from an operational database. Let's take a look at a few examples, again using the personal_info table from our employees database.

The command shown below retrieves all of the information contained within the personal_info table. Note that the asterisk is used as a wildcard in SQL. This literally means "Select everything from the personal_info table."

```
SELECT *  
FROM student
```

Alternatively, users may want to limit the attributes that are retrieved from the database. For example, the Human Resources department may require a list of the last names of all employees in the company. The following SQL command would retrieve only that information:

```
SELECT stname  
FROM student WHERE stno=20020872
```

2.7.3.UPDATE

The SQL UPDATE clause serves to update data in database table. The SQL UPDATE clause basic syntax looks like this:

```
UPDATE Table1  
SET Column1 = Value1, Column2 = Value2,
```

The first line of the above SQL UPDATE statement defines which table we are updating. The second line starts with the SET SQL keyword followed by one or more Column = Value pairs separated by commas. The second line of the UPDATE statement defines which table columns to update and with what value.

Please consider the following SQL UPDATE syntax:

```
UPDATE Weather  
SET AverageTemperature = 20
```

2.7.4.DELETE

The **SQL DELETE** clause is used to delete data from a database table. The simplest SQL DELETE syntax looks like this:

```
DELETE FROM Table1
```

The SQL DELETE statement above will delete all data from the Table1 table. If we wanted to delete all rows containing Weather data for New York, we would use the following SQL DELETE statement:

```
DELETE FROM Weather  
WHERE City = 'New York'
```

Be extremely careful when using SQL DELETE, as you cannot restore data once you delete it from the table. You might want to make a backup of important data before performing delete on it.

2.8.Tables in The Project

These are the databases that I use in my project.

Design Table 'Firm' in 'proje' on '(local)'				
	Column Name	Data Type	Length	Allow Nulls
PK	FirmID	int	4	
	FirmName	nvarchar	50	
	Tel	nvarchar	11	✓
	Address	nvarchar	50	✓
	Fax	nvarchar	11	✓
	Mail	nvarchar	50	✓
	Web	nvarchar	20	✓

Table 2.3 Firm Database Table.

Design Table 'Item' in 'proje' on '(local)'				
	Column Name	Data Type	Length	Allow Nulls
PK	ItemNo	int	4	
	Item	nvarchar	20	
	Quantity	int	4	
	Price	real	4	
	[Date]	nvarchar	10	
	Unitcost	real	4	✓
	Profit	real	4	✓
	Salescost	real	4	✓
	Company	nvarchar	20	✓
	BillNo	int	4	✓

Table 2.4 Item Database Table.

Design Table 'Receipt' in 'proje' on '(local)'				
	Column Name	Data Type	Length	Allow Nulls
PK	BillNo	int	4	
	Surname	nvarchar	10	
	Payment	real	4	
	Debt	real	4	
	Datee	nvarchar	20	✓

Table 2.5 Receipt Database Table.

Design Table 'Receiptto' in 'proje' on '(local)'				
	Column Name	Data Type	Length	Allow Nulls
▶	BillNo	int	4	
	Company	nvarchar	20	
	Payment	real	4	✓
	Debt	real	4	✓
	Datee	nvarchar	20	✓

Table 2.6 Receiptto Database Table.

Design Table 'bill' in 'proje' on '(local)'				
	Column Name	Data Type	Length	Allow Nulls
▶	BillNo	int	4	
	CustomerID	nvarchar	20	✓
	Company	nvarchar	30	✓
	Datee	nvarchar	12	
	Hourr	nvarchar	12	✓
	ItemNo	nvarchar	10	
	ItemName	nvarchar	20	
	Quantity	nvarchar	10	✓
	Price	nvarchar	50	✓

Table 2.7 bill Database Table.

Design Table 'BillNo' in 'proje' on '(local)'				
	Column Name	Data Type	Length	Allow Nulls
▶	BillNo	int	4	✓

Table 2.8 BillNo Database Table.

Design Table 'Person' in 'proje' on '(local)'				
	Column Name	Data Type	Length	Allow Nulls
PK	ID	int	4	
	UserID	nvarchar	12	✓
	Password	nvarchar	8	✓
	Name	char	10	
	Surname	char	16	
	Address	nvarchar	50	
	Tel1	nvarchar	11	
	Tel2	nvarchar	11	✓
	KimlikNo	nvarchar	11	
	Statu	nvarchar	3	✓

Table 2.9 Person Database Table.

Design Table 'liste' in 'proje' on '(local)'				
	Column Name	Data Type	Length	Allow Nulls
PK	ID	numeric	9	
	Name	char	10	
	Surname	char	10	
	Address	nvarchar	50	
	Telephone	nvarchar	12	
	TC_KimlikNo	nvarchar	12	✓

Table 2.10 Customer Database Table.

Design Table 'Debtpay' in 'proje' on '(local)'				
	Column Name	Data Type	Length	Allow Nulls
	Surname	nvarchar	20	
	Datee	nvarchar	10	✓
	BillNo	int	4	✓
	Payment	real	4	✓
	Balance	real	4	✓

Table 2.11 Debtpay Database Table.

CHAPTER III : BILLING AND PAYMENT SYSTEM PROGRAM

3.1. Login Page

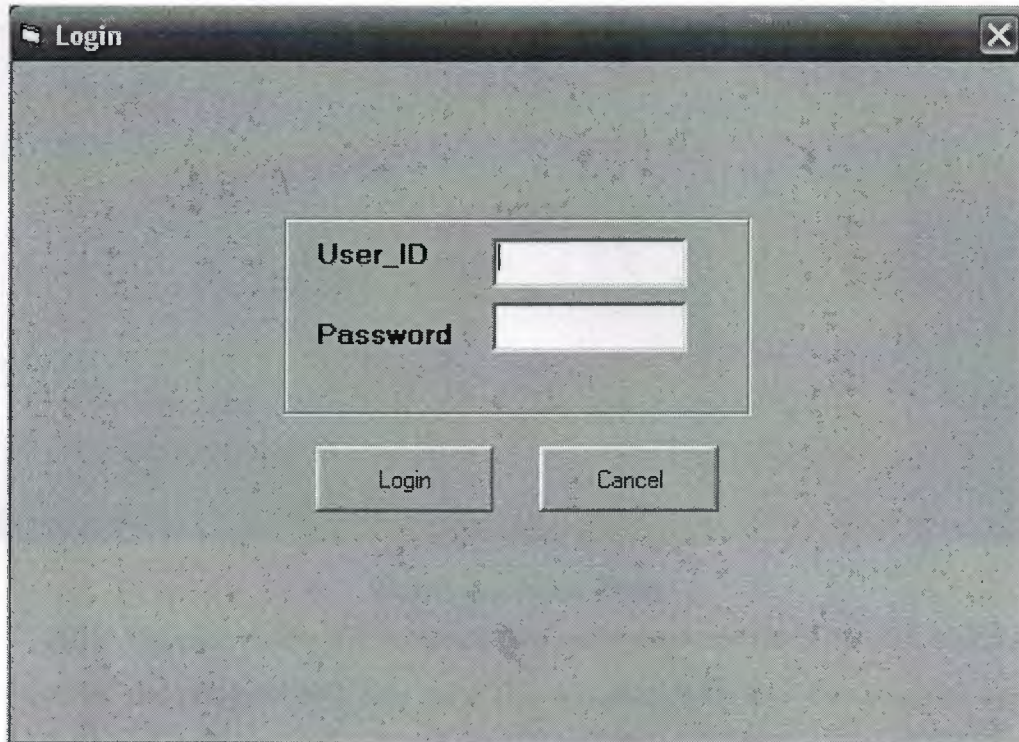


Figure 3.1 Login Page

When we run the program the login page appears and this page controls that the user can start the program or not. Also this page gives permissions to the user that which executions his/her can do.

3.2. Main Page

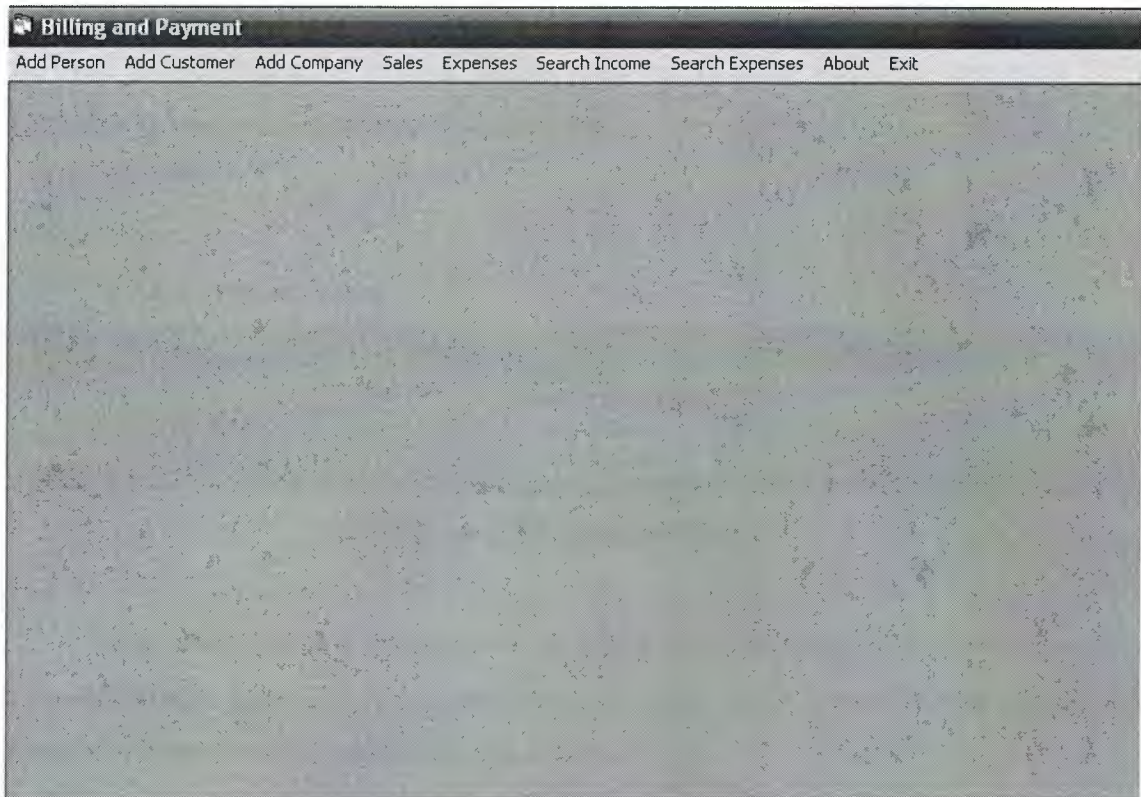


Figure 3.2 Main Page.

After the user enters the program this main page comes. The user controls the behaviour of the program with this page. The company who uses this program can add his logo, name and informations to the program on this page.

3.3 Menu Bar



Figure 3.3 Menu Bar.

This is the menu of the program.

3.3.1 Sales Menu

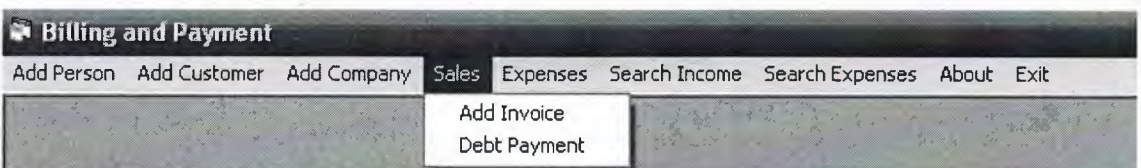


Figure 3.4 Sales Menu

This menu contains Add Invoice and Debt Payment menus. When the user press the Add Invoice menu the invoice adding page opens. Also when the user press Debt Payments menu the debt payments page opens.

3.3.2.Expenses Menu



Figure 3.5 Expenses Menu.

This menu contains Add Invoice and Debt Payment menus. When the user press the Add Invoice menu the invoice adding page opens. Also when the user press Debt Payments menu the debt payments page opens.

3.4 Add New Person Page

3.6 Add Person Page.

With this page the user can add new person that could be user or customer. Also user can update, search and delete person from the database.

3.5 Add New Company Page

The 'Add New Company' dialog box contains the following elements:

- Input fields for: Company ID, Company Name, Telephone, Fax, E-Mail, Web, and Address.
- A table with the following columns: FirmID, Firm Name, Telephone, Fax, E-Mail, Web, and Address.
- Buttons: Add, Update, Search, and Delete.
- A search section labeled 'By Name' with an input field.

Figure3.7 Add Company Page

The user can add company and its informations. Search the list of companies. If wanted , the company informations can be changed with update and also delete the company.

3.6 Invoice Page

The 'Invoice' dialog box contains the following elements:

- Fields for: Name, Company, Date (6/7/2008 18:36:59), and Bill No.
- A table with the following columns: Item No, Item, Quantity, Unit Cost, and Price.
- Buttons: Add, Go To Payment, and Cancel.

Figure 3.8 Invoice Page.

With this page the user controls the name of the customer and company, then add the items, quantities and prices that customer's bought things. The customer's bought things are added one by one. After that the user go to the Payment Page.

3.7 Payment Page

BillNo	CustomerID	Company	Datee	Item Name	Quantity	Price
--------	------------	---------	-------	-----------	----------	-------

Total

Discount %

SubTotal YTL

VAT %

TOTAL YTL

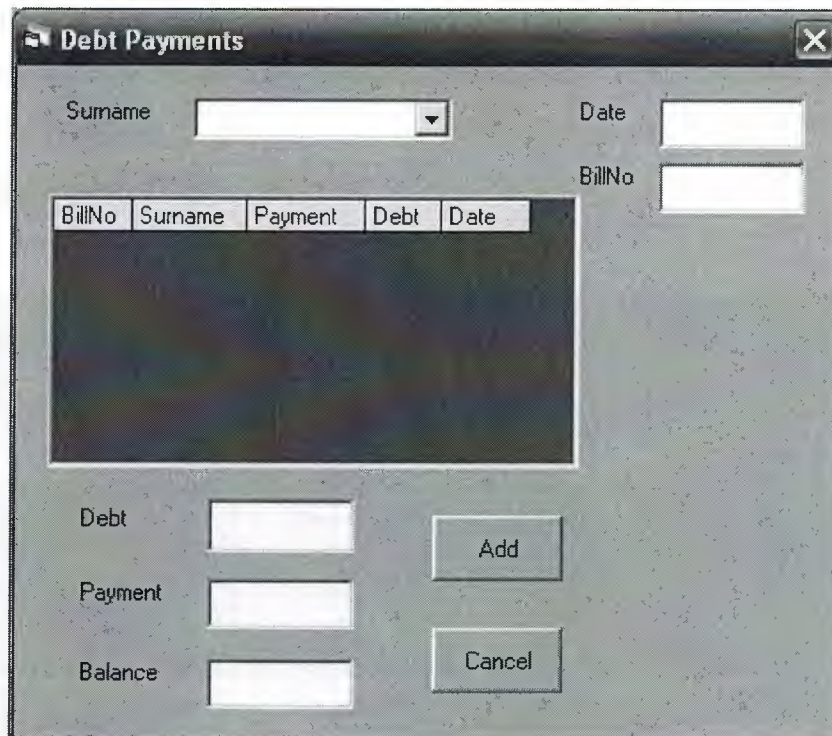
Payment

Debt

Figure 3.9 Payment Page

The user checks the customer's bought items with Bill No. Then calculate the item's prices and if there is enter the discount percent. The customer may be pay the total but it does not guaranty. So this Page adds the Customer's payments and debts to the database.

3.8 Debt Payment Page



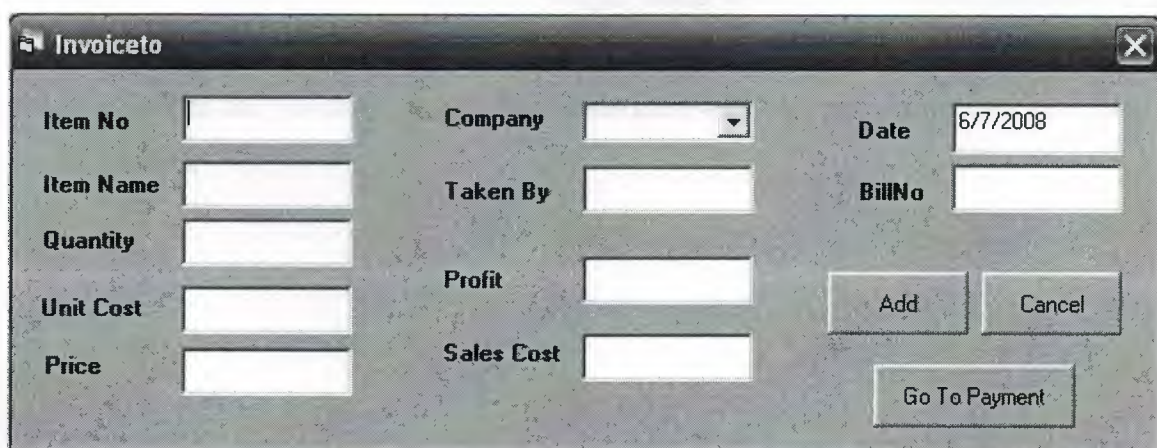
The screenshot shows a window titled "Debt Payments" with a close button (X) in the top right corner. The window contains the following elements:

- A "Surname" label followed by a text input field.
- A "Date" label followed by a text input field.
- A "BillNo" label followed by a text input field.
- A table with the following headers: "BillNo", "Surname", "Payment", "Debt", and "Date". The table body is currently empty.
- A "Debt" label followed by a text input field.
- A "Payment" label followed by a text input field.
- A "Balance" label followed by a text input field.
- An "Add" button located to the right of the "Debt" and "Payment" input fields.
- A "Cancel" button located to the right of the "Balance" input field.

Figure 3.10 Debt Payment Page.

This page first controls the customer's surname and his payment and debt informations. This is used for payment of the debt by debtors.

3.9 Invoiceto Page



The screenshot shows a window titled "Invoiceto" with a close button (X) in the top right corner. The window contains the following elements:

- A vertical list of labels on the left: "Item No", "Item Name", "Quantity", "Unit Cost", and "Price", each followed by a text input field.
- A "Company" label followed by a dropdown menu.
- A "Taken By" label followed by a text input field.
- A "Profit" label followed by a text input field.
- A "Sales Cost" label followed by a text input field.
- A "Date" label followed by a text input field containing the value "6/7/2008".
- A "BillNo" label followed by a text input field.
- An "Add" button and a "Cancel" button located to the right of the "Profit" and "Sales Cost" input fields.
- A "Go To Payment" button located at the bottom right of the window.

Figure 3.11 Invoiceto Page.

With this page the user add the items that are bought by the own company. Also the profit and sales prices of the added item is calculated here. Then save to database.

3.10 Paymentto Page

Paymentto

Bill No Search

BillNo	Company	ItemNo	Item Name	Quantity	Price	Date
--------	---------	--------	-----------	----------	-------	------

Total

Discount %

SubTotal YTL

VAT %

TOTAL YTL

Payment

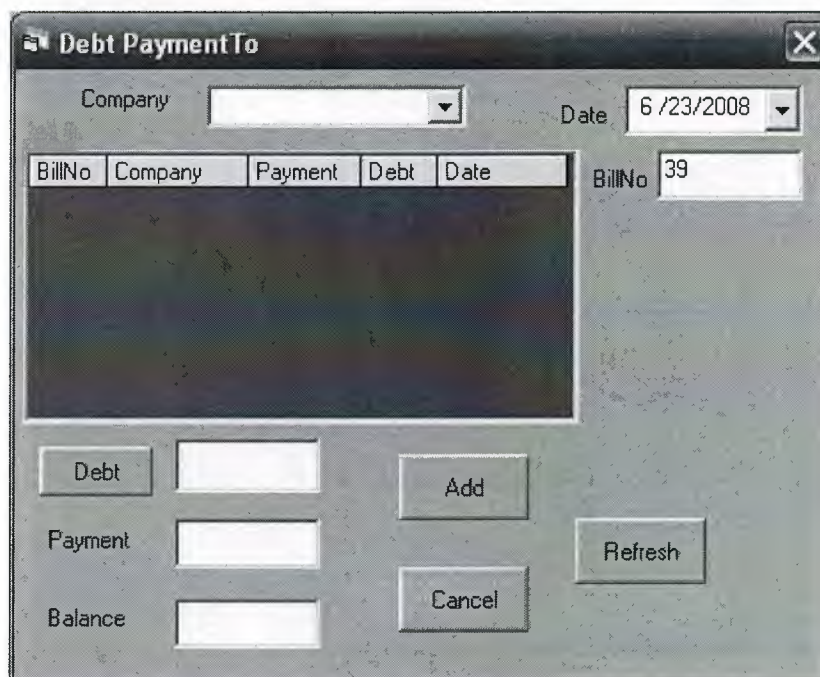
Debt

Compute Save Cancel

Figure 3.12 Paymentto Page.

The user checks the company's bought items with Bill No. Then calculate the item's prices and if there is enter the discount percent. The company may be pay the total but it does not guaranty. So this Page adds the own Company's payments and debts to the database.

3.11 Debt Paymentto Page

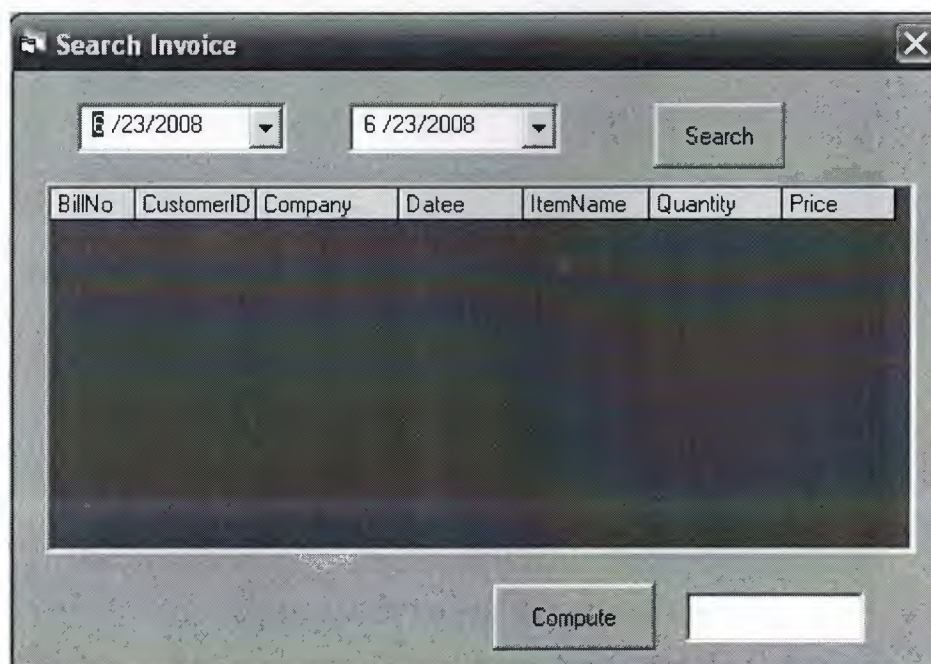


The screenshot shows a window titled "Debt Paymentto" with a close button (X) in the top right corner. At the top, there is a "Company" dropdown menu and a "Date" dropdown menu showing "6 /23/2008". Below these is a table with columns: BillNo, Company, Payment, Debt, and Date. The table is currently empty. To the right of the table is a "BillNo" input field containing the value "39". Below the table, there are three input fields labeled "Debt", "Payment", and "Balance". To the right of these fields are three buttons: "Add", "Cancel", and "Refresh".

Figure 3.13 Debt Paymentto Page

This page controls by the company name that our company bought items from them and make the debt payment of our company.

3.12 Serach Invoice Page



The screenshot shows a window titled "Search Invoice" with a close button (X) in the top right corner. At the top, there are two date dropdown menus, both showing "6 /23/2008", and a "Search" button. Below these is a table with columns: BillNo, CustomerID, Company, Datee, ItemName, Quantity, and Price. The table is currently empty. At the bottom, there is a "Compute" button and an empty input field.

Figure 3.14 Search Invoice Page

This page calculates the total price in the range of two dates.

3.13 Search Payment Page

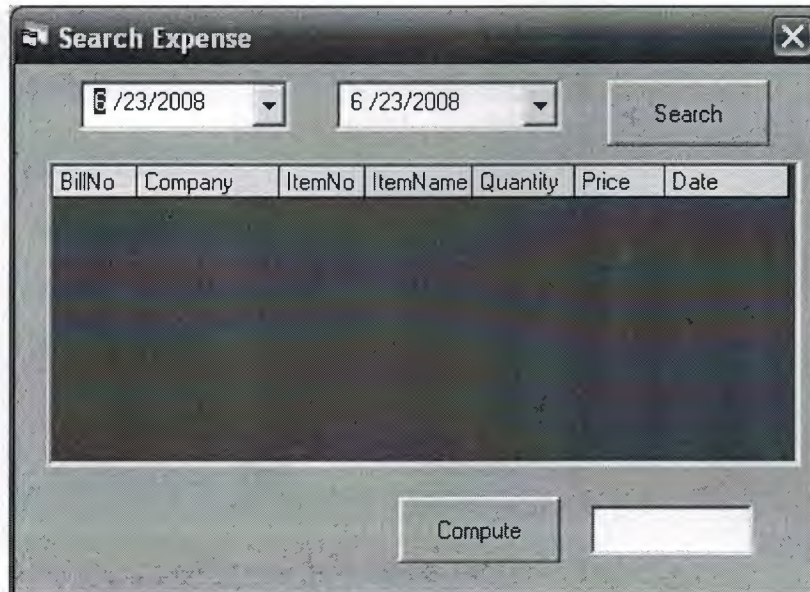


The screenshot shows a window titled "Search Payment" with a close button (X) in the top right corner. At the top, there are two date selection fields, both displaying "6 /23/2008", and a "Search" button to their right. Below these is a table with the following headers: "BillNo", "Surname", "Payment", "Debt", "Date", and "Statu". The table body is currently empty. At the bottom of the window, there is a "Compute" button, followed by two empty text input fields. Above the second input field, the labels "Payment" and "Debt" are positioned.

Figure 3.15 Search Payment Page

This page calculates the total payment and debts of the customers in the range of two date.

3.14 Search Expenses Page



The screenshot shows a window titled "Search Expense" with a close button (X) in the top right corner. At the top, there are two date selection fields, both displaying "6 /23/2008", and a "Search" button to their right. Below these is a table with the following headers: "BillNo", "Company", "ItemNo", "ItemName", "Quantity", "Price", and "Date". The table body is currently empty. At the bottom of the window, there is a "Compute" button followed by an empty text input field.

Figure 3.16 Search Expenses Page

This Page calculates the total price of our payment to the companies.

3.15 Search Paymentto Page

The 'Search Paymentto' dialog box features two date pickers at the top, both set to '6 /24/2008', and a 'Search' button. Below these is a table with the following headers: BillNo, Company, Payment, Debt, and Date. The table body is currently empty. At the bottom, there is a 'Compute' button, and two empty input fields labeled 'Payment' and 'Debt'.

Figure 3.17 Search Paymentto Page

This Page calculates the total payment and debts of our company to the other companies.

3.16 Add Customer Page

The 'Add Customer' dialog box contains several input fields on the left: ID, Name, Surname, Telephone, Identity_No, and Address. On the right, there is a table with headers: CID, Name, Surname, Tel, KimlikNo, and Address. Below the table is a search section with a 'By Name' label, an input field, and a 'Search' button. At the bottom right, there are four buttons: 'Add', 'Update', 'Delete', and 'Cancel'.

Figure 3.18 Add Customer Page

With this page user can add the customer information to the database, update and delete the records from the database also.

3.17 About Page

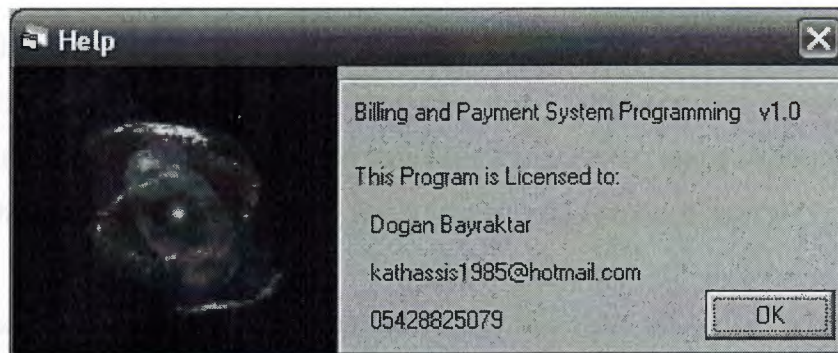


Figure 3.19 About Page.

The user can learn the informations about the designer of the program.

CONCLUSION

While doing this project many things are learned, because until now the students were just learning programs literary or doing small programs, but by the way of this program studied a lot in internet, read many books, and tried many examples in order to improve the knowledge about Visual Basic to make this program useful.

There are many advantages and disadvantages of using Visual Basic in programming, the one of advantages of using VB is easy in use, and one of the disadvantages of using Visual Basic is more slow than some of the other programming languages and it does not have so much properties.

This program will be updated in the future, some other properties will be added to it if conditions changes, and it can be used through the internet by adding extra codes, while doing the project wonderful imaginations occurs but did not applied them because there couldn't be able to finish this project on time, but the imaginations will be added the project in the future.

This program is a billing and payment system programming, this program is being done because of there is a thought of selling it to the companies. And this program is not designed for a private company so this program is useful in general purpose.

REFERENCES

1. Microsoft SQL Server 2000 Programming by Example E-Book
Fernando Guerrero,2001.
2. Yakup Varol and Hasan Koca, VB KodBank 1.7, E-Book, World Wide Web
www.vbkodbank.com ,2001.
3. A guide research for writing program. Retrieved December 15, 2006 from the
World Wide Web "<http://www.programlama.com>
4. A guide research for writing program. Retrieved January 3, 2007 from the
World Wide Web "[http:// www.devarticles.com](http://www.devarticles.com)
5. A guide research for describing the database.
World Wide Web <http://databases.about.com>
6. A guide research for describing the database.
. World Wide Web <http://www.mssqlcity.com>

APPENDICES

MDIForm1.frm

Private Sub Label1_Click()

add_person.Show

End Sub

Private Sub Label10_Click()

Unload Me

Unload add_person

Unload add_firm

Unload invoice

Unload debtpayment

Unload debtpaymentto

Unload add_item

End Sub

Private Sub Label2_Click()

add_firm.Show

End Sub

Private Sub Label4_Click()

invoice.Show

End Sub

Private Sub Label5_Click()

debtpayment.Show

End Sub

Private Sub Label7_Click()

add_item.Show

End Sub

Private Sub Label8_Click()

```
debtpaymentto.Show  
End Sub
```

```
Private Sub Label9_Click()
```

```
    help.Show
```

```
End Sub
```

add_person.frm

```
Dim database As New ADODB.Connection  
Dim table As New ADODB.Recordset  
Dim i As Integer  
Dim bilgi(0 To 9) As String  
Dim table1 As New ADODB.Recordset
```

```
Private Sub add_cmd_Click()
```

```
    If name_txt.Text <> "" And surname_txt.Text <> "" And tel1_txt.Text <> "" And  
    kimlikno_txt.Text <> "" And address_txt.Text <> "" Then
```

```
        Set table = New ADODB.Recordset
```

```
        table.Open          "INSERT                      INTO          Person  
(UserID,Password,Name,Surname,Address,Tel1,Tel2,KimlikNo,Statu) VALUES(" &  
        userid_txt.Text & "," & pass_txt.Text & "," & name_txt.Text & "," &  
        surname_txt.Text & "," & address_txt.Text & "," & tel1_txt.Text & "," &  
        tel2_txt.Text & "," & kimlikno_txt.Text & "," & statu_txt.Text & ")", database
```

```
        id_txt.Text = ""
```

```
        userid_txt.Text = ""
```

```
        pass_txt.Text = ""
```

```
        name_txt.Text = ""
```

```

surname_txt.Text = ""
tel1_txt.Text = ""
tel2_txt.Text = ""
kimlikno_txt.Text = ""
address_txt.Text = ""
statu_txt.Text = ""
search_txt.Text = ""
Else
    MsgBox "you cant leave the name and surname empty", 48, "Caution Message"
    If name_txt.Text = "" Then
        name_txt.SetFocus
    End If
    If surname_txt.Text = "" Then
        surname_txt.SetFocus
    End If
    If tel1_txt.Text = "" Then
        tel1_txt.SetFocus
    End If
    If kimlikno_txt.Text = "" Then
        kimlikno_txt.SetFocus
    End If
    If address_txt.Text = "" Then
        address_txt.SetFocus
    End If
End If

```

End Sub

```

Private Sub del_cmd_Click()

```



```

    answer = MsgBox("Do you want to DELETE?", vbQuestion + vbYesNo +
vbDefaultButton1)
    If answer = vbYes Then

        Set table = New ADODB.Recordset
        table.Open " DELETE FROM Person WHERE ID=" & id_txt & "", database
        End If
        Call ready
        Set table1 = New ADODB.Recordset
        table1.Open " SELECT * FROM Person WHERE Statu LIKE '%" & search_txt &
"%", database

        While Not table1.EOF
            G1.AddItem table1("ID") & Chr(9) & table1("UserID") & Chr(9) &
table1("Password") & Chr(9) & table1("Name") & Chr(9) & table1("Surname") &
Chr(9) & table1("Address") & Chr(9) & table1("Tel1") & Chr(9) & table1("Tel2") &
Chr(9) & table1("KimlikNo") & Chr(9) & table1("Statu")
            table1.MoveNext
        Wend

        id_txt.Text = ""
        userid_txt.Text = ""
        pass_txt.Text = ""
        name_txt.Text = ""
        surname_txt.Text = ""
        tel1_txt.Text = ""
        tel2_txt.Text = ""
        kimlikno_txt.Text = ""
        address_txt.Text = ""
        statu_txt.Text = ""
        search_txt.Text = ""
    End Sub

```

Private Sub Form_Load()

Call ready

Set database = New ADODB.Connection

Set table = New ADODB.Recordset

database.Open "Provider=SQLOLEDB;data source=(local);Initial
Catalog=proje;Integrated Security=SSPI"

Option1.Value = True

End Sub

Private Sub ready()

G1.Clear

G1.FormatString = "ID |User_ID |Password |Name |Surname |Address
|Tel1 |Tel2 |KimlikNo |Statu "

G1.Cols = 10

G1.Rows = 1

End Sub

Private Sub G1_Click()

For i = 0 To 9

bilgi(i) = G1.TextMatrix(G1.MouseRow, i)

Next

id_txt.Text = bilgi(0)

userid_txt.Text = bilgi(1)

```
pass_txt.Text = bilgi(2)
name_txt.Text = bilgi(3)
surname_txt.Text = bilgi(4)
tel1_txt.Text = bilgi(6)
tel2_txt.Text = bilgi(7)
kimlikno_txt.Text = bilgi(8)
address_txt.Text = bilgi(5)
statu_txt.Text = bilgi(9)
```

End Sub

Private Sub new_cmd_Click()

```
Call ready
id_txt.Text = ""
userid_txt.Text = ""
pass_txt.Text = ""
name_txt.Text = ""
surname_txt.Text = ""
tel1_txt.Text = ""
tel2_txt.Text = ""
kimlikno_txt.Text = ""
address_txt.Text = ""
statu_txt.Text = ""
search_txt.Text = ""
```

End Sub

Private Sub search_cmd_Click()

```
Set table = New ADODB.Recordset
If Option1.Value = True Then
```



```

    table.Open " SELECT * FROM Person WHERE Statu LIKE '%" & search_txt &
    "%'", database
    Else

    If Option2.Value = True Then
        table.Open "SELECT * FROM Person WHERE Name LIKE '%" & search_txt &
        "%'", database
    End If
    End If

    Call ready

    While Not table.EOF
        G1.AddItem table("ID") & Chr(9) & table("UserID") & Chr(9) &
        table("Password") & Chr(9) & table("Name") & Chr(9) & table("Surname") & Chr(9)
        & table("Address") & Chr(9) & table("Tel1") & Chr(9) & table("Tel2") & Chr(9) &
        table("KimlikNo") & Chr(9) & table("Statu")
        table.MoveNext
    Wend

End Sub
-----

Private Sub update_cmd_Click()

    Set table = New ADODB.Recordset

    table.Open " UPDATE Person SET UserID='" & userid_txt & "', Password='" &
    pass_txt & "', Name='" & name_txt & "', Surname='" & surname_txt & "', Address='" &
    address_txt & "', Tel1='" & tel1_txt & "', Tel2='" & tel2_txt & "', KimlikNo='" &
    kimlikno_txt & "', Statu='" & statu_txt & "' WHERE ID='" & id_txt & "'", database

```

```

id_txt.Text = ""
userid_txt.Text = ""
pass_txt.Text = ""
name_txt.Text = ""
surname_txt.Text = ""
tel1_txt.Text = ""
tel2_txt.Text = ""
kimlikno_txt.Text = ""
address_txt.Text = ""
statu_txt.Text = ""
search_txt.Text = ""

```

End Sub

add_firm.frm

```

Dim bilgi(0 To 7) As String
Dim i As Integer
Dim database As New ADODB.Connection
Dim table As New ADODB.Recordset
Dim table1 As New ADODB.Recordset

```

Private Sub add_cmd_Click()

```

    If firmname_txt.Text <> "" Then
        Set table = New ADODB.Recordset
        table.Open " INSERT INTO Firm (FirmName,Tel,Address,Fax,Mail,Web) VALUES
        (" & firmname_txt.Text & "," & tel_txt.Text & "," & address_txt.Text & "," &
        fax_txt.Text & "," & email_txt.Text & "," & web_txt.Text & ")", database
        firmid_txt.Text = ""
        firmname_txt.Text = ""
        address_txt.Text = ""
        tel_txt.Text = ""
    
```

web_txt.Text = ""

fax_txt.Text = ""

email_txt.Text = ""

Else

MsgBox "You Can't Leave Company Name Empty", 46, "Caution Message"

If firmname_txt.Text = "" Then

 firmname_txt.SetFocus

End If

End If

End Sub

Private Sub del_cmd_Click()

 answer = MsgBox("Do you want to DELETE?", vbQuestion + vbYesNo +
vbDefaultButton2)

 If answer = vbYes Then

 Set table1 = New ADODB.Recordset

 table1.Open " DELETE FROM Firm WHERE FirmID=" & firmid_txt & "",
database

 End If

 Set table = New ADODB.Recordset

 table.Open " SELECT * FROM Firm WHERE FirmName LIKE '%" & search_txt &
"%"", database

 Call ready

While Not table.EOF

```
G1.AddItem table("FirmID") & Chr(9) & table("FirmName") & Chr(9) &  
table("Tel") & Chr(9) & table("Fax") & Chr(9) & table("Mail") & Chr(9) &  
table("Web") & Chr(9) & table("Address")
```

```
table.MoveNext
```

Wend

```
firmed_txt.Text = ""
```

```
firname_txt.Text = ""
```

```
address_txt.Text = ""
```

```
tel_txt.Text = ""
```

```
web_txt.Text = ""
```

```
fax_txt.Text = ""
```

```
email_txt.Text = ""
```

End Sub

Private Sub Form_Load()

```
Set database = New ADODB.Connection
```

```
Set table = New ADODB.Recordset
```

```
database.Open "Provider=SQLOLEDB;data source=(local);Initial  
Catalog=proje;Integrated Security=SSPI"
```

```
Call ready
```

End Sub

Private Sub ready()

```
G1.Clear
```

```
G1.FormatString = "FirmID |Firm Name |Telephone |Fax |E-Mail |Web  
|Address            "  
G1.Cols = 7  
G1.Rows = 1
```

End Sub

Private Sub G1_Click()

```
For i = 0 To 6  
    bilgi(i) = G1.TextMatrix(G1.MouseRow, i)  
Next  
firmid_txt.Text = bilgi(0)  
firmname_txt.Text = bilgi(1)  
tel_txt.Text = bilgi(2)  
fax_txt.Text = bilgi(3)  
email_txt.Text = bilgi(4)  
web_txt.Text = bilgi(5)  
address_txt.Text = bilgi(6)
```

End Sub

Private Sub search_cmd_Click()

```
Set table = New ADODB.Recordset  
table.Open " SELECT * FROM Firm WHERE FirmName LIKE '%" & search_txt &  
"%"", database
```

Call ready

While Not table.EOF

```
G1.AddItem table("FirmID") & Chr(9) & table("FirmName") & Chr(9) &  
table("Tel") & Chr(9) & table("Fax") & Chr(9) & table("Mail") & Chr(9) &  
table("Web") & Chr(9) & table("Address")
```

```
table.MoveNext
```

```
Wend
```

```
End Sub
```

```
Private Sub update_cmd_Click()
```

```
Set table = New ADODB.Recordset
```

```
table.Open "UPDATE Firm SET FirmName='" & firmname_txt.Text & "', Tel='" &  
tel_txt.Text & "', Address='" & address_txt.Text & "', Fax='" & fax_txt.Text & "',  
Mail='" & email_txt.Text & "', Web='" & web_txt.Text & "' WHERE FirmID='" &  
firmid_txt.Text & "'", database
```

```
firmid_txt.Text = ""
```

```
firmname_txt.Text = ""
```

```
address_txt.Text = ""
```

```
tel_txt.Text = ""
```

```
web_txt.Text = ""
```

```
fax_txt.Text = ""
```

```
email_txt.Text = ""
```

```
End Sub
```

add_item.frm

```
Dim table As New ADODB.Recordset
```

```
Dim database As New ADODB.Connection
```

```
Dim i As Integer
```

```
Dim x(0 To 4) As String
```

Private Sub add_cmd_Click()

```
Set table = New ADODB.Recordset  
table.Open "INSERT INTO Item  
(ItemNo,Item,Quantity,Price,Date,Unitcost,Profit,Salescost,Company,BillNo)  
VALUES(" & itemno_txt.Text & "," & itemname_txt.Text & "," & quantity_txt.Text  
& "," & price_txt.Text & "," & date_txt.Text & "," & unitcost_txt.Text & "," &  
profit_txt.Text & "," & salescost_txt.Text & "," & Combol.Text & "," &  
billno_txt.Text & ")", database
```

```
itemno_txt.Text = ""  
itemname_txt.Text = ""  
quantity_txt.Text = ""  
price_txt.Text = ""  
salescost_txt.Text = ""  
profit_txt.Text = ""  
unitcost_txt.Text = ""
```

End Sub

Private Sub Command1_Click()

Unload Me

End Sub

Private Sub Command2_Click()

receiptto.Show

End Sub

Private Sub Form_Load()

Set database = New ADODB.Connection

Set table = New ADODB.Recordset

database.Open "Provider=SQLOLEDB;data source=(local);Initial
Catalog=proje;Integrated Security=SSPI"

date_txt.Text = Date

table.Open "Select * from Firm WHERE FirmName LIKE '%" & Combo1.Text &
"%"", database

While Not table.EOF

Combo1.AddItem table.Fields("FirmName")

table.MoveNext

Wend

date_txt.Text = Date

End Sub

Private Sub price_txt_GotFocus()

price_txt = Val(quantity_txt) * Val(unitcost_txt)

End Sub

Private Sub salescost_txt_GotFocus()

salescost_txt = Val(unitcost_txt) + Val(profit_txt)

End Sub

debtpayment.frm

Dim database As New ADODB.Connection

Dim table As New ADODB.Recordset

Dim table1 As New ADODB.Recordset

Private Sub Combo1_Click()

Set table = New ADODB.Recordset

table.Open "SELECT * FROM Receipt WHERE Surname LIKE '%" & Combo1.Text
& "%'", database

While Not table.EOF

G1.AddItem table("BillNo") & Chr(9) & table("Surname") & Chr(9) &
table("Payment") & Chr(9) & table("Debt") & Chr(9) & table("Datee")

table.MoveNext

Wend

End Sub

Private Sub Form_Load()

Set database = New ADODB.Connection

Set table = New ADODB.Recordset

database.Open "Provider=SQLOLEDB;data source=(local);Initial
Catalog=proje;Integrated Security=SSPI"

Call ready

table.Open "Select * from Receipt WHERE Surname LIKE '%" & Combo1.Text &
"%"", database

While Not table.EOF

Combo1.AddItem table.Fields("Surname")

table.MoveNext

Wend

End Sub

Private Sub ready()

G1.Clear

G1.FormatString = "BillNo |Surname |Payment |Debt |Date "

G1.Cols = 5

G1.Rows = 1

End Sub

help.frm

Private Sub Command1_Click()

Unload Me

End Sub

invoice.frm

```

Dim table As New ADODB.Recordset
Dim database As New ADODB.Connection
Dim table1 As New ADODB.Recordset
Dim table2 As New ADODB.Recordset

```

```

Private Sub add_cmd_Click()

```

```

    Set table = New ADODB.Recordset
    table.Open "INSERT INTO bill
(BillNo,CustomerID,Company,Datee,Hourr,ItemNo,ItemName,Quantity,Price)
VALUES('" & billno_txt.Text & "','" & name_txt.Text & "','" & Combo1.Text & "','" &
date_txt.Text & "','" & saat_txt.Text & "','" & itemno_txt.Text & "','" &
itemname_txt.Text & "','" & quantity_txt.Text & "','" & price_txt.Text & "')" , database

```

```

    itemno_txt.Text = ""
    itemname_txt.Text = ""
    quantity_txt.Text = ""
    unitcost_txt.Text = ""
    price_txt.Text = ""

```

```

End Sub

```

```

Private Sub Combo2_Click()

```

```

    Set table = New ADODB.Recordset
    table.Open " SELECT * FROM Person WHERE Surname='" & Combo2.Text & "'",
database
    If table.EOF Then
        Combo2.SetFocus
    Else

        If table.Fields("Surname") = Combo2.Text Then
            name__txt.Text = table.Fields("ID")

```

End If

End If

End Sub

Private Sub Command1_Click()

receipt.Show

End Sub

Private Sub Command2_Click()

Unload Me

End Sub

Private Sub Form_Load()

Set database = New ADODB.Connection

Set table = New ADODB.Recordset

database.Open "Provider=SQLOLEDB;data source=(local);Initial
Catalog=proje;Integrated Security=SSPI"

table.Open "Select * from Firm WHERE FirmName LIKE '%" & Combo1.Text &
"%"", database

While Not table.EOF

Combo1.AddItem table.Fields("FirmName")

table.MoveNext

Wend

Set table2 = New ADODB.Recordset


```
table2.Open "SELECT * FROM Person WHERE Statu ='4'", database
```

```
While Not table2.EOF
```

```
    Combo2.AddItem table2.Fields("Surname")
```

```
    table2.MoveNext
```

```
Wend
```

```
Set table1 = New ADODB.Recordset
```

```
table1.Open "SELECT * FROM BillNo", database
```

```
Dim i As Integer
```

```
While Not table1.EOF
```

```
    i = 1 + table1.Fields("BillNo")
```

```
    table1.MoveNext
```

```
Wend
```

```
billno_txt.Text = i
```

```
End Sub
```

```
-----  
Private Sub itemname_txt_GotFocus()
```

```
    Set table = New ADODB.Recordset
```

```
    table.Open " SELECT * FROM Item WHERE ItemNo='" & itemno_txt.Text & "'",  
database
```

```
    If table.EOF Then
```

```
        MsgBox " No item"
```

```
    Else
```

```
        If table.Fields("ItemNo") = itemno_txt.Text Then
```

```
            itemname_txt.Text = table.Fields("Item")
```

```
        End If
```

End If

End Sub

Private Sub price_txt_GotFocus()

price_txt = Val(quantity_txt) * Val(unitcost_txt)

If quantity_txt = "" And unitcost_txt = "" Then

price_txt = ""

End If

End Sub

Private Sub Timer1_Timer()

saat_txt.Text = Time

date_txt.Text = Date

End Sub

login.frm

Dim database As New ADODB.Connection

Dim table As New ADODB.Recordset

Private Sub Command1_Click()

Set table = New ADODB.Recordset

table.Open " Select * FROM Person where UserID='" & txt_userid.Text & "' and
Password='" & txt_pass.Text & "'", database

If txt_userid.Text = "" And txt_pass.Text = "" Then

Exit Sub

End If

```

If table.EOF Then
    MsgBox "Wrong User Name or Password!!"
    txt_userid.SetFocus
Else
    If txt_userid.Text = table.Fields("UserID") And txt_pass.Text =
table.Fields("Password") Then
        MDIForm1.Show
        Unload Me
    End If
End If

```

```

If table.EOF Then
    txt_userid.SetFocus
Else
    If table.Fields("Statu") = 2 Then
        transact.Command3.Visible = False
    Else
        If table.Fields("Statu") = 3 Then
            transact.Command1.Visible = False
            transact.Command2.Visible = False
            transact.Command3.Visible = False
        End If
    End If
End If

```

```

End Sub

```

```

Private Sub Command2_Click()

```

```

    Unload Me

```

```

End Sub

```

Private Sub Form_Load()

Set database = New ADODB.Connection

Set table = New ADODB.Recordset

database.Open "Provider=SQLOLEDB;data source=(local);Initial
Catalog=proje;integrated security=SSPI"

End Sub

Private Sub txt_pass_KeyPress(KeyAscii As Integer)

If KeyAscii = 13 Then

Call Command1_Click

End If

End Sub

receipt.frm

Dim table As New ADODB.Recordset

Dim database As New ADODB.Connection

Dim table1 As New ADODB.Recordset

Private Sub cancel_cmd_Click()

Unload Me

End Sub

Private Sub save_cmd_Click()

Set table = New ADODB.Recordset

table.Open " INSERT INTO Receipt(BillNo,Surname,Payment,Debt,Datee)
VALUES(" & Text1.Text & "," & invoice.Combo2.Text & "," & Text3.Text & " ','
& Text4.Text & "," & invoice.date_txt.Text & ")", database

End Sub

Private Sub search_cmd_Click()

Set table = New ADODB.Recordset

table.Open " SELECT * FROM bill WHERE BillNo LIKE '%" & Text1 & "%",
database

Call ready

While Not table.EOF

G1.AddItem table("BillNo") & Chr(9) & table("CustomerID") & Chr(9) &
table("Company") & Chr(9) & table("Datee") & Chr(9) & table("ItemName") & Chr(9)
& table("Quantity") & Chr(9) & table("Price")

table.MoveNext

Wend

End Sub

Private Sub Command2_Click()

Dim X As String

Dim toplam As Double

Dim i As Integer

For i = 0 To G1.Rows - 1

X = G1.TextMatrix(i, 6)

toplam = toplam + Val(X)

Next

Text2.Text = toplam

End Sub

Private Sub Command3_Click()

total = Val(Text2)

If Text23 = "" Then

Text24 = ""

End If

Text24 = (Val(total) * Val(Text23)) / 100

Text25 = Val(total) - Val(Text24)

If Text26 = "" Then

Text5 = ""

End If

Text5 = (Val(Text25) * Val(Text26)) / 100

Text6 = Val(Text5) + Val(Text25)

End Sub

Private Sub Form_Load()

Set database = New ADODB.Connection

Set table = New ADODB.Recordset

database.Open "Provider=SQLOLEDB;data source=(local);Initial
Catalog=proje;Integrated Security=SSPI"

Call ready

Text1.Text = invoice.billno_txt.Text

End Sub

Private Sub ready()

G1.Clear

G1.FormatString = "BillNo |CustomerID |Company |Datee |Item Name
|Quantity |Price |"

G1.Cols = 7

G1.Rows = 1

End Sub

Private Sub Text4_GotFocus()

Text4 = Val(Text6) - Val(Text3)

End Sub

receiptto.frm

Dim database As New ADODB.Connection

Dim table As New ADODB.Recordset

Private Sub Command1_Click()

Dim X As String

Dim toplam As Double

Dim i As Integer

For i = 0 To G1.Rows - 1

X = G1.TextMatrix(i, 5)

toplam = toplam + Val(X)

Next

Text2.Text = toplam

End Sub

Private Sub Command2_Click()

```

total = Val(Text2)
If Text23 = "" Then
    Text24 = ""
End If
Text24 = (Val(total) * Val(Text23)) / 100
Text25 = Val(total) - Val(Text24)
If Text26 = "" Then
    Text5 = ""
End If
Text5 = (Val(Text25) * Val(Text26)) / 100
Text6 = Val(Text5) + Val(Text25)

End Sub
-----

Private Sub Form_Load()

    Call ready

    Set database = New ADODB.Connection
    Set table = New ADODB.Recordset

    database.Open"Provider=SQLOLEDB;datasource=(local);Initial
Catalog=proje;Integrated Security=SSPI"

    Call ready

    Text1.Text = add_item.billno_txt.Text
    'Text7.Text = Date

End Sub
-----

Private Sub ready()

    G1.Clear

```

```

G1.FormatString = "BillNo |Company |ItemNo |Item Name |Quantity
|Price |Date "
G1.Cols = 7
G1.Rows = 1

```

```

End Sub

```

```

Private Sub save_cmd_Click()

```

```

    Set table = New ADODB.Recordset
    table.Open "INSERT INTO Receiptto (BillNo,Company,Payment,Debt,Datee)
VALUES(" & Text1.Text & "," & add_item.Combo1.Text & "," & Text3.Text & ","
& Text4.Text & "," & add_item.date_txt.Text & ")", database

```

```

End Sub

```

```

Private Sub search_cmd_Click()

```

```

    Set table = New ADODB.Recordset
    table.Open " SELECT * FROM Item WHERE BillNo LIKE '%" & Text1 & "%'",
database

```

```

    Call ready

```

```

    While Not table.EOF

```

```

        G1.AddItem table("BillNo") & Chr(9) & table("Company") & Chr(9) &
table("ItemNo") & Chr(9) & table("Item") & Chr(9) & table("Quantity") & Chr(9) &
table("Price") & Chr(9) & table("Date")

```

```

        table.MoveNext

```

```

    Wend

```

```

End Sub

```

```

Private Sub Text4_GotFocus()

```


Text4 = Val(Text6) - Val(Text3)

End Sub

debtpaymentto.frm

Dim database As New ADODB.Connection

Dim table As New ADODB.Recordset

Dim table1 As New ADODB.Recordset

Private Sub Combo1_Click()

Set table = New ADODB.Recordset

table.Open "SELECT * FROM Receiptto WHERE Surname LIKE '%" &
Combo1.Text & "%'", database

While Not table.EOF

G1.AddItem table("BillNo") & Chr(9) & table("Surname") & Chr(9) &
table("Payment") & Chr(9) & table("Debt") & Chr(9) & table("Datee")

table.MoveNext

Wend

End Sub

Private Sub Form_Load()

Set database = New ADODB.Connection

Set table = New ADODB.Recordset

database.Open "Provider=SQLOLEDB;data source=(local);Initial
Catalog=proje;Integrated Security=SSPI"

Call ready

```
table.Open "Select * from Receipt WHERE Surname LIKE '%" & Combo1.Text &
"%"", database
```

```
While Not table.EOF
```

```
    Combo1.AddItem table.Fields("Surname")
```

```
    table.MoveNext
```

```
Wend
```

```
End Sub
```

```
Private Sub ready()
```

```
    G1.Clear
```

```
    G1.FormatString = "BillNo |Surname |Payment |Debt |Date "
```

```
    G1.Cols = 5
```

```
    G1.Rows = 1
```

```
End Sub
```

sexpenses.frm

```
Dim Database As New ADODB.Connection
```

```
Dim table As New ADODB.Recordset
```

```
Private Sub Command1_Click()
```

```
tarih1 = DTPicker1
```

```
tarih2 = DTPicker2
```

```
'Dim gun, ay, yil As Integer
```

```
'gun = Mid(tarih1, 1, 2)
```

```
'ay = Mid(tarih1, 4, 2)
```

```
'yil = Mid(tarih1, 7, 4)
```

```
'tarih1 = gun & "/" & ay & "/" & yil
```

```
'gun = Mid(tarih2, 1, 2)
```

```
'ay = Mid(tarih2, 4, 2)
```

```
'yil = Mid(tarih2, 7, 4)
```

```
'tarih2 = gun & "/" & ay & "/" & yil
```

```
Set table = New ADODB.Recordset
```

```
table.Open "select * from Item where Datee between '" & tarih1 & "' and '" & tarih2 &
"' ", Database
```

```
While Not table.EOF
```

```
    G1.AddItem table.Fields("BillNo") & Chr(9) & table("Company") & Chr(9) &
table("ItemNo") & Chr(9) & table("Item") & Chr(9) & table("Quantity") & Chr(9) &
table("Price") & Chr(9) & table("Datee")
```

```
    table.MoveNext
```

```
Wend
```

```
End Sub
```

```
Private Sub Command2_Click()
```

```
    Dim X As String
```

```
    Dim toplam As Double
```

```
    Dim i As Integer
```

```
    For i = 0 To G1.Rows - 1
```

```
        X = G1.TextMatrix(i, 5)
```

```
        toplam = toplam + Val(X)
```

```
    Next
```

```
    Text1.Text = toplam
```

```
End Sub
```

```
Private Sub Form_Load()
```

```
    Set Database = New ADODB.Connection
```

```
    Set table = New ADODB.Recordset
```

```
    Database.Open "Provider=SQLOLEDB;data source=(local);Initial
Catalog=proje;Integrated Security=SSPI"
```


DTPicker1.Value = Date

DTPicker2.Value = Date

Call ready

End Sub

Private Sub ready()

G1.Clear

G1.FormatString = "BillNo |Company |ItemNo |ItemName|Quantity |Price
|Date |"

G1.Rows = 1

G1.Cols = 7

End Sub

sinvoice.frm

Dim table As New ADODB.Recordset

Dim Database As New ADODB.Connection

Private Sub Command1_Click()

Dim X As String

Dim toplam As Double

Dim i As Integer

For i = 0 To G1.Rows - 1

X = G1.TextMatrix(i, 6)

toplam = toplam + Val(X)

Next

Text1.Text = toplam

End Sub

Private Sub Command2_Click()

tarih1 = DTPicker1

tarih2 = DTPicker2

'Dim gun, ay, yil As Integer

'gun = Mid(tarih1, 1, 2)

'ay = Mid(tarih1, 4, 2)

'yil = Mid(tarih1, 7, 4)

'tarih1 = gun & "/" & ay & "/" & yil

'gun = Mid(tarih2, 1, 2)

'ay = Mid(tarih2, 4, 2)

'yil = Mid(tarih2, 7, 4)

'tarih2 = gun & "/" & ay & "/" & yil

Set table = New ADODB.Recordset

table.Open "select * from bill where Datee between '" & tarih1 & "' and '" & tarih2 & "'
", Database

While Not table.EOF

G1.AddItem table.Fields("BillNo") & Chr(9) & table("CustomerID") & Chr(9) &
table("Company") & Chr(9) & table("Datee") & Chr(9) & table("ItemName") & Chr(9)
& table("Quantity") & Chr(9) & table("Price")

table.MoveNext

Wend

End Sub

Private Sub Form_Load()

Set Database = New ADODB.Connection

Set table = New ADODB.Recordset

Database.Open "Provider=SQLOLEDB;data source=(local);Initial
Catalog=proje;Integrated Security=SSPI"

DTPicker1.Value = Date

DTPicker2.Value = Date

Call ready

End Sub

Private Sub ready()

G1.Clear

G1.FormatString = "BillNo |CustomerID|Company |Datee |ItemName
|Quantity |Price |"

G1.Rows = 1

G1.Cols = 7

End Sub

spayment.frm

Dim table As New ADODB.Recordset

Dim Database As New ADODB.Connection

Private Sub Command1_Click()

Dim X As String

Dim toplam As Double

Dim i As Integer

Dim y As String

Dim top As Double

Dim a As Integer

For i = 0 To G1.Rows - 1

 X = G1.TextMatrix(i, 2)

 toplam = toplam + Val(X)

Next

Text1.Text = toplam

For a = 0 To G1.Rows - 1

 y = G1.TextMatrix(a, 3)

 top = top + Val(y)

Next

Text2.Text = top

End Sub

Private Sub Command2_Click()

tarih1 = DTPicker1

tarih2 = DTPicker2

Set table = New ADODB.Recordset

table.Open "select * from Receipt where Datee between '" & tarih1 & "' and '" & tarih2 & "' ", Database

While Not table.EOF

 G1.AddItem table.Fields("BillNo") & Chr(9) & table("Surname") & Chr(9) &
table("Payment") & Chr(9) & table("Debt") & Chr(9) & table("Datee") & Chr(9) &
table("Statu")

 table.MoveNext

Wend

End Sub

Private Sub Form_Load()

Set Database = New ADODB.Connection

Set table = New ADODB.Recordset

Database.Open "Provider=SQLOLEDB;data source=(local);Initial
Catalog=proje;Integrated Security=SSPI"

DTPicker1.Value = Date

DTPicker2.Value = Date

Call ready

End Sub

Private Sub ready()

G1.Clear

G1.FormatString = "BillNo |Surname |Payment |Debt |Date |Statu
"

G1.Rows = 1

G1.Cols = 6

End Sub

spayto.frm

Dim table As New ADODB.Recordset

Dim Database As New ADODB.Connection

Private Sub Command1_Click()

Dim X As String

Dim toplam As Double

Dim i As Integer

For i = 0 To G1.Rows - 1

X = G1.TextMatrix(i, 3)

```

    toplam = toplam + Val(X)
Next
Text2.Text = toplam

Dim y As String
Dim top As Double
Dim a As Integer

For a = 0 To G1.Rows - 1
    y = G1.TextMatrix(a, 2)
    top = top + Val(y)
Next
Text1.Text = top

End Sub
-----

Private Sub Command2_Click()

    tarih1 = DTPicker1
    tarih2 = DTPicker2

    Set table = New ADODB.Recordset

    table.Open "select * from Receiptto where Datee between '" & tarih1 & "' and '" &
    tarih2 & "' ", Database

    While Not table.EOF
        G1.AddItem table.Fields("BillNo") & Chr(9) & table("Company") & Chr(9) &
        table("Payment") & Chr(9) & table("Debt") & Chr(9) & table("Datee")
        table.MoveNext
    Wend

End Sub
-----

```


Private Sub Form_Load()

Set Database = New ADODB.Connection

Set table = New ADODB.Recordset

Database.Open "Provider=SQLOLEDB;data source=(local);Initial
Catalog=proje;Integrated Security=SSPI"

DTPicker1.Value = Date

DTPicker2.Value = Date

Call ready

End Sub

Private Sub ready()

G1.Clear

G1.FormatString = "BillNo |Company |Payment |Debt |Date "

G1.Rows = 1

G1.Cols = 5

End Sub

add_customer.frm

Dim Database As New ADODB.Connection

Dim table As New ADODB.Recordset

Dim i As Integer

Dim bilgi(0 To 9) As String

Dim table1 As New ADODB.Recordset

Private Sub add_cmd_Click()

```

    If name_txt.Text <> "" And surname_txt.Text <> "" And tell_txt.Text <> "" And
kimlikno_txt.Text <> "" And address_txt.Text <> "" Then
        Set table = New ADODB.Recordset
        table.Open "INSERT INTO liste (Name,Surname,Address,Telephone,TC_KimlikNo)
VALUES('" & name_txt.Text & "','" & surname_txt.Text & "','" & address_txt.Text &
"',"' & tell_txt.Text & "','" & kimlikno_txt.Text & "'))", Database
        id_txt.Text = ""
        name_txt.Text = ""
        surname_txt.Text = ""
        tell_txt.Text = ""
        kimlikno_txt.Text = ""
        address_txt.Text = ""
        search_txt.Text = ""
    Else
        MsgBox "you cant leave the name and surname empty", 48, "Caution Message"
        If name_txt.Text = "" Then
            name_txt.SetFocus
        End If
        If surname_txt.Text = "" Then
            surname_txt.SetFocus
        End If
        If tell_txt.Text = "" Then
            tell_txt.SetFocus
        End If
        If kimlikno_txt.Text = "" Then
            kimlikno_txt.SetFocus
        End If
        If address_txt.Text = "" Then
            address_txt.SetFocus
        End If
    End If
End Sub

```

Private Sub Command1_Click()

Unload Me

End Sub

Private Sub del_cmd_Click()

answer = MsgBox("Do you want to DELETE?", vbQuestion + vbYesNo +
vbDefaultButton1)

If answer = vbYes Then

Set table = New ADODB.Recordset

table.Open " DELETE FROM liste WHERE CID='" & id_txt & "'", Database

End If

Call ready

Set table1 = New ADODB.Recordset

table1.Open " SELECT * FROM liste WHERE CID LIKE '%" & id_txt.Text & "%",
Database

While Not table1.EOF

G1.AddItem table1("CID") & Chr(9) & table1("Name") & Chr(9) &
table1("Surname") & Chr(9) & table1("Address") & Chr(9) & table1("Telephone") &
Chr(9) & table1("KimlikNo")

table1.MoveNext

Wend

id_txt.Text = ""

name_txt.Text = ""

surname_txt.Text = ""

tell_txt.Text = ""

kimlikno_txt.Text = ""

address_txt.Text = ""

search_txt.Text = ""

End Sub

Private Sub Form_Load()

Call ready

Set Database = New ADODB.Connection

Set table = New ADODB.Recordset

Database.Open "Provider=SQLOLEDB;data source=(local);Initial
Catalog=proje;Integrated Security=SSPI"

End Sub

Private Sub ready()

G1.Clear

G1.FormatString = "CID |Name |Surname |Tel |KimlikNo
|Address "

G1.Cols = 6

G1.Rows = 1

End Sub

Private Sub G1_Click()

For i = 0 To 5

bilgi(i) = G1.TextMatrix(G1.MouseRow, i)

Next

id_txt.Text = bilgi(0)

name_txt.Text = bilgi(1)

surname_txt.Text = bilgi(2)

tell_txt.Text = bilgi(3)

kimlikno_txt.Text = bilgi(4)

address_txt.Text = bilgi(5)

End Sub

Private Sub search_cmd_Click()

```
Set table = New ADODB.Recordset
table.Open "SELECT * FROM liste WHERE Name LIKE '%" & search_txt.Text &
"%"', Database
```

Call ready

While Not table.EOF

```
G1.AddItem table("CID") & Chr(9) & table("Name") & Chr(9) &
table("Surname") & Chr(9) & table("Address") & Chr(9) & table("Telephone") &
Chr(9) & table("TC_KimlikNo")
```

```
table.MoveNext
```

Wend

End Sub

Private Sub update_cmd_Click()

```
Set table = New ADODB.Recordset
table.Open " UPDATE liste SET Name='" & name_txt & "', Surname='" &
surname_txt & "', Address='" & address_txt & "', Telephone='" & tell_txt & "',
TC_KimlikNo='" & kimlikno_txt & "' WHERE CID='" & id_txt & "'", Database
```

```
id_txt.Text = ""
```

```
name_txt.Text = ""
```

```
surname_txt.Text = ""
```

```
tell_txt.Text = ""
```

```
kimlikno_txt.Text = ""
```

```
address_txt.Text = ""
```

```
search_txt.Text = ""
```

```
Call search_cmd_Click
```

End Sub