NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

HOSPITAL MANAGEMENT SYSTEM WITH DELPHI

Graduation Project
COM400

Student: Tuba KAYNAR (20012062)

Supervisor: Mr. Firudin MURADOV

Nicosia-2007

TABLE OF CONTENT

ACKNOWLEDGMENT

What is the succeed or What is the failure? In my opinion success is to be continue in struggle of life; it is not to be interrupt. Failure is to be hesitate or give up for determined aim that to be obtain perfect outcome. Otherwise it is not negative results that we encounter sometimes.

People must decide for an aim firstly. Besides it must be entirety and exactly. Here when the people did not decide or cannot decide; They disappear on the way that goes the aim, and they get the real failure.

There is no small aim or big aim. All aims are big for own who decided and planned it. As long as who can obtain the aim that is the outcome.

Here I decided an aim to on my own. Perhaps it get benefit for humanity. Yes perhaps it is not big project but it is beneficial project. And also I obtained success. Because I did not give up and I did not interrupt that. Here now I am in big pleasure.

I hope that this project gets benefit which is although small...

Also Well in this project I gained perseverence from ***Allah*** *and from my fiance I am grateful to my fiancee and all the people in my life who have supported me, advised me. They all the time helped and encouraged me to follow my dreams and ambitions.*

For intellectual support, encouragement I want to thank to my supervisor ***Mr. Firudin Muradov*** *who made this project contributions.*

And thank ***my dearest parents*** *who supported me to continue beyond my undergraduate studies, and also many thanks to my dear familiy who brought me till such meaning days.*

TUBA KAYNAR

ABSTRACT

The automation is also became a part of our lives. The people operate with automation systems in everywhere. Technology is entered to every platform of our life human needed to combine both software and hardware. Without software the machines are nothing. They need software to operate.

Human who tried to satisfy wonder.Such humanity came to nowadays as develop.Todays everyone says technology perfect developed.By means of technology all process gained velocity.This development acts to spend time to the people.

Doctors, yes doctors which are respectable people. Because they make an effort to heal for people health. But what a shame people do not share the same idea. Which can kill other people who can not adopt same opinion with other or who are not like with other. We owe many things for doctors. They strive day and night to give health people.

Thats why hospital project choosed to help for doctors, to decrease paper documentation time of a doctor. In this software doctor or user can keep patient informations, illnesses background information of the patient, operations background information of the patient,.With this software all informations will check easly, thus people will keep health.

INTRODUCTION

Do you imagine a life without computers. Especially a life without electricity. This imagine makes choke the people. Electricity also computers are part of human nowadays. Yes they are not real part like arms or legs but they are part like clothes; well they are indispensable. People tie to these parts day by day because which gives easiness and trustworthiness to them.

People adapt everything to computers because results of tie. Here I want to adapt health services to safe and rapid computer systems in this project.

In this project keep patients informations, illnesses informations, operation informations and sort them by patient id. These informations are investigated rapidly by program. Thats why the doctor obtain time to intersted in with patients.

The program provide a wide search criteria to users. Which property acts for all informations to be usable without null informations.

Now doctors will save null time that would be documentation and they interested in with patients as much more...

CHAPTER 1

DELPHI

1.1 INTRODUCTION TO DELPHI

Delphi is a communication structure aimed at producing detailed critical examination and discussion, not at forcing a quick compromise. Certainly quantification is a property, but only to serve the goal of quickly identifying agreement and disagreement in order to focus attention. It is often very common, even today, for people to come to a view of the Delphi method that reflects a particular application with which they are familiar. In 1975 Linstone and Turoff proposed a view of the Delphi method that they felt best summarized both the technique and its objective:

"Delphi may be characterized as a method for structuring a group communication process, so that the process is effective in allowing a group of individuals, as a whole, to deal with complex problems." The essence of Delphi is structuring of the group communication process. Given that there had been much earlier work on how to facilitate and structure face-to-face meetings, the other important distinction was that Delphi was commonly applied utilizing a paper and pencil communication process among groups in which the members were dispersed in space and time. Also, Delphis were commonly applied to groups of a size (30 to 100 individuals) that could not function well in a face-to-face environment, even if they could find a time when they all could get together.

The name "Delphi" was never a term with which either Olaf Helmer or Norman Dalkey (the founders of the method) were particular happy. Since many of the early Delphi studies focused on utilizing the technique to make forecasts of future occurrences, the name was first applied by some others at Rand as a joke. However, the name stuck. The resulting image of a priestess, sitting on a stool over a crack in the earth, inhaling sulfur fumes, and making vague and jumbled statements that could be interpreted in many different ways, did not exactly inspire confidence in the method.

The straightforward nature of utilizing an iterative survey to gather information "sounds" so easy to do that many people have done "one" Delphi, but never a second.

Since the name gives no obvious insight into the method and since the number of unsuccessful Delphi studies probably exceeds the successful ones, there has been a long history of diverse definitions and opinions about the method. Some of these misconceptions are expressed in statements such as the following that one finds in the literature:

It is a method for predicting future events.

It is a method for generating a quick consensus by a group.

It is the use of a survey to collect information.

It is the use of anonymity on the part of the participants.

It is the use of voting to reduce the need for long discussions.

It is a method for quantifying human judgement in a group setting.

Some of these statements are sometimes true; a few (e.g. consensus) are actually contrary to the purpose of a Delphi.

Additional opportunity has been added by the introduction of Computer Mediated Communication Systems (Hiltz and Turoff, 1978; Rice and Associates, 1984; Turoff, 1989; Turoff, 1991). These are computer systems that support group communications in either a synchronous (Group Decision Support Systems, Desanctis et. al., 1987) or an asynchronous manner (Computer Conferencing). Techniques that were developed and refined in the evolution of the Delphi Method (e.g. anonymity, voting) have been incorporated as basic facilities or tools in many of these computer based systems. As a result, any of these systems can be used to carry out some form of a Delphi process or Nominal Group Technique (Delbecq, et. al., 1975).

The result, however, is not merely confusion due to different names to describe the same things; but a basic lack of knowledge by many people working in these areas as to what was learned in the studies of the Delphi Method about how to properly employ these techniques and their impact on the communication process. There seems to be a great deal of "rediscovery" and repeating of earlier misconceptions and difficulties.

compiles the code you write and produces really tight, natively executable code for the target platform. In fact the most recent versions of Delphi optimise the compiled code and the resulting executables are as efficient as those compiled with any other compiler currently on the market.The term "visual" describes Delphi very well. All of the user interface development is conducted in a What You See Is What You Get environment (WYSIWYG), which means you can create polished, user friendly interfaces in a very short time, or prototype whole applications in a few hours.

1.3 DELPHI SYNTAX AND DELPHI STRUCTURE

Delphi is a Rapid Application Development (RAD) environment. It allows you to drag and drop components on to a blank canvas to create a program. Delphi will also allow you to use write console based DOS like programs.

Delphi is based around the Pascal language but is more developed object orientated derivative. Unlike Visual Basic, Delphi uses punctuation in its basic syntax to make the program easily readable and to help the compiler sort the code. Although Delphi code is not case sensitive there is a generally accepted way of writing Delphi code. The main reason for this is so that any programmer can read your code and easily understand what you are doing, because they write their code like you write yours.

For the purposes of this series I will be using Delphi 7. There are more recent versions available (2005 and 2006) however Delphi 7 should be available inexpensively compared to the new versions which will set you back a lot of money.

1.3.1 A Delphi Program

Notice the first line refers to the keyword program. You can rename this to HelloWorld. You can also remove the commented portion enclosed in curly brackets. The uses keyword allows you to list all units that you want to use in the program. At the moment just leave it as it is, SysUtils is all we need.

First thing is first, fire up your copy of Delphi and open the Project > Options menu. To compile a console application you need to change a setting on the Linker tab called 'Generate console application', check the box and click OK. Now select File > Close All if anything is already loaded. Then select File > New > Other > Console Application.

Your unit should now look like this:

Delphi Code:

```
program HelloWorld;

  {$APPTYPE CONSOLE}

 uses

  SysUtils;

 begin

 end.
```

Now what we have just done is written a program, it currently doesn't do a thing however. Hit the run button and see the result. Now wasn't that completely worthless. Luckily this isn't the end of the article so we'll actually have a worthwhile program at the end of it. All we need to do is insert some code in the main procedure we have just made.

Every good programmer's first program was 'Hello World' and you'll be no exception. All we need to do is use the WriteLn procedure to write 'Hello World!' to the console, simple.Notice the semicolon at the end of the line, at the end of any statement you need to add a semicolon. Run the program and see the results.

Delphi Code:

```
program HelloWorld;
 {$APPTYPE CONSOLE}
uses
  SysUtils;
begin
  WriteLn('Hello  World!' + #13#10 +
#13#10 +
   'Press RETURN to end...');
  ReadLn;
end.
```

1.3.2 Design Of Delphi

Coding style, the way you format your code and the way in which you present it on the page.At the end of the day who cares about my style, I can read it, and Delphi strips all the spaces out of it and doesn't care if I indent. Why waste my time?

Neatly present code which conforms to the accepted standards not only makes your code much easier for you to read and debug but also but any one else who might read your code to help you, or learn from you can do so with ease. After all which code is easier to follow, example 1 or 2?

Example 1:

```
procedure xyz();
 var  x,y,z,a:integer;
begin
x:=1;y:=2;
for z:=x to y do begin
a:=power(z,y);
showmessage(inttostr(a));
end;
end;
```

Example 2:

```
procedure XYZ();
 var X,Y,Z,A: Integer;
 begin
  X := 1;
  Y := 2;
  for Z := X to Y do
  begin
      A := Power(Z, Y);
      ShowMessage(IntToStr(A));
  end; // for end
end; // procedure end
```

Much of the ground-breaking work on design patterns was presented in the book Design Patterns: Elements of Reusable Object-Oriented Software by Gamma, Helm, Johnson and Vlissides. You might also have heard of the authors referred to as "the Gang of Four". If you haven't read this book before and you're designing objects, it's an excellent primer to help structure your design. To get the most out of these examples, I recommend reading the book as well.

Design patterns are frequently recurring structures and relationships in object-oriented design. Getting to know them can help you design better, more reusable code and also help you learn to design more complex systems.

Another good source of pattern concepts is the book Object Models: Strategies, Patterns and Applications by Peter Coad. Coad's examples are more business oriented and he emphasises learning strategies to identify patterns in your own work.

1.4 DELPHI CLASS DEFINITIONS

1.4.1 Unit Structure

Delphi units (.PAS files) allow declaration of interface and implementation sections. The interface defines the part that is visible to other units using that unit. The keyword

uses can be added to a unit's interface or implementation section to list the other units that your unit uses. This indicates to the compiler that your unit refers to parts of the used unit's interface. Parts of a unit declared in the implementation section are all private to that unit, i.e. never visible to any other unit. Types, functions and procedures declared in the interface of a unit must have a corresponding implementation, or be declared as external (e.g. a call to a function in a DLL).

1.4.2 Class Interfaces

Classes are defined as types in Delphi and may contain fields of standard data types or other objects, methods declared as functions or procedures, and properties. The type declaration of a class defines its interface and the scope of access to fields, methods and properties of the class. Class interfaces are usually defined in the interface of a unit to make them accessible to other modules using that unit. However they don't need to be. Sometimes a type declaration of a class may be used only within the implementation part of a unit.

1.4.3 Properties

Properties are a specialised interface to a field of a defined type, allowing access control through read and write methods. Properties are not virtual, you can replace a property with another property of the same name, but the parent class doesn't know about the new property. It is however possible to make the access methods of a property virtual.

1.4.4 Inheritance

Delphi's inheritance model is based on a single hierarchy. Every class inherits from TObject and can have only one parent.

A descendant class inherits all of the interface and functionality of its parent class, subject to the scope described below.

Multiple inheritance from more than one parent is not allowed directly. It can be implemented by using a container class to create instances one or more other classes and selectively expose parts of the contained classes.

Private, Protected, Public and Published ScopeScope refers to the visibility of methods and data defined in the interface of a class, i.e. what parts of the class are accessible to the rest of the application or to descendant classes.

The default scope is public, for instance the component instances you add to a form at design time. Public says "come and get me"; it makes the data or method visible to everything at runtime.

Published parts of a class are a specialized form of Public scope. They indicate special behaviour for classes derived from TPersistent. A persistent class can save and restore its published properties to persistent storage using Delphi's standard streaming methods. Published properties also interact with Delphi Object Inspector in the IDE. A class must descend from TPersistent in order to use Published. There's also not much point in publishing methods, since you can't store them, although Delphi's compiler doesn't stop you. Published also lets another application access details of the class through Delphi's runtime type information. This would be rarely used, except in Delphi's design time interaction with its VCL.

Encapsulation or information hiding is essential to object orientation, so Protected and Private scope let you narrow the access to parts of a class.Protected parts are visible only to descendant classes, or to other classes defined in the same unit.

Private parts are visible only to the defining class, or to other classes defined in the same unit.It's important to note that once something is given public or published scope, it cannot be hidden in descendant classes. Static, Virtual and Dynamic Methods; Override and Inherited. Methods declared as virtual or dynamic let you change their behaviour using override in a descendant class. You're unlikely to see a virtual method in the private part of a class, since it could only be overridden in the same unit, although Delphi's compiler doesn't stop you from doing this.

Override indicates that your new method replaces the method of the same name from the parent class. The override must be declared with the same name and parameters as the original method.When a method is overridden, a call to the parent class's method actually executes the override method in the real class of the object.Static methods on the other hand have no virtual or override declaration. You can replace a method of a class in a descendant class by redeclaring another method, however this is not object oriented. If you reference your descendant class as the parent type and try to call the replaced method, the static method of the parent class is executed. So in most cases, it's a bad idea to replace a static method.

Virtual and dynamic methods can be used interchangeably. They differ only in their treatment by the compiler and runtime library. Delphi's help explains that dynamic methods have their implementation resolved at compile time and run slightly faster, whereas virtual methods are resolved at runtime, resulting in slightly slower access but a smaller compiled program. Virtual is usually the preferred declaration. Delphi's help suggests using dynamic when you have a base class with many descendants that may not override the method.The inherited directive lets you refer back to a property or method as it was declared in the parent class. This is most often used in the implementation of an override method, to call the inherited method of the parent class and then supplement its behaviour.

1.4.5 Events

Events are also an important characteristic of Delphi, since they let you delegate extensible behaviour to instances of a class. Events are properties that refer to a method of another object. Events are not inherited in Delphi 1; Delphi 2 extends this behaviour to let you use inherited in an event. . Inherited in an event handler just uses the keyword inherited, there is no need to supply the name of the method to call.

Events are particularly important to component developers, since they provide a hook for the user of the component to modify its behaviour in a way that may not be foreseen at the time the component is written.

1.4.6 Messages

Delphi's handling of Windows messages is a special case of virtual methods. Message handlers are implemented in classes that descend from TControl. I.e classes that have a handle and can receive messages. Message handlers are always virtual and can be declared in the private part of a class interface, yet still allow the inherited method to be called. Inherited in a message handler just uses the keyword inherited, there is no need to supply the name of the method to call.

1.4.7 Constructors and Destructors

The constructor and destructor are two special types of methods. The constructor initializes a class instance (allocates memory initialized to 0) and returns a reference (pointer) to the object. The destructor deallocates memory used by the object (but not the memory of other objects created by the object).

Classes descended from TObject have a static constructor, Create, and a virtual destructor Destroy.TComponent introduces a new public property, the Owner of the component and this must be initialized in the constructor. TComponent's constructor is declared virtual, i.e. it can be overridden in descendant classes.It is essential when you override a virtual constructor or destructor in a TComponent descendant to include a call to the inherited method.

1.5 VISUAL COMPONENT LIBRARY

Applications Developers create complete applications by interacting with the Delphi visual environment (as mentioned earlier, this is a concept nonexistent in many other frameworks). These people use the VCL to create their user-interface and the other elements of their application: database connectivity, data validation, business rules, etc..

Applications Developers should know which properties, events, and methods each component makes available. Additionally, by understanding the VCL architecture,

Applications Developers will be able to easily identify where they can improve their applications by extending components or creating new ones. Then they can maximize the capabilities of these components, and create better applications.

1.5.1 The VCL to Component Creaters

Component Writers expand on the existing VCL, either by developing new components, or by increasing the functionality of existing ones. Many component writers make their components available for Applications Developers to use.

A Component Writer must take their knowledge of the VCL a step further than that of the Application Developer. For example, they must know whether to write a new component or to extend an existing one when the need for a certain characteristic arises. This requires a greater knowledge of the VCL's inner workings.

1.5.2 How Does Create the Component

Components are the building blocks that developers use to design the user-interface and to provide some non-visual capabilities to their applications. To an Application Developer, a component is an object most commonly dragged from the Component palette and placed onto a form. Once on the form, one can manipulate the component's properties and add code to the component's various events to give the component a specific behavior. To a Component Writer, components are objects in Object Pascal code. Some components encapsulate the behavior of elements provided by the system, such as the standard Windows 95 controls. Other objects introduce entirely new visual or non-visual elements, in which case the component's code makes up the entire behavior of the component.

The complexity of different components varies widely. Some might be simple while others might encapsulate a elaborate task. There is no limit to what a component can do or be made up of. You can have a very simple component like a TLabel, or a much more complex component which encapsulates the complete functionality of a spreadsheet.

1.5.3 Component Structure, Types and VCL hierarchy

Components are really just special types of objects. In fact, a component's structure is based on the rules that apply to Object Pascal. There are three fundamental keys to understanding the VCL.

First, you should know the special characteristics of the four basic component types: standard controls, custom controls, graphical controls and non-visual components.

Second, you must understand the VCL structure with which components are built. This really ties into your understanding of Object Pascal's implementation. Third, you should be familiar with the VCL hierarchy and you should also know where the four component types previously mentioned fit into the VCL hierarchy. The following paragraphs will discuss each of these keys to understanding the VCL.

1.5.4 Component Types

As a component writer, there four primary types of components that you will work with in Delphi: standard controls, custom controls, graphical controls, and non-visual components. Although these component types are primarily of interest to component writers, it's not a bad idea for applications developers to be familiar with them. They are the foundations on which applications are built.

1.5.4.1 Structure of a component

All components share a similar structure. Each component consists of common elements that allow developers to manipulate its appearance and function via properties, methods and events. The following sections in this paper will discuss these common elements as well as talk about a few other characteristics of components which don't apply to all components.

1.5.4.2 Component properties

Properties provide an extension of an object's fields. Unlike fields, properties do not store data: they provide other capabilities. For example, properties may use methods to read or write data to an object field to which the user has no access. This adds a certain level of protection as to how a given field is assigned data. Properties also cause "side effects" to occur when the user makes a particular assignment to the property. Thus what appears as a simple field assignment to the component user could trigger a complex operation to occur behind the scenes.

1.5.4.3 Standard Components

Some of the components provided by Delphi 2.0 encapsulate the behavior of the standard Windows controls: TButton, TListbox and Tedit, for example. You will find these components on the *Standard* page of the Component Palette. These components are Windows' common controls with Object Pascal wrappers around them.

Each standard component looks and works like the Windows' common control which it encapsulates. The VCL wrapper's simply makes the control available to you in the form of a Delphi component-it doesn't define the common control's appearance or functionality, but rather, surfaces the ability to modify a control's appearance/functionality in the form of methods and properties. If you have the VCL source code, you can examine how the VCL wraps these controls in the file STDCTRLS.PAS.

For example, the Windows class LISTBOX can display the list box items in multiple columns. This capability, however, isn't surfaced by Delphi's TListBox component (which encapsulates the Windows LISTBOX class). (TListBox only displays items in a single column.) Surfacing this capability requires that you override the default creation of the TListBox component.

1.5.4.4 Custom components

Unlike standard components, custom components are controls that don't already have a method for displaying themselves, nor do they have a defined behavior. The Component Writer must provide to code that tells the component how to draw itself and determines how the component behaves when the user interacts with it. Examples of existing custom components are the TPanel and TStringGrid components.

It should be mentioned here that both standard and custom components are *windowed* controls. A "windowed control" has a window associated with it and, therefore, has a window handle. Windowed controls have three characteristics: they can receive the input focus, they use system resources, and they can be parents to other controls. (Parents is related to containership, discussed later in this paper.) An example of a component which can be a container is the TPanel component.

1.5.4.5 Graphical components

Graphical components are visual controls which cannot receive the input focus from the user. They are non-windowed controls. Graphical components allow you to display something to the user without using up any system resources; they have less "overhead" than standard or custom components. Graphical components don't require a window handle-thus, they cannot can't get focus. Some examples of graphical components are the TLabel and TShape components.

1.5.4.6 Non-visual components

Non-visual components are components that do not appear on the form as controls at run-time. These components allow you to encapsulate some functionality of an entity within an object. You can manipulate how the component will behave, at design-time, through the Object Inspector. Using the Object Inspector, you can modify a non-visual component's properties and provide event handlers for its events. Examples of such components are the TOpenDialog, TTable, and TTimer components.

1.6 WHAT WILL MAKE WITH THE DELPHI?

The simple answer is "more or less anything". Because the code is compiled, it runs quickly, and is therefore suitable for writing more or less any program that you would consider a candidate for the Windows operating system.

You probably won't be using it to write embedded systems for washing machines, toasters or fuel injection systems, but for more or less anything else, it can be used (and the chances are that probably someone somewhere has!)

Some projects to which Delphi is suited:

- Simple, single user database applications
- Intermediate multi-user database applications
- Large scale multi-tier, multi-user database applications
- Internet applications
- Graphics Applications
- Multimedia Applications
- Image processing/Image recognition
- Data analysis
- System tools

This is not intended to be an exhaustive list, more an indication of the depth and breadth of Delphi's applicability. Because it is possible to access any and all of the Windows API, and because if all else fails, Delphi will allow you to drop a few lines of assembler code directly into your ordinary Pascal instructions, it is possible to do more or less anything. Delphi can also be used to write Dynamically Linked Libraries (DLLs) and can call out to DLLs written in other programming languages without difficulty.

Because Delphi is based on the concept of self contained Components (elements of code that can be dropped directly on to a form in your application, and exist in object form, performing their function until they are no longer required), it is possible to build applications very rapidly. Because Delphi has been available for quite some time, the number of pre-written components has been increasing to the point that now there is a component to do more or less anything you can imagine. The job of the programmer has become one of gluing together appropriate components with code that operates them as required.

1.7 DELPHI VERSIONS

In the very beginning, Windows produced SDKs (software development kits) that were totally non-visual (user interface development was totally separated from the development of the actual application), and required great patience and some genius to get anything working with. Whilst these tools slowly improved, they still required a really good understanding of the inner workings of Windows.

Borland (as they were then) has a long tradition in the creation of high speed compilers. One of their best known products was Turbo Pascal - a tool that many programmers cut their teeth on. With the rise in importance of the Windows environment, it was only a matter of time before development tools started to appear that were specific to this new environment.

To a great extent these criticisms were dispatched by the release of Microsoft's Visual Basic product, which attempted to bring Windows development to the masses. It achieved this to a great extent too, and remains a popular product today. However,it suffered from several drawbacks:

1) It wasn't as stable as it might have been

2) It was an interpreted language and hence was slow to run

3) It had as its underlying language BASIC, and most "real" programmers weren't so keen!

Into this environment arrived the eye opening Delphi I product, and in many ways the standard for visual development tools for Windows was set. This first version was a 16 bit compiler, and produced executable code that would run on Windows 3.1 and Windows 3.11. Of course, Microsoft have ensured (up to now) that their 32 bit operating systems (Win95, Win98, and Win NT) will all run 16 bit applications, however, many of the features that were introduced in these newer operating systems are not accessible to the 16 bit applications developed with Delphi I.

Delphi 2 was released quite soon after Delphi I, and in fact included a full distribution of Delphi I on the same CD. Delphi 2, (and all subsequent versions) have been 32 bit compilers, producing code that runs exclusively on 32bit Windows platforms. (We ignore for simplicity the WIN32S DLLs which allow Win 3.1x to run some 32 bit applications).

Delphi is currently standing at Version 4.0, with a new release (version 5.0) expected shortly. In its latest version, Delphi has become somewhat feature loaded, and as a result, we would argue, less stable than the earlier versions. However, in its defence, Delphi (and Borland products in general) have always been more stable than their competitors products, and the majority of Delphi 4's glitches are minor and forgivable - just don't try and copy/paste a selection of your code, midway through a debugging session!

The reasons for the version progression include the addition of new components, improvements in the development environment, the inclusion of more internet related support and improvements in the documentation. Delphi at version 4 is a very mature product, and Inprise has always been responsive in developing the product in the direction that the market requires it to go. Predominantly this means right now, the inclusion of more and more Internet, Web and CORBA related tools and components - a trend we are assured continues with the release of version 5.0

For each version of Delphi there are several sub-versions, varying in cost and features, from the most basic "Developer" version to the most complete (and expensive) "Client Server" version. The variation in price is substantial, and if you are contemplating a purchase, you should study the feature list carefully to ensure you are not paying for

features you will never use. Even the most basic "Developer" version contains the vast majority of the features you are likely to need on a day to day basis. Don't assume that you will need Client Server, simply because you are intending to write a large database application - The developer edition is quitcapable ofthis.

1.8 PROVIDE ACCESS TO STORAGE FIELDS

There are two ways that properties provide access to internal storage fields of components: directly or through access methods. Examine the code below which illustrates this process.

```
TCustomEdit = class(TWinControl)
private
  FMaxLength: Integer;
protected
  procedure SetMaxLength(Value: Integer);
...
published
  property MaxLength: Integer read
     FMaxLength write SetMaxLength default 0;
...
end;
```

The code above is snippet of the TCustomEdit component class. TCustomEdit is the base class for edit boxes and memo components such as TEdit, and TMemo.

TCustomEdit has an internal field FMaxLength of type Integer which specifies the maximum length of characters which the user can enter into the control. The user doesn't directly access the FMaxLength field to specify this value. Instead, a value is added to this field by making an assignment to the MaxLength property.

The property MaxLength provides the access to the storage field FMaxLength. The property definition is comprised of the property name, the property type, a read declaration, a write declaration and optional default value.

The read declaration specifies how the property is used to read the value of an internal storage field. For instance, the MaxLength property has direct read access to FMaxLength. The write declaration for MaxLength shows that assignments made to the MaxLength property result in a call to an *access method* which is responsible for assigning a value to the FMaxLength storage field. This access method is SetMaxLength.

1.8.1 Access methods property

Access methods take a single parameter of the same type as the property. One of the primary reasons for write access methods is to cause some side-effect to occur as a result of an assignment to a property. Write access methods also provide a method layer over assignments made to a component's fields. Instead of the component user making the assignment to the field directly, the property's write access method will assign the value to the storage field if the property refers to a particular storage field. For example, examine the implementation of the SetMaxLength method below.

```
procedure TCustomEdit.SetMaxLength(Value: Integer);

begin

 if FMaxLength <> Value then

 begin

  FMaxLength := Value;

  if HandleAllocated then

     SendMessage(Handle, EM_LIMITTEXT, Value, 0);
```

```
  end;

end;
```

The code in the SetMaxLength method checks if the user is assigning the same value as that which the property already holds. This is done as a simple optimization. The method then assigns the new value to the internal storage field, FMaxLength. Additionally, the method then sends an EM_LIMITTEXT Windows message to the window which the TCustomEdit encapsulates. The EM_LIMITTEXT message places a limit on the amount of text that a user can enter into an edit control. This last step is what is referred to as a *side-effect* when assigning property values. Side effects are any additional actions that occur when assigning a value to a property and can be quite sophisticated.

Providing access to internal storage fields through property access methods offers the advantage that the Component Writer can modify the implementation of a class without modifying the interface. It is also possible to have access methods for the read access of a property. The read access method might, for example, return a type which is different that that of a properties storage field. For instance, it could return the string representation of an integer storage field.

Another fundamental reason for properties is that properties are accessible for modification at run-time through Delphi's Object Inspector. This occurs whenever the declaration of the property appears in the published section of a component's declaration.

1.8.2 Types of properties

Properties can be of the standard data types defined by the Object Pascal rules. Property types also determine how they are edited in Delphi's Object Inspector. The table below shows the different property types as they are defined in Delphi's online help.

Property type	Object Inspector treatment
Simple	Numeric, character, and string properties appear in the Object Inspector as numbers, characters, and strings, respectively. The user can type and edit the value of the property directly.
Enumerated	Properties of enumerated types (including Boolean) display the value as defined in the source code. The user can cycle through the possible values by double-clicking the value column. There is also a drop-down list that shows all possible values of the enumerated type.
Set	Properties of set types appear in the Object Inspector looking like a set. By expanding the set, the user can treat each element of the set as a Boolean value: True if the element is included in the set or False if it's not included.
Object	Properties that are themselves objects often have their own property editors. However, if the object that is a property also has published properties, the Object Inspector allows the user to expand the list of object properties and edit them individually. Object properties must descend from TPersistent.
Array	Array properties must have their own property editors. The Object Inspector has no built-in support for editing array properties.

For more information on properties, refer to the "Component Writers Guide" which ships with Delphi.

1.8.3 Methods

Since components are really just objects, they can have methods. We will discuss some of the more commonly used methods later in this paper when we discuss the different levels of the VCL hierarchy.

1.8.4 Events

Events provide a means for a component to notify the user of some pre-defined occurrence within the component. Such an occurrence might be a button click or the pressing of a key on a keyboard.

Components contain special properties called events to which the component user assigns code. This code will be executed whenever a certain event occurs. For instance, if you look at the events page of a TEdit component, you'll see such events as OnChange, OnClick and OnDblClick. These events are nothing more than pointers to methods.

When the user of a component assigns code to one of those events, the user's code is referred to as an event handler. For example, by double clicking on the events page for a particular event causes Delphi to generate a method and places you in the Code Editor where you can add your code for that method. An example of this is shown in the code below, which is an OnClick event for a TButton component.

It becomes clearer that events are method pointers when you assign an event handler to an event programmatically. The above example was Delphi generated code. To link your own an event handler to a TButton's OnClick event at run time you must first create a method that you will assign to this event. Since this is a method, it must belong to an existing object. This object can be the form which owns the TButton component although it doesn't have to be. In fact, the event handlers which Delphi creates belong to the form on which the component resides. The code below illustrates how you would create an event handler method.

When you define methods for event handlers, these methods must be defined as the same type as the event property and the field to which the event property refers. For instance, the OnClick event refers to an internal data field, FOnClick. Both the property OnClick, and field FOnClick are of the type TNotifyEvent. TNotifyEvent is a procedural type as shown below:

```
TNotifyEvent = procedure (Sender: TObject) of object;
```

Note the use of the of object specification. This tells the compiler that the procedure definition is actually a method and performs some additional logic like ensuring that an implicit Self parameter is also passed to this method when called. Self is just a pointer reference to the class to which a method belongs.

1.8.5 Parenthood

Parenthood is a much different concept from ownership. It applies only to windowed components, which can be parents to other components. Later, when we discuss the VCL hierarchy, you will see the level in the hierarchy which introduces windowed controls.

Parent components are responsible for the display of other components. They call the appropriate methods internally that cause the children components to draw themselves. The Parent property of a component refers to the component which is its parent. Also, a component's parent does not have to be it's owner. Although the parent component is mainly responsible for the display of components, it also frees children components when it is destroyed.

Windowed components are controls which are visible user interface elements such as edit controls, list boxes and memo controls. In order for a windowed component to be displayed, it must be assigned a parent on which to display itself. This task is done automatically by Delphi's design-time environment when you drop a component from the Component Palette onto your form.

1.8.6 Containership

Some components in the VCL can own other components as well as be parents to other components. These two concepts have a different meaning as will be discussed in the section to follow.

1.8.7 Ownership

All components may be owned by other components but not all components can own other components. A component's Owner property contains a reference to the component which owns it.

The basic responsibility of the owner is one of resource management. The owner is responsible for freeing those components which it owns whenever it is destroyed. Typically, the form owns all components which appear on it, even if those components are placed on another component such as a TPanel. At design-time, the form automatically becomes the owner for components which you place on it. At run-time, when you create a component, you pass the owner as a parameter to the component's constructor. For instance, the code below shows how to create a TButton component at run-time and passes the form's implicit Self variable to the TButton's Create constructor. TButton.Create will then assign whatever is passed to it, in this case Self or rather the form, and assign it to the button's Owner property.

```
MyButton := TButton.Create(self);
```

When the form that now owns this TButton component gets freed, MyButton will also be freed.

You can create a component without an owner by passing nil to the component's Create constructor, however, you must ensure that the component is freed when it is no longer needed. The code below shows you how to do this for a TTable component.

CHAPTER 2

DATABASE

To identify anything system, actor or person in words we need a data or information.Every thing around us has a particular identity. So this information is valuable and in this advanced era we can store it in database and access this data by the blink of eye.For an instant if we go through the definitions of database we may find following definitions.

A database is a collection of related information. A database is an organized body of related information.

2.1 DATABASE MODELS

Various techniques are used to model data structures. Certain models are more easily implemented by some types of database management systems than others. For any one logical model various physical implementation may be possible. An example of this is the relational model: in larger systems the physical implementation often has indexes which point to the data; this is similar to some aspects of common implementations of the network model. But in small relational database the data is often stored in a set of files, one per table, in a flat, un-indexed structure. There is some confusion below and elsewhere in this article as to logical data model vs. its physical implementation.

2.1.1 Flat Model

The flat (or table) model consists of a single, two dimensional array of data elements, where all members of a given column are assumed to be similar values, and all members of a row are assumed to be related to one another. For instance, columns for name and password might be used as a part of a system security database. Each row would have the specific password associated with a specific user. Columns of the table

often have a type associated with them, defining them as character data, date or time information, integers, or floating point numbers. This model is the basis of the spreadsheet.

2.1.2 Network Model

The network model allows multiple datasets to be used together through the use of pointers (or references). Some columns contain pointers to different tables instead of data. Thus, the tables are related by references, which can be viewed as a network structure. A particular subset of the network model, the hierarchical model, limits the relationships to a tree structure, instead of the more general directed graph structure implied by the full network model.

2.1.3 Relational Model

The relational data model was introduced in an academic paper by E.F. Cod in 1970 as a way to make database management systems more independent of any particular application. It is a mathematical model defined in terms of predicate logic and set theory.

A relational database contains multiple tables, each similar to the one in the "flat" database model. However, unlike network databases, the tables are not linked by pointers. Instead, keys are used to match up rows of data in different tables. A key is just one or more columns in one table that correspond to columns in other tables. Any column can be a key, or multiple columns can be grouped together into a single key. Unlike pointers, it's not necessary to define all the keys in advance; a column can be used as a key even if it wasn't originally intended to be one.

Although the basic idea of a relational database has been very popular, relatively few people understand the mathematical definition and only a few obscure DBMSs implement it completely and without extension. Oracle, for example, can be used in a purely relational way, but it also allow tables to be defined that allow duplicate rows an extension (or violation) of the relational model. In common English usage, a DBMS is

called relational if it supports relational operational operations, regardless of whether it enforces strict adherence to the relational model. The following is an informal, non-technical explanation of how "relational" database management systems commonly work.

A key that can be used to uniquely identify a row in a table is called a unique key. Typically one of the unique keys is the preferred way to refer to row; this is defined as the table's primary key.

When a key consists of data that has an external, real-world meaning (such as a person's name, a book's ISBN, or a car's serial number), it's called a "natural" key. If no nature key is suitable, an arbitrary key can be assigned (such as by given employees ID numbers). In practice, most databases have both generated and natural keys, because generated keys can be used internally to create links between rows that can't break, while natural keys can be used, less reliably, for searches and for integration with other databases.

2.2 DATA MODELING

In information system design, data modeling is the analysis and design of the information in the system, concentrating on the logical entities and the logical dependencies between these entities. Data modeling is an abstraction activity in that the details of the values of individual data observations are ignored in favor of the structure, relationships, names and formats of the data of interest, although a list of valid values is frequently recorded. It is by the data model that definitions of what the data means is related to the data structures.

While a common term for this activity is "Data Analysis" the activity actually has more in common with the ideas and methods of synthesis (putting things together), than it does in the original meaning of the term analysis (taking things apart). This is because the activity strives to bring the data structures of interest together in a cohesive, inseparable, whole by eliminating unnecessary data redundancies and relating data structures by relationships.

2.2.1 Normalization

Database normalization is a series of steps followed to obtain a database design that allows for consistent storage and efficient access of data in a relational database. These steps reduce data redundancy and the risk of data becoming inconsistent.

However, many relational DBMS lack sufficient separation between the logical database design and the physical implementation of the data store, such that queries against a fully normalized database often perform poorly. In this case de-normalizations are sometimes used to improve performance, at the cost of reduced consistency.

2.2.2 Primary Key

In the relational model of data, a primary key is a candidate key chosen as the main method of uniquely identifying a relation. Practical telephone books, dictionaries and libraries can not use names, words or Dewey Decimal System Numbers as candidate keys because they do not uniquely identify telephone numbers, word definitions or books. In some design situations it is impossible to find a natural key that uniquely identifies a relation. A surrogate key can be used as the primary key. In other situations there may be more than one candidate key for a relation, and no candidate key is obviously preferred. A surrogate key may be used as the primary key to avoid giving one candidate key artificial primacy over the others. In addition to the requirement that the primary key be a candidate key, there are several other factors which may make a particular choice of key better than others for a given relation.

In database design, a primary key is a value that can be used to identify a particular row in a table. Attributes are associated with it. Examples are names in a telephone book (to look up telephone numbers), words in a dictionary (to look up definitions) and Dewey Decimal Numbers

The primary key should generally be short to minimize the amount of data that needs to be stored by other relations that reference it. A compound key is usually not

appropriate. (However, this is a design consideration, and some database management systems may be better than others in this regard.)

The primary key should be immutable, meaning its value should not be changed during the course of normal operations of the database. (Recall that a primary key is the means of uniquely identifying a tuple, and that identity by definition, never changes.) This avoids the problem of dangling references or orphan records created by other relations referring to a tuple whose primary key has changed. If the primary key is immutable, this can never happen.

2.2.3 Foreign Key

A foreign key is a field in a database record under one primary key that points to a key field of another database record in another table where the foreign key of one table refers to the primary key of the other table. This way references can be made to link information together and it is an essential part of database normalization.

For example, a person sending an e-mail needs not to include the entire text of a book in the e-mail. Instead, they can include the ISBN of the book, and interested persons can then use the number to get information about the book, or even the book itself. The ISBN is the primary key of the book, and it is used as a foreign key in the e-mail.

Note that using a foreign key often assumes its existence as a primary key somewhere else. Improper foreign key/primary key relationships are the source of many database problems.

2.2.4 Composite Key

In database design, a composite key (also called a compound key) is a key that consists on 2 or more attributes.No restriction is applied to the attribute regarding their (initial) ownership within the data model. This means that any one, none or all, of the multiple attributes within the composite key can be foreign keys. Indeed, a foreign key may, itself, be a composite key.

Composite keys almost always originate from attributive or associative entities (tables) within the model, but this is not an absolute value.

2.3 MERITS OF DATABASE

The modern era is known as the golden age computer sciences and technology. In a simple phrase we can express that the modern age is built on the foundation of database.If we carefully watch our daily life we can examine that some how our daily life is being connected with database. There are several benefits of database developments.Now with the help of computerized database we can access data in a second.By the development of the database we can make data more secure.By the development of database we can reduce the cost.

2.4 DEMERITS OF ABSENCE OF DATABASE

In the past when there wasn't proper system of database, Much paper work was need to do and to handle great deal of written paper documentation was giant among the problems itself.A glance on the past will may help us to reveal the drawbacks in case of absence of database.In the huge networks to deal with equally bulky data, more workers are needed which affidavit cost much labor expanses.

The old criteria for saving data and making identification was much time consuming such as if we want to search the particular data of a person.Before the Development of Computer database it was a great problem to search for some thing. Efforts to avoid the headache of search often results in new establishments of data.Before the development of database it seemed very unsafe to keep the worthy information. In Some situation some big organization had to employee the special persons in order to secure the data.Before the implementation of database any firm had to face the plenty of difficulties in order to maintain their Management. To hold the check on the expenses of the firm, the manager faced difficulties.

2.5 DESIGN OF DATABASE

The design of a database has to do with the way data is stored and how that data is related. The design process is performed after you determine exactly what information needs to be stored and how it is to be retrieved.

A collection of programs that enables you to store, modify, and extract information from a database. There are many different types of DBMS ranging from small systems that run on personal computers to huge systems that run on mainframes. The following are examples of database applications:

Computerized library systems

Automated teller machines

Flight reservation systems

Computerized parts inventory systems

From a technical standpoint, DBMS can differ widely. The terms relational, network, flat, and hierarchical all refer to the way a DBMS organizes information internally. The internal organization can affect how quickly and flexibly you can extract information.Requests for information from a database are made in the form of a query.

Database design is a complex subject. A properly designed database is a model of a business, Country Database or some other in the real world. Like their physical model counterparts, data models enable you to get answers about the facts that make up the objects being modeled. It's the questions that need answers that determine which facts need to be stored in the data model.

In the relational model, data is organized in tables that have the following characteristics: every record has the same number of facts, every field contains the same type of facts (Data) in each record, and there is only one entry for each fact. No two records are exactly the same. The more carefully you design, the better the physical database meets users' needs. In the process of designing a complete system, you must consider user needs from a variety of viewpoints.

2.5.1 What is the advantage of relational database design

Maintaining a simple, so-called flat database consisting of a single table doesn't require much knowledge of database theory. On the other hand, most database worth maintaining are quite a bit more complicated than that. Real life databases often have hundreds of thousands or even millions of records, with data that are very intricately related. This is where using a full-fledged relational database program becomes essential. Consider, for example, the Library of Congress, which has over 16 million books in its collection. For reasons that will become apparent soon, a single table simply will not do for this database.

2.6 RELATIONSHIPS BETWEEN TABLES

When you create tables for an application, you should also consider the relationships between them. These relationships give a relational database much of its power. There are three types of relationships between tables: one-to-one, one-to-many and many-to-many relationships.

2.6.1 One-To-One Relationships

In a one-to-one relationship, each record in one table corresponds to a single record in a second table. This relationship is not very common, but it can offer several benefits. First, you can put the fields from both tables into a single, combined table. One reason for using two tables is that each field is a property of a separate entity, such as owner operators and their tracks. Each operator can operate just one truck at a time, but the fields for the operator and truck tables refer to different entities.

A one-to-one relationship can also reduce the time needed to open a large table by placing some of the table's columns in a second, separate table. This approach makes particular sense when a table has some fields that are used infrequently. Finally, a one-to-one relationship can support in a table requires security, placing them in a separate

table lets your application restrict to certain fields. Your application can link the restricted table back to the main table via a one-to-one relationship so that people with proper permissions can edit, delete, and add new records to these fields.

2.6.2 One-To-Many Relationships

A one-to-many relationship, in which a row from one table corresponds to one or more rows from a second table, is more common. This kind of relationship can form the basis for a Many-To-Many relationship as well.

2.7 MYSQL

This chapter provides a tutorial introduction to MySQL by showing how to use the mysql client program to create and use a simple database. mysql (sometimes referred to as the ``terminal monitor" or just ``monitor") is an interactive program that allows you to connect to a MySQL server, run queries, and view the results. mysql may also be used in batch mode: you place your queries in a file beforehand, then tell mysql to execute the contents of the file. Both ways of using mysql are covered here.

To see a list of options provided by mysql, invoke it with the --help option:

shell> mysql --help

This chapter assumes that mysql is installed on your machine and that a MySQL server is available to which you can connect. If this is not true, contact your MySQL administrator. (If you are the administrator, you will need to consult other sections of this manual.)

This chapter describes the entire process of setting up and using a database. If you are interested only in accessing an already-existing database, you may want to skip over the sections that describe how to create the database and the tables it contains.

Because this chapter is tutorial in nature, many details are necessarily left out. Consult the relevant sections of the manual for more information on the topics covered here.

2.8 WHAT IS MYSQL?

Mysql is an open source software relational database management system which uses a SQL (Structured Query Language) SQL is the standard language used for interacting with databases.

2.9 WHAT IS THE DIFFERENS OF MYSQL FROM OTHERS?

The two main open source (free) alternatives to these are PostgreSQL and MySQL. PostgreSQL is arguably the better of the two, but MySQL is better supported on Windows, and is a popular choice among Web hosts that provide support for PHP.

There are many relational databases available to use, so why choose MySQL? We are specifically interested in databases which PHP supports; these include Oracle, IBM's DB2 and Microsoft's SQL Server (all of which cost money).

Here are some of MySQL's advantages

- It is cross platform
- There is a wide community of technical support
- It's secure
- It's fast
- It's free to use, and commercial licenses are reasonable
- It's easy to use

- Supports large databases

- It is designed specifically for web base applications and hence works very well command with PHP

STARTING THE SERVER FOR THE FIRST TIME

Testing from a DOS command prompt is the best thing to do because the server prints messages, so if something is wrong with your configuration, you will see a more accurate error message which will make it easier to identify and fix any problems.

Make sure you're in the right directory *(C:\>cd \mysql\bin),*

To install mysqld as a standalone program, enter:

C:\mysql\bin> mysqld-max --standalone

To install mysql as a service (Windows 2000), enter:

C:\mysql\bin> mysqld-nt --install

Now you can start and stop mysqld as follows:

C:\>NET START MySQL C:\>NET STOP MySQL

C:\>NET START MySQL

To start the MySQL Monitor, enter:

The MySql service is starting.

The MySQL service was started successfully.

C:\>cd \mysql

C:\mysql>bin\mysql

Welcome to the MySQL Monitor. Commands end with ; or \g. Your MySQL connection id is 1 to server version 3.23.49-nt Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> (enter a command or enter 'QUIT' to quit)

mysql> QUIT Bye

C:\mysql>NET STOP MySQL The MySQL service is stopping.

The MySQL service was stopped successfully.

C:\mysql>

2.11 ENVIRONMENT OF THE WINDOWS MYSQL

Starting with MySQL 3.23.38, the Windows distribution includes both the normal and the MySQL- Max server binaries. Here is a list of the different MySQL servers you can use:

mysqld	Compiled with full debugging and automatic memory allocation checking, symbolic links, InnoDB and DBD tables.
mysql-opt	Optimized binary with no support for transactional tables.
mysqld-nt	Optimized binary for NT with support for named pipes. You can run this version on Win98, but in this case no named pipes are created and you must have TCP/IP installed.
mysqld-max	Optimized binary with support for symbolic links, InnoDB and DBD tables.
mysqld-max-nt	Like mysqld-max, but compiled with support for named pipes.

All of the above binaries are optimized for the Pentium Pro processor but should work on any Intel processor >=i386

In the following circumstance, you will need to use the MySQL configuration file:

- The install/data directories are different than the default 'c:\mysql' and 'c:\mysql\data'.

- If you want to use one of these servers:

  - mysqld.exe

  - mysqld-max.exe

  - mysqld-max-nt.exe

- If you need to tune the server settings.

There are two configuration files with the same function: 'my.cnf' and 'my.ini' file, however, only one of these can/should be used. Both files are plain text. The 'my.cnf' file should be created in the root directory of drive C and the 'my.ini' file in the WinDir directory e.g.: C:\WINDOWS or C:\WINNT. If your PC uses a boot loader where the C drive isn't the boot drive, then your only option is to use the 'my.ini' file. Also note that if you use the WinMySQLAdmin tool, only the

'my.ini' file is used. The '\mysql\bin' directory contains a help file with instructions for using this tool.

Using Notepad, create the configuration file and edit the base section and keys:

[mysqld]

basedir = the_install_path # e.g. 'c:\mysql'

datadir = the_data_path # e.g. 'c:\mysql\data' or 'd:\mydata\data'

If the data directory is other than the default 'c:\mysql\data', you must cut the whole

'\data\mysql' directory and paste it on the your option new directory, e.g. 'd:\mydata\mysql'.

If you want to use the InnoDB transaction tables, you need to manually create two new directories to hold the InnoDB data and log files, e.g. 'c:\ibdata' and 'c:\iblogs'. You will also need to create some extra lines to the configuration file.

If you don't want to use, add the skip-innodb option to the configuration file.

Now you are ready to test starting the server.

2.12 QUERIES

Make sure you are connected to the server, as discussed in the previous section. Doing so will not in itself select any database to work with, but that's okay. At this point, it's more important to find out a little about how to issue queries than to jump right in creating tables, loading data into them, and retrieving data from them. This section describes the basic principles of entering commands, using several queries you can try out to familiarize yourself with how mysql works.

Here's a simple command that asks the server to tell you its version number and the current date. Type it in as shown below following the mysql> prompt and hit the RETURN key:

mysql> SELECT VERSION( ), CURRENT_DATE;

version()	CURRENT_DATE
3.22.20a-log	1999-03-19

row in set (0.01 sec)

mysql>

This query illustrates several things about mysql:

A command normally consists of a SQL statement followed by a semicolon. (There are some exceptions where a semicolon is not needed. QUIT, mentioned earlier, is one of them. We'll get to others later.)

When you issue a command, mysql sends it to the server for execution and displays the results, then prints another mysql> to indicate that it is ready for another command.

Mysql displays query output as a table (rows and columns). The first row contains labels for the columns. The rows following are the query results. Normally, column labels are the names of the columns you fetch from database tables. If you're retrieving the value of an expression rather than a table column (as in the example just shown), mysql labels the column using the expression itself.

Mysql shows how many rows were returned and how long the query took to execute, which gives you a rough idea of server performance. These values are imprecise because they represent wall clock time (not CPU or machine time), and because they are affected by factors such as server load and network latency. (For brevity, the ``rows in set'' line is not shown in the remaining examples in this chapter.)

Keywords may be entered in any lettercase. The following queries are equivalent:

mysql> SELECT VERSION(), CURRENT_DATE; mysql> select version(), current_date; mysql> SELECT VERSION(), current_DATE;
*mysql> SELECT SIN(PI()/4), (4+1)*5;*

The commands shown thus far have been relatively short, single-line statements. You can even enter multiple statements on a single line. Just end each one with a semicolon:

mysql> SELECT VERSION(); SELECT NOW();

A command need not be given all on a single line, so lengthy commands that require several lines are not a problem. mysql determines where your statement ends by looking for the terminating semicolon, not by looking for the end of the input line. (In other words, mysql accepts free-format input: it collects input lines but does not execute them until it sees the semicolon.)

Here's a simple multiple-line statement:

mysql> SELECT USER(),CURRENT_DATE;

USER()	CURRENT_DATE
joesmith@localhost	1999-03-18

In this example, notice how the prompt changes from mysql> to -> after you enter the first line of a multiple-line query. This is how mysql indicates that it hasn't seen a complete statement and is waiting for the rest. The prompt is your friend, because it provides valuable feedback. If you use that feedback, you will always be aware of what mysql is waiting for.

If you decide you don't want to execute a command that you are in the process of entering, cancel it by typing \c:

mysql> SELECT USER() \c mysql>

Here, too, notice the prompt. It switches back to mysql> after you type \c, providing feedback to indicate that mysql is ready for a new command.

The following table shows each of the prompts you may see and summarizes what they mean about the state that mysql is in:

Prompt	**Meaning**
mysql>	Ready for new command.
->	Waiting for next line of multiple-line command.
'>	Waiting for next line, collecting a string that begins with a single quote (`').
">	Waiting for next line, collecting a string that begins with a double quote (`").

Multiple-line statements commonly occur by accident when you intend to issue a command on a single line, but forget the terminating semicolon. In this case, mysql waits for more input:

mysql> SELECT USER()
->

If this happens to you (you think you've entered a statement but the only response is a -> prompt), most likely mysql is waiting for the semicolon. If you don't notice what the prompt is telling you, you might sit there for a while before realizing what you need to do. Enter a semicolon to complete the statement, and mysql will execute it:

mysql> SELECT USER()

-> ;

USER()

joesmith@localhost

The '> and "> prompts occur during string collection. In MySQL, you can write strings surrounded by either `'` or `"` characters (for example, 'hello' or "goodbye"), and mysql lets you enter strings that span multiple lines. When you see a '> or "> prompt, it means that you've entered a line containing a string that begins with a `'` or `"` quote character, but have not yet entered the matching quote that terminates the string. That's fine if you really are entering a multiple-line string, but how likely is that? Not very. More often, the '> and "> prompts indicate that you've inadvertantly left out a quote character. For example:

*mysql> SELECT * FROM my_table WHERE name = "Smith AND age < 30;*

">

If you enter this SELECT statement, then hit RETURN and wait for the result, nothing will happen. Instead of wondering why this query takes so long, notice the clue provided by the "> prompt. It tells you that mysql expects to see the rest of an unterminated string. (Do you see the error in the statement? The string "Smith is missing the second quote.)

At this point, what do you do? The simplest thing is to cancel the command. However, you cannot just type \c in this case, because mysql interprets it as part of the string that it

is collecting! Instead, enter the closing quote character (so mysql knows you've finished the string), then type

*mysql> SELECT * FROM my_table WHERE name = "Smith AND age < 30;*

"> "\c mysql>

The prompt changes back to mysql>, indicating that mysql is ready for a new command.

It's important to know what the '> and "> prompts signify, because if you mistakenly enter an unterminated string, any further lines you type will appear to be ignored by mysql -- including a line containing QUIT! This can be quite confusing, especially if you don't know that you need to supply the terminating quote before you can cancel the current command.

2.13 CONNECTING TO AND DISCONNECTING FROM THE SERVER

To connect to the server, you'll usually need to provide a MySQL user name when you invoke mysql and, most likely, a password. If the server runs on a machine other than the one where you log in, you'll also need to specify a hostname. Contact your administrator to find out what connection parameters you should use to connect (that is, what host, user name, and password to use). Once you know the proper parameters, you should be able to connect like this:

shell> mysql -h host -u user -p

Enter password: ********

The ******** represents your password; enter it when mysql displays the Enter password:

prompt.

If that works, you should see some introductory information followed by a mysql> prompt:

shell> mysql -h host -u user -p

*Enter password: ********

Welcome to the MySQL monitor. Commands end with ; or \g. Your MySQL connection id is 459 to server version: 3.22.20a-log

Type 'help' for help.

mysql>

The prompt tells you that mysql is ready for you to enter commands.

Some MySQL installations allow users to connect as the anonymous (unnamed) user to the server running on the local host. If this is the case on your machine, you should be able to connect to that server by invoking mysql without any options:

shell> mysql

After you have connected successfully, you can disconnect any time by typing QUIT at the *mysql>*

prompt: mysql> QUIT Bye

You can also disconnect by pressing Control-D.

Most examples in the following sections assume you are connected to the server. They indicate this by the mysql> prompt.

CHAPTER 3

USER MANUEL

I will try to describe the hospital application program in this chapter.When the program started the main interface would came.This interface is very easy to use.

Figure 3.1

As seen on Figure 3.1 there are 6 buttons on interface, also there is a big picture on it.This picture signify tree of life.And all colors of hospital about with color of the picture.Under the user screen there is date and time displayer.

When we push the firts button which *New Patient* button (Figure 3.2); a new interface (Figure 3.3) will come and the main page will hide.

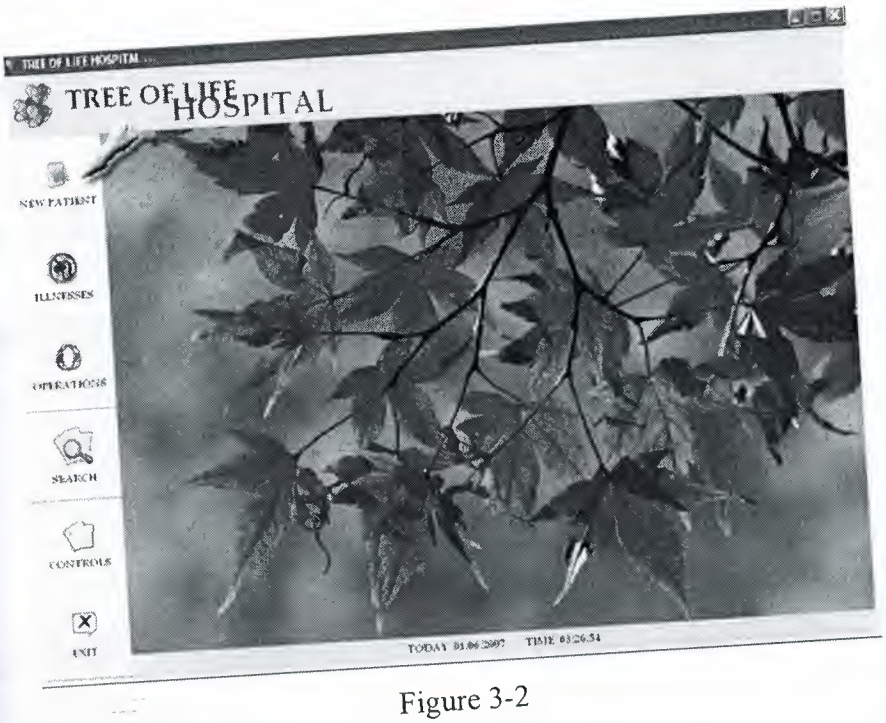

Figure 3-2

The new user screen will be this:

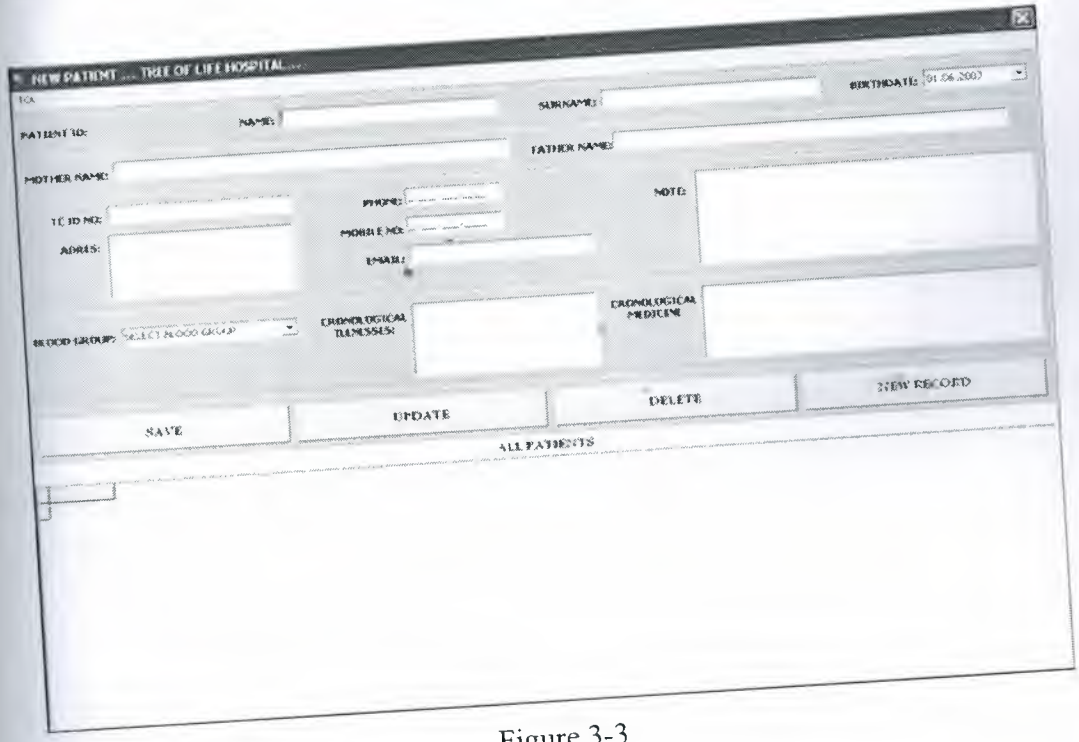

Figure 3-3

On this page there are 6 editbox, 4 memo, 2 maskedit, 1 combobox and 4 button. When I was started to design the program, I thought as important name, surname, birthdate, mother name, father name, tcid no and address.Thats why when we push save or update buton (Figure 3.3) without complate these boxes and attention message will show.

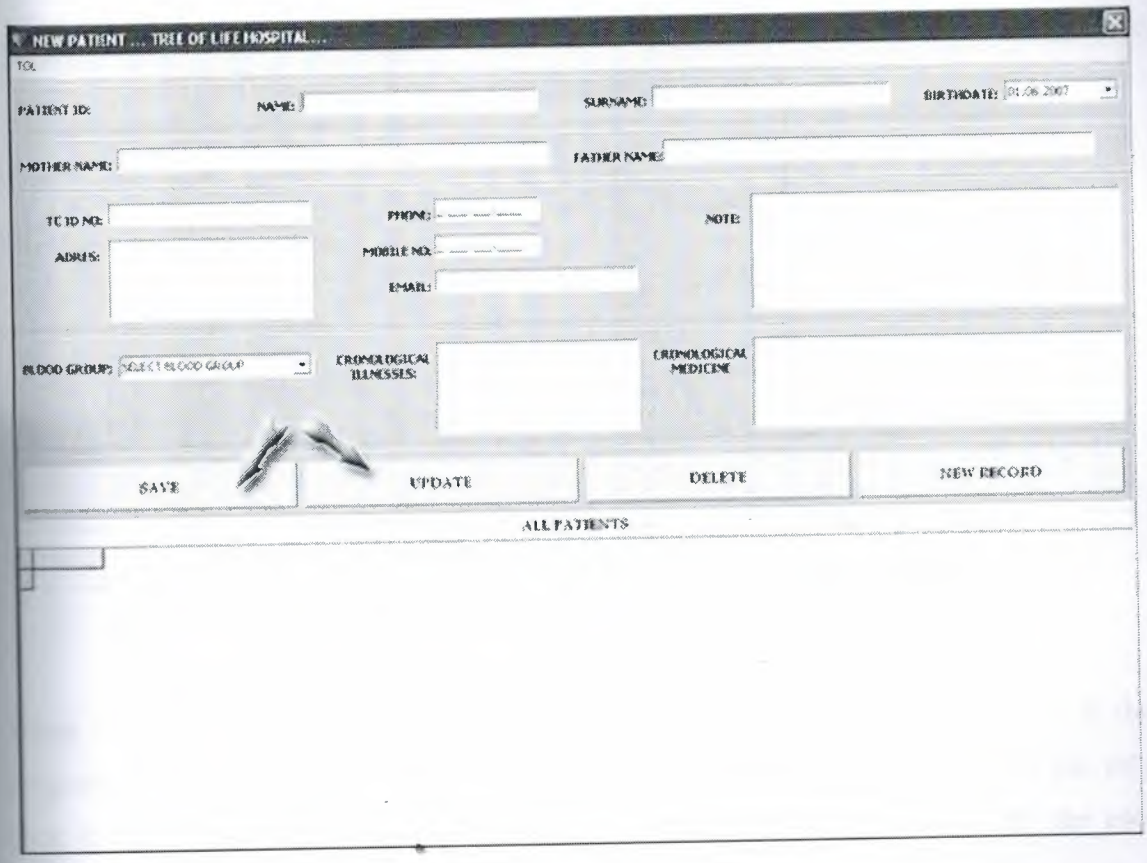

Figure 3.4

If user push the save or update button (Above Figure 3.4) by mistake the program will show am message (Below Figure 3.5) In this message shows empty places or mistake places. When the user coreect them the program accept the information.

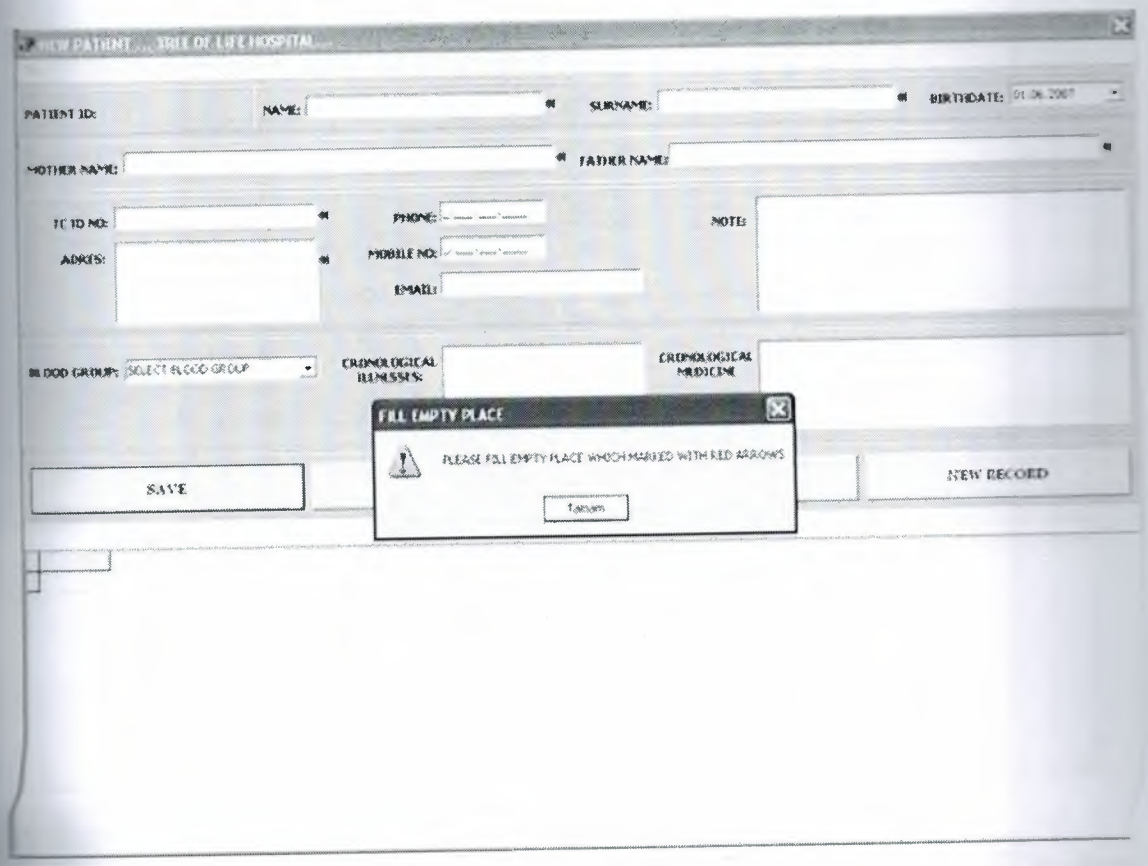

Figure 3.5

When the delete button pushed, the program cannot delete directly, instead of it the program shows a messagebox, and ask "Are you sure?" (Figure 3.6 Below). If the user click the yes then the program will generate to delete selected information. If the user didnot select information to delete the program will show a message like "Please select information to delete". These controls are importat for regularity of the program. If we didnot make like this controls then who dont know about what will happen.

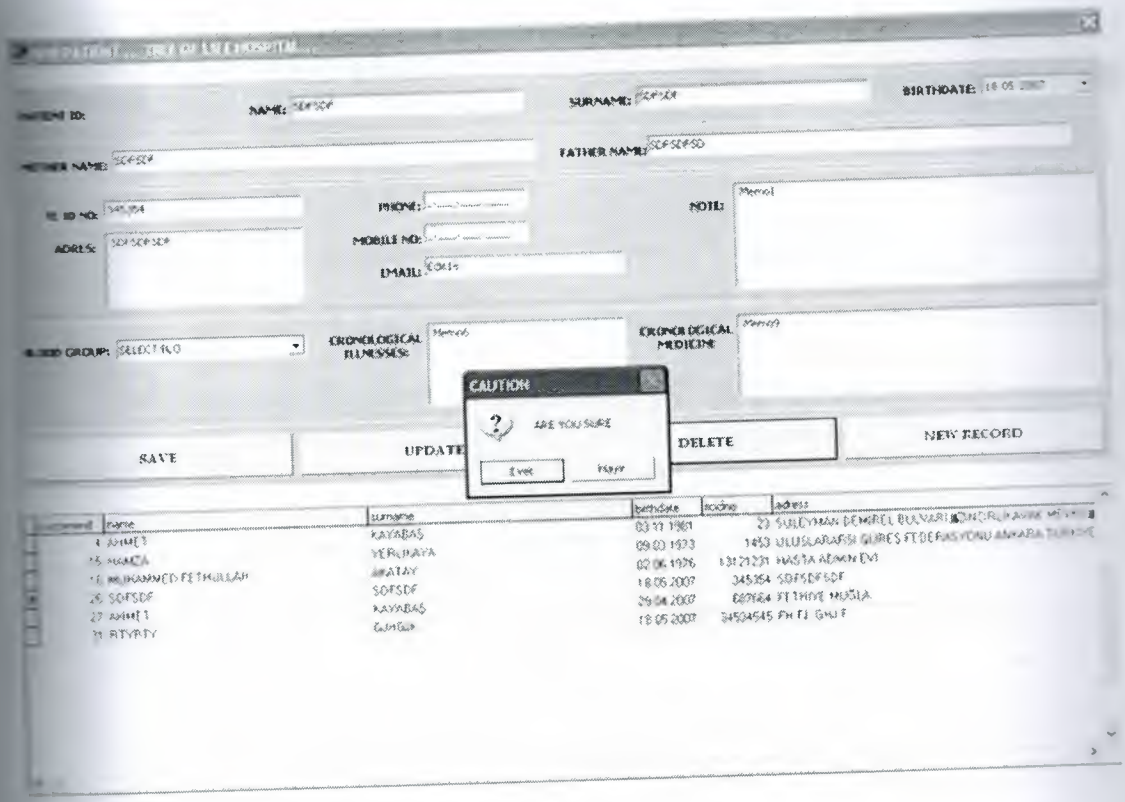

Figure 3.6

If we click illnesses button on main page (Figure 3.7) then illnesses form will be shown (Figure 3.8).

Figure 3.7

Figure 3.8

In illnesses form user can save or update or delete illnesses informations of patient. When the illnesses interface then empty boxes and empty datagrid come. This form designed to be practical. As seen anybody can use it easily. Well the program will help them to use own. For example the user click save or update button without filling the boxes, the program will show message to fill empty place, or to select patient (Figure 3.9)

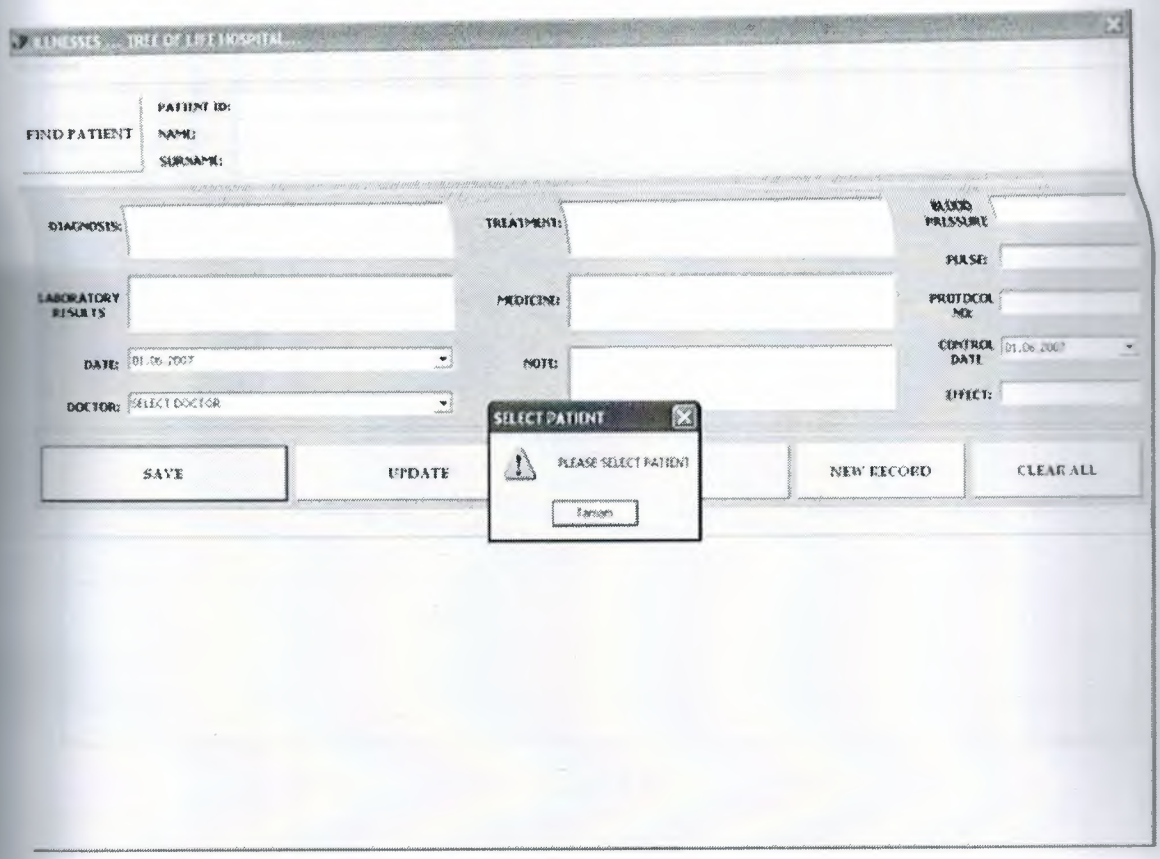

Figure 3.9

Cause of this to sort of all illnesses information by patient. Thus only related patient information will come on screen. Thats why user can control related illnesses information easly.

When the user see it, who must click Find Patient button (Figure 3.10).If user click it a new form will appear in which all patient are listed (Figure 3.11)

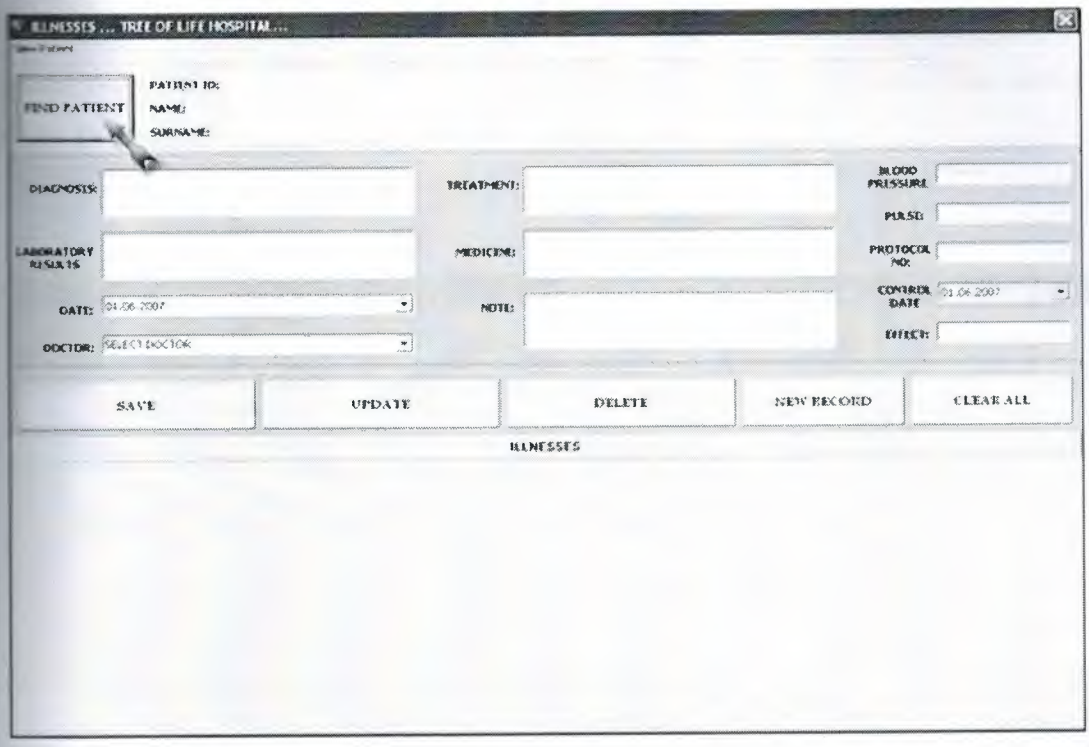

Figure 3.10

Figure 3.11

On illnesses form you can see New Record button (Figure 3.12).This button clear only illnesses boxes, not patient boxes. This acts quick registration processes. Thus user can transfer all illnesses informations from papers to program as quickly.

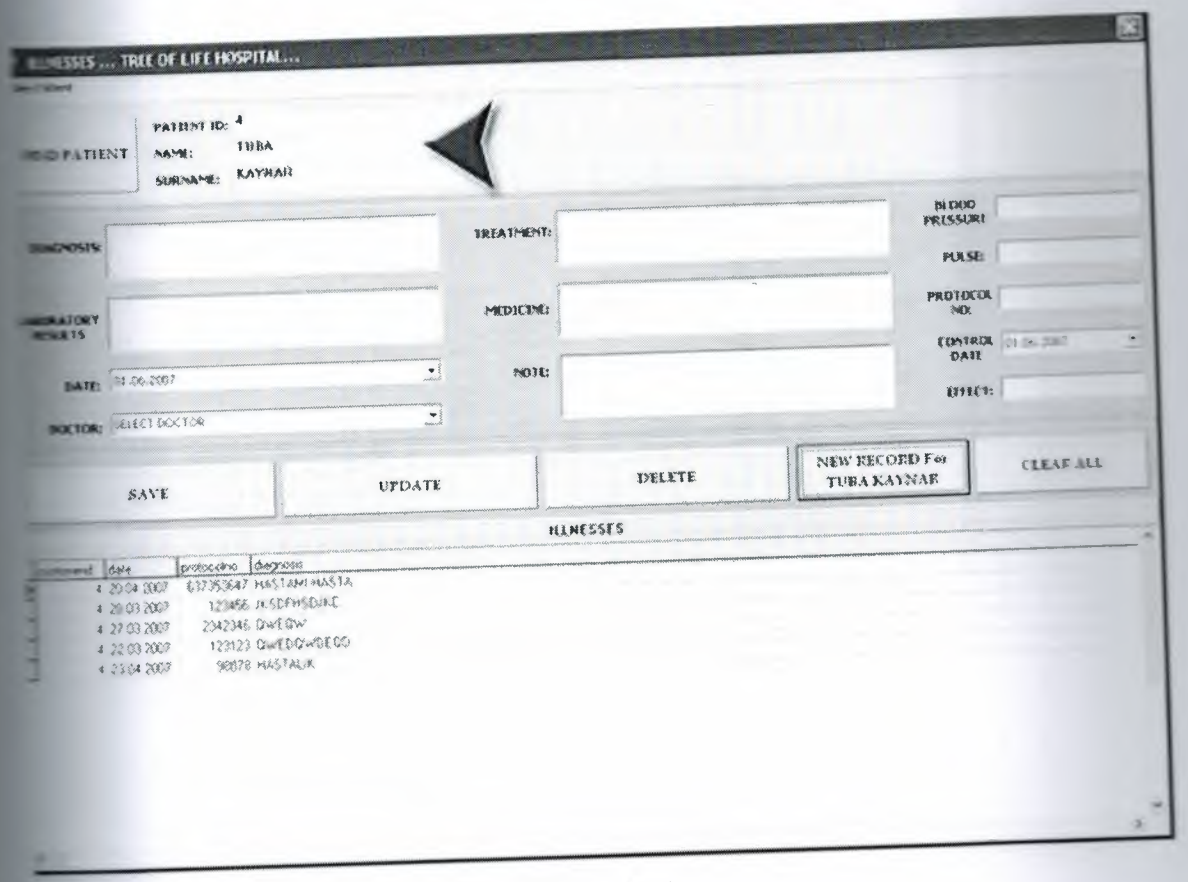

Figure 3.12

This property different from others. Because as known transfer processes is very boring processes.At the same time it gets long time. Thats why I designed this to spend time. I hope user will happy for this property.And also this property is used on operation form as same idea.

If we click operations button on main page then operations form will be shown (Figure 3.12).

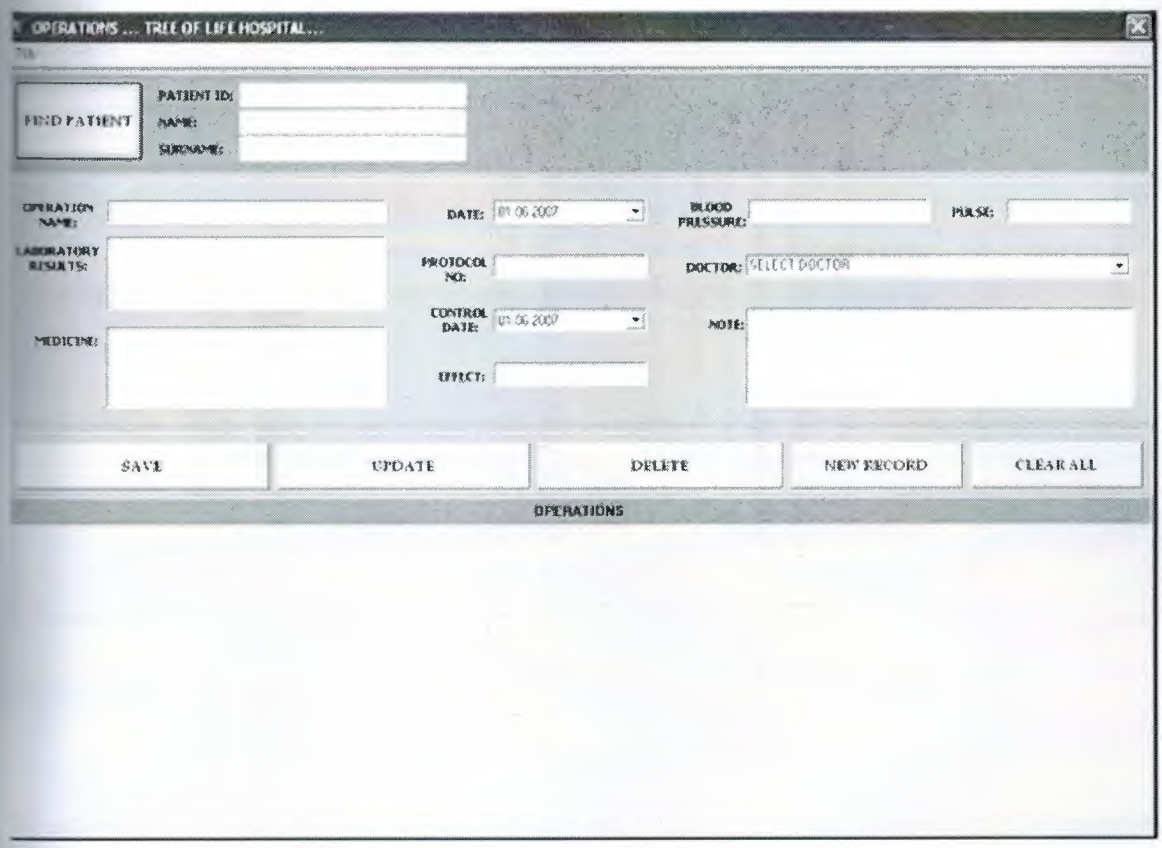

Figure 3.13

On this page there are 7 editbox, 3 memo, 3 combobox and 6 button. When I was started to design the program, I thought as important operation name, date, control date, doctor, blood pressure, pulse and protocol no.Thats why when we push save or update buton (Figure 3.14) without complate these boxes and attention message will show.

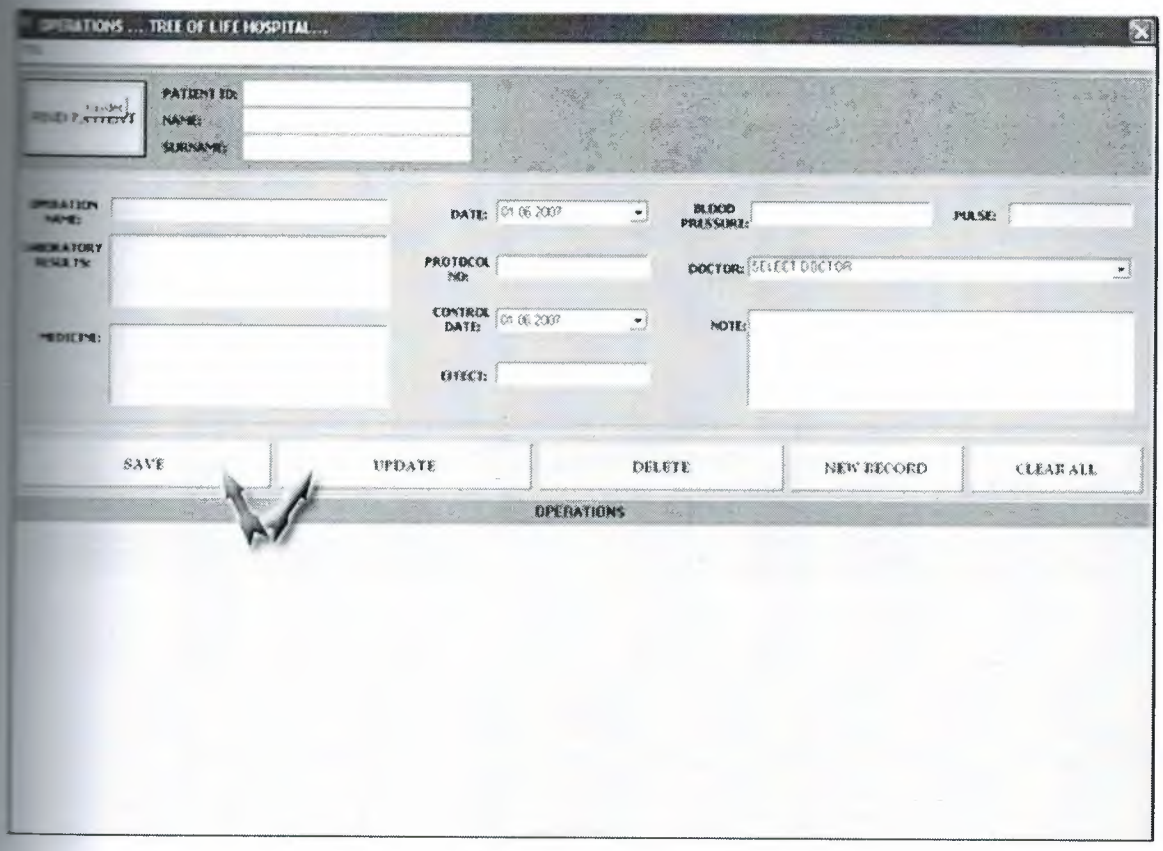

Figure 3.14

In operations form user can save or update or delete operations informations of patient. When the operations interface then empty boxes and empty datagrid come. This form designed to be practical. As seen anybody can use it easily. Well the program will help them to use own. For example the user click save or update button without filling the boxes, the program will show message to fill empty place, or to select patient (Figure 3.15)

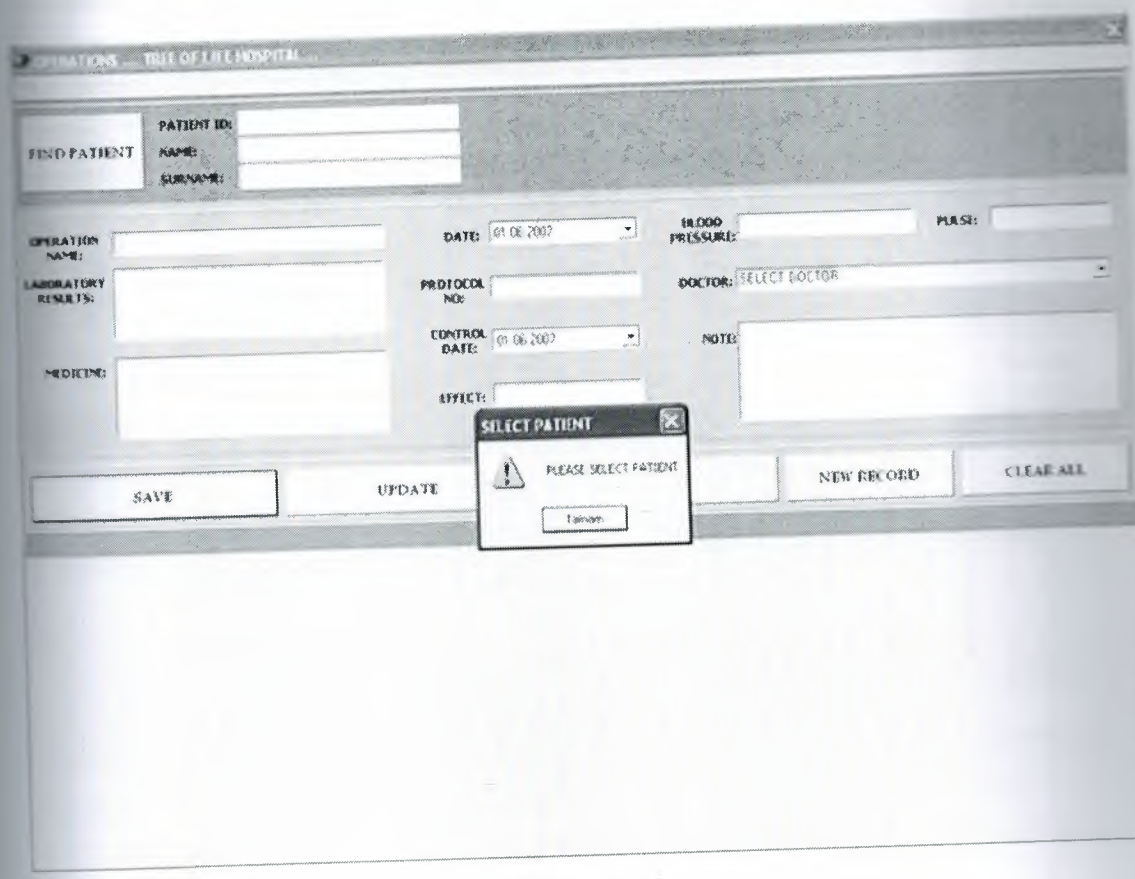

Figure 3.15

When the delete button pushed, the program cannot delete directly, instead of it the program shows a messagebox, and ask "Are you sure?". If the user click the yes then the program will generate to delete selected information. If the user didnot select information to delete the program will show a message like "Please select information to delete". These controls are importat for regularity of the program. If we didnot make like this controls then who dont know about what will happen.

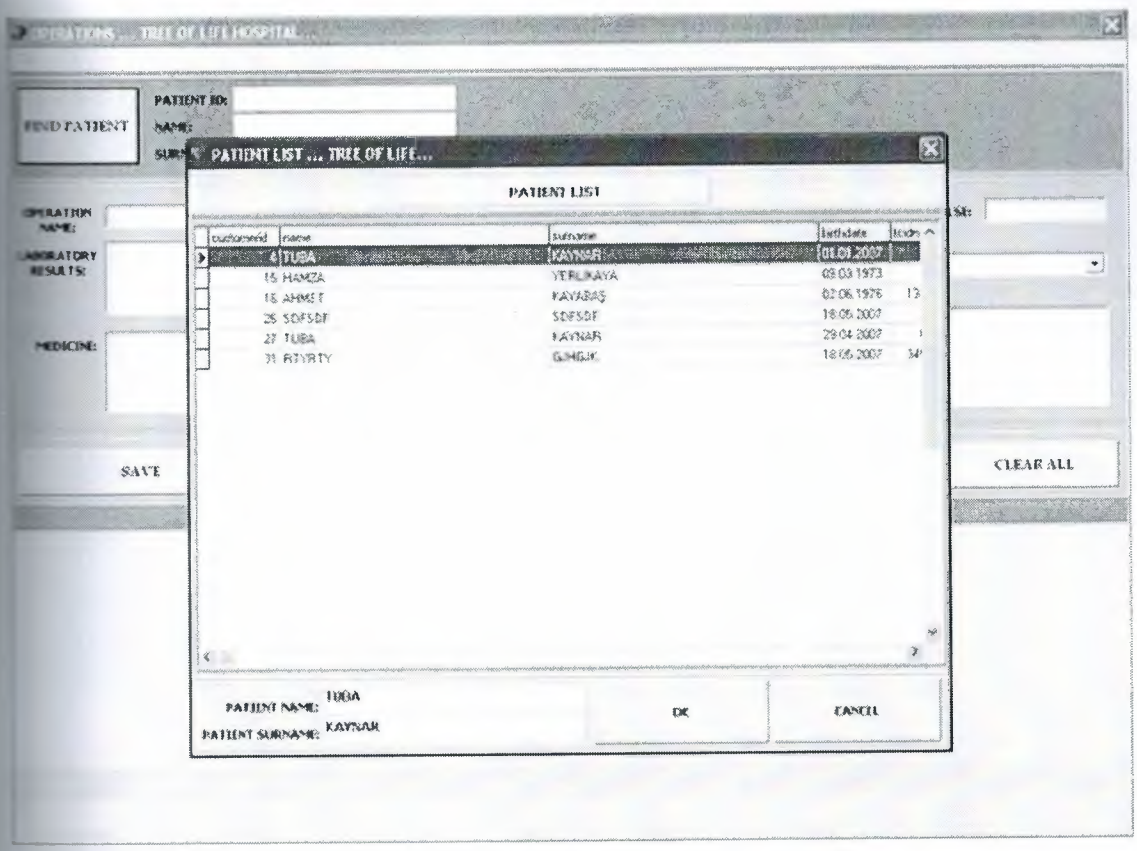

Figure 3.16

When the user see patient list, who must click Find Patient button If user click it a new form will appear in which all patient are listed.

As seen above (Figure 3.16) there is a patient list form.Cause of this to sort of all illnesses information by patient. Thus only related patient information will come on screen. Thats why user can control related operation information easly.

If we click search button on main page (Figure 3.17). then search form will be shown (Figure 3.18).

Figure 3.17

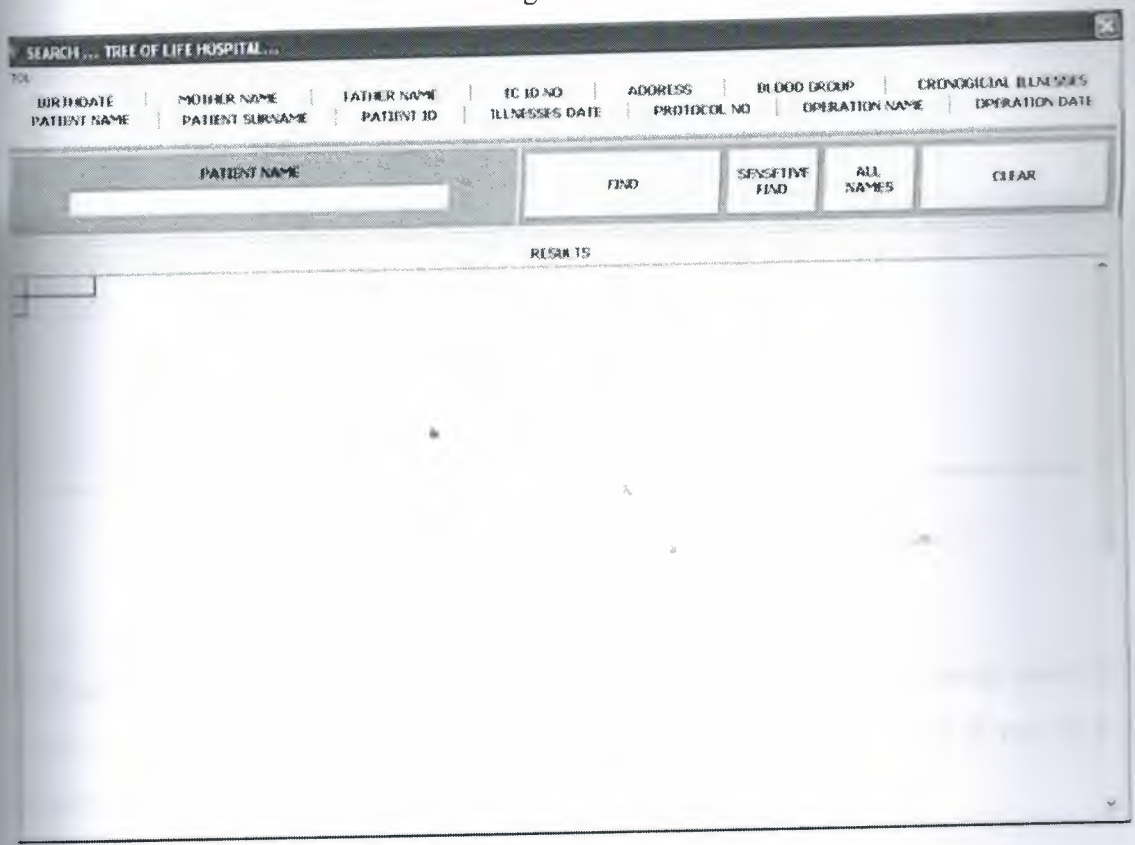

Figure 3.18

On search interface there are 14 search criteria. These criterias are: Patient name, Patient surname, Mother name, Father name, Birthdate, Patient id, Tcid no, Address, Blood group, Cronological illnesses, Protocol no, Illnesses date, Operation date and Operation name. I designed search system as powerfull. Because as known unnecessary informations are not shown. Well why registrate it. Thats why I designed to consider informations.

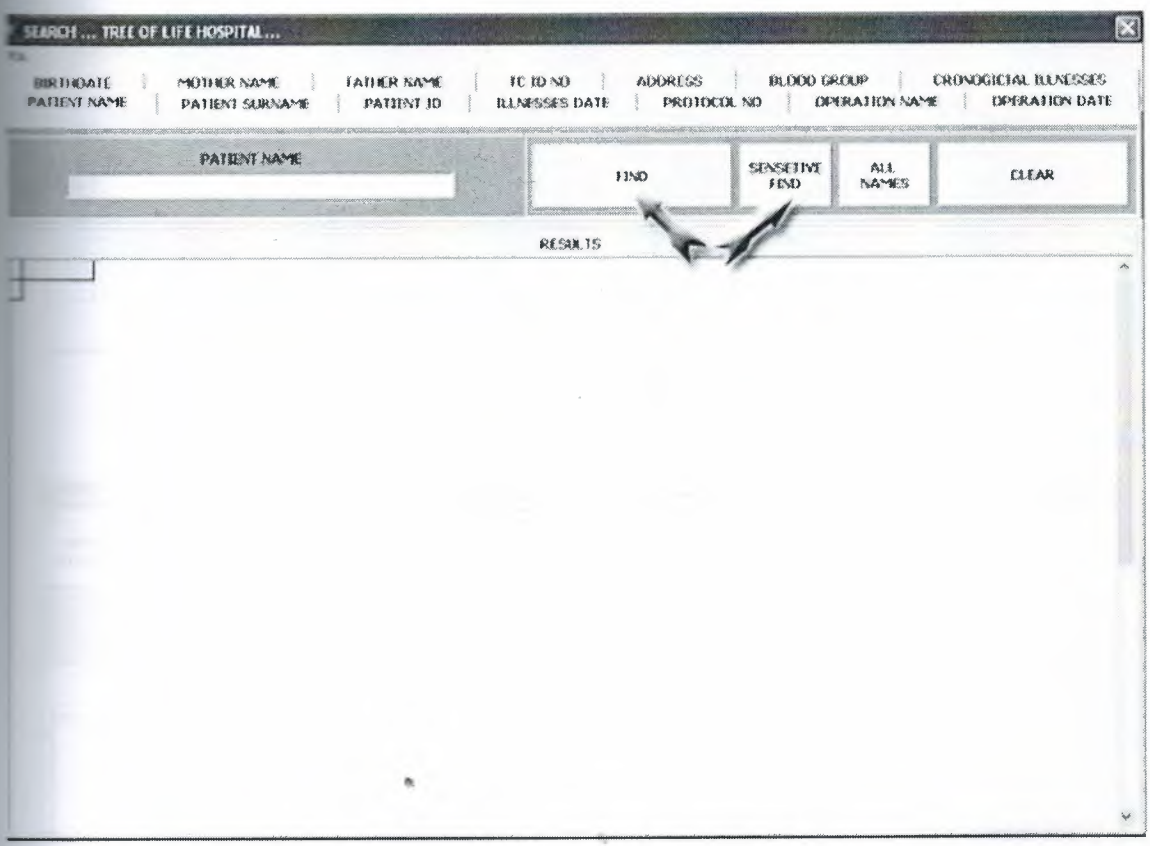

Figure 3.19

Find button search equally with written in box, but sensetive find button search similary. If you do not enter any word the program will show a message (Figure 3.20). If you click all names button then the program will show all patient names (Figure 3.21).

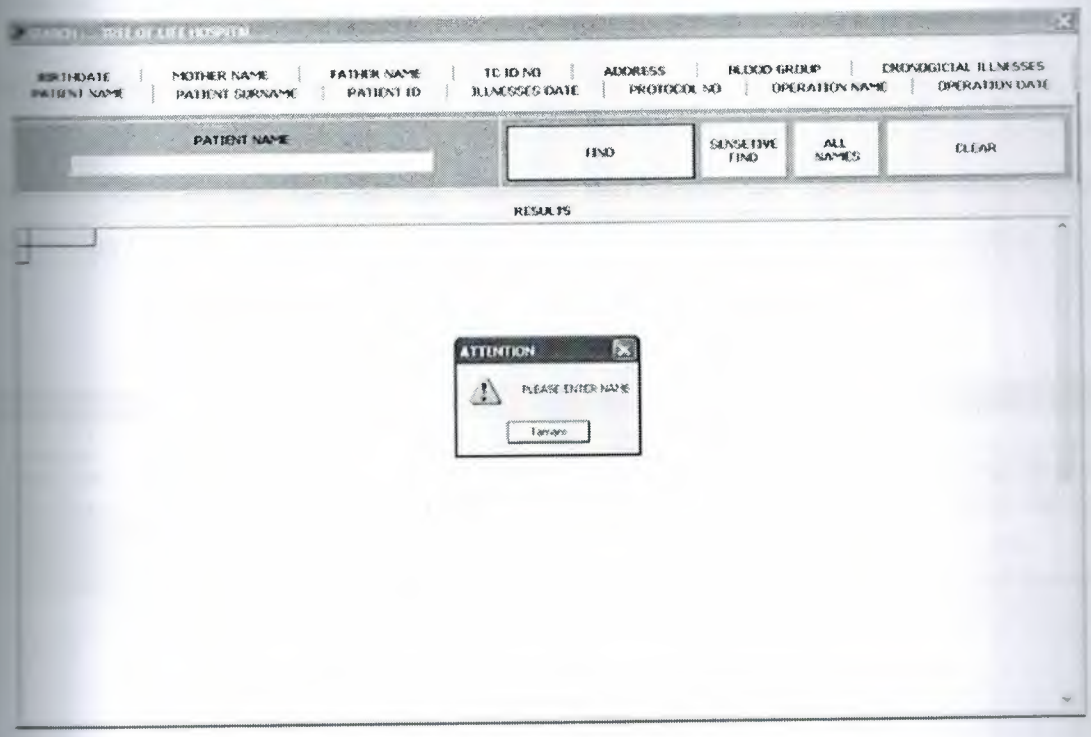

Figure 3.20

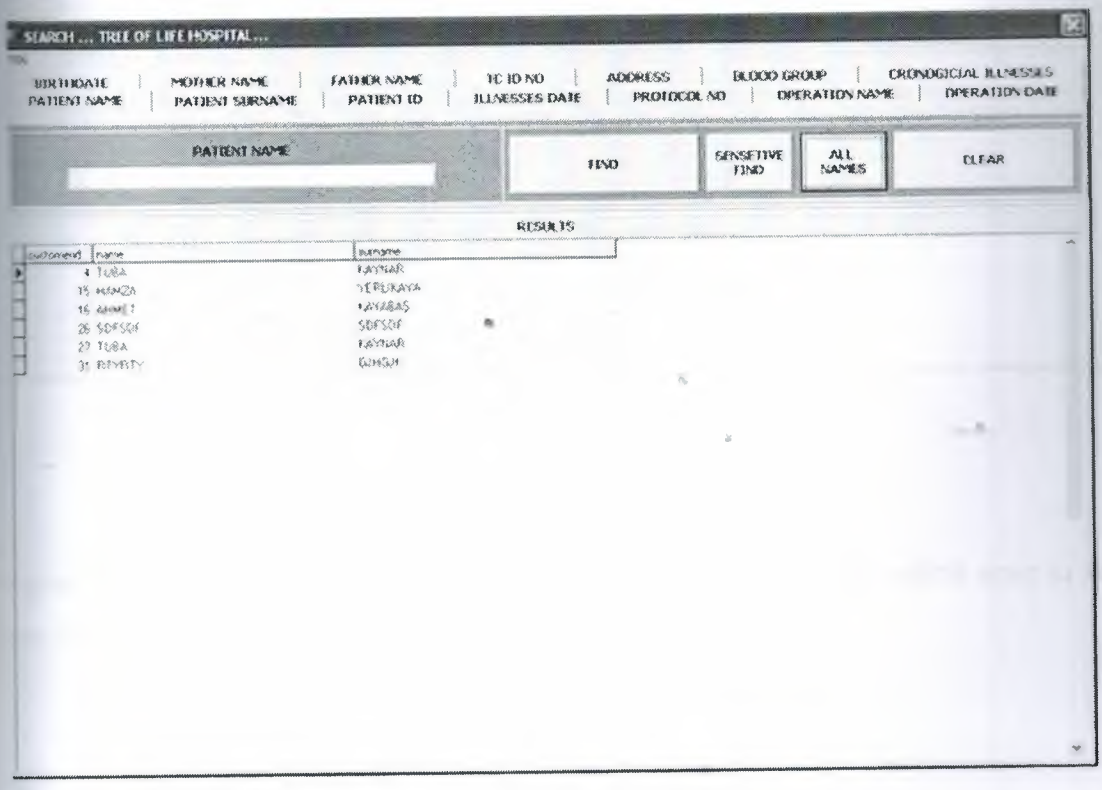

Figure 3.21

Birthdate or illnesses date or operation date search has 7 date criteria. These are "<", ">", "><", ">=", "<>", "=", "Between" (Figure 3.22). User must select one of them. After that who must click button. Then search processes will manipulate. This wide search criteria acts easility to users. Because they without lossing time will make searching process.

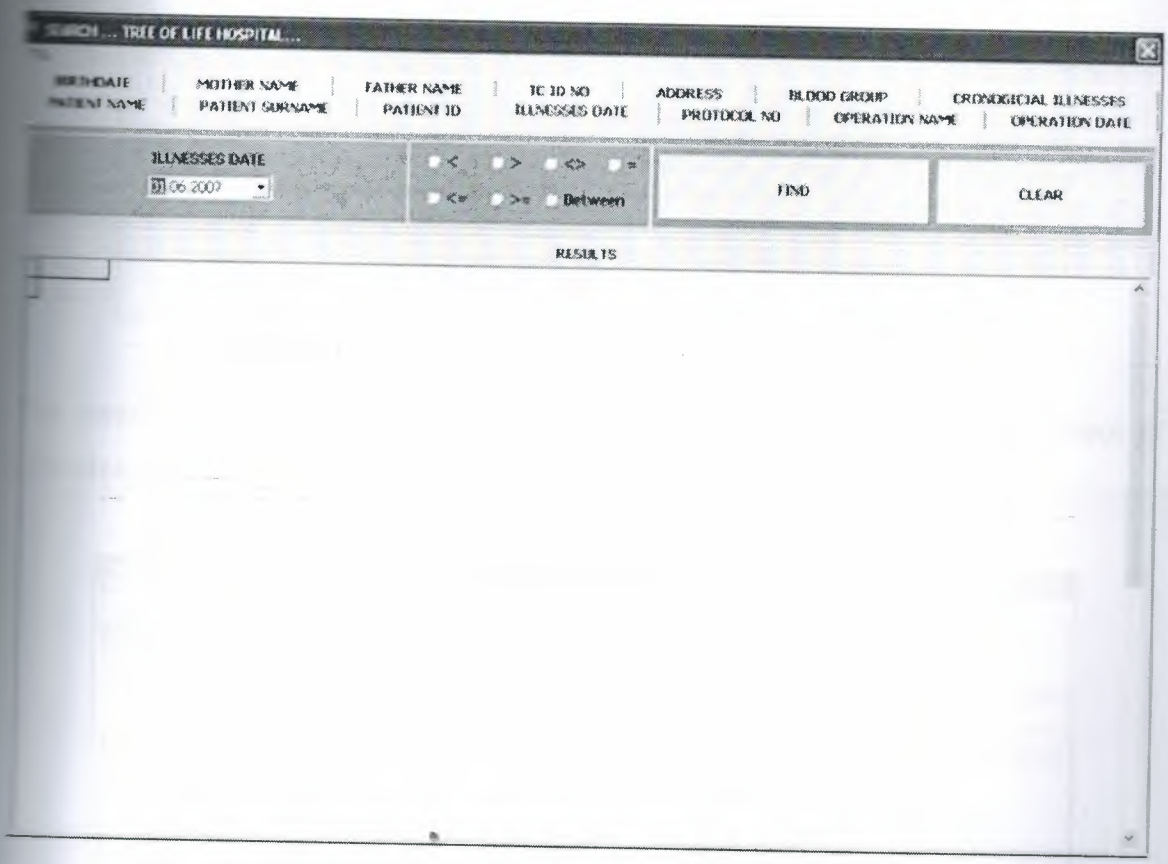

Figure 3.22

Program will search date and will sort them. Well user can find date which want to see. Thus the program will be quickly and safetly searching process.

If we click control date button on main page then control date form will be shown

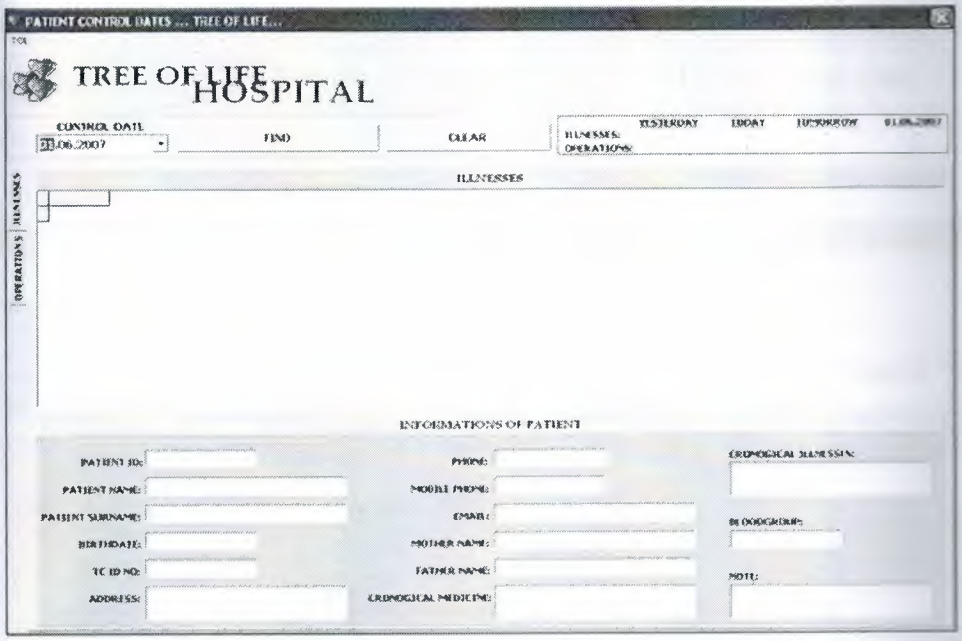

Figure 3.23

The control date interface checks available controls of illnesses patients and control of operation patients.

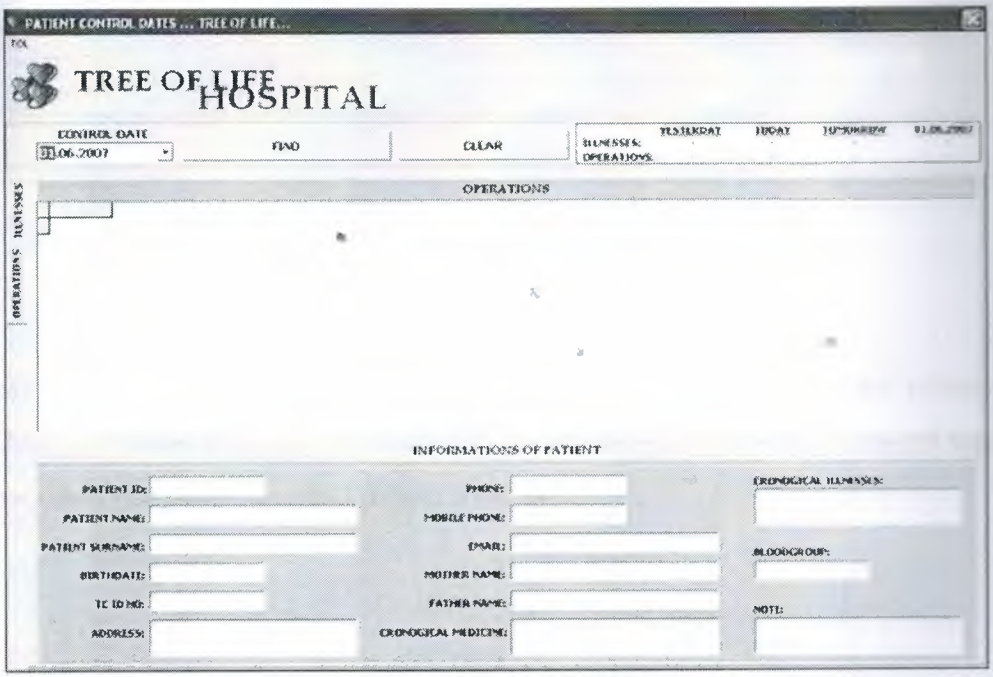

Figure 3.24

In here user will select control date and click find button. Thus all controls will list as by selected date. If there is no any control program will change of interface and will show a message. (Figure 3.25).Then user can understand there is no control for this date.

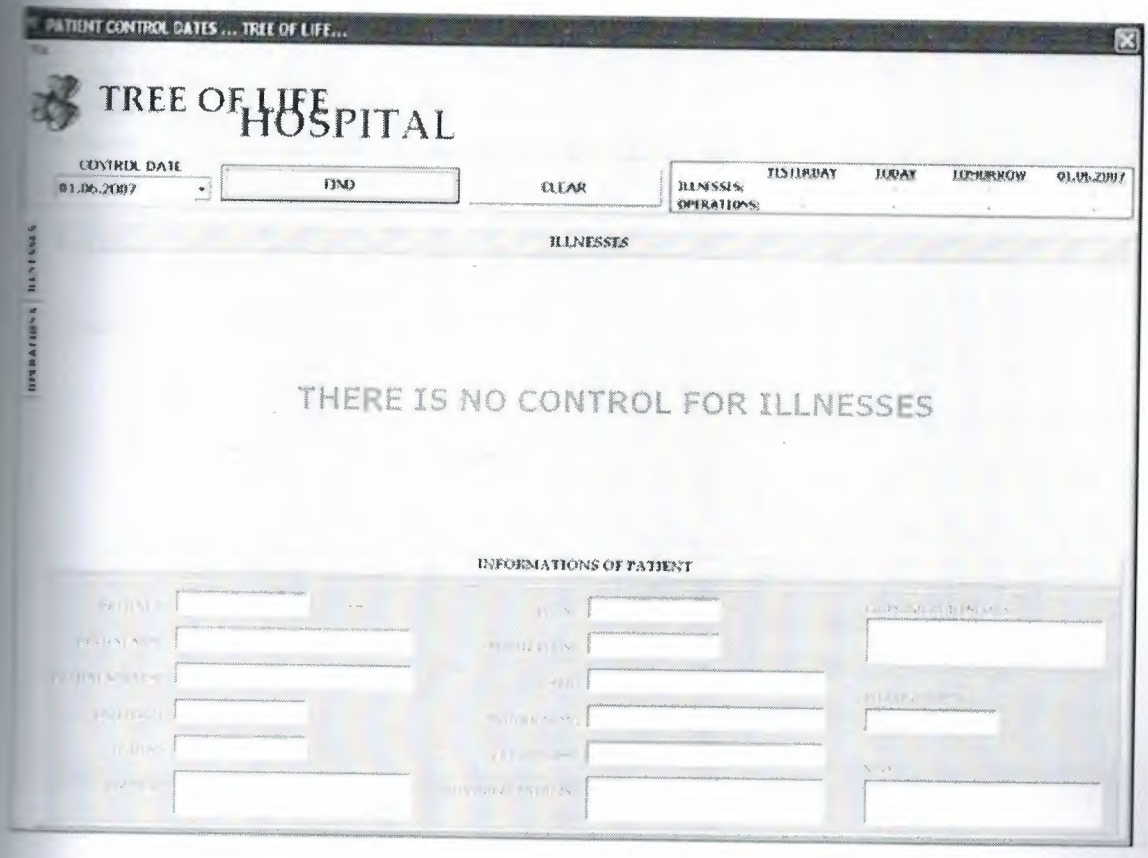

Figure 3.25

As seen there is an analysis part on interface. This show how many did patient come yesterday, how many did patient come today, how many patient will be control tomorrow and how many did patient controll on signified date.

If any patient are listed on control date interface, you can double click on it. Thus you can access all information about this patient (Figure 3.26). This way is very fast access for all information. Thats why doctor see passed illnesses or operations and will give correct and safetly treatment to patients.

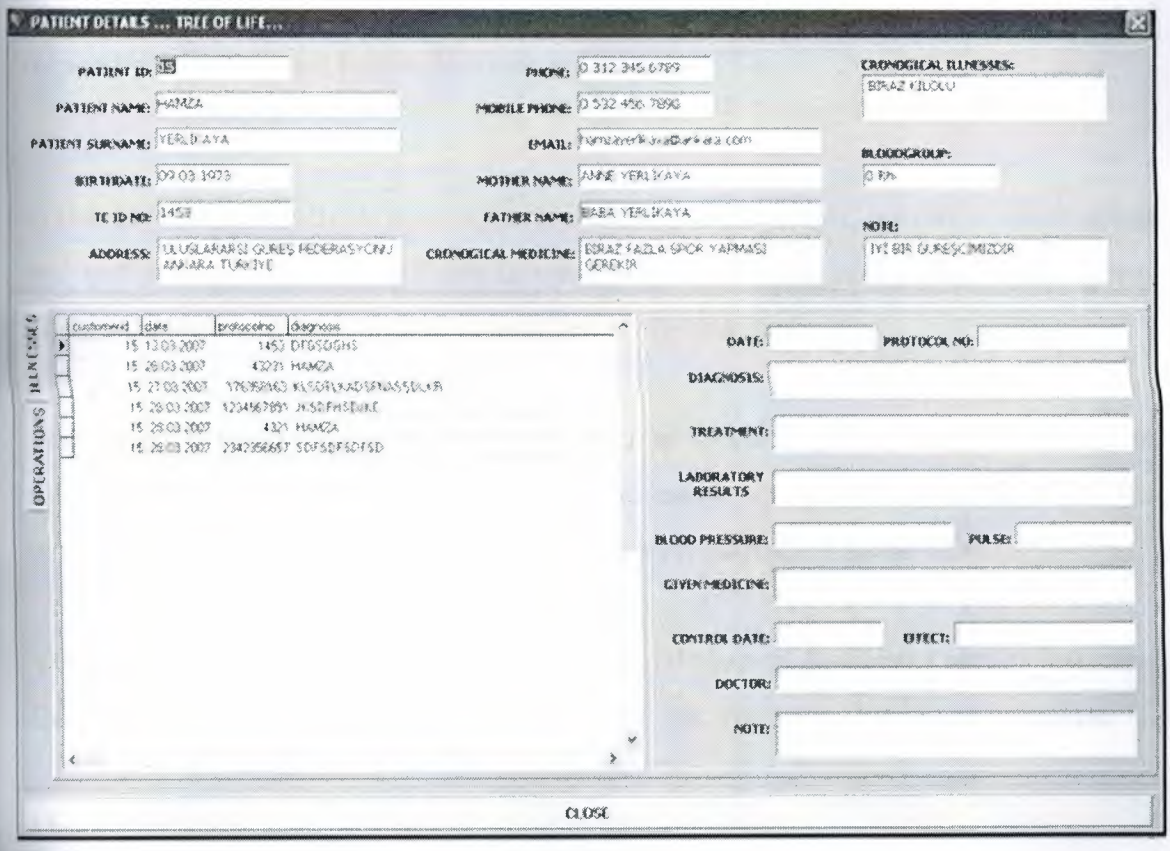

Figure 3.26

When the user click close button, this interface will hide, thus you can back real screen. This property is very important for doctors and patients. Because patients background is very important in health services.

CONCLUSION

Examination of the data for internal consistency and comparisons with externally available data indicates that the delphi study appears reliable.Mysql and delphi are powerful program. Well the study was difficult to carry out owing to difficulties in obtaining answers from possible respondents. Thus, if a larger survey is to be undertaken to include all building components, it is recommended that committed respondents be obtained before devising the survey.

In this project keep patients informations, illnesses informations, operation informations and sort them by patient id. These informations are investigated rapidly by program. Thats why the doctor obtain time to intersted in with patients.

The program provide a wide search criteria to users. Which property acts for all informations to be usable without null informations.Now doctors will save null time that would be documentation and they interested in with patients as much more...

Yes perhaps it is not big project but it is beneficial project. And also I obtained success. Because I did not give up and I did not interrupt that. Here now I am in big pleasure.

I hope that this project gets benefit which is although small...

APPENDIX

1.UNIT1 CODES

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, ExtCtrls, StdCtrls, Buttons, jpeg;

type
  TForm1 = class(TForm)
    Panel1: TPanel;
    Image1: TImage;
    Image2: TImage;
    Label2: TLabel;
    Image3: TImage;
    Label1: TLabel;
    SpeedButton1: TSpeedButton;
    SpeedButton2: TSpeedButton;
    SpeedButton3: TSpeedButton;
    SpeedButton4: TSpeedButton;
    Image4: TImage;
    Panel2: TPanel;
    Timer1: TTimer;
    SpeedButton5: TSpeedButton;
    SpeedButton6: TSpeedButton;
    procedure Timer1Timer(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure SpeedButton3Click(Sender: TObject);
    procedure SpeedButton5Click(Sender: TObject);
    procedure SpeedButton1Click(Sender: TObject);
    procedure SpeedButton6Click(Sender: TObject);
    procedure SpeedButton2Click(Sender: TObject);
    procedure SpeedButton4Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

uses Unit2, Unit3, Unit4, Unit5, Unit6;

{$R *.dfm}

procedure TForm1.Timer1Timer(Sender: TObject);
begin
  PANEL2.Caption:='TODAY  '+DATETOSTR(DATE)+'        TIME
'+TIMETOSTR(TIME);
end;
```

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  PANEL2.Caption:='TODAY  '+DATETOSTR(DATE)+'        TIME
'+TIMETOSTR(TIME);
end;

procedure TForm1.SpeedButton3Click(Sender: TObject);
begin
  FORM2.SHOW;
  FORM1.Hide;
end;

procedure TForm1.SpeedButton5Click(Sender: TObject);
begin
  FORM3.SHOW;
  FORM1.Hide;
end;

procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
  FORM4.SHOW;
  FORM1.Hide;
end;

procedure TForm1.SpeedButton6Click(Sender: TObject);
begin
  CLOSE;
end;

procedure TForm1.SpeedButton2Click(Sender: TObject);
begin
  FORM5.SHOW;
  FORM1.Hide;
end;

procedure TForm1.SpeedButton4Click(Sender: TObject);
begin
  FORM6.SHOW;
  FORM1.Hide;
end;

end.
```

2. UNIT2 CODES

```
unit Unit2;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, Grids, DBGrids, ComCtrls, StdCtrls, Buttons, ExtCtrls,
Mask,
  DBCtrls, DB, ADODB, Menus;

type
  TForm2 = class(TForm)
```

```
GroupBox1: TGroupBox;
Label1: TLabel;
Label2: TLabel;
Label3: TLabel;
Button1: TButton;
Edit1: TEdit;
Edit2: TEdit;
Edit3: TEdit;
GroupBox2: TGroupBox;
Label4: TLabel;
Label5: TLabel;
Label6: TLabel;
Label7: TLabel;
Label8: TLabel;
Label9: TLabel;
Label10: TLabel;
Label11: TLabel;
Label12: TLabel;
Label13: TLabel;
DateTimePicker1: TDateTimePicker;
DateTimePicker2: TDateTimePicker;
Panel1: TPanel;
GroupBox3: TGroupBox;
Button3: TButton;
Button2: TButton;
Button4: TButton;
Button5: TButton;
Label14: TLabel;
Label15: TLabel;
Memo1: TMemo;
Memo2: TMemo;
Edit4: TEdit;
Memo3: TMemo;
Memo4: TMemo;
Memo5: TMemo;
Edit5: TEdit;
Edit7: TEdit;
Edit8: TEdit;
ADOQuery1: TADOQuery;
ADOQuery2: TADOQuery;
DataSource1: TDataSource;
DataSource2: TDataSource;
DBGrid1: TDBGrid;
ComboBox1: TComboBox;
ADOQuery3: TADOQuery;
DataSource3: TDataSource;
Button6: TButton;
MainMenu1: TMainMenu;
NewPatient1: TMenuItem;
Label16: TLabel;
Label17: TLabel;
Label18: TLabel;
Label19: TLabel;
Label20: TLabel;
Label21: TLabel;
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure SpeedButton1Click(Sender: TObject);
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Edit1Change(Sender: TObject);
```

```
    procedure FormShow(Sender: TObject);
    procedure DBGrid1CellClick(Column: TColumn);
    procedure Button3Click(Sender: TObject);
    procedure DBGrid1KeyUp(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure Button4Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Memo1Change(Sender: TObject);
    procedure Memo2Change(Sender: TObject);
    procedure ComboBox1Change(Sender: TObject);
    procedure Edit4Change(Sender: TObject);
    procedure Edit5Change(Sender: TObject);
    procedure Edit7Change(Sender: TObject);
    procedure Edit7KeyPress(Sender: TObject; var Key: Char);

  private
    { Private declarations }
  public

    { Public declarations }
  end;

var
  Form2: TForm2;
  DURUM:INTEGER;
  ILLDBG:INTEGER; //ILLNESSES DBGRID'INDEN BİR KAYIT SEÇİLİP
SEÇİLMEDİĞİNİ KONTROL  ETMEK İÇİN

  PROCEDURE DATABASETARIH(); forward;
  PROCEDURE NORMALTARIH(); forward;

  //Delphide bir unit içindeki interface ile implementation arasında
yapılan tüm değişken,procedure,function tanımlamaları
  //bu uniti kullanan tüm unitlerde de kullanılabilir.
  //Ancak Dikkat edildiği üzere yapmış olduğum Procedure
tanımlanımlamasının arkasına FORWARD diye bir reserve kelime koydum.
  //Bunun amacı bu procedure'in içeriğinin ileride olduğunu belirtmek
içindi; Eğer bunu kullanmaz isek delphi şu hatayı verecektir:
  //--- Unsatisfied forward or external declaration:
'TForm2.DATABASETARIH' ---.İşte bu hatayı önlemek için "Forward"
reserve kelimesi
  //kullanulır.
  //Bununla birlikte yukarıda anlatılanları yapmış olsak bile bir hata
ile daha karşılaşabiliriz.bu hata da şudur:
  //--- Field definition not allowed after methods or properties ---.
Bu hatanın sebebi ise değişken tanımlamalarından önce procedure'i
  //tanımladığımız için; yani VAR'dan önce tanımladığımız için.Bunu
çözmek için implementation'dan hemen önce yani tanımlamalardan hemen
  //sonra procedure'mizi veya function'ımızı tanımlarsak hatasız bir
biçimde bu procedure veya function'ları diğer unit'lerde
kullanabiliriz.

  implementation

uses Unit1, Unit7, Unit8, Unit9, Unit4;

{$R *.dfm}

PROCEDURE ILLNESSESDATABASE();
```

```
BEGIN

form2.ADOQuery1.FieldByName('customerid').AsFloat:=strtofloat(form2.Ed
it1.Text);
  //burayı konrol et istersen <<--

form2.ADOQuery1.FieldByName('date').AsDateTime:=form2.DateTimePicker1.
Date;

form2.ADOQuery1.FieldByName('protocolno').AsFloat:=strtofloat(form2.Ed
it7.Text);
  form2.ADOQuery1.FieldByName('diagnosis').AsString:=form2.Memo1.Text;
  form2.ADOQuery1.FieldByName('treatment').AsString:=form2.Memo2.Text;

form2.ADOQuery1.FieldByName('laboratory_results').AsString:=form2.Memo
3.Text;

form2.ADOQuery1.FieldByName('bloodpressure').AsString:=form2.Edit4.Tex
t;
  form2.ADOQuery1.FieldByName('pulse').AsString:=form2.Edit5.Text;

form2.ADOQuery1.FieldByName('givenmedicine').AsString:=form2.Memo4.Tex
t;

form2.ADOQuery1.FieldByName('controldate').AsString:=datetostr(form2.D
ateTimePicker2.Date);
  form2.ADOQuery1.FieldByName('effect').AsString:=form2.Edit8.Text;

form2.ADOQuery1.FieldByName('doctor').AsString:=form2.ComboBox1.Text;
  form2.ADOQuery1.FieldByName('note').AsString:=form2.Memo5.Text;
END;

PROCEDURE ILLNESSESSEARCH();
BEGIN
  FORM2.ADOQuery2.SQL.Text:='SELECT * FROM illnesses where
customerid='+#39+form2.Edit1.Text+#39;
  form2.ADOQuery2.Open;
END;

PROCEDURE COMBOBOXIDOLDUR();
BEGIN
  FORM2.ADOQuery3.SQL.Text:='SELECT distinct doctor from illnesses';
  form2.ADOQuery3.Open;
  form2.ComboBox1.Items.Clear;
  while not form2.ADOQuery3.Eof do
  begin
    form2.ComboBox1.Items.Add(form2.ADOQuery3['doctor']);
    form2.ADOQuery3.Next;
  end;
  FORM2.ADOQuery3.SQL.Text:='SELECT distinct doctor from operations';
  form2.ADOQuery3.Open;
  while not form2.ADOQuery3.Eof do
  begin
    form2.ComboBox1.Items.Add(form2.ADOQuery3['doctor']);
    form2.ADOQuery3.Next;
  end;
END;

PROCEDURE DATABASETARIH();
BEGIN
```

```
  shortdateformat:='YYYY/MM/DD';
  dateseparator:='-';
END;

PROCEDURE NORMALTARIH();
BEGIN
  shortdateformat:='DD/MM/YYYY';
  dateseparator:='.';
END;

PROCEDURE ILLSAVE();
BEGIN
  //ADOQUERY1'İ KAYIT İŞLEMİ İÇİN AÇIYOR
  form2.ADOQuery1.Insert;
  //FORMDAKİ COMPONENTLERDEKİ VERİLERİN DATABASE'DEKİ HANGİ FIELDLERE
YERLEŞECEĞİNİ BELİRTİYOR
  ILLNESSESDATABASE();
  //ADOQUERY1 İLE COMPONENTLERDEKİ VERİLER GEREKLİ YERLERE (DATABASE -
-> TABLE --> FIELDS) ULAŞTIRILDI
  form2.ADOQuery1.Post;
  showmessage('THE RECORD HAS BEEN SAVED SUCCESSFULLY');
  //DBGRID'DE YAPILAN DEĞİŞİKLERİN GÖRÜNTÜLENMESİNİ SAĞLAR
  ILLNESSESSEARCH();
  //DOCTOR İSMİ BÖLÜMÜNÜN YENİ BİR DOKTOR KAYDI YAPILMIŞSA
GÜNCELLENMESİNİ SAĞLAR.
  COMBOBOXIDOLDUR();
END;

PROCEDURE ILLHATAGOSTER();
BEGIN
  IF FORM2.Memo1.Text='' THEN
    FORM2.Label16.Visible:=TRUE
  ELSE
    FORM2.Label16.Visible:=FALSE;

  IF FORM2.Memo2.Text='' THEN
    FORM2.Label18.Visible:=TRUE
  ELSE
    FORM2.Label18.Visible:=FALSE;

  IF FORM2.ComboBox1.Text='SELECT DOCTOR' THEN
    FORM2.Label17.Visible:=TRUE
  ELSE
    FORM2.Label17.Visible:=FALSE;

  IF FORM2.Edit4.Text='' THEN
    FORM2.Label19.Visible:=TRUE
  ELSE
    FORM2.Label19.Visible:=FALSE;

  IF FORM2.Edit5.Text='' THEN
    FORM2.Label20.Visible:=TRUE
  ELSE
    FORM2.Label20.Visible:=FALSE;

  IF FORM2.Edit7.Text='' THEN
    FORM2.Label21.Visible:=TRUE

  ELSE
    FORM2.Label21.Visible:=FALSE;
```

```
END;
PROCEDURE ILLNESSESHATAGOSTERICITEMIZLE();
BEGIN
  FORM2.Label16.Visible:=FALSE;
  FORM2.Label17.Visible:=FALSE;
  FORM2.Label18.Visible:=FALSE;
  FORM2.Label19.Visible:=FALSE;
  FORM2.Label20.Visible:=FALSE;
  FORM2.Label21.Visible:=FALSE;
END;

procedure TForm2.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  ILLNESSESHATAGOSTERICITEMIZLE();
  FORM1.SHOW;
  FORM2.Hide;
  form2.Memo1.Clear;
  form2.Memo3.Clear;
  form2.DateTimePicker1.Date:=date;
  form2.ComboBox1.Text:='SELECT DOCTOR';
  form2.Memo2.Clear;
  form2.Memo4.Clear;
  form2.Memo5.Clear;
  form2.Edit4.Clear;
  form2.Edit5.Clear;
  form2.Edit7.Clear;
  form2.DateTimePicker2.Date:=date;
  form2.Edit8.Clear;
  form2.Edit1.Clear;
  form2.Edit2.Clear;
  form2.Edit3.Clear;
  FORM2.Button5.Click;
  form2.Button5.Caption:='NEW RECORD';
  FORM2.DBGrid1.Visible:=FALSE;
  FORM2.ADOQuery1.Close;
  FORM2.ADOQuery2.Close;
  FORM2.ADOQuery3.Close;
end;

procedure TForm2.SpeedButton1Click(Sender: TObject);
begin
  FORM7.SHOW;
end;

procedure TForm2.Button1Click(Sender: TObject);
begin
  //FORM8'E İSTEĞİN FORM2 (ILLNESSES)'DEN GELDİĞİNİ BELİRTİYOR
  DURUM:=2;
  FORM8.SHOW;
end;

procedure TForm2.Button2Click(Sender: TObject);
begin
  DATABASETARIH();
  IF (FORM2.Edit1.Text <> '') AND (FORM2.Memo1.Text<>'') AND
(FORM2.Memo2.Text<>'') AND (FORM2.Edit4.Text<>'') AND
(FORM2.Edit5.Text<>'') AND (FORM2.Edit7.Text<>'') AND
(FORM2.ComboBox1.Text<>'SELECT DOCTOR') AND
(FORM2.DateTimePicker2.Date > FORM2.DateTimePicker1.Date) AND
```

```
(DATETOSTR(FORM2.DateTimePicker1.Date) <>
DATETOSTR(FORM2.DateTimePicker2.Date)) THEN
  BEGIN
    //PROTOCOL NO'SUNUN ILLNESSES TABLOSUNDA VAR OLUP OLMADIĞINI
KONTROL ETMEK İÇİN ADOQUERY1'E GEREKLİ KOMUTU GÖNDERİYOR...
    form2.ADOQuery1.SQL.Text:='select protocolno from illnesses where
protocolno='+#39+form2.Edit7.Text+#39;
    //ADOQUERY1'DEKİ KOMUTU ÇALIŞTIRIP ILLNESSES TABLOSUNU AÇIYOR...
    form2.ADOQuery1.Open;
    //EĞER PROTOCOL NO ILLNESSES TABLOSUNDA YOKSA RECORD COUNT SONUCU
0 (SIFIR) BULACAKTIR.YANİ BU PROTOCOL NO TABLODA YOKTUR.DOLAYISIYLA
KAYIT YAPILABİLİR.
    if form2.ADOQuery1.RecordCount=0 then
    BEGIN
      FORM2.ADOQuery1.SQL.Clear;
      FORM2.ADOQuery1.SQL.Text:='SELECT * FROM illnesses';
      FORM2.ADOQuery1.Open;
      ILLSAVE();
      ILLNESSESHATAGOSTERICITEMIZLE();
    END
    ELSE
    BEGIN
      MESSAGEBEEP(MB_ICONHAND);
      SHOWMESSAGE('THE PROTOCOL NO HAS BEEN USED BEFORE');
    END;
  END
  ELSE IF FORM2.Edit1.Text='' THEN
  BEGIN
    APPLICATION.MessageBox('PLEASE SELECT PATIENT','SELECT
PATIENT',MB_OK+MB_ICONEXCLAMATION);
  END
  ELSE IF (FORM2.Memo1.Text='') OR (FORM2.Memo2.Text='') OR
(FORM2.Edit4.Text='') OR (FORM2.Edit5.Text='') OR
(FORM2.Edit7.Text='') OR (FORM2.ComboBox1.Text='SELECT DOCTOR') THEN
  BEGIN
    APPLICATION.MessageBox('PLEASE FILL EMPTY PLACE WHICH MARKED WITH
RED ARROWS','FILL EMPTY PLACE',MB_OK+MB_ICONEXCLAMATION);
    ILLHATAGOSTER()
  END
  ELSE IF (FORM2.DateTimePicker2.Date < FORM2.DateTimePicker1.Date)
THEN
  BEGIN
    APPLICATION.MessageBox('CONTROL DATE MUST BE GREATER THAN
DATE','PLEASE CHECK CONTROL DATE',MB_OK+MB_ICONEXCLAMATION);
  END
  ELSE IF (DATETOSTR(FORM2.DateTimePicker1.Date)=
DATETOSTR(FORM2.DateTimePicker2.Date)) THEN
  BEGIN
    APPLICATION.MessageBox('PLEASE CHECK CONTROL DATE AND DATE','CHECK
DATE AND CONTROL DATE',MB_OK+MB_ICONEXCLAMATION);
  END;
  NORMALTARIH();
END;

procedure TForm2.Edit1Change(Sender: TObject);
begin
  //EĞER PATIENT ID KISMI DOLU ISE DBGRID'IN GOSTERİLMESİ VE
BİLGİLERİN VERİTABANINDAN GETİRİLMESİNİ SAĞLAR
  if form2.Edit1.Text <>'' then
  BEGIN
```

```
    form2.DBGrid1.Visible:=true;
    ILLNESSESSEARCH();
  END
  else
    form2.DBGrid1.Visible:=false;

end;

procedure TForm2.FormShow(Sender: TObject);
begin
  COMBOBOXIDOLDUR();
  FORM2.DateTimePicker1.Date:=DATE;
  FORM2.DateTimePicker2.Date:=DATE;
  ILLDBG:=0;
end;

procedure TForm2.DBGrid1CellClick(Column: TColumn);
begin
  IF (FORM2.DBGrid1.FieldCount <> 0) AND (FORM2.DBGrid1.Fields[0].Text
<> '') AND (FORM2.DBGrid1.Fields[0].Text <> NULL) THEN
  BEGIN
    FORM2.Edit1.Text:=FORM2.DBGrid1.Fields[0].Text;

FORM2.DateTimePicker1.Date:=STRTODATE(FORM2.DBGrid1.Fields[1].Text);
    FORM2.Edit7.Text:=FORM2.DBGrid1.Fields[2].Text;
    FORM2.Memo1.Text:=FORM2.DBGrid1.Fields[3].Text;
    FORM2.Memo2.Text:=FORM2.DBGrid1.Fields[4].Text;
    FORM2.Memo3.Text:=FORM2.DBGrid1.Fields[5].Text;
    FORM2.Edit4.Text:=FORM2.DBGrid1.Fields[6].Text;
    FORM2.Edit5.Text:=FORM2.DBGrid1.Fields[7].Text;
    FORM2.Memo4.Text:=FORM2.DBGrid1.Fields[8].Text;

FORM2.DateTimePicker2.Date:=STRTODATE(FORM2.DBGrid1.Fields[9].Text);
    FORM2.Edit8.Text:=FORM2.DBGrid1.Fields[10].Text;
    FORM2.ComboBox1.Text:=FORM2.DBGrid1.Fields[11].Text;
    FORM2.Memo5.Text:=FORM2.DBGrid1.Fields[12].Text;
    ILLDBG:=1;
  END
  ELSE IF (FORM2.DBGrid1.FieldCount = 0) THEN
    APPLICATION.MessageBox('DATASET CLOSED.YOU CAN NOT SELECT ANY
ITEM','ATTENTION',MB_OK+MB_ICONEXCLAMATION)
   ELSE IF (FORM2.DBGrid1.Fields[0].Text = '') THEN
    APPLICATION.MessageBox('THERE IS NO DATA ABOUT THIS
PATIENT','ATTENTION',MB_OK+MB_ICONEXCLAMATION);
end;

procedure TForm2.Button3Click(Sender: TObject);

begin
  DATABASETARIH();
  IF (FORM2.Edit1.Text <> '') AND (ILLDBG = 1) AND
(FORM2.Memo1.Text<>'') AND (FORM2.Memo2.Text<>'') AND
(FORM2.Edit4.Text<>'') AND (FORM2.Edit5.Text<>'') AND
(FORM2.Edit7.Text<>'') AND (FORM2.ComboBox1.Text<>'SELECT DOCTOR') AND
(FORM2.DateTimePicker2.Date > FORM2.DateTimePicker1.Date) AND
(DATETOSTR(FORM2.DateTimePicker1.Date) <>
DATETOSTR(FORM2.DateTimePicker2.Date)) THEN

BEGIN
```

```
    FORM2.ADOQuery1.SQL.Text:='SELECT * FROM illnesses WHERE
customerid='+#39+form2.Edit1.Text+#39+' and
protocolno='+#39+form2.Edit7.Text+#39;
    FORM2.ADOQuery1.Open;
    FORM2.ADOQuery1.Edit;
    ILLNESSESDATABASE();
    FORM2.ADOQuery1.Post;
    SHOWMESSAGE('THE RECORD HAS BEEN UPDATED SUCCESSFULLY');
    ILLNESSESHATAGOSTERICITEMIZLE();
    ILLNESSESSEARCH();
    COMBOBOXIDOLDUR();
    ILLDBG:=0;
  END
  ELSE IF FORM2.Edit1.Text='' THEN
  BEGIN
    APPLICATION.MessageBox('PLEASE SELECT PATIENT','SELECT
PATIENT',MB_OK+MB_ICONEXCLAMATION);
  END
  ELSE IF (FORM2.Edit1.Text <> '') AND (ILLDBG=0) THEN
  BEGIN
    APPLICATION.MessageBox('PLEASE SELECT ILLNESSES INFORMATION FROM
ILLNESSES LIST THAT WILL UPDATE','SELECT
ILLNESSES',MB_OK+MB_ICONEXCLAMATION);
  END
  ELSE IF (FORM2.Memo1.Text='') OR (FORM2.Memo2.Text='') OR
(FORM2.Edit4.Text='') OR (FORM2.Edit5.Text='') OR
(FORM2.Edit7.Text='') OR (FORM2.ComboBox1.Text='SELECT DOCTOR') THEN
  BEGIN
    APPLICATION.MessageBox('PLEASE FILL EMPTY PLACE WHICH MARKED WITH
RED ARROWS','FILL EMPTY PLACE',MB_OK+MB_ICONEXCLAMATION);
    ILLHATAGOSTER()
  END
  ELSE IF (FORM2.DateTimePicker2.Date < FORM2.DateTimePicker1.Date)
THEN
  BEGIN
    APPLICATION.MessageBox('CONTROL DATE MUST BE GREATER THAN
DATE','PLEASE CHECK CONTROL DATE',MB_OK+MB_ICONEXCLAMATION);
  END
  ELSE IF (DATETOSTR(FORM2.DateTimePicker1.Date)=
DATETOSTR(FORM2.DateTimePicker2.Date)) THEN
  BEGIN
    APPLICATION.MessageBox('PLEASE CHECK CONTROL DATE AND
DATE','PLEASE CHECK DATE AND CONTROL DATE',MB_OK+MB_ICONEXCLAMATION);
  END;
  NORMALTARIH();
END;

procedure TForm2.DBGrid1KeyUp(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  FORM2.DBGrid1CellClick(NIL);
end;

procedure TForm2.Button4Click(Sender: TObject);
VAR
  SIL:WORD;
begin
  DATABASETARIH();

IF (FORM2.Edit1.Text <>'') AND (ILLDBG=1) THEN
```

```
    BEGIN
    SIL:=APPLICATION.MessageBox('ARE YOU SURE TO
DELETE','CAUTION',MB_YESNO+MB_ICONQUESTION);
    IF SIL=IDYES THEN
    BEGIN
      FORM2.ADOQuery1.SQL.Text:='delete FROM illnesses WHERE
customerid='+#39+form2.Edit1.Text+#39+' and
protocolno='+#39+form2.Edit7.Text+#39;
      form2.ADOQuery1.ExecSQL;
      ILLNESSESSEARCH();
      FORM2.Button5.Click; //EKRANI TEMİZLEMESİ İÇİN
      COMBOBOXIDOLDUR();
      ILLDBG:=0;
    END;
  END
  ELSE IF FORM2.Edit1.Text='' THEN
  BEGIN
    APPLICATION.MessageBox('PLEASE SELECT PATIENT','SELECT
PATIENT',MB_OK+MB_ICONEXCLAMATION);
  END
  ELSE IF (FORM2.Edit1.Text<>'') AND (ILLDBG=0) THEN
  BEGIN
    APPLICATION.MessageBox('PLEASE SELECT ILLNESSES FROM ILLNESSES
LIST THAT YOU WANT TO DELETE','SELECT
OPERATION',MB_OK+MB_ICONEXCLAMATION);
  END;
  NORMALTARIH();
end;

procedure TForm2.Button6Click(Sender: TObject);
begin
  form2.Edit1.Clear;
  form2.Edit2.Clear;
  form2.Edit3.Clear;
  FORM2.Button5.Click;
  form2.Button5.Caption:='NEW RECORD';
  FORM2.DBGrid1.Visible:=FALSE;
end;

procedure TForm2.Button5Click(Sender: TObject);
begin
  form2.Memo1.Clear;
  form2.Memo3.Clear;
  form2.DateTimePicker1.Date:=date;
  form2.ComboBox1.Text:='SELECT DOCTOR';
  form2.Memo2.Clear;
  form2.Memo4.Clear;
  form2.Memo5.Clear;
  form2.Edit4.Clear;
  form2.Edit5.Clear;
  form2.Edit7.Clear;
  form2.DateTimePicker2.Date:=date;
  form2.Edit8.Clear;
  ILLNESSESHATAGOSTERICITEMIZLE();
  COMBOBOXIDOLDUR();
  ILLNESSESSEARCH();
end;

procedure TForm2.Edit7KeyPress(Sender: TObject; var Key: Char);
begin
```

```
  //SADECE RAKAMLAR GİREBİLMEK İÇİN, BACKSPACE VE DELETE TUŞUNA
BASABİLMEK İÇİN
  IF (KEY<>'0') AND (KEY<>'1') AND (KEY<>'2') AND (KEY<>'3') AND
(KEY<>'4') AND (KEY<>'5') AND (KEY<>'6') AND (KEY<>'7') AND (KEY<>'8')
AND (KEY<>'9') AND (KEY<>#8) AND (KEY<>#127) THEN
    //EĞER BAŞKA BİR TUŞ BASILMIŞSA İPTAL ETMESİ İÇİN
    KEY:=#0;
end;

end.
```

3. UNIT3 CODES

```
unit Unit3;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, Grids, DBGrids, ComCtrls, StdCtrls, Buttons, ExtCtrls,
Menus, DB, ADODB;

type
  TForm3 = class(TForm)
    GroupBox1: TGroupBox;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Button1: TButton;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    GroupBox3: TGroupBox;
    Label14: TLabel;
    Label16: TLabel;
    Label17: TLabel;
    Label18: TLabel;
    Label19: TLabel;
    Label20: TLabel;
    Label21: TLabel;
    Label22: TLabel;
    Label23: TLabel;
    Memo7: TMemo;
    Edit8: TEdit;
    Edit9: TEdit;
    Memo8: TMemo;
    DateTimePicker3: TDateTimePicker;
    Edit11: TEdit;
    DateTimePicker4: TDateTimePicker;
    Edit12: TEdit;
    Memo1: TMemo;
    Label4: TLabel;
    GroupBox2: TGroupBox;
    Button5: TButton;
    Button4: TButton;
    Button3: TButton;
    Button2: TButton;
```

```
    DBGrid1: TDBGrid;
    Panel1: TPanel;
    Label5: TLabel;
    Edit4: TEdit;
    MainMenu1: TMainMenu;
    OL1: TMenuItem;
    ComboBox1: TComboBox;
    Button6: TButton;
    ADOQuery1: TADOQuery;
    ADOQuery2: TADOQuery;
    ADOQuery3: TADOQuery;
    DataSource1: TDataSource;
    DataSource2: TDataSource;
    DataSource3: TDataSource;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure Button1Click(Sender: TObject);
    procedure SpeedButton2Click(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure DBGrid1CellClick(Column: TColumn);
    procedure DBGrid1KeyUp(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure Edit1Change(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
    procedure Edit11KeyPress(Sender: TObject; var Key: Char);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form3: TForm3;
  OPRTNDBG: INTEGER;
implementation

uses Unit1, Unit8, Unit7, Unit2, Unit4;

{$R *.dfm}


PROCEDURE OPERATIONSDATABASE();
BEGIN

form3.ADOQuery1.FieldByName('customerid').AsFloat:=strtofloat(form3.Ed
it1.Text);

form3.ADOQuery1.FieldByName('date').AsDateTime:=form3.DateTimePicker3.
Date;

form3.ADOQuery1.FieldByName('operationname').AsString:=form3.Edit12.Te
xt;
```

```
form3.ADOQuery1.FieldByName('protocolno').AsFloat:=strtofloat(form3.Ed
it11.Text);

form3.ADOQuery1.FieldByName('laboratory_results').AsString:=form3.Memo
7.Text;

form3.ADOQuery1.FieldByName('bloodpressure').AsString:=form3.Edit8.Tex
t;
  form3.ADOQuery1.FieldByName('pulse').AsString:=form3.Edit9.Text;

form3.ADOQuery1.FieldByName('givenmedicine').AsString:=form3.Memo8.Tex
t;

form3.ADOQuery1.FieldByName('controldate').AsString:=datetostr(form3.D
ateTimePicker4.Date);
  form3.ADOQuery1.FieldByName('effect').AsString:=form3.Edit4.Text;

form3.ADOQuery1.FieldByName('doctor').AsString:=form3.ComboBox1.Text;
  form3.ADOQuery1.FieldByName('note').AsString:=form3.Memo1.Text;
END;

PROCEDURE OPERATIONSSEARCH();
BEGIN
  FORM3.ADOQuery2.SQL.Text:='SELECT * FROM  operations where
customerid='+#39+form3.Edit1.Text+#39;
  form3.ADOQuery2.Open;
END;

PROCEDURE OPRCOMBOBOXINIDOLDUR();
BEGIN
  FORM3.ADOQuery3.SQL.Text:='SELECT distinct doctor from illnesses';
  form3.ADOQuery3.Open;
  form3.ComboBox1.Items.Clear;
  while not form3.ADOQuery3.Eof do
  begin
    form3.ComboBox1.Items.Add(form3.ADOQuery3['doctor']);
    form3.ADOQuery3.Next;
  end;
  FORM3.ADOQuery3.SQL.Text:='SELECT distinct doctor from operations';
  form3.ADOQuery3.Open;
  while not form3.ADOQuery3.Eof do
  begin
    form3.ComboBox1.Items.Add(form3.ADOQuery3['doctor']);
    form3.ADOQuery3.Next;
  end;
END;

PROCEDURE OPERATIONSHATAGOSTERICITEMIZLE();
BEGIN
  FORM3.Label6.Visible:=FALSE;
  FORM3.Label7.Visible:=FALSE;
  FORM3.Label8.Visible:=FALSE;
  FORM3.Label9.Visible:=FALSE;
  FORM3.Label10.Visible:=FALSE;
END;

PROCEDURE OPERATIONSSAVE();
BEGIN
  //ADOQUERY1'İ KAYIT İŞLEMİ İÇİN AÇIYOR
```

```
  form3.ADOQuery1.Insert;
  //FORMDAKİ COMPONENTLERDEKİ VERİLERİN DATABASE'DEKİ HANGİ FIELDLERE
YERLEŞECEĞİNİ BELİRTİYOR
   OPERATIONSDATABASE();
   //ADOQUERY1 İLE COMPONENTLERDEKİ VERİLER GEREKLİ YERLERE (DATABASE -
-> TABLE --> FIELDS) ULAŞTIRILDI
   form3.ADOQuery1.Post;
   showmessage('THE RECORD HAS BEEN SAVED SUCCESSFULLY');
   //DBGRID'DE YAPILAN DEĞİŞİKLERİN GÖRÜNTÜLENMESİNİ SAĞLAR
   OPERATIONSSEARCH();
   //DOCTOR İSMİ BÖLÜMÜNÜN YENİ BİR DOKTOR KAYDI YAPILMIŞSA
GÜNCELLENMESİNİ SAĞLAR.
   OPRCOMBOBOXINIDOLDUR();
END;

PROCEDURE OPRHATAGOSTER();
BEGIN
   IF FORM3.Edit12.Text='' THEN
     FORM3.Label6.Visible:=TRUE
   ELSE
     FORM3.Label6.Visible:=FALSE;

   IF FORM3.Edit11.Text='' THEN
     FORM3.Label9.Visible:=TRUE
   ELSE
     FORM3.Label9.Visible:=FALSE;

   IF FORM3.ComboBox1.Text='SELECT DOCTOR' THEN
     FORM3.Label10.Visible:=TRUE
   ELSE
     FORM3.Label10.Visible:=FALSE;

   IF FORM3.Edit8.Text='' THEN
     FORM3.Label7.Visible:=TRUE
   ELSE
     FORM3.Label7.Visible:=FALSE;

   IF FORM3.Edit9.Text='' THEN
     FORM3.Label8.Visible:=TRUE
   ELSE
     FORM3.Label8.Visible:=FALSE;
END;

procedure TForm3.FormClose(Sender: TObject; var Action: TCloseAction);
begin
   OPERATIONSHATAGOSTERICITEMIZLE();
   FORM1.SHOW;
   FORM3.Hide;
   form3.Edit1.Clear;
   form3.Edit2.Clear;
   form3.Edit3.Clear;
   FORM3.Edit12.Clear;
   form3.DateTimePicker3.Date:=date;
   form3.ComboBox1.Text:='SELECT DOCTOR';
   form3.Memo7.Clear;
   form3.Memo8.Clear;
   form3.Memo1.Clear;
   form3.Edit8.Clear;
   form3.Edit9.Clear;
   form3.Edit11.Clear;
```

```
  form3.DateTimePicker4.Date:=date;
  form3.Edit4.Clear;
  form3.Button3.Caption:='NEW RECORD';
  FORM3.DBGrid1.Visible:=FALSE;
  FORM3.ADOQuery1.Close;
  FORM3.ADOQuery2.Close;
  FORM3.ADOQuery3.Close;

end;

procedure TForm3.Button1Click(Sender: TObject);
begin
  //FORM8'E İSTEĞİN FORM2 (ILLNESSES)'DEN GELDİĞİNİ BELİRTİYOR
  DURUM:=3;
  FORM8.SHOW;
end;

procedure TForm3.SpeedButton2Click(Sender: TObject);
begin
FORM7.SHOW;
end;

procedure TForm3.FormShow(Sender: TObject);
begin
  OPRCOMBOBOXINIDOLDUR();
  FORM3.DateTimePicker3.Date:=DATE;
  FORM3.DateTimePicker4.Date:=DATE;
  OPRTNDBG:=0;
end;

procedure TForm3.DBGrid1CellClick(Column: TColumn);
begin
  IF (FORM3.DBGrid1.FieldCount <> 0) AND (FORM3.DBGrid1.Fields[0].Text
<> '') AND (FORM3.DBGrid1.Fields[0].Text <> NULL) THEN
  BEGIN
    FORM3.Edit1.Text:=FORM3.DBGrid1.Fields[0].Text;

FORM3.DateTimePicker3.Date:=STRTODATE(FORM3.DBGrid1.Fields[1].Text);
    FORM3.Edit12.Text:=FORM3.DBGrid1.Fields[2].Text;
    FORM3.Edit11.Text:=FORM3.DBGrid1.Fields[3].Text;
    FORM3.Memo8.Text:=FORM3.DBGrid1.Fields[4].Text;
    FORM3.Edit8.Text:=FORM3.DBGrid1.Fields[5].Text;
    FORM3.Edit9.Text:=FORM3.DBGrid1.Fields[6].Text;
    FORM3.Memo8.Text:=FORM3.DBGrid1.Fields[7].Text;

FORM3.DateTimePicker4.Date:=STRTODATE(FORM3.DBGrid1.Fields[8].Text);
    FORM3.Edit4.Text:=FORM3.DBGrid1.Fields[9].Text;
    FORM3.ComboBox1.Text:=FORM3.DBGrid1.Fields[10].Text;
    FORM3.Memo1.Text:=FORM3.DBGrid1.Fields[11].Text;
    OPRTNDBG:=1;
  END
  ELSE IF (FORM3.DBGrid1.FieldCount = 0) THEN
    APPLICATION.MessageBox('DATASET CLOSED.YOU CAN NOT SELECT ANY
ITEM','ATTENTION',MB_OK+MB_ICONEXCLAMATION)
   ELSE IF (FORM3.DBGrid1.Fields[0].Text = '') THEN
    APPLICATION.MessageBox('THERE IS NO DATA ABOUT THIS
PATIENT','ATTENTION',MB_OK+MB_ICONEXCLAMATION);
end;

procedure TForm3.DBGrid1KeyUp(Sender: TObject; var Key: Word;
```

```
    Shift: TShiftState);
  begin
    FORM3.DBGrid1CellClick(NIL);
  end;

  procedure TForm3.Edit1Change(Sender: TObject);
  begin
    //EĞER PATIENT ID KISMI DOLU ISE DBGRID'IN GOSTERİLMESİ VE
  BİLGİLERİN VERİTABANINDAN GETİRİLMESİNİ SAĞLAR
    if form3.Edit1.Text <>'' then
    BEGIN
      form3.DBGrid1.Visible:=true;
      OPERATIONSSEARCH();
    END
    else
      form3.DBGrid1.Visible:=false;
  end;

  procedure TForm3.Button4Click(Sender: TObject);
  begin
    DATABASETARIH();
    IF (FORM3.Edit1.Text <> '') AND (FORM3.Edit12.Text<>'') AND
  (FORM3.Edit11.Text<>'') AND (FORM3.Edit8.Text<>'') AND
  (FORM3.Edit9.Text<>'')  AND (FORM3.ComboBox1.Text<>'SELECT DOCTOR')
  AND (FORM3.DateTimePicker4.Date > FORM3.DateTimePicker3.Date) AND
  (DATETOSTR(FORM3.DateTimePicker3.Date) <>
  DATETOSTR(FORM3.DateTimePicker4.Date)) THEN
    BEGIN
      //PROTOCOL NO'SUNUN OPERATIONS TABLOSUNDA VAR OLUP OLMADIĞINI
  KONTROL ETMEK İÇİN ADOQUERY1'E GEREKLİ KOMUTU GÖNDERİYOR...
      form3.ADOQuery1.SQL.Text:='select protocolno from operations where
  protocolno='+#39+form3.Edit11.Text+#39;
      //ADOQUERY1'DEKİ KOMUTU ÇALIŞTIRIP ILLNESSES TABLOSUNU AÇIYOR...
      form3.ADOQuery1.Open;
      //EĞER PROTOCOL NO ILLNESSES TABLOSUNDA YOKSA RECORD COUNT SONUCU
  0 (SIFIR) BULACAKTIR.YANİ BU PROTOCOL NO TABLODA YOKTUR.DOLAYISIYLA
  KAYIT YAPILABİLİR.
      if form3.ADOQuery1.RecordCount=0 then
      BEGIN
        FORM3.ADOQuery1.SQL.Clear;
        FORM3.ADOQuery1.SQL.Text:='SELECT * FROM operations';
        FORM3.ADOQuery1.Open;
        OPERATIONSSAVE();
        OPERATIONSHATAGOSTERICITEMIZLE();
      END
      ELSE
      BEGIN
        MESSAGEBEEP(MB_ICONHAND);
        SHOWMESSAGE('THE PROTOCOL NO HAS BEEN USED BEFORE');
      END;
    END
    ELSE IF FORM3.Edit1.Text='' THEN
    BEGIN
      APPLICATION.MessageBox('PLEASE SELECT PATIENT','SELECT
  PATIENT',MB_OK+MB_ICONEXCLAMATION);
    END
    ELSE IF (FORM3.Edit12.Text='') OR (FORM3.Edit8.Text='') OR
  (FORM3.Edit9.Text='') OR (FORM3.Edit11.Text='') OR
  (FORM3.ComboBox1.Text='SELECT DOCTOR') THEN
    BEGIN
```

```
    APPLICATION.MessageBox('PLEASE FILL EMPTY PLACE WHICH MARKED WITH
RED ARROWS','FILL EMPTY PLACE',MB_OK+MB_ICONEXCLAMATION);
    OPRHATAGOSTER();
  END
  ELSE IF (FORM3.DateTimePicker4.Date < FORM3.DateTimePicker3.Date)
THEN
  BEGIN
    APPLICATION.MessageBox('CONTROL DATE MUST BE GREATER THAN
DATE','PLEASE CHECK CONTROL DATE',MB_OK+MB_ICONEXCLAMATION);
   END
  ELSE IF (DATETOSTR(FORM3.DateTimePicker3.Date)=
DATETOSTR(FORM3.DateTimePicker4.Date)) THEN
  BEGIN
    APPLICATION.MessageBox('PLEASE CHECK CONTROL DATE AND DATE','CHECK
DATE AND CONTROL DATE',MB_OK+MB_ICONEXCLAMATION);
  END;
  NORMALTARIH();
  OPRTNDBG:=0;
end;

procedure TForm3.Button5Click(Sender: TObject);
begin
DATABASETARIH();
  IF (FORM3.Edit1.Text <> '') AND (OPRTNDBG = 1) AND
(FORM3.Edit8.Text<>'') AND (FORM3.Edit9.Text<>'') AND
(FORM3.Edit12.Text<>'') AND (FORM3.Edit11.Text<>'') AND
(FORM3.ComboBox1.Text<>'SELECT DOCTOR') AND
(FORM3.DateTimePicker4.Date > FORM3.DateTimePicker3.Date) AND
(DATETOSTR(FORM3.DateTimePicker3.Date) <>
DATETOSTR(FORM3.DateTimePicker4.Date)) THEN
  BEGIN
    FORM3.ADOQuery1.SQL.Text:='SELECT * FROM operations WHERE
customerid='+#39+form3.Edit1.Text+#39+' and
protocolno='+#39+form3.Edit11.Text+#39;
    FORM3.ADOQuery1.Open;
    FORM3.ADOQuery1.Edit;
    OPERATIONSDATABASE();
    FORM3.ADOQuery1.Post;
    SHOWMESSAGE('THE RECORD HAS BEEN UPDATED SUCCESSFULLY');
    OPERATIONSHATAGOSTERICITEMIZLE();
    OPERATIONSSEARCH();
    OPRCOMBOBOXINIDOLDUR();
    OPRTNDBG:=0;
  END
  ELSE IF FORM3.Edit1.Text='' THEN
  BEGIN
    APPLICATION.MessageBox('PLEASE SELECT PATIENT','SELECT
PATIENT',MB_OK+MB_ICONEXCLAMATION);
  END
  ELSE IF (FORM3.Edit1.Text <> '') AND (OPRTNDBG=0) THEN
  BEGIN
    APPLICATION.MessageBox('PLEASE SELECT OPERATIONS INFORMATION FROM
OPERATIONS LIST THAT WILL UPDATE','SELECT
OPERATION',MB_OK+MB_ICONEXCLAMATION);
  END
  ELSE IF (FORM3.Memo1.Text='') OR (FORM3.Edit12.Text='') OR
(FORM3.Edit8.Text='') OR (FORM3.Edit9.Text='') OR
(FORM3.Edit11.Text='') OR (FORM3.ComboBox1.Text='SELECT DOCTOR') THEN
  BEGIN
```

```
    APPLICATION.MessageBox('PLEASE FILL EMPTY PLACE WHICH MARKED WITH
RED ARROWS','FILL EMPTY PLACE',MB_OK+MB_ICONEXCLAMATION);
    OPRHATAGOSTER()
  END
  ELSE IF (FORM3.DateTimePicker4.Date < FORM3.DateTimePicker3.Date)
THEN
  BEGIN
    APPLICATION.MessageBox('CONTROL DATE MUST BE GREATER THAN
DATE','PLEASE CHECK CONTROL DATE',MB_OK+MB_ICONEXCLAMATION);
  END
  ELSE IF (DATETOSTR(FORM3.DateTimePicker3.Date)=
DATETOSTR(FORM3.DateTimePicker4.Date)) THEN
  BEGIN
    APPLICATION.MessageBox('PLEASE CHECK CONTROL DATE AND
DATE','PLEASE CHECK DATE AND CONTROL DATE',MB_OK+MB_ICONEXCLAMATION);
  END;
  NORMALTARIH();
end;

procedure TForm3.Button2Click(Sender: TObject);
VAR
  SIL:WORD;
begin
  DATABASETARIH();
  IF (FORM3.Edit1.Text <>'') AND (OPRTNDBG=1) THEN
    BEGIN
    SIL:=APPLICATION.MessageBox('ARE YOU SURE TO
DELETE','CAUTION',MB_YESNO+MB_ICONQUESTION);
    IF SIL=IDYES THEN
    BEGIN
      FORM3.ADOQuery1.SQL.Text:='delete FROM operations WHERE
customerid='+#39+form3.Edit1.Text+#39+' and
protocolno='+#39+form3.Edit11.Text+#39;
      form3.ADOQuery1.ExecSQL;
      OPERATIONSSEARCH();
      FORM3.Button3.Click; //EKRANI TEMİZLEMESİ İÇİN
      OPRCOMBOBOXINIDOLDUR();
      OPRTNDBG:=0;
    END;
  END
  ELSE IF FORM3.Edit1.Text='' THEN
  BEGIN
    APPLICATION.MessageBox('PLEASE SELECT PATIENT','SELECT
PATIENT',MB_OK+MB_ICONEXCLAMATION);
  END
  ELSE IF (FORM3.Edit1.Text<>'') AND (OPRTNDBG=0) THEN
  BEGIN
    APPLICATION.MessageBox('PLEASE SELECT OPERATION FROM OPERATIONS
LIST THAT YOU WANT TO DELETE','SELECT
OPERATION',MB_OK+MB_ICONEXCLAMATION);
  END;
  NORMALTARIH();
end;

procedure TForm3.Button3Click(Sender: TObject);
begin
  FORM3.Edit12.Clear;
  form3.DateTimePicker3.Date:=date;
  form3.ComboBox1.Text:='SELECT DOCTOR';
  form3.Memo7.Clear;
```

```
  form3.Memo8.Clear;
  form3.Memo1.Clear;
  form3.Edit8.Clear;
  form3.Edit9.Clear;
  form3.Edit11.Clear;
  form3.DateTimePicker4.Date:=date;
  form3.Edit4.Clear;
  OPERATIONSHATAGOSTERICITEMIZLE();
  OPRCOMBOBOXINIDOLDUR();
  OPERATIONSSEARCH();
end;

procedure TForm3.Button6Click(Sender: TObject);
begin
  form3.Edit1.Clear;
  form3.Edit2.Clear;
  form3.Edit3.Clear;
  FORM3.Button3.Click;
  form3.Button3.Caption:='NEW RECORD';
  FORM3.DBGrid1.Visible:=FALSE;
end;

procedure TForm3.Edit11KeyPress(Sender: TObject; var Key: Char);
begin
  //SADECE RAKAMLAR GİREBİLMEK İÇİN, BACKSPACE VE DELETE TUŞUNA
BASABİLMEK İÇİN
  IF (KEY<>'0') AND (KEY<>'1') AND (KEY<>'2') AND (KEY<>'3') AND
(KEY<>'4') AND (KEY<>'5') AND (KEY<>'6') AND (KEY<>'7') AND (KEY<>'8')
AND (KEY<>'9') AND (KEY<>#8) AND (KEY<>#127) THEN
    //EĞER BAŞKA BİR TUŞ BASILMIŞSA İPTAL ETMESİ İÇİN
    KEY:=#0;
end;

end.
```

4. UNIT4 CODES

```
unit Unit4;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls, Mask, ExtCtrls, Grids, DBGrids, DB, ADODB,
ComCtrls, Menus;

type
  TForm4 = class(TForm)
    GroupBox8: TGroupBox;
    Label33: TLabel;
    Edit17: TEdit;
    GroupBox4: TGroupBox;
    Label15: TLabel;
    Label24: TLabel;
    Label25: TLabel;
    Edit13: TEdit;
    Edit14: TEdit;
```

```
    GroupBox5: TGroupBox;
    Label26: TLabel;
    Label27: TLabel;
    Label28: TLabel;
    Label29: TLabel;
    Label30: TLabel;
    Edit15: TEdit;
    Memo5: TMemo;
    MaskEdit2: TMaskEdit;
    MaskEdit3: TMaskEdit;
    Edit16: TEdit;
    GroupBox6: TGroupBox;
    Label31: TLabel;
    Label32: TLabel;
    Memo6: TMemo;
    Memo9: TMemo;
    GroupBox7: TGroupBox;
    Button6: TButton;
    Button7: TButton;
    GroupBox1: TGroupBox;
    Label1: TLabel;
    Edit1: TEdit;
    Label2: TLabel;
    Edit2: TEdit;
    Label3: TLabel;
    ComboBox1: TComboBox;
    Label4: TLabel;
    Memo1: TMemo;
    DBGrid1: TDBGrid;
    Panel1: TPanel;
    Button1: TButton;
    Button2: TButton;
    DateTimePicker1: TDateTimePicker;
    ADOQuery1: TADOQuery;
    ADOQuery2: TADOQuery;
    DataSource1: TDataSource;
    DataSource2: TDataSource;
    ADOConnection1: TADOConnection;
    MainMenu1: TMainMenu;
    OL1: TMenuItem;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    ADOQuery3: TADOQuery;
    DataSource3: TDataSource;

  procedure Button3Click(Sender: TObject);
  procedure FormClose(Sender: TObject; var Action: TCloseAction);
  procedure Button6Click(Sender: TObject);
  procedure Button7Click(Sender: TObject);
  procedure DBGrid1CellClick(Column: TColumn);
  procedure Button2Click(Sender: TObject);
  procedure Button1Click(Sender: TObject);
  procedure DBGrid1KeyUp(Sender: TObject; var Key: Word;
    Shift: TShiftState);
  procedure Edit15KeyPress(Sender: TObject; var Key: Char);
  procedure FormShow(Sender: TObject);
```

```
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form4: TForm4;
  PTNTDBG:INTEGER;
implementation

uses Unit1, Unit2;

{$R *.dfm}

PROCEDURE PATIENTSEARCH(); //DBGRID'in refresh yapıp yeni yapılan
işlemi göstermesi için
BEGIN
  FORM4.ADOQuery2.SQL.Text:='SELECT * FROM PATIENTS';
  FORM4.ADOQuery2.Open;
END;

PROCEDURE COMPONENTDATABASE ();
BEGIN
  form4.ADOQuery1.FieldByName('name').AsString:=form4.Edit13.Text;
  form4.ADOQuery1.FieldByName('surname').AsString:=form4.Edit14.Text;

form4.ADOQuery1.FieldByName('birthdate').AsString:=datetostr(form4.Dat
eTimePicker1.Date);

form4.ADOQuery1.FieldByName('tcidno').AsFloat:=strtofloat(form4.Edit15
.Text);
  form4.ADOQuery1.FieldByName('adress').AsString:=form4.Memo5.Text;
  form4.ADOQuery1.FieldByName('telno').AsString:=form4.MaskEdit2.Text;

form4.ADOQuery1.FieldByName('mobileno').AsString:=form4.MaskEdit3.Text
;
  form4.ADOQuery1.FieldByName('email').AsString:=form4.Edit16.Text;

form4.ADOQuery1.FieldByName('cronological_illnesses').AsString:=form4.
Memo6.Text;

form4.ADOQuery1.FieldByName('cronological_medicine').AsString:=form4.M
emo9.Text;

form4.ADOQuery1.FieldByName('bloodgroup').AsString:=form4.ComboBox1.Te
xt;

form4.ADOQuery1.FieldByName('mothername').AsString:=form4.Edit1.Text;

form4.ADOQuery1.FieldByName('fathername').AsString:=form4.Edit2.Text;
  form4.ADOQuery1.FieldByName('note').AsString:=form4.Memo1.Text;
END;

PROCEDURE PATIENTKAYDET();
BEGIN
  FORM4.ADOQuery1.SQL.Text:='select * from patients';
  form4.ADOQuery1.Open;
  form4.ADOQuery1.Insert;
  COMPONENTDATABASE();
```

```
  form4.ADOQuery1.Post;
  showmessage('RECORD HAS BEEN SAVED SUCCESFULLY'+#13+'---------------
----------------------------------'+#13+#13+'NAME.......:
'+FORM4.Edit13.Text+#13+'SURNAME..: '+FORM4.Edit14.Text+#13+'PATIENT
ID: '+inttostr(FORM4.ADOQuery1['customerid']));
  form4.Edit17.Text:=inttostr(FORM4.ADOQuery1['customerid']);
  PATIENTSEARCH();
END;

PROCEDURE PATIENTHATAGOSTERICI();
BEGIN
  IF FORM4.Edit13.Text='' THEN
    FORM4.Label5.Visible:=TRUE
  ELSE
    FORM4.Label5.Visible:=FALSE;

  IF FORM4.Edit14.Text='' THEN
    FORM4.Label6.Visible:=TRUE
  ELSE
    FORM4.Label6.Visible:=FALSE;

  IF FORM4.Edit1.Text='' THEN
    FORM4.Label7.Visible:=TRUE
  ELSE
    FORM4.Label7.Visible:=FALSE;

  IF FORM4.Edit2.Text='' THEN
    FORM4.Label8.Visible:=TRUE
  ELSE
    FORM4.Label8.Visible:=FALSE;

  IF FORM4.Edit15.Text='' THEN
    FORM4.Label9.Visible:=TRUE
  ELSE
    FORM4.Label9.Visible:=FALSE;

  IF FORM4.Memo5.Text='' THEN
    FORM4.Label10.Visible:=TRUE
  ELSE
    FORM4.Label10.Visible:=FALSE;
END;

PROCEDURE PATIENTHATAGOSTERICITEMIZLE();
BEGIN
  FORM4.Label5.Visible:=FALSE;
  FORM4.Label6.Visible:=FALSE;
  FORM4.Label7.Visible:=FALSE;
  FORM4.Label8.Visible:=FALSE;
  FORM4.Label9.Visible:=FALSE;
  FORM4.Label10.Visible:=FALSE;
END;

PROCEDURE PATIENTFORMUNUTEMIZLE();
BEGIN
  FORM4.Edit17.Clear;
  FORM4.Edit13.Clear;
  FORM4.Edit14.Clear;
  FORM4.Edit1.Clear;
  FORM4.Edit2.Clear;
  FORM4.Edit15.Clear;
```

```
  FORM4.Edit16.Clear;
  FORM4.Memo5.Clear;
  FORM4.Memo6.Clear;
  FORM4.Memo9.Clear;
  FORM4.Memo1.Clear;
  FORM4.MaskEdit2.Clear;
  FORM4.MaskEdit3.Clear;
  FORM4.ComboBox1.Text:='SELECT BLOOD GROUP';
  FORM4.DateTimePicker1.Date:=DATE;
END;
procedure TForm4.Button3Click(Sender: TObject);
begin
  FORM1.SHOW;
  FORM4.Hide;
end;

procedure TForm4.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  FORM1.Show;
  FORM4.Hide;
  PATIENTFORMUNUTEMIZLE();
  PATIENTHATAGOSTERICITEMIZLE();
  FORM4.ADOQuery1.Close;
  FORM4.ADOQuery2.Close;
  FORM4.ADOQuery3.Close;
end;

procedure TForm4.Button6Click(Sender: TObject);
  VAR
    T:WORD;
begin
  DATABASETARIH();
  IF (FORM4.Edit13.Text <> '') AND (FORM4.Edit14.Text <> '') AND
(FORM4.Edit1.Text <> '') AND (FORM4.Edit2.Text <> '') AND
(FORM4.Edit15.Text <> '') AND (FORM4.Memo5.Text <> '') AND
(DATETOSTR(FORM4.DateTimePicker1.Date) = DATETOSTR(DATE)) OR
(FORM4.Edit13.Text <> '') AND (FORM4.Edit14.Text <> '') AND
(FORM4.Edit1.Text <> '') AND (FORM4.Edit2.Text <> '') AND
(FORM4.Edit15.Text <> '') AND (FORM4.Memo5.Text <> '') AND
(FORM4.DateTimePicker1.Date < DATE) THEN
  BEGIN
//YUKARIDA <= YAPMAYIP < VE = YAPMAMIN NEDENİ DELPHİ ='İ >0LARAK KABUL
EDİYOR.ONDAN = OR < YAPTIMKİ İKİSİNİDE NORMAL OLARAK ALGILASIN
    FORM4.ADOQuery3.SQL.Text:='SELECT tcidno FROM patients where
tcidno='+#39+form4.Edit15.Text+#39;
    FORM4.ADOQuery3.Open;
    IF (FORM4.ADOQuery3.RecordCount = 0) and
(DATETOSTR(form4.DateTimePicker1.Date) <> DATETOSTR(date)) THEN
    BEGIN
      PATIENTKAYDET();
      PATIENTHATAGOSTERICITEMIZLE();
    END
    ELSE IF (FORM4.ADOQuery3.RecordCount = 0) and
(DATETOSTR(form4.DateTimePicker1.Date) = DATETOSTR(date)) THEN
    BEGIN
      T:=APPLICATION.MessageBox('ARE YOU SURE TO BE BIRTHDATE IS
TODAY?','ATTENTION',MB_YESNO+MB_ICONEXCLAMATION);
      IF T=ID_YES THEN
      BEGIN
        PATIENTKAYDET();
```

```
        PATIENTHATAGOSTERICITEMIZLE();
      END;
    END
    ELSE IF (FORM4.ADOQuery3.RecordCount <> 0) THEN
      APPLICATION.MessageBox('THE PATIENT HAS BEEN SAVED
BEFORE','CAUTION',MB_OK+MB_ICONEXCLAMATION);
  END
  ELSE IF (FORM4.Edit13.Text = '') OR (FORM4.Edit14.Text = '') OR
(FORM4.Edit1.Text = '') OR (FORM4.Edit2.Text = '') OR
(FORM4.Edit15.Text = '') OR (FORM4.Memo5.Text = '') THEN
  BEGIN
    APPLICATION.MessageBox('PLEASE FILL EMPTY PLACE WHICH MARKED WITH
RED ARROWS','FILL EMPTY PLACE',MB_OK+MB_ICONEXCLAMATION);
    PATIENTHATAGOSTERICI();
  END
  ELSE IF (FORM4.DateTimePicker1.Date > DATE) THEN
    APPLICATION.MessageBox('THE PATIENT BIRTDATE CAN NOT BE GREATER
THAN TODAY','CAUTION',MB_OK+MB_ICONEXCLAMATION);

  NORMALTARIH();
  PTNTDBG:=0;
end;

procedure TForm4.Button7Click(Sender: TObject);
begin
  IF (FORM4.Edit13.Text <> '') AND (FORM4.Edit14.Text <> '') AND
(FORM4.Edit1.Text <> '') AND (FORM4.Edit2.Text <> '') AND
(FORM4.Edit15.Text <> '') AND (FORM4.Memo5.Text <> '') AND
(DATETOSTR(FORM4.DateTimePicker1.Date) = DATETOSTR(DATE)) AND (PTNTDBG
= 1) OR (FORM4.Edit13.Text <> '') AND (FORM4.Edit14.Text <> '') AND
(FORM4.Edit1.Text <> '') AND (FORM4.Edit2.Text <> '') AND
(FORM4.Edit15.Text <> '') AND (FORM4.Memo5.Text <> '') AND
(FORM4.DateTimePicker1.Date < DATE) AND (PTNTDBG = 1) THEN
  BEGIN
//YUKARIDA <= YAPMAYIP < VE = YAPMAMIN NEDENİ DELPHİ ='İ >OLARAK KABUL
EDİYOR.ONDAN = OR < YAPTIMKİ İKİSİNİDE NORMAL OLARAK ALGILASIN
    FORM4.ADOQuery1.SQL.Text:='SELECT * FROM patients where
customerid='+#39+form4.Edit17.Text+#39;
    form4.ADOQuery1.Open;
    FORM4.ADOQuery1.Edit;
    COMPONENTDATABASE();
    FORM4.ADOQuery1.Post;
    SHOWMESSAGE('THE RECORD HAS BEEN UPDATED SUCCESFULLY');
    PATIENTHATAGOSTERICITEMIZLE();
    PATIENTSEARCH();
    PTNTDBG:=0;
  END
  ELSE IF PTNTDBG = 0 THEN
  BEGIN
    APPLICATION.MessageBox('PLEASE SELECT PATIENT TO EDIT INFORMATION
FROM THE PATIENT LIST','SELECT PATIENT',MB_OK+MB_ICONEXCLAMATION);
  END
  ELSE IF (FORM4.Edit13.Text = '') OR (FORM4.Edit14.Text = '') OR
(FORM4.Edit1.Text = '') OR (FORM4.Edit2.Text = '') OR
(FORM4.Edit15.Text = '') OR (FORM4.Memo5.Text = '') THEN
  BEGIN
    APPLICATION.MessageBox('PLEASE FILL EMPTY PLACE WHICH MARKED WITH
RED ARROWS','FILL EMPTY PLACE',MB_OK+MB_ICONEXCLAMATION);
    PATIENTHATAGOSTERICI();
  END
```

```
  ELSE IF (FORM4.DateTimePicker1.Date > DATE) THEN
    APPLICATION.MessageBox('THE PATIENT BIRTDATE CAN NOT BE GREATER
THAN TODAY','CAUTION',MB_OK+MB_ICONEXCLAMATION);
  NORMALTARIH();
end;

procedure TForm4.DBGrid1CellClick(Column: TColumn);
begin
  IF (FORM4.DBGrid1.FieldCount <> 0) AND (FORM4.DBGrid1.Fields[0].Text
<> '') AND (FORM4.DBGrid1.Fields[0].Text <> NULL) THEN
  BEGIN
    PTNTDBG:=1;
    form4.Edit17.Text:=form4.DBGrid1.Fields[0].Text;
    form4.Edit13.Text:=form4.DBGrid1.Fields[1].Text;
    form4.Edit14.Text:=form4.DBGrid1.Fields[2].Text;

form4.DateTimePicker1.Date:=strtodate(form4.DBGrid1.Fields[3].Text);
    form4.Edit1.Text:=form4.DBGrid1.Fields[12].Text;
    form4.Edit2.Text:=form4.DBGrid1.Fields[13].Text;
    form4.Edit15.Text:=form4.DBGrid1.Fields[4].Text;
    form4.Memo5.Text:=form4.DBGrid1.Fields[5].Text;
    form4.MaskEdit2.Text:=form4.DBGrid1.Fields[6].Text;
    form4.MaskEdit3.Text:=form4.DBGrid1.Fields[7].Text;
    form4.Edit16.Text:=form4.DBGrid1.Fields[8].Text;
    form4.Memo1.Text:=form4.DBGrid1.Fields[14].Text;
    form4.ComboBox1.Text:=form4.DBGrid1.Fields[11].Text;
    form4.Memo6.Text:=form4.DBGrid1.Fields[9].Text;
    form4.Memo9.Text:=form4.DBGrid1.Fields[10].Text;
  END
  ELSE IF (FORM4.DBGrid1.FieldCount = 0) THEN
    APPLICATION.MessageBox('DATASET CLOSED.YOU CAN NOT SELECT ANY
ITEM','ATTENTION !!!',MB_OK+MB_ICONEXCLAMATION)
   ELSE IF (FORM4.DBGrid1.Fields[0].Text = '') THEN
    APPLICATION.MessageBox('THERE IS NO DATA','ATTENTION
!!!',MB_OK+MB_ICONEXCLAMATION);
end;

procedure TForm4.Button2Click(Sender: TObject);
begin
  PATIENTFORMUNUTEMIZLE();
  PATIENTSEARCH();
  PTNTDBG:=0;
  PATIENTHATAGOSTERICITEMIZLE();
  NORMALTARIH();
end;

procedure TForm4.Button1Click(Sender: TObject);
VAR
  SIL:WORD;
begin
  IF PTNTDBG = 1 THEN
  BEGIN
    SIL:=APPLICATION.MessageBox('ARE YOU
SURE','CAUTION',MB_YESNO+MB_ICONQUESTION);
    IF SIL=IDYES THEN
    BEGIN
      FORM4.ADOQuery1.SQL.Text:='delete FROM patients WHERE
customerid='+#39+form4.Edit17.Text+#39;
      form4.ADOQuery1.ExecSQL;
      FORM4.Button2.Click;
```

```
      PATIENTSEARCH();
    END;
  END
  ELSE IF PTNTDBG = 0 THEN
    APPLICATION.MessageBox('PLEASE SELECT PATIENT TO EDIT INFORMATION
FROM THE PATIENT LIST','SELECT PATIENT',MB_OK+MB_ICONEXCLAMATION);
NORMALTARIH();
end;

procedure TForm4.DBGrid1KeyUp(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  FORM4.DBGrid1CellClick(NIL);
end;

procedure TForm4.Edit15KeyPress(Sender: TObject; var Key: Char);
begin
  //SADECE RAKAMLAR GİREBİLMEK İÇİN, BACKSPACE VE DELETE TUŞUNA
BASABİLMEK İÇİN
  IF (KEY<>'0') AND (KEY<>'1') AND (KEY<>'2') AND (KEY<>'3') AND
(KEY<>'4') AND (KEY<>'5') AND (KEY<>'6') AND (KEY<>'7') AND (KEY<>'8')
AND (KEY<>'9') AND (KEY<>#8) AND (KEY<>#127) THEN
    //EĞER BAŞKA BİR TUŞ BASILMIŞSA İPTAL ETMESİ İÇİN
    KEY:=#0;
end;

procedure TForm4.FormShow(Sender: TObject);
begin
  FORM4.DateTimePicker1.Date:=DATE;
  PATIENTSEARCH();
  PTNTDBG:=0;
end;

end.
```

5. UNIT5 CODES

```
unit Unit5;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, Grids, DBGrids, ComCtrls, StdCtrls, ExtCtrls, Menus, DB,
ADODB;

type
  TForm5 = class(TForm)
    Panel1: TPanel;
    Image1: TImage;
    Image2: TImage;
    Label2: TLabel;
    Image3: TImage;
    Label1: TLabel;
    GroupBox1: TGroupBox;
    PageControl1: TPageControl;
```

```
    TabSheet1: TTabSheet;
    TabSheet2: TTabSheet;
    DBGrid1: TDBGrid;
    Panel2: TPanel;
    GroupBox2: TGroupBox;
    Panel3: TPanel;
    Panel4: TPanel;
    DBGrid2: TDBGrid;
    Panel5: TPanel;
    GroupBox3: TGroupBox;
    Label3: TLabel;
    DateTimePicker1: TDateTimePicker;
    Button1: TButton;
    Button2: TButton;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    Label14: TLabel;
    Label15: TLabel;
    Label16: TLabel;
    Label17: TLabel;
    Label18: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Edit5: TEdit;
    Memo1: TMemo;
    Edit6: TEdit;
    Edit7: TEdit;
    Edit8: TEdit;
    Edit9: TEdit;
    Edit10: TEdit;
    Memo2: TMemo;
    Memo3: TMemo;
    Memo4: TMemo;
    Edit11: TEdit;
    Label19: TLabel;
    Label20: TLabel;
    Label21: TLabel;
    Label22: TLabel;
    Label23: TLabel;
    Label24: TLabel;
    Label25: TLabel;
    Label26: TLabel;
    Label27: TLabel;
    Label28: TLabel;
    Label29: TLabel;
    Label30: TLabel;
    Label31: TLabel;
    Label32: TLabel;
    Label33: TLabel;
    Memo5: TMemo;
```

```
    Edit12: TEdit;
    Edit13: TEdit;
    Edit14: TEdit;
    Edit15: TEdit;
    Edit16: TEdit;
    Memo6: TMemo;
    Edit17: TEdit;
    Edit18: TEdit;
    Edit19: TEdit;
    Edit20: TEdit;
    Edit21: TEdit;
    Memo7: TMemo;
    Edit22: TEdit;
    Memo8: TMemo;
    Shape1: TShape;
    Label34: TLabel;
    Label35: TLabel;
    Label36: TLabel;
    Label37: TLabel;
    Label38: TLabel;
    Label39: TLabel;
    Label40: TLabel;
    Label41: TLabel;
    Label42: TLabel;
    Label43: TLabel;
    Label44: TLabel;
    Label45: TLabel;
    Label46: TLabel;
    Label47: TLabel;
    MainMenu1: TMainMenu;
    OL1: TMenuItem;
    ADOQuery1: TADOQuery;
    ADOQuery2: TADOQuery;
    ADOQuery3: TADOQuery;
    ADOQuery4: TADOQuery;
    ADOQuery5: TADOQuery;
    DataSource1: TDataSource;
    DataSource2: TDataSource;
    DataSource3: TDataSource;
    DataSource4: TDataSource;
    DataSource5: TDataSource;
    Timer1: TTimer;
    Label48: TLabel;
    Label49: TLabel;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure Timer1Timer(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure DateTimePicker1Change(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure DBGrid1CellClick(Column: TColumn);
    procedure DBGrid2CellClick(Column: TColumn);
    procedure DBGrid1KeyUp(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure DBGrid2KeyUp(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure DBGrid1DblClick(Sender: TObject);
    procedure DBGrid2DblClick(Sender: TObject);
  private
    { Private declarations }
```

```
  public
    { Public declarations }
  end;

var
  Form5: TForm5;
  illyesterday,illtoday,illtomorrow,illcontroldate:integer;
  opyesterday,optoday,optomorrow,opcontroldate:integer;
implementation

uses Unit1, Unit2, Unit4, Unit3, Unit9;

{$R *.dfm}

PROCEDURE CONTROLFORMUNUTEMIZLE();
BEGIN
  FORM5.Edit1.Clear;
  FORM5.Edit2.Clear;
  FORM5.Edit3.Clear;
  FORM5.Edit4.Clear;
  FORM5.Edit5.Clear;
  FORM5.Memo1.Clear;
  FORM5.Edit6.Clear;
  FORM5.Edit7.Clear;
  FORM5.Edit8.Clear;
  FORM5.Edit9.Clear;
  FORM5.Edit10.Clear;
  FORM5.Memo2.Clear;
  FORM5.Memo3.Clear;
  FORM5.Edit11.Clear;
  FORM5.Memo4.Clear;
  FORM5.ADOQuery1.Close;
  {FORM5.GroupBox2.Visible:=FALSE;
  FORM5.DBGrid1.Visible:=FALSE;
  FORM5.Panel2.Visible:=FALSE;
  FORM5.Panel3.Visible:=FALSE;}

  FORM5.Edit16.Clear;
  FORM5.Edit15.Clear;
  FORM5.Edit14.Clear;
  FORM5.Edit13.Clear;
  FORM5.Edit12.Clear;
  FORM5.Memo5.Clear;
  FORM5.Edit21.Clear;
  FORM5.Edit20.Clear;
  FORM5.Edit19.Clear;
  FORM5.Edit18.Clear;
  FORM5.Edit17.Clear;
  FORM5.Memo6.Clear;
  FORM5.Memo7.Clear;
  FORM5.Edit22.Clear;
  FORM5.Memo8.Clear;
  FORM5.ADOQuery2.Close;
  {FORM5.GroupBox3.Visible:=FALSE;
  FORM5.DBGrid2.Visible:=FALSE;
  FORM5.DateTimePicker1.Date:=DATE;
  FORM5.Panel4.Visible:=FALSE;
  FORM5.Panel5.Visible:=FALSE;}
END;
```

```
PROCEDURE ANALIZ();
BEGIN
  DATABASETARIH();
  //BURADA ILLNESSES ANALİZİ YAPILIYOR
  //--> DÜN YAPILAN HASTALIK KAYITLARI SAYISI
  FORM5.ADOQuery5.SQL.Text:='SELECT COUNT(customerid) as illyesterday
FROM illnesses WHERE date='+#39+DATETOSTR(DATE-1)+#39;
  FORM5.ADOQuery5.Open;
  if (form5.ADOQuery5['illyesterday'] <> 0) and
(form5.ADOQuery5['illyesterday'] <> null) then
  BEGIN
    FORM5.Label40.Font.Color:=$00FF8000;
    form5.Label40.Caption:=form5.ADOQuery5['illyesterday'];
  END
  else
  BEGIN
    FORM5.Label40.Font.Color:=CLRED;
    form5.Label40.Caption:='-';
  END;

  //--> BUGÜN YAPILAN HASTALIK KAYITLARI SAYISI
  form5.ADOQuery5.SQL.Clear;
  FORM5.ADOQuery5.SQL.Text:='SELECT COUNT(customerid) as illtoday FROM
illnesses WHERE date='+#39+DATETOSTR(DATE)+#39;
  FORM5.ADOQuery5.Open;
  if (form5.ADOQuery5['illtoday'] <> 0) and
(form5.ADOQuery5['illtoday'] <> null) then
  BEGIN
    FORM5.Label41.Font.Color:=$00FF8000;
    form5.Label41.Caption:=form5.ADOQuery5['illtoday']
  END
  else
  BEGIN
    FORM5.Label41.Font.Color:=CLRED;
    form5.Label41.Caption:='-';
  END;

  //--> YARIN YAPILACAK HASTALIK CONTROLLERİ SAYISI
  form5.ADOQuery5.SQL.Clear;
  FORM5.ADOQuery5.SQL.Text:='SELECT COUNT(customerid) as illtomorrow
FROM illnesses WHERE controldate='+#39+DATETOSTR(DATE+1)+#39;
  FORM5.ADOQuery5.Open;
  if (form5.ADOQuery5['illtomorrow'] <> 0) and
(form5.ADOQuery5['illtomorrow'] <> null) then
  BEGIN
    FORM5.Label42.Font.Color:=$00FF8000;
    form5.Label42.Caption:=form5.ADOQuery5['illtomorrow']
  END
  else
  BEGIN
    FORM5.Label42.Font.Color:=CLRED;
    form5.Label42.Caption:='-';
  END;

  //--> BELİRTİLEN TARİHDE YAPILACAK HASTALIK KONTROLU SAYISI
  form5.ADOQuery5.SQL.Clear;
  FORM5.ADOQuery5.SQL.Text:='SELECT COUNT(customerid) as
illcontroldate FROM illnesses WHERE
date='+#39+datetostr(form5.DateTimePicker1.Date)+#39;
  FORM5.ADOQuery5.Open;
```

```
  if (form5.ADOQuery5['illcontroldate'] <> 0) and
(form5.ADOQuery5['illcontroldate'] <> null) then
   BEGIN
    FORM5.Label43.Font.Color:=$00FF8000;
    form5.Label43.Caption:=form5.ADOQuery5['illcontroldate']
  END
  else
  BEGIN
    FORM5.Label43.Font.Color:=CLRED;
    form5.Label43.Caption:='-';
  END;

//BURADA OPERATION ANALİZLERİ BAŞLIYOR
  //--> DÜN YAPILAN OPERASYON KAYITLARI SAYISI
  FORM5.ADOQuery5.SQL.Text:='SELECT COUNT(customerid) as opryesterday
FROM operations WHERE date='+#39+DATETOSTR(DATE-1)+#39;
  FORM5.ADOQuery5.Open;
  if (form5.ADOQuery5['opryesterday'] <> 0) and
(form5.ADOQuery5['opryesterday'] <> null) then
  BEGIN
    FORM5.Label44.Font.Color:=$00FF8000;
    form5.Label44.Caption:=form5.ADOQuery5['opryesterday'];
  END
  else
  BEGIN
    FORM5.Label44.Font.Color:=CLRED;
    form5.Label44.Caption:='-';
  END;

  //--> BUGÜN YAPILAN OPERASYON KAYITLARI SAYISI
  form5.ADOQuery5.SQL.Clear;
  FORM5.ADOQuery5.SQL.Text:='SELECT COUNT(customerid) as oprtoday FROM
operations WHERE date='+#39+DATETOSTR(DATE)+#39;
  FORM5.ADOQuery5.Open;
  if (form5.ADOQuery5['oprtoday'] <> 0) and
(form5.ADOQuery5['oprtoday'] <> null) then
  BEGIN
    FORM5.Label45.Font.Color:=$00FF8000;
    form5.Label45.Caption:=form5.ADOQuery5['oprtoday']
  END
  else
  BEGIN
    FORM5.Label45.Font.Color:=CLRED;
    form5.Label45.Caption:='-';
  END;

   //--> YARIN YAPILACAK OPERASYON CONTROLLERİ SAYISI
  form5.ADOQuery5.SQL.Clear;
  FORM5.ADOQuery5.SQL.Text:='SELECT COUNT(customerid) as oprtomorrow
FROM operations WHERE controldate='+#39+DATETOSTR(DATE+1)+#39;
  FORM5.ADOQuery5.Open;
  if (form5.ADOQuery5['oprtomorrow'] <> 0) and
(form5.ADOQuery5['oprtomorrow'] <> null) then
  BEGIN
    FORM5.Label46.Font.Color:=$00FF8000;
    form5.Label46.Caption:=form5.ADOQuery5['oprtomorrow']
  END
  else
  BEGIN
    FORM5.Label46.Font.Color:=CLRED;
```

```
    form5.Label46.Caption:='-';
  END;

  //--> BELİRTİLEN TARİHDE YAPILACAK OPERASYON KONTROLU SAYISI
  form5.ADOQuery5.SQL.Clear;
  FORM5.ADOQuery5.SQL.Text:='SELECT COUNT(customerid) as
oprcontroldate FROM illnesses WHERE
date='+#39+datetostr(form5.DateTimePicker1.Date)+#39;
  FORM5.ADOQuery5.Open;
  if (form5.ADOQuery5['oprcontroldate'] <> 0) and
(form5.ADOQuery5['oprcontroldate'] <> null) then
   BEGIN
    FORM5.Label47.Font.Color:=$00FF8000;
    form5.Label47.Caption:=form5.ADOQuery5['oprcontroldate']
  END
  else
  BEGIN
    FORM5.Label47.Font.Color:=CLRED;
    form5.Label47.Caption:='-';
  END;
END;

PROCEDURE ILLGRUPBOXDISABLE();
BEGIN
  FORM5.Edit1.Enabled:=FALSE;
  FORM5.Edit2.Enabled:=FALSE;
  FORM5.Edit3.Enabled:=FALSE;
  FORM5.Edit4.Enabled:=FALSE;
  FORM5.Edit5.Enabled:=FALSE;
  FORM5.Memo1.Enabled:=FALSE;
  FORM5.Edit6.Enabled:=FALSE;
  FORM5.Edit7.Enabled:=FALSE;
  FORM5.Edit8.Enabled:=FALSE;
  FORM5.Edit9.Enabled:=FALSE;
  FORM5.Edit10.Enabled:=FALSE;
  FORM5.Memo2.Enabled:=FALSE;
  FORM5.Memo3.Enabled:=FALSE;
  FORM5.Edit11.Enabled:=FALSE;
  FORM5.Memo4.Enabled:=FALSE;
  FORM5.Label4.Enabled:=FALSE;
  FORM5.Label5.Enabled:=FALSE;
  FORM5.Label6.Enabled:=FALSE;
  FORM5.Label7.Enabled:=FALSE;
  FORM5.Label8.Enabled:=FALSE;
  FORM5.Label9.Enabled:=FALSE;
  FORM5.Label10.Enabled:=FALSE;
  FORM5.Label11.Enabled:=FALSE;
  FORM5.Label12.Enabled:=FALSE;
  FORM5.Label13.Enabled:=FALSE;
  FORM5.Label14.Enabled:=FALSE;
  FORM5.Label15.Enabled:=FALSE;
  FORM5.Label16.Enabled:=FALSE;
  FORM5.Label17.Enabled:=FALSE;
  FORM5.Label18.Enabled:=FALSE;
END;

PROCEDURE ILLGRUPBOXENABLE();
BEGIN
  FORM5.Edit1.Enabled:=TRUE;
  FORM5.Edit2.Enabled:=TRUE;
```

```
  FORM5.Edit3.Enabled:=TRUE;
  FORM5.Edit4.Enabled:=TRUE;
  FORM5.Edit5.Enabled:=TRUE;
  FORM5.Memo1.Enabled:=TRUE;
  FORM5.Edit6.Enabled:=TRUE;
  FORM5.Edit7.Enabled:=TRUE;
  FORM5.Edit8.Enabled:=TRUE;
  FORM5.Edit9.Enabled:=TRUE;
  FORM5.Edit10.Enabled:=TRUE;
  FORM5.Memo2.Enabled:=TRUE;
  FORM5.Memo3.Enabled:=TRUE;
  FORM5.Edit11.Enabled:=TRUE;
  FORM5.Memo4.Enabled:=TRUE;
  FORM5.Label4.Enabled:=TRUE;
  FORM5.Label5.Enabled:=TRUE;
  FORM5.Label6.Enabled:=TRUE;
  FORM5.Label7.Enabled:=TRUE;
  FORM5.Label8.Enabled:=TRUE;
  FORM5.Label9.Enabled:=TRUE;
  FORM5.Label10.Enabled:=TRUE;
  FORM5.Label11.Enabled:=TRUE;
  FORM5.Label12.Enabled:=TRUE;
  FORM5.Label13.Enabled:=TRUE;
  FORM5.Label14.Enabled:=TRUE;
  FORM5.Label15.Enabled:=TRUE;
  FORM5.Label16.Enabled:=TRUE;
  FORM5.Label17.Enabled:=TRUE;
  FORM5.Label18.Enabled:=TRUE;
END;

PROCEDURE OPRGRUPBOXDISABLE();
BEGIN
  FORM5.Edit16.Enabled:=FALSE;
  FORM5.Edit15.Enabled:=FALSE;
  FORM5.Edit14.Enabled:=FALSE;
  FORM5.Edit13.Enabled:=FALSE;
  FORM5.Edit12.Enabled:=FALSE;
  FORM5.Memo5.Enabled:=FALSE;
  FORM5.Edit21.Enabled:=FALSE;
  FORM5.Edit20.Enabled:=FALSE;
  FORM5.Edit19.Enabled:=FALSE;
  FORM5.Edit18.Enabled:=FALSE;
  FORM5.Edit17.Enabled:=FALSE;
  FORM5.Memo6.Enabled:=FALSE;
  FORM5.Memo7.Enabled:=FALSE;
  FORM5.Edit22.Enabled:=FALSE;
  FORM5.Memo8.Enabled:=FALSE;
  FORM5.Label19.Enabled:=FALSE;
  FORM5.Label20.Enabled:=FALSE;
  FORM5.Label21.Enabled:=FALSE;
  FORM5.Label22.Enabled:=FALSE;
  FORM5.Label23.Enabled:=FALSE;
  FORM5.Label24.Enabled:=FALSE;
  FORM5.Label25.Enabled:=FALSE;
  FORM5.Label26.Enabled:=FALSE;
  FORM5.Label27.Enabled:=FALSE;
  FORM5.Label28.Enabled:=FALSE;
  FORM5.Label29.Enabled:=FALSE;
  FORM5.Label30.Enabled:=FALSE;
  FORM5.Label31.Enabled:=FALSE;
```

```
  FORM5.Label32.Enabled:=FALSE;
  FORM5.Label33.Enabled:=FALSE;
END;

PROCEDURE OPRGRUPBOXENABLE();
BEGIN
  FORM5.Edit16.Enabled:=TRUE;
  FORM5.Edit15.Enabled:=TRUE;
  FORM5.Edit14.Enabled:=TRUE;
  FORM5.Edit13.Enabled:=TRUE;
  FORM5.Edit12.Enabled:=TRUE;
  FORM5.Memo5.Enabled:=TRUE;
  FORM5.Edit21.Enabled:=TRUE;
  FORM5.Edit20.Enabled:=TRUE;
  FORM5.Edit19.Enabled:=TRUE;
  FORM5.Edit18.Enabled:=TRUE;
  FORM5.Edit17.Enabled:=TRUE;
  FORM5.Memo6.Enabled:=TRUE;
  FORM5.Memo7.Enabled:=TRUE;
  FORM5.Edit22.Enabled:=TRUE;
  FORM5.Memo8.Enabled:=TRUE;
  FORM5.Label19.Enabled:=TRUE;
  FORM5.Label20.Enabled:=TRUE;
  FORM5.Label21.Enabled:=TRUE;
  FORM5.Label22.Enabled:=TRUE;
  FORM5.Label23.Enabled:=TRUE;
  FORM5.Label24.Enabled:=TRUE;
  FORM5.Label25.Enabled:=TRUE;
  FORM5.Label26.Enabled:=TRUE;
  FORM5.Label27.Enabled:=TRUE;
  FORM5.Label28.Enabled:=TRUE;
  FORM5.Label29.Enabled:=TRUE;
  FORM5.Label30.Enabled:=TRUE;
  FORM5.Label31.Enabled:=TRUE;
  FORM5.Label32.Enabled:=TRUE;
  FORM5.Label33.Enabled:=TRUE;
END;

procedure TForm5.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  FORM1.SHOW;
  FORM5.Hide;
  FORM5.Timer1.Enabled:=FALSE;
  CONTROLFORMUNUTEMIZLE();
  FORM5.DBGrid1.Visible:=TRUE;
  FORM5.DBGrid2.Visible:=TRUE;
  FORM5.Label48.Visible:=FALSE;
  FORM5.Label49.Visible:=FALSE;
  FORM5.ADOQuery1.Close;
  FORM5.ADOQuery2.Close;
  FORM5.ADOQuery3.Close;
  FORM5.ADOQuery4.Close;
  FORM5.ADOQuery5.Close;
end;

procedure TForm5.Timer1Timer(Sender: TObject);
begin
ANALIZ();
end;
```

```
procedure TForm5.Button2Click(Sender: TObject);
begin
  form5.DateTimePicker1.Date:=date;
  //ZATEN DATETIMEPICKER1 (CONTROL DATE) DEĞİŞTİĞİNDE FORMU
TEMİZLİYOR.ONDAN DOLAYI BU KODLARI YAZMAK YERİNE
  //FORM5 DATETIMEPICKER1'IN OLAYINI (EVENTS) AYNEN AKTARIYORUM.
  //NIL DEMEMIN SEBEBİ; BOŞ OLARAK DİREKT O OLAYI ALMASI İÇİN. BİR
PARAMETRE VERMEZSEN ÇALIŞMAZ ZATEN
  form5.DateTimePicker1.OnChange(nil);
end;

procedure TForm5.DateTimePicker1Change(Sender: TObject);
begin
 NORMALTARIH();
 FORM5.Label37.Caption:=DATETOSTR(FORM5.DateTimePicker1.Date);
 ANALIZ();
 CONTROLFORMUNUTEMIZLE();
 ILLGRUPBOXENABLE;
 OPRGRUPBOXENABLE;
 FORM5.DBGrid1.Visible:=TRUE;
 FORM5.DBGrid2.Visible:=TRUE;
 FORM5.Label48.Visible:=FALSE;
 FORM5.Label49.Visible:=FALSE;
end;

procedure TForm5.FormShow(Sender: TObject);
begin
NORMALTARIH();
FORM5.DateTimePicker1.Date:=DATE;
FORM5.Label37.Caption:=DATETOSTR(FORM5.DateTimePicker1.Date);
FORM5.Timer1.Enabled:=TRUE;
ANALIZ();
end;

procedure TForm5.Button1Click(Sender: TObject);
begin
  DATABASETARIH();
  //ILLNESSES İÇİN KONTROL GÜNÜNÜ ARAŞTIRIYOR
  FORM5.ADOQuery1.SQL.Text:='SELECT * FROM illnesses WHERE
controldate='+#39+datetostr(form5.DateTimePicker1.Date)+#39;
  form5.ADOQuery1.Open;
  //EĞER BİR KONTROL BULUNMAZ İSE DBGRID'DEKİ İLK FIELD OLAN
customerid BOŞ VEYA NULL OLARAK GELECEK
  //İŞTE BÖYLE BİR DURUM OLURSA DBGRID'İN GİZLENMESİNİ, "KAYIT YOK"
YAZISININ ÇIKMASINI VE PATIENT
  //BİLGİLERİNİ GÖSTEREN KISIMLARIN DISABLE OLMASINI İSTİYORUZ
  IF (FORM5.DBGrid1.Fields[0].Text = '') OR
(FORM5.DBGrid1.Fields[0].Text = NULL) THEN
  BEGIN
    FORM5.Label48.Visible:=TRUE;
    FORM5.DBGrid1.Visible:=FALSE;
    ILLGRUPBOXDISABLE();
  END
  ELSE IF (FORM5.DBGrid1.Fields[0].Text <> '') AND
(FORM5.DBGrid1.Fields[0].Text <> NULL) THEN
  BEGIN
    FORM5.Label48.Visible:=FALSE;
    FORM5.DBGrid1.Visible:=TRUE;
    ILLGRUPBOXENABLE();
  END;
```

```
  //OPERATIONS İÇİN KONTROL GÜNÜNÜ ARAŞTIRIYOR
  FORM5.ADOQuery2.SQL.Text:='SELECT * FROM operations WHERE
controldate='+#39+datetostr(form5.DateTimePicker1.Date)+#39;
  form5.ADOQuery2.Open;
  //EĞER BİR KONTROL BULUNMAZ İSE DBGRID'DEKİ İLK FIELD OLAN
customerid BOŞ VEYA NULL OLARAK GELECEK
  //İŞTE BÖYLE BİR DURUM OLURSA DBGRID'İN GİZLENMESİNİ, "KAYIT YOK"
YAZISININ ÇIKMASINI VE PATIENT
  //BİLGİLERİNİ GÖSTEREN KISIMLARIN DISABLE OLMASINI İSTİYORUZ
  IF (FORM5.DBGrid2.Fields[0].Text = '') OR
(FORM5.DBGrid2.Fields[0].Text = NULL) THEN
  BEGIN
    FORM5.Label49.Visible:=TRUE;
    FORM5.DBGrid2.Visible:=FALSE;
    OPRGRUPBOXDISABLE();
  END
  ELSE IF (FORM5.DBGrid2.Fields[0].Text <> '') AND
(FORM5.DBGrid2.Fields[0].Text <> NULL) THEN
  BEGIN
    FORM5.Label49.Visible:=FALSE;
    FORM5.DBGrid2.Visible:=TRUE;
    OPRGRUPBOXENABLE();
  END;
  NORMALTARIH();
end;

procedure TForm5.DBGrid1CellClick(Column: TColumn);
begin
  IF (FORM5.DBGrid1.FieldCount <> 0) AND (FORM5.DBGrid1.Fields[0].Text
<> '') AND (FORM5.DBGrid1.Fields[0].Text <> NULL) THEN
  BEGIN
    form5.ADOQuery3.SQL.Text:='SELECT * FROM patients WHERE
customerid='+#39+form5.DBGrid1.Fields[0].Text+#39;
    form5.ADOQuery3.Open;
    if (form5.ADOQuery3.RecordCount <> 0) and
(form5.ADOQuery3['customerid'] <> null) then
    begin
      form5.Edit1.Text:=form5.ADOQuery3['customerid'];
      form5.Edit2.Text:=form5.ADOQuery3['name'];
      form5.Edit3.Text:=form5.ADOQuery3['surname'];
      form5.Edit4.Text:=form5.ADOQuery3['birthdate'];
      form5.Edit5.Text:=form5.ADOQuery3['tcidno'];
      form5.Memo1.Text:=form5.ADOQuery3['adress'];
      form5.Edit6.Text:=form5.ADOQuery3['telno'];
      form5.Edit7.Text:=form5.ADOQuery3['mobileno'];
      form5.Edit8.Text:=form5.ADOQuery3['email'];
      form5.Edit9.Text:=form5.ADOQuery3['mothername'];
      form5.Edit10.Text:=form5.ADOQuery3['fathername'];
      form5.Memo2.Text:=form5.ADOQuery3['cronological_illnesses'];
      form5.Memo3.Text:=form5.ADOQuery3['cronological_medicine'];
      form5.Edit11.Text:=form5.ADOQuery3['bloodgroup'];
      form5.Memo4.Text:=form5.ADOQuery3['note'];
    end;
  END;
end;

procedure TForm5.DBGrid2CellClick(Column: TColumn);
begin
IF (FORM5.DBGrid2.FieldCount <> 0) AND (FORM5.DBGrid2.Fields[0].Text
<> '') AND (FORM5.DBGrid2.Fields[0].Text <> NULL) THEN
```

```
  BEGIN
    form5.ADOQuery4.SQL.Text:='SELECT * FROM patients WHERE
customerid='+#39+form5.DBGrid2.Fields[0].Text+#39;
    form5.ADOQuery4.Open;
    if (form5.ADOQuery4.RecordCount <> 0) and
(form5.ADOQuery4['customerid'] <> null) then
    begin
      form5.Edit16.Text:=form5.ADOQuery4['customerid'];
      form5.Edit15.Text:=form5.ADOQuery4['name'];
      form5.Edit14.Text:=form5.ADOQuery4['surname'];
      form5.Edit13.Text:=form5.ADOQuery4['birthdate'];
      form5.Edit12.Text:=form5.ADOQuery4['tcidno'];
      form5.Memo5.Text:=form5.ADOQuery4['adress'];
      form5.Edit21.Text:=form5.ADOQuery4['telno'];
      form5.Edit20.Text:=form5.ADOQuery4['mobileno'];
      form5.Edit19.Text:=form5.ADOQuery4['email'];
      form5.Edit18.Text:=form5.ADOQuery4['mothername'];
      form5.Edit17.Text:=form5.ADOQuery4['fathername'];
      form5.Memo6.Text:=form5.ADOQuery4['cronological_illnesses'];
      form5.Memo7.Text:=form5.ADOQuery4['cronological_medicine'];
      form5.Edit22.Text:=form5.ADOQuery4['bloodgroup'];
      form5.Memo8.Text:=form5.ADOQuery4['note'];
    end;
 END;
end;

procedure TForm5.DBGrid1KeyUp(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
FORM5.DBGrid1CellClick(NIL);
end;

procedure TForm5.DBGrid2KeyUp(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
FORM5.DBGrid2CellClick(NIL);
end;

procedure TForm5.DBGrid1DblClick(Sender: TObject);
begin
  IF (FORM5.DBGrid1.FieldCount <> 0) AND (FORM5.DBGrid1.Fields[0].Text
<> '') AND (FORM5.DBGrid1.Fields[0].Text <> NULL) THEN
  BEGIN
    FORM9.Edit1.Text:=form5.DBGrid1.Fields[0].Text;
    form9.Show;
  END;
end;

procedure TForm5.DBGrid2DblClick(Sender: TObject);
begin
IF (FORM5.DBGrid2.FieldCount <> 0) AND (FORM5.DBGrid2.Fields[0].Text
<> '') AND (FORM5.DBGrid2.Fields[0].Text <> NULL) THEN
  BEGIN
    FORM9.Edit1.Text:=form5.DBGrid2.Fields[0].Text;
    form9.Show;
END;
end;

end.
```

6. UNIT6 CODES

```
unit Unit6;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, ComCtrls, StdCtrls, ExtCtrls, Grids, DBGrids, Menus, DB,
ADODB;

type
  TForm6 = class(TForm)
    PageControl1: TPageControl;
    TabSheet1: TTabSheet;
    TabSheet2: TTabSheet;
    TabSheet3: TTabSheet;
    TabSheet4: TTabSheet;
    TabSheet5: TTabSheet;
    TabSheet6: TTabSheet;
    TabSheet7: TTabSheet;
    TabSheet8: TTabSheet;
    TabSheet9: TTabSheet;
    TabSheet10: TTabSheet;
    TabSheet11: TTabSheet;
    TabSheet12: TTabSheet;
    TabSheet13: TTabSheet;
    TabSheet14: TTabSheet;
    GroupBox1: TGroupBox;
    Label1: TLabel;
    Memo1: TMemo;
    GroupBox2: TGroupBox;
    Button1: TButton;
    Button2: TButton;
    GroupBox3: TGroupBox;
    GroupBox4: TGroupBox;
    Label2: TLabel;
    ComboBox1: TComboBox;
    Button3: TButton;
    Button4: TButton;
    GroupBox5: TGroupBox;
    GroupBox6: TGroupBox;
    Button5: TButton;
    Button6: TButton;
    Memo2: TMemo;
    Label3: TLabel;
    GroupBox7: TGroupBox;
    GroupBox8: TGroupBox;
    Edit1: TEdit;
    Label4: TLabel;
    Button7: TButton;
    Button8: TButton;
    GroupBox9: TGroupBox;
    GroupBox10: TGroupBox;
    Label5: TLabel;
    Edit2: TEdit;
    Button9: TButton;
```

```
    Button10: TButton;
    GroupBox11: TGroupBox;
    GroupBox12: TGroupBox;
    Edit3: TEdit;
    Label6: TLabel;
    Button11: TButton;
    Button12: TButton;
    GroupBox13: TGroupBox;
    GroupBox14: TGroupBox;
    Button13: TButton;
    Button14: TButton;
    Label7: TLabel;
    DateTimePicker1: TDateTimePicker;
    GroupBox15: TGroupBox;
    GroupBox16: TGroupBox;
    DateTimePicker2: TDateTimePicker;
    Button15: TButton;
    Button16: TButton;
    Label8: TLabel;
    GroupBox17: TGroupBox;
    GroupBox18: TGroupBox;
    Button17: TButton;
    Button18: TButton;
    Label9: TLabel;
    Edit4: TEdit;
    GroupBox19: TGroupBox;
    GroupBox20: TGroupBox;
    Button19: TButton;
    Button20: TButton;
    Edit5: TEdit;
    GroupBox21: TGroupBox;
    GroupBox22: TGroupBox;
    Button21: TButton;
    Button22: TButton;
    DateTimePicker3: TDateTimePicker;
    Label11: TLabel;
    GroupBox23: TGroupBox;
    GroupBox24: TGroupBox;
    Button23: TButton;
    Button24: TButton;
    Edit6: TEdit;
    Label12: TLabel;
    GroupBox25: TGroupBox;
    GroupBox26: TGroupBox;
    Edit7: TEdit;
    Button25: TButton;
    Button26: TButton;
    Label13: TLabel;
    GroupBox27: TGroupBox;
    GroupBox28: TGroupBox;
    Button27: TButton;
    Button28: TButton;
    Edit8: TEdit;
    Label14: TLabel;
    GroupBox29: TGroupBox;
    Panel1: TPanel;
    DBGrid1: TDBGrid;
    Edit9: TEdit;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
```

```
MainMenu1: TMainMenu;
OL1: TMenuItem;
ADOQuery1: TADOQuery;
DataSource1: TDataSource;
Button29: TButton;
Button30: TButton;
Button31: TButton;
GroupBox30: TGroupBox;
RadioButton7: TRadioButton;
RadioButton8: TRadioButton;
RadioButton6: TRadioButton;
RadioButton5: TRadioButton;
RadioButton4: TRadioButton;
RadioButton3: TRadioButton;
DateTimePicker4: TDateTimePicker;
Label10: TLabel;
GroupBox31: TGroupBox;
RadioButton9: TRadioButton;
RadioButton10: TRadioButton;
RadioButton11: TRadioButton;
RadioButton12: TRadioButton;
RadioButton13: TRadioButton;
RadioButton14: TRadioButton;
Button32: TButton;
Button33: TButton;
Button34: TButton;
Button35: TButton;
Button36: TButton;
Button37: TButton;
GroupBox32: TGroupBox;
RadioButton15: TRadioButton;
RadioButton16: TRadioButton;
RadioButton17: TRadioButton;
RadioButton18: TRadioButton;
RadioButton19: TRadioButton;
RadioButton20: TRadioButton;
Label15: TLabel;
Label16: TLabel;
DateTimePicker5: TDateTimePicker;
DateTimePicker6: TDateTimePicker;
Button38: TButton;
RadioButton21: TRadioButton;
Label17: TLabel;
RadioButton22: TRadioButton;
RadioButton23: TRadioButton;
Button39: TButton;
Button40: TButton;
Button41: TButton;
Button42: TButton;
Button43: TButton;
Button44: TButton;
Button45: TButton;
Button46: TButton;
Button47: TButton;
Button48: TButton;
Button49: TButton;
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure Button13Click(Sender: TObject);
procedure Button14Click(Sender: TObject);
procedure Button11Click(Sender: TObject);
```

```
procedure Button12Click(Sender: TObject);
procedure Button10Click(Sender: TObject);
procedure Button8Click(Sender: TObject);
procedure Button6Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button28Click(Sender: TObject);
procedure Button26Click(Sender: TObject);
procedure Button24Click(Sender: TObject);
procedure Button22Click(Sender: TObject);
procedure Button20Click(Sender: TObject);
procedure Button18Click(Sender: TObject);
procedure Button16Click(Sender: TObject);
procedure Button4Click(Sender: TObject);
procedure Button9Click(Sender: TObject);
procedure Button7Click(Sender: TObject);
procedure Button5Click(Sender: TObject);
procedure Button3Click(Sender: TObject);
procedure Button1Click(Sender: TObject);
procedure Button27Click(Sender: TObject);
procedure Button25Click(Sender: TObject);
procedure Button23Click(Sender: TObject);
procedure Button21Click(Sender: TObject);
procedure Button19Click(Sender: TObject);
procedure Button17Click(Sender: TObject);
procedure Button15Click(Sender: TObject);
procedure Edit6KeyPress(Sender: TObject; var Key: Char);
procedure Edit9KeyPress(Sender: TObject; var Key: Char);
procedure Edit5KeyPress(Sender: TObject; var Key: Char);
procedure Edit1KeyPress(Sender: TObject; var Key: Char);
procedure Button29Click(Sender: TObject);
procedure RadioButton20Click(Sender: TObject);
procedure RadioButton15Click(Sender: TObject);
procedure RadioButton16Click(Sender: TObject);
procedure RadioButton17Click(Sender: TObject);
procedure RadioButton18Click(Sender: TObject);
procedure RadioButton19Click(Sender: TObject);
procedure RadioButton7Click(Sender: TObject);
procedure RadioButton3Click(Sender: TObject);
procedure RadioButton4Click(Sender: TObject);
procedure RadioButton5Click(Sender: TObject);
procedure RadioButton6Click(Sender: TObject);
procedure RadioButton8Click(Sender: TObject);
procedure RadioButton14Click(Sender: TObject);
procedure RadioButton9Click(Sender: TObject);
procedure RadioButton10Click(Sender: TObject);
procedure RadioButton11Click(Sender: TObject);
procedure RadioButton12Click(Sender: TObject);
procedure RadioButton13Click(Sender: TObject);
procedure RadioButton21Click(Sender: TObject);
procedure RadioButton22Click(Sender: TObject);
procedure RadioButton23Click(Sender: TObject);
procedure Button30Click(Sender: TObject);
procedure Button31Click(Sender: TObject);
procedure Button39Click(Sender: TObject);
procedure Button40Click(Sender: TObject);
procedure Button47Click(Sender: TObject);
procedure Button49Click(Sender: TObject);
procedure Button38Click(Sender: TObject);
procedure Button32Click(Sender: TObject);
procedure Button48Click(Sender: TObject);
```

```
    procedure Button33Click(Sender: TObject);
    procedure Button42Click(Sender: TObject);
    procedure Button34Click(Sender: TObject);
    procedure Button41Click(Sender: TObject);
    procedure Button35Click(Sender: TObject);
    procedure Button43Click(Sender: TObject);
    procedure Button36Click(Sender: TObject);
    procedure Button44Click(Sender: TObject);
    procedure Button45Click(Sender: TObject);
    procedure Button37Click(Sender: TObject);
    procedure Button46Click(Sender: TObject);
    procedure PageControl1Change(Sender: TObject);
    procedure DBGrid1DblClick(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form6: TForm6;

implementation

uses Unit1, Unit4, Unit2, Unit9;

{$R *.dfm}

PROCEDURE ADODURUMU();
BEGIN
  IF FORM6.ADOQuery1.RecordCount <> 0 THEN
  BEGIN
    FORM6.DBGrid1.Visible:=TRUE;
    FORM6.Label17.Visible:=FALSE;
  END
  ELSE
  IF FORM6.ADOQuery1.RecordCount = 0 THEN
  BEGIN
    FORM6.DBGrid1.Visible:=FALSE;
    FORM6.Label17.Visible:=TRUE;
  END;
END;

PROCEDURE SEARCHFORMUNUTEMIZLE();
BEGIN
  FORM6.RadioButton15.Checked:=FALSE;
  FORM6.RadioButton16.Checked:=FALSE;
  FORM6.RadioButton17.Checked:=FALSE;
  FORM6.RadioButton18.Checked:=FALSE;
  FORM6.RadioButton19.Checked:=FALSE;
  FORM6.RadioButton20.Checked:=FALSE;
  FORM6.RadioButton21.Checked:=FALSE;
  FORM6.DateTimePicker1.Date:=DATE;
  FORM6.DateTimePicker5.Date:=DATE;
  FORM6.RadioButton15Click(NIL);
  form6.Edit3.Clear;
  form6.Edit2.Clear;
  form6.Edit1.Clear;
  form6.Memo2.Clear;
```

```
  form6.ComboBox1.Text:='SELECT BLOOD GROUP';
  form6.Memo1.Clear;
  form6.Edit8.Clear;
  form6.Edit7.Clear;
  form6.Edit6.Clear;
  FORM6.RadioButton3.Checked:=FALSE;
  FORM6.RadioButton4.Checked:=FALSE;
  FORM6.RadioButton5.Checked:=FALSE;
  FORM6.RadioButton6.Checked:=FALSE;
  FORM6.RadioButton7.Checked:=FALSE;
  FORM6.RadioButton8.Checked:=FALSE;
  FORM6.RadioButton22.Checked:=FALSE;
  FORM6.DateTimePicker3.Date:=DATE;
  FORM6.DateTimePicker4.Date:=DATE;
  form6.RadioButton3Click(NIL);
  form6.RadioButton1.Checked:=false;
  form6.RadioButton2.Checked:=false;
  form6.Edit5.Clear;
  form6.Edit9.Clear;
  form6.Edit4.Clear;
  FORM6.RadioButton9.Checked:=FALSE;
  FORM6.RadioButton10.Checked:=FALSE;
  FORM6.RadioButton11.Checked:=FALSE;
  FORM6.RadioButton12.Checked:=FALSE;
  FORM6.RadioButton13.Checked:=FALSE;
  FORM6.RadioButton14.Checked:=FALSE;
  FORM6.RadioButton23.Checked:=FALSE;
  FORM6.DateTimePicker2.Date:=DATE;
  FORM6.DateTimePicker6.Date:=DATE;
  form6.RadioButton9Click(NIL);
  FORM6.DBGrid1.Visible:=TRUE;
  FORM6.Label17.Visible:=FALSE;
  FORM6.ADOQuery1.Close;
END;

procedure TForm6.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  FORM1.SHOW;
  FORM6.Hide;
  SEARCHFORMUNUTEMIZLE();
end;

procedure TForm6.Button13Click(Sender: TObject);
begin
  DATABASETARIH();
  IF FORM6.RadioButton15.Checked=TRUE THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT * FROM patients WHERE birthdate
< '+#39+datetostr(form6.DateTimePicker1.Date)+#39;
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE IF FORM6.RadioButton16.Checked=TRUE THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT * FROM patients WHERE birthdate
<= '+#39+datetostr(form6.DateTimePicker1.Date)+#39;
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE IF FORM6.RadioButton17.Checked=TRUE THEN
```

```
   BEGIN
     FORM6.ADOQuery1.SQL.Text:='SELECT * FROM patients WHERE birthdate
> '+#39+datetostr(form6.DateTimePicker1.Date)+#39;
     form6.ADOQuery1.Open;
     ADODURUMU();
   END
   ELSE IF FORM6.RadioButton18.Checked=TRUE THEN
   BEGIN
     FORM6.ADOQuery1.SQL.Text:='SELECT * FROM patients WHERE birthdate
>= '+#39+datetostr(form6.DateTimePicker1.Date)+#39;
     form6.ADOQuery1.Open;
     ADODURUMU();
   END
   ELSE IF FORM6.RadioButton19.Checked=TRUE THEN
   BEGIN
     FORM6.ADOQuery1.SQL.Text:='SELECT * FROM patients WHERE birthdate
<> '+#39+datetostr(form6.DateTimePicker1.Date)+#39;
     form6.ADOQuery1.Open;
     ADODURUMU();
   END
   ELSE IF FORM6.RadioButton20.Checked=TRUE THEN
   BEGIN
     FORM6.ADOQuery1.SQL.Text:='SELECT * FROM patients WHERE birthdate
between '+#39+datetostr(form6.DateTimePicker1.Date)+#39+' and
'+#39+datetostr(form6.DateTimePicker5.Date)+#39;
     form6.ADOQuery1.Open;
     ADODURUMU();
   END
   ELSE IF FORM6.RadioButton21.Checked=TRUE THEN
   BEGIN
     FORM6.ADOQuery1.SQL.Text:='SELECT * FROM patients WHERE birthdate
= '+#39+datetostr(form6.DateTimePicker1.Date)+#39;
     form6.ADOQuery1.Open;
     ADODURUMU();
   END
   ELSE IF (FORM6.RadioButton15.Checked=FALSE) AND
(FORM6.RadioButton16.Checked=FALSE) AND
(FORM6.RadioButton17.Checked=FALSE) AND
(FORM6.RadioButton18.Checked=FALSE) AND
(FORM6.RadioButton19.Checked=FALSE) AND
(FORM6.RadioButton20.Checked=FALSE) AND
(FORM6.RadioButton21.Checked=FALSE) THEN
     APPLICATION.MessageBox('PLEASE SELECT ANY DATE
CRITERIA','CAUTION',MB_OK+MB_ICONEXCLAMATION);
   NORMALTARIH();
end;

procedure TForm6.Button14Click(Sender: TObject);
begin
   FORM6.RadioButton15.Checked:=FALSE;
   FORM6.RadioButton16.Checked:=FALSE;
   FORM6.RadioButton17.Checked:=FALSE;
   FORM6.RadioButton18.Checked:=FALSE;
   FORM6.RadioButton19.Checked:=FALSE;
   FORM6.RadioButton20.Checked:=FALSE;
   FORM6.RadioButton21.Checked:=FALSE;
   FORM6.DateTimePicker1.Date:=DATE;
   FORM6.DateTimePicker5.Date:=DATE;
   FORM6.RadioButton15Click(NIL);
   FORM6.DBGrid1.Visible:=TRUE;
```

```
  FORM6.Label17.Visible:=FALSE;
  FORM6.ADOQuery1.Close;
end;

procedure TForm6.Button11Click(Sender: TObject);
begin
  IF FORM6.Edit3.Text <> '' THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT
customerid,name,surname,mothername,birthdate,tcidno,adress,telno,mobil
eno,email,cronological_illnesses,cronological_medicine,bloodgroup,fath
ername,note FROM patients WHERE mothername='+#39+form6.Edit3.Text+#39;
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE
    APPLICATION.MessageBox('PLEASE ENTER MOTHER
NAME','ATTENTION',MB_OK+MB_ICONEXCLAMATION);
end;

procedure TForm6.Button12Click(Sender: TObject);
begin
  form6.Edit3.Clear;
  FORM6.DBGrid1.Visible:=TRUE;
  FORM6.Label17.Visible:=FALSE;
  FORM6.ADOQuery1.Close;
end;

procedure TForm6.Button10Click(Sender: TObject);
begin
  form6.Edit2.Clear;
  FORM6.DBGrid1.Visible:=TRUE;
  FORM6.Label17.Visible:=FALSE;
  FORM6.ADOQuery1.Close;
end;

procedure TForm6.Button8Click(Sender: TObject);
begin
  form6.Edit1.Clear;
  FORM6.DBGrid1.Visible:=TRUE;
  FORM6.Label17.Visible:=FALSE;
  FORM6.ADOQuery1.Close;
end;

procedure TForm6.Button6Click(Sender: TObject);
begin
  form6.Memo2.Clear;
  FORM6.DBGrid1.Visible:=TRUE;
  FORM6.Label17.Visible:=FALSE;
  FORM6.ADOQuery1.Close;
end;

procedure TForm6.Button2Click(Sender: TObject);
begin
  form6.Memo1.Clear;
  FORM6.DBGrid1.Visible:=TRUE;
  FORM6.Label17.Visible:=FALSE;
  FORM6.ADOQuery1.Close;
end;
```

```
procedure TForm6.Button28Click(Sender: TObject);
begin
  form6.Edit8.Clear;
  FORM6.DBGrid1.Visible:=TRUE;
  FORM6.Label17.Visible:=FALSE;
  FORM6.ADOQuery1.Close;
end;

procedure TForm6.Button26Click(Sender: TObject);
begin
  form6.Edit7.Clear;
  FORM6.DBGrid1.Visible:=TRUE;
  FORM6.Label17.Visible:=FALSE;
  FORM6.ADOQuery1.Close;
end;

procedure TForm6.Button24Click(Sender: TObject);
begin
  form6.Edit6.Clear;
  FORM6.DBGrid1.Visible:=TRUE;
  FORM6.Label17.Visible:=FALSE;
  FORM6.ADOQuery1.Close;
end;

procedure TForm6.Button22Click(Sender: TObject);
begin
  FORM6.RadioButton3.Checked:=FALSE;
  FORM6.RadioButton4.Checked:=FALSE;
  FORM6.RadioButton5.Checked:=FALSE;
  FORM6.RadioButton6.Checked:=FALSE;
  FORM6.RadioButton7.Checked:=FALSE;
  FORM6.RadioButton8.Checked:=FALSE;
  FORM6.RadioButton22.Checked:=FALSE;
  FORM6.DateTimePicker3.Date:=DATE;
  FORM6.DateTimePicker4.Date:=DATE;
  form6.RadioButton3Click(NIL);
  FORM6.DBGrid1.Visible:=TRUE;
  FORM6.Label17.Visible:=FALSE;
  FORM6.ADOQuery1.Close;
end;

procedure TForm6.Button20Click(Sender: TObject);
begin
  form6.RadioButton1.Checked:=false;
  form6.RadioButton2.Checked:=false;
  form6.Edit5.Clear;
  form6.Edit9.Clear;
  FORM6.DBGrid1.Visible:=TRUE;
  FORM6.Label17.Visible:=FALSE;
  form6.ADOQuery1.Close;
end;

procedure TForm6.Button18Click(Sender: TObject);
begin
  form6.Edit4.Clear;
  FORM6.DBGrid1.Visible:=TRUE;
  FORM6.Label17.Visible:=FALSE;
  form6.ADOQuery1.Close;
end;
```

```
procedure TForm6.Button16Click(Sender: TObject);
begin
  FORM6.RadioButton9.Checked:=FALSE;
  FORM6.RadioButton10.Checked:=FALSE;
  FORM6.RadioButton11.Checked:=FALSE;
  FORM6.RadioButton12.Checked:=FALSE;
  FORM6.RadioButton13.Checked:=FALSE;
  FORM6.RadioButton14.Checked:=FALSE;
  FORM6.RadioButton23.Checked:=FALSE;
  FORM6.DateTimePicker2.Date:=DATE;
  FORM6.DateTimePicker6.Date:=DATE;
  form6.RadioButton9Click(NIL);
  FORM6.DBGrid1.Visible:=TRUE;
  FORM6.Label17.Visible:=FALSE;
  FORM6.ADOQuery1.Close;
end;

procedure TForm6.Button4Click(Sender: TObject);
begin
  form6.ComboBox1.Text:='SELECT BLOOD GROUP';
  FORM6.DBGrid1.Visible:=TRUE;
  FORM6.Label17.Visible:=FALSE;
  form6.ADOQuery1.Close;
end;

procedure TForm6.Button9Click(Sender: TObject);
begin
  IF FORM6.Edit2.Text <> '' THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT
customerid,name,surname,fathername,birthdate,tcidno,adress,telno,mobil
eno,email,cronological_illnesses,cronological_medicine,bloodgroup,moth
ername,note FROM patients WHERE fathername='+#39+form6.Edit2.Text+#39;
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE
    APPLICATION.MessageBox('PLEASE ENTER FATHER
NAME','ATTENTION',MB_OK+MB_ICONEXCLAMATION);
end;

procedure TForm6.Button7Click(Sender: TObject);
begin
  IF FORM6.Edit1.Text <> '' THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT * FROM patients WHERE
tcidno='+#39+form6.Edit1.Text+#39;
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE
    APPLICATION.MessageBox('PLEASE ENTER TC ID
NO','ATTENTION',MB_OK+MB_ICONEXCLAMATION);
end;

procedure TForm6.Button5Click(Sender: TObject);
begin
  IF FORM6.Memo2.Text <> '' THEN
  BEGIN
```

```
    FORM6.ADOQuery1.SQL.Text:='SELECT * FROM patients WHERE
adress='+#39+form6.Memo2.Text+#39;
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE
    APPLICATION.MessageBox('PLEASE ENTER
ADDRESS','ATTENTION',MB_OK+MB_ICONEXCLAMATION);
end;

procedure TForm6.Button3Click(Sender: TObject);
begin
  IF form6.ComboBox1.Text <> 'SELECT BLOOD GROUP' THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT
customerid,name,surname,bloodgroup,birthdate,tcidno,adress,telno,mobil
eno,email,cronological_illnesses,cronological_medicine,fathername,moth
ername,note FROM patients WHERE
bloodgroup='+#39+form6.ComboBox1.Text+#39;
    form6.ADOQuery1.Open;
    ADODURUMU();
    END
  ELSE
    APPLICATION.MessageBox('PLEASE ENTER BLOOD
GROUP','ATTENTION',MB_OK+MB_ICONEXCLAMATION);
end;

procedure TForm6.Button1Click(Sender: TObject);
begin
  IF FORM6.Memo1.Text <> '' THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT
customerid,name,surname,cronological_illnesses,birthdate,tcidno,adress
,telno,mobileno,email,bloodgroup,cronological_medicine,fathername,moth
ername,note FROM patients WHERE
cronological_illnesses='+#39+form6.Memo1.Text+#39;
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE
    APPLICATION.MessageBox('PLEASE ENTER CRONOLOGICAL
ILLNESSES','ATTENTION',MB_OK+MB_ICONEXCLAMATION);
end;

procedure TForm6.Button27Click(Sender: TObject);
begin
  IF FORM6.Edit8.Text <> '' THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT * FROM patients WHERE
name='+#39+form6.Edit8.Text+#39;
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE
    APPLICATION.MessageBox('PLEASE ENTER
NAME','ATTENTION',MB_OK+MB_ICONEXCLAMATION);
end;

procedure TForm6.Button25Click(Sender: TObject);
begin
```

```
  IF FORM6.Edit7.Text <> '' THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT * FROM patients WHERE
surname='+#39+form6.Edit7.Text+#39;
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE
    APPLICATION.MessageBox('PLEASE ENTER
SURNAME','ATTENTION',MB_OK+MB_ICONEXCLAMATION);
end;

procedure TForm6.Button23Click(Sender: TObject);
begin
  IF FORM6.Edit6.Text <> '' THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT * FROM patients WHERE
customerid='+#39+form6.Edit6.Text+#39;
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE
    APPLICATION.MessageBox('PLEASE ENTER PATIENT
ID','ATTENTION',MB_OK+MB_ICONEXCLAMATION);
end;

procedure TForm6.Button21Click(Sender: TObject);
begin
  DATABASETARIH();
  IF FORM6.RadioButton3.Checked=TRUE THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT * FROM illnesses WHERE date <
'+#39+datetostr(form6.DateTimePicker3.Date)+#39;
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE IF FORM6.RadioButton4.Checked=TRUE THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT * FROM illnesses WHERE date <=
'+#39+datetostr(form6.DateTimePicker3.Date)+#39;
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE IF FORM6.RadioButton5.Checked=TRUE THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT * FROM illnesses WHERE date >
'+#39+datetostr(form6.DateTimePicker3.Date)+#39;
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE IF FORM6.RadioButton6.Checked=TRUE THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT * FROM illnesses WHERE date
>='+#39+datetostr(form6.DateTimePicker3.Date)+#39;
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE IF FORM6.RadioButton8.Checked=TRUE THEN
  BEGIN
```

```
      FORM6.ADOQuery1.SQL.Text:='SELECT * FROM illnesses WHERE date <>
    '+#39+datetostr(form6.DateTimePicker3.Date)+#39;
      form6.ADOQuery1.Open;
      ADODURUMU();
    END
    ELSE IF FORM6.RadioButton7.Checked=TRUE THEN
    BEGIN
      FORM6.ADOQuery1.SQL.Text:='SELECT * FROM illnesses WHERE date
  between '+#39+datetostr(form6.DateTimePicker3.Date)+#39+' and
  '+#39+datetostr(form6.DateTimePicker4.Date)+#39;
      form6.ADOQuery1.Open;
      ADODURUMU();
    END
    ELSE IF FORM6.RadioButton22.Checked=TRUE THEN
    BEGIN
      FORM6.ADOQuery1.SQL.Text:='SELECT * FROM illnesses WHERE date =
  '+#39+datetostr(form6.DateTimePicker3.Date)+#39;
      form6.ADOQuery1.Open;
      ADODURUMU();
    END
    ELSE IF (FORM6.RadioButton3.Checked=FALSE) AND
  (FORM6.RadioButton4.Checked=FALSE) AND
  (FORM6.RadioButton5.Checked=FALSE) AND
  (FORM6.RadioButton6.Checked=FALSE) AND
  (FORM6.RadioButton7.Checked=FALSE) AND
  (FORM6.RadioButton8.Checked=FALSE) AND
  (FORM6.RadioButton22.Checked=FALSE) THEN
      APPLICATION.MessageBox('PLEASE SELECT ANY DATE
  CRITERIA','CAUTION',MB_OK+MB_ICONEXCLAMATION);
    NORMALTARIH();
  end;

  procedure TForm6.Button19Click(Sender: TObject);
  begin
    if (FORM6.RadioButton1.Checked=TRUE) AND (FORM6.Edit5.Text <> '')
  THEN
    BEGIN
      FORM6.ADOQuery1.SQL.Text:='SELECT * FROM illnesses WHERE
  protocolno='+#39+form6.Edit5.Text+#39;
      form6.ADOQuery1.Open;
      ADODURUMU();
    END
    ELSE IF (FORM6.RadioButton2.Checked=TRUE) AND (FORM6.Edit9.Text <>
  '') THEN
    BEGIN
      FORM6.ADOQuery1.SQL.Text:='SELECT * FROM operations WHERE
  protocolno='+#39+form6.Edit9.Text+#39;
      form6.ADOQuery1.Open;
      ADODURUMU();
    END
    ELSE if (FORM6.RadioButton1.Checked=TRUE) AND (FORM6.Edit5.Text =
  '') THEN
    BEGIN
      APPLICATION.MessageBox('PLEASE ENTER ILLNESSES PROTOCOL
  NO','ATTENTION',MB_OK+MB_ICONEXCLAMATION);
    END
    ELSE if (FORM6.RadioButton2.Checked=TRUE) AND (FORM6.Edit9.Text =
  '') THEN
    BEGIN
```

```
    APPLICATION.MessageBox('PLEASE ENTER OPERATIONS PROTOCOL
NO','ATTENTION',MB_OK+MB_ICONEXCLAMATION);
  END
  ELSE IF (FORM6.RadioButton1.Checked=FALSE) AND
(FORM6.RadioButton2.Checked=FALSE) THEN
    APPLICATION.MessageBox('PLEASE SELECT ILLNESSES PROTOCOL NO OR
OPERATIONS PROTOCOL NO','ATTENTION',MB_OK+MB_ICONEXCLAMATION);
end;

procedure TForm6.Button17Click(Sender: TObject);
begin
  IF FORM6.Edit4.Text <> '' THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT * FROM operations WHERE
operationname='+#39+form6.Edit4.Text+#39;
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE
    APPLICATION.MessageBox('PLEASE ENTER OPERATION
NAME','ATTENTION',MB_OK+MB_ICONEXCLAMATION);
end;

procedure TForm6.Button15Click(Sender: TObject);
begin
  DATABASETARIH();

  IF FORM6.RadioButton9.Checked=TRUE THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT * FROM operations WHERE date <
'+#39+datetostr(form6.DateTimePicker2.Date)+#39;
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE IF FORM6.RadioButton10.Checked=TRUE THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT * FROM operations WHERE date <=
'+#39+datetostr(form6.DateTimePicker2.Date)+#39;
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE IF FORM6.RadioButton11.Checked=TRUE THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT * FROM operations WHERE date >
'+#39+datetostr(form6.DateTimePicker2.Date)+#39;
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE IF FORM6.RadioButton12.Checked=TRUE THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT * FROM operations WHERE date
>='+#39+datetostr(form6.DateTimePicker2.Date)+#39;
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE IF FORM6.RadioButton13.Checked=TRUE THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT * FROM operations WHERE date <>
'+#39+datetostr(form6.DateTimePicker2.Date)+#39;
    form6.ADOQuery1.Open;
```

```
    ADODURUMU();
  END
  ELSE IF FORM6.RadioButton14.Checked=TRUE THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT * FROM operations WHERE date
between '+#39+datetostr(form6.DateTimePicker2.Date)+#39+' and
'+#39+datetostr(form6.DateTimePicker6.Date)+#39;
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE IF FORM6.RadioButton23.Checked=TRUE THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT * FROM operations WHERE date =
'+#39+datetostr(form6.DateTimePicker2.Date)+#39;
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE IF (FORM6.RadioButton9.Checked=FALSE) AND
(FORM6.RadioButton10.Checked=FALSE) AND
(FORM6.RadioButton11.Checked=FALSE) AND
(FORM6.RadioButton12.Checked=FALSE) AND
(FORM6.RadioButton13.Checked=FALSE) AND
(FORM6.RadioButton14.Checked=FALSE) AND
(FORM6.RadioButton23.Checked=FALSE) THEN
    APPLICATION.MessageBox('PLEASE SELECT ANY DATE
CRITERIA','CAUTION',MB_OK+MB_ICONEXCLAMATION);
  NORMALTARIH();
end;

procedure TForm6.Edit6KeyPress(Sender: TObject; var Key: Char);
begin
  //SADECE RAKAMLAR GİREBİLMEK İÇİN, BACKSPACE VE DELETE TUŞUNA
BASABİLMEK İÇİN
  IF (KEY<>'0') AND (KEY<>'1') AND (KEY<>'2') AND (KEY<>'3') AND
(KEY<>'4') AND (KEY<>'5') AND (KEY<>'6') AND (KEY<>'7') AND (KEY<>'8')
AND (KEY<>'9') AND (KEY<>#8) AND (KEY<>#127) THEN
    //EĞER BAŞKA BİR TUŞ BASILMIŞSA İPTAL ETMESİ İÇİN
    KEY:=#0;
end;


procedure TForm6.Edit9KeyPress(Sender: TObject; var Key: Char);
begin
  FORM6.Edit6KeyPress(NIL,KEY);
end;

procedure TForm6.Edit5KeyPress(Sender: TObject; var Key: Char);
begin
  FORM6.Edit6KeyPress(NIL,KEY);
end;

procedure TForm6.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
  FORM6.Edit6KeyPress(NIL,KEY);
end;

procedure TForm6.Button29Click(Sender: TObject);
begin
  IF FORM6.Edit8.Text <> '' THEN
```

```
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT * FROM patients WHERE name
like'+#39+'%'+form6.Edit8.Text+'%'+#39;
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE
    APPLICATION.MessageBox('PLEASE ENTER
NAME','ATTENTION',MB_OK+MB_ICONEXCLAMATION);
end;

procedure TForm6.RadioButton20Click(Sender: TObject);
begin
  FORM6.DateTimePicker1.Left:=40;
  FORM6.Label16.Visible:=TRUE;
  FORM6.DateTimePicker5.Visible:=TRUE;
end;

procedure TForm6.RadioButton15Click(Sender: TObject);
begin
  FORM6.DateTimePicker1.Left:=112;
  FORM6.Label16.Visible:=FALSE;
  FORM6.DateTimePicker5.Visible:=FALSE;
end;

procedure TForm6.RadioButton16Click(Sender: TObject);
begin
  FORM6.RadioButton15Click(NIL);
end;

procedure TForm6.RadioButton17Click(Sender: TObject);
begin
  FORM6.RadioButton15Click(NIL);
end;

procedure TForm6.RadioButton18Click(Sender: TObject);
begin
  FORM6.RadioButton15Click(NIL);
end;

procedure TForm6.RadioButton19Click(Sender: TObject);
begin
  FORM6.RadioButton15Click(NIL);
end;

procedure TForm6.RadioButton7Click(Sender: TObject);
begin
  FORM6.DateTimePicker3.Left:=40;
  FORM6.Label10.Visible:=TRUE;
  FORM6.DateTimePicker4.Visible:=TRUE;
end;

procedure TForm6.RadioButton3Click(Sender: TObject);
begin
  FORM6.DateTimePicker3.Left:=112;
  FORM6.Label10.Visible:=FALSE;
  FORM6.DateTimePicker4.Visible:=FALSE;
end;

procedure TForm6.RadioButton4Click(Sender: TObject);
```

```
begin
  Form6.RadioButton3Click(NIL);
end;

procedure TForm6.RadioButton5Click(Sender: TObject);
begin
  Form6.RadioButton3Click(NIL);
end;

procedure TForm6.RadioButton6Click(Sender: TObject);
begin
  Form6.RadioButton3Click(NIL);
end;

procedure TForm6.RadioButton8Click(Sender: TObject);
begin
  Form6.RadioButton3Click(NIL);
end;

procedure TForm6.RadioButton14Click(Sender: TObject);
begin
  FORM6.DateTimePicker2.Left:=40;
  FORM6.Label15.Visible:=TRUE;
  FORM6.DateTimePicker6.Visible:=TRUE;
end;

procedure TForm6.RadioButton9Click(Sender: TObject);
begin
  FORM6.DateTimePicker2.Left:=112;
  FORM6.Label15.Visible:=FALSE;
  FORM6.DateTimePicker6.Visible:=FALSE;
end;

procedure TForm6.RadioButton10Click(Sender: TObject);
begin
  FORM6.RadioButton9Click(NIL);
end;

procedure TForm6.RadioButton11Click(Sender: TObject);
begin
  FORM6.RadioButton9Click(NIL);
end;

procedure TForm6.RadioButton12Click(Sender: TObject);
begin
  FORM6.RadioButton9Click(NIL);
end;

procedure TForm6.RadioButton13Click(Sender: TObject);
begin
  FORM6.RadioButton9Click(NIL);
end;

procedure TForm6.RadioButton21Click(Sender: TObject);
begin
  FORM6.RadioButton15Click(NIL);
end;

procedure TForm6.RadioButton22Click(Sender: TObject);
begin
```

```
  Form6.RadioButton3Click(NIL);
end;

procedure TForm6.RadioButton23Click(Sender: TObject);
begin
  FORM6.RadioButton9Click(NIL);
end;

procedure TForm6.Button30Click(Sender: TObject);
begin
  IF FORM6.Edit7.Text <> '' THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT * FROM patients WHERE surname
like '+#39+'%'+form6.Edit7.Text+'%'+#39;
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE
    APPLICATION.MessageBox('PLEASE ENTER
SURNAME','ATTENTION',MB_OK+MB_ICONEXCLAMATION);
end;

procedure TForm6.Button31Click(Sender: TObject);
begin
  IF FORM6.Edit6.Text <> '' THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT * FROM patients WHERE customerid
like '+#39+'%'+form6.Edit6.Text+'%'+#39;
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE
    APPLICATION.MessageBox('PLEASE ENTER PATIENT
ID','ATTENTION',MB_OK+MB_ICONEXCLAMATION);
end;

procedure TForm6.Button39Click(Sender: TObject);
begin
  FORM6.ADOQuery1.SQL.Text:='SELECT customerid,name,surname FROM
patients';
  form6.ADOQuery1.Open;
  ADODURUMU();
end;

procedure TForm6.Button40Click(Sender: TObject);
begin
  FORM6.ADOQuery1.SQL.Text:='SELECT customerid,surname,name FROM
patients';
  form6.ADOQuery1.Open;
  ADODURUMU();
end;

procedure TForm6.Button47Click(Sender: TObject);
begin
  FORM6.ADOQuery1.SQL.Text:='SELECT customerid,name,surname FROM
patients';
  form6.ADOQuery1.Open;
  ADODURUMU();
end;
```

```
procedure TForm6.Button49Click(Sender: TObject);
begin
  FORM6.ADOQuery1.SQL.Text:='SELECT
customerid,operationname,date,protocolno FROM operations';
  form6.ADOQuery1.Open;
  ADODURUMU();
end;

procedure TForm6.Button38Click(Sender: TObject);
begin
  IF FORM6.Edit4.Text <> '' THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT * FROM operations WHERE
operationname like '+#39+'%'+form6.Edit4.Text+'%'+#39;
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE
    APPLICATION.MessageBox('PLEASE ENTER OPERATION
NAME','ATTENTION',MB_OK+MB_ICONEXCLAMATION);
end;

procedure TForm6.Button32Click(Sender: TObject);
begin
if (FORM6.RadioButton1.Checked=TRUE) AND (FORM6.Edit5.Text <> '') THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT * FROM illnesses WHERE
protocolno like '+#39+'%'+form6.Edit5.Text+'%'+#39;
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE IF (FORM6.RadioButton2.Checked=TRUE) AND (FORM6.Edit9.Text <>
'') THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT * FROM operations WHERE
protocolno like '+#39+'%'+form6.Edit9.Text+'%'+#39;
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE if (FORM6.RadioButton1.Checked=TRUE) AND (FORM6.Edit5.Text =
'') THEN
  BEGIN
    APPLICATION.MessageBox('PLEASE ENTER ILLNESSES PROTOCOL
NO','ATTENTION',MB_OK+MB_ICONEXCLAMATION);
  END
  ELSE if (FORM6.RadioButton2.Checked=TRUE) AND (FORM6.Edit9.Text =
'') THEN
  BEGIN
    APPLICATION.MessageBox('PLEASE ENTER OPERATIONS PROTOCOL
NO','ATTENTION',MB_OK+MB_ICONEXCLAMATION);
  END
  ELSE IF (FORM6.RadioButton1.Checked=FALSE) AND
(FORM6.RadioButton2.Checked=FALSE) THEN
    APPLICATION.MessageBox('PLEASE SELECT ILLNESSES PROTOCOL NO OR
OPERATIONS PROTOCOL NO','ATTENTION',MB_OK+MB_ICONEXCLAMATION);

end;

procedure TForm6.Button48Click(Sender: TObject);
begin
```

```
  if FORM6.RadioButton1.Checked=TRUE THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT
customerid,protocolno,diagnosis,date FROM illnesses';
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE IF FORM6.RadioButton2.Checked=TRUE THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT
customerid,protocolno,operationname,date FROM operations';
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE IF (FORM6.RadioButton1.Checked=FALSE) AND
(FORM6.RadioButton2.Checked=FALSE) THEN
    APPLICATION.MessageBox('PLEASE SELECT ILLNESSES PROTOCOL NO OR
OPERATIONS PROTOCOL NO','ATTENTION',MB_OK+MB_ICONEXCLAMATION);

end;

procedure TForm6.Button33Click(Sender: TObject);
begin
  IF FORM6.Edit3.Text <> '' THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT
customerid,name,surname,mothername,birthdate,tcidno,adress,telno,mobil
eno,email,cronological_illnesses,cronological_medicine,bloodgroup,fath
ername,note FROM patients WHERE mothername like
'+#39+'%'+form6.Edit3.Text+'%'+#39;
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE
    APPLICATION.MessageBox('PLEASE ENTER MOTHER
NAME','ATTENTION',MB_OK+MB_ICONEXCLAMATION);

end;

procedure TForm6.Button42Click(Sender: TObject);
begin
  FORM6.ADOQuery1.SQL.Text:='SELECT customerid,mothername FROM
patients';
  form6.ADOQuery1.Open;
  ADODURUMU();
end;

procedure TForm6.Button34Click(Sender: TObject);
begin
  IF FORM6.Edit2.Text <> '' THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT
customerid,name,surname,fathername,birthdate,tcidno,adress,telno,mobil
eno,email,cronological_illnesses,cronological_medicine,bloodgroup,moth
ername,note FROM patients WHERE fathername like
'+#39+'%'+form6.Edit2.Text+'%'+#39;
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE
```

```
    APPLICATION.MessageBox('PLEASE ENTER FATHER
NAME','ATTENTION',MB_OK+MB_ICONEXCLAMATION);
end;

procedure TForm6.Button41Click(Sender: TObject);
begin
  FORM6.ADOQuery1.SQL.Text:='SELECT customerid,fathername FROM
patients';
  form6.ADOQuery1.Open;
  ADODURUMU();
end;

procedure TForm6.Button35Click(Sender: TObject);
begin
  IF FORM6.Edit1.Text <> '' THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT * FROM patients WHERE tcidno
like '+#39+'%'+form6.Edit1.Text+'%'+#39;
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE
    APPLICATION.MessageBox('PLEASE ENTER TC ID
NO','ATTENTION',MB_OK+MB_ICONEXCLAMATION);
end;

procedure TForm6.Button43Click(Sender: TObject);
begin
  FORM6.ADOQuery1.SQL.Text:='SELECT customerid,tcidno,name,surname
FROM patients';
  form6.ADOQuery1.Open;
  ADODURUMU();
end;

procedure TForm6.Button36Click(Sender: TObject);
begin
  IF FORM6.Edit1.Text <> '' THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT * FROM patients WHERE adress
like '+#39+'%'+form6.Memo2.Text+'%'+#39;
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE
    APPLICATION.MessageBox('PLEASE ENTER
ADDRESS','ATTENTION',MB_OK+MB_ICONEXCLAMATION);
end;

procedure TForm6.Button44Click(Sender: TObject);
begin
  FORM6.ADOQuery1.SQL.Text:='SELECT customerid,adress,name,surname
FROM patients';
  form6.ADOQuery1.Open;
  ADODURUMU();
end;

procedure TForm6.Button45Click(Sender: TObject);
begin
  FORM6.ADOQuery1.SQL.Text:='SELECT customerid,bloodgroup,name,surname
FROM patients';
```

```
  form6.ADOQuery1.Open;
  ADODURUMU();
end;

procedure TForm6.Button37Click(Sender: TObject);
begin
  IF FORM6.Memo1.Text <> '' THEN
  BEGIN
    FORM6.ADOQuery1.SQL.Text:='SELECT
customerid,name,surname,cronological_illnesses,birthdate,tcidno,adress
,telno,mobileno,email,bloodgroup,cronological_medicine,fathername,moth
ername,note FROM patients WHERE cronological_illnesses like
'+#39+'%'+form6.Memo1.Text+'%'+#39;
    form6.ADOQuery1.Open;
    ADODURUMU();
  END
  ELSE
    APPLICATION.MessageBox('PLEASE ENTER CRONOLOGICAL
ILLNESSES','ATTENTION',MB_OK+MB_ICONEXCLAMATION);
end;

procedure TForm6.Button46Click(Sender: TObject);
begin
  FORM6.ADOQuery1.SQL.Text:='SELECT
customerid,cronological_illnesses,name,surname FROM patients';
  form6.ADOQuery1.Open;
  ADODURUMU();
end;

procedure TForm6.PageControl1Change(Sender: TObject);
begin
  SEARCHFORMUNUTEMIZLE();
end;

procedure TForm6.DBGrid1DblClick(Sender: TObject);
begin
IF (FORM6.DBGrid1.FieldCount <> 0) AND (FORM6.DBGrid1.Fields[0].Text
<> '') AND (FORM6.DBGrid1.Fields[0].Text <> NULL) THEN
  BEGIN
    FORM9.Edit1.Text:=form6.DBGrid1.Fields[0].Text;
    form9.Show;
  END;
end;

end.
```

7. UNIT8 CODES

```
unit Unit8;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls, Grids, DBGrids, ExtCtrls, DB, ADODB;

type
```

```
  TForm8 = class(TForm)
    GroupBox1: TGroupBox;
    Panel1: TPanel;
    GroupBox2: TGroupBox;
    DBGrid1: TDBGrid;
    GroupBox3: TGroupBox;
    Label1: TLabel;
    Edit1: TEdit;
    GroupBox4: TGroupBox;
    Button1: TButton;
    Button2: TButton;
    Edit2: TEdit;
    Label2: TLabel;
    DataSource1: TDataSource;
    ADOQuery1: TADOQuery;
    procedure DBGrid1CellClick(Column: TColumn);
    procedure Button1Click(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure DBGrid1DblClick(Sender: TObject);
    procedure DBGrid1KeyUp(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure FormKeyPress(Sender: TObject; var Key: Char);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form8: TForm8;

implementation

uses Unit4, Unit2, Unit3;

{$R *.dfm}

procedure TForm8.DBGrid1CellClick(Column: TColumn);
begin
  FORM8.Edit1.Text:=FORM8.DBGrid1.Fields[1].Text;
  FORM8.Edit2.Text:=FORM8.DBGrid1.Fields[2].Text;
end;

procedure TForm8.Button1Click(Sender: TObject);
begin
  //BURADAKİ DURUM DEĞİŞKENİ HANGİ FORMDAN BİLGİ İSTENDİĞİNİ
BELİRTİYOR VE ONA GÖRE O FORMA GEREKLİ BİLGİLERİ GÖNDERİYOR
  IF DURUM = 2 THEN
  BEGIN
    FORM2.Edit1.Text:=FORM8.DBGrid1.Fields[0].Text;
    FORM2.Edit2.Text:=FORM8.DBGrid1.Fields[1].Text;
    FORM2.Edit3.Text:=FORM8.DBGrid1.Fields[2].Text;
    FORM2.Button5.Caption:='NEW RECORD For'+' '+form2.Edit2.Text+'
'+form2.Edit3.Text;
  end
  ELSE  IF DURUM = 3 THEN
  BEGIN
    FORM3.Edit1.Text:=FORM8.DBGrid1.Fields[0].Text;
    FORM3.Edit2.Text:=FORM8.DBGrid1.Fields[1].Text;
```

```
    FORM3.Edit3.Text:=FORM8.DBGrid1.Fields[2].Text;
    FORM3.Button3.Caption:='NEW RECORD For'+' '+form3.Edit2.Text+'
'+form3.Edit3.Text;
  end;
  FORM8.Hide;
END;

procedure TForm8.FormShow(Sender: TObject);
begin
  //HER FORMUN AÇILIŞNDA DBGRID'DE PATIEN'LERİN GÖZÜKMESİ İÇİN YAZILDI
  FORM8.Edit1.Clear;//PATIENT LIST'DEKİ NAME KISMININ TEMİZLENMESİ
İÇİN
  FORM8.Edit2.Clear;//PATIENT LIST'DEKİ SURNAME KISMININ TEMİZLENMESİ
İÇİN
  FORM8.ADOQuery1.SQL.Text:='SELECT * FROM patients';
  FORM8.ADOQuery1.Open;
end;

procedure TForm8.DBGrid1DblClick(Sender: TObject);
begin
  //BURADA DBGRID'E ÇİFT TIKLANINCA; OK TUŞUNA BASILMIŞ GİBİ İŞLEM
YAPMASINI SAĞLADIM
  FORM8.Button1.Click;
end;

procedure TForm8.DBGrid1KeyUp(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  //BURADA AŞAĞI VE YUKARI OK TUŞLARINA BASILINCA FORMDAKİ DEĞERLERİN
AYNI DBGRID'IN ÜZERİNE BASILDIĞINDA OLDUĞU GİBİ DEĞİŞMESİ İÇİN YAPTIM
  //NIL DEĞERI BOŞ COLUMN SEÇMESİ İÇİN.YANİ KODUN AKTİF OLMASI İÇİN
  FORM8.DBGrid1CellClick(NIL);
end;

procedure TForm8.FormKeyPress(Sender: TObject; var Key: Char);
begin
  //BURADA ENTER'A BASILINCA; OK TUŞUNA BASILMIŞ GİBİ İŞLEM YAPMASINI
SAĞLADIM
  IF KEY=#13 THEN
    FORM8.Button1.Click;
end;

procedure TForm8.Button2Click(Sender: TObject);
begin
  //BURADA ŞEÇİM İŞLEMİNİN İPTAL EDİLEREK FORMUN KAPATILMASINI
SAĞLADIM
  FORM8.Edit1.Clear;
  FORM8.Edit2.Clear;
  FORM8.Hide;
end;

end.
```

8. UNIT9 CODES

```
unit Unit9;

interface
```

```
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls, Grids, DBGrids, ComCtrls, DB, ADODB;

type
  TForm9 = class(TForm)
    GroupBox2: TGroupBox;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    Label14: TLabel;
    Label15: TLabel;
    Label16: TLabel;
    Label17: TLabel;
    Label18: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Edit5: TEdit;
    Memo1: TMemo;
    Edit6: TEdit;
    Edit7: TEdit;
    Edit8: TEdit;
    Edit9: TEdit;
    Edit10: TEdit;
    Memo2: TMemo;
    Memo3: TMemo;
    Memo4: TMemo;
    Edit11: TEdit;
    GroupBox1: TGroupBox;
    PageControl1: TPageControl;
    TabSheet1: TTabSheet;
    TabSheet2: TTabSheet;
    GroupBox3: TGroupBox;
    GroupBox4: TGroupBox;
    DBGrid1: TDBGrid;
    DBGrid2: TDBGrid;
    GroupBox5: TGroupBox;
    Button1: TButton;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label19: TLabel;
    Label20: TLabel;
    Label21: TLabel;
    Label22: TLabel;
    Label23: TLabel;
    Label24: TLabel;
    Label25: TLabel;
    Edit12: TEdit;
```

```
    Edit13: TEdit;
    Memo5: TMemo;
    Edit14: TEdit;
    Edit15: TEdit;
    Memo6: TMemo;
    Edit16: TEdit;
    Edit17: TEdit;
    Edit18: TEdit;
    Memo7: TMemo;
    Label26: TLabel;
    Label27: TLabel;
    Label28: TLabel;
    Label29: TLabel;
    Label30: TLabel;
    Label31: TLabel;
    Label32: TLabel;
    Label33: TLabel;
    Label34: TLabel;
    Label35: TLabel;
    Label36: TLabel;
    Label37: TLabel;
    Edit19: TEdit;
    Edit20: TEdit;
    Memo8: TMemo;
    Edit21: TEdit;
    Edit22: TEdit;
    Memo9: TMemo;
    Edit23: TEdit;
    Edit24: TEdit;
    Edit25: TEdit;
    Memo10: TMemo;
    Memo11: TMemo;
    Memo12: TMemo;
    Edit26: TEdit;
    Label38: TLabel;
    ADOQuery1: TADOQuery;
    ADOQuery2: TADOQuery;
    ADOQuery3: TADOQuery;
    DataSource1: TDataSource;
    DataSource2: TDataSource;
    DataSource3: TDataSource;
    Label39: TLabel;
    Label40: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure DBGrid1CellClick(Column: TColumn);
    procedure DBGrid1KeyUp(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure DBGrid2CellClick(Column: TColumn);
    procedure DBGrid2KeyUp(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure Edit1Change(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form9: TForm9;
```

```
OCEDURE PATIENTDETAILSADODURUMU(); forward;

plementation

es Unit1, Unit2, Unit4, Unit5, Unit6;

R *.dfm}

OCEDURE PATIENTDETAILSADODURUMU();
GIN
 IF FORM9.ADOQuery2.RecordCount <> 0 THEN
 BEGIN
   FORM9.DBGrid1.Visible:=TRUE;
   FORM9.Label39.Visible:=FALSE;
   FORM9.Edit19.Enabled:=TRUE;
   FORM9.Edit20.Enabled:=TRUE;
   FORM9.Memo11.Enabled:=TRUE;
   FORM9.Memo8.Enabled:=TRUE;
   FORM9.Memo12.Enabled:=TRUE;
   FORM9.Edit21.Enabled:=TRUE;
   FORM9.Edit22.Enabled:=TRUE;
   FORM9.Memo9.Enabled:=TRUE;
   FORM9.Edit23.Enabled:=TRUE;
   FORM9.Edit24.Enabled:=TRUE;
   FORM9.Edit25.Enabled:=TRUE;
   FORM9.Memo10.Enabled:=TRUE;
   FORM9.Label26.Enabled:=TRUE;
   FORM9.Label27.Enabled:=TRUE;
   FORM9.Label28.Enabled:=TRUE;
   FORM9.Label29.Enabled:=TRUE;
   FORM9.Label30.Enabled:=TRUE;
   FORM9.Label31.Enabled:=TRUE;
   FORM9.Label32.Enabled:=TRUE;
   FORM9.Label33.Enabled:=TRUE;
   FORM9.Label34.Enabled:=TRUE;
   FORM9.Label35.Enabled:=TRUE;
   FORM9.Label36.Enabled:=TRUE;
   FORM9.Label37.Enabled:=TRUE;
 END
 ELSE IF FORM9.ADOQuery2.RecordCount = 0 THEN
 BEGIN
   FORM9.DBGrid1.Visible:=FALSE;
   FORM9.Label39.Visible:=TRUE;
   //FORMDAKİ İLLNESSES KONTROLLERİNİN SÖNÜK GÖZÜKMESİ İÇİN
   FORM9.Edit19.Enabled:=FALSE;
   FORM9.Edit20.Enabled:=FALSE;
   FORM9.Memo11.Enabled:=FALSE;
   FORM9.Memo8.Enabled:=FALSE;
   FORM9.Memo12.Enabled:=FALSE;
   FORM9.Edit21.Enabled:=FALSE;
   FORM9.Edit22.Enabled:=FALSE;
   FORM9.Memo9.Enabled:=FALSE;
   FORM9.Edit23.Enabled:=FALSE;
   FORM9.Edit24.Enabled:=FALSE;
   FORM9.Edit25.Enabled:=FALSE;
   FORM9.Memo10.Enabled:=FALSE;
   //LABELLERIN SÖNÜK GÖZÜKMESİ İÇİN
   FORM9.Label26.Enabled:=FALSE;
   FORM9.Label27.Enabled:=FALSE;
   FORM9.Label28.Enabled:=FALSE;
```

```
  FORM9.Label29.Enabled:=FALSE;
  FORM9.Label30.Enabled:=FALSE;
  FORM9.Label31.Enabled:=FALSE;
  FORM9.Label32.Enabled:=FALSE;
  FORM9.Label33.Enabled:=FALSE;
  FORM9.Label34.Enabled:=FALSE;
  FORM9.Label35.Enabled:=FALSE;
  FORM9.Label36.Enabled:=FALSE;
  FORM9.Label37.Enabled:=FALSE;
END;
IF FORM9.ADOQuery3.RecordCount <> 0 THEN
BEGIN
  FORM9.DBGrid2.Visible:=TRUE;
  FORM9.Label40.Visible:=FALSE;
  FORM9.Edit12.Enabled:=TRUE;
  FORM9.Edit15.Enabled:=TRUE;
  FORM9.Edit26.Enabled:=TRUE;
  FORM9.Edit13.Enabled:=TRUE;
  FORM9.Memo5.Enabled:=TRUE;
  FORM9.Edit14.Enabled:=TRUE;
  FORM9.Memo6.Enabled:=TRUE;
  FORM9.Edit16.Enabled:=TRUE;
  FORM9.Edit17.Enabled:=TRUE;
  FORM9.Edit18.Enabled:=TRUE;
  FORM9.Memo7.Enabled:=TRUE;
  FORM9.Label1.Enabled:=TRUE;
  FORM9.Label2.Enabled:=TRUE;
  FORM9.Label3.Enabled:=TRUE;
  FORM9.Label19.Enabled:=TRUE;
  FORM9.Label20.Enabled:=TRUE;
  FORM9.Label21.Enabled:=TRUE;
  FORM9.Label22.Enabled:=TRUE;
  FORM9.Label23.Enabled:=TRUE;
  FORM9.Label24.Enabled:=TRUE;
  FORM9.Label25.Enabled:=TRUE;
  FORM9.Label38.Enabled:=TRUE;

END
ELSE IF FORM9.ADOQuery3.RecordCount = 0 THEN
BEGIN
  FORM9.DBGrid2.Visible:=FALSE;
  FORM9.Label40.Visible:=TRUE;
  FORM9.Edit12.Enabled:=FALSE;
  FORM9.Edit15.Enabled:=FALSE;
  FORM9.Edit26.Enabled:=FALSE;
  FORM9.Edit13.Enabled:=FALSE;
  FORM9.Memo5.Enabled:=FALSE;
  FORM9.Edit14.Enabled:=FALSE;
  FORM9.Memo6.Enabled:=FALSE;
  FORM9.Edit16.Enabled:=FALSE;
  FORM9.Edit17.Enabled:=FALSE;
  FORM9.Edit18.Enabled:=FALSE;
  FORM9.Memo7.Enabled:=FALSE;
  FORM9.Label1.Enabled:=FALSE;
  FORM9.Label2.Enabled:=FALSE;
  FORM9.Label3.Enabled:=FALSE;
  FORM9.Label19.Enabled:=FALSE;
  FORM9.Label20.Enabled:=FALSE;
  FORM9.Label21.Enabled:=FALSE;
  FORM9.Label22.Enabled:=FALSE;
```

```
      FORM9.Label23.Enabled:=FALSE;
      FORM9.Label24.Enabled:=FALSE;
      FORM9.Label25.Enabled:=FALSE;
      FORM9.Label38.Enabled:=FALSE;
    END;
 END;

 procedure TForm9.Button1Click(Sender: TObject);
 begin
 form9.Hide;

 end;

 procedure TForm9.DBGrid1CellClick(Column: TColumn);
 begin
 IF (FORM9.DBGrid1.FieldCount <> 0) AND (FORM9.DBGrid1.Fields[0].Text
 <> '') AND (FORM9.DBGrid1.Fields[0].Text <> NULL) THEN
   BEGIN
     FORM9.Edit19.Text:=FORM9.DBGrid1.Fields[1].Text;
     FORM9.Edit20.Text:=FORM9.DBGrid1.Fields[2].Text;
     FORM9.Memo11.Text:=FORM9.DBGrid1.Fields[3].Text;
     FORM9.Memo8.Text:=FORM9.DBGrid1.Fields[4].Text;
     FORM9.Memo12.Text:=FORM9.DBGrid1.Fields[5].Text;
     FORM9.Edit21.Text:=FORM9.DBGrid1.Fields[6].Text;
     FORM9.Edit22.Text:=FORM9.DBGrid1.Fields[7].Text;
     FORM9.Memo9.Text:=FORM9.DBGrid1.Fields[8].Text;
     FORM9.Edit23.Text:=FORM9.DBGrid1.Fields[9].Text;
     FORM9.Edit24.Text:=FORM9.DBGrid1.Fields[10].Text;
     FORM9.Edit25.Text:=FORM9.DBGrid1.Fields[11].Text;
     FORM9.Memo10.Text:=FORM9.DBGrid1.Fields[12].Text;
   END;
 end;

 procedure TForm9.DBGrid1KeyUp(Sender: TObject; var Key: Word;
   Shift: TShiftState);
 begin
   form9.DBGrid1CellClick(nil);
 end;

 procedure TForm9.DBGrid2CellClick(Column: TColumn);
 begin
 IF (FORM9.DBGrid2.FieldCount <> 0) AND (FORM9.DBGrid2.Fields[0].Text
 <> '') AND (FORM9.DBGrid2.Fields[0].Text <> NULL) THEN
   BEGIN
     FORM9.Edit12.Text:=FORM9.DBGrid2.Fields[1].Text;
     FORM9.Edit26.Text:=FORM9.DBGrid2.Fields[3].Text;
     FORM9.Edit13.Text:=FORM9.DBGrid2.Fields[2].Text;
     FORM9.Memo5.Text:=FORM9.DBGrid2.Fields[4].Text;
     FORM9.Edit4.Text:=FORM9.DBGrid2.Fields[5].Text;
     FORM9.Edit15.Text:=FORM9.DBGrid2.Fields[6].Text;
     FORM9.Memo6.Text:=FORM9.DBGrid2.Fields[7].Text;
     FORM9.Edit16.Text:=FORM9.DBGrid2.Fields[8].Text;
     FORM9.Edit17.Text:=FORM9.DBGrid2.Fields[9].Text;
     FORM9.Edit18.Text:=FORM9.DBGrid2.Fields[10].Text;
     FORM9.Memo7.Text:=FORM9.DBGrid2.Fields[11].Text;
   END;
 end;

 procedure TForm9.DBGrid2KeyUp(Sender: TObject; var Key: Word;
   Shift: TShiftState);
```

```
egin
 Form9.DBGrid2CellClick(nil);
nd;

rocedure TForm9.Edit1Change(Sender: TObject);
egin
    IF FORM9.Edit1.Text <> '' THEN
    BEGIN
      form9.ADOQuery1.SQL.Text:='select * from patients where
ustomerid = '+#39+FORM9.Edit1.Text+#39;
      form9.ADOQuery1.Open;
      IF form9.ADOQuery1.RecordCount <> 0 THEN
      BEGIN
        FORM9.Edit1.Text:=FORM9.ADOQuery1['customerid'];
        FORM9.Edit2.Text:=FORM9.ADOQuery1['name'];
        FORM9.Edit3.Text:=FORM9.ADOQuery1['surname'];
        FORM9.Edit4.Text:=FORM9.ADOQuery1['birthdate'];
        FORM9.Edit5.Text:=FORM9.ADOQuery1['tcidno'];
        FORM9.Memo1.Text:=FORM9.ADOQuery1['adress'];
        FORM9.Edit6.Text:=FORM9.ADOQuery1['telno'];
        FORM9.Edit7.Text:=FORM9.ADOQuery1['mobileno'];
        FORM9.Edit8.Text:=FORM9.ADOQuery1['email'];
        FORM9.Edit9.Text:=FORM9.ADOQuery1['mothername'];
        FORM9.Edit10.Text:=FORM9.ADOQuery1['fathername'];
        FORM9.Memo2.Text:=FORM9.ADOQuery1['cronological_medicine'];
        FORM9.Memo3.Text:=FORM9.ADOQuery1['cronological_illnesses'];
        FORM9.Edit11.Text:=FORM9.ADOQuery1['bloodgroup'];
        FORM9.Memo4.Text:=FORM9.ADOQuery1['note'];
      END;
      form9.ADOQuery2.SQL.Text:='select * from illnesses where
ustomerid = '+#39+FORM9.Edit1.Text+#39;
      form9.ADOQuery2.Open;
      form9.ADOQuery3.SQL.Text:='select * from operations where
ustomerid = '+#39+FORM9.Edit1.Text+#39;
      form9.ADOQuery3.Open;
      PATIENTDETAILSADODURUMU();
    END;
nd;

nd.
```