NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

HOSPITAL AUTOMATION SYSTEM

Graduation Project

COM-400

Student: Hanife Otbasan (20020962)

Supervisor: Ümit İlhan

Nicosia - 2007

ABSTRACT

My graduation project is a hospital automation program. Hospital automation programs are very important computer programs in our time. Parallel to the speed of technological changes in recent years, the need for technological equipments in hospitals is increased. Growing population of our countries and parallel increase of need for health services require fast, true, and easy ways of providing qualified health services. Hospital automation programs help hospitals to provide these qualified health services.

Hospital contains departments of eye disorders, dentistry, cardiology, neurology, internal medicine, and orthopedics.

The aim of the program is patient registration, arranging appointment to a registered patient, having doctor registration, providing doctor the opportunity to prepare an examination paper for registered patient and reaching these documents when doctor wants, making out an invoice, and reflecting changing examination prices to program easily.

TABLE OF CONTENTS

INTRODUCTION

In traditional or procedural applications, the application itself controls which portions of code execute and in what sequence. Execution starts with the first line of code and follows a predefined path through the application calling procedures as needed.

In an event-driven application, the code doesn't follow a predetermined path –it executes different code sections in response to events. Events can be triggered by the user's actions, by messages from the system or other applications, or even from the application itself. The sequence of these events determines the sequence in which the code executes, thus the path through the application's code differs each time the program runs.

Chapter One describes Visual Basic in details,History of Visual Basic, Visual Basic environment, Visual Basic controls, variables, data bases, messages, events, logical comparisons, conditional statements, and functions.

Chapter Two describes; access, and how to form a table in access, also the tables of the data bases of my program.

Chapter Three describes; hospital automation program. This program is designed for three different users: doctor, secretary, and administrator. All users have to enter the program with user name and password. Doctor can only reach to preparing examination for the patient form. Secretary can only reach registration, making out of invoice and appointment forms. Administrator provides entrance to database for new users. Program is composed of

- Doctor record
- Patient record
- Department record
- Appointment
- Examination
- Invoice
- Report

CHAPTER ONE: WHAT IS VISUAL BASIC?

1.1 History

Microsoft released Visual Basic in 1987. It was the first visual development tool from Microsoft, and it was to compete with C, C++, Pascal and other well-known programming languages. From the start, Visual Basic wasn't a hit. It wasn't until release 2.0 in 1991 that people really discovered the potential of the language, and with release 3.0 it had become the fastest-growing programming language on the market.

1.1.1 What Is Visual Basic?

Programmers have undergone a major change in many years of programming various machines. For example what could be created in minutes with Visual Basic could take days in other languages such: as "C" or "Pascal". Visual Basic provides many interesting sets of tools to aid you in building exciting applications. Visual Basic provides these tools to make your life far more easier because all the real hard code is already written for you.

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

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

1.2 Areas of Application

The term "*Personal Programming*" refers to the idea that, wherever you work, whatever you do, you can expand your computer's usefulness by writing applications to use in your own job. Personal Programming is what Visual Basic is all about.Using Visual Basic's tools, you quickly translate an abstract idea into a program design you can actually see on the screen. VB encourages you to experiment, revise, correct, and network your design until the new project meets your requirements. However , most of all, it inspires your imagination and creativity. Visual Basic is ideal for developing applications that run in the new Windows 95 operating system. VB presents a 3-step approach for creating programs:

Design the appearance of your application.

Assign property settings to the objects of your program.

Write the code to direct specific tasks at runtime.

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

Eucation

Research

Medecine

Business

Commerce

Marketing and Sales

Accounting

Consulting

Law

Science

1.3 The Visual Basic Environment

Visual Basic (VB) was introduced as an environment in which users were achieving a great deal of reuse. But there are reasons for this, not necessarily those that we as SSR researchers can leverage. Our working group garnered a great deal of incite into the SSR problem from this environment example.

Visual Basic (VB) provides a nice architectural framework, and allows the everyday user to develop a straightforward mental model with a well-defined process for plugging together widgets. But there are a great number of architectural assumptions made in VB. If the user is willing to buy into these assumptions (primarily interface style and functionality), then a tremendous amount of speedup can be achieved in designing a very specific class of applications, namely small, single user applications with relatively uncomplicated backends that can be implemented in a basic variant fairly easily.

Users of VB essentially do design with reuse, since they are not given the proper tools to reasonably extend the widget set. In this way the questions in the VB and SSR communities are different. The SSR community is concerned (1) with issues integrating design with and for reuse, and (2) is concerned with the design of fairly large, multiple implementor, ``long time to completion" projects.

VB provides the user with pre-designed VBXs, which are the vehicles for all of the interface widgets (buttons, sliders, etc.) in the toolbox. In addition, a wide variety of components are available both commercially and in the public domain. But from a design for reuse standpoint, it turns out to be incredibly hard to build such widgets. They come "shrink- wrapped" and are not intended to be modified by users.

Robert Biddle compared VB to the Tcl/Tk toolkit . Both of these environments are evidence that tremendous progress has been made since X in rapidly prototyping interfaces. Less progress has been made in tools to rapidly prototype fairly complex backends, which tend to be more domain dependent than the interface architecture.

In summary, in VB, users:

tend to want their problem solved very quickly

tend to want good prototyping tools

tend to want to do it themselves

tend to be non-software engineers (modelers, biologists, chemists, technical people not in the business of writing software, i.e, *users*.)

Other environments with VB-like qualities were briefly discussed:

Visual C++

Spreadsheets

Scripting languages (Unix shells, Tcl/Tk, Applescript, etc.)

Multi-media development languages (Lingo, Hypertalk, Authorware, etc.)

1.4 Introduction to Windows Controls

A Windows control, also called a control, is an object that allows the user to interact with the computer. Such an object must be displayed on the screen or somehow made available to the user who can then click it, move it, resize it, type in it or retrieve something from it. Because there are so many operations a user can perform on the computer, controls are separate in categories according to their functionality and their roles in an application. Nevertheless, to make your application effective, as the developer, you will decide what the user can do with your application and what should be excluded.

When creating your application, you add controls to it as you judge them relevant for the possible assignments that can be performed on your application. While working, you will deal with two big categories of controls: those that can act parents and those that (always) need some parenting.

1.4.1 Controls Fundamentals

You as the developer will decide what control should be available in your application, what functionality that control should provide, and what the user can do with it. Some of the functionality is controlled by the operating system because such a functionality is part of the computer's behavior. Some other aspects are under your control.

When creating your application, you will most likely start from a form. We will have a better study of forms when the time comes. Other controls are added to the form. To use one of them, you will get it from an object called the Toolbox and then add them to the form. Once a control is available to you, you can customize its appearance and behavior, which is the subject of this site.

To implement their intended assignment, one of the most regular operations a control perform is to fire events.

1.4.2 Control Design

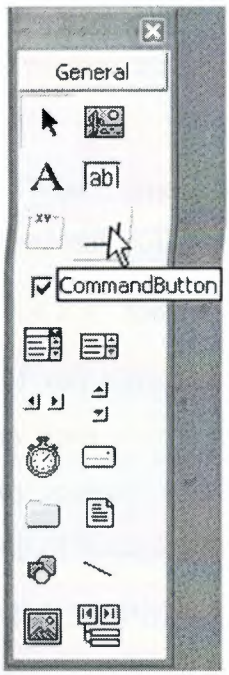
To interactive with the computer, the user submits requests to the machine and the computer processes these assignments. This exchange of information is done through

objects called controls, also called components. Almost any object you see on your screen is a window and we are going to learn how to create and use them with Visual Basic. When you start Visual Basic and select Standard EXE, it creates a form for you: this is the first and the most commonly object you will use in your applications

1.4.2.1 Control Selection

To provide the necessary functionality for your application, you will use controls from the Toolbox and add them to another component such as a form. The control you pick up from the Toolbox is also referred to as a child control. the control or object on which you add a child control is referred to as its parent or host. This can be a form or another object that has the capacity to host other controls.

To identify a control on the Toolbox, you can position the mouse on it. A tool tip would appear:



From now on, we will call each control by the tool tip that appears on it.

To add a control to a host, on the toolbox, you can double-click it. Alternatively, you can click the control on the toolbox and then "draw" it on the host. You can keep adding controls to a host as necessary.

If you want to add a control over and over again, you can press and hold Ctrl, click the control on the Toolbox, then draw it in the desired area on the host. Every time you draw, the control would be added to the form or host. Once you have added enough controls, you can release Ctrl.

If you select a control by mistake, you can simply click another. The new one would become selected. If you clicked a control but don't want any control at all, you can click the Pointer button  .

You cannot select more than one control to add to a host.

1.4.2.2 Selecting Controls

Most of the time, before doing anything on a control, you must first select it. In the same way to perform an action on a group of controls, you must first select them.

To select one control on the form, you can just click it. Alternatively, you can click and hold the mouse somewhere on the form but close to the control. Then drag as if you were drawing a line. Once you have touched the control, you can release the mouse. The control would be selected.

To select more than one control at random, click one of them, press and hold Shift or Ctrl, then click each of the desired controls. Once you are satisfied with the group, release the key you were pressing.

To select more than one control in a range, click and drag to draw a rectangle. Any control that would be touched by the fake rectangle would be included in the group.

1.4.2.3 Deleting Controls

If you have one of more controls that you don't need anymore, you can remove them from your form. To remove one control, select it and press Delete. To remove many controls, first select them, then press Delete. You can also select a control or a group of control, then right-click and click Cut.

1.4.3 Properties of Controls

If you access a code when designing the application, it is said that you are working at design time. If you access a control with code, it is said that you are at run time. Therefore, design time refers to the form being designed while displaying in Visual Basic. Run time refers to the time the control is displaying to the user.

After adding a control to the application, you can customize it. For example, you can change some parts of its appearance. You can also give it assignments. These are done from two parts: the Properties window and the Code editor.

Controls are broadly classified in two groups. A control is referred to as graphical if the user can see it. There are other controls that will work behind the scenes at run time. Such control are not graphical (an example is the Timer). They can be referred to as static. The user never sees these controls. There are some other controls not considered graphical because the user cannot directly change their values. For example, a control that displays only text (such is the case for the Label) is not considered graphical.

A Windows control is an object that imitates a real world object. As such, it is made of characteristics that define it. A characteristic is also called a property. A property is any aspect that describes an object.

Once you have a control, you can change its properties in the Properties window. This is considered that you are controlling the properties at "design time". To change the properties of a control, first select it, then proceed with changing the desired properties in the Properties window.

To control a form's properties with code, you will refer to itself. A form refers to itself using the me keyword. To change the properties of a control with code, you refer to it by its name. Whether dealing with a form or a control, after typing **Me** for the form, or the name of the control, type a period. A list of the properties (and possibly other objects that we will know eventually) will appear. You can continue typing or simply select from the list. And continue with your coding.

Not all properties can be changed with code

1.4.3.1 Controls' Names

Everything on a computer must have a name. In the same way, to refer to a control in your code, you must give it a name. When you add a new control to your application, it receives a default name. When necessary, which will be almost all the time, you should change that name to a more recognizable one.

To change the name of a control, first select it. Then, in the Properties window, click (Name) and type the desired name. Refrain from changing the name of a control with code.

1.4.3.2 Controls' Text and Caption

Some controls are meant to display or sometimes request text from the user. For such controls, this text is referred to as caption while it is simply called text for some

other controls. This property is not available for all controls.

If a control displays text, it then has a **Caption** or a **Text** field in the Properties window. After adding such a control to a form, its **Caption** or its **Text** field may the same text as its name. At design time, to change the text of the control, click either its **Caption** or its **Text** field and type the desired value. For most controls, there are no strict rules to follow for this text. Therefore, it is your responsibility to type the right value.

The text provided in a **Caption** or **Text** fields of a text-based control can only be set "as is" at during design. If you want the text to change while the application is running, you can format it. For example, such a control can display the current time or the name of the user who is logged in. These format attributes cannot be set at deign time. To change the text of a text-based control at run time, either assign a simple string or provide a formatted string to either the **Caption** or the Text property.

1.4.3.3 Controls' Visibility

For the user to directly use a control, he or she must be able to see that control. For example, the user cannot type an employee's name if there is not control to receive that text. Based on this, objects provide the ability to control their visibility or absence. This characteristic is controlled by the Visible property.

The default visibility of graphical controls have their Visible property set to True. To hide a control, set its Visible property to False. You can change this value at design time using the Properties window. You can also change it programmatically.

1.4.3.4 Controls' Availability

Even if a control is visibility, it doesn't necessary make its services available to the user. This means that a control can enable its role or lock them. When a control is enabled, the user can click it or type in it. You can also prevent this type of action by disabling the control.

The ability to enable or disable an object is controlled by the Enabled property. If you set it to True, which is its default value, the service of the control are available to the user. If you set this property to False, the control appears gray.

Even if a control is visibility, it doesn't necessary make its services available to the user. This means that a control can enable its role or lock them. When a control is

enabled, the user can click it or type in it. You can also prevent this type of action by disabling the control.

The ability to enable or disable an object is controlled by the Enabled property. If you set it to True, which is its default value, the service of the control are available to the user. If you set this property to False, the control appears gray.

1.4.3.5 Tab Sequence

When a form contains many controls, the user can navigate to different ones by clicking them. Alternatively, the user can press Tab to move the focus from one control to another. The controls that can be accessed using the Tab key belong to a group. For a control to participate to this group, it must have its **TabStop** property set to **True**. All graphical controls are automatically added to this group by default when they are picked from the Toolbox and added to a form. If you don't want a control to receive focus as a result of the user pressing Tab, set its **TabStop** to **False**.

Each control in the Tab sequence group has a unique incremental number. This number is called **TabIndex**. The first control added receives a number of 0. The second receives 1, etc. The control whose **TabIndex** is the lowest would receive focus when the form comes up. If you want, you can change the default sequence by changing the **TabIndex** values of the controls

1.5 Controls Messages and Events

1.5.1 Controls Events

Because your computer is made of various objects, these are under the control of three entities:

The operating system is first in charge of all the basic operations that must be performed in order for the computer to be usable. Some of these include checking that the various parts (hardware) are ready to be used, displaying the time, checking that a printer is working, etc

The person who creates a program, that is you, is in charge of a particular application. For example, as we will learn on this site, when you create a program, you decide what it can do and what it shouldn't do. You also decide if and when it should do something

The user, that is, a person who uses the computer or a program that you have written also controls such aspects as when to open a program, when to perform a certain processing

Every time you do something on the computer, the object you interact with composes a message like an email and sends that message to another entity, normally the operating system, that can process that message. This means that, whether you click an object, type something using the keyboard, move the mouse on the screen, or press and click at the same time, a message is composed. Once a message has been composed, it is sent to the OS. Once a message is received, it is analyzed and interpreted. Then a result may be sent to the application from where the message was sent.

The action of sending a message is called an event. There are so many objects you can use on a computer, and/or on an application, that Microsoft Windows is referred to as an event-driven operating system.

After the computer has been launched, it becomes "static" and displays a blank desktop to the user. For example, the computer cannot start a program on its own and it cannot just start typing words on the desktop... (Even if this happens as a result of a script, it doesn't mean that the computer did it on its own, it means that somebody asked it to do it; it is important to understand that the computer is a dumb object that doesn't think. For the computer to do something automatically, something has to ask to do it, whether it is a script you wrote (Windows Script Host(WSH)) or a virus somebody sent you; the computer can't just decide to do something on its own.).

Because there are so many actions that can be performed, the computer cannot predict what should be done next. Therefore, it leaves it up to the user to initiate an action. Again, because there are so many things that could happen at any time, for the computer to do what is needed, it expects good and precise directives. Based on this, Microsoft Windows uses a mechanism like a mail you send to somebody through the post office.

When a message is sent, some conditions must be met for the message to be processed. If a message is not well defined, either the computer would ignore it (best case scenario) or it would crash. For this reason, as a programmer, you need to know what messages you can send (there are are so many messages for different reasons),

when you can send a certain message (you cannot just send any message at any time), why (it is important that you know the reason for sending a message, otherwise you may send the right message at the right time but the message cannot be processed because either it is not needed, not necessary, or not efficient).

After a message has been formulated, it must be sent. The action of sending a message is called an event.

In order for a message to be processed, it must provide at least three pieces of information

1.5.1.1. The Sender of a Message

The first piece of information necessary for each message is:

WHO sent the message? The computer contains many applications and each application is made of various internal objects. Any application can create a message anytime and send it. Since there can be so many messages sent to the operating system at any time, the application or the object that sent the message must be identified. In some cases the operating system may need to send a response when it has finished processing the message, in some other cases, it needs to identify the object so it would know how the message must be processed.

Therefore, the coding of each event starts with the **Private** keyword:

Private

An event is really an assignment you ask the application, the form, or the control to perform in response to a particular action happening. You can even ask an object to perform an action based on the behavior of another object or based on the computer doing something (such as singing when the clock displays 12:00 PM). Since there are so many assignments you will give to different components to perform, these actions are called procedures. There are two kinds of procedures: **Function**s and **Sub** procedures. Both are written in Visual Basic.

A **Function** is a general assignment you write in Visual Basic. This assignment is a resource for other events or actions to get results. For example, if many controls on a form would require a particular value or the result of a particular calculation, you can write a function that all desired events can refer to and get the appropriate result. Since

other events and functions would expect a particular result from it, a function is expected to *Return* a value. We will learn what kind of value a function can return.

A **Sub** procedure is a form of assignment that applies to an event associated with particular application, form, or control. It is used to "enclose" the coded assignment you want an event to carry. Since each event is a procedure, now we have:

Private Sub

As mentioned already, the object that sent a message must be identified. Therefore, the Sub keyword would be followed by that object:

Private Sub *MessageSender*

1.5.1.2 The Type of Message

After the sender of a message has been identified, the operating system would need to know:

WHAT message was sent? There are various objects in the computer and applications. Some objects can send the same type of message. Some other objects have particular messages that only they can send. Because one type of object can send different types of messages, even if the operating system has been able to identify the sender, it needs to know the type of message that was sent. By convention, and as we will see later on, the name of a message produces the name of the event

By convention, the name of the event is written after the name of the object that sent the message. To distinguish between a control's name and its event, Visual Basic uses a convention of displaying an underscore between them, like this:

Private Sub *MessageSender_Event*

1.5.1.3 The Message Accessories

Once the operating system knows what object sent the message and what that message is, depending on the message, it may need to know:

the accessories needed to process the message

While one message may appear easy, such as clicking an object, another message would need additional information such as where (the coordinates of the mouse cursor) the clicking occurred. Therefore, some events will need some values from you. In some situations it will be one value; in this case the accessory is called an argument. Another type of event may need more than one accessory, thus many arguments. Again,

depending on the event, this could be one argument, or it could be as many arguments as necessary. When we move on, we will see what events need what argument(s)

The argument or group of arguments that the event may need is listed in parentheses on the right side of the event name, like this:

Private Sub *MessageSender_Event*(*Argument1*, *Argument2*, *Argument_n*)

Even if an event doesn't need an argument, you must provide empty parentheses, like this:

Private Sub *MessageSender_Event*()

1.5.2 Categories of Events

1.5.2.1 The Keyboard Events

Word processing consists of manipulating text and characters on your computer until you get the fantastic result you long for. To display these characters, you press some keys on your keyboard. If the application is configured to receive text, your pressing actions will display characters on the screen.

The keyboard is also used to perform various other actions such as accepting what a dialog box displays or dismissing one.

When you press the keys on your keyboard, you are sending keyboard events

1.5.2.2 The Click Event

The mouse has become a very important object of computer use. It is used by pressing one of its buttons.

1.5.2.3 The Double-Click Event

When you press the left mouse button once, the event is called the Click event. Another action you can perform is to click the button twice but very fast. This is referred to as double-clicking

1.5.2.4 The Right-Click Event

Since Microsoft Windows 95, the mouse buttons are intensely used and both buttons have become important object of the computer daily use. By default, the users click the left mouse button for all routine work. The other button, the right one, is used in various circumstances, such as displaying a context menu.

The Right-Click action is performed by clicking the right mouse button. The actions that the right-clicking produce completely depend on the programmer.

When writing code for the right-click button, you will have to find out what button was clicked, and then write code accordingly.

1.5.2.5 The Focus Events

Microsoft Windows operating systems allow you to work on more that one application at the same time. They also allow you to work on many forms as the computer can handle. But only one application can receive instructions at a given time. For example, although you can edit text on a word processor while a spreadsheet is running in the background, you can only perform one action at a time. You have the ability to display the desired application when needed. This applies to applications.

Many dialog boxes have more than one input control, such as the Font dialog box we used earlier. Although all these controls are available, you can work from only one control at a time.

If many applications are running on your computer while you are working, the program that is currently being edited or receiving input from you is said to have focus. If you have two forms, you can open both of them but at a given time, you can work on only one of them. On a form that is equipped with many controls, only one control can be changed at a time; such a control is said to have focus.

The application or the form that has focus usually has its title bar with the active window color as set in Control Panel. In a form with many controls, the one that has focus will usually have a cursor or a dotted line around its selection.

When an application, a form, or a control has focus, Microsoft Visual Basic applies the GotFocus event. If the focus shifts to another application, form, or control, Microsoft applies the LostFocus to the same component.

1.5.2.6 Launching and Loading A Program

Your computer is filled with a lot of programs, some of which you use all the time, some of which you use some time to time, and some of which you probably never or rarely use. Since the computer can't predict what you want to do, it keeps all these programs in a storage area called the hard drive. They simply stay there and wait. When

you want to use one of these programs, you ask the computer to bring it to you. There is another, temporary, storage area in your computer called the memory (RAM). This is where the computer puts the programs you are using currently. When, you decide to use a program, the computer brings it up. When you have finished using the program, the computer puts it back into the hard drive. Of course, the computer can put as many programs as possible into the memory (or as many as the capacities of the computer allow it).

To use a program, you have to "Load" it into memory (the computer will do it for you). And to load a program you have to select and start it. That's why you need to find it and...

When a program starts, it is said to be launched. Visual Basic considers that the program is Opening. It takes just a few seconds for a program to launch or open. Some of them display a "Splash Screen" while they are launching. After the program has been launched, it is said to be Loaded. Once a program is loaded, it is said to be running. Actually, loaded and running would mean the same thing, especially in Visual Basic.

1.6 Variables and Data Types

When you write a program as a time sheet, you may decide that a user will type her weekly hours in one box and her salary in another box; then another box will display her weekly salary. When you are designing the program, you cannot predict the names of the people who will be using the program and you definitely cannot know the weekly hours they will get week after week. What you have to do is ask the computer to create temporary storage areas that one user can use while the program is running. If that box can be used to store a salary, when another user is using the same program, that box should be ready to receive new inputs, new salary for that other user.

The computer memory is made of small storage areas used to hold the things that a program needs while it is running. As a programmer, you specify these things, or you provide them to the computer; the computer then puts them in these storage areas. When you need one of them, you let the computer know. The machine located it and makes it available to you to use as you see fit.

A variable is a value you are ask the computer to store in its memory while the program is running.

1.6.1 Using a Variable

As stated already, a variable is an area of computer memory you use in your program. To use a variable, you must give it a name. There are rules you should, and usually must, follow when naming your variables. The name of a variable:

Must begin with a letter

Cannot have a period (remember that we use the period to set a property; in other words the period is an operator)

Can have up to 255 characters. Please, just because it is allowed, don't use 255 characters.

Must be unique inside of the procedure or the module it is used in (we will learn what a module is)

Once a variable has a name, you can use it as you see fit. For example, you can assign it a value and then use the variable in your program as if it represented that value

1.6.2 Variable Declaration

Unlike languages referred to as strongly typed, Visual Basic is so flexible you can use any variable just by specifying its name. When you provide this name, the computer directly creates an area in memory for it. Based on this, consider the followingcodesection:

```
Private Sub Form_Click()
    SameColor = vbBlue
    SomeColor = vbRed
    SumColor = vbRed
    BackColor = SameColor
End Sub

Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
    SameColor = vbBlue
    SomeColor = vbRed
    SumColor = vbGreen
```

```
    BackColor = SumColor
End Sub

Private Sub Form_Load()
    SameColor = vbBlue
    SomeColor = vbRed
    SumColor = vbGreen
    BackColor = SomeColor
End Sub
```

If you execute this program, when the form displays, it would be painted in red. If the user clicks the form, it would be painted in blue. If the user presses a key, the form would be painted in green. There is some confusion in the program. It uses a variable that seems to have a name but initialize three times with different colors. Visual Basic allows you to directly use any name for a variable as you see fit. Fortunately, to eliminate the possibility of this confusion, you can first let Visual Basic know that you will be using a certain variable. Informing Visual Basic about a variable prior to using that variable is referred to as declaring a variable. When a variable has been declared, just like the variable not declared, the computer

reserves an area of memory for it.

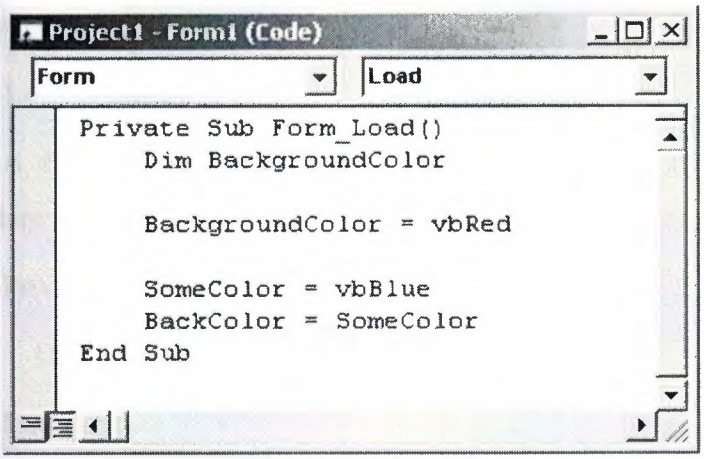
To declare a variable, type the **Dim** keyword, like this:

```
Dim
```

On the right side of Dim, you must type a name for the variable, following the same rules we reviewed above. Here is an example of declaring and using a variable:

```
Private Sub Form_Load()
    Dim BackgroundColor
    BackgroundColor = vbRed
    BackColor = BackgroundColor
End Sub
```

Declaring a variable simply communicates to Visual Basic the name of that variable. You can still use a mix of declared and not-declared variable. This is demonstrated in the following event:



Once again, the compiler believes that you are using two variables; one is called BackgroundColor and the other is called SomeColor. This can still create a great deal of confusion because you may be trying to use the same variable referred to twice. The solution to this possible confusion is to tell Visual Basic that a variable cannot be used if it has not been primarily declared. To communicate this, on top of each file you use in the Code Editor, type

Option Explicit

This can also be done automatically for each file by checking the Require Variable Declaration in the Options dialog box.

1.6.3.Introduction to Data Types

When you decide to use a variable, you are in fact asking the computer to use a certain amount of space to hold that variable. Since different variables will be used for different purposes, you should specify the kind of variable you intend to use, then the computer will figure out how much space is needed for a particular variable. Each variable you use will utilize a certain amount of space in the computer's memory.

Before declaring or using a variable, first decide what kind of role that variable will play in your program. Different variables are meant for different situations. The kind of variable you want to use is referred to as a data type. To specify the kind of variable you want to use, you type the **As** keyword on the right side of the variable's name. The formula to declare such a variable is:

Dim *VariableName* **As** *DataType*

Once you know what kind of variable you will need, choose the appropriate data type. Data types are organized in categories such as numbers, characters, or other objects.

1.6.3.1. String

A string is an empty text, a letter, a word or a group of words considered. To declare a string variable, use the String data type. Here is an example:

```
Private Sub Form_Load()
    Dim CountryName As String
End Sub
```

After declaring the variable, you can initialize. If you want its area of memory to be empty, you can assign it two double-quotes. Here is an example:

```
Private Sub Form_Load()
    Dim CountryName As String
    CountryName = ""
End Sub
```

If you want to store something in the memory space allocated to the variable, assign it a word or group of words included between double-quotes. Here is an example:

```
Private Sub Form_Load()
    Dim CountryName As String
    CountryName = "Great Britain"
     End Sub
```

You can also initialize a string variable with another.

1.6.3.2. Boolean

A Boolean variable is one whose value can be only either True or False. To declare such a variable, use the Boolean keyword. Here is an example:

```
Private Sub Form_Load()
```

```
    Dim IsMarried As Boolean
End Sub
```

After declaring a Boolean variable, you can initialize by assigning it either True or False. Here is an example:

```
Private Sub Form_Load()
    Dim IsMarried As Boolean
    IsMarried = False
End Sub
```

Like any other variable, after initializing the variable, it keeps its value until you change its value again.

1.6.3.3. Byte

A byte is a small natural positive number that ranges from 0 to 255. A variable of byte type can be used to hold small values such as a person's age, the number of fingers on an animal, etc.

To declare a variable for a small number, use the Byte keyword. Here is an example:

```
Private Sub Form_Load()
    Dim StudentAge As Byte
End Sub
```

1.6.3.4. Integer

An integer is a natural number larger than the **Byte**. It can hold a value between -32,768 and 32,767. Examples of such ranges are: the number of pages of a book.

To declare a variable of type integer, use the **Integer** keyword. Here is an example:

```
Private Sub Form_Load()
    Dim MusicTracks As Integer
End Sub
```

1.6.3.5. Long Integer

A long integer is a natural number whose value is between –2,147,483,648 and 2,147,483,642. Examples are the population of a city, the distance between places of different countries, the number of words of a book.

To declare a variable that can hold a very large natural number, use the **Long** keyword. Here is an example:

```
Private Sub Form_Load()
    Dim Population As Long
End Sub
```

1.7 Logical Comparisons

Sometimes while a person is using your program, you may need to check whether something is true or it is false. This type of operation is performed using operators referred to as comparison operators. Visual Basic provides various operators that can be used in appropriate types of comparisons

1.7.1. Boolean Variables

The Boolean data type is used to declare a variable whose value would be set as true (1) or false (0). To declare such a value, you use the **Boolean** keyword. The variable can then be initialized with a starting value. The Boolean constant is used to check that the state of a variable (or a function) is true or false. You can declare such a variable as:

```
dim GotThePassingGrade as Boolean;
```

Later in the program, for a student who got a failing grade, you can assign the other value, like this

```
GotThePassingGrade = False;
```

1.7.2 Logical Operators

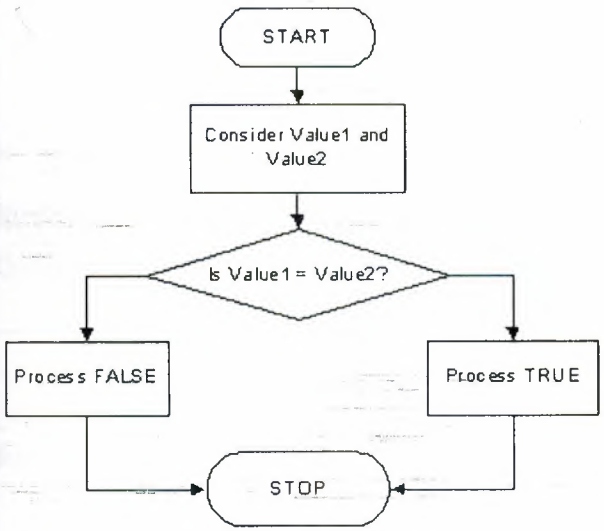
1.7.2.1. Equality =

To compare two values for equality, use the = operator. Its formula is:

Value1 = Value2

The equality operation is used to find out whether two variables (or one variable and a constant) hold the same value. From our formula, the compiler would compare the value of Value1 with that of Value2. If Value1 and Value2 hold the same value, the comparison produces a true result. If they are different, the comparison renders false or 0.

Figure Equity:



1.7.2.2. Logical Not

When a variable is declared and receives a value (this could be done through initialization or a change of value) in a program, it becomes alive. It can then participate in any necessary operation. The compiler keeps track of every variable that exists in the program being processed. When a variable is not being used or is not available for processing (in visual programming, it would be considered as disabled) to make a variable (temporarily) unusable, you can nullify its value. To render a variable unavailable during the evolution of a program, apply the logical not operator which is **Not**. Its formula is:

Not *Value*

There are two main ways you can use the logical **Not** operator. As we will learn when studying conditional statements, the most classic way of using the logical **Not** operator is to check the state of a variable.

To nullify a variable, you can write **Not** to its left. When used like that, you can display its value. You can even assign it to another variable. Here is an example:

```
Private Sub Command1_Click()
  Dim value1 As Integer
  Dim value2 As Boolean
  value1 = 250
  value2 = Not value1
   Text1.Text = value2
End Sub
```

When a variable holds a value, it is "alive". To make it not available, you can "not" it. When a variable has been "notted", its logical value has changed. Therefore, you can inverse the logical value of a variable by "notting" or not "notting" it. This is done by typing Not to its left

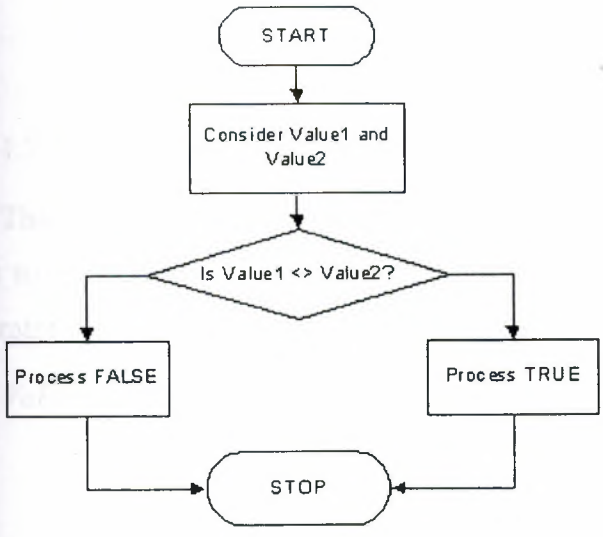
1.7.2.3. Inequality <>

Visual Basic provides an operator used to compare two values for inequality. Its formula is:

Value1 <> Value2

is a binary operator (like all logical operators except the logical **Not**, which is a unary operator) that is used to compare two values. The values can come from two variables as in Variable1 <> Variable2. Upon comparing the values, if both variables hold different values, the comparison produces a true or positive value. Otherwise, the comparison renders false or a null value.

Figure Inequity:



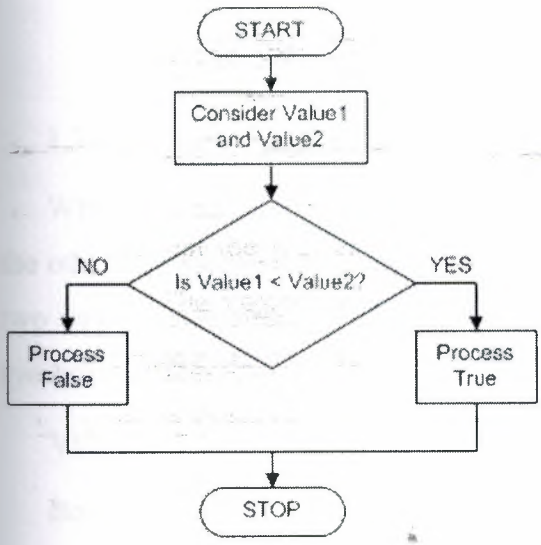
1.7.2.4. Less Than <

To find out whether one value is lower than another, use the < operator. Its formula is:

Value1 < *Value2*

The value held by Value1 is compared to that of Value2. As it would be done with other operations, the comparison can be made between two variables, as in Variable1 < Variable2. If the value held by Variable1 is lower than that of Variable2, the comparison produces a true or positive result.

Figure Less:



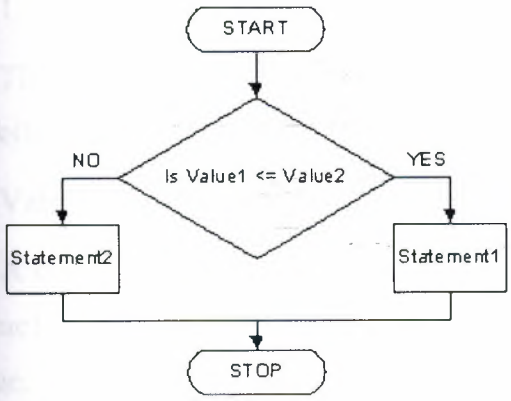
1.7.2.5 Less Than Or Equal <=

The previous two operations can be combined to compare two values. This allows you to know if two values are the same or if the first is less than the second. The operator used is <= and its formula is:

Value1 <= *Value2*

The <= operation performs a comparison as any of the last two. If both Value1 and Value2 hold the same value, the result is true or not null. If the left operand, in this case *Value1*, holds a value lower than the second operand, in this case *Value2*, the result is still true.

Figure Less or Equity:



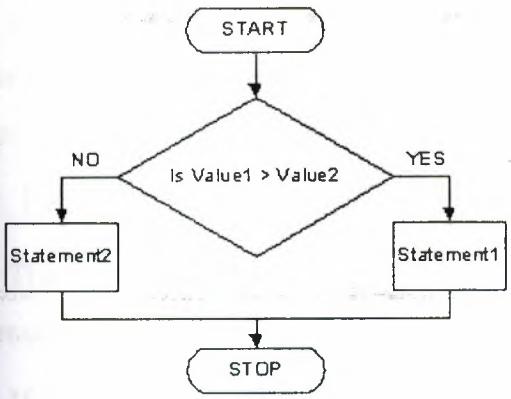
1.7.2.6 Greater Than >

When two values of the same type are distinct, one of them is usually higher than the other. Visual Basic provides a logical operator that allows you to find out if one of two values is greater than the other. The operator used for this operation uses the > symbol. Its formula is:

Value1 > Value2

Both operands, in this case Value1 and Value2, can be variables or the left operand can be a variable while the right operand is a constant. If the value on the left of the > operator is greater than the value on the right side or a constant, the comparison produces a true or positive value. Otherwise, the comparison renders false or null. This can be illustrated as follows:

Figure Equity:



1.7.2.7 Greater Than or Equal >=

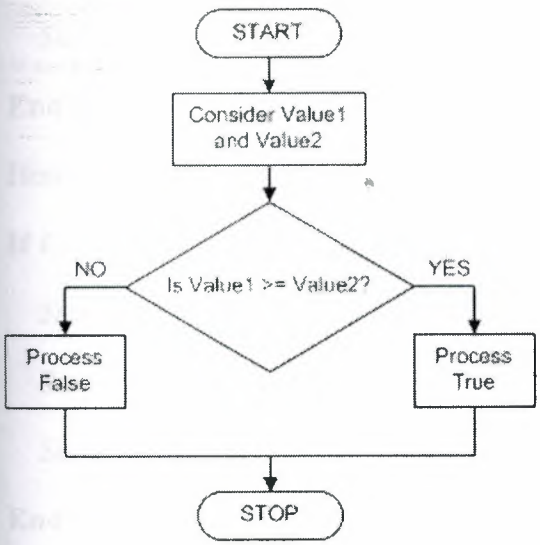
The greater than or the equality operators can be combined to produce an operator as follows: >=. This is the "greater than or equal to" operator. Its formula is:

Value1 >= Value2

A comparison is performed on both operands: Value1 and Value2. If the value of Value1 and that of Value2 are the same, the comparison produces a true or positive value. If the value of the left operand is greater than that of the right operand, the comparison produces true or positive also. If the value of the left operand is strictly less than the value of the right operand, the comparison produces a false or null result.

This can be illustrated as follows

Figure Greater or Equity:



Microsoft Visual Basic uses various conditional statements for almost any situation your computer can encounter. As the application developer, it is up to you to anticipate these situations and make your program act accordingly.

1.8.1 The If...Then Statement

The **If...Then** statement examines the truthfulness of an expression. Structurally, its formula is:

If *ConditionToCheck* **Then** *Statement*

Therefore, the program examines a condition, in this case *ConditionToCheck*. This *ConditionToCheck* can be a simple expression or a combination of expressions. If the *ConditionToCheck* is true, then the program will execute the *Statement*.

There are two ways you can use the **If...Then** statement. If the conditional formula is short enough, you can write it on one line, like this:

If *ConditionToCheck* **Then** *Statement*

If there are many statements to execute as a truthful result of the condition, you should write the statements on alternate lines. Of course, you can use this technique even if the condition you are examining is short. In this case, one very important rule to keep is to terminate the conditional statement with **End If**. Here is an example:

If *ConditionToCheck* **Then**

Statement

End If

Here is another example:

If *Condition* **Then**

Statement1

Statement2

Statementn

End If

1.8.2 The If...Then...Else Statement

The **If...Then** statement offers only one alternative: to act if the condition is true. Whenever you would like to apply an alternate expression in case the condition is false, you can use the **If...Then...Else** statement. The formula of this statement is:

If *ConditionToCheck* **Then**

Statement1

Else

Statement2

End If

When this section of code is executed, if the *ConditionToCheck* is true, then the first statement, *Statement1*, is executed. If the *ConditionToCheck* is false, the second statement, in this case *Statement2*, is executed.

1.8.3 The If...Then...ElseIf Statement

The **If...Then...ElseIf** statement acts like the **If...Then...Else** expression, except that it offers as many choices as necessary. The formula is:

If *Condition1* **Then**

Statement1

ElseIf *Condition2* **Then**

Statement2

ElseIf *Conditionk* **Then**

Statementk

End If

The program will first examine *Condition1*. If *Condition1* is true, the program will execute *Statment1* and stop examining conditions. If *Condition1* is false, the program will examine *Condition2* and act accordingly. Whenever a condition is false, the program will continue examining the conditions until it finds one. Once a true condition has been found and its statement executed, the program will terminate the conditional examination at **End If**.

There is still a possibility that none of the stated conditions is true. In this case, you should provide a "catch all" condition. This is done with a last **Else** section. The **Else** section must be the last in the list of conditions and would act if none of the primary conditions is true. The formula to use would be:

If *Condition1* **Then**

Statement1

ElseIf *Condition2* **Then**

Statement2

ElseIf *Conditionk* **Then**

Statementk

Else

CatchAllStatement

End If

1.8.4 The Select Case Statement

If you have a large number of conditions to examine, the **If...Then...Else** will go through each one of them. Visual Basic offers the alternative of jumping to the statement that applies to the state of the condition.

The formula of the **Select Case** is:

Select Case *Expression*

Case *Expression1*

Statement1

Case *Expression2*

Statement2

Case *Expressionk*

Statementk

End Select

The *Expression* will examined and evaluated once. Then it will compare the result of this examination with the *Expression* of each case. Once it finds one that matches, it would

execute the corresponding *Statement*.

If you anticipate that there could be no match between the *Expression* and one of the *Expressions*, you can use a **Case Else** statement at the end of the list. The statement would then look like this:

Select Case *Expression*

Case *Expression1*

Statement1

Case *Expression2*

Statement2

Case *Expressionk*

Statementk

Case Else

Statementk

End Select

1.8.5 The Do...While Loop Statement

The formula of the **Do... While** loop is:

Do While *Condition*

Statement(s)

Loop

This expression examines the *Condition*. If the condition is true, then it executes the *Statement* or statements. After executing the statement(s), it goes back to examine the *Condition*. AS LONG AS the *Condition* is true, the *Statement* will be executed and the *Condition* will be tested again. If the *Condition* is false or once the condition becomes false, the statement will not be executed and the the program will move on. As you may guess already, the Condition must provide a way for it to be true and to be false.

1.8.6. The Do...Loop...While Statement

Since the **Do...While** statement tests the *Condition* first before executing the *Statement*, sometimes you will want the program to execute the *Statement* first, then go back and test the *Condition*. Visual Basic offers a reverse to the formula, which is:

Do

Statement(s)

Loop While *Condition*

In this case, the Statement or Statements will be executed first. Then the *Condition* will be tested. If the *Condition* is true, the program will execute the *Statement* again. The program will continue this examination-execution as long as the *Condition* is true. The big difference here is that even if the Condition is false, the program will have executed the *Condition* at least once.

1.8.7. The Do...Until...Loop Statement

An alternative to the **Do...While** loop is the **Do...Until** loop. Its formula is:

Do **Until** *Condition*

Statement(s)

Loop

This loop will first examine the *Condition*, instead of examining whether the *Condition* is true, it will test whether the *Condition* is false.

1.8.8. The Do...Loop...Until Statement

An alternative to the **Do...Until...loop** consists of executing the the *Statement* first. The formula is:

Do

Statement(s)

Loop Until *Condition*

This express executes the *Statement* first. After executing the *Statement*, it would examine the *Condition*. If the *Condition* is False, then it would go back and execute the

Statement again and re-check the *Condition*. Once the *Condition* becomes true, it would stop and move on; but as long as the *Condition* is False, the *Statement* would be executed.

1.8.9. The For...To...Next Loop

One of the loop counters you can use is **For...To...Next**. Its formula is:

For *Counter* = *Start* **To** *End*

Statement(s)

Next

Used for counting, the expression begins counting at the *Start* point. Then it examines whether the current value (after starting to count) is greater than *End*. If that's the case, it then executes the *Statement*(s). Next, it increments the value of *Counter* by 1 and examines the condition again. This process goes on until the value of *Counter* becomes equal to the *End* value. Once this condition is reache, the looping stops.

1.9 Built-In Functions

A procedure is referred to as "built-in" if it shipped with your language. To make your job a little easier, Microsoft Visual Basic comes equipped with many functions that you can use right away in your program. Based on this, before creating your own function, first check whether the functionality you are looking is already implementing in one of the available procedures because those that ship with Visual Basic are highly reliable and should be preferred.

Before using a built-in procedure, you must of course be familiar with it. This comes either by consulting the documentation or by experience. This means that you must know its name, its argument(s), its return value, and its role.

1.9.1 Conversion Functions

The first action you should take when dealing with a value or an expression is to convert it to the appropriate type. There are various conversion functions adapted to the different possible kinds of values. The general syntax of the conversion functions is:

ReturnType = FunctionName(Expression)

The *Expression* could be of any kind. For example, it could be a string or value the user would have entered in a form. It could also be the result of a calculation performed

on another field or function. The conversion function would take such a value, string, or expression and attempt to convert it. If the conversion is successful, the function would return a new value that is of the type specified by the *ReturnType* in our syntax.

The conversion functions are as follows

Function		
CBool	Boolean	Converts an expression into a Boolean value
CByte	Byte	Converts an expression into Byte number
CDate	Date	Converts and expression into a date or time value
CDbl	Double	Converts an expression into a flowing-point (decimal) number
CInt	Integer	Converts an expression into an integer (natural) number
CCur	Currency	Converts an expression into a currency (monetary) value
CLng	Long	Converts an expression into a long integer (a large natural) number
CSng	Single	Converts an expression into a flowing-point (decimal) number
CStr	String	Converts an expression into a string

1.9.2 String-Based Functions

A string-based function is one that deals with functions; either it manipulates them or returns them. Microsoft Visual Basic allows you to be specific about the return value you are expecting. Some of the functions you will be using can be configured to return exactly a string. Such functions use the $ suffix that states it clearly.

1.9.2.1 Message Boxes

A message box is a special form used to display a piece of information to the user. As opposed to a regular form, the user cannot type anything on the box. There are usually two kinds of dialog boxes you will create: one that simply displays information and one that expects the user to make a decision.

To create a message box, you can use the **MsgBox** function. There are two techniques to use it. To display a simple message with just an OK button, use the **MsgBox** method whose syntax is

MsgBox *Message*

The parameter, Message, is the string to present to the user. As a normal, it should be passed in double-quotes. Here is an example:

```
Private Sub Form_Load()
    MsgBox "Welcome to the wonderful world of Microsoft Visual Basic"
End Sub
```

When the above version of the the MsgBox function is used, a rectangular form (we will learn later on that this type of form is called a dialog box) is presented to the user, display a string message and an OK button:

Another version of the MsgBox function allows you to present a message that asks a question to the user, expecting a decision. This version displays a more informative prompt with more than one button. The user makes a decision by clicking one of the presented buttons. After the user has clicked a button, you can then retrieve the result and use it as you see fit. The syntax of this version is:

MsgBox *Message*, [*Buttons*], [*Title*], [*HelpFile*], [*Context*]

The *Message* argument is the string that the user will see displaying on the message box. As a string, you can display it in double quotes. You can also create it from other pieces of strings. The *Message* argument can be made of up to 1024 characters. To display the *Message* on multiple lines, you can use either the constant **vbCrLf** or the combination **Chr(10) & Chr(13)** between any two strings.

Besides the *Message* parameter, this version allows you to display more than one button. If you don't need to, you don't have to specify the buttons. If you don't, the message box would appear with only an OK button. Otherwise, you can specify what buttons to display. This is done using the *Buttons* argument. There are different kinds of buttons available and Visual Basic recognizes them by a numeric value assigned to each. The buttons are

Button	Value	Display
vbOKOnly	0	OK
vbOKCancel	1	OK Cancel
vbAbortRetryIgnore	2	Abort Retry Ignore
vbYesNoCancel	3	Yes No Cancel
vbYesNo	4	Yes No
vbRetryCancel	5	Retry Cancel

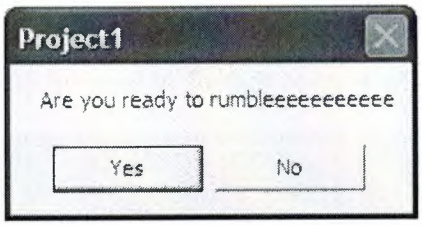
Here is an example of a message box that display a Yes and a No buttons:

```
Private Sub Form_Load()
```

MsgBox "Are you ready to rumbleeeeeeeeeee", **vbYesNo**

End Sub

This would produce:



When a message box displays more than one button, one of the buttons usually has a thick button. That button is also called the default button. If the user presses Enter upon reading the message, the compiler would behave as if the default button was clicked. There are some buttons that are set automatically as default when you create the message box. If you don't like the set button to be the default, you can specify which one you prefer as default. To do that combine a second value with one of the above values for the buttons. You can set the default argument using the following table

Option	Value
vbDefaultButton1	0
vbDefaultButton2	256
vbDefaultButton3	512
vbDefaultButton4	768

To combine one of these values with one of the buttons, use the OR operator between them. Here is an example:

Private Sub Form_Load()

MsgBox "Are you ready to rumbleeeeeeeeeee", **vbYesNo Or vbDefaultButton2**

End Sub

This would produce:

These additional buttons can be used to further control what the user can do:

Constant	Value	Description
vbApplicationModal	0	
vbSystemModal	4096	

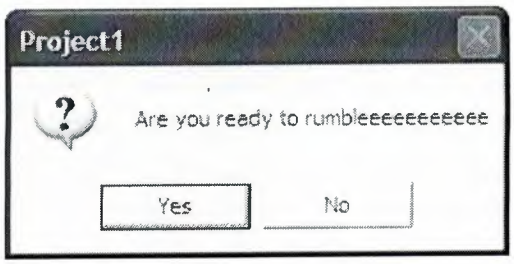
Besides the message and the button(s), you can also display a friendly icon on the message box. To do that, combine the button value with one of the following:

Icon	Value	Description
vbCritical	16	
vbQuestion	32	
vbExclamation	48	
vbInformation	64	

Here is an example:

```
Private Sub Form_Load()
    MsgBox "Are you ready to rumbleeeeeeeeeee", vbYesNo Or vbQuestion
End Sub
```

This would produce:



As you can see on the message boxes we have used so far, by default, a message box displays the name of the application it belongs to in its title bar. If you want, you can display your own title. This is done using the *Title* argument which is also called the caption of the message box. It is a string whose word or words you can enclose between parentheses or that you can get from a created string.

If your application is using a help file, you can specify this and let the message box use it. The *HelpFile* argument is a string that specifies the name of the help file, and the *Context* argument provides the number that corresponds to the appropriate help topic for the message box.

The way we have been using it so far, the **MsgBox** is called a method. If you want to use it as a function, that is, if you want it to return a value, you must call it as a function. In other words, its list of arguments must be included in parentheses. The above message can be created as follows:

```
Private Sub Form_Load()
    MsgBox("Are you ready to rumbleeeeeeeeeee", vbYesNo Or vbQuestion)
End Sub
```

When treated as a function, MsgBox returns a value. This value corresponds to the button the

user clicks on the message box. Depending on the buttons the message box is displaying, after the user has clicked, the **MsgBox** function can return one of the following values:

Button	Return	Value
OK	vbOK	1

Cancel	vbCancel	2
Abort	vbAbort	3
Retry	vbRetry	4
Ignore	vbIgnore	5
Yes	vbYes	6
No	vbNo	7

1.9.2.2 The Input Box

Like a message box, an input box is a (relatively) small form (in reality, it is a dialog box) that displays a message to the user. Unlike a message box, an input box presents a small text box that expects the user to enter a value. After using it, the user can either send the form with the new value or dismiss it without any change.

To create an input box, you can use the **InputBox** function procedure prompts the user to enter some information in a message box, and the function will return the content of that box.

1.9.2.3 The Character To ASCII Conversion

The **Chr** function is used to associate an entered character with its ASCII character equivalent. It could be used to convert a number to a character. It could also be used to break a line in a long expression. The syntax of the **Chr** function is:

Chr(Number)

A combination of **Chr(13)** and **Chr(10)** would break a line in an expression.

1.9.2.4 Case Conversion

If you are presented with a string or an expression whose cases must be the same, you can convert all of its characters in either uppercase or lowercase.

To convert a character, a string or an expression to uppercase, you can call the **UCase** or the **UCase$** function. These functions take one argument as the string or expression to be considered. The syntaxes are:

Function UCase(Letter As Char) As Char

Function UCase(Expression As String) As String

The first version receives one character as argument. If the character is already in uppercase, it would be return the same. If the character is not a readable character, no conversion would happen and the function would return it. If the character is in lowercase, it would be converted to uppercase and the function would then return the uppercase equivalent.

The second version considers the argument supplied as a string. Any letter that is in lowercase in the string would be converted to uppercase. Any letter that is in uppercase would be preserved and would not be changed. Any non-alphabetic character in the string would be kept "as is".

1.9.3 Logical Functions

1.9.3.1 Is it Empty?

A logical function is one that checks whether an expression is true or false and then return a Boolean value.

The **IsEmpty** function check whether a field is empty. Its syntax is:

IsEmpty(*Expression*)

In this case, the *Expression* argument will be checked. If it is empty, the IsEmpty function returns True. If the expression or field is not empty, that is, if it contains something, the function returns False.

1.9.3.2 Is it Null?

Another problem you may encounter when involving an operation or the contents of a control is whether it has never contained a value. This operation is sometimes confused with that of checking whether a field is empty. Here is the difference (it is important to understand this because it is used in many other environments):

Imagine a text box control is used for first name and the field displays Paul. If the user comes to that record, the field is not empty, it already contains a name, which is

this case is Paul. If the user clicks in the field and deletes Paul, the field becomes empty. It is not null

Imagine a field is used for first name. If the user comes to a new record, the field for the first name may be empty (if you did not give it a default value). In this case, the field is Null: it is not empty because it has never contained anything. If the user types a name, and then deletes it, the field is not considered Null anymore: it has become empty

To check whether an expression or the value of a control is null, you can call the **IsNull()** function. Its syntax is:

IsNull(*Expression*)

Also used on fields, the IsNull() function checks the state of a field (remember, this functions does not check whether a field is empty or not; it checks if the field has ever contained a value). If the field it null, this function returns True. If the field is not null, this function returns False.

1.9.4 Date and Time Functions

1.9.4.1 Current Data and Time

Microsoft Visual Basic provides various functions to perform date and time related operations. These functions allow you to add dates or times, find the difference between dates or times, or add constant values to dates or times.

The current date is represented by a function called **Date**. The **Date()** function is used to get the system date of the computer. You can use it to display today's date, provided your computer has the correct date.

The current time of the computer is represented by a function called **Time**. The **Time()** function is used to get the system time of the computer.

The **Date()** and **Time()** functions can be combined and are represented by a function called **Now**.

1.9.4.2 Day - Month – Year

The **Day** function is used to get the numeric value that represents a day in the month. It ranges from 1 to 31 included.

The formula of the Day function is Day(DateValue)

The **Month** function displays the numeric month of a date. It ranges from 1 to 12 included.

The formula of the Month function is Month(DateValue)

The **Year** function returns the numerical year of a date.

The formula is the Year function is Year(DateValue)

1.9.4.3 **Adding a Date**

The **DateAdd** function is used to add a date value to another date. It can be used to add a number of days, weeks, months, or years to another date. The formula for the **DateAdd** function is

DateAdd(*Interval*, *Number*, *date*)

Required, the *Interval* argument specifies the kind of value you want as a result. This argument will be enclosed between double quotes and can have one of the following values:

Interval	Used To Get
s	Second
n	Minute
h	Hour
w	Numeric Weekday
ww	Week of the Year
d	Day
y	Numeric Day of the Year
m	Month

q	Quarter
yyyy	Year

Required also, the *Number* argument specifies the number of units you want to add. If you set it as positive, its value will be added. On the other hand, if you want to subtract, make it negative.

The number represents the units of the *Interval* argument you want to add.

The *date* argument is the date to which you want to add the number.

1.9.4.4 Subtracting a Date

The **DateDiff** function is used to find the difference between two date or time values. It allows you to find the number of seconds, minutes, hours, days, weeks, months, or years from two valid date or time values. The **DateDiff** function takes 5 arguments, 3 are required and 2 are optional.

The formula of the function is

DateDiff(*Interval*, *Date1*, *Date2*, *Option1*, *Option2*)

Required, the *Interval* argument specifies what kind of value you want as a result. This argument will be enclosed between double quotes and can have one of the following values:

Interval	Used To Get
s	Second
n	Minute
h	Hour
w	Numeric Weekday
ww	Week of the Year
d	Day
y	Numeric Day of the

	Year
m	Month
q	Quarter
yyyy	Year

Required also, the *Date1* and Date2 argument specify the date or time values that will be used when performing the operation.

By default, the days of a week are counted starting on Sunday. If you want to start counting those days on another day, supply the *Option1* argument using one of the following values: **vbSunday**, **vbMonday**, **vbTuesday**, **vbWednesday**, **vbThursday**, **vbFriday**, **vbSaturday**. There are other variances to that argument.

If your calculation involves weeks or finding the number of weeks, by default, the weeks are counted starting January 1st. If you want to count your weeks starting at a different date, use the *Option2* argument to specify where the program should start.

CHAPTER TWO: WHAT IS ACCESS AND HOW TO DO TABLE

2.1. Microsoft Access Database

Microsoft Access is a powerful program to create and manage your databases. It has many built in features to assist you in constructing and viewing your information. Access is much more involved and is a more genuine database application than other programs such as Microsoft Works.

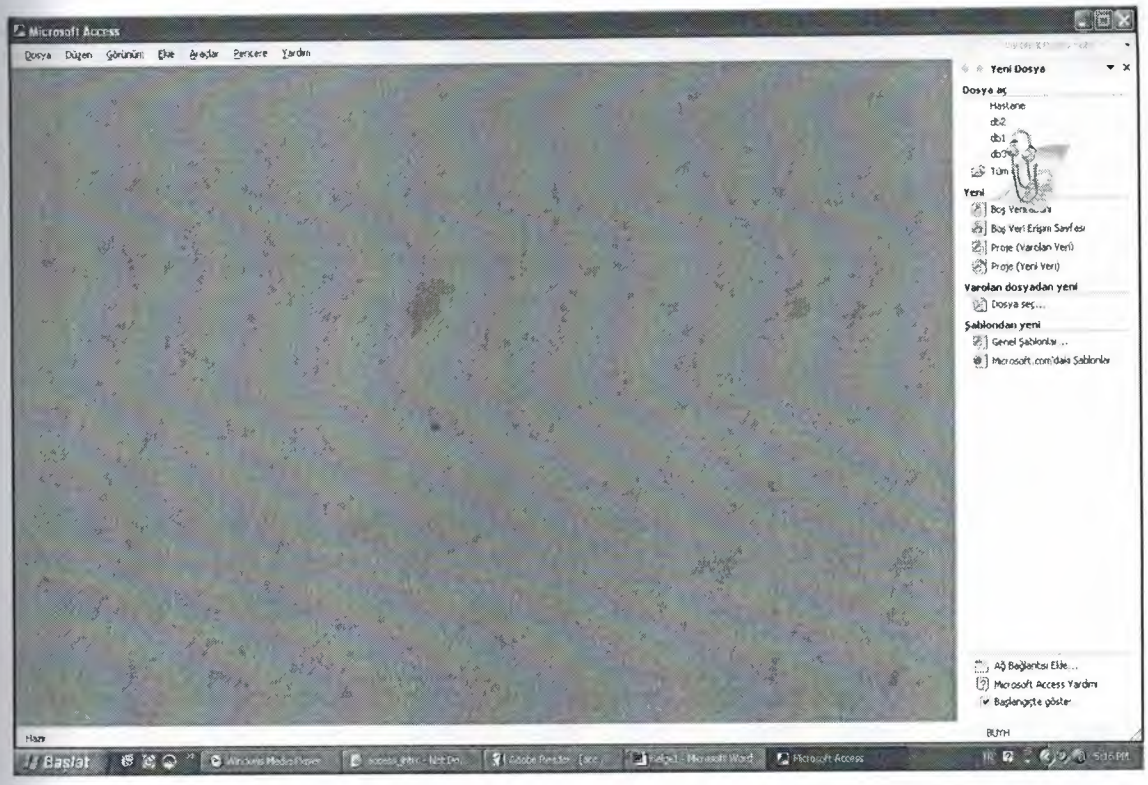
2.2. How To Form a Table in Database

A database is a collection of information that's related. Access allow you to manage your information in one database file. Within Access there are four major areas: Tables, Queries, Forms and Reports

- Tables store your data in your database
- Queries ask questions about information stored in your tables
- Forms allow you to view data stored in your tables
- Reports allow you to print data based on queries/tables that you have Created

Creating a Database

1)StartAccess

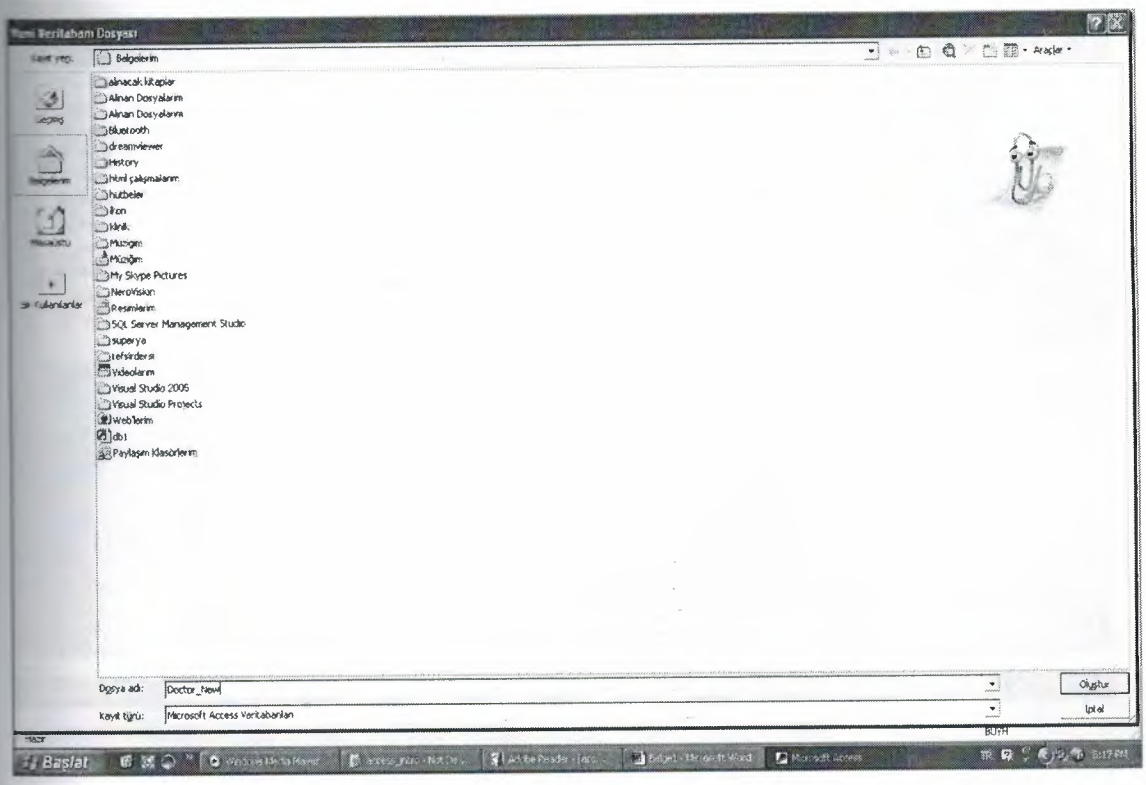


2) In the task pane, select Blank Database

3) In the File New Database dialog box, select the location where you want to

store the database

4) Type a name for the database file at the bottom of the dialog box



5) Click Create

Access will display the database window.

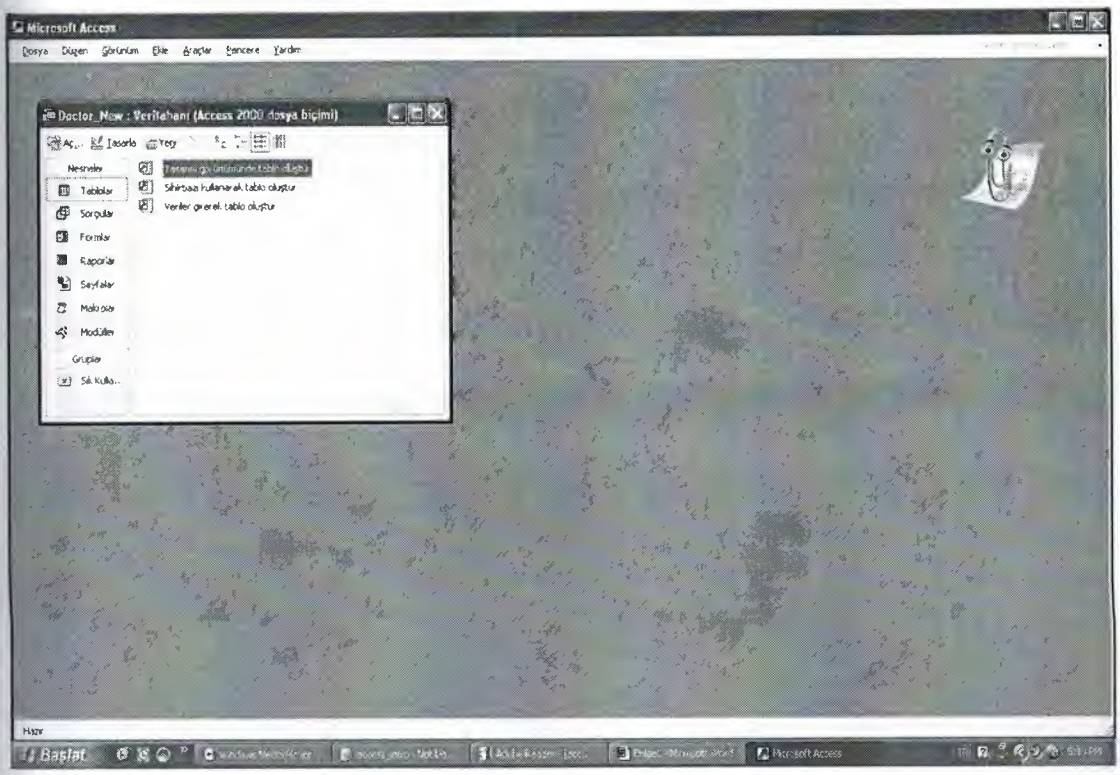
Creating a Table

A table is a collection of data about a specific topic, such as employee information. products or customers.

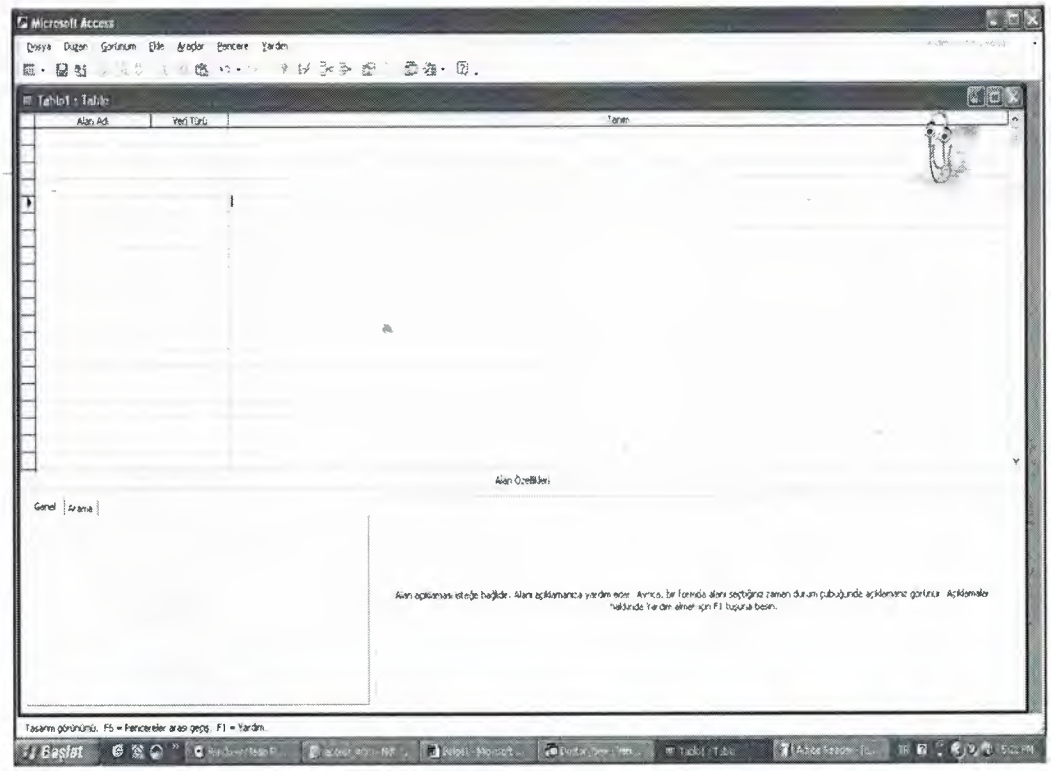
To Create a Table in Design View:

1) Click the Tables object button

2) Double-click Create table in Design view



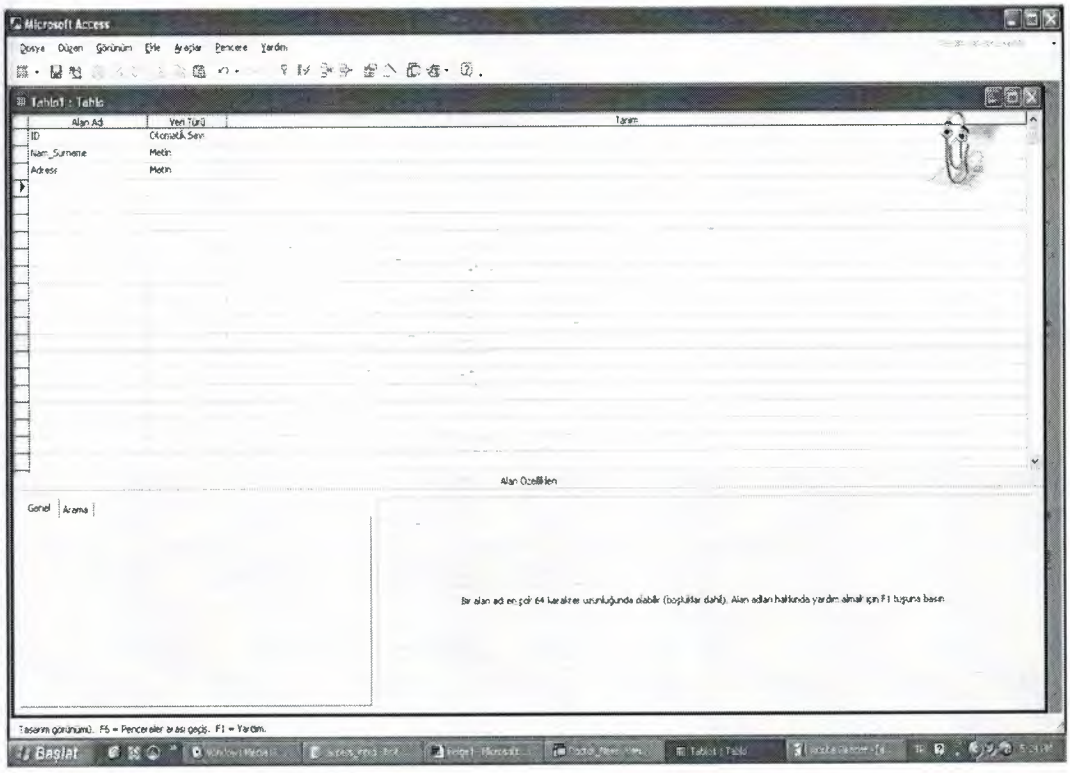
Below is a picture of the Table Design screen as it will appear:



3) Enter a field name in the first field name cell, then press the Tab key to enter the Data Type column (Limited to 64 characters per field)

4) Click the down-arrow at the right end of the Data Type cell, and then select an appropriate data type for the field

Note: The order that you enter the field names is the order the fields will appear in the table

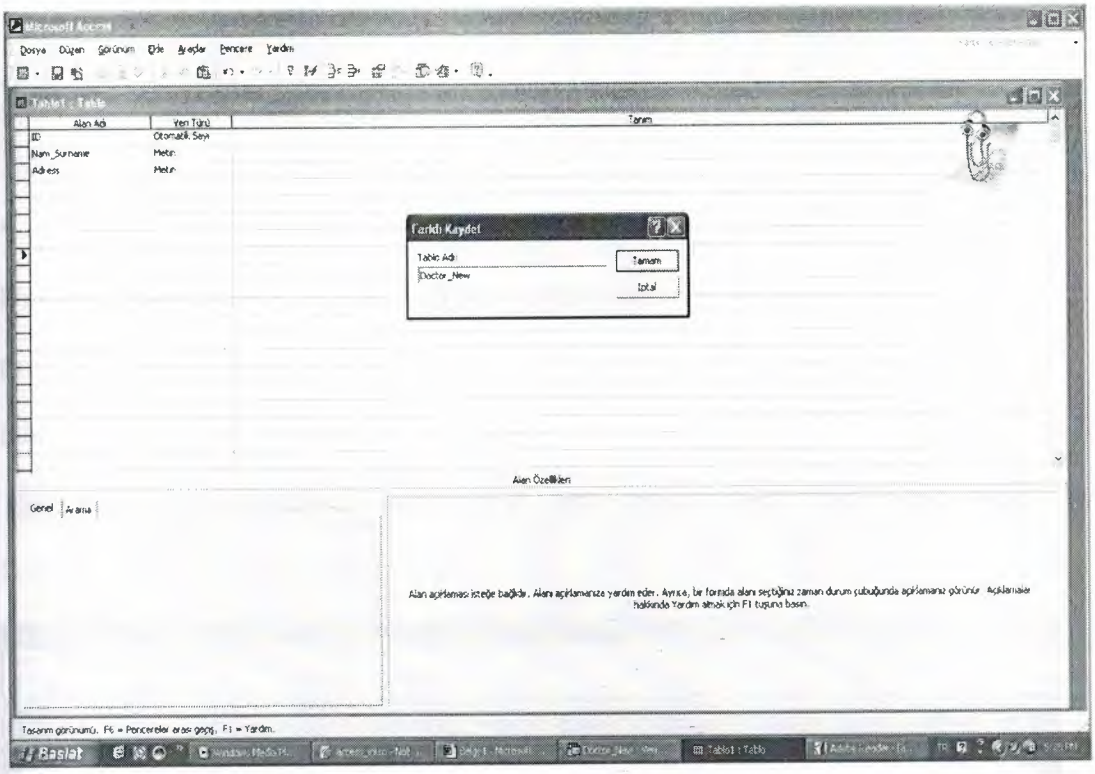


5) Continue until all fields are inserted

To Save the Table:

1) Click the Save icon on the toolbar

2) Enter a name for the table



3) Click OK

Access will ask you if you would like it to create a primary key. Click No to letting Access create a primary key field. This will be explained later.

Setting a Primary Key

The primary key is the unique identifier for each record in a table. Access will not allow duplicate entries in a primary key field.

To Set a Primary Key:

1. Double click on the table if necessary to open it

2. In Design View, position your cursor in the field you wish to set as the primary key

3. Click the primary key button on the tool bar

To Switch Between Design View and Datasheet View:

1) Double click on the table

2) Select View Menu, Design View

Or

View Menu, Datasheet View

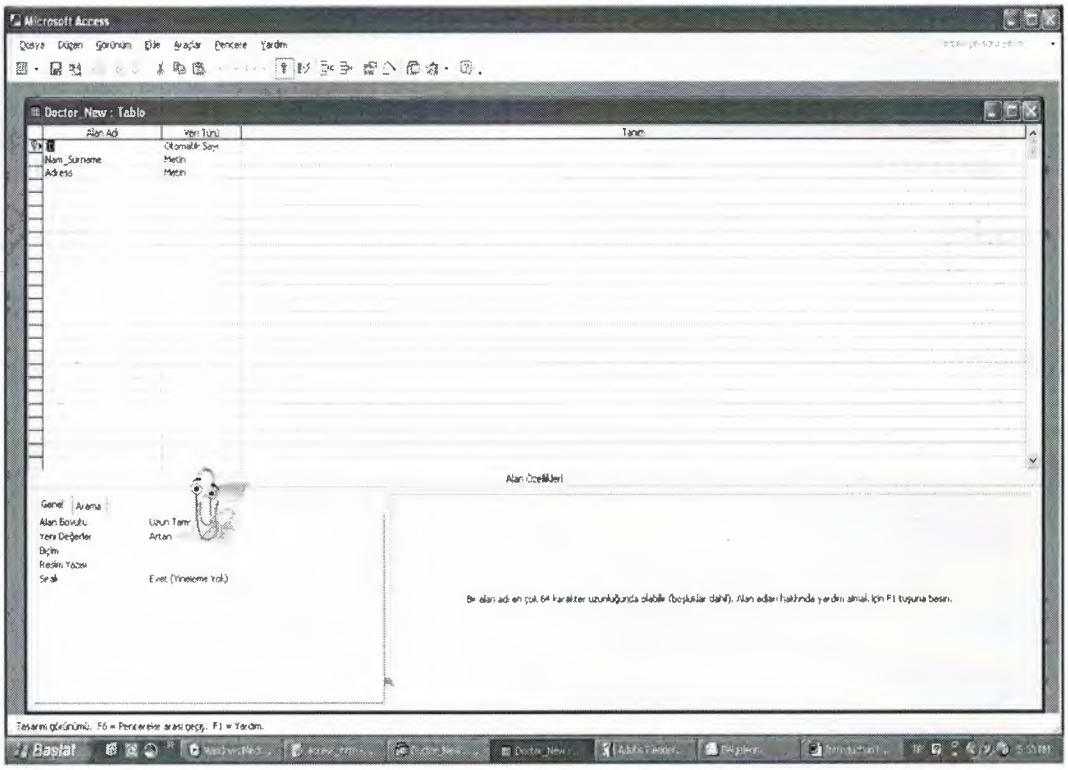
There are 2 basic views when you work in a table: Design View and Datasheet View:

•

Design View is used to set the data types, insert or delete fields, set the primary key

Datasheet View is used to enter the data for the records

The primary key symbol will appear in the gray row header at the left end of the field as seen in the picture below:



3) Save the table

Input Masks

An input mask is used to pre-format a field to "look/act" a certain way when a user inputs data.

Example: social security number input mask automatically inserts the dash; phone numbers automatically inserts the parentheses and dashes

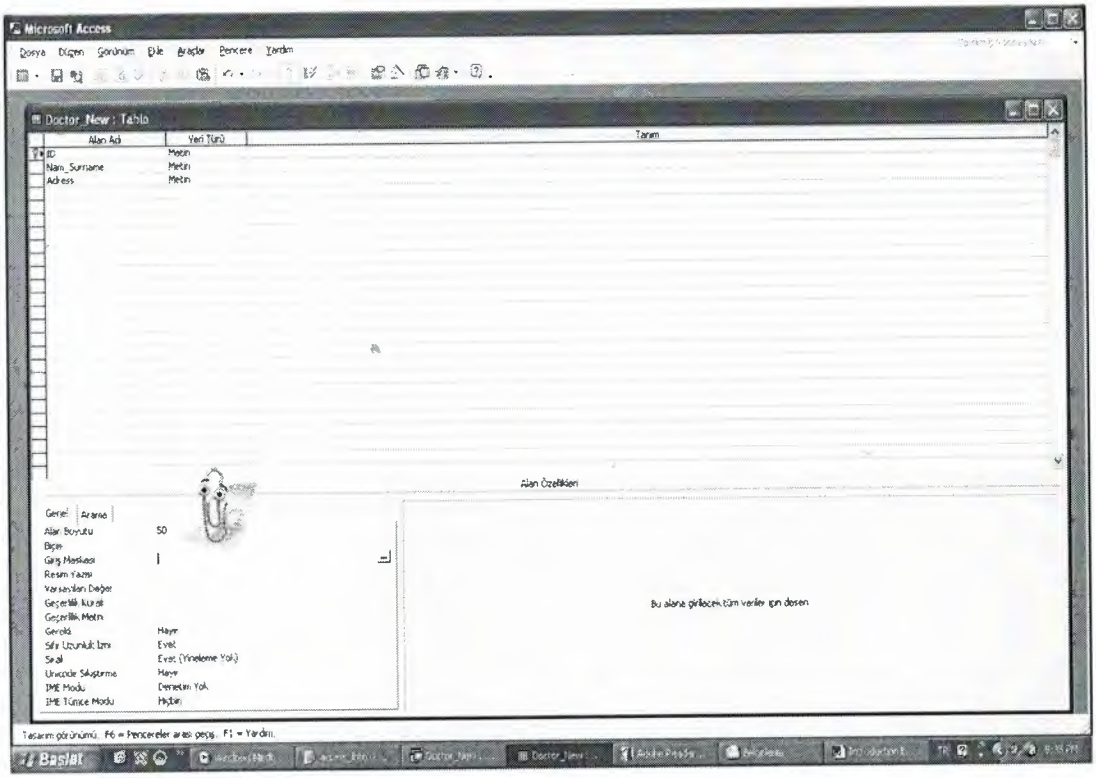
The input mask data can either be stored in the table or simply displayed and not stored.

To Create an Input Mask for a Field

1) Open a table in Design View

2) Click in a field for which you'd like to create an input mask

3) In the Field Properties section at the bottom of the screen, click in the Input Mask line and notice the Build button that appears at the right end of the line (see below):



Input Masks

An input mask is used to pre-format a field to "look/act" a certain way when a user inputs data.

Example: social security number input mask automatically inserts the dash; phone numbers automatically inserts the parentheses and dashes

The input mask data can either be stored in the table or simply displayed and not stored.

To Create an Input Mask for a Field

1) Open a table in Design View

2) Click in a field for which you'd like to create an input mask

3) In the Field Properties section at the bottom of the screen, click in the Input Mask line and notice the Build button that appears at the right end of the line (see below): 3) Save the table

Input Masks

An input mask is used to pre-format a field to "look/act" a certain way when a user inputs data.

Example: social security number input mask automatically inserts the dash; phone numbers automatically inserts the parentheses and dashes

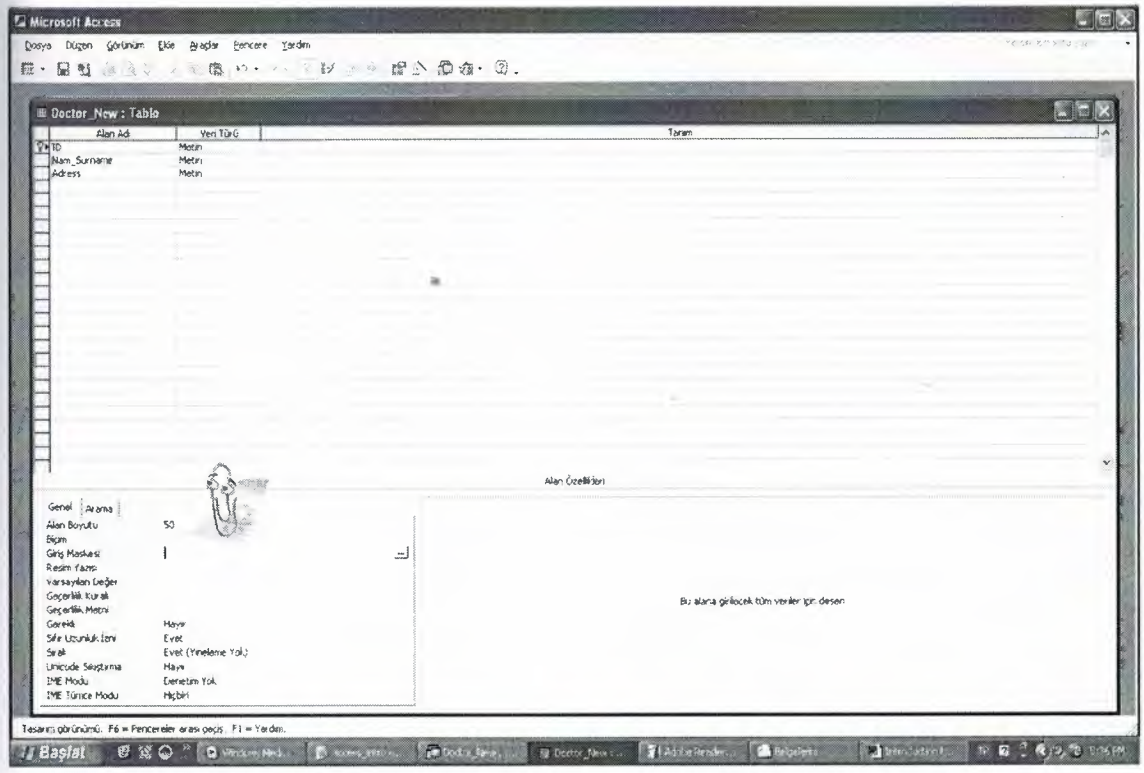
The input mask data can either be stored in the table or simply displayed and not stored.

To Create an Input Mask for a Field

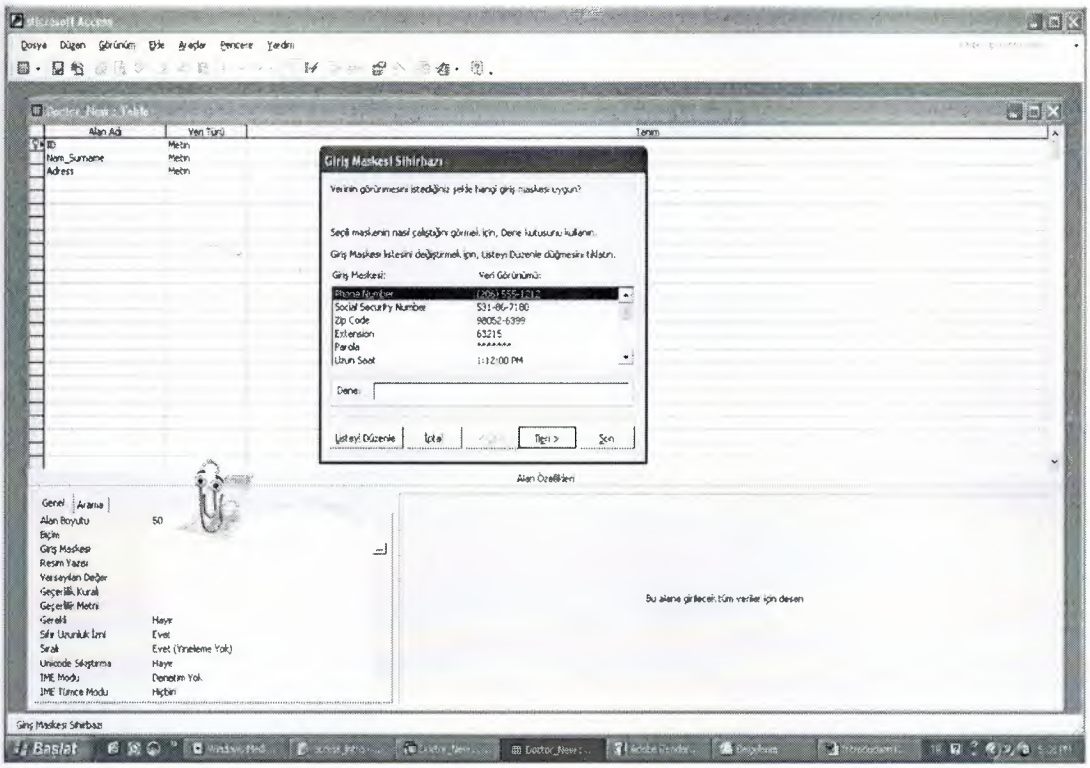
1) Open a table in Design View

2) Click in a field for which you'd like to create an input mask

3) In the Field Properties section at the bottom of the screen, click in the Input Mask line and notice the Build button that appears at the right end of the line (see below):



4) Click the Build button to start the Input Mask Wizard (shown below).



5) Select the appropriate input mask

6) Click Next

7) Click Next for additional screens on which you can set options for the input mask

8) Click Finish on the last screen of the input mask wizard

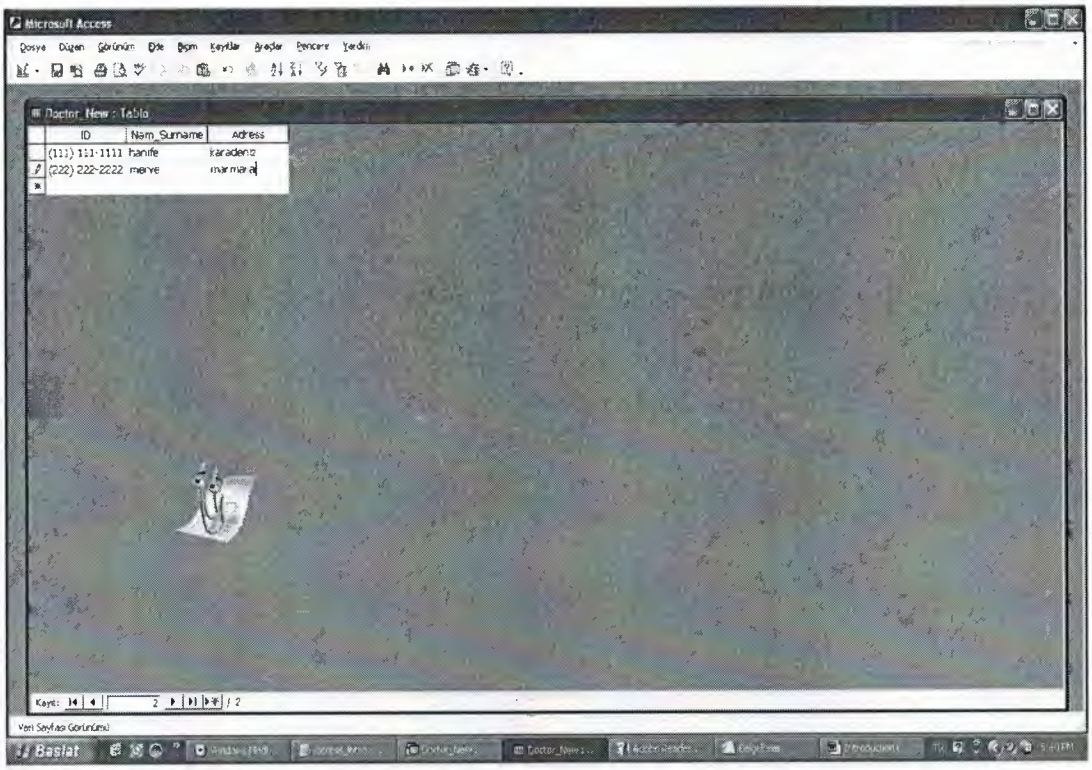
Viewing and Navigating Tables:

1) Select the Tables object

2) Double-click a table's name to open the table

3) Enter the data into the table by pressing the tab key to move from one cell to another

4) When you have completed the record (row), press Enter



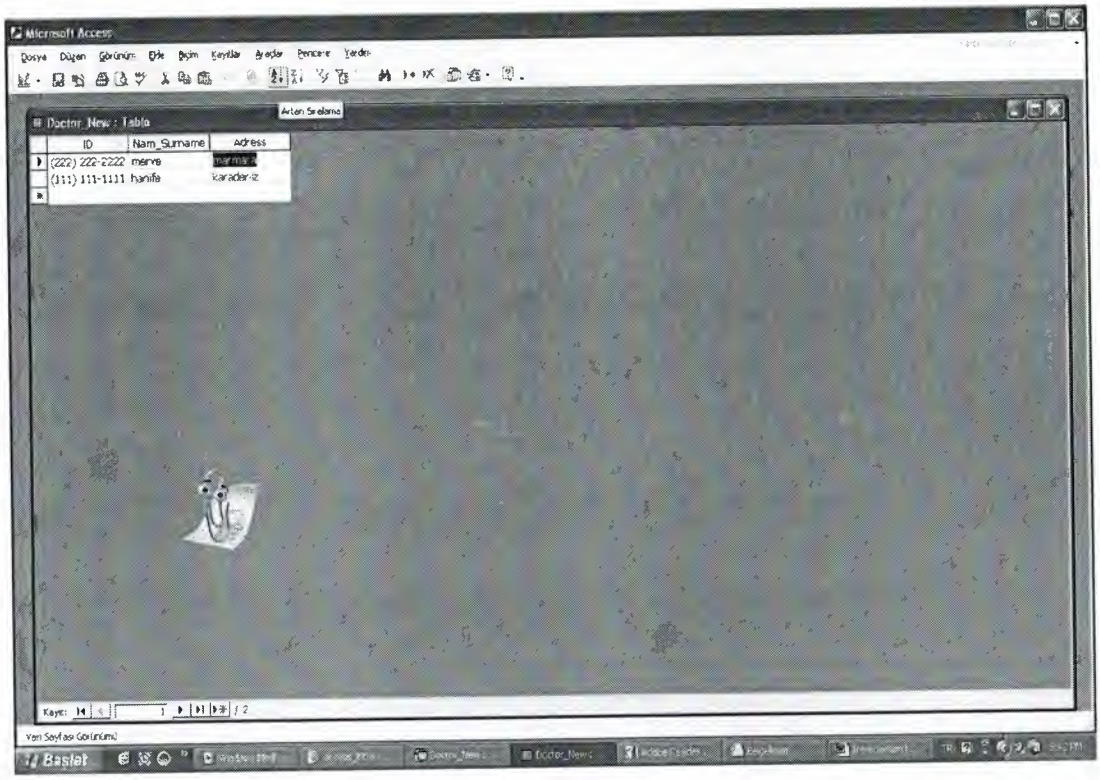
se the arrows to navigate from the first record, previous record, next record, last record, and create a new record. (as shown in the picture above)

Notice that the total number of records in the table is shown at the right end of the navigation arrows.

To Sort Records in a Table:

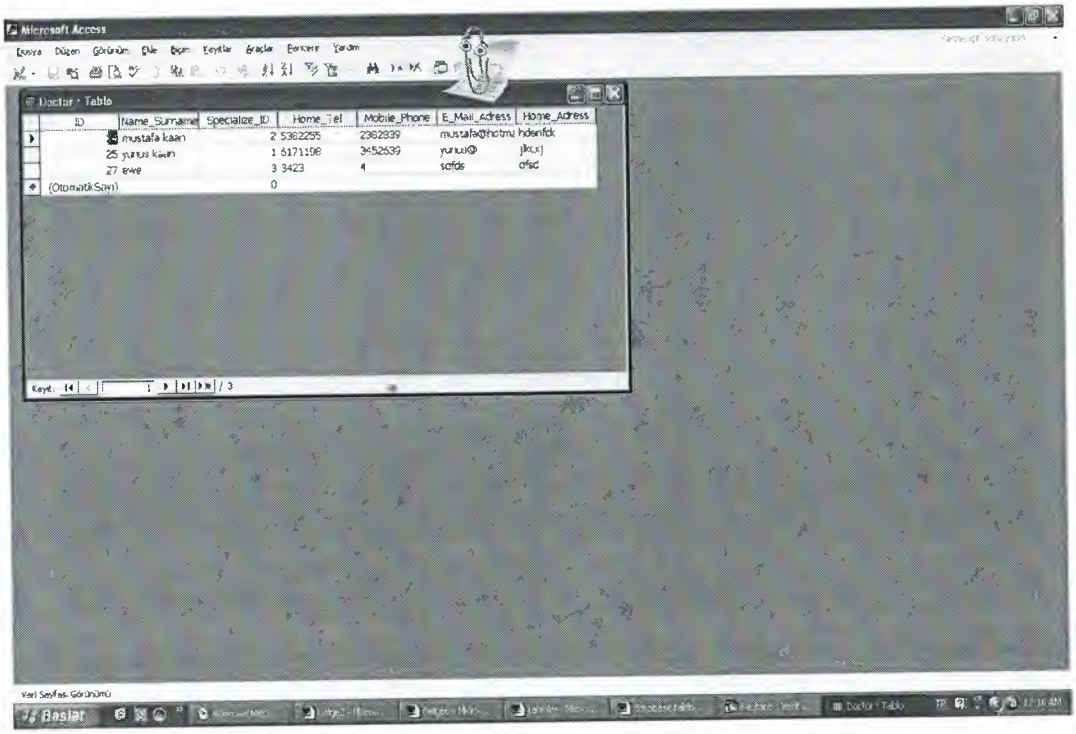
1) Position your cursor in the field that you wish to sort by clicking on any record in the table (make sure your cursor is positioned in the field you are sorting by)

2) Click either the Sort Ascending or Sort Descending icon

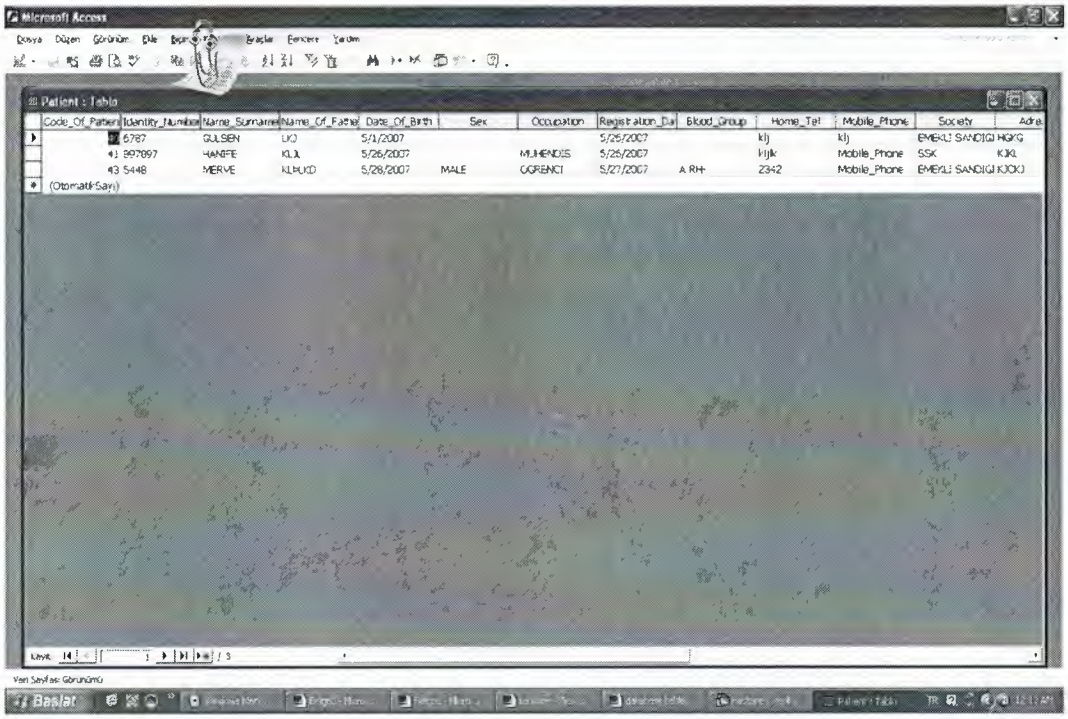


2.3. Tables of Hospital Automation

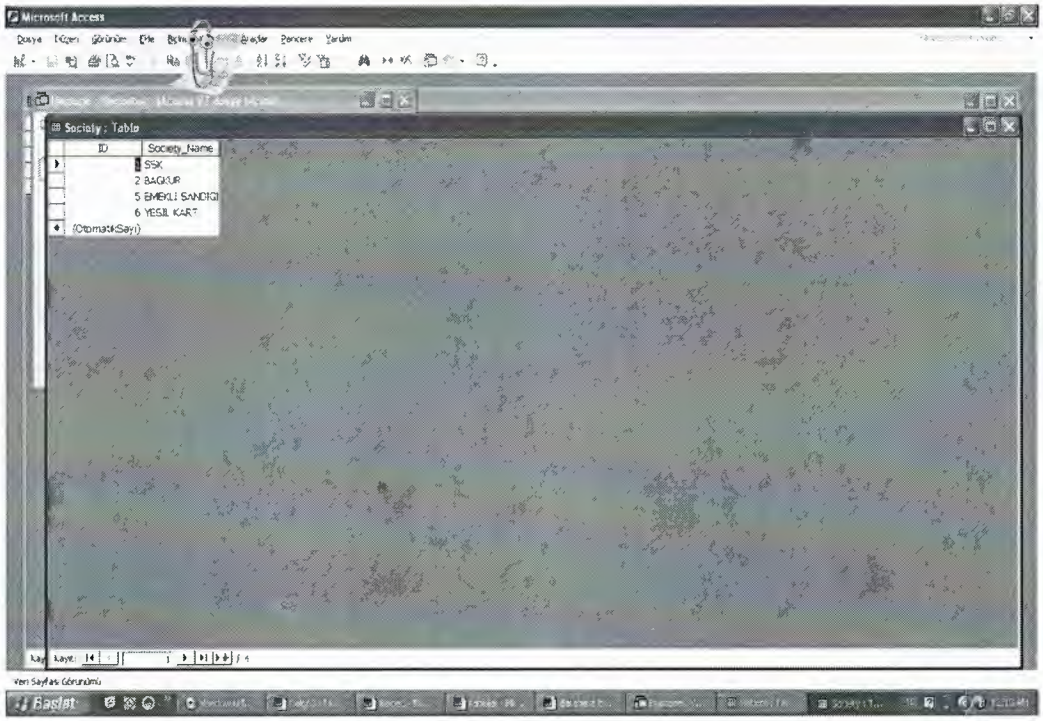
Doctor Table



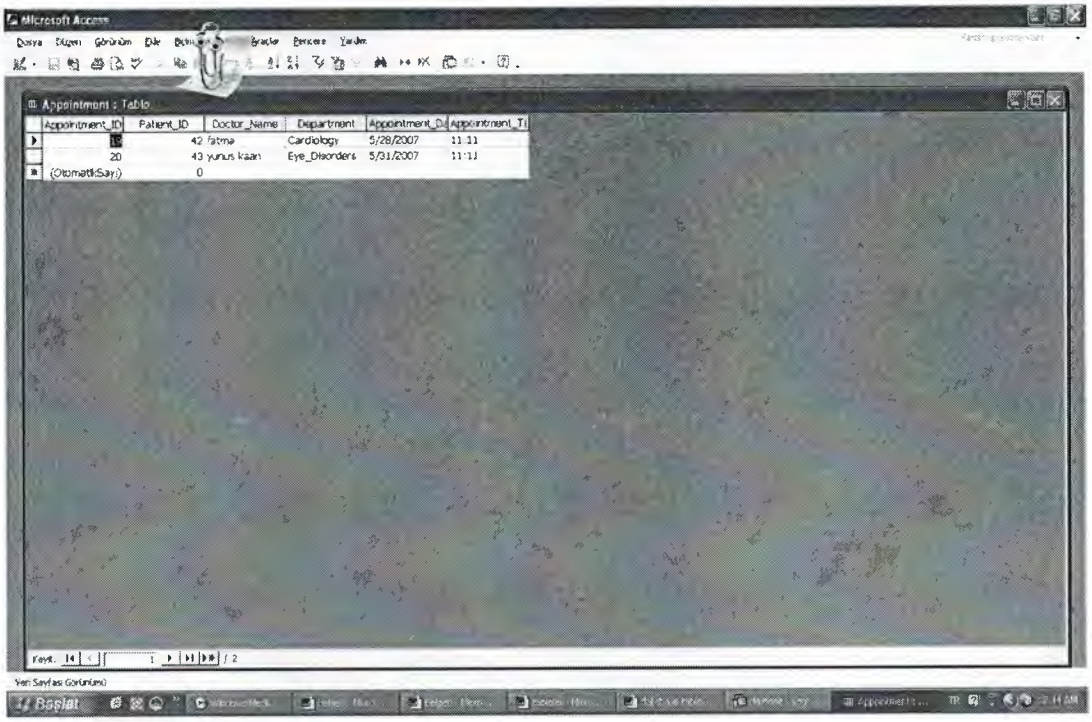
Patient Table



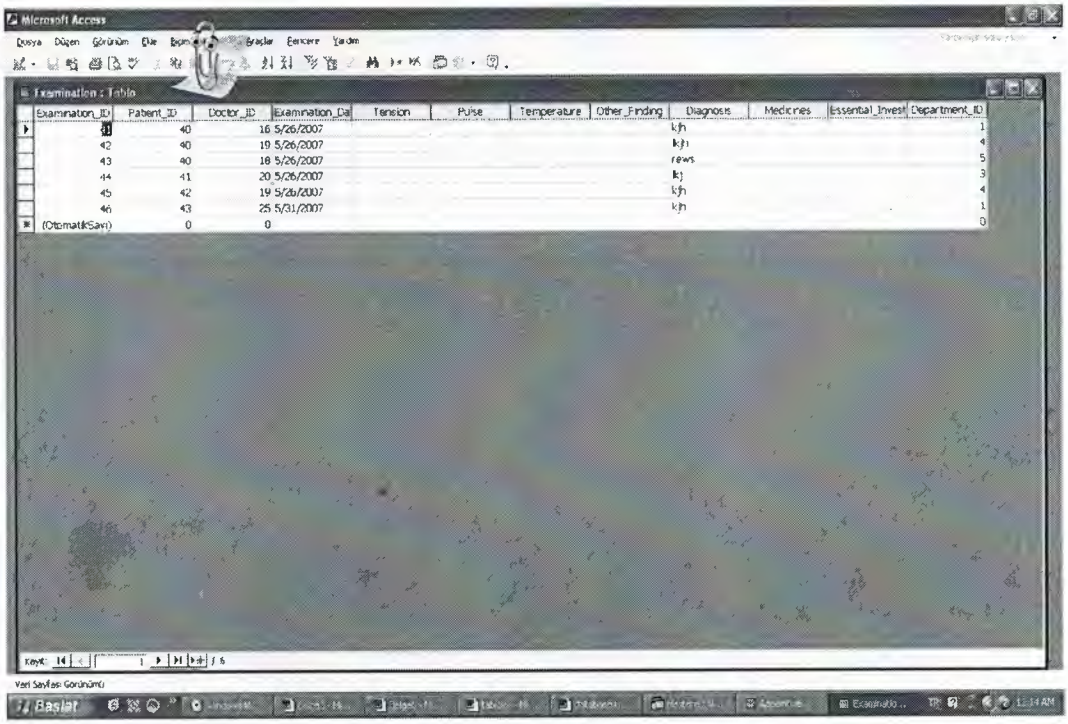
Society Table



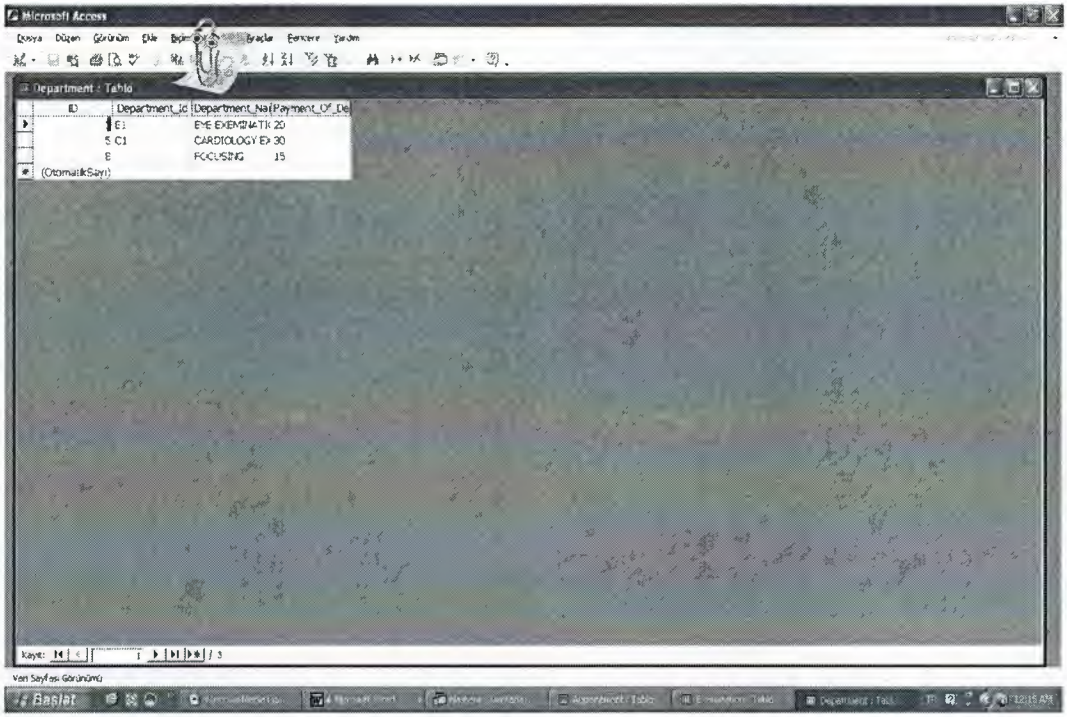
Appointment Table



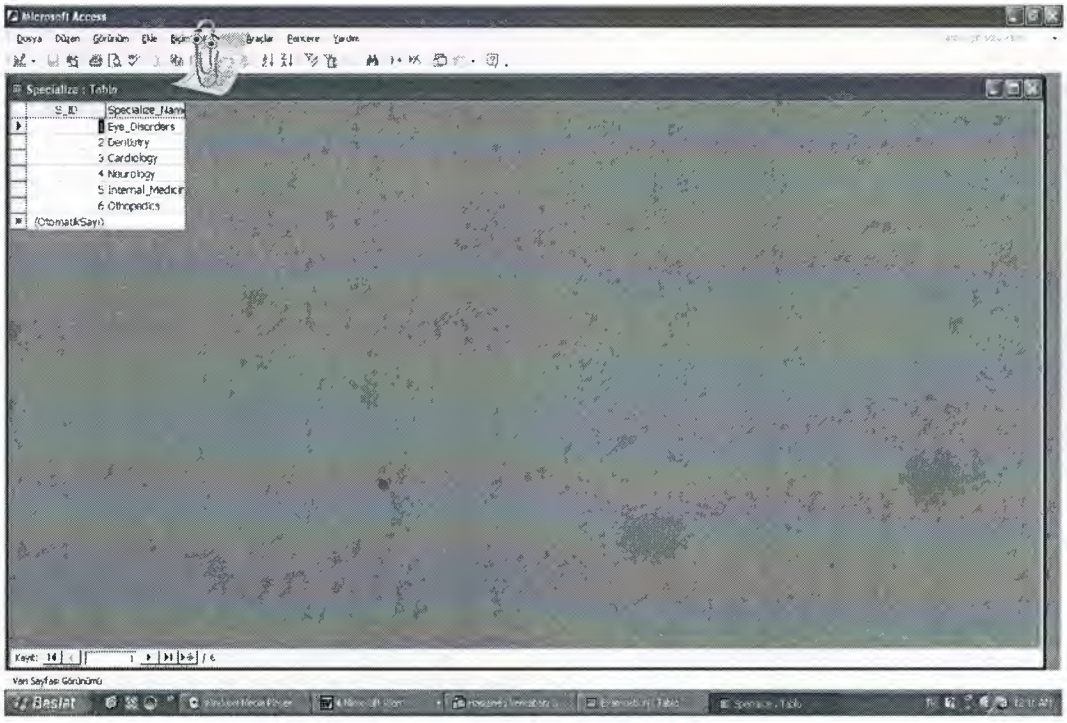
ExaminationTable



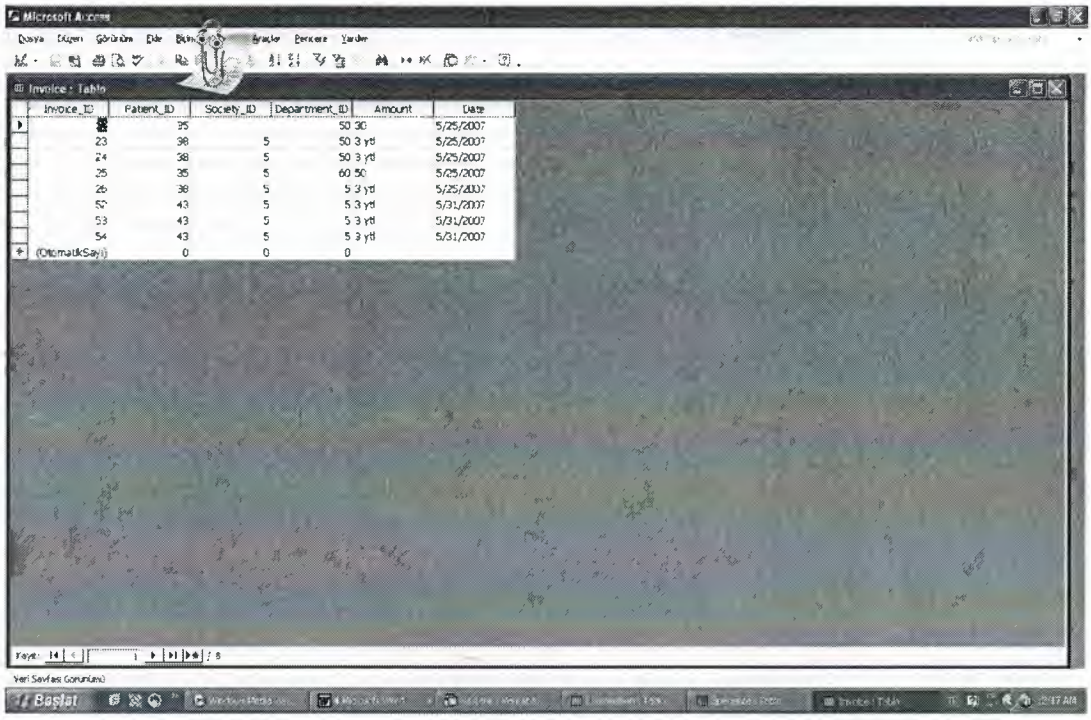
Department Table



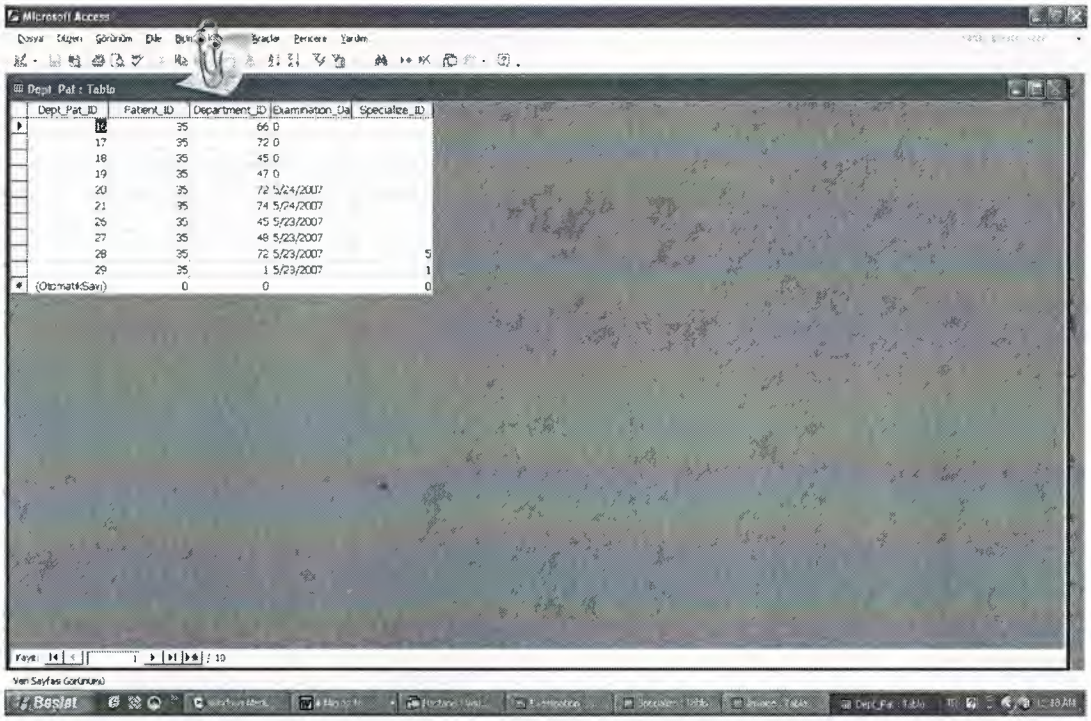
Specialize Table



Invoice Table

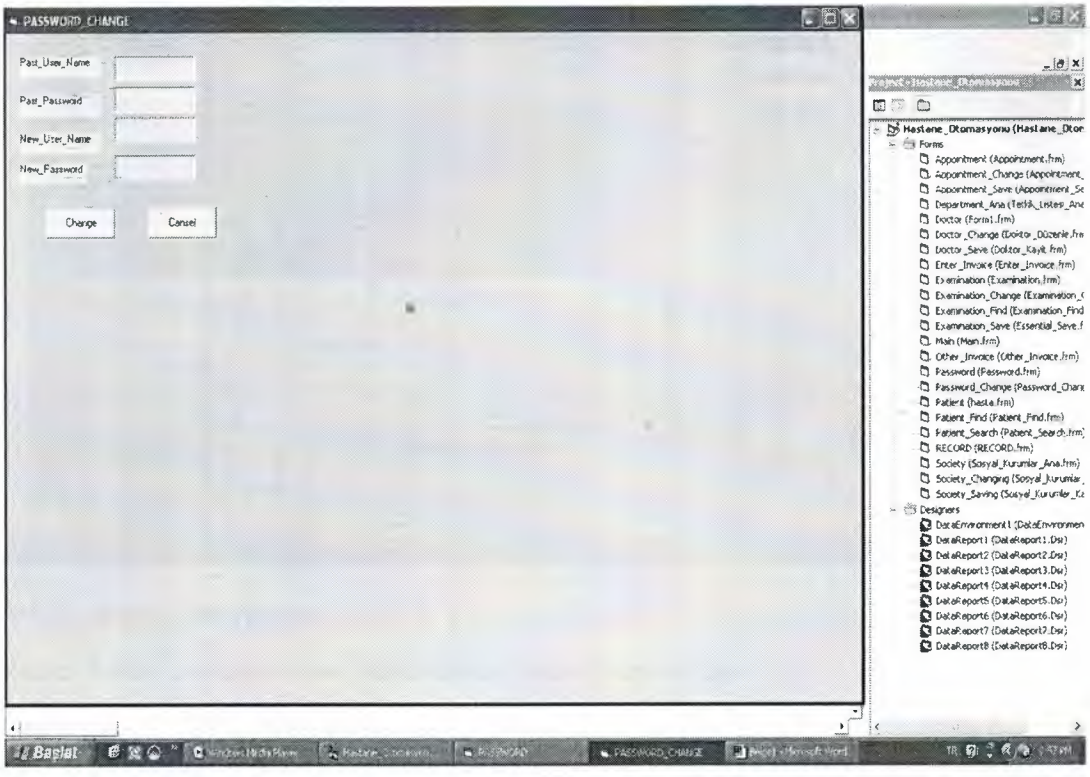


Dept_Pat table

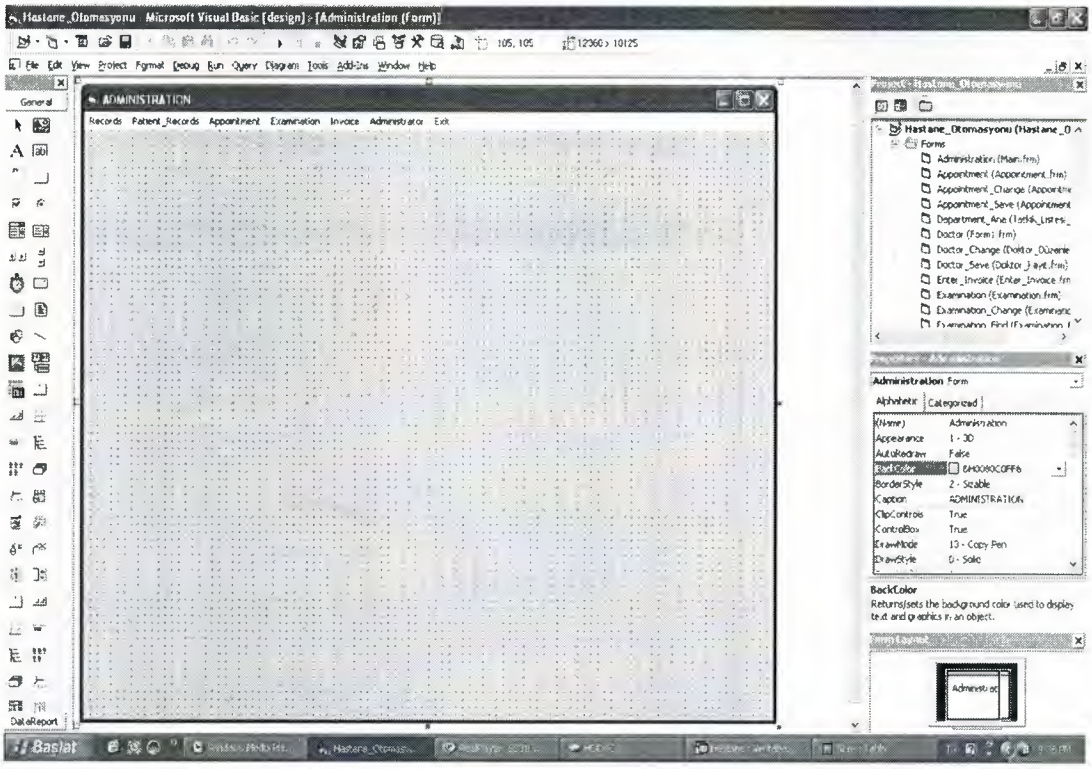


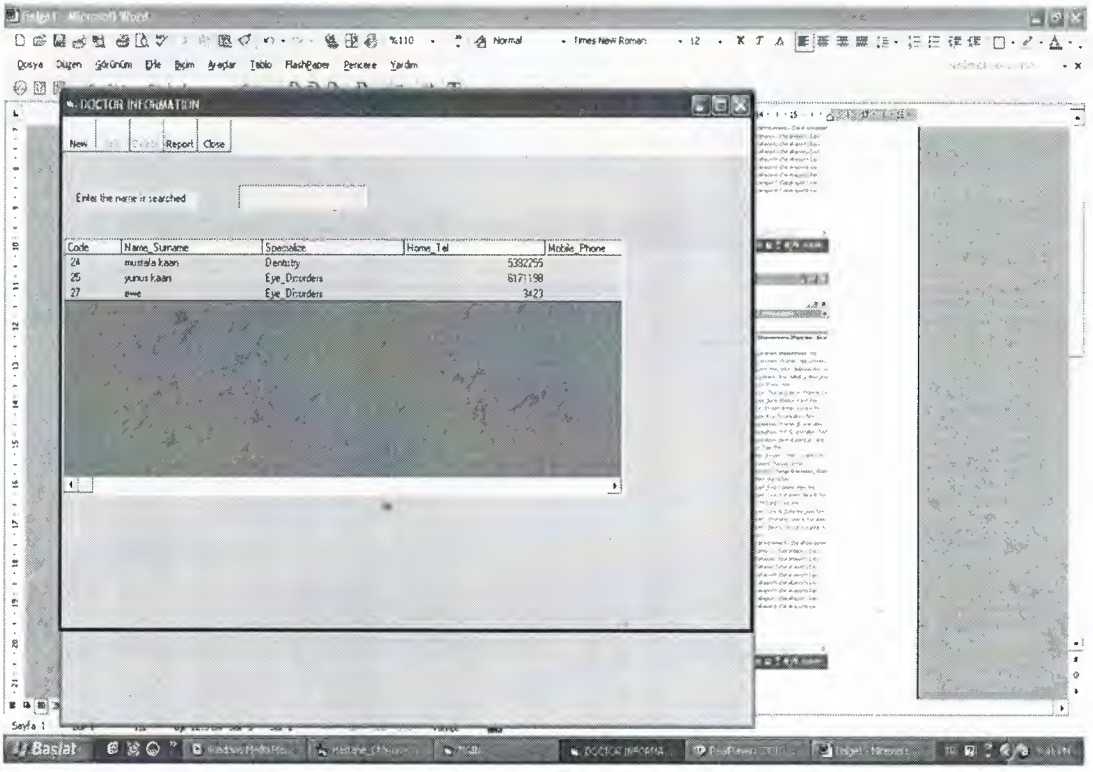
CHAPTER THREE: HOSPITAL AUTOMATION

Each user may have tu has a user name and password to enter the program and he can change his password with anything.



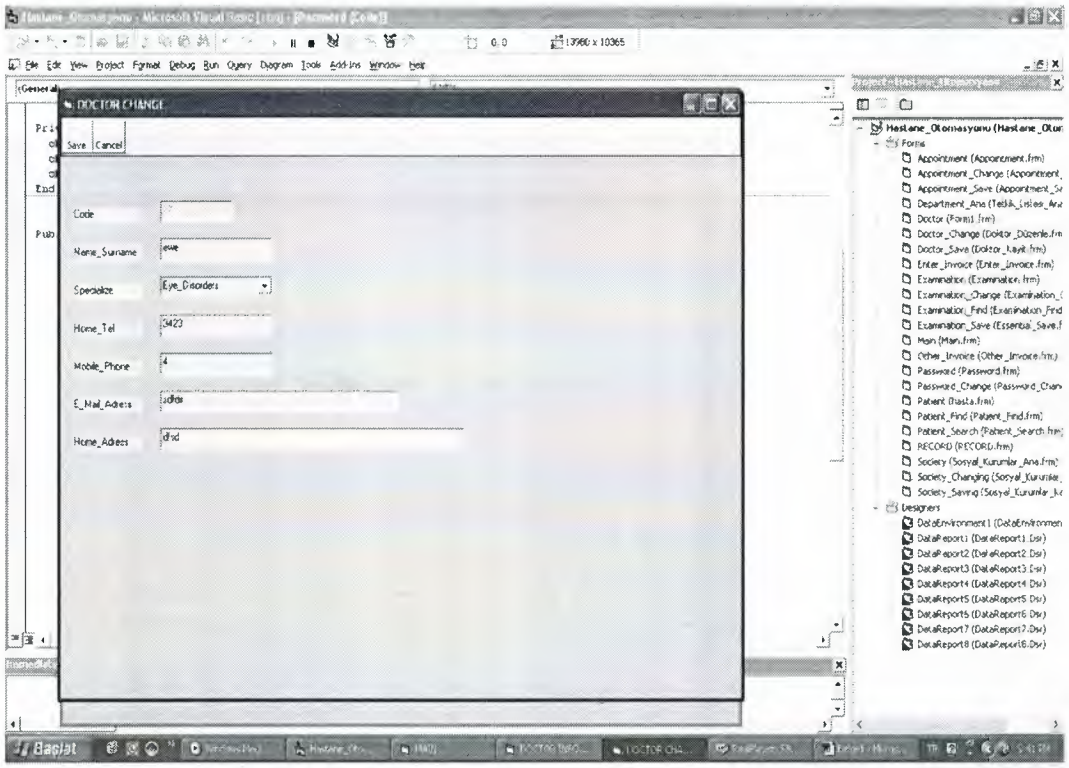
User areach main menu,after entering password.

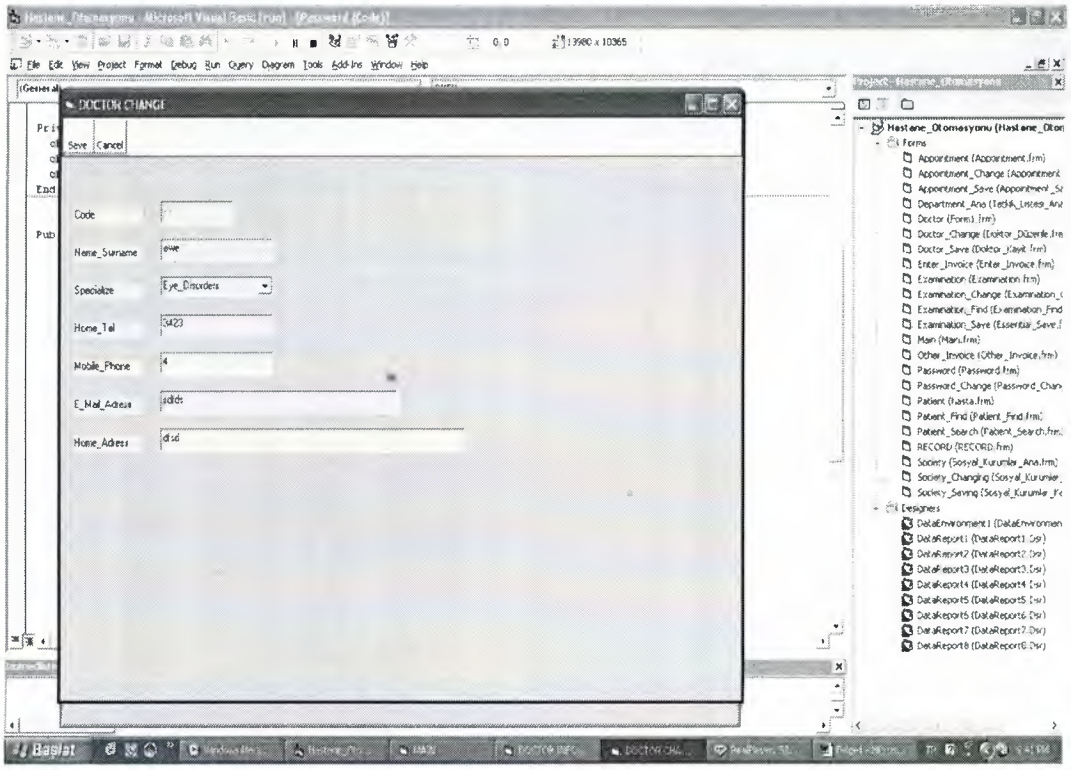




When doctor information table is loaded it shows the doctors who are registered in database. Form contains new, edit, delete, report and close buttons. When form first opened there are new, report, and close buttons. When new button is pressed, doctor save table which makes new registers is opened. This table includes general information about doctor, name- surname, specialize, home telephone, mobile phone, e-

mail address, and home address. All these entries are compulsory and code automatically comes from database.



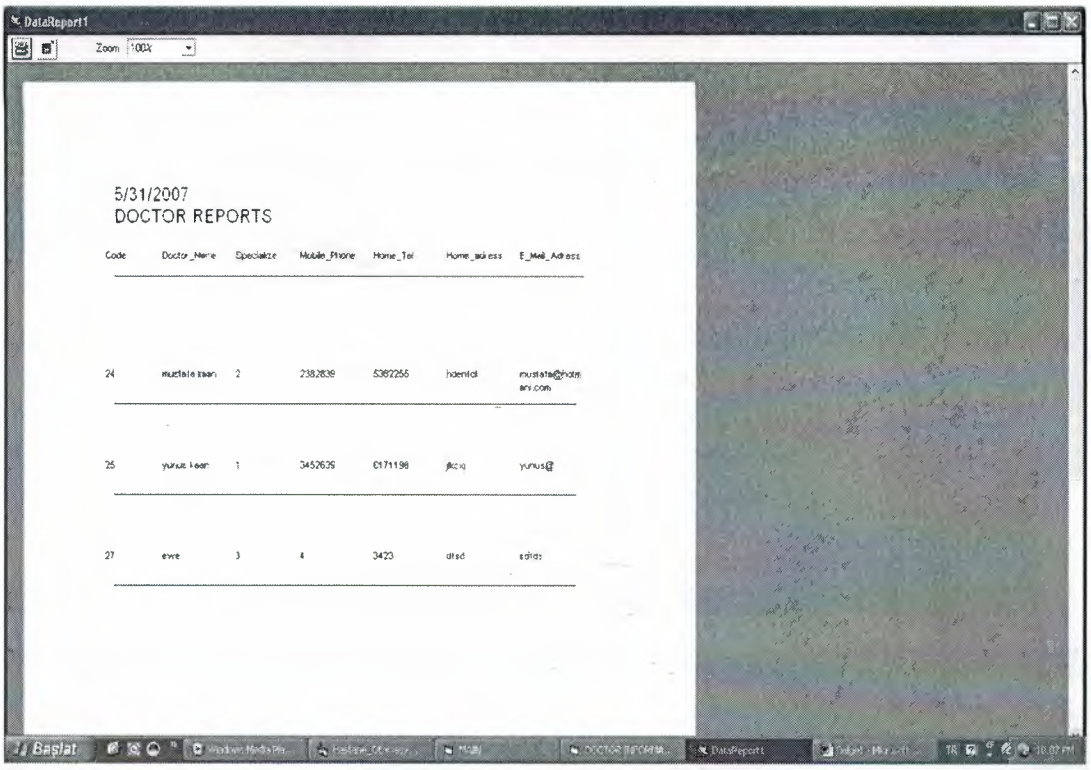


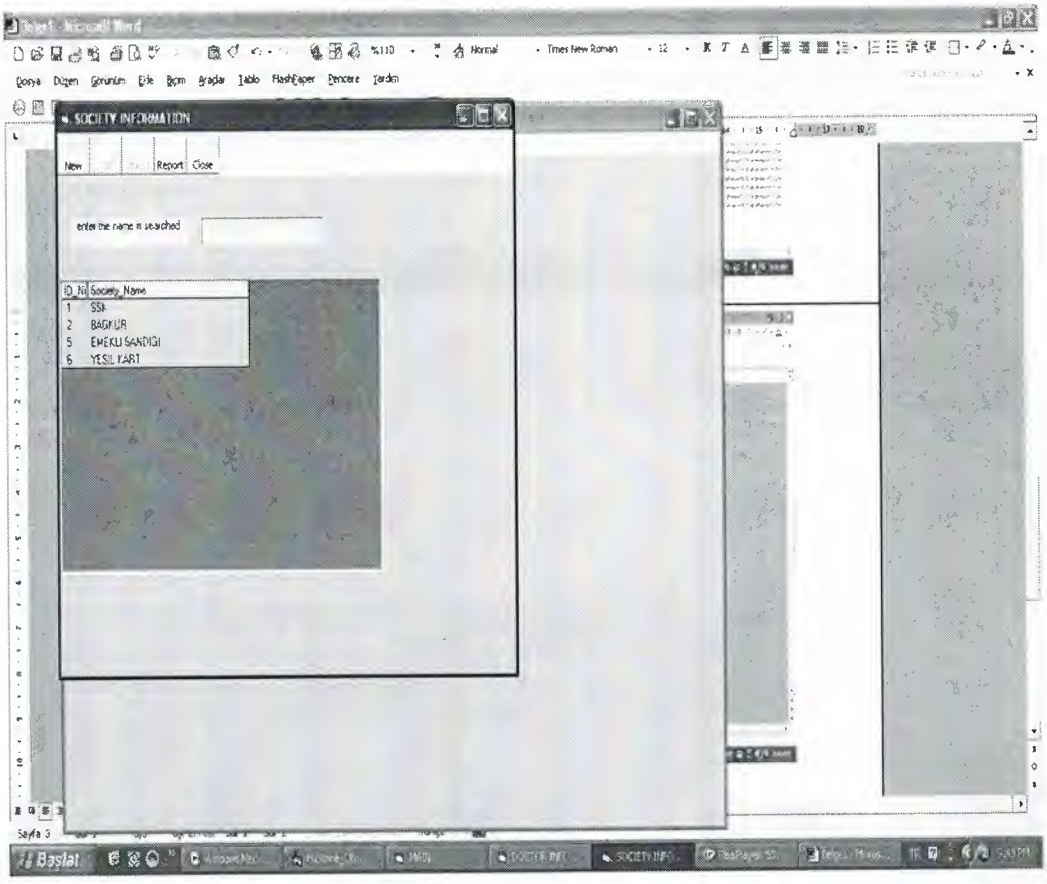
To make edit and delete buttons active, we click the information of the desired doctor on the grid and that paper becomes active. Then we can edit or delete this

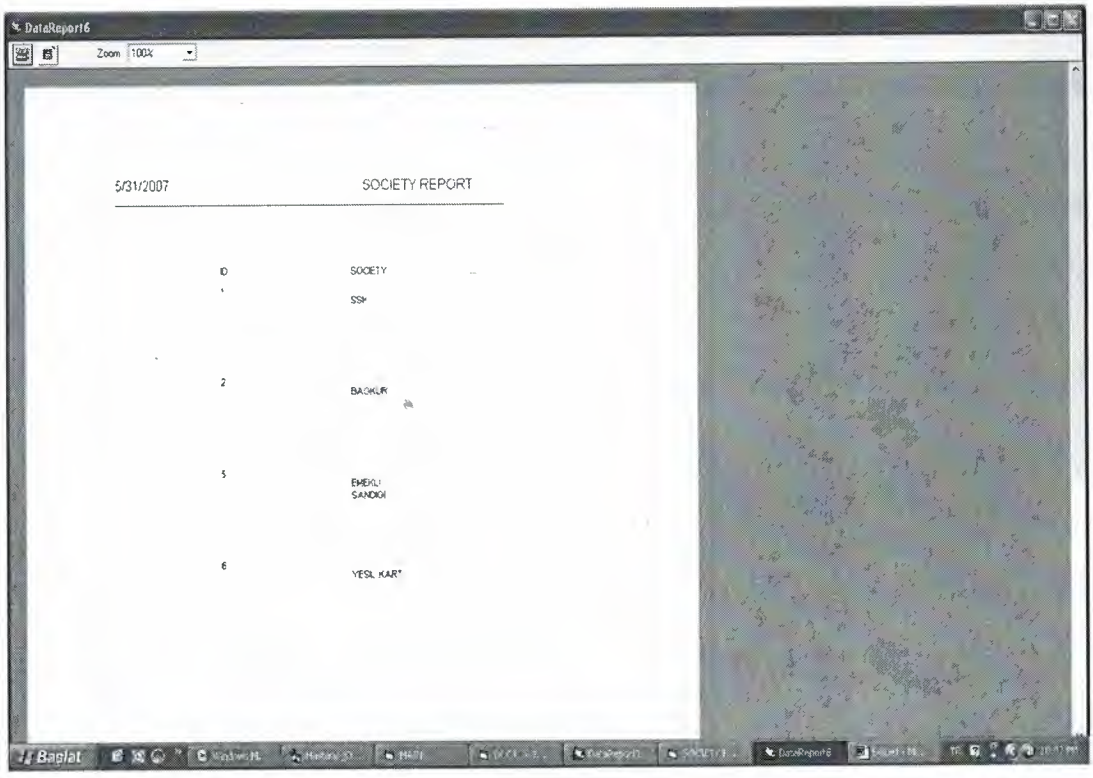
information. The doctor_code form where the information of the activated paper comes from is opened to edit.

Doctor search is made according to doctor name in doctor_information form.

And report buton;

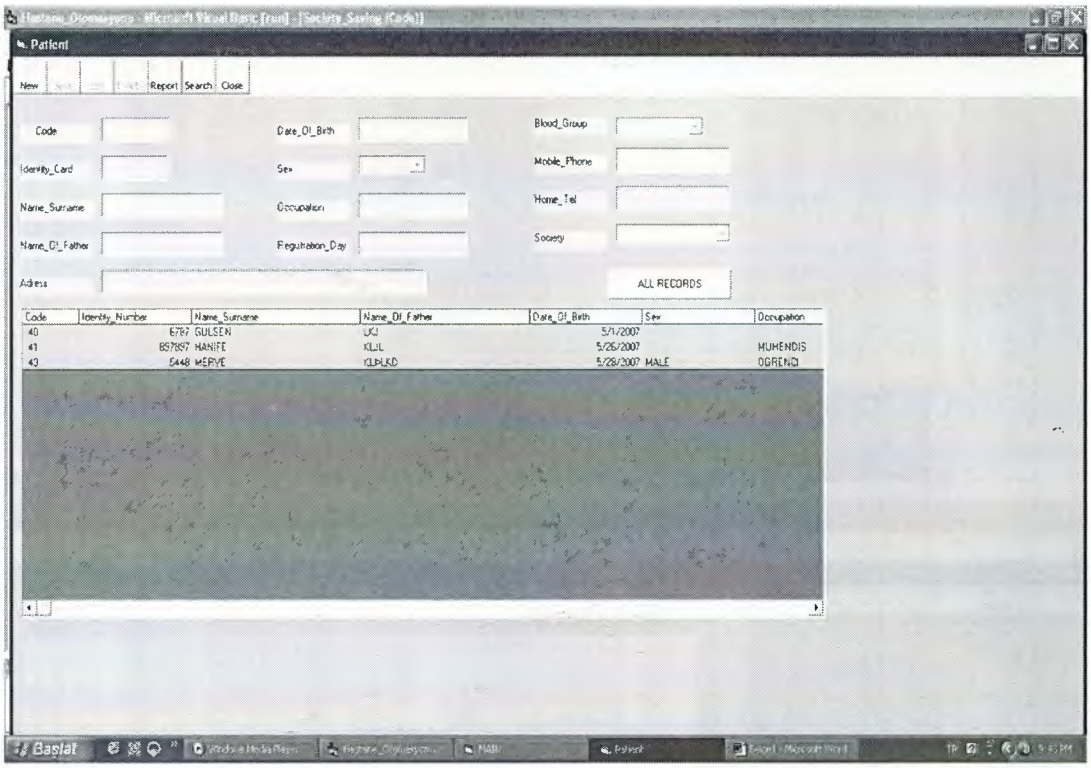




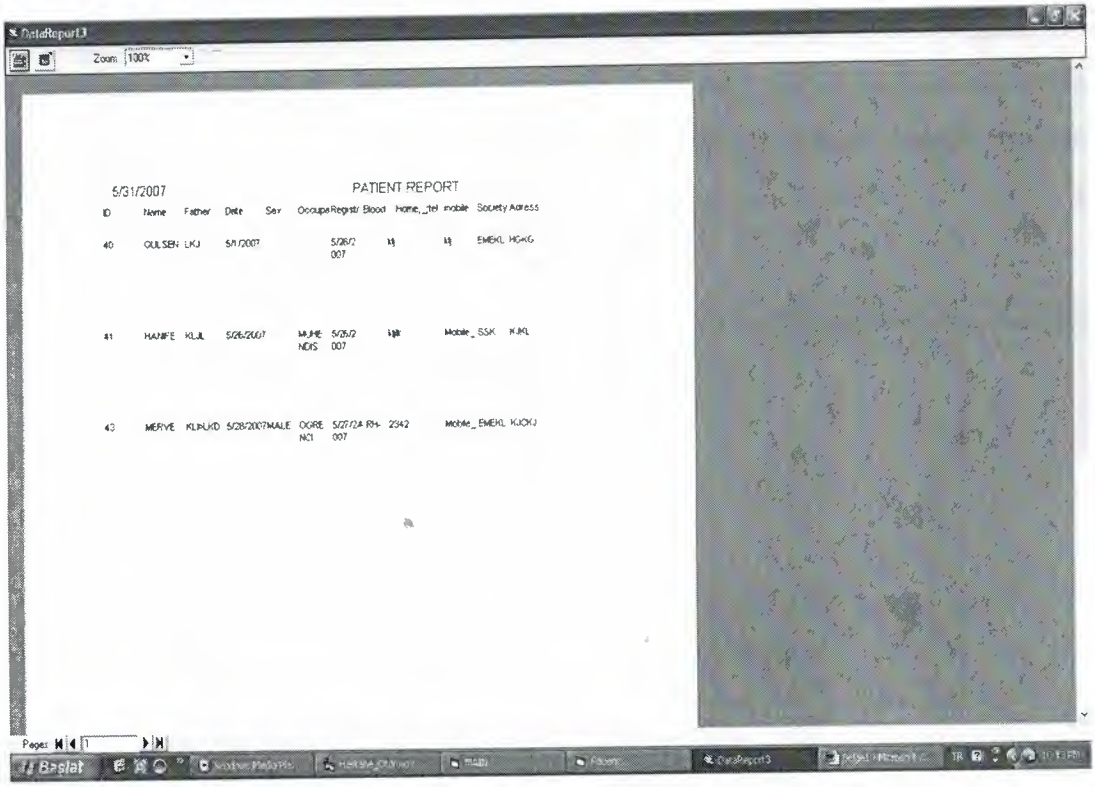


Society_information is a form that shows social institutions. There are new, edit, delete, close, and report buttons. Only new, report and close buttons are active when the form first opened. When we want to enter new institution we click new button and

society_save form is opened. We register the information of the institution by this form. To activate the edit and delete buttons we click the desired registration on grid. Then we can edit and delete. Society_change form in which the information if the institution is opened to edit.

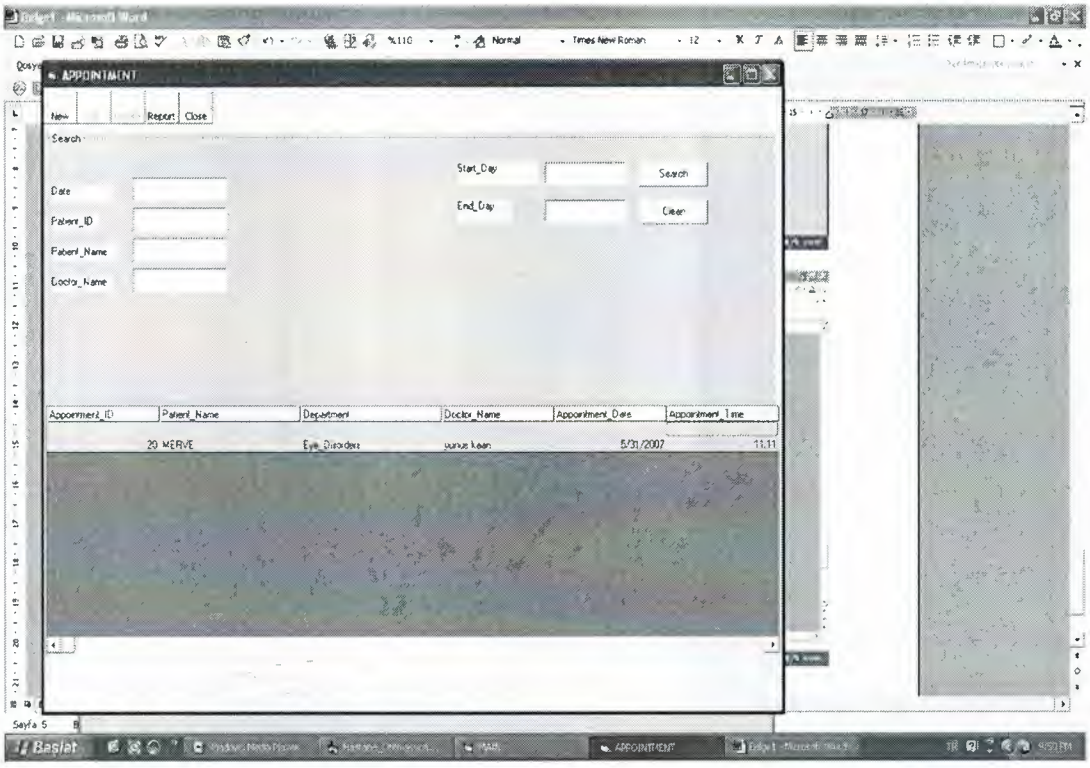


Patient form is used for new patient registration. There are save, new, edit, delete, report, search and close buttons. Only new, report, search and close buttons are active when the form first opened. This form has identity card, name-surname, name of father, address, date of birth, sex, occupation, registration day, blood group, mobile phone, home telephone and society information of patient. But only name-surname, name of father, address, date of birth, and mobile phone entries are compulsory.

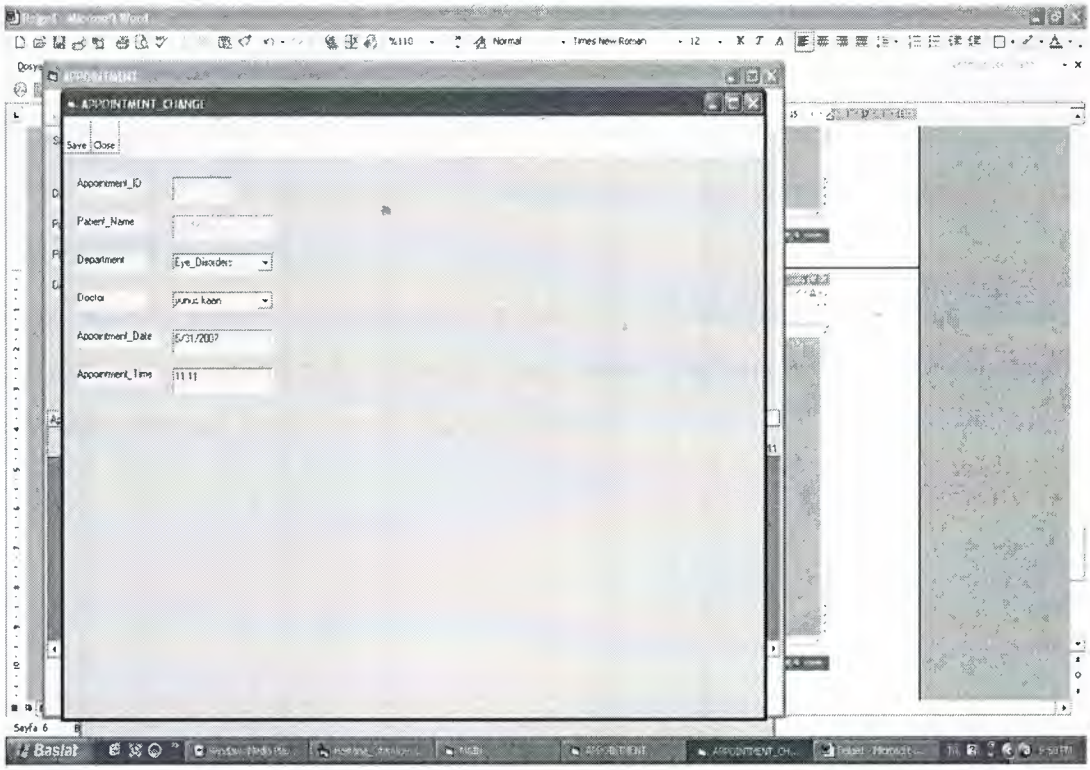


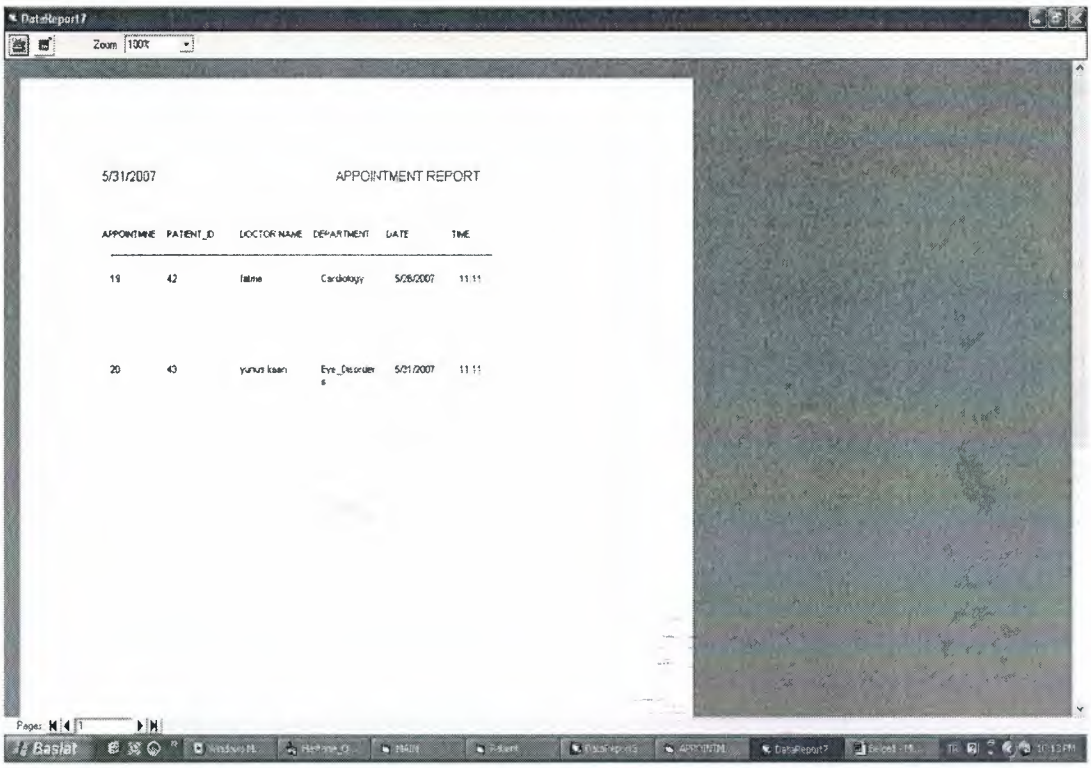
To activate the edit and delete buttons we should select a patient from grid. Patient information is loaded on patient form when we select the patient and edition can be made.

When search button is pressed patient_search form is opened. Search is made with code, name-surname, father's name or identity card in form.



Appointment form is prepared for new appointments. Appointments are listed on the main grid of the appointment form. There are new, edit, delete, report, and close buttons. Only new, report and lose buttons are active when it is first opened. Also in main form search can be made with date, patient ID, patient name, doctor name or between two dates.

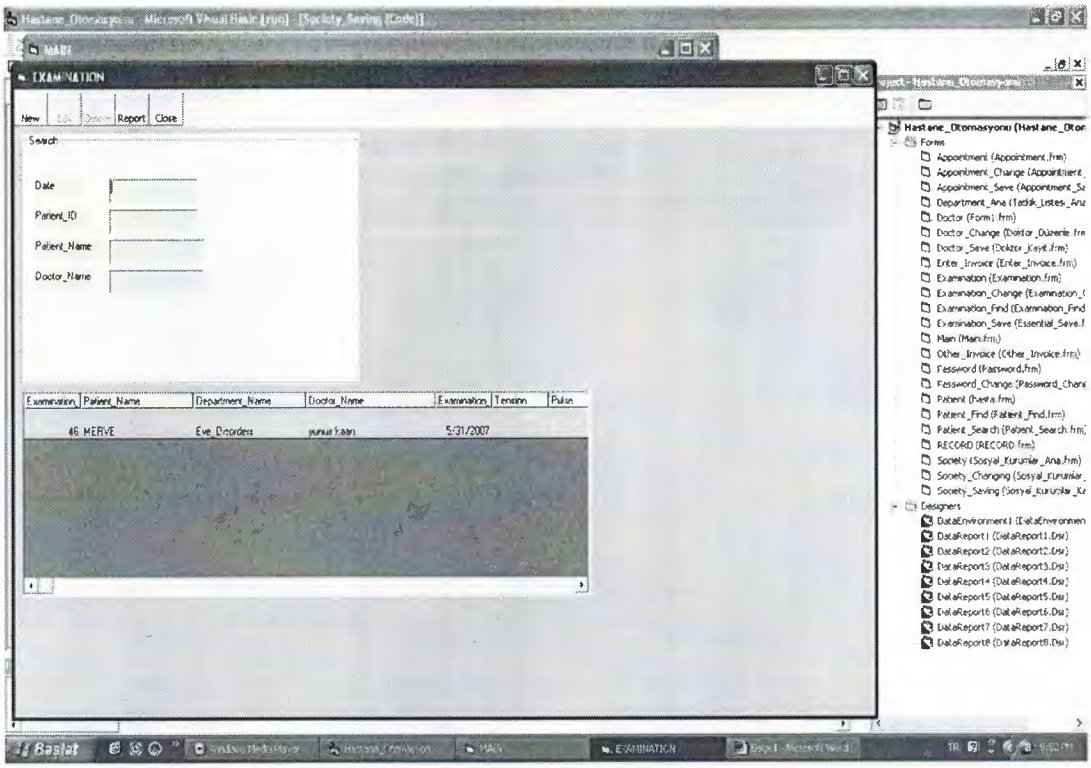




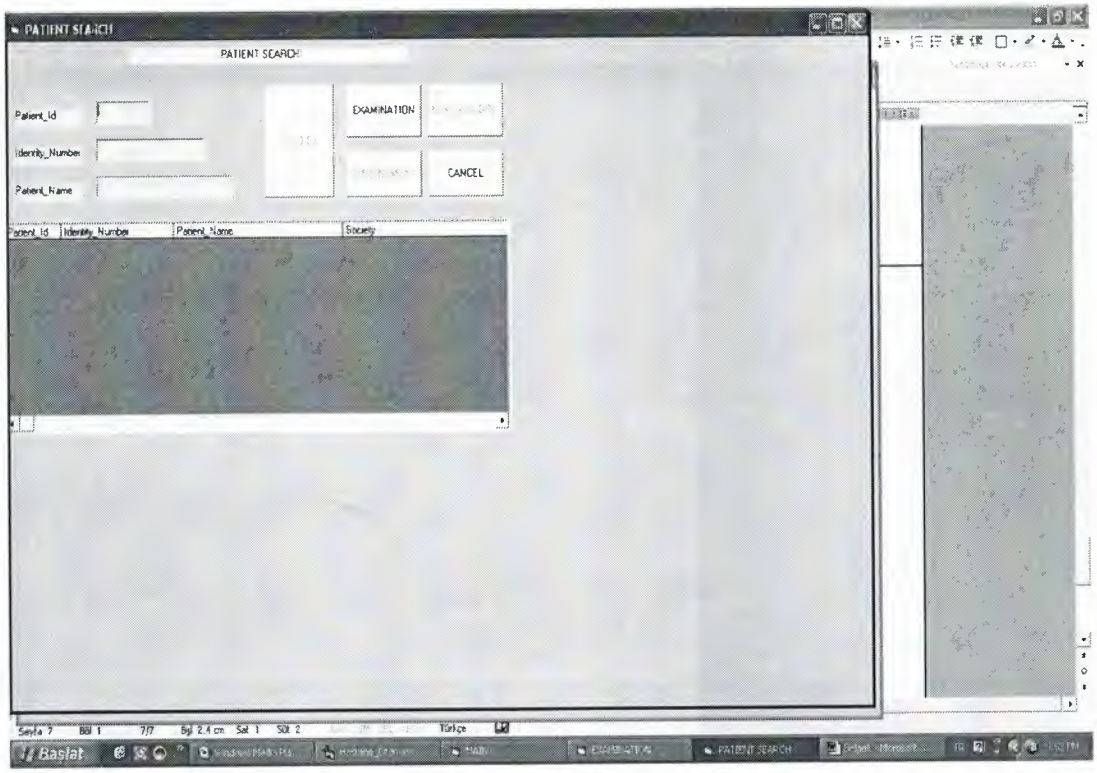
When new button is clicked patient_search form is opened. In this form patient search is made patient ID, identity number or patient name. The found patient is transferred to the grid. When we click the patient on the grid and then click the appointment button, appointment_save form is opened (if user doctor enters patient_search form with his status only examination button becomes active, if secretary enters with his status invoice, appointment and new_record buttons become active.)

Appointment_save form is composed of user's_name, department, doctor name, appointment_date and appointment_time entries. Secretary gives appointment to the patient by using this form.

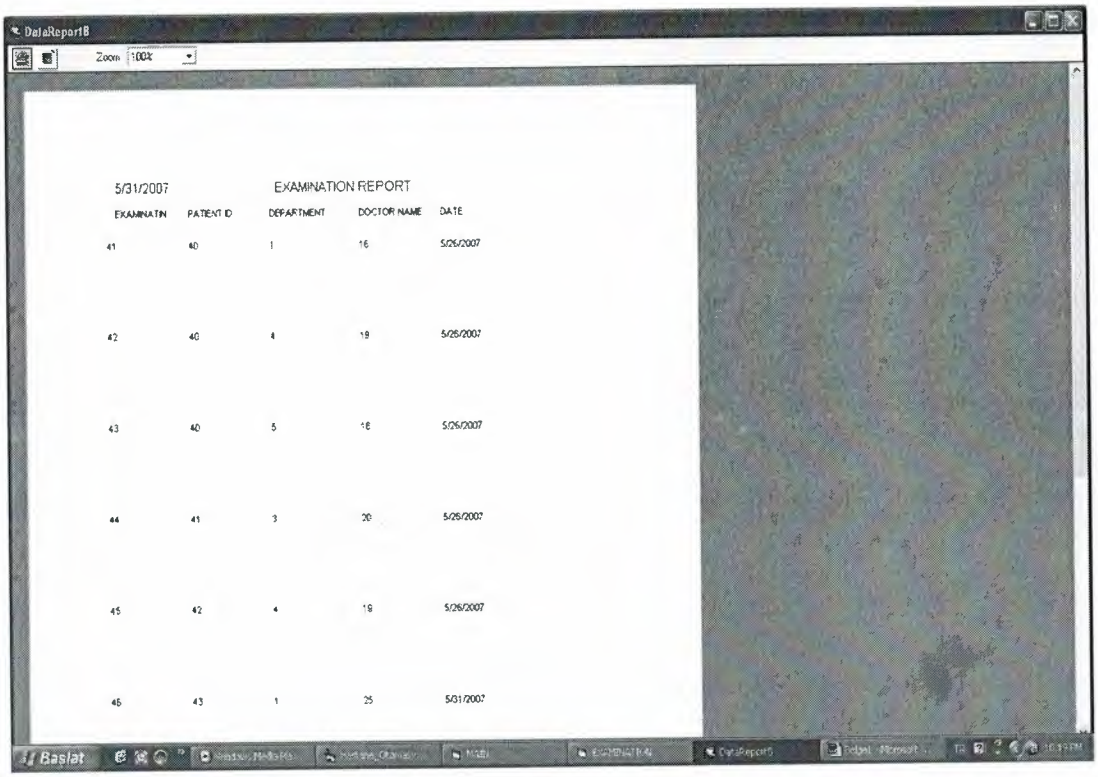
If we want to edit the appointment, we find the appointment of the patient and click delete button and appointment_change form where the appointment information is transferred is opened. Then we edit the appointment.



Examination form is composed of new, edit, delete, report and close buttons. Only new, report and close buttons are active when it is first opened. Search can be made with date, patient ID, patient name, or doctor name. When we click new button patient_search form is opened and doctor can made search with patient ID, identity number, or patient name. Then doctor finds the patient and when he clicks examination button examination_save form is opened.

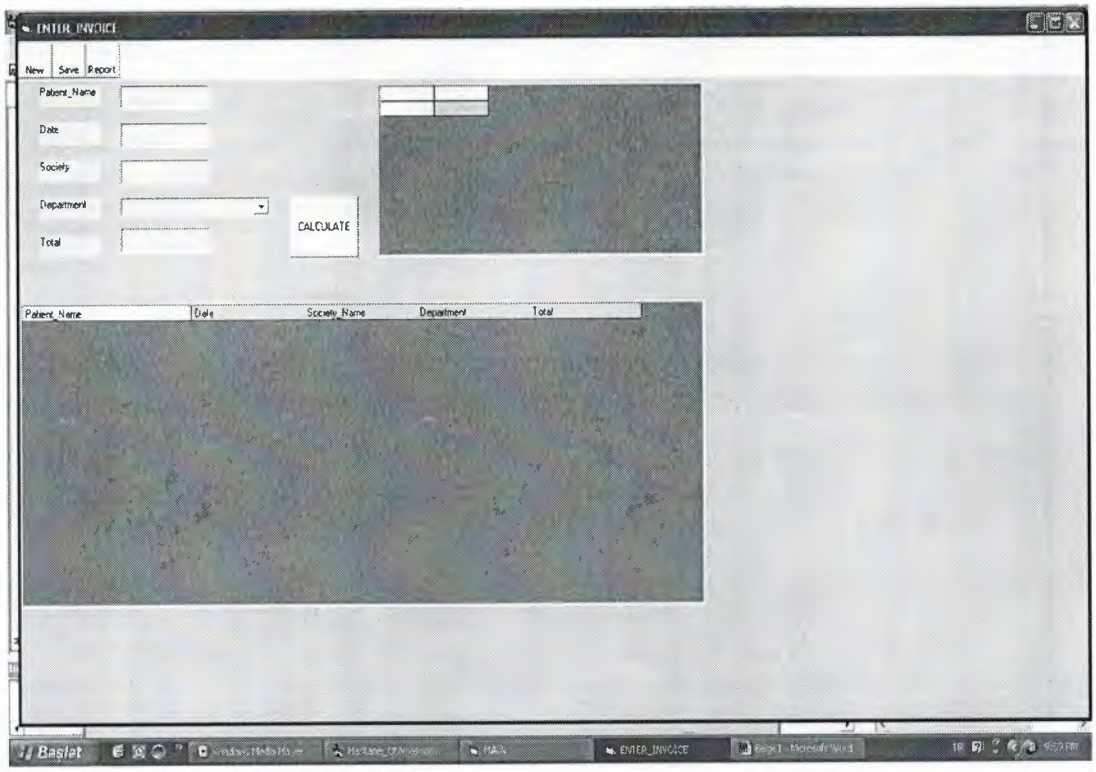

PATIENT SEARCH
PATIENT SEARCH
Patient_Id
Identity_Number
Patient_Name
EXAMINATION
CANCEL
Patient_Id
Identity_Number
Patient_Name
Society
Başlat


EXAMINATION_SAVE
Save
Close
ID
Department_Name
Doctor_Name
Patient_Name
MERVE
Examination_Date
Tension
Pulse
Temperature
Other_Finding
Diagnosis
Medicines
Essential_Investigation
ADD
Başlat



Examination_save form is composed of department_name, doctor_name, examination_date, tension, pulse_temperature, other_findings, diagnosis, medicines, and essential_investigation entries. Doctor prepares an examination form by filling these entries. If doctor wants some extra tests and examinations, he selects these from essential_investigation combos and adds list box.

If there is something that needs to be changed in examination form, patient is selected from grid, then edit button is clicked examination change button is opened and changes are made.



In enter_invoice form, prepares examination invoice according to the department in which the patient has an examination.

It composed of patient name, society, department, and total fee entries. If the patient has a society, the examination fee is fixed and reflected to the total, otherwise examination fee changes according to department and is reflected to the total. And in this form invoice search can be made with patient name, patient number or department. When new button is clicked patient_search form is opened, and patient is found and invoice button is clicked.

Other_invoice form is used in making out invoice of extra tests after examination and examination. Its working style is exactly same as the previous form.

CONCLUSION

In this graduation project was prepared a hospital automation program. In the first chapter of this study was mentioned the general structure of Visual Basic. I also explained Visual Basic's operating principles. In the second chapter was mentioned Access and forming tables. In the third chapter was explained my program with codes and forms.

This projext was designed my program to help doctors and secretaries at their work area. Doctor can easily reach the information of the patients and make registration of the examination for the patient. On the other hand doctor can only reach at those parts only necessary for him. This makes using of my program more basic and faster for them. Thus they can serve much more patients.

Secretary can make patient, doctor, and department registration and cannot react at examination form. This provides a reliable system for the hospital, thus patients information cannot be changed by the secretary. Secretary is only interested in appointment and invoice systems. Prevention of secretary to non-interested areas makes program easier for secretary; this also helps secretary to be faster.

Administrator enables password and user names for new users. These users can change their password later if they want. This also increases the security of my program.

By considering all these information we can say that three different users can use the same program without any confusion.

REFERENCES

[1]Microsoft,*Microsoft Visual Basic 5.0 Programmer's Guide,*Microsoft Pres,america,1957

[2]David I.Schneider,*Inroduction to Visual Basic 6.0*,Pentice Hall,America

[3] Ihsan Karagülle,Zeydin Pala,*Visual Basic Pro 5.0*,Turkmen Kitabevi,Bitlis,1997

[4] Polikilinik Yönetim Modeli,2007 from the World Wide Web:"http:/www.tipdata.com.tr/klinik.php".

[5] Hastane Otomasyon Yöntemi,January 1,2006 from the World Wide Web:"http:/www.medisofthadtane.com".

[6]Hastane Yönetim Sistemi 2000 from the World Wide Web:"http:/www.medipro2000.com".

APPENDIX

SOURCE CODE

PASSWORD FORM;

```
Private Sub Form_Load()
  cb_Status.AddItem "DOCTOR"
  cb_Status.AddItem "SECRETARY"
  cb_Status.AddItem "ADMINISTRATION"
End Sub

Public Sub OKEY()
   Set mydata = OpenDatabase("C:\Hastane.mdb")
   st_sql = "select * from Sifre WHERE User_Status='" & cb_Status.Text & " ' and " &
_
   "User_Name_Surname='" & tx_User_Name & "' and " & _
   "User_Sifre='" & tx_Password & "'"

   Set mytable = mydata.OpenRecordset(st_sql)

   If mytable.RecordCount > 0 Then
     Main.Show (modal)
      If cb_Status = "DOCTOR" Then
        Main.M_Records.Enabled = False
        Main.M_Appointment.Enabled = False
        Main.M_Examination.Enabled = True
        Main.M_Patient_Records.Enabled = False
        Main.M_Enter_Invoice.Enabled = False
        Main.M_Administrator.Enabled = False
        Main.M_Invoice.Enabled = False
        Main.M_Other_Invoice.Enabled = False
        Main.M_Administrator.Enabled = True
        status = "DOCTOR"

      ElseIf cb_Status = "SECRETARY" Then
        Main.M_Records.Enabled = True
        Main.M_Appointment.Enabled = True
        Main.M_Examination.Enabled = False
        Main.M_Patient_Records.Enabled = True
        Main.M_Enter_Invoice.Enabled = True
        Main.M_Administrator.Enabled = False
        Main.M_Invoice.Enabled = True
        Main.M_Other_Invoice.Enabled = True
        Main.M_Administrator.Enabled = false
        status = "SECRETARY"
      ElseIf cb_Status = "ADMINISTRATOR" Then
        Main.M_Records.Enabled = false
        Main.M_Appointment.Enabled = false
```

```
        Main.M_Examination.Enabled = false
        Main.M_Patient_Records.Enabled = false
        Main.M_Enter_Invoice.Enabled = false
        Main.M_Administrator.Enabled = false
        Main.M_Invoice.Enabled = false
        Main.M_Other_Invoice.Enabled = false
        Main.M_Administrator.Enabled = true
      End If

  Else
    MsgBox ("Error Entering")
  End If

  mytable.Close
  mydata.Close
  Unload Password
End Sub

Private Sub tx_Password_KeyPress(KeyAscii As Integer)

   If KeyAscii = 13 Then
    OKEY
   End If

End Sub
```

PASSWORD CHANGE FORM;

```
Dim mydata As Database
Dim mytable As Recordset

Private Sub cm_Cancel_Click()
  Unload Password_Change
End Sub

Private Sub cm_Change_Click()
    MsgBox (Password.cb_Status.Text)
    Set mydata = OpenDatabase("C:\Hastane.mdb")
    Set mytable = mydata.OpenRecordset("Sifre")
    st_sql = "Update Sifre Set User_Name_Surname='" & tx_New_User_Name & "'
,User_Sifre='" & tx_New_Password & "'
 where User_Name_Surname='" & tx_Past_User_Name & "'
AND User_Sifre='" & tx_Past_Password & "'
 AND User_Status='" & Password.cb_Status.Text & """
    mydata.Execute (st_sql)
    mytable.Close
    mydata.Close
    Unload Password_Change
End Sub
```

MAIN FORM;

```
Private Sub M_Administrator_Click()
    RECORD.Show
End Sub

Private Sub M_Appointment_Click()
    Appointment.Show (modal)
End Sub

Private Sub M_Department_Records_Click()
    Department_Ana.Show (modal)
End Sub

Private Sub M_Doctor_Records_Click()
    Doctor.Show (modal)
End Sub

Private Sub M_Enter_Invoice_Click()
   Enter_Invoice.Show (modal)
End Sub

Private Sub M_Examination_Click()
    Examination.Show (modal)
End Sub

Private Sub M_Exit_Click()
    Unload Main
End Sub

Private Sub M_Other_Invoice_Click()
    Other_Invoice.Show (modal)
End Sub

Private Sub M_Patient_Records_Click()
    Patient.Show (modal)
End Sub

Private Sub M_Society_Records_Click()
    Society.Show (modal)
End Sub
```

DOCTOR FORM;

```
Private Sub Command1_Click()
    DataReport1.Show
End Sub
```

```
Private Sub fg_Doktor_Click()

    'grid tiklandiginde row'un icindekileri degiskenlere aktarma

    id = fg_Doktor.TextMatrix(fg_Doktor.Row, 0)
    Name_Surname = fg_Doktor.TextMatrix(fg_Doktor.Row, 1)
    Specialize = fg_Doktor.TextMatrix(fg_Doktor.Row, 2)
    Home_Tel = fg_Doktor.TextMatrix(fg_Doktor.Row, 3)
    Mobile_Phone = fg_Doktor.TextMatrix(fg_Doktor.Row, 4)
    E_Mail_Adress = fg_Doktor.TextMatrix(fg_Doktor.Row, 5)
    Home_Adress = fg_Doktor.TextMatrix(fg_Doktor.Row, 6)

    'Toolbar Controlling

    Tb_Doktor.Buttons.Item(1).Enabled = False
    Tb_Doktor.Buttons.Item(2).Enabled = True
    Tb_Doktor.Buttons.Item(3).Enabled = True
    Tb_Doktor.Buttons.Item(4).Enabled = True
    Tb_Doktor.Buttons.Item(5).Enabled = True

End Sub

Private Sub Form_Load()

    'Arranging Grid
    Grid_Düzenleme

    fg_Doktor.ColWidth(0) = 1000
    fg_Doktor.ColWidth(1) = 2500
    fg_Doktor.ColWidth(2) = 2500
    fg_Doktor.ColWidth(3) = 2500
    fg_Doktor.ColWidth(4) = 2500
    fg_Doktor.ColWidth(5) = 5000
    fg_Doktor.ColWidth(6) = 9000

    Tb_Doktor.Buttons.Item(1).Enabled = True
    Tb_Doktor.Buttons.Item(2).Enabled = False
    Tb_Doktor.Buttons.Item(3).Enabled = False
    Tb_Doktor.Buttons.Item(4).Enabled = True
    Tb_Doktor.Buttons.Item(5).Enabled = True


   'Transfer the datas in database to grid

    Set mydata = OpenDatabase("C:\Hastane.mdb")
    st_sql = "select * from Doctor"
    Set mytable = mydata.OpenRecordset(st_sql)


    While Not mytable.EOF
```

```
    St_sql1 = "select * from Specialize where S_ID=" &
Val(mytable.Fields("Specialize_ID")) & " "
        Set mytable1 = mydata.OpenRecordset(St_sql1)
          While Not mytable1.EOF
            name1 = mytable1.Fields("Specialize_Name")
            mytable1.MoveNext
         Wend
         mytable1.Close

    fg_Doktor.AddItem Str(mytable.Fields("ID")) & Chr(9) & _
                mytable.Fields("Name_Surname") & Chr(9) & _
                name1 & Chr(9) & _
                mytable.Fields("Home_Tel") & Chr(9) & _
                mytable.Fields("Mobile_Phone") & Chr(9) & _
                mytable.Fields("E_Mail_Adress") & Chr(9) & _
                mytable.Fields("Home_Adress")
  mytable.MoveNext
  Wend
  mytable.Close
  mydata.Close


End Sub

Private Sub Tb_Doktor_ButtonClick(ByVal Button As ComctlLib.Button)

  ' Arranging 'tb_Doktor'

  If Button.Key = "New" Then
     Doctor_Save.Show (modal)
  ElseIf Button.Key = "Edit" Then
     Doctor_Change.Show (modal)
     Doctor_Change.tx_Kodu = id
     Doctor_Change.tx_Adi_Soyadi = Name_Surname
     Doctor_Change.Cb_Specialize = Specialize
     Doctor_Change.tx_Ev_Tel = Home_Tel
     Doctor_Change.tx_Cep_Tel = Mobile_Phone
     Doctor_Change.tx_E_Mail_Adresi = E_Mail_Adress
     Doctor_Change.tx_Ev_Adresi = Home_Adress
   ElseIf Button.Key = "Delete" Then
     j = MsgBox("Do you want to delete?", 33)
       If j = 1 Then
         Set mydata = OpenDatabase("C:\Hastane.mdb")
         Set mytable = mydata.OpenRecordset("Doctor")
         st_sql = "delete from Doctor where ID=" & id & " "
         mydata.Execute (st_sql)
         mytable.Close
         mydata.Close
         Grid_Düzenleme
```

```
        Set mydata = OpenDatabase("C:\Hastane.mdb")
        st_sql = "select * from Doctor"
        Set mytable = mydata.OpenRecordset(st_sql)


       While Not mytable.EOF
         St_sql1 = "select * from Specialize where S_ID=" &
Val(mytable.Fields("Specialize_ID")) & " "
          Set mytable1 = mydata.OpenRecordset(St_sql1)
           While Not mytable1.EOF
             name1 = mytable1.Fields("Specialize_Name")
             mytable1.MoveNext
           Wend
          mytable1.Close

          fg_Doktor.AddItem Str(mytable.Fields("ID")) & Chr(9) & _
                    mytable.Fields("Name_Surname") & Chr(9) & _
                    name1 & Chr(9) & _
                Str(mytable.Fields("Home_Tel")) & Chr(9) & _
                Str(mytable.Fields("Mobile_Phone")) & Chr(9) & _
                    mytable.Fields("E_Mail_Adress") & Chr(9) & _
                    mytable.Fields("Home_Adress")

          mytable.MoveNext
       Wend

        'tollbar'in düzenlenmesi

        Doctor.Tb_Doktor.Buttons.Item(1).Enabled = True
        Doctor.Tb_Doktor.Buttons.Item(2).Enabled = False
        Doctor.Tb_Doktor.Buttons.Item(3).Enabled = False
        Doctor.Tb_Doktor.Buttons.Item(4).Enabled = True
        Doctor.Tb_Doktor.Buttons.Item(5).Enabled = True
        mytable.Close
        mydata.Close

      Else: Doctor.Show (modal)
      End If

  ElseIf Button.Key = "Close" Then

      Unload Doctor
  ElseIf Button.Key = "Report" Then

      DataReport1.Show
  End If

End Sub

Private Sub tx_Ara_Change()
```

```
fg_Doktor.FixedRows = 1
fg_Doktor.FixedCols = 0
fg_Doktor.Cols = 7
fg_Doktor.Rows = 2
fg_Doktor.Clear
fg_Doktor.Row = 0
fg_Doktor.Col = 0
fg_Doktor.Text = "Code"
fg_Doktor.Col = 1
fg_Doktor.Text = "Name_Surname"
fg_Doktor.Col = 2
fg_Doktor.Text = "Specialize"
fg_Doktor.Col = 3
fg_Doktor.Text = "Home_Tel"
fg_Doktor.Col = 4
fg_Doktor.Text = "Mobile_Phone"
fg_Doktor.Col = 5
fg_Doktor.Text = "E_Mail_Adress"
fg_Doktor.Col = 6
fg_Doktor.Text = "Home_Adress"
Set mydata = OpenDatabase("C:\Hastane.mdb")
st_sql = "select * from Doctor where Name_Surname='" & tx_Ara & "'"
Set mytable = mydata.OpenRecordset(st_sql)


While Not mytable.EOF
  St_sql1 = "select * from Specialize where S_ID=" &
Val(mytable.Fields("Specialize_ID")) & " "
  Set mytable1 = mydata.OpenRecordset(St_sql1)
    While Not mytable1.EOF
      name1 = mytable1.Fields("Specialize_Name")
      mytable1.MoveNext
    Wend
    mytable1.Close


  fg_Doktor.AddItem Str(mytable.Fields("ID")) & Chr(9) & _
                    mytable.Fields("Name_Surname") & Chr(9) & _
                    name1 & Chr(9) & _
                  Str(mytable.Fields("Home_Tel")) & Chr(9) & _
                  Str(mytable.Fields("Mobile_Phone")) & Chr(9) & _
                      mytable.Fields("E_Mail_Adress") & Chr(9) & _
                      mytable.Fields("Home_Adress")
  mytable.MoveNext
Wend
mytable.Close
mydata.Close
```

```
End Sub

Public Sub Grid_Düzenleme()

   fg_Doktor.FixedRows = 1
   fg_Doktor.FixedCols = 0
   fg_Doktor.Cols = 7
   fg_Doktor.Rows = 1
   fg_Doktor.Clear
   fg_Doktor.Row = 0
   fg_Doktor.Col = 0
   fg_Doktor.Text = "Code"
   fg_Doktor.Col = 1
   fg_Doktor.Text = "Name_Surname"
   fg_Doktor.Col = 2
   fg_Doktor.Text = "Specialize"
   fg_Doktor.Col = 3
   fg_Doktor.Text = "Home_Tel"
   fg_Doktor.Col = 4
   fg_Doktor.Text = "Mobile_Phone"
   fg_Doktor.Col = 5
   fg_Doktor.Text = "E_Mail_Adress"
   fg_Doktor.Col = 6
   fg_Doktor.Text = "Home_Adress"

End Sub
```

DOCTOR_CHANGE FORM;

```
Private Sub Form_Load()

 Set mydata = OpenDatabase("C:\Hastane.mdb")
 st_sql = "select * from Specialize"
 Set mytable = mydata.OpenRecordset(st_sql)
 While Not mytable.EOF
    Cb_Specialize.AddItem mytable.Fields("Specialize_Name")
    mytable.MoveNext
 Wend
 mytable.Close
 mydata.Close
 End Sub

Private Sub Toolbar1_ButtonClick(ByVal Button As ComctlLib.Button)
   If Button.Key = "Save" Then

         If tx_Kodu = "" Or _
           tx_Adi_Soyadi = " " Or _
           tx_Bransi = " " Or _
           tx_Ev_Tel = " " Or _
           tx_Cep_Tel = " " Or _
           tx_Email = " " Or _
```

```
        tx_Ev_Adresi = " " Then
            MsgBox ("Bosluklari Doldurunuz")
        Else

            If IsNumeric(tx_Ev_Tel.Text) And IsNumeric(tx_Cep_Tel.Text) Then
               j = MsgBox("Do you want to edit?", 33)
                 If j = 1 Then
                   Set mydata = OpenDatabase("C:\Hastane.mdb")
                   St_sql1 = "select * from Specialize where Specialize_Name='" &
Cb_Specialize & "'"
                   Set mytable1 = mydata.OpenRecordset(St_sql1)
                   While Not mytable1.EOF
                     id = mytable1.Fields("S_ID")
                     mytable1.MoveNext
                   Wend
                   mytable1.Close

                   Set mytable = mydata.OpenRecordset("Doctor")
                   st_sql = "Update Doctor Set Name_Surname= '" & tx_Adi_Soyadi
&                 "',Specialize_ID=" & id & ",Home_Tel=" & Val(tx_Ev_Tel) &
",Mobile_Phone=" & Val(tx_Cep_Tel) & ",E_Mail_Adress='" & tx_E_Mail_Adresi &
"',Home_Adress='" & tx_Ev_Adresi & "' where ID=" & Val(tx_Kodu) & " "
                   mydata.Execute (st_sql)
                   mytable.Close
                   mydata.Close

                   Doctor.fg_Doktor.FixedRows = 1
                   Doctor.fg_Doktor.FixedCols = 0
                   Doctor.fg_Doktor.Cols = 7
                   Doctor.fg_Doktor.Rows = 1
                   Doctor.fg_Doktor.Clear
                   Doctor.fg_Doktor.Row = 0
                   Doctor.fg_Doktor.Col = 0
                   Doctor.fg_Doktor.Text = "Code"
                   Doctor.fg_Doktor.Col = 1
                   Doctor.fg_Doktor = "Name_Surname"
                   Doctor.fg_Doktor.Col = 2
                   Doctor.fg_Doktor.Text = "Specialize"
                   Doctor.fg_Doktor.Col = 3
                   Doctor.fg_Doktor.Text = "Home_Tel"
                   Doctor.fg_Doktor.Col = 4
                   Doctor.fg_Doktor.Text = "Mobile-Phone"
                   Doctor.fg_Doktor.Col = 5
                   Doctor.fg_Doktor.Text = "E_Mail_Adress"
                   Doctor.fg_Doktor.Col = 6
                   Doctor.fg_Doktor.Text = "Home_Adress"

                   Set mydata = OpenDatabase("C:\Hastane.mdb")
                   st_sql = "select * from Doctor"
                   Set mytable = mydata.OpenRecordset(st_sql)
```

```
                While Not mytable.EOF
                  St_sql1 = "select * from Specialize where S_ID=" &
Val(mytable.Fields("Specialize_ID")) & " "
                  Set mytable1 = mydata.OpenRecordset(St_sql1)
                    While Not mytable1.EOF
                      name1 = mytable1.Fields("Specialize_Name")
                      mytable1.MoveNext
                    Wend
                    mytable1.Close

                Doctor.fg_Doktor.AddItem Str(mytable.Fields("ID")) & Chr(9) &
_
                                mytable.Fields("Name_Surname") & Chr(9) &
_
                                name1 & Chr(9) & _
                              Str(mytable.Fields("Home_Tel")) & Chr(9) & _
                              Str(mytable.Fields("Mobile_Phone")) & Chr(9) &
_
                                mytable.Fields("E_Mail_Adress") & Chr(9) & _
                                mytable.Fields("Home_Adress")
                                mytable.MoveNext
                Wend
                mytable.Close
                mydata.Close

                Unload Doctor_Change
                Doctor.Tb_Doktor.Buttons.Item(1).Enabled = True
                Doctor.Tb_Doktor.Buttons.Item(2).Enabled = False
                Doctor.Tb_Doktor.Buttons.Item(3).Enabled = False
                Doctor.Tb_Doktor.Buttons.Item(4).Enabled = True
                Doctor.Tb_Doktor.Buttons.Item(5).Enabled = True
              Else
                Doctor_Change.Show (modal)
              End If

        ElseIf (Not IsNumeric(tx_Ev_Tel.Text)) And (Not
IsNumeric(tx_Cep_Tel.Text)) Then
            MsgBox ("Ev_Tel ve Cep_Teli numara giriniz")
            tx_Ev_Tel = " "
            tx_Cep_Tel = " "
            tx_Ev_Tel.SetFocus
        ElseIf Not IsNumeric(tx_Ev_Tel.Text) Then
            MsgBox ("Ev_Teli numara giriniz")
            tx_Ev_Tel = " "
            tx_Ev_Tel.SetFocus
        Else
            MsgBox ("Cep_Teli numara giriniz")
            tx_Cep_Tel = " "
```

```
            tx_Cep_Tel.SetFocus
        End If
        End If

ElseIf Button.Key = "Cansel" Then

    Unload Doctor_Change
End If
End Sub
```

DEPARTMENT_FORM;

```
Private Sub fg_Department_Click()
    id = fg_Department.TextMatrix(fg_Department.Row, 0)
    dept_id = fg_Department.TextMatrix(fg_Department.Row, 1)
    dept_name = fg_Department.TextMatrix(fg_Department.Row, 2)
    dept_pay = fg_Department.TextMatrix(fg_Department.Row, 3)
    tx_Dept_Id = dept_id
    tx_Dept_Name = dept_name
    tx_Dept_Payment = dept_pay

    tb_Dept.Buttons.Item(1).Enabled = False
    tb_Dept.Buttons.Item(2).Enabled = False
    tb_Dept.Buttons.Item(3).Enabled = True
    tb_Dept.Buttons.Item(4).Enabled = True
    tb_Dept.Buttons.Item(5).Enabled = True
    tb_Dept.Buttons.Item(6).Enabled = True

End Sub

Private Sub Form_Load()

    Tab_Dept.TabIndex = 0
    Set mydata = OpenDatabase("C:\Hastane.mdb")
    st_sql = "select * from Department where Department_Id like '" & "E" & "*' "
    Set mytable = mydata.OpenRecordset(st_sql)
    GRID_DUZENLEME
    fg_Department.ColWidth(0) = 1000
    fg_Department.ColWidth(1) = 2000
    fg_Department.ColWidth(2) = 4000
    fg_Department.ColWidth(3) = 2000

    'database baglanti kurulur ve bilgler gride aktarilir.


    While Not mytable.EOF
```

```
        fg_Department.AddItem mytable.Fields("ID") & Chr(9) &
mytable.Fields("Department_Id") & Chr(9) & mytable.Fields("Department_Name") &
Chr(9) & mytable.Fields("Payment_Of_Department")
        mytable.MoveNext
    Wend
    mytable.Close
    mydata.Close

    'Arrange grid
     toolbar_new


End Sub


Private Sub Tab_Dept_Click(PreviousTab As Integer)
   If Tab_Dept.Tab = 0 Then

      Set mydata = OpenDatabase("C:\Hastane.mdb")
      st_sql = "select * from Department where Department_Id like '" & "E" & "*' "
      Set mytable = mydata.OpenRecordset(st_sql)
      GRID_DUZENLEME


    'database baglanti kurulur ve bilgler gride aktarilir.

      While Not mytable.EOF
      fg_Department.AddItem mytable.Fields("ID") & Chr(9) &
mytable.Fields("Department_Id") & Chr(9) & mytable.Fields("Department_Name") &
Chr(9) & mytable.Fields("Payment_Of_Department")
      mytable.MoveNext
      Wend
      mytable.Close
      mydata.Close


    ElseIf Tab_Dept.Tab = 1 Then

      Set mydata = OpenDatabase("C:\Hastane.mdb")
      st_sql = "select * from Department where Department_Id like '" & "D" & "*'"
      Set mytable = mydata.OpenRecordset(st_sql)
      GRID_DUZENLEME


     While Not mytable.EOF
      fg_Department.AddItem mytable.Fields("ID") & Chr(9) &
mytable.Fields("Department_Id") & Chr(9) & mytable.Fields("Department_Name") &
Chr(9) & mytable.Fields("Payment_Of_Department")
      mytable.MoveNext
      Wend
```

```
        mytable.Close
        mydata.Close

    ElseIf Tab_Dept.Tab = 2 Then

        Set mydata = OpenDatabase("C:\Hastane.mdb")
        st_sql = "select * from Department where Department_Id like '" & "C" & "*'"
        Set mytable = mydata.OpenRecordset(st_sql)
        GRID_DUZENLEME

        While Not mytable.EOF
        fg_Department.AddItem mytable.Fields("ID") & Chr(9) &
mytable.Fields("Department_Id") & Chr(9) & mytable.Fields("Department_Name") &
Chr(9) & mytable.Fields("Payment_Of_Department")
        mytable.MoveNext
        Wend
        mytable.Close
        mydata.Close

    ElseIf Tab_Dept.Tab = 3 Then

        Set mydata = OpenDatabase("C:\Hastane.mdb")
        st_sql = "select * from Department where Department_Id like '" & "N" & "*'"
        Set mytable = mydata.OpenRecordset(st_sql)
        GRID_DUZENLEME

        While Not mytable.EOF
        fg_Department.AddItem mytable.Fields("ID") & Chr(9) &
mytable.Fields("Department_Id") & Chr(9) & mytable.Fields("Department_Name") &
Chr(9) & mytable.Fields("Payment_Of_Department")
        mytable.MoveNext
        Wend
        mytable.Close
        mydata.Close

    ElseIf Tab_Dept.Tab = 4 Then

        Set mydata = OpenDatabase("C:\Hastane.mdb")
        st_sql = "select * from Department where Department_Id like '" & "I" & "*'"
        Set mytable = mydata.OpenRecordset(st_sql)
        GRID_DUZENLEME

        While Not mytable.EOF
        fg_Department.AddItem mytable.Fields("ID") & Chr(9) &
mytable.Fields("Department_Id") & Chr(9) & mytable.Fields("Department_Name") &
Chr(9) & mytable.Fields("Payment_Of_Department")
        mytable.MoveNext
        Wend
        mytable.Close
        mydata.Close
```

```
    ElseIf Tab_Dept.Tab = 5 Then

      Set mydata = OpenDatabase("C:\Hastane.mdb")
      st_sql = "select * from Department where Department_Id like '" & "O" & "*'"
      Set mytable = mydata.OpenRecordset(st_sql)
      GRID_DUZENLEME

      While Not mytable.EOF
      fg_Department.AddItem mytable.Fields("ID") & Chr(9) &
mytable.Fields("Department_Id") & Chr(9) & mytable.Fields("Department_Name") &
Chr(9) & mytable.Fields("Payment_Of_Department")
      mytable.MoveNext
      Wend
      mytable.Close
      mydata.Close

    End If


End Sub

Private Sub tb_Dept_ButtonClick(ByVal Button As ComctlLib.Button)

  If Button.Key = "New" Then

        tx_Dept_Id.SetFocus
        tb_Dept.Buttons.Item(1).Enabled = False
        tb_Dept.Buttons.Item(2).Enabled = True
        tb_Dept.Buttons.Item(3).Enabled = False
        tb_Dept.Buttons.Item(4).Enabled = False
        tb_Dept.Buttons.Item(5).Enabled = True
        tb_Dept.Buttons.Item(6).Enabled = True

  ElseIf Button.Key = "Save" Then

      If tx_Dept_Id = "" Or tx_Dept_Name = "" Or tx_Dept_Payment = "" Then
            MsgBox ("Please,fill the blanks")
      Else
            If IsNumeric(tx_Dept_Payment) Then
              result = MsgBox("Do you want to save?", 33)
              If result = 1 Then
                Set mydata = OpenDatabase("C:\Hastane")
                Set mytable = mydata.OpenRecordset("Department")
                mytable.AddNew
                mytable.Fields("Department_Id") = UCase(tx_Dept_Id)
                mytable.Fields("Department_Name") = UCase(tx_Dept_Name)
                mytable.Fields("Payment_Of_Department") = tx_Dept_Payment
                On Error GoTo err_found
                mytable.Update
```

```
                yazdirma
                tx_Dept_Id.Text = ""
                tx_Dept_Name.Text = ""
                tx_Dept_Payment.Text = ""
              Else
                tx_Dept_Id = ""
                tx_Dept_Name = ""
                tx_Dept_Payment = ""
              End If
            Else
              MsgBox ("please enter the number in DEPT_PAYMENT")
              tx_Dept_Payment.SetFocus

            End If


        End If

    ElseIf Button.Key = "Edit" Then

        If tx_Dept_Id = "" Or tx_Dept_Name = "" Or tx_Dept_Payment = "" Then
            MsgBox ("Please,fill the blanks")
        Else
            If IsNumeric(tx_Dept_Payment) Then
                result = MsgBox("do ou want to edit?", 33)
                If result = 1 Then
                    Set mydata = OpenDatabase("C:\Hastane")
                    st_sql = "Update Department Set Department_Name= '" &
tx_Dept_Name & "',Payment_Of_Department='" & tx_Dept_Payment & "' where ID="
& Val(id) & " "
                    mydata.Execute (st_sql)
                    yazdirma
                Else
                    tx_Dept_Id = ""
                    tx_Dept_Name = ""
                    tx_Dept_Payment = ""
                End If
            Else
              MsgBox ("enter number in DEPT_PAYMENT")
            End If
             toolbar_new
        End If

    ElseIf Button.Key = "Delete" Then

       result = MsgBox("Do you want to delete?", 33)
       If result = 1 Then
         Set mydata = OpenDatabase("C:\Hastane.mdb")
         Set mytable = mydata.OpenRecordset("Department")
         st_sql = "delete from Department where ID=" & id & " "
```

```
            mydata.Execute (st_sql)
            yazdirma
            tx_Dept_Id = ""
            tx_Dept_Name = ""
            tx_Dept_Payment = ""
          Else
            tx_Dept_Id = ""
            tx_Dept_Name = ""
            tx_Dept_Payment = ""
          End If
          toolbar_new

    ElseIf Button.Key = "Report" Then

          DataReport5.Show

    ElseIf Button.Key = "Close" Then

          Unload Department_Ana

    End If


err_found:
  Select Case Err.Number
    Case 3022
      MsgBox ("var")
  End Select


End Sub

Public Sub yazdirma()
      fg_Department.FixedCols = 0
      fg_Department.FixedRows = 1
      fg_Department.Cols = 4
      fg_Department.Rows = 1
      fg_Department.Cols = 4
      fg_Department.Rows = 1
      fg_Department.Clear
      fg_Department.Row = 0
      fg_Department.Col = 0
      fg_Department.Text = "ID"
      fg_Department.Row = 0
      fg_Department.Col = 1
      fg_Department.Text = "Department_Id"
      fg_Department.Row = 0
      fg_Department.Col = 2
      fg_Department.Text = "Department_Name"
      fg_Department.Row = 0
```

```
fg_Department.Col = 3
fg_Department.Text = "Payment_Of_Department"


If  Tab_Dept.Tab = 0  Then
   st_sql = "select * from Department where Department_Id like '" & "E" & "*' "
   Set mytable1 = mydata.OpenRecordset(st_sql)
     While Not mytable1.EOF
      fg_Department.AddItem    Str(mytable1.Fields("ID"))    &    Chr(9)    &
mytable1.Fields("Department_Id") & Chr(9) & mytable1.Fields("Department_Name")
& Chr(9) & mytable1.Fields("Payment_Of_Department")
     mytable1.MoveNext
     Wend
   mytable1.Close


ElseIf Tab_Dept.Tab = 1 Then
   st_sql = "select * from Department where Department_Id like '" & "D" & "*' "
   Set mytable1 = mydata.OpenRecordset(st_sql)
     While Not mytable1.EOF
      fg_Department.AddItem    Str(mytable1.Fields("ID"))    &    Chr(9)    &
mytable1.Fields("Department_Id") & Chr(9) & mytable1.Fields("Department_Name")
& Chr(9) & mytable1.Fields("Payment_Of_Department")
     mytable1.MoveNext
     Wend

   mytable1.Close


ElseIf Tab_Dept.Tab = 2 Then
   st_sql = "select * from Department where Department_Id like '" & "C" & "*' "
   Set mytable1 = mydata.OpenRecordset(st_sql)
     While Not mytable1.EOF
      fg_Department.AddItem    Str(mytable1.Fields("ID"))    &    Chr(9)    &
mytable1.Fields("Department_Id") & Chr(9) & mytable1.Fields("Department_Name")
& Chr(9) & mytable1.Fields("Payment_Of_Department")
     mytable1.MoveNext
     Wend

   mytable1.Close


ElseIf Tab_Dept.Tab = 3 Then
   st_sql = "select * from Department where Department_Id like '" & "N" & "*' "
   Set mytable1 = mydata.OpenRecordset(st_sql)
     While Not mytable1.EOF
      fg_Department.AddItem    Str(mytable1.Fields("ID"))    &    Chr(9)    &
mytable1.Fields("Department_Id") & Chr(9) & mytable1.Fields("Department_Name")
& Chr(9) & mytable1.Fields("Payment_Of_Department")
```

```
        mytable1.MoveNext
        Wend

      mytable1.Close


    ElseIf Tab_Dept.Tab = 4 Then
      st_sql = "select * from Department where Department_Id like '" & "I" & "*' "
      Set mytable1 = mydata.OpenRecordset(st_sql)
        While Not mytable1.EOF
          fg_Department.AddItem  Str(mytable1.Fields("ID"))  &  Chr(9)  &
mytable1.Fields("Department_Id") & Chr(9) & mytable1.Fields("Department_Name")
& Chr(9) & mytable1.Fields("Payment_Of_Department")
        mytable1.MoveNext
        Wend

      mytable1.Close


    ElseIf Tab_Dept.Tab = 5 Then
    st_sql = "select * from Department where Department_Id like '" & "O" & "*' "
      Set mytable1 = mydata.OpenRecordset(st_sql)
        While Not mytable1.EOF
          fg_Department.AddItem  Str(mytable1.Fields("ID"))  &  Chr(9)  &
mytable1.Fields("Department_Id") & Chr(9) & mytable1.Fields("Department_Name")
& Chr(9) & mytable1.Fields("Payment_Of_Department")
        mytable1.MoveNext
        Wend

      mytable1.Close

      End If


End Sub

Public Sub GRID_DUZENLEME()

    fg_Department.FixedCols = 0
    fg_Department.FixedRows = 0
    fg_Department.Cols = 4
    fg_Department.Rows = 1
    fg_Department.Clear
    fg_Department.Row = 0
    fg_Department.Col = 0
    fg_Department.Text = "ID"
    fg_Department.Row = 0
    fg_Department.Col = 1
    fg_Department.Text = "Department_Id"
```

```
    fg_Department.Row = 0
    fg_Department.Col = 2
    fg_Department.Text = "Department_Name"
    fg_Department.Row = 0
    fg_Department.Col = 3
    fg_Department.Text = "Payment_Of_Department"

End Sub

Private Sub tx_Dept_Id_KeyPress(KeyAscii As Integer)
 If KeyAscii = 13 Then
  tx_Dept_Name.SetFocus
 End If
End Sub

Private Sub tx_Dept_Name_KeyPress(KeyAscii As Integer)
 If KeyAscii = 13 Then
  tx_Dept_Payment.SetFocus
 End If
End Sub

Private Sub tx_Dept_Payment_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
If tx_Dept_Id = "" Or tx_Dept_Name = "" Or tx_Dept_Payment = "" Then
            MsgBox ("Please,fill the blanks")
        Else
            If IsNumeric(tx_Dept_Payment) Then
                result = MsgBox("Do you want to save?", 33)
                If result = 1 Then
                    Set mydata = OpenDatabase("C:\Hastane")
                    Set mytable = mydata.OpenRecordset("Department")
                    mytable.AddNew
                    mytable.Fields("Department_Id") = UCase(tx_Dept_Id)
                    mytable.Fields("Department_Name") = UCase(tx_Dept_Name)
                    mytable.Fields("Payment_Of_Department") = tx_Dept_Payment
                    On Error GoTo err_found
                    mytable.Update
                    yazdirma
                    tx_Dept_Id.Text = ""
                    tx_Dept_Name.Text = ""
                    tx_Dept_Payment.Text = ""
                 Else
                    tx_Dept_Id = ""
                    tx_Dept_Name = ""
                    tx_Dept_Payment = ""
                 End If
            Else
                MsgBox ("please enter the number in DEPT_PAYMENT")
            End If
        End If
```

```
        End If

err_found:
   Select Case Err.Number
   Case 3022
   MsgBox ("There is record in database")
   End Select
End Sub



Public Sub toolbar_new()
      tb_Dept.Buttons.Item(1).Enabled = True
      tb_Dept.Buttons.Item(2).Enabled = False
      tb_Dept.Buttons.Item(3).Enabled = False
      tb_Dept.Buttons.Item(4).Enabled = False
      tb_Dept.Buttons.Item(5).Enabled = True
      tb_Dept.Buttons.Item(6).Enabled = True
End Sub
```

SOCIETY FORM;

```
Private Sub fg_Society_Click()
     id = fg_Society.TextMatrix(fg_Society.Row, 0)
     Society_Name = fg_Society.TextMatrix(fg_Society.Row, 1)
     tb_Society.Buttons.Item(1).Enabled = False
     tb_Society.Buttons.Item(2).Enabled = True
     tb_Society.Buttons.Item(3).Enabled = True
     tb_Society.Buttons.Item(4).Enabled = False
End Sub

Private Sub Form_Load()

   'Gridin düzenlenmesi

   fg_Society.FixedRows = 1
   fg_Society.FixedCols = 0
   fg_Society.Cols = 2
   fg_Society.Rows = 1
   fg_Society.Row = 0
   fg_Society.Col = 0
   fg_Society.Text = "ID_Number"
   fg_Society.Row = 0
   fg_Society.Col = 1
   fg_Society.Text = "Society_Name"
   fg_Society.ColWidth(0) = 500
   fg_Society.ColWidth(1) = 3000

   'Gride düzenleme
   Set mydata = OpenDatabase("C:\Hastane.mdb")
```

```
    st_sql = "select * from Society"
    Set mytable = mydata.OpenRecordset(st_sql)


    While Not mytable.EOF
      fg_Society.AddItem Str(mytable.Fields("ID")) & Chr(9) & _
                         mytable.Fields("Society_Name")
    mytable.MoveNext
    Wend
    mytable.Close
    mydata.Close
    'Arranging toolbar
    tb_Society.Buttons.Item(1).Enabled = True
    tb_Society.Buttons.Item(2).Enabled = False
    tb_Society.Buttons.Item(3).Enabled = False
    tb_Society.Buttons.Item(4).Enabled = True


End Sub

Private Sub tb_Society_ButtonClick(ByVal Button As ComctlLib.Button)

 If Button.Key = "New" Then
    Society_Saving.Show (modal)
 ElseIf Button.Key = "Edit" Then
       Society_Changing.Show (modal)
       Society_Changing.tx_ID_Number = id
       Society_Changing.tx_Society_Name = Society_Name
       tb_Society.Buttons.Item(1).Enabled = True
       tb_Society.Buttons.Item(2).Enabled = False
       tb_Society.Buttons.Item(3).Enabled = False
       tb_Society.Buttons.Item(4).Enabled = True
  ElseIf Button.Key = "Delete" Then
       j = MsgBox("Do you want to delete?", 33)
         If j = 1 Then
           Set mydata = OpenDatabase("C:\Hastane.mdb")
           Set mytable = mydata.OpenRecordset("Sosyal_Kurumlar")
           st_sql = "delete from Sosyal_Kurumlar where ID=" & id & " "
           mydata.Execute (st_sql)
           fg_Society.FixedRows = 1
           fg_Society.FixedCols = 0
           fg_Society.Cols = 2
           fg_Society.Rows = 1
           fg_Society.Clear
           fg_Society.Row = 0
           fg_Society.Col = 0
           fg_Society.Text = "ID Number"
           fg_Society.Col = 1
           fg_Society.Text = "Society_Name"
```

```
        While Not mytable.EOF
           Society.fg_Society.AddItem Str(mytable.Fields("ID")) & Chr(9) & _
                                      mytable.Fields("Society_Name")
           mytable.MoveNext
        Wend
        mytable.Close
        mydata.Close

      Else: Society.Show (modal)
      End If
      tb_Society.Buttons.Item(1).Enabled = True
      tb_Society.Buttons.Item(2).Enabled = False
      tb_Society.Buttons.Item(3).Enabled = False
      tb_Society.Buttons.Item(4).Enabled = True


  ElseIf Button.Key = "Close" Then
    Unload Society
  ElseIf Button.Key = "Report" Then
    DataReport6.Show
  End If


End Sub

Private Sub tx_Ara_Change()
    fg_Society.Clear
    fg_Society.FixedRows = 1
    fg_Society.FixedCols = 0
    fg_Society.Cols = 2
    fg_Society.Rows = 1
    fg_Society.Row = 0
    fg_Society.Col = 0
    fg_Society.Text = "ID Number"
    fg_Society.Row = 0
    fg_Society.Col = 1
    fg_Society.Text = "Society_Name"
    Set mydata = OpenDatabase("C:\Hastane.mdb")
    st_sql = "select * from Sosyal_Kurumlar where Society_Name like '*" &
UCase(tx_Ara) & "*'"
    Set mytable = mydata.OpenRecordset(st_sql)
      While Not mytable.EOF
          fg_Society.AddItem Str(mytable.Fields("ID")) & Chr(9) & _
                             mytable.Fields("Society_Name")
                             mytable.MoveNext
      Wend
    mytable.Close
    mydata.Close
End Sub
```

PATIENT FORM;

```
Private Sub cb_Blood_Group_KeyPress(KeyAscii As Integer)
 If KeyAscii = 13 Then
   tx_Mobile_Phone.SetFocus
 End If
End Sub

Private Sub cb_Sex_KeyPress(KeyAscii As Integer)
 If KeyAscii = 13 Then
   tx_Occupation.SetFocus
 End If
End Sub

Private Sub cb_Society_KeyPress(KeyAscii As Integer)
 If KeyAscii = 13 Then
   tx_Adress.SetFocus
 End If
End Sub

Private Sub cm_All_Records_Click()
  Grid_Düzenleme
  YAZDIR
End Sub

Private Sub fg_Patient_Click()
   'Grid'in içindekileri bilgileri degiskenlere aktarama

   code = fg_Patient.TextMatrix(fg_Patient.Row, 0)
   identity_card = fg_Patient.TextMatrix(fg_Patient.Row, 1)
   Name_Surname = fg_Patient.TextMatrix(fg_Patient.Row, 2)
   name_of_father = fg_Patient.TextMatrix(fg_Patient.Row, 3)
   date_of_birth = fg_Patient.TextMatrix(fg_Patient.Row, 4)
   sex = fg_Patient.TextMatrix(fg_Patient.Row, 5)
   occupation = fg_Patient.TextMatrix(fg_Patient.Row, 6)
   registration_day = fg_Patient.TextMatrix(fg_Patient.Row, 7)
   blood_group = fg_Patient.TextMatrix(fg_Patient.Row, 8)
   Home_Tel = fg_Patient.TextMatrix(fg_Patient.Row, 9)
   mobile = fg_Patient.TextMatrix(fg_Patient.Row, 10)
   Society = fg_Patient.TextMatrix(fg_Patient.Row, 11)
   adress = fg_Patient.TextMatrix(fg_Patient.Row, 12)

   EnableAyarlaAcik

   tx_Code = code
   tx_Identity_Card = identity_card
   tx_Name_Surname = Name_Surname
   tx_Name_Of_Father = name_of_father
   tx_Date_Of_Birth = date_of_birth
```

```
  cb_Sex = sex
  tx_Occupation = occupation
  tx_Registration_Day = registration_day
  cb_Blood_Group = blood_group
  tx_Home_Tel = Home_Tel
  tx_Mobile_Phone = Mobile_Phone
  cb_Society = Society
  tx_Adress = adress

  tb_Patient.Buttons.Item(1).Enabled = False
  tb_Patient.Buttons.Item(2).Enabled = False
  tb_Patient.Buttons.Item(3).Enabled = True
  tb_Patient.Buttons.Item(4).Enabled = True
  tb_Patient.Buttons.Item(5).Enabled = True
  tb_Patient.Buttons.Item(6).Enabled = False
  tb_Patient.Buttons.Item(7).Enabled = True

End Sub

Private Sub Form_Load()
  Grid_Düzenleme
  YAZDIR
  M_Birth.Visible = False
  tx_Code.Enabled = False
  tb_Patient.Buttons.Item(1).Enabled = True
  tb_Patient.Buttons.Item(2).Enabled = False
  tb_Patient.Buttons.Item(3).Enabled = False
  tb_Patient.Buttons.Item(4).Enabled = False
  tb_Patient.Buttons.Item(5).Enabled = True
  tb_Patient.Buttons.Item(6).Enabled = True
  tb_Patient.Buttons.Item(7).Enabled = True

  EnableAyarlaKapali
  cb_Sex.Clear
  cb_Sex.AddItem "Male"
  cb_Sex.AddItem "Female"
  cb_Blood_Group.Clear
  cb_Blood_Group.AddItem "A RH+"
  cb_Blood_Group.AddItem "A RH-"
  cb_Blood_Group.AddItem "B RH+"
  cb_Blood_Group.AddItem "B RH-"
  cb_Blood_Group.AddItem "AB RH+"
  cb_Blood_Group.AddItem "AB RH-"
  cb_Blood_Group.AddItem "0 RH+"
  cb_Blood_Group.AddItem "0 RH-"
  cb_Society.Clear
  cb_Society.AddItem "SSK"
  cb_Society.AddItem "EMEKLI SANDIGI"
  cb_Society.AddItem "YESIL KART"
  cb_Society.AddItem "BAGKUR"
```

```
  tx_Identity_Card = ""
  tx_Name_Surname = ""
  tx_Name_Of_Father = ""
  tx_Date_Of_Birth = ""
  cb_Sex = ""
  tx_Occupation = ""
  tx_Registration_Day = ""
  cb_Blood_Group = ""
  tx_Home_Tel = ""
  tx_Mobile_Phone = ""
  cb_Society = ""
  tx_Adress = ""
End Sub
Private Sub M_Birth_DateClick(ByVal DateClicked As Date)
  tx_Date_Of_Birth.Text = M_Birth.Value
  M_Birth.Visible = False
End Sub

Private Sub tb_Patient_ButtonClick(ByVal Button As ComctlLib.Button)
   If Button.Key = "New" Then

      EnableAyarlaAcik
      tx_Registration_Day = Date
      tx_Identity_Card.SetFocus
      tb_Patient.Buttons.Item(1).Enabled = False
      tb_Patient.Buttons.Item(2).Enabled = True
      tb_Patient.Buttons.Item(3).Enabled = False
      tb_Patient.Buttons.Item(4).Enabled = False
      tb_Patient.Buttons.Item(5).Enabled = True
      tb_Patient.Buttons.Item(6).Enabled = False
      tb_Patient.Buttons.Item(7).Enabled = True

   ElseIf Button.Key = "Save" Then

      tx_Code.Enabled = False
         If tx_Identity_Card.Text <> "" And _
           tx_Name_Surname.Text <> "" And _
           tx_Name_Of_Father.Text <> "" And _
           tx_Date_Of_Birth.Text <> "" And _
           tx_Mobile_Phone.Text <> "" And _
           tx_Adress.Text <> "" Then
           result = MsgBox("do you want to save", 33)
              If result = 1 Then
                'New datas records in database
                Set mydata = OpenDatabase("C:\Hastane")
                Set mytable = mydata.OpenRecordset("Patient")
                mytable.AddNew
                mytable.Fields("Identity_Number") = tx_Identity_Card
                mytable.Fields("Name_Surname") = UCase(tx_Name_Surname)
```

```
  mytable.Fields("Name_Of_Father") = UCase(tx_Name_Of_Father)
  mytable.Fields("Date_Of_Birth") = tx_Date_Of_Birth
  mytable.Fields("Sex") = UCase(cb_Sex.Text)
  mytable.Fields("Occupation") = UCase(tx_Occupation)
  mytable.Fields("Registration_Day") = tx_Registration_Day
  mytable.Fields("Blood_Group") = cb_Blood_Group.Text
  mytable.Fields("Home_Tel") = tx_Home_Tel
  mytable.Fields("Mobile_Phone") = tx_Mobile_Phone
  mytable.Fields("Society") = UCase(cb_Society.Text)
  mytable.Fields("Adress") = UCase(tx_Adress)
  mytable.Update
  Grid_Düzenleme
  YAZDIR

  tx_Identity_Card = ""
  tx_Name_Surname = ""
  tx_Name_Of_Father = ""
  tx_Date_Of_Birth = ""
  cb_Sex = ""
  tx_Occupation = ""
  tx_Registration_Day = ""
  cb_Blood_Group = ""
  tx_Home_Tel = ""
  tx_Mobile_Phone = ""
  cb_Society = ""
  tx_Adress = ""
  tx_Identity_Card.Enabled = False
  EnableAyarlaKapali

  'Arrange toolbar
  tb_Patient.Buttons.Item(1).Enabled = True
  tb_Patient.Buttons.Item(2).Enabled = False
  tb_Patient.Buttons.Item(3).Enabled = False
  tb_Patient.Buttons.Item(4).Enabled = False
  tb_Patient.Buttons.Item(5).Enabled = True
  tb_Patient.Buttons.Item(6).Enabled = True
  tb_Patient.Buttons.Item(7).Enabled = True
Else
  tx_Identity_Card = ""
  tx_Name_Surname = ""
  tx_Name_Of_Father = ""
  tx_Date_Of_Birth = ""
  cb_Sex = ""
  tx_Occupation = ""
  tx_Registration_Day = ""
  cb_Blood_Group = ""
  tx_Home_Tel = ""
  tx_Mobile_Phone = ""
  cb_Society = ""
  tx_Adress = ""
```

```
            End If
        ElseIf tx_Identity_Card.Text = "" Then
            MsgBox ("enter,identity card number")
            tx_Identity_Card.SetFocus
        ElseIf tx_Name_Surname.Text = "" Then
            MsgBox ("enter,name surname")
            tx_Name_Surname.SetFocus
        ElseIf tx_Name_Of_Father.Text = "" Then
            MsgBox ("enter,father name")
            tx_Name_Of_Father.SetFocus
        ElseIf tx_Date_Of_Birth.Text = "" Then
            MsgBox ("enter birthday")
            tx_Date_Of_Birth.SetFocus
        ElseIf tx_Mobile_Phone.Text = "" Then
            MsgBox ("emter ,mobile phone")
            tx_Mobile_Phone.SetFocus
        ElseIf tx_Adress.Text = "" Then
            MsgBox ("enter,adress")
            tx_Adress.SetFocus
        End If

    ElseIf Button.Key = "Edit" Then

        If tx_Identity_Card.Text <> "" And _
          tx_Name_Surname.Text <> "" And _
          tx_Name_Of_Father.Text <> "" And _
          tx_Date_Of_Birth.Text <> "" And _
          tx_Mobile_Phone.Text <> "" And _
          tx_Adress.Text <> "" Then
          result = MsgBox("Do you want to Edit", 33)
            If result = 1 Then
              Set mydata = OpenDatabase("C:\Hastane")
              st_sql = "update Patient set Identity_Number='" & tx_Identity_Card
& "',Name_Surname='" & UCase(tx_Name_Surname) & "',Name_Of_Father='" &
UCase(tx_Name_Of_Father) & "',Date_Of_Birth='" & tx_Date_Of_Birth & "',Sex='" &
UCase(cb_Sex) & "',Occupation='" & UCase(tx_Occupation) & "',Registration_Day='"
& UCase(tx_Registration_Day) & "',Blood_Group='" & UCase(cb_Blood_Group) &
"',Home_Tel='" & tx_Home_Tel & "',Mobile_Phone='" & tx_Mobile_Phone &
"',Society='" & UCase(cb_Society) & "',Adress='" & UCase(tx_Adress) & "' where
Code_Of_Patient=" & code & ""
              mydata.Execute (st_sql)
              mydata.Close
              Grid_Düzenleme
              YAZDIR
              tx_Code = ""
              tx_Identity_Card = ""
              tx_Name_Surname = ""
              tx_Name_Of_Father = ""
              tx_Date_Of_Birth = ""
              cb_Sex = ""
```

```
        tx_Occupation = ""
        tx_Registration_Day = ""
        cb_Blood_Group = ""
        tx_Home_Tel = ""
        tx_Mobile_Phone = ""
        cb_Society = ""
        tx_Adress = ""
        EnableAyarlaKapali

        'Arranging toolbar

        tb_Patient.Buttons.Item(1).Enabled = True
        tb_Patient.Buttons.Item(2).Enabled = False
        tb_Patient.Buttons.Item(3).Enabled = False
        tb_Patient.Buttons.Item(4).Enabled = False
        tb_Patient.Buttons.Item(5).Enabled = True
        tb_Patient.Buttons.Item(6).Enabled = True
        tb_Patient.Buttons.Item(7).Enabled = True
      Else
        tx_Identity_Card = ""
        tx_Name_Surname = ""
        tx_Name_Of_Father = ""
        tx_Date_Of_Birth = ""
        cb_Sex = ""
        tx_Occupation = ""
        tx_Registration_Day = ""
        cb_Blood_Group = ""
        tx_Home_Tel = ""
        tx_Mobile_Phone = ""
        cb_Society = ""
        tx_Adress = ""
        EnableAyarlaKapali
      End If
ElseIf tx_Identity_Card.Text = "" Then
   MsgBox ("enter,identity card number")
   tx_Identity_Card.SetFocus
ElseIf tx_Name_Surname.Text = "" Then
   MsgBox ("enter,name surname")
   tx_Name_Surname.SetFocus
ElseIf tx_Name_Of_Father.Text = "" Then
   MsgBox ("enter,father name")
   tx_Name_Of_Father.SetFocus
ElseIf tx_Date_Of_Birth.Text = "" Then
   MsgBox ("enter birthday")
   tx_Date_Of_Birth.SetFocus
ElseIf tx_Mobile_Phone.Text = "" Then
   MsgBox ("emter ,mobile phone")
   tx_Mobile_Phone.SetFocus
ElseIf tx_Adress.Text = "" Then
   MsgBox ("enter,adress")
```

```
            tx_Adress.SetFocus
        End If

ElseIf Button.Key = "Delete" Then

    result = MsgBox("do you want to delete?", 33)
        If result = 1 Then
            Set mydata = OpenDatabase("C:\Hastane")
            st_sql = "delete from patient where Code_Of_Patient=" & code & ""
            mydata.Execute (st_sql)
            Grid_Düzenleme
            YAZDIR
            tx_Code = ""
            tx_Identity_Card = ""
            tx_Name_Surname = ""
            tx_Name_Of_Father = ""
            tx_Date_Of_Birth = ""
            cb_Sex = ""
            tx_Occupation = ""
            tx_Registration_Day = ""
            cb_Blood_Group = ""
            tx_Home_Tel = ""
            tx_Mobile_Phone = ""
            cb_Society = ""
            tx_Adress = ""
            EnableAyarlaKapali
            'Arranging toolbar

            tb_Patient.Buttons.Item(1).Enabled = True
            tb_Patient.Buttons.Item(2).Enabled = False
            tb_Patient.Buttons.Item(3).Enabled = False
            tb_Patient.Buttons.Item(4).Enabled = False
            tb_Patient.Buttons.Item(5).Enabled = True
            tb_Patient.Buttons.Item(6).Enabled = True
            tb_Patient.Buttons.Item(7).Enabled = True

        Else

            tx_Identity_Card = ""
            tx_Name_Surname = ""
            tx_Name_Of_Father = ""
            tx_Date_Of_Birth = ""
            cb_Sex = ""
            tx_Occupation = ""
            tx_Registration_Day = ""
            cb_Blood_Group = ""
            tx_Home_Tel = ""
            tx_Mobile_Phone = ""
            cb_Society = ""
            tx_Adress = ""
```

```
            EnableAyarlaKapali

          End If

    ElseIf Button.Key = "Search" Then

      Patient_Search.Show (modal)

    ElseIf Button.Key = "Close" Then

      Unload Patient

    ElseIf Button.Key = "Report" Then

      DataReport3.Show

    End If

    End Sub

Public Sub YAZDIR()
    'Show data on the grid from database

    Set mydata = OpenDatabase("C:\Hastane.mdb")
    st_sql = "select * from Patient"
    Set mytable = mydata.OpenRecordset(st_sql)


    While Not mytable.EOF
       fg_Patient.AddItem Str(mytable.Fields("Code_Of_Patient")) & Chr(9) & _
                          mytable.Fields("Identity_Number") & Chr(9) & _
                          mytable.Fields("Name_Surname") & Chr(9) & _
                          mytable.Fields("Name_Of_Father") & Chr(9) & _
                          mytable.Fields("Date_Of_Birth") & Chr(9) & _
                          mytable.Fields("Sex") & Chr(9) & _
                          mytable.Fields("Occupation") & Chr(9) & _
                          mytable.Fields("Registration_Day") & Chr(9) & _
                          mytable.Fields("Blood_Group") & Chr(9) & _
                          mytable.Fields("Home_Tel") & Chr(9) & _
                          mytable.Fields("Mobile_Phone") & Chr(9) & _
                          mytable.Fields("Society") & Chr(9) & _
                          mytable.Fields("Adress")
                          mytable.MoveNext
    Wend
    mytable.Close
    mydata.Close

End Sub
```

```
Private Sub tx_Adress_KeyPress(KeyAscii As Integer)
    If KeyAscii = 13 Then
      If tx_Identity_Card.Text <> "" And _
        tx_Name_Surname.Text <> "" And _
        tx_Name_Of_Father.Text <> "" And _
        tx_Date_Of_Birth.Text <> "" And _
        tx_Mobile_Phone.Text <> "" And _
        tx_Adress.Text <> "" Then
        result = MsgBox("do you want to save", 33)
            If result = 1 Then
              Set mydata = OpenDatabase("C:\Hastane")
              Set mytable = mydata.OpenRecordset("Patient")
              mytable.AddNew
              mytable.Fields("Identity_Number") = tx_Identity_Card
              mytable.Fields("Name_Surname") = UCase(tx_Name_Surname)
              mytable.Fields("Name_Of_Father") = UCase(tx_Name_Of_Father)
              mytable.Fields("Date_Of_Birth") = tx_Date_Of_Birth
              mytable.Fields("Sex") = UCase(cb_Sex.Text)
              mytable.Fields("Occupation") = UCase(tx_Occupation)
              mytable.Fields("Registration_Day") = tx_Registration_Day
              mytable.Fields("Blood_Group") = cb_Blood_Group.Text
              mytable.Fields("Home_Tel") = tx_Home_Tel
              mytable.Fields("Mobile_Phone") = tx_Mobile_Phone
              mytable.Fields("Society") = UCase(cb_Society.Text)
              mytable.Fields("Adress") = UCase(tx_Adress)
              mytable.Update
              Grid_Düzenleme
              YAZDIR

              tx_Identity_Card = ""
              tx_Name_Surname = ""
              tx_Name_Of_Father = ""
              tx_Date_Of_Birth = ""
              cb_Sex = ""
              tx_Occupation = ""
              tx_Registration_Day = ""
              cb_Blood_Group = ""
              tx_Home_Tel = ""
              tx_Mobile_Phone = ""
              cb_Society = ""
              tx_Adress = ""
              tx_Identity_Card.Enabled = False
              EnableAyarlaKapali
              'Arranging toolbar

              tb_Patient.Buttons.Item(1).Enabled = True
              tb_Patient.Buttons.Item(2).Enabled = False
              tb_Patient.Buttons.Item(3).Enabled = False
              tb_Patient.Buttons.Item(4).Enabled = False
              tb_Patient.Buttons.Item(5).Enabled = True
```

```
                tb_Patient.Buttons.Item(6).Enabled = True
                tb_Patient.Buttons.Item(7).Enabled = True
              Else
                tx_Identity_Card = ""
                tx_Name_Surname = ""
                tx_Name_Of_Father = ""
                tx_Date_Of_Birth = ""
                cb_Sex = ""
                tx_Occupation = ""
                tx_Registration_Day = ""
                cb_Blood_Group = ""
                tx_Home_Tel = ""
                tx_Mobile_Phone = ""
                cb_Society = ""
                tx_Adress = ""

              End If
        ElseIf tx_Identity_Card.Text = "" Then
                MsgBox ("enter,identity card number")
                tx_Identity_Card.SetFocus
        ElseIf tx_Name_Surname.Text = "" Then
                MsgBox ("enter,name surname")
                tx_Name_Surname.SetFocus
        ElseIf tx_Name_Of_Father.Text = "" Then
                MsgBox ("enter,father name")
                tx_Name_Of_Father.SetFocus
        ElseIf tx_Date_Of_Birth.Text = "" Then
                MsgBox ("enter birthday")
                tx_Date_Of_Birth.SetFocus
        ElseIf tx_Mobile_Phone.Text = "" Then
                MsgBox ("emter ,mobile phone")
                tx_Mobile_Phone.SetFocus
        ElseIf tx_Adress.Text = "" Then
                MsgBox ("enter,adress")
                tx_Adress.SetFocus
        End If
      End If
End Sub

Private Sub tx_Code_KeyPress(KeyAscii As Integer)
  If KeyAscii = 13 Then
    tx_Identity_Card.SetFocus
  End If

End Sub

Private Sub tx_Date_Of_Birth_Change()
   M_Birth.Visible = False
End Sub
```

```
Private Sub tx_Date_Of_Birth_Click()
   M_Birth.Visible = True
End Sub

Private Sub tx_Date_Of_Birth_KeyPress(KeyAscii As Integer)
 If KeyAscii = 13 Then
    cb_Sex.SetFocus
 End If
End Sub


Private Sub tx_Home_Tel_KeyPress(KeyAscii As Integer)
 If KeyAscii = 13 Then
   cb_Society.SetFocus
 End If
End Sub

Private Sub tx_Identity_Card_KeyPress(KeyAscii As Integer)
   If KeyAscii = 13 Then
     tx_Name_Surname.SetFocus
   End If
End Sub

Private Sub tx_Mobile_Phone_KeyPress(KeyAscii As Integer)
 If KeyAscii = 13 Then
    tx_Home_Tel.SetFocus
 End If
End Sub


Private Sub tx_Name_Of_Father_KeyPress(KeyAscii As Integer)
 If KeyAscii = 13 Then
    tx_Date_Of_Birth.SetFocus
    M_Birth.Visible = True
 End If
End Sub

Private Sub tx_Name_Surname_KeyPress(KeyAscii As Integer)
 If KeyAscii = 13 Then
    tx_Name_Of_Father.SetFocus
 End If
End Sub


Private Sub tx_Occupation_KeyPress(KeyAscii As Integer)
 If KeyAscii = 13 Then
    tx_Registration_Day.SetFocus
 End If
End Sub
```

```
Private Sub tx_Registration_Day_KeyPress(KeyAscii As Integer)
  If KeyAscii = 13 Then
    cb_Blood_Group.SetFocus
  End If
End Sub

Public Sub EnableAyarlaAcik()
    tx_Identity_Card.Enabled = True
    tx_Name_Surname.Enabled = True
    tx_Name_Of_Father.Enabled = True
    tx_Date_Of_Birth.Enabled = True
    cb_Sex.Enabled = True
    tx_Occupation.Enabled = True
    tx_Registration_Day.Enabled = True
    cb_Blood_Group.Enabled = True
    tx_Home_Tel.Enabled = True
    tx_Mobile_Phone.Enabled = True
    cb_Society.Enabled = True
    tx_Adress.Enabled = True
End Sub

Public Sub EnableAyarlaKapali()
    tx_Identity_Card.Enabled = False
    tx_Name_Surname.Enabled = False
    tx_Name_Of_Father.Enabled = False
    tx_Date_Of_Birth.Enabled = False
    cb_Sex.Enabled = False
    tx_Occupation.Enabled = False
    tx_Registration_Day.Enabled = False
    cb_Blood_Group.Enabled = False
    tx_Home_Tel.Enabled = False
    tx_Mobile_Phone.Enabled = False
    cb_Society.Enabled = False
    tx_Adress.Enabled = False
End Sub

Public Sub Grid_Düzenleme()
    fg_Patient.FixedCols = 0
    fg_Patient.FixedRows = 1
    fg_Patient.Cols = 13
    fg_Patient.Rows = 1
    fg_Patient.Row = 0
    fg_Patient.Col = 0
    fg_Patient.Clear
    fg_Patient.Text = "Code"
    fg_Patient.Row = 0
    fg_Patient.Col = 1
```

```
    fg_Patient.Text = "Identity_Number"
    fg_Patient.Row = 0
    fg_Patient.Col = 2
    fg_Patient.Text = "Name_Surname"
    fg_Patient.Row = 0
    fg_Patient.Col = 3
    fg_Patient.Text = "Name_Of_Father"
    fg_Patient.Row = 0
    fg_Patient.Col = 4
    fg_Patient.Text = "Date_Of_Birth"
    fg_Patient.Row = 0
    fg_Patient.Col = 5
    fg_Patient.Text = "Sex"
    fg_Patient.Row = 0
    fg_Patient.Col = 6
    fg_Patient.Text = "Occupation"
    fg_Patient.Row = 0
    fg_Patient.Col = 7
    fg_Patient.Text = "Registration_Day"
    fg_Patient.Row = 0
    fg_Patient.Col = 8
    fg_Patient.Text = "Blood_Group"
    fg_Patient.Row = 0
    fg_Patient.Col = 9
    fg_Patient.Text = "Home_Tel"
    fg_Patient.Row = 0
    fg_Patient.Col = 10
    fg_Patient.Text = "Mobile_Phone"
    fg_Patient.Row = 0
    fg_Patient.Col = 11
    fg_Patient.Text = "Society"
    fg_Patient.Row = 0
    fg_Patient.Col = 12
    fg_Patient.Text = "Adress"
    fg_Patient.ColWidth(0) = 1000
    fg_Patient.ColWidth(1) = 2000
    fg_Patient.ColWidth(2) = 3000
    fg_Patient.ColWidth(3) = 3000
    fg_Patient.ColWidth(4) = 2000
    fg_Patient.ColWidth(5) = 2000
    fg_Patient.ColWidth(6) = 2000
    fg_Patient.ColWidth(7) = 2000
    fg_Patient.ColWidth(8) = 2000
    fg_Patient.ColWidth(9) = 2000
    fg_Patient.ColWidth(10) = 2000
    fg_Patient.ColWidth(11) = 2000
    fg_Patient.ColWidth(12) = 9000

End Sub
```

APPOINTMENT FORM

```
Private Sub cb_Clean_Click()
  tx_Start.Text = ""
  tx_End.Text = ""
End Sub

Private Sub cb_Search_Click()
  Grid_Düzenleme
  Set mydata = OpenDatabase("C:\Hastane.mdb")
  st_sql = "select * from Appointment where Appointment_Date between '" & tx_Start
& "' and '" & tx_End & "'"
  Set mytable = mydata.OpenRecordset(st_sql)
  Grid_Yazdirma
End Sub

Private Sub fg_Appointment_Click()

 'Transerfer datas in grid to variables
 appointment_id = fg_Appointment.TextMatrix(fg_Appointment.Row, 0)
 patient_name = fg_Appointment.TextMatrix(fg_Appointment.Row, 1)
 department = fg_Appointment.TextMatrix(fg_Appointment.Row, 2)
 doctor_name = fg_Appointment.TextMatrix(fg_Appointment.Row, 3)
 appointment_date = fg_Appointment.TextMatrix(fg_Appointment.Row, 4)
 appointment_time = fg_Appointment.TextMatrix(fg_Appointment.Row, 5)
 'Arranging Toolbar
 tb_Appointment.Buttons.Item(1).Enabled = False
 tb_Appointment.Buttons.Item(2).Enabled = True
 tb_Appointment.Buttons.Item(3).Enabled = True
 tb_Appointment.Buttons.Item(4).Enabled = True
 tb_Appointment.Buttons.Item(5).Enabled = True
End Sub

Private Sub Form_Load()
   Grid_Düzenleme

   tb_Appointment.Buttons.Item(1).Enabled = True
   tb_Appointment.Buttons.Item(2).Enabled = False
   tb_Appointment.Buttons.Item(3).Enabled = False
   tb_Appointment.Buttons.Item(4).Enabled = True
   tb_Appointment.Buttons.Item(5).Enabled = True

   M_Start.Visible = False
   M_End.Visible = False
   M_Date.Visible = False
   'gride gunun randevularini yukler

   Set mydata = OpenDatabase("C:\Hastane.mdb")
   st_sql = "select * from Appointment where Appointment_Date='" & Date & "'"
   Set mytable = mydata.OpenRecordset(st_sql)
   While Not mytable.EOF
```

```
    St_sql1 = "select * from Patient where Code_Of_Patient=" &
Val(mytable.Fields("Patient_ID")) & ""
    Set mytable1 = mydata.OpenRecordset(St_sql1)
      While Not mytable1.EOF
        name1 = mytable1.Fields("Name_Surname")
        mytable1.MoveNext
      Wend
      mytable1.Close
  fg_Appointment.AddItem mytable.Fields("Appointment_ID") _
  & Chr(9) & name1 _
  & Chr(9) & mytable.Fields("Department") _
  & Chr(9) & mytable.Fields("Doctor_Name") _
  & Chr(9) & mytable.Fields("Appointment_Date") _
  & Chr(9) & mytable.Fields("Appointment_Time")
  mytable.MoveNext
  Wend

  mytable.Close
  mydata.Close

  End Sub

Private Sub Toolbar1_ButtonClick(ByVal Button As ComctlLib.Button)

End Sub

Private Sub M_Search_DateClick(ByVal DateClicked As Date)

tx_Start.Text = M_Search.Value
M_Search.Visible = False


End Sub

Private Sub fr_Appointment_DragDrop(Source As Control, X As Single, Y As Single)

End Sub

Private Sub M_Date_DateClick(ByVal DateClicked As Date)
tx_Date = M_Date.Value
M_Date.Visible = False
End Sub

Private Sub M_End_DateClick(ByVal DateClicked As Date)
tx_End = M_End.Value
M_End.Visible = False
End Sub

Private Sub M_Start_DateClick(ByVal DateClicked As Date)
```

```
tx_Start = M_Start.Value
M_Start.Visible = False
End Sub

Private Sub tb_Appointment_ButtonClick(ByVal Button As ComctlLib.Button)

   If Button.Key = "New" Then

      Patient_Find.Show (modal)

   ElseIf Button.Key = "Edit" Then

      Appointment_Change.Show (modal)
      Appointment_Change.tx_Appointment_ID = appointment_id
      Appointment_Change.tx_Patient_Name = patient_name
      Appointment_Change.cb_Department = department
      Appointment_Change.cb_Doctor = doctor_name
      Appointment_Change.tx_Appointment_Date = appointment_date
      Appointment_Change.tx_Appointment_time = appointment_time

   ElseIf Button.Key = "Delete" Then

      j = MsgBox("Do you want to delete?", 33)
         If j = 1 Then
            Set mydata = OpenDatabase("C:\Hastane.mdb")
            Set mytable = mydata.OpenRecordset("Appointment")
            st_sql = "delete from Appointment where Appointment_ID=" &
Val(appointment_id) & " "
            mydata.Execute (st_sql)
            mytable.Close
            mydata.Close

            Grid_Düzenleme

            tb_Appointment.Buttons.Item(1).Enabled = True
            tb_Appointment.Buttons.Item(2).Enabled = False
            tb_Appointment.Buttons.Item(3).Enabled = False
            tb_Appointment.Buttons.Item(4).Enabled = True
            tb_Appointment.Buttons.Item(5).Enabled = True

            'Appointment of daily install in grid

            Set mydata = OpenDatabase("C:\Hastane.mdb")
            st_sql = "select * from Appointment where Appointment_Date='" & Date &
"'"
            Set mytable = mydata.OpenRecordset(st_sql)
            While Not mytable.EOF
               St_sql1 = "select * from Patient where Code_Of_Patient=" &
Val(mytable.Fields("Patient_ID")) & ""
               Set mytable1 = mydata.OpenRecordset(St_sql1)
```

```
                While Not mytable1.EOF
                    name1 = mytable1.Fields("Name_Surname")
                    mytable1.MoveNext
                Wend
            fg_Appointment.AddItem mytable.Fields("Appointment_ID") _
                        & Chr(9) & name1 _
                        & Chr(9) & mytable.Fields("Department") _
                        & Chr(9) & mytable.Fields("Doctor_Name") _
                        & Chr(9) & mytable.Fields("Appointment_Date") _
                        & Chr(9) & mytable.Fields("Appointment_Time")
                        mytable.MoveNext
            Wend
            mytable1.Close
            mytable.Close
            mydata.Close
           Else
            Unload Appointment_Change
           End If

    ElseIf Button.Key = "Close" Then

    Unload Appointment
    ElseIf Button.Key = "Report" Then
    DataReport7.Show

    End If


End Sub

Private Sub tx_Date_Change()
  Grid_Düzenleme
  Set mydata = OpenDatabase("C:\Hastane.mdb")
  st_sql = "select * from Appointment where Appointment_Date='" & tx_Date & "'"
  Set mytable = mydata.OpenRecordset(st_sql)
  Grid_Yazdirma
End Sub

Private Sub tx_Date_Click()
  M_Date.Visible = True
End Sub

Private Sub tx_Doctor_Name_Change()
 Grid_Düzenleme
 Set mydata = OpenDatabase("C:\Hastane.mdb")
 st_sql = "select * from Appointment where Doctor_Name='" & tx_Doctor_Name &
"'"
 Set mytable = mydata.OpenRecordset(st_sql)
 Grid_Yazdirma
```

```
End Sub

Private Sub tx_End_Click()
  M_End.Visible = True
End Sub

Private Sub tx_Patient_ID_Change()
  Grid_Düzenleme
  Set mydata = OpenDatabase("C:\Hastane")
  st_sql = "select * from Appointment where Patient_ID=" & Val(tx_Patient_ID) & ""
  Set mytable = mydata.OpenRecordset(st_sql)
  Grid_Yazdirma
End Sub

Private Sub tx_Patient_Name_Change()
  Set mydata = OpenDatabase("C:\Hastane.mdb")
  st_sql = "select * from Patient where Name_Surname='" & tx_Patient_Name & "'"
  Set mytable = mydata.OpenRecordset(st_sql)
     While Not mytable.EOF
        code1 = mytable.Fields("Code_Of_Patient")
        mytable.MoveNext
     Wend
  Grid_Düzenleme
  st_sql = "select * from Appointment where Patient_ID=" & Val(code1) & ""
  Set mytable = mydata.OpenRecordset(st_sql)
  Grid_Yazdirma

End Sub

Private Sub tx_Start_Click()
  M_Start.Visible = True
End Sub

Public Sub Grid_Düzenleme()

    fg_Appointment.FixedCols = 0
    fg_Appointment.FixedRows = 1
    fg_Appointment.Cols = 7
    fg_Appointment.Rows = 2
    fg_Appointment.Col = 0
    fg_Appointment.Row = 0
    fg_Appointment.Clear
    fg_Appointment.Text = "Appointment_ID"
    fg_Appointment.Col = 1
    fg_Appointment.Row = 0
    fg_Appointment.Text = "Patient_Name"
    fg_Appointment.Col = 2
    fg_Appointment.Row = 0
    fg_Appointment.Text = "Department"
    fg_Appointment.Col = 3
```

```
    fg_Appointment.Row = 0
    fg_Appointment.Text = "Doctor_Name"
    fg_Appointment.Col = 4
    fg_Appointment.Row = 0
    fg_Appointment.Text = "Appointment_Date"
    fg_Appointment.Col = 5
    fg_Appointment.Row = 0
    fg_Appointment.Text = "Appointment_Time"

    fg_Appointment.ColWidth(0) = 2000
    fg_Appointment.ColWidth(1) = 2500
    fg_Appointment.ColWidth(2) = 2500
    fg_Appointment.ColWidth(3) = 2000
    fg_Appointment.ColWidth(4) = 2000
    fg_Appointment.ColWidth(5) = 2000

End Sub

Public Sub Grid_Yazdirma()
     While Not mytable.EOF
      St_sql1 = "select * from Patient where Code_Of_Patient=" &
Val(mytable.Fields("Patient_ID")) & ""
      Set mytable1 = mydata.OpenRecordset(St_sql1)
        While Not mytable1.EOF
          name1 = mytable1.Fields("Name_Surname")
          mytable1.MoveNext
        Wend
        mytable1.Close
    fg_Appointment.AddItem mytable.Fields("Appointment_ID") _
    & Chr(9) & name1 _
    & Chr(9) & mytable.Fields("Department") _
    & Chr(9) & mytable.Fields("Doctor_Name") _
    & Chr(9) & mytable.Fields("Appointment_Date") _
    & Chr(9) & mytable.Fields("Appointment_Time")
    mytable.MoveNext
    Wend
    mytable.Close
    mydata.Close


End Sub
```

EXAMINATION FORM;

```
Private Sub fg_Examination_Click()

    examination_id = fg_Examination.TextMatrix(fg_Examination.Row, 0)
    patient_name1 = fg_Examination.TextMatrix(fg_Examination.Row, 1)
    department_name = fg_Examination.TextMatrix(fg_Examination.Row, 2)
```

```
    doctor_name1 = fg_Examination.TextMatrix(fg_Examination.Row, 3)
    Examination_Date = fg_Examination.TextMatrix(fg_Examination.Row, 4)
    tension = fg_Examination.TextMatrix(fg_Examination.Row, 5)
    pulse = fg_Examination.TextMatrix(fg_Examination.Row, 6)
    temperature = fg_Examination.TextMatrix(fg_Examination.Row, 7)
    other_finding = fg_Examination.TextMatrix(fg_Examination.Row, 8)
    diagnosis = fg_Examination.TextMatrix(fg_Examination.Row, 9)
    medicines = fg_Examination.TextMatrix(fg_Examination.Row, 10)
    essential_investigation = fg_Examination.TextMatrix(fg_Examination.Row, 11)



    tb_Examination.Buttons.Item(1).Enabled = False
    tb_Examination.Buttons.Item(2).Enabled = True
    tb_Examination.Buttons.Item(3).Enabled = True
    tb_Examination.Buttons.Item(4).Enabled = True
    tb_Examination.Buttons.Item(5).Enabled = True

End Sub

Private Sub Form_Load()

    Grid_Düzenleme
    tb_Examination.Buttons.Item(1).Enabled = True
    tb_Examination.Buttons.Item(2).Enabled = False
    tb_Examination.Buttons.Item(3).Enabled = False
    tb_Examination.Buttons.Item(4).Enabled = True
    tb_Examination.Buttons.Item(5).Enabled = True

    M_Examination.Visible = False
    Set mydata = OpenDatabase("C:\Hastane.mdb")
    st_sql = "select * from Examination where Examination_Date='" & Date & "'"
    Set mytable = mydata.OpenRecordset(st_sql)
    While Not mytable.EOF
      St_sql1 = "select * from Patient where Code_Of_Patient=" &
Val(mytable.Fields("Patient_ID")) & ""
      st_sql2 = "select * from Doctor where ID=" & Val(mytable.Fields("Doctor_ID"))
& ""
      st_sql3 = "select * from Specialize where S_ID=" &
Val(mytable.Fields("Department_ID")) & ""
      Set mytable1 = mydata.OpenRecordset(St_sql1)
        While Not mytable1.EOF
          patient_name = mytable1.Fields("Name_Surname")
          mytable1.MoveNext
        Wend
        mytable1.Close
      Set mytable2 = mydata.OpenRecordset(st_sql2)
        While Not mytable2.EOF
          doctor_name = mytable2.Fields("Name_Surname")
```

```
        mytable2.MoveNext
      Wend
      mytable2.Close
    Set mytable3 = mydata.OpenRecordset(st_sql3)
      While Not mytable3.EOF
        department_name = mytable3.Fields("Specialize_Name")
        mytable3.MoveNext
      Wend
      mytable3.Close



  fg_Examination.AddItem Val(mytable.Fields("Examination_ID")) _
                & Chr(9) & patient_name _
                & Chr(9) & department_name _
                & Chr(9) & doctor_name _
                & Chr(9) & mytable.Fields("Examination_Date") _
                & Chr(9) & mytable.Fields("Tension") _
                & Chr(9) & mytable.Fields("Pulse") _
                & Chr(9) & mytable.Fields("Temperature") _
                & Chr(9) & mytable.Fields("Other_Finding") _
                & Chr(9) & mytable.Fields("Diagnosis") _
                & Chr(9) & mytable.Fields("Medicines") _
                & Chr(9) & mytable.Fields("Essential_Investigation")

  mytable.MoveNext
  Wend

  mytable.Close
  mydata.Close
End Sub



Private Sub M_Examination_DateClick(ByVal DateClicked As Date)
    tx_Date.Text = M_Examination.Value
    M_Examination.Visible = False
End Sub

Private Sub tb_Examination_ButtonClick(ByVal Button As ComctlLib.Button)

  If Button.Key = "New" Then
    Patient_Find.Show (modal)
  ElseIf Button.Key = "Edit" Then
    Examination_Change.Show (modal)
    Examination_Change.tx_Id = examination_id
    Examination_Change.tx_Patient_Name = patient_name1
    Examination_Change.cb_Doctor = doctor_name1
    Examination_Change.cb_Department = department_name
```

```
    Examination_Change.tx_Date = Examination_Date
    Examination_Change.tx_Diagnosis = diagnosis
    Examination_Change.tx_Medicines = medicines
    Examination_Change.tx_Other_Finding = other_finding
    Examination_Change.tx_Pulse = pulse
    Examination_Change.tx_Temperature = temperature
    Examination_Change.tx_Tension = tension



    Set mydata = OpenDatabase("C:\Hastane")
    st_sql = "select * from Patient where Name_Surname='" & patient_name1 & "'"
    Set mytable = mydata.OpenRecordset(st_sql)
    While Not mytable.EOF
        pat_id = mytable.Fields("Code_Of_Patient")
        mytable.MoveNext
    Wend
    mytable.Close
    st_sql = "select * from Specialize where Specialize_Name='" & department_name
& "'"
    Set mytable = mydata.OpenRecordset(st_sql)
    While Not mytable.EOF
        dept_id = mytable.Fields("S_ID")
        mytable.MoveNext
    Wend
    mytable.Close
    St_sql1 = "select  *  from Dept_Pat where Patient_ID=" & Val(pat_id) & " and
Examination_Date='" & Examination_Date & "' and Specialize_ID=" & Val(dept_id)
& ""
    Set mytable2 = mydata.OpenRecordset(St_sql1)
    While Not mytable2.EOF
        dept_id = mytable2.Fields("Department_ID")
        st_sql3 = "select * from Department where Id=" & dept_id & ""
     Set mytable3 = mydata.OpenRecordset(st_sql3)
        While Not mytable3.EOF
            Examination_Change.lb_Examination.AddItem
mytable3.Fields("Department_Name")
            mytable3.MoveNext
        Wend
        mytable3.Close
        mytable2.MoveNext
     Wend
     mytable2.Close
     mydata.Close

  ElseIf Button.Key = "Delete" Then

     j = MsgBox("Do you want to delete?", 33)
       If j = 1 Then
          Set mydata = OpenDatabase("C:\Hastane.mdb")
```

```
        st_sql = "select * from Patient where Name_Surname='" & patient_name1 &
"'"
        Set mytable = mydata.OpenRecordset(st_sql)
        While Not mytable.EOF
            pat_id = mytable.Fields("Code_Of_Patient")
            mytable.MoveNext
        Wend
        mytable.Close
        St_sql1 = "delete from Examination where Examination_ID=" &
Val(examination_id) & ""
        st_sql2 = "delete from Dept_Pat where Patient_ID=" & pat_id & " and
Examination_Date='" & Examination_Date & "'"
        mydata.Execute (St_sql1)
        mydata.Execute (st_sql2)
        mydata.Close

        Grid_Düzenleme
        Set mydata = OpenDatabase("C:\Hastane.mdb")
        st_sql = "select * from Examination where Examination_Date='" & Date &
"'"
        Set mytable = mydata.OpenRecordset(st_sql)
        While Not mytable.EOF
            St_sql1 = "select * from Patient where Code_Of_Patient=" &
Val(mytable.Fields("Patient_ID")) & ""
            st_sql2 = "select * from Doctor where ID=" &
Val(mytable.Fields("Doctor_ID")) & ""
            st_sql3 = "select * from Specialize where S_ID=" &
Val(mytable.Fields("Department_ID")) & ""
            Set mytable1 = mydata.OpenRecordset(St_sql1)
              While Not mytable1.EOF
                patient_name = mytable1.Fields("Name_Surname")
                mytable1.MoveNext
              Wend
              mytable1.Close
            Set mytable2 = mydata.OpenRecordset(st_sql2)
              While Not mytable2.EOF
                doctor_name = mytable2.Fields("Name_Surname")
                mytable2.MoveNext
              Wend
              mytable2.Close
            Set mytable3 = mydata.OpenRecordset(st_sql3)
              While Not mytable3.EOF
                department_name = mytable3.Fields("Specialize_Name")
                mytable3.MoveNext
              Wend
              mytable3.Close
```

```
                fg_Examination.AddItem Val(mytable.Fields("Examination_ID")) _
                & Chr(9) & patient_name _
                & Chr(9) & department_name _
                & Chr(9) & doctor_name _
                & Chr(9) & mytable.Fields("Examination_Date") _
                & Chr(9) & mytable.Fields("Tension") _
                & Chr(9) & mytable.Fields("Pulse") _
                & Chr(9) & mytable.Fields("Temperature") _
                & Chr(9) & mytable.Fields("Other_Finding") _
                & Chr(9) & mytable.Fields("Diagnosis") _
                & Chr(9) & mytable.Fields("Medicines") _
                & Chr(9) & mytable.Fields("Essential_Investigation")

            mytable.MoveNext
            Wend
            mytable.Close
            mydata.Close
          Else
             Examination.Show (modal)
          End If

  ElseIf Button.Key = "Close" Then

  Unload Examination
  ElseIf Button.Key = "Report" Then
  DataReport8.Show

  End If
End Sub

Public Sub Grid_Düzenleme()
fg_Examination.FixedCols = 0
fg_Examination.FixedRows = 1
fg_Examination.Cols = 12
fg_Examination.Rows = 2
fg_Examination.Clear
fg_Examination.Col = 0
fg_Examination.Row = 0
fg_Examination.Text = "Examination_ID"
fg_Examination.Col = 1
fg_Examination.Row = 0
fg_Examination.Text = "Patient_Name"
fg_Examination.Col = 2
fg_Examination.Row = 0
fg_Examination.Text = "Department_Name"
fg_Examination.Col = 3
fg_Examination.Row = 0
fg_Examination.Text = "Doctor_Name"
fg_Examination.Col = 4
fg_Examination.Row = 0
```

```
fg_Examination.Text = "Examination_Date"
fg_Examination.Col = 5
fg_Examination.Row = 0
fg_Examination.Text = "Tension"
fg_Examination.Col = 6
fg_Examination.Row = 0
fg_Examination.Text = "Pulse"
fg_Examination.Col = 7
fg_Examination.Row = 0
fg_Examination.Text = "Temperature"
fg_Examination.Col = 8
fg_Examination.Row = 0
fg_Examination.Text = "Other_Finding"
fg_Examination.Col = 9
fg_Examination.Row = 0
fg_Examination.Text = "Diagnosis"
fg_Examination.Col = 10
fg_Examination.Row = 0
fg_Examination.Text = "Medicines"
fg_Examination.Col = 11
fg_Examination.Row = 0
fg_Examination.Text = "Essential_Investigation"
fg_Examination.ColWidth(0) = 1000
fg_Examination.ColWidth(1) = 2000
fg_Examination.ColWidth(2) = 2000
fg_Examination.ColWidth(3) = 2300
fg_Examination.ColWidth(4) = 1000
fg_Examination.ColWidth(5) = 1000
fg_Examination.ColWidth(6) = 1000
fg_Examination.ColWidth(7) = 5500
fg_Examination.ColWidth(8) = 5500
fg_Examination.ColWidth(9) = 5500
fg_Examination.ColWidth(10) = 5500
fg_Examination.ColWidth(10) = 5500
End Sub

Public Sub Grid_Yazdirma()

   While Not mytable.EOF
     St_sql1 = "select * from Patient where Code_Of_Patient=" &
Val(mytable.Fields("Patient_ID")) & ""
     st_sql2 = "select * from Doctor where ID=" & Val(mytable.Fields("Doctor_ID"))
& ""
     st_sql3 = "select * from Specialize where S_ID=" &
Val(mytable.Fields("Department_ID")) & ""
     Set mytable1 = mydata.OpenRecordset(St_sql1)
       While Not mytable1.EOF
         patient_name = mytable1.Fields("Name_Surname")
         mytable1.MoveNext
       Wend
```

```
        mytable1.Close
      Set mytable2 = mydata.OpenRecordset(st_sql2)
        While Not mytable2.EOF
          doctor_name = mytable2.Fields("Name_Surname")
          mytable2.MoveNext
        Wend
        mytable2.Close
      Set mytable3 = mydata.OpenRecordset(st_sql3)
        While Not mytable3.EOF
          department_name = mytable3.Fields("Specialize_Name")
          mytable3.MoveNext
        Wend
        mytable3.Close
    fg_Examination.AddItem Val(mytable.Fields("Examination_ID")) _
                & Chr(9) & patient_name _
                & Chr(9) & department_name _
                & Chr(9) & doctor_name _
                & Chr(9) & mytable.Fields("Examination_Date") _
                & Chr(9) & mytable.Fields("Tension") _
                & Chr(9) & mytable.Fields("Pulse") _
                & Chr(9) & mytable.Fields("Temperature") _
                & Chr(9) & mytable.Fields("Other_Finding") _
                & Chr(9) & mytable.Fields("Diagnosis") _
                & Chr(9) & mytable.Fields("Medicines") _
                & Chr(9) & mytable.Fields("Essential_Investigation")

    mytable.MoveNext
    Wend

    mytable.Close
    mydata.Close

End Sub


Private Sub tx_Date_Change()
   Grid_Düzenleme
   Set mydata = OpenDatabase("C:\Hastane.mdb")
   st_sql = "select * from Examination where Examination_Date='" & tx_Date & "'"
   Set mytable = mydata.OpenRecordset(st_sql)
   Grid_Yazdirma
End Sub

Private Sub tx_Date_Click()
   M_Examination.Visible = True
End Sub

Private Sub tx_Doctor_Name_Change()
  Grid_Düzenleme
```

```
  Set mydata = OpenDatabase("C:\Hastane.mdb")
  St_sql1 = "select * from Doctor where Name_Surname='" & tx_Doctor_Name & "'"
  Set mytable1 = mydata.OpenRecordset(St_sql1)
  While Not mytable1.EOF
      pat_id = mytable1.Fields("ID")
      mytable1.MoveNext
  Wend
  mytable1.Close
  st_sql = "select * from Examination where Doctor_ID=" & Val(pat_id) & ""
  Set mytable = mydata.OpenRecordset(st_sql)
  Grid_Yazdirma
End Sub

Private Sub tx_Patient_ID_Change()
  Grid_Düzenleme
  Set mydata = OpenDatabase("C:\Hastane.mdb")
  st_sql = "select * from Examination where Patient_ID=" & Val(tx_Patient_ID) & ""
  Set mytable = mydata.OpenRecordset(st_sql)
  Grid_Yazdirma
End Sub

Private Sub tx_Patient_Name_Change()
  Grid_Düzenleme
  Set mydata = OpenDatabase("C:\Hastane.mdb")
  St_sql1 = "select * from Patient where Name_Surname='" & tx_Patient_Name & "'"
  Set mytable1 = mydata.OpenRecordset(St_sql1)
  While Not mytable1.EOF
      pat_id = mytable1.Fields("Code_Of_Patient")
      mytable1.MoveNext
  Wend
  mytable1.Close
  st_sql = "select * from Examination where Patient_ID=" & Val(pat_id) & ""
  Set mytable = mydata.OpenRecordset(st_sql)
  Grid_Yazdirma

End Sub
```