NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

Selling Package Program Using Visual Basic

Graduation Project
COM-400

Student: Mahmut SAVRANOĞLU
(960217)

Supervisor: Mr. Ümit İLHAN

Nicosia-2005

ACKNOWLEDGEMENTS

"First, I would like to thank my supervisor Mr. Ümit İlhan for his invaluable advice and belief in my work and my self over the course of this Graduation Project.

Second,Iwould like to express my gratitude to Near East University for the scholarship that made the study possible.

Third, I would like to thank my family for their constant ancouragement and support during the preparation of this Project.

Finally, I would like to thank my all friends for their advice and support."

ABSTRACT

Commercial success and value creation depend on the dedication and commitment of employees in all functions and at all levels of the corporate hierarchy. Stock programs are designed to motivate employees and enhance their personal performance by allowing them to benefit directly from the company's success.

Eligibility for the different programs varies with an employee's position in the company, since people at higher levels of management are considered to have a greater direct influence on the company's economic performance and thus on the performance of stock. Therefore, stock program *Selling Package Program* offered.

Selling Package Program offers the opportunity to receive cash awards which depend on the share price development of the stocks over a period of years.

TABLE OF CONTENTS

CHAPTER TWO: A BRIEF HISTORY OF VISUAL BASIC

STOCK CONTROL AND INVENTORY

OVERVIEW

Stock control, otherwise known as inventory control, is about how much stock you have at any one time, and how you keep track of it.

It applies to every item you use to produce a product or service, from raw materials to finished goods. It covers stock at every stage of the production process, from purchase and delivery to using the stock and re-ordering.

Efficient stock control will mean you have the right amount of stock in the right place at the right time. It ensures that capital is not tied up unnecessarily, and protects production when there are problems with the supply chain.

This guide explains different stock control methods, will help you set one up and tells you where to find more information.

1.STOCK CONTROL

1.1 Types of stock

Everything you use to make your products, provide your services and to run your business is part of your stock.

There are four main types of stock:

- raw materials and components - ready to use in production
- work in progress - stocks of unfinished goods
- finished goods ready for sale
- consumables - for example, fuel and stationery

The type of stock can influence how much you should keep - see the page in this guide on how much stock should you keep?

1.2 Stock value

You can categorise stock further, according to its value. For example, you could put items into low, medium and high value categories. If you feel your stock levels are limited by capital, this will help you to plan expenditure on new and replacement stock.

You may choose to concentrate resources on the areas of greatest value. However, low-cost items can be crucial to your production process and should not be overlooked.

1.3 How much stock should you keep?

Deciding how much stock to keep depends on the size and nature of your business, and the type of stock involved.

Table .1 Keeping little or no stock

Advantages	**Disadvantages**
Efficient and flexible. You only have what you need, when you need it.	Meeting stock needs can become complicated and expensive.
Lower stock and storage costs.	You might run out of stock if there's a hitch in the system.
You can keep up-to-date and develop new products without wasting stock.	You are dependent on the efficiency of your suppliers.

This might suit your business if it's in a fast-moving environment where products develop rapidly, the stock is expensive to buy and store, the items are perishable or replenishing stock is quick and easy.

Table.2-Keeping lots of stock

Advantages	**Disadvantages**
Easy to manage.	Higher stock, storage and insurance costs.
Low management costs.	Certain goods might perish.
You never run out.	Stock may become obsolete before it is used.
	Your capital is tied up.

This might suit your business if sales are difficult to predict (and it is hard to pin down how much stock you need and when), you can store plenty of stock cheaply, the components or materials you buy are unlikely to go through rapid developments or they take a long time to re-order.

Stock levels depending on type of stock

There are four main types of stock:

1.4 Raw materials and components

Ask yourself some key questions to help decide how much stock you should keep:

- How reliable is the supply?
- Are the components produced or delivered in batches?
- Can you predict demand?
- Is the price steady?
- Are there discounts if you buy in bulk?

1.5 Work in progress - stocks of unfinished goods

Keeping stocks of unfinished goods can be a useful way to protect production if there are problems down the line with other supplies.

1.6 Finished goods ready for sale

You might keep stocks of finished goods when:

- demand is certain goods are produced in batches you are completing a large order

1.7 Consumables

For example, fuel and stationery. How much stock you keep will depend on factors such as:

- reliability of supply
- expectations of price rises
- how steady demand is discounts for buying in bulk .

1.8 Stock control methods

There are several methods for controlling stock, all designed to provide an efficient system for deciding what, when and how much to order.

You may opt for one method or a mixture of two or more if you have various types of stock. For further information, see the page in this guide on types of stock.

Minimum stock level - you identify a minimum stock level, and re-order when stock reaches that level. This is known as the Re-order Level (ROL).

Stock review - you have regular reviews of stock. At every review you place an order to return stocks to a predetermined level.

Just In Time (JIT) - this aims to reduce costs by cutting stock to a minimum - see our guide on how to avoid the problems of overtrading. Items are delivered when they are needed and used immediately. There is a risk of running out of stock, so you need to be confident that your supplier can deliver.

These methods can be used alongside other processes to refine the system. For example:

Re-order lead time - allows for the time between placing an order and receiving it.

Economic Order Quantity (EOQ) - a standard formula used to arrive at a balance between holding too much or too little stock. It's quite a complex calculation, so you may find it easier to use stock control software.

Batch control - managing the production of goods in batches. You need to make sure that you have the right number of components to cover your needs until the next batch.

If your needs are predictable, you may order a fixed quantity every time, or at a fixed interval - say every week or month. In effect, you're placing a standing order, so you need to keep the quantities and prices under review.

First in, first out - a system to ensure that perishable stock is used efficiently so that it doesn't deteriorate. Stock is identified by date received and moves on through each stage of production in strict order.

1.9 Stock control systems - keeping track manually

Stocktaking involves making an inventory, or list, of stock, and noting its location and value. It's often an annual exercise - a kind of audit to work out the value of the stock as part of the accounting process.

Codes, including barcodes, can make the whole process much easier but it can still be quite time-consuming. Checking stock more frequently - a rolling stocktake - avoids a massive annual exercise, but demands constant attention throughout the year.

Any stock control system must enable you to:

- track stock levels
- make orders
- issue stock

The simplest manual system is the stock book, which suits small businesses with few stock items. It enables you to keep a log of stock received and stock issued.

It can be used alongside a simple re-order system. For example, the two-bin system works by having two containers of stock items. When one is empty, it's time to start using the second bin and order more stock to fill up the empty one.

Stock cards are used for more complex systems. Each type of stock has an associated card, with information such as:

- description
- value
- location
- re-order levels quantities and lead times (if this method is used)
- supplier details information about past stock history

More sophisticated manual systems incorporate coding to classify items. Codes might indicate the value of the stock, its location and which batch it is from, which is useful for quality control.

1.10 Stock control systems - keeping track using computer software

Computerised stock control systems run on similar principles to manual ones, but are more flexible and information is easier to retrieve. You can quickly get a stock valuation or find out how well a particular item of stock is moving.

A computerised system is a good option for businesses dealing with many different types of stock. Other useful features include:

Stock and pricing data integrating with accounting and invoicing systems. All the systems draw on the same set of data, so you only have to input the data once. Sales Order Processing (SOP) and Purchase Order Processing (POP) can be integrated in the system so that stock balances and statistics are automatically updated as orders are processed.

Automatic stock monitoring, triggering orders when the re-order level is reached.

Automatic batch control if you produce goods in batches.

Identifying the cheapest and fastest suppliers.

Bar coding systems which speed up processing and recording. The software will print and read bar codes from your computer.

The system will only be as good as the data put into it. Run a thorough stocktake before it goes "live" to ensure accurate figures. It's a good idea to run the previous system alongside the new one for a while, giving you a back-up.

1.11 Choose a system

There are many software systems available. Talk to others in your line of business about the software they use, or contact your trade association for advice.

Your needs might include:

- multiple prices for items prices in different currencies automatic updating, selecting groups of items to update, single-item updating
- using more than one warehouse
- ability to adapt to your changing needs
- quality control and batch tracking
- integration with other packages
- multiple users at the same time

Avoid choosing software that's too complicated for your needs as it will be a waste of time and money.

1.12 Stock security

Keeping stock secure depends on knowing what you have, where it is located and how much it is worth - so good records are essential. Stock that is portable, does not feature the business' logo, or is easy to sell on is at particular risk.

1.13 Thieves and shoplifters

A thief coming in from outside is an obvious threat. Check the security around your premises to keep the risk to a minimum. In a shop, thieves may steal in groups - some providing a distraction while others take goods. Teach your staff to be alert and to recognise behaviour like this. Set up a clear policy and make sure staff are trained in dealing with thieves.

Offering to help a customer if you are suspicious will often prevent a theft. Avoid using confrontational words like "steal" if you do have to approach a suspected thief, and avoid getting into a dangerous situation.

1.14 Theft by staff

Employee theft can sometimes be a problem. To prevent this:

Create an honest culture among your staff. Training about the cost of stock theft will help, as many people aren't aware of the implications for company turnover and job security.

Set up procedures to prevent theft. Staff with financial responsibilities should not be in charge of stock records.

Restrict access to warehouses, stockrooms and stationery cupboards.

Identify and mark expensive portable equipment (such as computers).

Don't leave equipment hanging around after delivery. Put it away in a secure place, record it and clear up packaging.

Regularly change staff controlling stock to avoid collusion or bad practice.

1.15 Control the quality of your stock

Quality control is a vital aspect of stock control - especially as it may affect the safety of customers.

Efficient stock control should incorporate stock tracking and batch tracking. This means being able to trace a particular item backwards or forwards from source to finished product, and identifying the other items in its batch.

Goods should be checked systematically for quality, faults identified and the affected batch weeded out, enabling you to raise the matter with your supplier and at the same time to demonstrate the safety and quality of your product.

With a good computerised stock control system, this kind of tracking is relatively straightforward. Manual stock control methods can also use codes to systematise tracking and make it easier to trace particular batches.

The British Standards Institution (BSI) has a scheme to certify businesses that have achieved a certain standard of quality management. Achieving the standard is one way of showing customers and regulators that you take quality control seriously.

The details vary according to the particular business sector. For some industries, it is a legal requirement to be certified.

1.16 Stock control administration

There are many administrative tasks associated with stock control. Depending on the size and complexity of your business, they may be done as part of an administrator's duties, or by a dedicated stock controller.

For security reasons, it's good practice to have different staff responsible for finance and stock.

Typical paperwork to be processed includes:

- delivery and supplier notes for incoming goods
- purchase orders, receipts and credit notes
- returns notes
- requisitions and issue notes for outgoing goods

Stock can tie up a large slice of your business capital, so accurate information about stock levels and values is essential for your company's accounting.

Figures should be checked systematically, either through a regular audit of stock - stocktaking - or an ongoing programme of checking stock - rolling stocktake.

If the figures don't add up, you need to investigate as there could be stock security problems or a failure in the system.

1.17 Health and safety

Health and safety aspects of stock control are related to the nature of the stock itself. Issues such as where and how items are stored, how they are moved and who moves them might be significant - depending on what they are.

You might have hazardous materials on your premises, goods that deteriorate with time or items that are very heavy or awkward to move.

1.18 Here's how I control my stock

Andrea Jones is managing director of Liversedge-based Systems (Telecoms) Limited, a business specialising in the next-day delivery of refurbished telecommunications equipment.

2.A BRIEF HİSTORY OF VİSUAL BASİC

Before Visual Basic 1.0 was introduced to the world in 1991, developers had to be well versed in C++ programming, as well as the rudimentary building blocks (Windows API) of the Windows system itself. This complexity meant that only the dedicated and properly trained were capable of turning out software that could run on Windows. Visual Basic changed all of that, and it has been estimated that there are now as many lines of production code written in Visual Basic as in any other language.

Visual Basic changed the face of Windows programming by removing the complex burden of writing code for the user interface (UI). By allowing programmers to *draw* their own UI, it freed them to concentrate on the business problems they were trying to solve. Once the UI is drawn, the programmer can then add code to react to events.

Visual Basic has also been extensible from the very beginning. Third-party vendors quickly saw the market for reusable modules to aid developers. These modules, or controls, were original referred to as VBXs (named after their file extension). If you didn't like the way a button behaved you could either buy or create your own. However, these controls had to be written in C or C++. Database access utilities were some of the first controls available.

When Microsoft introduced Visual Basic 3.0, the programming world changed again. Now you could build database applications directly accessible to users (so called front-end applications) completely with Visual Basic. There was no need to rely on third-party controls. Microsoft accomplished this task with the introduction of the Data Access Objects **(DAO),**

which allowed programmers to manipulate data with the same ease as manipulating the user interface.

Versions 4.0 and 5.0 extended the capabilities of version 3.0 in order to allow developers to target the new Windows 95 platform. Crucially they also made it easier for developers to write code, which could then be manipulated in order to be used by other language developers. Version 6.0 gave us a new way to access databases with the integration of ActiveX Data Objects (ADO). ADO was developed by Microsoft to aid web developers using Active Server Pages to access databases.

With all of the improvements to Visual Basic over the years, it ensured its dominant place in the programming world. It helps developers write robust and maintainable applications in record time.

With the release of Visual Basic .NET, many of the restrictions that used to exist have been obliterated. In the past, Visual Basic has been criticized and maligned as a "toy" language, as it did not provide all of the features of more sophisticated languages such as C++ and Java. Now, Microsoft has removed these restrictions and made Visual Basic .NET a very powerful development tool. Visual Basic .NET has become a great choice for programmers of all levels.

You don't actually need the Visual Basic .NET product to write applications in the Visual Basic .NET language. The actual ability to run Visual Basic .NET code is included with the .NET Framework. You could actually just write all of your Visual Basic .NET using a text editor such as Notepad.

However, by far the easiest way to write in Visual Basic .NET is by using the Visual Studio.NET Integrated Development Environment, also known as the IDE. This is what you actually see when working with Visual Basic .NET – the windows, boxes, etc. The IDE provides a wealth of features that are unavailable in ordinary text editors – such as code checking, visual representations of the finished application, and an explorer that displays all of the files that make up your project.

2.1 The Profile Setup Page

An IDE is a way of bringing together a suite of tools that make developing software a lot easier. Let's fire up Visual Basic .NET and see what we've got. If you used the default installation, go to your Windows Start menu and then Programs | Microsoft Visual Studio.NET 7.0 | Microsoft Visual Studio.NET 7.0. A splash screen will briefly appear and then you should find yourself present the Start screen's My Profile tab:

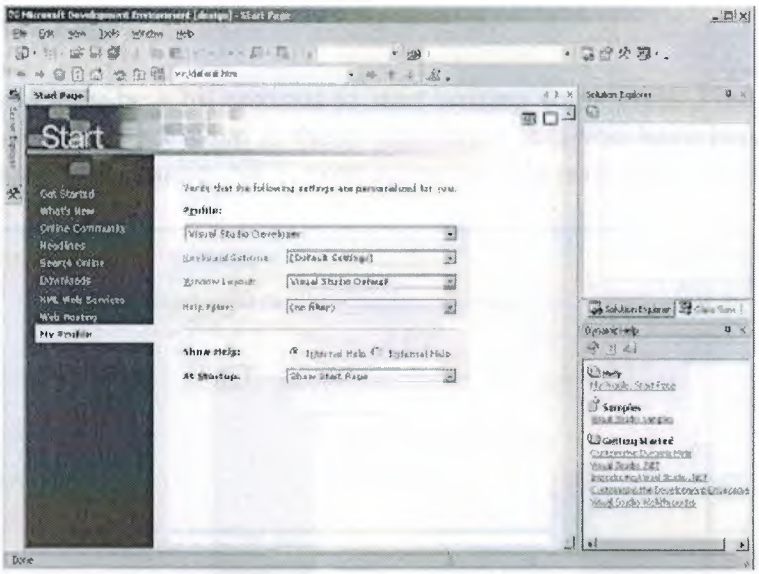

Figure.1

This screen allows us to do some basic configuration of the IDE so that it serves us better. Since this IDE serves all the Visual Studio .NET languages, there are some settings to tailor it to our particular development interests.

2.2 Try It Out – Setting Up Our Profile

1. We are learning how to program using Visual Basic .NET, so select Visual Basic Developer from the drop-down box. The IDE will now rearrange itself (it actually looks similar to the Visual Basic 6 IDE).
2. You will also notice that the Keyboard Scheme and Window Layout options have changed to show: Visual Basic 6. If you are at all familiar with earlier versions of Visual Basic, then this should make you feel right at home. The Window Layout options rearrange the windows in the IDE to look similar to previous versions of other Microsoft development tools. How you lay out your windows in the future will be a matter of preference, but for now let's use the Visual Basic 6 option.
3. The Help Filter drop-down box also allows the help system to focus better on what you will find most useful. Set the Help Filter to Visual Basic.
4. The Show Help radio buttons allow us to select where help topics are displayed – Internal Help shows topics within the IDE window (the same window where you see the Start Page), whereas the External Help option opens topics in a separate window. This choice between Help displayed within the IDE or in a separate window is a matter of personal preference. However, until you are comfortable manipulating the various windows of the IDE, the external option might prove more useful, as the IDE remains constant while Help is open. You may receive a message that the change will not take effect until the next time you start Visual Studio .NET.
5. The At Startup pull-down permits you to define what you see whenever you first start Visual Studio .NET. The options here are:
6. Once you have configured your profile to your liking, select Get Started from the vertical menu bar to the left to begin using Visual Basic .NET.

2.3 The Get Started Page

By now, you may be a bit anxious to start writing some code. First, let's take a brief look at Started tab and see what is there. Assuming that you have been following along while setting up Visual Studio .NET, your screen should now look something like this:

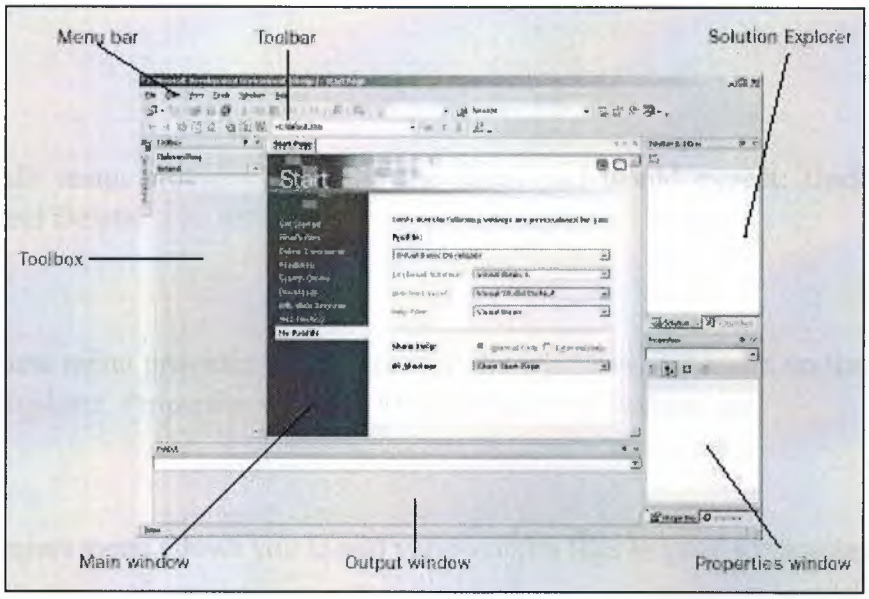

Figure.2

Let's begin our exploration of the Visual Basic .NET IDE by looking at the toolbar and menu, which as you'll learn are not really that different from toolbars and menus you'll have seen in other Microsoft software such as Word, Excel, and PowerPoint.

2.4 The Menu

Visual Studio .NET's menu is dynamic, meaning that items will be added or removed depending on what you are trying to do. While we are still looking at the Get Started page, the menu bar will only consist of the File, Edit, View, Tools, Window, and Help menus. However, when you start working on a project, the full Visual Studio .NET menu appears as:

Figure.3

At this point, there is no need to cover each menu topic in great detail. You will become familiar with each as you progress through the book. Here is a quick rundown of what activities each menu item pertains to:

2.4.1 File

It seems every Windows program has a File menu. It has become the standard where you should find, if nothing else, a way to exit the application. In this case, you can also find ways of opening and closing single files and whole projects.

2.4.2 Edit

The Edit menu provides access to the items you would expect: Undo, Redo, Cut, Copy, Paste, and Delete.

2.4.3 View

The View menu provides quick access to the windows that make up the IDE, such as the Solution Explorer, Properties window, Output window, Toolbox, etc.

2.4.4 Project

The Project menu allows you to add various extra files to your application.

2.4.5 Build

The Build menu becomes important when you have completed your application and want to be able to run it without the use of the Visual Basic .NET environment (perhaps running it directly from your Windows Start menu as you would any other application such as Word or Access).

2.4.6 Debug

The Debug menu allows you to start and stop running your application within the Visual Basic .NET IDE. It also gives you access to the Visual Studio .NET **debugger**. The debugger allows you to step through your code while it is running to see how it is behaving.

2.4.7 Data

The Data menu helps you use information that comes from a database. It only appears when you are working with the visual part of your application (the [Design] tab will be the active one in the main window), not when you are writing code. Chapters 15 and 16 will introduce you to working with databases.

2.4.8 Format

The Format menu also only appears when you are working with the visual part of your application. Items on the Format menu allow you to manipulate how the windows you create will appear to the users of your application.

2.4.9 Tools

The Tools menu has commands to configure the Visual Studio .NET IDE, as well as links to other external tools that may have been installed.

2.4.10 Window

The Window menu has become standard for any application that allows more than one window to be open at a time, such as Word or Excel. The commands on this menu allow you to change the physical layout of the windows in the IDE.

2.4.11 Help

The Help menu provides access to the Visual Studio .NET documentation. There are many different ways to access this information (for example, via the help contents, an index, or a search). The Help menu also has options that connect to the Microsoft Web site to obtain updates or report problems.

2.5 The Toolbars

There are many toolbars available within the IDE, including Formatting, Image Editor, and Text Editor, which you can add to and remove from the IDE via the View | Toolbars menu option. Each one provides quick access to often-used commands, preventing you from having to navigate through a series of menu options. For example, the leftmost icon on the toolbar shown below (New Project) is available from the menu by navigating to File | New | Project.

The default toolbar (called Standard) appears at the top of the IDE as:

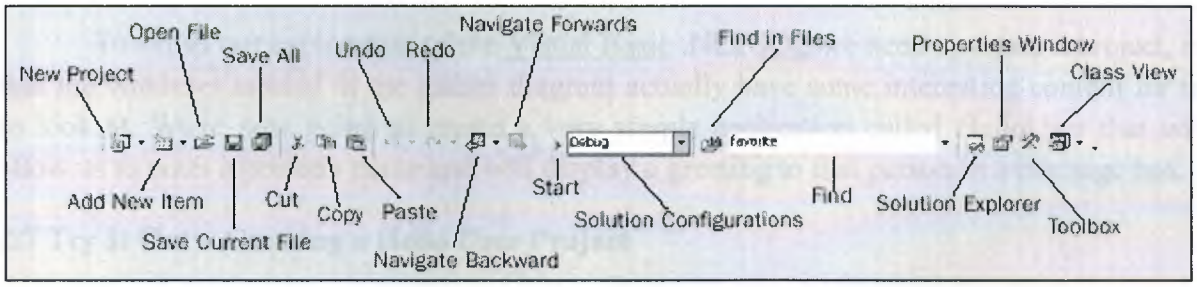

Figure.4

The toolbar is segmented into groups of related options, which are separated by a vertical bar. The first five icons provide access to the commonly used project and file manipulation options available through the File and Project menus, such as opening and saving files. The next group of icons is for editing (Cut, Copy, and Paste).

The third group of icons is for editing and navigation. The navigation buttons replicate functionality found in the View menu and allow us to cycle through the tabs at the top of the main window.

The fourth group of icons provides the ability to start your application running (via the blue triangle) and to specify build configurations. There are times when you want certain parts of your code only to appear in a debug version, a bit like a rough draft version of your application. For example, you may have code in your application that is only useful for tracking down problems in the application.

When it is time to release your application to the world, you will want to exclude this code by setting the Solution Configurations settings to Release. You can also access the functionality offered by this group via the Build and Debug menus.

The next section allows you to locate parts of your code quickly. The simplest way to search is to type some text into the Find textbox and hit Enter. If the text is found, it will be highlighted in the central window. The Find in Files option allows you to specify more sophisticated searches, including matching the case of the text, looking in specific files or projects, and replacing the found text with new text. The search functionality can also be accessed via the Edit | Find and Replace menu option.

The next group of icons provides quick links back to the Solution Explorer, Properties window, Toolbox, and Class View. If any of these windows are closed, clicking the appropriate icon will bring it back into view.

We could continue to look at each of the other windows directly from the Start Page. But, as you can see they're all empty at this stage, and therefore not too revealing. The best way to look at the capabilities of the IDE is to use it

2.6 Creating a Simple Application

To finish our exploration of the Visual Basic .NET IDE we need to create a project, so that the windows labeled in the earlier diagram actually have some interesting content for us to look at. We're now going to create a very simple application called HelloUser that will allow us to enter a person's name and will display a greeting to that person in a message box.

2.7 Try It Out – Creating a Hello User Project

a. Click on the New Project button on the Start Page.

b. The New Project dialog box will open. Make sure you have Visual Basic Projects selected in the Project Types tree-view box to the left. Next, select Windows Application in the Templates box on the right. If you need to save this project to a location other than the default, be sure to enter it into the Location box. Finally, type HelloUser in the Name text box and click on the OK button:

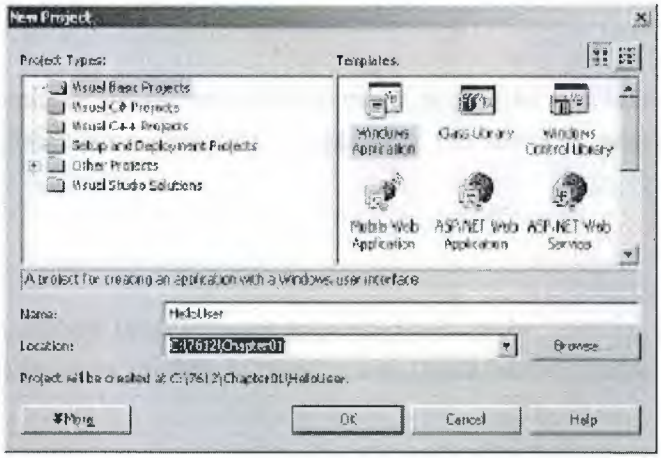

Figure.5

c. Visual Basic .NET will then create an empty Windows application for us. So far, our HelloUser program consists of one blank window called a Windows Form (or sometimes just a form), with the default name of Form1.vb:

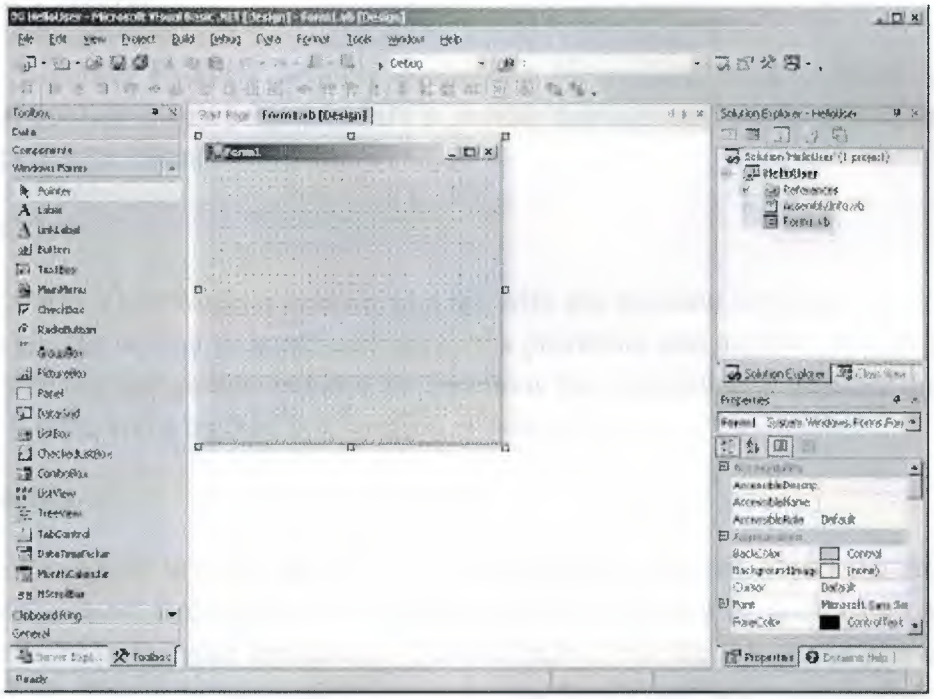

Figure.6

At this point, you can see that the various windows in the IDE are beginning to show their purposes, and we will take a brief look at them now before we come back to the Try It Out. Note that if any of these windows are not visible on your screen, you can use the View menu to select and show them. Also if you do not like the location of any particular window you can move it by clicking on its title bar (the blue bar at the top) and dragging it to a new location. The windows in the IDE can float (stand out on their own) or be dockable (as they appear above).

2.8 Server Explorer

The Server Explorer gives you management access to the servers on your network. Here you can create database connections, and view the services provided by the available servers.

2.8.1 Toolbox

The Toolbox contains reusable components that can be inserted into your application. These can range from buttons to data connectors to customized controls either purchased or developed yourself.

2.8.2 Design Window

The Design window is where a lot of the action takes place. This is where you will draw your user interface and write your code. This window is sometimes referred to as the Designer.

2.9 Solution Explorer

The Solution Explorer window contains a hierarchical view of your solution. A solution can contain many projects while a project contains code and code references that solve a particular problem.

2.9.1 Class View

The Class View window (shown as a tab with the Solution Explorer) gives you a tree view of the classes in your program and shows the properties and methods that each contains. A class is code file that groups data and the functions that manipulate it together into one unit. A property is data, and a method is a function or subroutine.

2.9.2 Properties

The Properties window shows what properties the selected object makes available. Although you can set these properties in your code, sometimes it is much easier to set them while you are designing your application. You will notice that the File Name property has the value Form1.vb. This is the physical file name for the form's code and layout information.

2.9.3 Task List

The Task List window highlights any errors encountered when you try to run your code. Clicking on the item in this window will take you to the line of code containing the error.

2.9.4 Output

When you run your code the progress made in reading it (or compiling it) is registered via messages posted in the Output window.

2.9.5 Dynamic Help

The Dynamic Help window displays a list of help topics that relate to whatever in the IDE has focus. If you click on the form in the Design Window and then open Dynamic Help, you will see a list of help topics relating to forms.

2.10 Try It Out – Creating a Hello User Project – Continued

a. Let's change the name of our form to something more indicative of what our application is. Click on Form1.vb in the Solution Explorer window. Then, in the Properties window, change the File Name property from Form1.vb to HelloUser.vb and hit *Enter*: When changing properties you must either hit *Enter* or click off the property for it to take effect.

Figure.7

b. Notice that the form's file name has also been updated in the Solution Explorer to read HelloUser.vb:

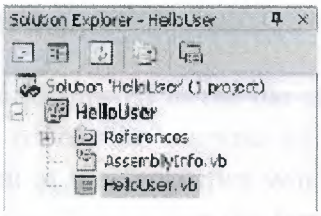

Figure.8

c. Now click on the form displayed in the main window; the Properties window will change to display the form's Form properties (instead of the File properties, which we have just been looking at). You will notice that the Properties window is dramatically different. The difference is the result of two different views of the same file.

When the form name is highlighted in the Solution Explorer window, the physical file properties of the form are displayed. When the form in the Design View is highlighted, the visual properties and logical properties of the form are displayed.

The Properties window allows us to easily set a control's properties. Remember properties are a particular object's set of internal data. Properties usually describe appearance or behavior. In the screenshot, you can see that properties are grouped together in categories – Accessibility, Appearance, and Behavior are the ones shown in here:

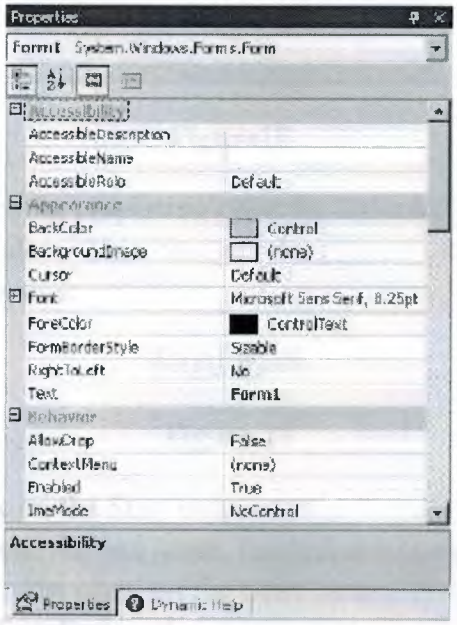

Figure.9

You can see that under the Appearance category are such properties as BackColor (the form's background color), Font (the typeface used for text on the form), and Text (the form's caption displayed in the title bar). These are all examples of properties that describe the form's appearance. One property, FormBorderStyle, tells Windows how to draw the form. In this case, the window is resizable, meaning that when the form is displayed, the user can change its size (like most application windows).

d. Right now, the title of our form (displayed in the bar at the top) is Form1. This isn't very descriptive, so let's change it to reflect the purpose of this application. Locate the Text property in the Appearance section of the Properties window and change its value to Hello from Visual Basic .NET and hit Enter. Notice that the form's title has been updated to reflect the change:

Figure10

If you have trouble finding properties, click the little AZ button on the toolbar towards the top of the Properties window. This changes the property listing from being ordered by category to being ordered by name:

Figure.11

e. We are now finished. Click on the Start button on the Visual Studio .NET toolbar (the blue triangle) to run the application. As you work through the book, whenever we say "run the project" or "start the project", just click on the Start button. An empty window with the title Hello from Visual Basic .NET is displayed:

Figure.12

OK, that was simple, but our little application isn't doing much at the moment. Let's make it a little more interactive. To do this we are going to add some controls – a label, textbox, and two buttons to the form. This will let us see how the toolbox makes adding functionality quite simple. You may be wondering at this point when we will actually look at some code. Soon! The great thing about Visual Basic .NET is that you can develop a fair amount of your application without writing any code. Sure, the code is still there, behind the scenes, but as we'll see, Visual Basic .NET writes a lot of it for us.

2.11 The Toolbox

The Toolbox is accessed via the View | Toolbox menu option, the Toolbox icon on the Standard menu bar, or by pressing Ctrl + Alt + X.

The Toolbox contains a tabbed view of the various controls and components that can be placed onto your form. Controls such as textboxes, buttons, radio buttons, and drop-down boxes can be selected and then drawn onto your form. For the HelloUser application, we will only be using the controls on the Windows Forms tab:

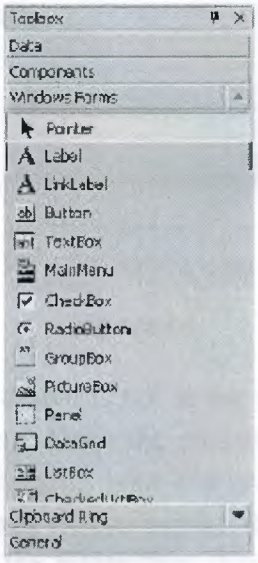

Figure.13

Here we can see a listing of standard .NET controls for Windows forms. The down arrow button to the right of the Clipboard Ring tab title actually scrolls the Windows Forms control list down as there are too many to fit in otherwise. The up arrow button on the Windows Forms tab scrolls the list up. Note that the order in which your controls appear may be different. Controls can be added to your forms in any order, so it does not matter if we add the label control after the textbox or the buttons before the label.

2.12 Try It Out – Adding Controls to the Hello User Application

a. Stop the project if it's still running, as we now want to add some controls to our form. The simplest way to do this is to click on the X button in the top right corner of the form. Alternatively, you can click on the blue square in the Visual Studio .NET IDE (which displays the text Stop Debugging if you hover over it with your mouse pointer).

b. Let's add a Label control to the form. Click on Label in the Toolbox to select it. Move the cursor over the form's Designer. You'll notice that the cursor looks like a crosshair with a little floating letter A beneath it. Click and hold the mouse button where you want the top left corner of the label and drag the mouse to where you want the bottom right. (Placing controls can also be accomplished by double-clicking on the required control in the Toolbox.)

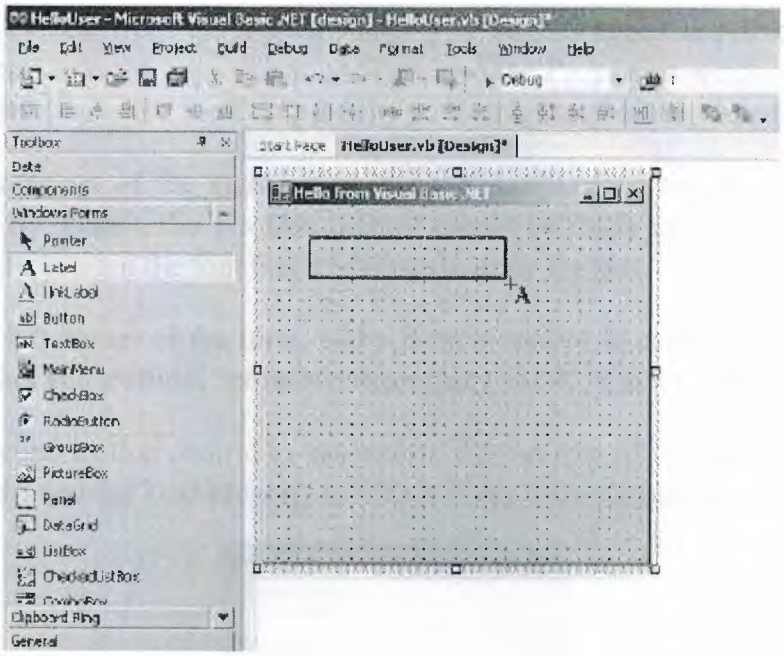

Figure.14

c. If the Label control you have just drawn is not in the your desired location, or is too big or too small, that is not really a problem. Once the control is on the form you can resize it or move it around. The image opposite shows what the control looks like after you place it on the form. To move it, click on the gray dotted border and drag it to the desired location. To resize it, click and drag on one of the white box "handles" and stretch the control in the needed direction. The corner handles resize both the horizontal and vertical dimension at the same time:

Figure.15

d. After drawing a control on the form, we should at least configure its name and the text that it will display. You'll see that the Properties window to the right of the Designer has changed to Label1, telling you that you're currently examining the properties for it.

e. Now, directly beneath the label, we want to add a textbox, so that we can enter a name. We're going to repeat the procedure we followed for adding the label, but this time make sure

you select the TextBox from the toolbar. Once you have dragged-and-dropped (or double-clicked) the control into the appropriate position, use the Properties window to set its Name property to txtName and clear the Text property so that the textbox now appears to be blank.

Notice how, out of the eight sizing handles surrounding the control, only two are shown in white. By default, the TextBox control cannot be made any taller than the absolute height necessary to contain the font that it will use to draw the text.

f. In the bottom left corner of the form, add a Button control in exactly the same manner as you added the label and textbox. Set its Name property to OK, and its Text property to &OK.

g. Now add a second Button control to the bottom right corner of the form and set the Name property to btnExit and the Text property to E&xit. Your form should look similar to this:

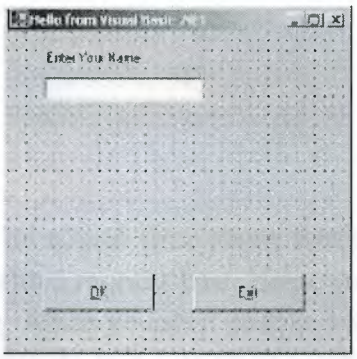

Figure.16

2.13 Modified Hungarian Notation

You may have noticed that the names given to the controls look a little funny. Each name is prefixed with a shorthand identifier describing the type of control it is. This makes it much easier to understand what type of control we are working with when we are looking through code. For example, say we had a control called simply Name, without a prefix of lbl or txt, we would not know whether we were working with a textbox that accepted a name or a label that displayed a name. Imagine if, in the previous Try It Out, we had named our label Name1 and our textbox Name2 – we'd very quickly become confused. How about if we left our application for a month or two, and then came back to it to make some changes?

When working with other developers, it is very important to keep the coding style consistent. One of the most commonly used styles used for controls within application development in many languages is Modified Hungarian notation. The notion of prefixing control names to identify their use was brought forth by Dr. Charles Simonyi.

He worked for the Xerox Palo Alto Research Center (XPARC), before joining Microsoft. He came up with short prefix mnemonics that allowed programmers to easily identify the type of information a variable might contain. Since Dr. Simonyi is Hungarian, and the prefixes make the names look a little foreign, the name Hungarian Notation stuck. Since the original notation was used in C/C++ development, the notation for Visual Basic .NET is

termed Modified. Here is a table of some of the commonly used prefixes that we shall be using in this book:

Table.3

Control	Prefix
Button	btn
ComboBox	cbo
CheckBox	chk
Label	lbl
ListBox	lst
MainMenu	mnu
RadioButton	rdb
PictureBox	pic
TextBox	txt

Hungarian Notation can be a real time-saver when looking at code someone else wrote, or at code that you have written months past. However, by far the most important thing is to be consistent in your naming. When you start coding, pick a convention for your naming. It is recommended that you use the de facto standard Modified-Hungarian for Visual Basic .NET, but it is not required. Once you pick a convention, stick to it. When modifying someone else's code, use theirs. There is very little code that is ever written, put into production and then forgotten. A standard naming convention followed throughout a project will save countless hours when the application is maintained. Now let's get back to the application. It is now time to write some actual code.

2.14 The Code Editor

Now that we have the HelloUser form defined, we have to add some code to actually make it do something interesting. We have already seen how easy it is to add controls to a form. Providing the functionality behind those on-screen elements is not much more difficult. To add the code for a control, just double-click on it. This will open the code editor in the main window:

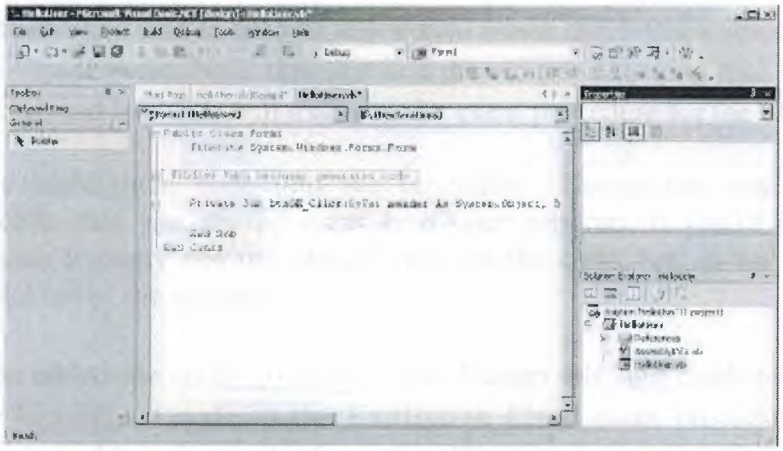

Figure.17

Notice that an additional tab has been created in the main window. Now we have the Design tab and the code tab. We drew the controls on the Design tab, and we write code on the code tab. One thing to note here is that we have not created a separate file for the code. The visual definition and the code behind it both exist in the same file: HelloUser.vb. This is actually the reason why building applications with Visual Basic .NET is so slick and easy. Using the Design view you can visually lay out your application, and then using the Code view add just the bits of code to implement your desired functionality.

You will also notice that there are two pull-downs at the top of the window. These provide shortcuts to the various parts of our code. If you pull down the one on the left, Form1 (HelloUser), you will see a list of all of the objects within our application. If you pull down the one on the right, (Declarations), you will see a list of all of the defined functions or subroutines. If this particular form had a lot of code behind it, these pull-downs would make navigating to the desired area very quick – jumping to the selected area. However, since all of the code fits in the window, there are not a lot of places to get lost.

Now let's look at the code in the window. The code in Visual Studio .NET is set up into regions designated by the plus (+) and minus (-) buttons along the left side. These regions can be collapsed and expanded in order to simplify what you are looking at. If you expand the region labeled Windows Form Designer generated code, you will see a lot of code that Visual Basic .NET has automatically generated for us, which takes care of defining each of the controls on the form and how the form itself should behave. We do not have to worry about the code in this region, so collapse the Windows Form Designer generated code region once more and let's just concentrate on the code we have to write ourselves.

2.15 Try It Out – Adding Code to the Hello User Project

a. To begin adding the necessary code, click on the Design tab to show the form again. Then double-click on the OK button. The code window will reopen with the following code. This is the shell of button's Click event and is the place where we enter code that we want to be run when we click on the button. This code is known as an event handler, and sometimes is also referred to as an event procedure: Sub is an example of a keyword. In programming terms, a

keyword is a special word that is used to tell Visual Basic .NET to do something special. In this case, it tells Visual Basic .NET that this is a procedure. Anything that we type between the lines Private Sub and End Sub will make up the event procedure for the OK button.

b. Now add the highlighted code into the procedure: Throughout this book, you'll be presented with code that you should enter into your program if you're following along. Usually, we'll make it pretty obvious where you put the code, but as we go we'll explain anything that looks out of the ordinary.

c. After you have added the code, go back to the Design tab, and double-click on the Exit button. Add the highlighted code to the ExitButton_Click event procedure. You may be wondering what Me is. Me refers to the form. Just like the pronoun, me, it is just a shorthand for referring to oneself.

d. Now that the code is finished, the moment of truth has arrived and we can see our creation. First though, save your work by using File | Save from the menu, or by clicking the disk icon on the toolbar.

e. Now click on the Start button on the toolbar. You will notice a lot of activity in the Output window at the bottom of your screen. Providing you haven't made any mistakes in entering the code, this information just lets you know what files are being loaded to run your application.

It's at this point that Visual Studio .NET will compile the code. Compiling is the activity of taking the Visual Basic .NET source code that you've written and translating it into a form that the computer understands. After the compilation is complete, Visual Studio .NET will run (also known as execute) the program and we'll be able to see the results.

If Visual Basic .NET encountered any errors, they will be displayed as tasks in the Task List window. Double-clicking on a task will transport you to the offending line of code.

f. When the application loads you will see the main form. Enter a name and click on OK (or press the Alt+O key combination):

Figure.18

g. A window known as a message box appears, welcoming the person whose name was entered in the textbox to Visual Basic .NET – in this case Deborah:

Figure.19

h) After you close the message box by clicking on its OK button, click on the Exit button on our form. The application will close and you will be brought back to the Visual Basic .NET IDE.

That completes our first Visual Basic .NET application. Well done! Before we move on to the next chapter, let's take a quick look at the Visual Studio .NET Help system.

2.16 Using the Help System

The Help system included with Visual Basic .NET is an improvement over Help systems in previous versions. As you begin to learn Visual Basic .NET, you will probably become very familiar with the Help system. However, it is worthwhile to give you an overview, just to help speed your searches for information.

The Help menu appears as:

Figure.20

As you can see this menu contains many more entries than the typical Windows application. The main reason for this is the vastness of the documentation. Few people could keep it all in their heads – but luckily, that's not a problem, as we can always quickly and easily refer to the Help system. Think of it as a safety net for your brain.

One really fantastic new feature is Dynamic Help. If you turn this option on (by selecting Dynamic Help from the Help menu), a window will display a list of relevant topics

for whatever you may be working on. If you followed the default installation and have not rearranged the IDE, the Dynamic Help is displayed as a tab behind Properties.

Let's say for example, that we are working with a textbox (perhaps the textbox in the HelloUser application) and want to find out some information; we just select the textbox on our form and we can see all the help topics that pertain to textboxes:

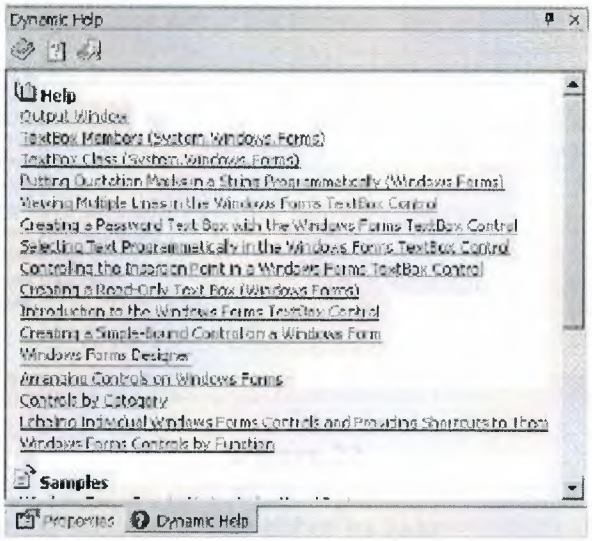

Figure.21

The other help commands in the Help menu (Contents, Index, and Search), function just as they would in any other Windows application.

The domain of help topics to search and display is defined by the profile that we defined at installation. However, the Edit Filters… menu option allows you to further focus what types of documentation to include in the search:

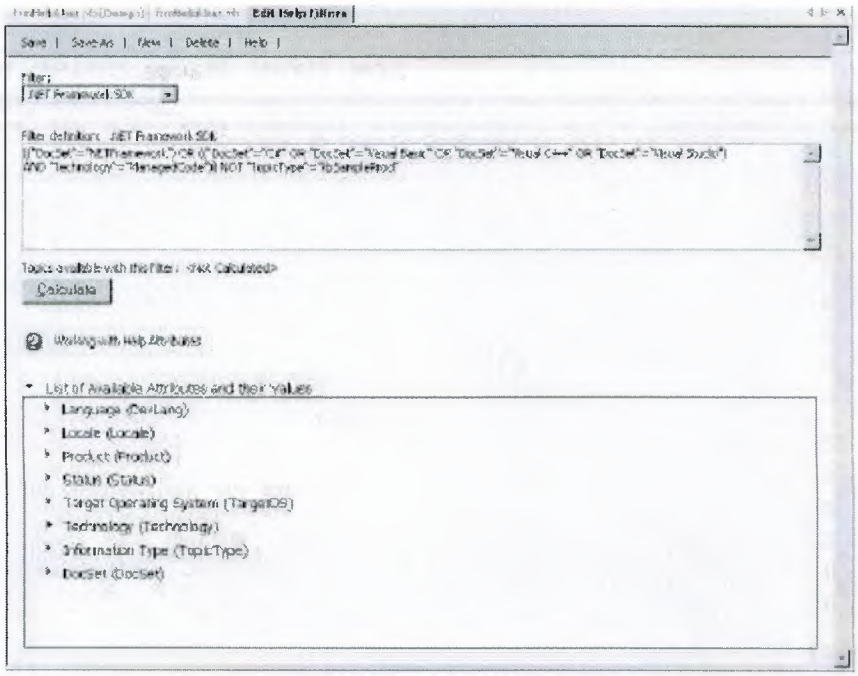

Figure.22

The filter is a logical statement that either includes or excludes documentation, based on some predefined types. The statement is in the upper text area; the document attributes and values are listed below. Although the filter equation can be edited directly, selecting attributes from the list automatically updates the statement. You can then save your changes to the current filter name, or use Save As to create a new one based on the existing equation.


MainMenu
ITEMS
CUSTOMERS
EMPLOYEES
ACCOUNT
REPORTS
ABOUT

Figure.23

Figure.24

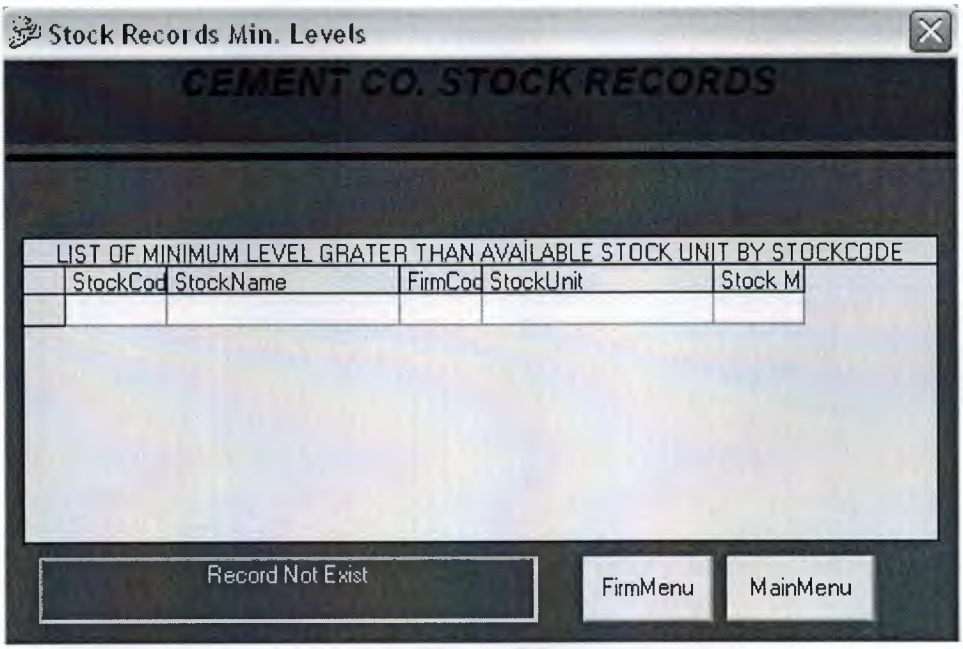

Figure.25


Customer Records
CEMENT CO. CUSTOMERS RECORDS
Customer Code :
3
Customer Name :
emine
Customer Phone :
0(636)373-73-77
Customer Tax No :
737-738-838-883-888
Customer City :
turkey
Customer Address :
manisa
Cus Reg Date :
23.12.2004
Selling
New
Save
Update
Delete
Find
MainMenu
01:42
21.10.2005

Figure.26

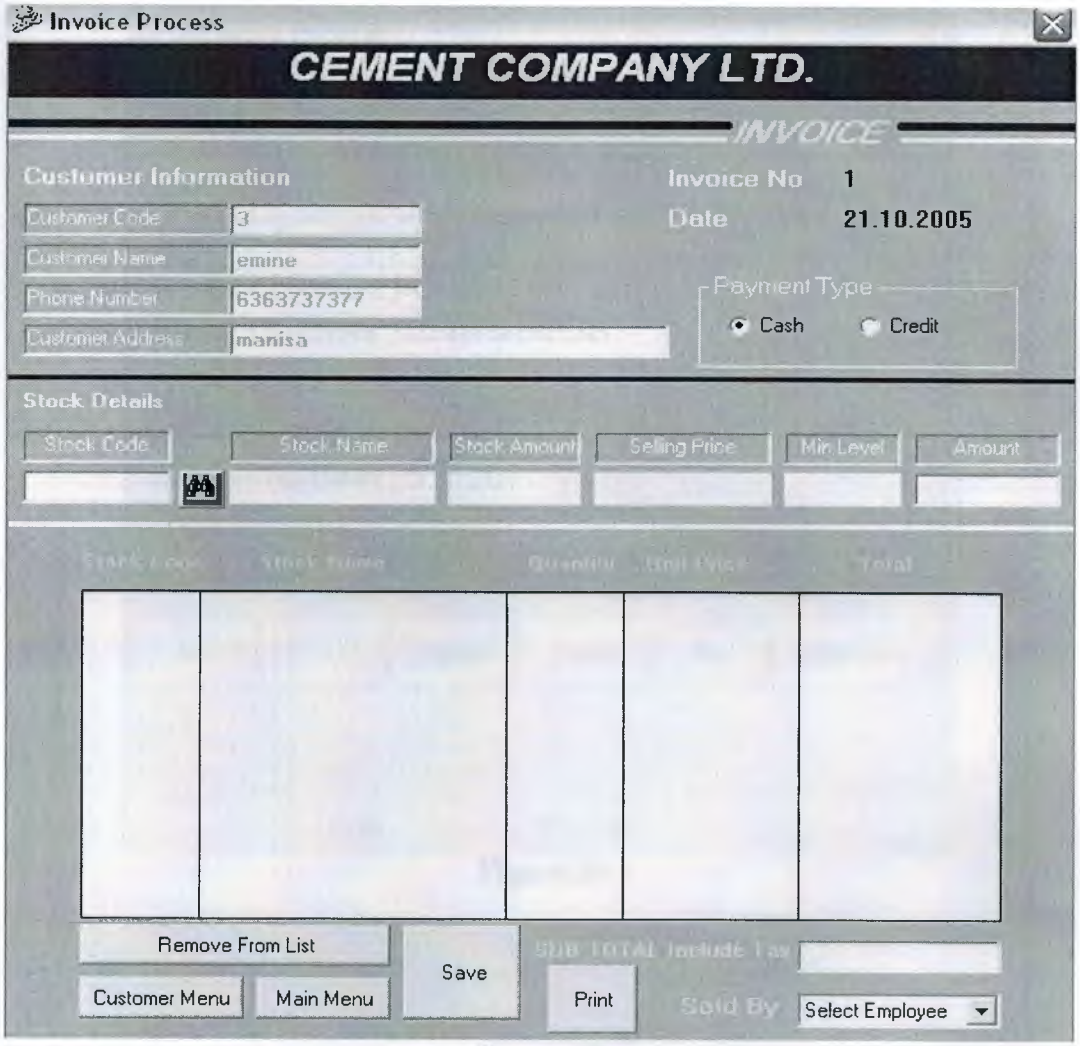

Figure.27

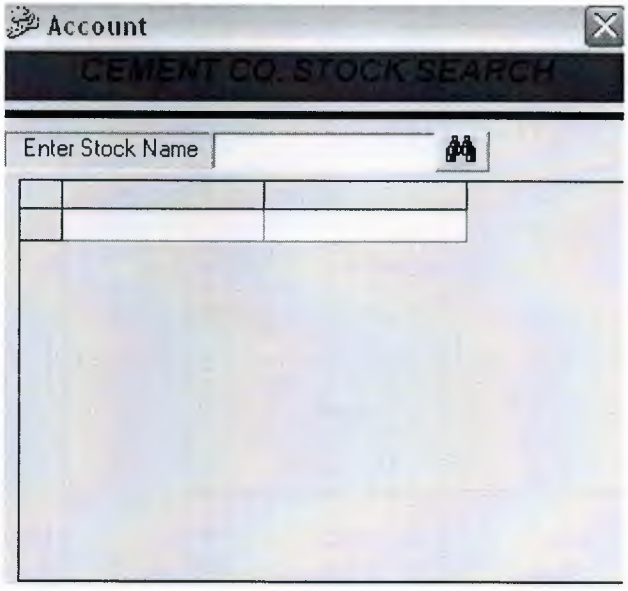

Figure.28


Employee Records
CEMENT CO. EMPLOYEES RECORDS
Employee Code :
1
Employee Name :
ahmet
Employee Phone :
0(333)333-33-33
Employee Security No
444-444-444-444-444
Employee City :
belçika
Employee Address :
belçika
Empployee Reg Date :
21.10.2005
Employee Salary
627288282
New
Save
Update
Delete
Find
MainMenu
01:49
21.10.2005

Figure.29

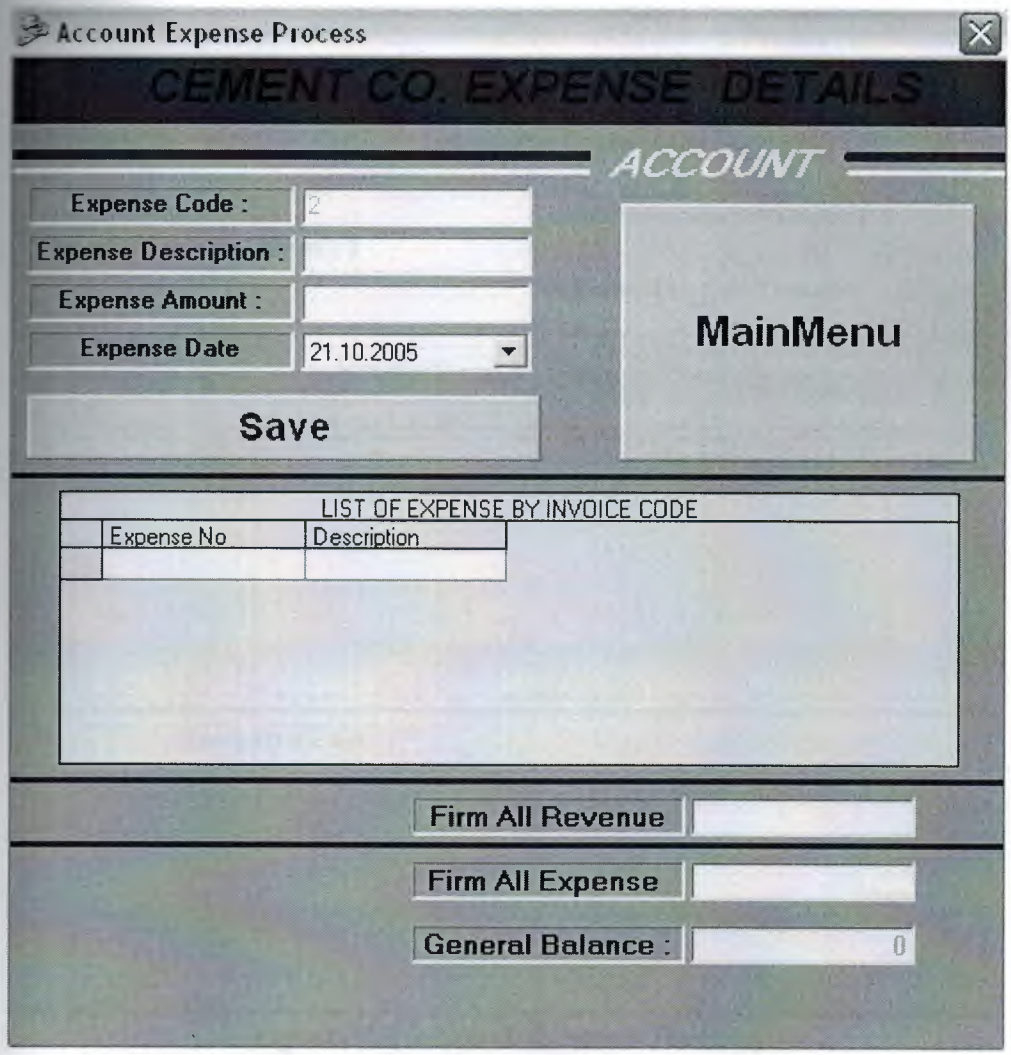
Account Expense Process
CEMENT CO. EXPENSE DETAILS
ACCOUNT
Expense Code :
2
Expense Description :
Expense Amount :
Expense Date
21.10.2005
Save
MainMenu
LIST OF EXPENSE BY INVOICE CODE
Expense No
Description
Firm All Revenue
Firm All Expense
General Balance :
0

Figure.30

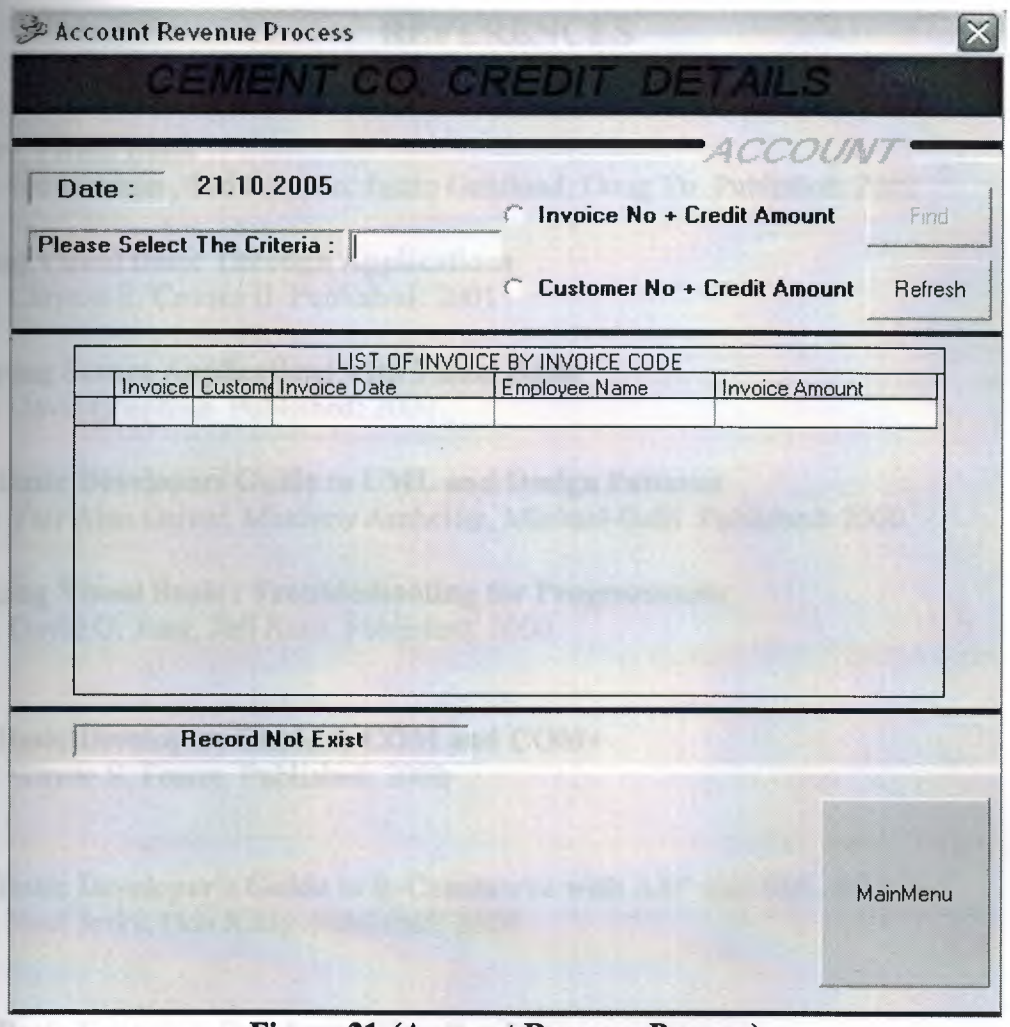

Figure.31-(Account Revenue Process)

REFERENCES

Effective Visual Basic
Author: Joe Hummel, Ted Pattison, Justin Gehtland, Doug Tu Published: 2001

Learning Visual Basic Through Applications
Author: Clayton E. Crooks II Published: 2001

Developing Secure Applications with Visual Basic
Author: Davis Chapman Published: 2000

Visual Basic Developers Guide to UML and Design Patterns
Author: Yair Alan Griver, Matthew Arnheiter, Michael Gelli Published: 2000

Debugging Visual Basic : Troubleshooting for Programmers
Author: David G. Jung, Jeff Kent Published: 2000

Visual Basic Developers Guide to COM and COM+
Author: Wayne S. Freeze Published: 2000

Visual Basic Developer`s Guide to E-Commerce with ASP and SQL Server
Author: Noel Jerke, Don Kiely Published: 2000

Visual Basic Language Developers Handbook
Author: Ken Getz, Mike Gilbert Published: 2000

Real Visual Basic: A Practical Approach to Enterprise Development in the Corporate World
Author: Dan Petit Published: 2000

Professional Windows DNA: Building Distributed Web Applications with VB, COM+, MSMQ, SOAP, and ASP
Author: Christopher Blexrud, et al. Published: 2000

Developing Applications with Visual Basic and UML
Author: Paul R. Reed, Jr. Published: 2000

Professional Visual Basic 6 XML
Author: James G. Britt Published: 2000

Serious ADO: Universal Data Access with Visual Basic
Author: Robert Macdonald Published: 2000

APPENDIX A

Main menu (code)

```
Private Sub mnucr_Click()
form1.Enabled = False
Form5.Show
End Sub
Private Sub mnucrp_Click()
DataReport3.Show
End Sub
Private Sub mnue_Click()
Unload Me
End Sub
Private Sub mnuemp_Click()
form1.Enabled = False
Form6.Show
End Sub
Private Sub mnuer_Click()
DataReport4.Show
End Sub
Private Sub mnuexp_Click()
Form9.Show
End Sub
Private Sub mnufr1_Click()
DataReport1.Show
End Sub
Private Sub mnufrr_Click()
```

```
form1.Enabled = False
Form2.Show
End Sub
Private Sub mnuh_Click()
Form12.Show
End Sub
Private Sub mnuid_Click()
frmcus.Show
End Sub
Private Sub mnuir_Click()
DataReport5.Show
End Sub
Private Sub mnupr_Click()
Form3.Show
End Sub
Private Sub mnurev_Click()
Form8.Show
End Sub
Private Sub mnusp_Click()
DataReport2.Show
End Sub
Private Sub mnuv_Click()
Form11.Show
End Sub
```

```
Private Sub Timer1_Timer()
ilkharf = Left(Label1.Caption, 1)
   yazi = Right(Label1.Caption, Len(Label1.Caption) - 1)
   Label1.Caption = yazi + ilkharf
   Dim sString As String
      sString = "This Program Created By Mahmut Savranoğlu..."
   If Timer1.Tag = 0 Then
      Me.Caption = sString
      Timer1.Tag = 1
   ElseIf Timer1.Tag < Len(sString) Then
      Me.Caption = Right(sString, Len(sString) - Timer1.Tag)
      Timer1.Tag = Timer1.Tag + 1
   ElseIf Timer1.Tag = Len(sString) Then
      Me.Caption = sString
      Timer1.Tag = 0
   End If
End Sub
```

Firm Records (code)

```
Private Sub Command1_Click()
Command7.Visible = False
clear
coun
Command2.Enabled = True
Command3.Enabled = False
Command4.Enabled = False
```

```
Text2.SetFocus

End Sub

Private Sub Command10_Click()

Form12.Show

End Sub

Private Sub Command2_Click()

Dim sql, ask, ask1

ask = MsgBox("Do You Want To Save That ?", vbInformation + vbYesNo, "Save")

If ask = vbYes Then

If Text2.text <> "" And MaskEdBox1.text <> "" And MaskEdBox2.text <> "" And MaskEdBox3.text <> "" And Text3.text <> "" And Text4.text <> "" Then

sql = "insert into firms(firmname, firmcode, firmphone, firmfax, firmtaxno, firmcity, firmadres, firmregdate) values("

sql = sql & "'" & Text2.text & "',"

sql = sql & "" & Text1.text & ","

sql = sql & "'" & MaskEdBox1.text & "',"

sql = sql & "'" & MaskEdBox2.text & "',"

sql = sql & "'" & MaskEdBox3.text & "',"

sql = sql & "'" & Text3 & "',"

sql = sql & "'" & Text4 & "',"

sql = sql & "'" & DTPicker1.Value & "')"

database.Execute (sql)

Dim i As Integer

ProgressBar1.Min = 0

ProgressBar1.Max = 1000

For i = ProgressBar1.Min To ProgressBar1.Max

ProgressBar1.Visible = True
```

```
ProgressBar1.Value = i
Next
ProgressBar1.Visible = False
ask1 = MsgBox("Firm Information Save Successful! ", , "Saved")
Command2.Enabled = False
Command7.Visible = True
Command3.Enabled = True
Command4.Enabled = True
Command6.Enabled = True
Else
ask1 = MsgBox("Please Fill The Other Texts!")
Command7.Visible = False
Command3.Enabled = False
Command4.Enabled = False
Command6.Enabled = True
Text2.SetFocus
End If
End If
End Sub
Private Sub Command3_Click()
Dim ask As String
If Text1.text <> "" And Text2.text <> "" And MaskEdBox1.text <> "" Then
ask = MsgBox("Do You Want To Update Firm Information?", vbCritical + vbYesNo,
"Update")
If ask = vbYes Then
conn
```

```
ersoy = "update firms set firmname='" & Text2.text & "', firmphone='" & MaskEdBox1.text
& "', firmfax='" & MaskEdBox2.text & "', firmtaxno='" & MaskEdBox3.text & "', firmcity='"
& Text3.text & "', firmadres='" & Text4.text & "', firmregdate='" & DTPicker1.Value & "'
where firmcode=" & Text1.text & " "

database.Execute (ersoy)

MsgBox ("Firm Information Updated!")

End If

Else

MsgBox ("Please Find Any Firm!")

End If

Command7.Visible = False

Command3.Enabled = False

Command4.Enabled = False

clear

End Sub

Private Sub Command4_Click()

Dim ask

If Text1.text <> "" Then

ask = MsgBox("Do You Want To Delete This Firm Detail?", vbExclamation + vbYesNo,
"Delete")

If ask = vbYes Then

conn

ersoy = "delete * from firms where firmcode=" & Text1.text & ""

database.Execute (ersoy)

ersoy = "delete * from stocks where firmcode=" & Text1.text & ""

database.Execute (ersoy)

MsgBox ("Firm Information Deleted!")

End If
```

```
Else
MsgBox ("Please Find Any Firm!")
End If
Command7.Visible = False
Command3.Enabled = False
Command4.Enabled = False
clear
coun
End Sub
Private Sub Command5_Click()
database.Close
Unload Me
form1.Show
form1.Enabled = True
End Sub
Private Sub Command6_Click()
Dim find As Integer
conn
find = Val(InputBox("Please Insert The Wanted Firm Code!"))
ersoy = "select * from firms where firmcode=" & find & ""
   Set ezgi = database.Execute(ersoy)
   If ezgi.EOF Then
   MsgBox ("The Wanted Firm is Not Available!")
   Else
   Dim i As Integer
ProgressBar1.Min = 0
```

```
ProgressBar1.Max = 1000
For i = ProgressBar1.Min To ProgressBar1.Max
ProgressBar1.Visible = True
ProgressBar1.Value = i
Next
ProgressBar1.Visible = False
   Text1.text = ezgi![firmcode]
   Text2.text = ezgi![firmname]
   MaskEdBox1.text = ezgi![firmphone]
   MaskEdBox2.text = ezgi![firmfax]
   MaskEdBox3.text = ezgi![firmtaxno]
   Text3.text = ezgi![firmcity]
   Text4.text = ezgi![firmadres]
   DTPicker1.Value = ezgi![firmregdate]
   Text2.SetFocus
   Command2.Enabled = False
   Command3.Enabled = True
   Command4.Enabled = True
   Command7.Visible = True
   End If
   ezgi.Close
End Sub
Private Sub Command7_Click()
Form4.Text1.text = Text1.text
Form4.Text2.text = Text2.text
Form4.Show
```

```
Unload Form2
End Sub
Private Sub Form_Load()
ProgressBar1.Align = vbAlignBottom
ProgressBar1.Visible = False
coun
With StatusBar1.Panels
  Set p = .Add(, , , sbrTime)
  Set p = .Add(, , , sbrDate)
End With
DTPicker1.Value = Date
End Sub
Private Sub clear()
Text1.text = ""
Text2.text = ""
Text3.text = ""
Text4.text = ""
MaskEdBox1.Mask = ""
MaskEdBox1.text = ""
MaskEdBox1.Mask = "0(999)999-99-99"
MaskEdBox2.Mask = ""
MaskEdBox2.text = ""
MaskEdBox2.Mask = "0(999)999-99-99"
MaskEdBox3.Mask = ""
MaskEdBox3.text = ""
MaskEdBox3.Mask = "999-999-999-999-999"
```

```
DTPicker1.Value = Date
End Sub
Private Sub coun()
Dim Count, Count1
conn
Set ezgi = New ADODB.Recordset
Count = "select * from Firms"
Set ezgi = database.Execute(Count)
   If ezgi.EOF Then
    Command6.Enabled = False
    Text1.text = 1
   Else
   Count1 = "select max(firmcode) as cis from firms"
   Set ezgi = database.Execute(Count1)
   Text1.text = ezgi![cis] + 1
   End If
 ezgi.Close
End Sub
Public Sub conn()
Set database = New ADODB.Connection
   database.CursorLocation = adUseServer
   ersoy = "provider=Microsoft.jet.oledb.3.51; Data Source=" & App.Path & "\ezgi.mdb"
   database.Open ersoy
End Sub
Private Sub Form_Unload(Cancel As Integer)
Unload Me
```

```
form1.Enabled = True

End Sub
```

Esc-Exit (code)

```
Private Declare Function DrawText Lib "user32" Alias "DrawTextA" (ByVal hdc As Long, ByVal lpStr As String, ByVal nCount As Long, lpRect As RECT, ByVal wFormat As Long) As Long

Private Declare Function GetTickCount Lib "kernel32" () As Long

Const DT_CENTER As Long = &H1

Const DT_LEFT As Long = &H0

Const DT_RIGHT As Long = &H2

Private Type RECT

    Left As Long

    Top As Long

    Right As Long

    Bottom As Long

End Type

Dim kare As RECT

Const yazı As String = "This Project Created By Mahmut Savranoğlu..." & vbCrLf & _

            vbCrLf & vbCrLf & _

            "COM 430" & vbCrLf & _

            vbCrLf & vbCrLf & _

            "Graduation Project" & _

            vbCrLf & vbCrLf & _

            "Cement Selling Package Program" & _

            vbCrLf & vbCrLf & _

            ">>>> NEAR EAST UNIVERSITY <<<<"

Private Sub Form_Load()
```

```
kare.Left = 0
kare.Top = pano.ScaleHeight
kare.Right = pano.ScaleWidth
kare.Bottom = pano.ScaleHeight
End Sub
Private Sub Timer1_Timer()
kare.Top = kare.Top - 1
kare.Bottom = kare.Bottom
If kare.Top = -170 Then
  kare.Top = pano.ScaleHeight
End If
pano.Cls
DrawText pano.hdc, yazı, -1, kare, DT_CENTER
pano.Refresh
End Sub
Private Sub Form_KeyPress(KeyAscii As Integer)
  Unload Me
End Sub
Private Sub Form_Unload(Cancel As Integer)
  Unload Me
  form1.Enabled = True
  form1.Show
End Sub
Private Sub Frame1_Click()
  Unload Me
End Sub
```

```
Private Sub Form_Activate()
form1.Enabled = False
End Sub
```

Stock Records (code)

```
 Private database As ADODB.Connection
Private ezgi As ADODB.Recordset
Private ersoy As String
Dim p As Panel
Dim ah As Boolean
Dim a, b, d, e, g, h, f, l  As Double
Dim m, n, decrip, sql1
Private Sub Command1_Click()
clear
coun
Command2.Enabled = True
Command3.Enabled = False
Command4.Enabled = False
Text4.SetFocus
End Sub
Private Sub Command2_Click()
care
If ah = True Then
Dim sql, sql1, ask, ask1
ask = MsgBox("Do You Want To Save That ?", vbInformation + vbYesNo, "Save")
If ask = vbYes Then
```

```
If Text4.text <> "" And Text5.text <> "" And Text6.text <> "" And Text8.text <> "" And Text10.text <> "" And Text11.text <> "" Then

sql = "insert into stocks(firmname, firmcode, stockcode, stockname, stockminl, stockbd, stockunit, stockbp, stockpperc, stocksellp) values("

sql = sql & "'" & Text2.text & "',"

sql = sql & "" & Text1.text & ","

sql = sql & "" & Text3.text & ","

sql = sql & "'" & Text4.text & "',"

sql = sql & "" & Text11 & ","

sql = sql & "'" & Label11.Caption & "',"

sql = sql & "" & a & ","

sql = sql & "" & b & ","

sql = sql & "" & h & ","

sql = sql & "'" & Text9 & "')"

database.Execute (sql)

m = Text5.text

l = Text6.text

n = m * l

decrip = Text3 + "," + "no" + "st"

sql1 = "insert into account(accdate, expense, revenue, description, expcode) values("

sql1 = sql1 & "'" & Label11.Caption & "',"

sql1 = sql1 & "" & n & ","

sql1 = sql1 & "" & 0 & ","

sql1 = sql1 & "'" & decrip & "',"

sql1 = sql1 & "" & Text3.text & ")"

database.Execute (sql1)

Dim i As Integer
```

```
ProgressBar1.Min = 0

ProgressBar1.Max = 1000

For i = ProgressBar1.Min To ProgressBar1.Max

ProgressBar1.Visible = True

ProgressBar1.Value = i

Next

ProgressBar1.Visible = False

ask1 = MsgBox("stock Information Save Successful! ", , "Saved")

Command2.Enabled = False

Command3.Enabled = True

Command4.Enabled = True

Else

ask1 = MsgBox("Please Fill The Other Texts!")

Command3.Enabled = False

Command4.Enabled = False

Text4.SetFocus

End If

End If

End If

End Sub

Private Sub Command3_Click()

Frame1.Visible = True

Command3.Enabled = False

Text4.Enabled = False

Text5.Enabled = False

Text6.Enabled = False
```

```
Text8.Enabled = False

Text10.Enabled = False

Text11.Enabled = False

End Sub

Private Sub Command4_Click()

Dim ask

If Text3.text <> "" Then

ask = MsgBox("Do You Want To Delete This Stock Detail?", vbExclamation + vbYesNo,
"Delete")

If ask = vbYes Then

conn

ersoy = "delete * from stocks where stockcode=" & Text3.text & ""

database.Execute (ersoy)

MsgBox ("Stock Information Deleted!")

End If

Else

MsgBox ("Please Find Any Stock!")

End If

Command3.Enabled = False

Command4.Enabled = False

clear

coun

End Sub

Private Sub Command5_Click()

database.Close

Unload Me
```

```
Form2.Show
End Sub
Private Sub Command6_Click()
Dim find As Integer
conn
find = Val(InputBox("Please Insert The Wanted Stock Code!"))
ersoy = "select * from stocks where stockcode=" & find & ""
   Set ezgi = database.Execute(ersoy)
   If ezgi.EOF Then
   MsgBox ("The Wanted Stock is Not Available!")
   Else
   Dim i As Integer
ProgressBar1.Min = 0
ProgressBar1.Max = 1000
For i = ProgressBar1.Min To ProgressBar1.Max
ProgressBar1.Visible = True
ProgressBar1.Value = i
Next
ProgressBar1.Visible = False
   Text1.text = ezgi![firmcode]
   Text2.text = ezgi![firmname]
   Text3.text = ezgi![stockcode]
   Text4.text = ezgi![stockname]
   Text5.text = ezgi![stockunit]
   Text6.text = ezgi![stockbp]
   Text8.text = ezgi![stockpperc]
```

```
    Text9.text = ezgi![stocksellp]
    Label11.Caption = ezgi![stockbd]
    Text11.text = ezgi![stockminl]
    Command2.Enabled = False
    Command3.Enabled = True
    Command4.Enabled = True
    End If
    ezgi.Close
End Sub
Private Sub Command7_Click()
database.Close
Unload Me
Unload Form2
form1.Show
form1.Enabled = True
End Sub
Private Sub Command8_Click()
Dim ask As String
If Text12.text <> "" And Text13.text <> "" And Text14.text <> "" And Text15.text <> "" And
Text16.text <> "" Then
ask = MsgBox("Do You Want To Update Stock Information?", vbCritical + vbYesNo,
"Update")
If ask = vbYes Then
conn
ersoy = "update stocks set stockunit=" & Text5.text & ", stockbp=" & Text6.text & ",
stockpperc=" & Text8.text & ", stocksellp='" & Text9.text & "', stockminl=" & Text11.text &
"  where stockcode=" & Text3.text & " "
database.Execute (ersoy)
```

```
m = Text12.text

l = Text13.text

n = m * l

decrip = Text3 + "," + "no" + "st"

sql1 = "insert into account(accdate, expense, revenue, description, expcode) values("

sql1 = sql1 & "'" & Label11.Caption & "',"

sql1 = sql1 & "" & n & ","

sql1 = sql1 & "" & 0 & ","

sql1 = sql1 & "'" & decrip & "',"

sql1 = sql1 & "" & Text3.text & ")"

database.Execute (sql1)

Dim i As Integer

ProgressBar1.Min = 0

ProgressBar1.Max = 1000

For i = ProgressBar1.Min To ProgressBar1.Max

ProgressBar1.Visible = True

ProgressBar1.Value = i

Next

ProgressBar1.Visible = False

MsgBox ("Stock Information Updated!")

End If

Else

MsgBox ("Please Enter The New Stock Information!")

Command3.Enabled = False

End If

Text12.text = ""
```

```
Text13.text = ""
Text14.text = ""
Text15.text = ""
Text16.text = ""
Command3.Enabled = True
Command4.Enabled = False
Command6.Enabled = True
Frame1.Visible = False
Text4.Enabled = True
Text5.Enabled = True
Text6.Enabled = True
Text8.Enabled = True
Text10.Enabled = True
Text11.Enabled = True
End Sub
Private Sub Command9_Click()
Frame1.Visible = False
Text12.text = ""
Text13.text = ""
Text14.text = ""
Text15.text = ""
Text16.text = ""
clear
End Sub
Private Sub Form_Load()
ProgressBar1.Align = vbAlignBottom
```

```
ProgressBar1.Visible = False
coun
With StatusBar1.Panels
   Set p = .Add(, , , sbrTime)
   Set p = .Add(, , , sbrDate)
End With
Label11.Caption = Date
End Sub
Private Sub Form_Unload(Cancel As Integer)
Unload Me
Form2.Show
End Sub
Private Sub Text10_Change()
On Error Resume Next
a = Text5.text
b = Text6.text
h = Text8.text
d = Val(a) * Val(b)
e = ((d * Val(h)) / 100) + d
f = e / a
g = ((f * Val(Text10.text)) / 100) + f
Text9.text = g
End Sub
Private Sub coun()
Dim Count, Count1
conn
```

```
Set ezgi = New ADODB.Recordset
Count = "select * from stocks"
Set ezgi = database.Execute(Count)
  If ezgi.EOF Then
   Command6.Enabled = False
   Text3.text = 1
  Else
  Count1 = "select max(stockcode) as cis from stocks"
  Set ezgi = database.Execute(Count1)
  Text3.text = ezgi![cis] + 1
  End If
  ezgi.Close
End Sub
Public Sub conn()
Set database = New ADODB.Connection
  database.CursorLocation = adUseClient
  ersoy = "provider=Microsoft.jet.oledb.3.51; Data Source=" & App.Path & "\ezgi.mdb"
  database.Open ersoy
End Sub
Private Sub clear()
Text4.text = ""
Text5.text = ""
Text6.text = ""
Text8.text = ""
Text9.text = ""
Text10.text = ""
```

```
Text11.text = ""
End Sub
Private Sub Text10_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
   KeyAscii = 0
   SendKeys "{Tab}"
ElseIf InStr(("1234567890" & vbBack & ""), Chr(KeyAscii)) = 0 Then
   KeyAscii = 0
End If
End Sub
Private Sub Text12_LostFocus()
a = Val(Text12.text)
b = a + Val(Text5.text)
Text5.text = b
End Sub
Private Sub Text13_LostFocus()
a = Val(Text13.text)
Text6.text = a
End Sub
Private Sub Text14_LostFocus()
a = Val(Text14.text)
Text8.text = a
End Sub
Private Sub Text15_LostFocus()
a = Val(Text15.text)
Text10.text = a
```

```
End Sub
Private Sub Text16_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
a = Val(Text14.text)
Text8.text = a
End If
End Sub
Private Sub Text16_LostFocus()
If Val(Text5.text) <= Val(Text16.text) Then
MsgBox ("Please Enter Amount Smaller Than Unit!")
Text16.SetFocus
Else
a = Val(Text16.text)
Text11.text = a
End If
End Sub

Private Sub Text5_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
   KeyAscii = 0
   SendKeys "{Tab}"
ElseIf InStr(("1234567890" & vbBack & ""), Chr(KeyAscii)) = 0 Then
   KeyAscii = 0
End If
End Sub
Private Sub Text6_KeyPress(KeyAscii As Integer)
```

```
If KeyAscii = 13 Then
   KeyAscii = 0
   SendKeys "{Tab}"
ElseIf InStr(("1234567890" & vbBack & ""), Chr(KeyAscii)) = 0 Then
   KeyAscii = 0
End If
End Sub
Private Sub Text7_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
   KeyAscii = 0
   SendKeys "{Tab}"
ElseIf InStr(("1234567890" & vbBack & ""), Chr(KeyAscii)) = 0 Then
   KeyAscii = 0
End If
End Sub
Private Sub Text8_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
   KeyAscii = 0
   SendKeys "{Tab}"
ElseIf InStr(("1234567890" & vbBack & ""), Chr(KeyAscii)) = 0 Then
   KeyAscii = 0
End If
End Sub
Private Sub care()
If Len(Text4.text) = 0 Then
MsgBox ("Please Fill The Texts!")
```

```
Text4.SetFocus
Else
If Val(Text5.text) <= Val(Text11.text) Then
ah = False
MsgBox ("Please Enter Amount Smaller Than Unit!")
Text11.SetFocus
Else
ah = True
End If
End If
End Sub
```

Customer Records (code)

```
Private Sub Command1_Click()
clear
coun
Command2.Enabled = True
Command3.Enabled = False
Command4.Enabled = False
Command7.Visible = False
Text2.SetFocus
End Sub
Private Sub Command2_Click()
Dim sql, ask, ask1
ask = MsgBox("Do You Want To Save That ?", vbInformation + vbYesNo, "Save")
If ask = vbYes Then
```

```
If Text2.text <> "" And MaskEdBox1.text <> "" And Text3.text <> "" And MaskEdBox3.text
<> "" And Text3.text <> "" And Text4.text <> "" Then

sql = "insert into cus(cusname, cuscode, cusphone, custaxno, cuscity, cusadres, cusrd)
values("

sql = sql & "'" & Text2.text & "',"

sql = sql & "" & Text1.text & ","

sql = sql & "'" & MaskEdBox1.text & "',"

sql = sql & "'" & MaskEdBox3.text & "',"

sql = sql & "'" & Text3 & "',"

sql = sql & "'" & Text4 & "',"

sql = sql & "'" & DTPicker1.Value & "')"

database.Execute (sql)

Dim i As Integer

ProgressBar1.Min = 0

ProgressBar1.Max = 1000

For i = ProgressBar1.Min To ProgressBar1.Max

ProgressBar1.Visible = True

ProgressBar1.Value = i

Next

ProgressBar1.Visible = False

ask1 = MsgBox("Customer Information Save Successful! ", , "Saved")

Command2.Enabled = False

Command3.Enabled = True

Command4.Enabled = True

Command7.Visible = True

Command6.Enabled = True

Else
```

```
ask1 = MsgBox("Please Fill The Other Texts!", vbCritical, "Customer")

Command6.Enabled = True

Command3.Enabled = False

Command4.Enabled = False

Text2.SetFocus

End If

End If

End Sub

Private Sub Command3_Click()

Dim ask As String

If Text1.text <> "" And Text2.text <> "" And MaskEdBox1.text <> "" Then

ask = MsgBox("Do You Want To Update Customer Information?", vbCritical + vbYesNo,
"Update")

If ask = vbYes Then

conn

ersoy = "update cus set cusname='" & Text2.text & "', cusphone='" & MaskEdBox1.text & "',
custaxno='" & MaskEdBox3.text & "', cuscity='" & Text3.text & "', cusadres='" & Text4.text
& "', cusrd='" & DTPicker1.Value & "' where cuscode=" & Text1.text & " "

database.Execute (ersoy)

MsgBox ("Customer Information Updated!")

End If

Else

MsgBox ("Please Find Any Customer!")

End If

Command3.Enabled = False

Command4.Enabled = False

clear
```

```
End Sub

Private Sub Command4_Click()

Dim ask

If Text1.text <> "" Then

ask = MsgBox("Do You Want To Delete This Customer Detail?", vbExclamation + vbYesNo,
"Delete")

If ask = vbYes Then

conn

ersoy = "delete * from cus where cuscode=" & Text1.text & ""

database.Execute (ersoy)

MsgBox ("Customer Information Deleted!")

End If

Else

MsgBox ("Please Find Any Customer!")

End If

Command3.Enabled = False

Command4.Enabled = False

clear

coun

End Sub

Private Sub Command5_Click()

Unload Me

form1.Show

form1.Enabled = True

End Sub

Private Sub Command6_Click()
```

```
Dim find As Integer
conn
find = Val(InputBox("Please Insert The Wanted Customer Code!"))
ersoy = "select * from cus where cuscode=" & find & ""
  Set ezgi = database.Execute(ersoy)
  If ezgi.EOF Then
  MsgBox ("The Wanted Customer is Not Available!")
  Else
  Dim i As Integer
ProgressBar1.Min = 0
ProgressBar1.Max = 1000
For i = ProgressBar1.Min To ProgressBar1.Max
ProgressBar1.Visible = True
ProgressBar1.Value = i
Next
ProgressBar1.Visible = False
  Text1.text = ezgi![cuscode]
  Text2.text = ezgi![cusname]
  MaskEdBox1.text = ezgi![cusphone]
  MaskEdBox3.text = ezgi![custaxno]
  Text3.text = ezgi![cuscity]
  Text4.text = ezgi![cusadres]
  DTPicker1.Value = ezgi![cusrd]
  Text2.SetFocus
  Command7.Visible = True
  Command2.Enabled = False
```

```
    Command3.Enabled = True
    Command4.Enabled = True
    End If
    ezgi.Close
End Sub
Private Sub Command7_Click()
Form7.Text4.text = Text1.text
Form7.Text5.text = Text2.text
Form7.Text7.text = MaskEdBox1.text
Form7.Text6.text = Text4.text
Form7.Show
End Sub
Private Sub Form_Load()
ProgressBar1.Align = vbAlignBottom
ProgressBar1.Visible = False
coun
With StatusBar1.Panels
    Set p = .Add(, , , sbrTime)
    Set p = .Add(, , , sbrDate)
End With
DTPicker1.Value = Date
End Sub
Private Sub clear()
Text1.text = ""
Text2.text = ""
Text3.text = ""
```

```
Text4.text = ""
MaskEdBox1.Mask = ""
MaskEdBox1.text = ""
MaskEdBox1.Mask = "0(999)999-99-99"
MaskEdBox3.Mask = ""
MaskEdBox3.text = ""
MaskEdBox3.Mask = "999-999-999-999-999"
DTPicker1.Value = Date
End Sub
Private Sub coun()
Dim Count, Count1
conn
Set ezgi = New ADODB.Recordset
Count = "select * from cus"
Set ezgi = database.Execute(Count)
  If ezgi.EOF Then
   Command6.Enabled = False
   Text1.text = 1
  Else
  Count1 = "select max(cuscode) as cis from cus"
  Set ezgi = database.Execute(Count1)
  Text1.text = ezgi![cis] + 1
  End If
  ezgi.Close
End Sub
Public Sub conn()
```

```
Set database = New ADODB.Connection
  database.CursorLocation = adUseClient
  ersoy = "provider=Microsoft.jet.oledb.3.51; Data Source=" & App.Path & "\ezgi.mdb"
  database.Open ersoy
End Sub
Private Sub Form_Unload(Cancel As Integer)
Unload Me
form1.Show
form1.Enabled = True
End Sub
```

Employee Records (code)

```
Private Sub Command1_Click()
clear
coun
Command2.Enabled = True
Command3.Enabled = False
Command4.Enabled = False
Text2.SetFocus
End Sub
Private Sub Command2_Click()
Dim sql, ask, ask1
ask = MsgBox("Do You Want To Save That ?", vbInformation + vbYesNo, "Save")
If ask = vbYes Then
If Text2.text <> "" And MaskEdBox1.text <> "" And Text3.text <> "" And MaskEdBox3.text
<> "" And Text3.text <> "" And Text4.text <> "" Then
sql = "insert into emp(empname, empcode, empphone, empsc, empcity, empadres,empsal,
emprd) values("
```

```
sql = sql & "'" & Text2.text & "',"

sql = sql & "" & Text1.text & ","

sql = sql & "'" & MaskEdBox1.text & "',"

sql = sql & "'" & MaskEdBox3.text & "',"

sql = sql & "'" & Text3 & "',"

sql = sql & "'" & Text4 & "',"

sql = sql & "" & Text5.text & ","

sql = sql & "'" & DTPicker1.Value & "')"

database.Execute (sql)

Dim i As Integer

ProgressBar1.Min = 0

ProgressBar1.Max = 1000

For i = ProgressBar1.Min To ProgressBar1.Max

ProgressBar1.Visible = True

ProgressBar1.Value = i

Next

ProgressBar1.Visible = False

ask1 = MsgBox("Employee Information Save Successful! ", , "Saved")

Command2.Enabled = False

Command3.Enabled = True

Command4.Enabled = True

Command6.Enabled = True

Else

ask1 = MsgBox("Please Fill The Other Texts!")

Command3.Enabled = False

Command4.Enabled = False
```

```
Command6.Enabled = True

Text2.SetFocus

End If

End If

End Sub

Private Sub Command3_Click()

Dim ask As String

If Text1.text <> "" And Text2.text <> "" And MaskEdBox1.text <> "" Then

ask = MsgBox("Do You Want To Update Employee Information?", vbCritical + vbYesNo,
"Update")

If ask = vbYes Then

conn

ersoy = "update emp set empname='" & Text2.text & "', empphone='" & MaskEdBox1.text &
"', empsc='" & MaskEdBox3.text & "', empcity='" & Text3.text & "', empadres='" &
Text4.text & "',empsal=" & Text5.text & ", emprd='" & DTPicker1.Value & "' where
empcode=" & Text1.text & " "

database.Execute (ersoy)

MsgBox ("Employee Information Updated!")

End If

Else

MsgBox ("Please Find Any Employee!")

End If

Command3.Enabled = False

Command4.Enabled = False

clear

End Sub

Private Sub Command4_Click()

Dim ask
```

```
If Text1.text <> "" Then

ask = MsgBox("Do You Want To Delete This Employee Detail?", vbExclamation + vbYesNo, "Delete")

If ask = vbYes Then

conn

ersoy = "delete * from emp where empcode=" & Text1.text & ""

database.Execute (ersoy)

MsgBox ("Employee Information Deleted!")

End If

Else

MsgBox ("Please Find Any Employee!")

End If

Command3.Enabled = False

Command4.Enabled = False

clear

coun

End Sub

Private Sub Command5_Click()

database.Close

Unload Me

form1.Show

form1.Enabled = True

End Sub

Private Sub Command6_Click()

Dim find As Integer

conn
```

```
find = Val(InputBox("Please Insert The Wanted Employee Code!"))
ersoy = "select * from emp where empcode=" & find & ""
  Set ezgi = database.Execute(ersoy)
  If ezgi.EOF Then
  MsgBox ("The Wanted Employee is Not Available!")
  Else
  Dim i As Integer
ProgressBar1.Min = 0
ProgressBar1.Max = 1000
For i = ProgressBar1.Min To ProgressBar1.Max
ProgressBar1.Visible = True
ProgressBar1.Value = i
Next
ProgressBar1.Visible = False
  Text1.text = ezgi![empcode]
  Text2.text = ezgi![empname]
  MaskEdBox1.text = ezgi![empphone]
  MaskEdBox3.text = ezgi![empsc]
  Text3.text = ezgi![empcity]
  Text4.text = ezgi![empadres]
  Text5.text = ezgi![empsal]
  DTPicker1.Value = ezgi![emprd]
  Text2.SetFocus
  Command2.Enabled = False
  Command3.Enabled = True
  Command4.Enabled = True
```

```
    End If
    ezgi.Close
End Sub
Private Sub Form_Load()
ProgressBar1.Align = vbAlignBottom
ProgressBar1.Visible = False
coun
With StatusBar1.Panels
 Set p = .Add(, , , sbrTime)
 Set p = .Add(, , , sbrDate)
End With
DTPicker1.Value = Date
End Sub
Private Sub clear()
Text1.text = ""
Text2.text = ""
Text3.text = ""
Text4.text = ""
Text5.text = ""
MaskEdBox1.Mask = ""
MaskEdBox1.text = ""
MaskEdBox1.Mask = "0(999)999-99-99"
MaskEdBox3.Mask = ""
MaskEdBox3.text = ""
MaskEdBox3.Mask = "999-999-999-999-999"
DTPicker1.Value = Date
```

```
End Sub
Private Sub coun()
Dim Count, Count1
conn
Set ezgi = New ADODB.Recordset
Count = "select * from emp"
Set ezgi = database.Execute(Count)
   If ezgi.EOF Then
    Command6.Enabled = False
    Text1.text = 1
   Else
   Count1 = "select max(empcode) as cis from emp"
   Set ezgi = database.Execute(Count1)
   Text1.text = ezgi![cis] + 1
   End If
   ezgi.Close
End Sub
Public Sub conn()
Set database = New ADODB.Connection
   database.CursorLocation = adUseClient
   ersoy = "provider=Microsoft.jet.oledb.3.51; Data Source=" & App.Path & "\ezgi.mdb"
   database.Open ersoy
End Sub
Private Sub Form_Unload(Cancel As Integer)
Unload Me
form1.Show
```

```
form1.Enabled = True
End Sub
Private Sub Text5_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
   KeyAscii = 0
   SendKeys "{Tab}"
ElseIf InStr(("1234567890" & vbBack & ""), Chr(KeyAscii)) = 0 Then
   KeyAscii = 0
End If
End Sub
```

Invoice Process (code)

```
Private Sub Command1_Click()
database.Close
Unload Me
Unload Form5
form1.Show
form1.Enabled = True
End Sub
Private Sub Command2_Click()
database.Close
Unload Me
Form5.Command7.Visible = False
Form5.Show
Form5.Enabled = True
End Sub
Private Sub Command3_Click()
```

```
Dim cevap, sec, i

Dim index As Integer

sec = List1(index).ListIndex

If sec < 0 Then

cevap = MsgBox("Select the removed item", vbCritical, "Remove Item")

Else

cevap = MsgBox("Are you sure to remove this item from the list ?", vbYesNo + vbQuestion,
"Remove item")

If cevap = vbYes Then

 For i = 0 To 4

 List1(i).RemoveItem List1(i).ListIndex

 Next

 Dim j, a, b

For j = 0 To List1(4).ListCount

a = Val(List1(4).List(j))

b = b + a

Next

Text2.text = b

End If

End If

End Sub

Private Sub coun()

Dim Count, Count1

conn

Set ezgi = New ADODB.Recordset

Count = "select incode from invoice2"
```

```
Set ezgi = database.Execute(Count)
  If ezgi.EOF Then
   Label9.Caption = 1
  Else
  Count1 = "select max(incode) as cis from invoice2"
  Set ezgi = database.Execute(Count1)
  Label9.Caption = ezgi![cis] + 1
  End If
  ezgi.Close
End Sub
Private Sub Command4_Click()
Dim k, i, ask, toplam, stk, stk2, sq, sq1, sq2, decrip
If DBCombo1.text = "Select Employee" Then
ask = MsgBox("Please select the employee name", vbInformation, "Invoice")
Else
ask = MsgBox("Do You Want to Save / Print The Invoice ?", vbYesNo + vbQuestion,
"Invoice Saving")
If ask = vbYes Then
conn
  For i = 0 To List1(0).ListCount - 1
  stk = Val(List1(0).List(i))
  stk2 = List1(1).List(i)
  sq = Val(List1(2).List(i))
  sq1 = Val(List1(3).List(i))
  sq2 = Val(List1(4).List(i))
sql1 = "insert into invoice1(incode, cuscode, stockcode, sname, quan, uprice, totp) values ("
```

```
sql1 = sql1 & "" & Label9 & ","

sql1 = sql1 & "" & Text4.text & ","

sql1 = sql1 & "" & stk & ","

sql1 = sql1 & "'" & stk2 & "',"

sql1 = sql1 & "" & sq & ","

sql1 = sql1 & "" & sq1 & ","

sql1 = sql1 & "" & sq2 & ")"

database.Execute (sql1)

sql2 = "update stocks set stockunit=stockunit-" & sq & " where stockcode=" & stk & ""

database.Execute (sql2)

  Next

If Option1 = True Then

toplam = 0

decrip = Label9 + "," + "no" + "Invoice"

sql3 = "insert into account(accdate, expense, revenue, description, expcode) values('" &
Label12 & "', " & 0 & ", " & Text2.text & " ,'" & decrip & "'," & toplam & "  )"

database.Execute (sql3)

End If

If Option2 = True Then

toplam = Text2.text

End If

sql2 = "insert into invoice2(incode, cuscode, subtot, invdate, empname) values (" & Label9 &
", " & Text4 & ", " & toplam & ", '" & Label12 & "', '" & DBCombo1.text & "')"

database.Execute (sql2)

ask = MsgBox("Invoice Printed/saved succesfully", vbInformation, "Invoice")

Command2_Click

End If
```

```
End If
End Sub
Private Sub Command5_Click()
Form10.Show
End Sub
Private Sub Command6_Click()
Dim ask As Integer
On Error Resume Next
ask = MsgBox("Are You Sure Print This Information ?", vbYesNo, "Account")
'CommonDialog1.Action = 5
If ask = vbYes Then iprint
End Sub
Private Sub Form_Activate()
On Error Resume Next
Text8.SetFocus
End Sub
Private Sub Form_Load()
Option1 = True
coun
Label12 = Date
End Sub
Private Sub Form_Unload(Cancel As Integer)
Unload Me
Form5.Show
Form5.Enabled = True
End Sub
```

```
Private Sub List1_Click(index As Integer)
Dim secind, topin, j
On Error Resume Next
secind = List1(index).ListIndex
topin = List1(index).TopIndex
 For j = 0 To 4
 List1(j).ListIndex = secind
 List1(j).TopIndex = topin
 Next
End Sub
Private Sub Text11_KeyPress(KeyAscii As Integer)
Dim ans
If KeyAscii = 13 Then
KeyAscii = 0
If Val(Text11.text) > Val(Text9.text) Then
ans = MsgBox("Stock not enough to sell this amount / Available stock is =" & Text9 & " unit", vbCritical, "Invoice")
Else
List1(0).AddItem Text8.text
List1(1).AddItem Text1.text
List1(2).AddItem Text11.text
List1(3).AddItem Text10.text
List1(4).AddItem (Val(Text10.text) * Val(Text11.text))
Text11.Enabled = False
clear
Text8.text = ""
```

```
Text8.SetFocus
Dim i, a, b
For i = 0 To List1(4).ListCount
a = Val(List1(4).List(i))
b = b + a
Next
Text2.text = b
End If
ElseIf InStr(("1234567890" & vbBack & ""), Chr(KeyAscii)) = 0 Then
  KeyAscii = 0
End If
End Sub
Private Sub Text8_KeyPress(KeyAscii As Integer)
Dim i, a, b
Dim ans
If KeyAscii = 13 Then
  If Len(Text8.text) = 0 Then
  ans = MsgBox("Please enter the stock code", vbCritical, "Invoice")
  Else
  KeyAscii = 0
  b = Text8.text
  For i = 0 To List1(0).ListCount
    If (List1(0).List(i)) = b Then
    ans = MsgBox("You entered this stock before please enter another stock / for Re-enter
remove stock = " & Text8 & " from the list ", vbCritical, "Invoice")
    Text8.text = ""
```

```
    Text8.SetFocus
    Exit Sub
    End If
   Next
  conn
  sql = "select * from  stocks where stockcode=" + Text8.text + ""
Set ezgi = database.Execute(sql)
If ezgi.EOF Then
ans = MsgBox("Stock Code Not Found", vbCritical, "Search")
clear
Text8.SetFocus
Else
Text1.text = ezgi![stockname]
Text9.text = ezgi![stockunit]
Text10.text = ezgi![stocksellp]
Text3.text = ezgi![stockminl]
ezgi.Close
Text11.Enabled = True
Text11.text = ""
Text11.SetFocus
End If
End If
ElseIf InStr(("1234567890" & vbBack & ""), Chr(KeyAscii)) = 0 Then
  KeyAscii = 0
End If
End Sub
```

```
Private Sub conn()
Set database = New ADODB.Connection
   database.CursorLocation = adUseClient
   ersoy = "provider=Microsoft.jet.oledb.3.51; Data Source=" & App.Path & "\ezgi.mdb"
   database.Open ersoy
End Sub
Public Sub clear()
Text8.text = ""
Text1.text = ""
Text9.text = ""
Text10.text = ""
Text11.text = ""
Text3.text = ""
End Sub
Sub iprint()
   Dim X As Printer
   Dim y, x1, x2, i, artim, yb, ys, x6, x5, x3, x4, k, x7
   On Error GoTo ass
   Printer.ScaleMode = 6
   Printer.FontName = "Courier New Tr"
   Printer.FontSize = 10
   y = 10: x1 = 5: x2 = x1 + 50: artim = 7
   Printer.CurrentX = 1
   Printer.CurrentY = 2
   Printer.Print "Cement Co. Account Process Invoice" & "        " & Format(Date, "Short
Date")
```

```
Printer.Line (0, 9)-(Printer.ScaleWidth, 9)
If Not IsNull(Text4.text) Then
   y = y + artim
   Printer.CurrentX = x1
   Printer.CurrentY = y
   Printer.Print "Customer Code :"
   Printer.CurrentX = x2
   Printer.CurrentY = y
   Printer.Print Text4.text
End If
If Not IsNull(Text5.text) Then
   y = y + artim
   Printer.CurrentX = x1
   Printer.CurrentY = y
   Printer.Print "Customer Name :"
   Printer.CurrentX = x2
   Printer.CurrentY = y
   Printer.Print Text5.text
End If
If Not IsNull(Text7.text) Then
   y = y + artim
   Printer.CurrentX = x1
   Printer.CurrentY = y
   Printer.Print "Customer Phone :"
   Printer.CurrentX = x2
   Printer.CurrentY = y
```

```
    Printer.Print Text7.text
End If
If Not IsNull(Text6.text) Then
    y = y + artim
    Printer.CurrentX = x1
    Printer.CurrentY = y
    Printer.Print "Customer City :"
    Printer.CurrentX = x2
    Printer.CurrentY = y
    Printer.Print Text6.text
End If
If Not IsNull(Label9.Caption) Then
    y = y + artim
    Printer.CurrentX = x1
    Printer.CurrentY = y
    Printer.Print "Invoice No :"
    Printer.CurrentX = x2
    Printer.CurrentY = y
    Printer.Print Label9.Caption
End If
If Not IsNull(Label12.Caption) Then
    y = y + artim
    Printer.CurrentX = x1
    Printer.CurrentY = y
    Printer.Print "Invoice Date :"
    Printer.CurrentX = x2
```

```
    Printer.CurrentY = y
    Printer.Print Label12.Caption
End If
If Not IsNull(DBCombo1.text) Then
    y = y + artim
    Printer.CurrentX = x1
    Printer.CurrentY = y
    Printer.Print "Employee Name :"
    Printer.CurrentX = x2
    Printer.CurrentY = y
    Printer.Print DBCombo1.text
End If
If Not IsNull(Text2.text) Then
    y = y + artim
    Printer.CurrentX = x1
    Printer.CurrentY = y
    Printer.Print "Sub Total :"
    Printer.CurrentX = x2
    Printer.CurrentY = y
    Printer.Print Text2.text
End If
    Printer.Print ""
    Printer.Print ""
    Printer.Print ""
    Printer.Print ""
yb = Printer.CurrentY
```

```
x7 = Printer.TextWidth(Space(70))
x6 = x7 + Printer.TextWidth(Space(25))
x5 = x6 + Printer.TextWidth(Space(25))
x3 = x5 + Printer.TextWidth(Space(25))
x4 = x3 + Printer.TextWidth(Space(25))
Printer.Line (30, yb)-(x4 - 30, yb)
Printer.CurrentX = 30
Printer.Print "Stock Code";
Printer.CurrentX = x7 + 30
Printer.Print "Stock Name";
Printer.CurrentX = x6 + 30
Printer.Print "Quantity";
Printer.CurrentX = x5 + 30
Printer.Print "Unit Price";
Printer.CurrentX = x3 + 30
Printer.Print "Total Amount"
Printer.Line (30, Printer.CurrentY)-(x4 - 30, Printer.CurrentY)
For k = 0 To List1(0).ListCount - 1
Printer.CurrentX = 30
Printer.Print List1(0).List(k);
Printer.CurrentX = x7 + 30
Printer.Print List1(1).List(k);
Printer.CurrentX = x6 + 30
Printer.Print List1(2).List(k);
Printer.CurrentX = x5 + 30
Printer.Print List1(3).List(k);
```

```
    Printer.CurrentX = x3 + 30

    Printer.Print List1(4).List(k)
    Printer.Line (30, Printer.CurrentY)-(x4 - 30, Printer.CurrentY)
Next k

    ys = Printer.CurrentY
Printer.Line (30, yb)-(30, ys)
Printer.Line (x7 + 30, yb)-(x7 + 30, ys)
Printer.Line (x6 + 30, yb)-(x6 + 30, ys)
Printer.Line (x5 + 30, yb)-(x5 + 30, ys)
Printer.Line (x3 + 30, yb)-(x3 + 30, ys)
Printer.Line (x4 + 30, yb)-(x4 + 30, ys)
Printer.EndDoc
Exit Sub
ass:
MsgBox "Error :" & Err.Description, 16, "Account"
End Sub
```

Account Revenue Rrocess

```
Private Sub Command1_Click()
Dim ask
If Option1 = True Then
On Error Resume Next
If Text1.text = "" Then
ask = MsgBox("Please Enter The Selected Criteria!", vbCritical, "Account")
Frame1.Visible = False
Text1.SetFocus
Else
```

```
conne

text = "select incode, cuscode, invdate, empname, subtot from invoice2 where incode = " &
Text1.text & " and subtot>0 "

Set rst = database.Execute(text)

If rst.EOF Then

ask = MsgBox("Wanted Invoice No Not Exist!", vbCritical, "Account")

Label2.Visible = False

Label3.Visible = False

Frame1.Visible = False

Else

textq = "select  sum(subtot) as ah  from invoice2 where incode=" & Text1.text & ""

Set rst1 = database.Execute(textq)

a = rst![cuscode]

Set DataGrid1.DataSource = rst

DataGrid1.Caption = "LIST OF INVOICE BY INVOICE CODE"

DataGrid1.Columns(0).Caption = "Invoice No"

DataGrid1.Columns(1).Caption = "Customer No"

DataGrid1.Columns(2).Caption = "Invoice Date"

DataGrid1.Columns(3).Caption = "Employee Name"

DataGrid1.Columns(4).Caption = "Invoice Amount"

DataGrid1.Columns(4).Alignment = dbgRight

'DataGrid1.Columns(4).NumberFormat = "#,##0"

Label3.Caption = rst1![ah]

Label2.Visible = True

Label3.Visible = True

Frame1.Visible = True
```

```
End If

End If

End If

If Option2 = True Then

On Error Resume Next

If Text1.text = "" Then

ask = MsgBox("Please Enter The Selected Criteria!", vbCritical, "Account")

Frame1.Visible = False

Text1.SetFocus

Else

conne

text = "select incode, cuscode, invdate, empname, subtot from invoice2 where cuscode = " &
Text1.text & " and Subtot>0 "

Set rst = database.Execute(text)

If rst.EOF Then

ask = MsgBox("Wanted Customer No Not Exist!", vbCritical, "Account")

Label2.Visible = False

Label3.Visible = False

Frame1.Visible = False

Else

textq = "select  sum(subtot) as ah  from invoice2 where cuscode=" & Text1.text & ""

Set rst1 = database.Execute(textq)

a = rst![cuscode]

Set DataGrid1.DataSource = rst

DataGrid1.Caption = "LIST OF INVOICE BY CUSTOMER CODE"

DataGrid1.Columns(0).Caption = "Invoice No"
```

```
DataGrid1.Columns(1).Caption = "Customer No"
DataGrid1.Columns(2).Caption = "Invoice Date"
DataGrid1.Columns(3).Caption = "Employee Name"
DataGrid1.Columns(4).Caption = "Invoice Amount"
DataGrid1.Columns(4).Alignment = dbgRight
'DataGrid1.Columns(4).NumberFormat = "#,##0"
Label3.Caption = rst1![ah]
Label2.Visible = True
Label3.Visible = True
Frame1.Visible = True
End If
End If
End If
Text1.text = ""
Text1.SetFocus
End Sub
Private Sub Command2_Click()
com
Text1.text = ""
Text1.SetFocus
Label2.Visible = False
Label3.Visible = False
Label4.Visible = False
Label5.Visible = False
Label6.Visible = True
Option1 = False
```

```
Option2 = False

Command1.Enabled = False

End Sub

Private Sub Command3_Click()

Dim ask

If Text2.text <> "" And Text3.text <> "" Then

decrip = Text2 + "," + "no" + "Invoice"

b = 0

c = Text3.text

Set rst2 = database.Execute("update invoice2 set subtot=subtot-" & c & " where incode=" &
Text2 & " and cuscode=" & a & " ")

conne

Set rst3 = database.Execute("insert into account(revenue, accdate, description, expense )
values(" & Text3.text & ", '" & Label12 & "', '" & decrip & "'," & b & ")")

Frame1.Visible = False

Text1.text = ""

Text1.SetFocus

ask = MsgBox("Invoice Paid", vbInformation, "Account")

Text2.text = ""

Text3.text = ""

Else
ask = MsgBox("Please Insert The Invoice No / Amount!", vbCritical, "Account")

Text2.text = ""

Text3.text = ""

Text2.SetFocus

End If

End Sub
```

```
Private Sub Command4_Click()
Frame1.Visible = False
Text1.text = ""
Text1.SetFocus
End Sub
Private Sub Command5_Click()
database.Close
Unload Me
form1.Show
End Sub
Private Sub Form_Load()
Label12.Caption = Date
com
Command1.Enabled = False
End Sub
Private Sub com()
On Error Resume Next
conne
Set rst = database.Execute("Select incode, cuscode, invdate, empname, subtot from invoice2
where subtot >0")
Set DataGrid1.DataSource = rst
DataGrid1.Caption = "LIST OF INVOICE BY INVOICE CODE"
DataGrid1.Columns(0).Caption = "Invoice No"
DataGrid1.Columns(1).Caption = "Customer No"
DataGrid1.Columns(2).Caption = "Invoice Date"
DataGrid1.Columns(3).Caption = "Employee Name"
```

```
DataGrid1.Columns(4).Caption = "Invoice Amount"
DataGrid1.Columns(4).Alignment = dbgRight
'DataGrid1.Columns(4).NumberFormat = "#,##0"
Select Case rst.RecordCount
Case Is > 1
Label8.Caption = "Total  " & Trim(Str(rst.RecordCount)) & "  Record Exist"
Case Is = 1
Label8.Caption = "Total  " & Trim(Str(rst.RecordCount)) & "  Record Exist"
Case Is = 0
Label8.Caption = "Record Not Exist"
End Select
form1.Enabled = False
End Sub
Private Sub Form_Unload(Cancel As Integer)
Unload Me
form1.Show
form1.Enabled = True
End Sub
Private Sub Option1_Click()
Label4.Visible = True
Label5.Visible = False
Label6.Visible = False
Text1.SetFocus
Command1.Enabled = True
End Sub
Private Sub Option2_Click()
```

```
Label5.Visible = True
Label6.Visible = False
Text1.SetFocus
Command1.Enabled = True
End Sub
Private Sub conne()
Set database = New ADODB.Connection
   database.CursorLocation = adUseClient
   ersoy = "provider=Microsoft.jet.oledb.3.51; Data Source=" & App.Path & "\ezgi.mdb"
   database.Open ersoy
End Sub
Private Sub Text1_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
   KeyAscii = 0
   SendKeys "{Tab}"
ElseIf InStr(("1234567890" & vbBack & ""), Chr(KeyAscii)) = 0 Then
   KeyAscii = 0
End If
End Sub
Private Sub Text2_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
   KeyAscii = 0
   SendKeys "{Tab}"
ElseIf InStr(("1234567890" & vbBack & ""), Chr(KeyAscii)) = 0 Then
   KeyAscii = 0
End If
```

```
End Sub
Private Sub Text3_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
   KeyAscii = 0
   SendKeys "{Tab}"
ElseIf InStr(("1234567890" & vbBack & ""), Chr(KeyAscii)) = 0 Then
   KeyAscii = 0
End If
End Sub
```

Account Expense Process (code)

```
Private Sub Command1_Click()
database.Close
Unload Me
form1.Show
form1.Enabled = True
End Sub
Private Sub Command2_Click()
Dim sql, ask, ask1
ask = MsgBox("Do You Want To Save That ?", vbInformation + vbYesNo, "Save")
If ask = vbYes Then
If Text2.text <> "" And Text3.text <> "" And Text3.text <> "" Then
sql = "insert into account(expcode, description, expense, accdate) values("
sql = sql & "" & Text1.text & ","
sql = sql & "'" & Text2.text & "',"
sql = sql & "" & Text3 & ","
sql = sql & "'" & DTPicker1.Value & "')"
```

```
database.Execute (sql)
Dim i As Integer
ProgressBar1.Min = 0
ProgressBar1.Max = 1000
For i = ProgressBar1.Min To ProgressBar1.Max
ProgressBar1.Visible = True
ProgressBar1.Value = i
Next
ProgressBar1.Visible = False
ask1 = MsgBox("Employee Information Save Successful! ", , "Saved")
coun
Text2.text = ""
Text3.text = ""
Text2.SetFocus
Else
ask1 = MsgBox("Please Fill The Other Texts!", vbInformation, "Account")
End If
End If
End Sub
Private Sub Form_Load()
On Error Resume Next
form1.Enabled = False
coun
Adodc1.RecordSource = "select expcode ,description, accdate, expense from account where
expcode > 0 "
Set DataGrid1.DataSource = Adodc1
```

```
DataGrid1.Caption = "LIST OF EXPENSE BY INVOICE CODE"
DataGrid1.Columns(0).Caption = "Expense No"
DataGrid1.Columns(1).Caption = "Description"
DataGrid1.Columns(2).Caption = "Expense Date"
DataGrid1.Columns(3).Caption = "Expense Amount"
DataGrid1.Columns(3).Alignment = dbgRight
conn
sql = "select  sum(expense) as ah  from account"
Set ezgi = database.Execute(sql)
Text6.text = ezgi![ah]
sql1 = "select  sum(revenue) as ah  from account"
Set ezgi = database.Execute(sql1)
Text4.text = ezgi![ah]
Text5.text = Val(Text4.text) - Val(Text6.text)
ProgressBar1.Align = vbAlignBottom
ProgressBar1.Visible = False
End Sub
Private Sub Form_Unload(Cancel As Integer)
Unload Me
form1.Show
form1.Enabled = True
End Sub
Public Sub conn()
Set database = New ADODB.Connection
   database.CursorLocation = adUseServer
   ersoy = "provider=Microsoft.jet.oledb.3.51; Data Source=" & App.Path & "\ezgi.mdb"
```

```
    database.Open ersoy
End Sub
Private Sub coun()
Dim Count, Count1
conn
Set ezgi = New ADODB.Recordset
Count = "select expcode from account"
Set ezgi = database.Execute(Count)
    If ezgi.EOF Then
    Text1.text = 1
    Else
    Count1 = "select max(expcode) as cis from account"
    Set ezgi = database.Execute(Count1)
    Text1.text = ezgi![cis] + 1
    End If
    ezgi.Close
End Sub
Private Sub Text3_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
    KeyAscii = 0
    SendKeys "{Tab}"
ElseIf InStr(("1234567890" & vbBack & ""), Chr(KeyAscii)) = 0 Then
    KeyAscii = 0
End If
End Sub
```

Account (code)

```
Private Sub Command1_Click()

If Len(Text1.text) = 0 Then

ask = MsgBox("Please Enter The Stock Name", vbCritical, "Search")

Text1.SetFocus

Else

conne

ezgi = "select stockcode, stockname, firmname from stocks where stockname like '" & Text1.text & "%" & "' "
Set rst = database.Execute(ezgi)

If rst.EOF Then

ask = MsgBox("Wanted Product Name Not Exist!", vbCritical, "Account")
Text1.text = ""

Text1.SetFocus

Else
Set DataGrid1.DataSource = rst

DataGrid1.Caption = "LIST OF PRODUCT BY PRODUCT NAME"

DataGrid1.Columns(0).Caption = "Stock Code"

DataGrid1.Columns(1).Caption = "Stock Name"

DataGrid1.Columns(2).Caption = "Firm Name"

Text1.text = ""

Text1.SetFocus

End If

End If

End Sub
```

```
Private Sub conne()

Set database = New ADODB.Connection
   database.CursorLocation = adUseClient
   ersoy = "provider=Microsoft.jet.oledb.3.51; Data Source=" & App.Path & "\ezgi.mdb"
   database.Open ersoy

End Sub

Private Sub Form_Load()

Form7.Enabled = False

End Sub

Private Sub Form_Unload(Cancel As Integer)

Form7.Enabled = True

End Sub
```

Stock Records Min. Level (code)

```
Private Sub Command1_Click()

Unload Me

Form2.Show

End Sub

Private Sub Command2_Click()

Unload Me

form1.Show

End Sub
Private Sub Form_Load()

com

End Sub

Private Sub com()

On Error Resume Next
```

```
conne

Set rst = database.Execute("Select stockcode, stockname,firmcode, stockunit,stockminl from stocks where stockminl >= stockunit")

Set DataGrid1.DataSource = rst

DataGrid1.Caption = "LIST OF MINIMUM LEVEL GRATER THAN AVAİLABLE STOCK UNIT BY STOCKCODE"

DataGrid1.Columns(0).Caption = "StockCode"

DataGrid1.Columns(1).Caption = "StockName"

DataGrid1.Columns(2).Caption = "FirmCode"

DataGrid1.Columns(3).Caption = "StockUnit"

DataGrid1.Columns(4).Caption = "Stock Min.Level"

DataGrid1.Columns(4).Alignment = dbgRight

Select Case rst.RecordCount

Case Is > 1

Label1.Caption = "Total  " & Trim(Str(rst.RecordCount)) & " Min. Level >= Stock Quantity Record Exist"

Case Is = 1

Label1.Caption = "Total  " & Trim(Str(rst.RecordCount)) & " Min. Level >= Stock Quantity Record Exist"

Case Is = 0

Label1.Caption = "Record Not Exist"

End Select

form1.Enabled = False

End Sub

Private Sub conne()

Set database = New ADODB.Connection

  database.CursorLocation = adUseClient

  ersoy = "provider=Microsoft.jet.oledb.3.51; Data Source=" & App.Path & "\ezgi.mdb"
```

```
    database.Open ersoy

End Sub

Private Sub Form_Unload(Cancel As Integer)
Unload Me

Unload Form2

form1.Show

form1.Enabled = True

End Sub

form1.Enabled = False

 Set grdDataGrid.DataSource = datPrimaryRS.Recordset("ChildCMD").UnderlyingValue

End Sub

Private Sub Form_Resize()

 On Error Resume Next

 grdDataGrid.Width = Me.ScaleWidth

 grdDataGrid.Height = Me.ScaleHeight - grdDataGrid.Top - datPrimaryRS.Height - 30 -
picButtons.Height

End Sub

Private Sub Form_Unload(Cancel As Integer)

 Unload Me

form1.Show

form1.Enabled = True

 Screen.MousePointer = vbDefault

End Sub

Private Sub datPrimaryRS_Error(ByVal ErrorNumber As Long, Description As String,
ByVal Scode As Long, ByVal Source As String, ByVal HelpFile As String, ByVal
HelpContext As Long, fCancelDisplay As Boolean)

 MsgBox "Data error event hit err:" & Description
```

```
End Sub

Private Sub datPrimaryRS_MoveComplete(ByVal adReason As ADODB.EventReasonEnum,
ByVal pError As ADODB.Error, adStatus As ADODB.EventStatusEnum, ByVal pRecordset
As ADODB.Recordset)

 datPrimaryRS.Caption = "Record: " & CStr(datPrimaryRS.Recordset.AbsolutePosition)

End Sub

Private Sub datPrimaryRS_WillChangeRecord(ByValadReason As
ADODB.EventReasonEnum, ByVal cRecords As Long, adStatus As
ADODB.EventStatusEnum, ByVal pRecordset As ADODB.Recordset) Dim bCancel As
Boolean

 Select Case adReason

 Case adRsnAddNew

 Case adRsnClose

 Case adRsnDelete

 Case adRsnFirstChange

 Case adRsnMove

 Case adRsnRequery

 Case adRsnResynch

 Case adRsnUndoAddNew

 Case adRsnUndoDelete

 Case adRsnUndoUpdate

 Case adRsnUpdate

 End Select

 If bCancel Then adStatus = adStatusCancel

End Sub

Private Sub cmdRefresh_Click()

 On Error GoTo RefreshErr
```

```
 datPrimaryRS.Refresh
 Set grdDataGrid.DataSource = datPrimaryRS.Recordset("ChildCMD").UnderlyingValue
 Exit Sub
RefreshErr:
 MsgBox Err.Description
End Sub
Private Sub cmdClose_Click()
 form1.Show
 form1.Enabled = True
 Unload Me
End Sub
```

Module (code)

```
Public Sub Main()
 Dim Count, Count1, i
  Load form1
  Form11.Show
  DoEvents
  Form11.ProgressBar1.Min = 0
  Form11.ProgressBar1.Max = 13000
  Form11.ProgressBar1.Value = 0
  For i = 0 To 13000 - 1
     Form11.ProgressBar1.Value = Form11.ProgressBar1.Value + 1
  Next i
  Unload Form11
 conn
  Set ezgi = New ADODB.Recordset
```

```
    Count = "select * from stocks where stockminl>=Stockunit"
    Set ezgi = database.Execute(Count)
    If ezgi.EOF Then
        form1.Show
    Else
    Form12.Show
    End If
    ezgi.Close
End Sub
Public Sub conn()
Set database = New ADODB.Connection
    database.CursorLocation = adUseClient
    ersoy = "provider=Microsoft.jet.oledb.3.51; Data Source=" & App.Path & "\ezgi.mdb"
    database.Open ersoy
End Sub
```

APPENDIX B

account : Tablo

	accdate	expense	revenue	description	expcode
▶	21.10.2005	12		hahhajj	1
*					

Table.account

cus : Tablo

	cuscode	cusname	cusphone	custaxno	cuscity	cusadres	cusrd
▶	2	mahmut	2366677776	8282882882892	izmir	turkey/izmir	23.12.2004
	3	emine	6363737377	7377388388836	turkey	manisa	23.12.2004
*							

Table.customer

emp : Tablo

	empcode	empname	empphone	empsc	empcity	empadres	emprd	empsal
▶	1	ahmet	3333333333	4444444444444	belçika	belçika	21.10.2005	627288282
*								

Table.employee

Firms : Tablo

	firmname	firmcode	firmphone	firmfax	firmtaxno	firmcity	firmadres	firmregdate
▶	a	1	11	1	11	1	1	01.08.2003
	nnnmfgdfgtt	2	6656565455	5656565657	6575675656554	45645654	5464565464564	22.12.2004
*								

Table.firms

invoice1 : Tablo

	incode	cuscode	stockcode	sname	quan	uprice	totp
▶							

Table.invoice

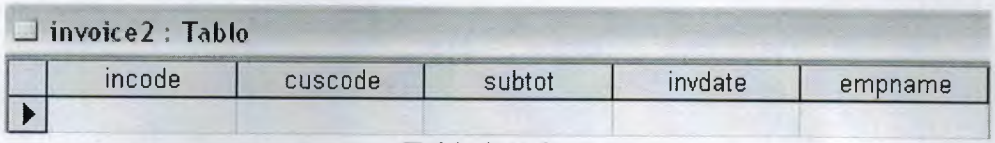

Table.invoice2

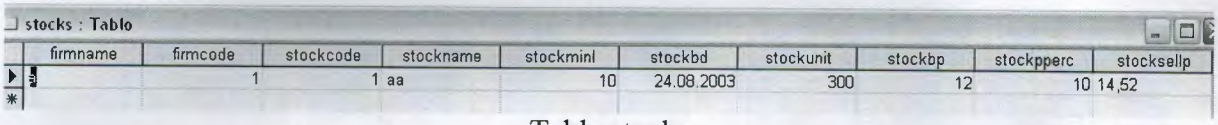

Table.stocks