NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

Inventory Control System

Graduation Project

COM- 400

Student: Adem ATÇEKEN (20010705)

Supervisor: Mr. Umit ILHAN

Nicosia – 2006

ACKNOWLEDGEMENTS

Firstly I would like to thank my Instructor Mr. Umit Ilhan and my best friends who have contributed in the preparation of my project to complete it successfully. I would also like to thanks all instructors of department of Computer Engineering for their support.

I am also grately endebted to codeproject.com administrators, when I have stack in my project, I write my problems to them, they gave their precious time to help me and giving me their ever devotion and all valuable information which I really need to complete my project.

Finally, I give my best regards to my family for providing me financial support during all my educational life and for their psychological support in all parts of my life by providing me comfort. Especially my father he resisted to send me payments during University life and encoureged me to study and complete studing.s

TABLE OF CONTENTS

ABSTRACT

The Adventory Control System is a program that solves problems that made by hand manually but by the way of this program we can do it by machine in faster, easy and more reliable way.

The program provides, manage and take hold of business transactions' record, personnel records, company records, stock records, customer records, at a small or medium computer store. A scheduled user manual prepared for helping the users to select a suitable action.

I took this project because, firms sold electronic products especially computer products that company need a software to use in their workplace, when they work manual works it takes a lot of time and needes more worker. Therefore I decided to make a program to satisfy their software needs. If there be an oportunuty for me to market and sale this program, I will devolop my project and then sale this sofware to the purchaser.

The Adventory Control System have two different accesing module, default access is trial 30 days version. This version devoid some properties, in trial program user cannot access the extras forms. This form shows the extra summary information with datagrid. If the user enter the serial number on the registration page to deserve these best properties. Registration controls with windows registry systems. The Adventory Control System aim's to help company manager, company personnel and customers, the system provides easy, quick and more reliable process on company works.

INTRODUCTION

The companies were doing their processes manually, such as hold record of customers and suppliers, stock controlling, accounting, and other business transactions. This is unreliable and waste workers time. But recently IT (Information Technology) started to help companies or firms. Then it has been very popular because it is faster, cheaper than manually and so easy work with IT.

It is necessary to companies work with computer programs in their work to more valuable present and more efficiently working. Managers, workers and customers feel their self they are really in a technological company when they work or being in a program like this.

At this point Computer Stock Control System will provide easiness and quickness of company processes that are customer records, purchased and sold products, stock controlling, accounting records, personnel records. Also system has a authorization steps to determine and control the access levels, to use this feature every user have different authorization levels that given by managers with username and password.

I prepared this program by using Visual Basic .Net programming language and Database was established by using Microsoft Sql Server 2000 that connected with Ado .net program. Which includes every .net platform.

This system designed for using at Windows Operating System. This program have an easily setup package. This setup package created in Vb.net setup project application.

CHAPTER 1

1. Visual Basic .Net

1.1 Introduction

With its release for the .NET platform, the Visual Basic language has undergone dramatic changes. For example:

- The language itself is now fully object-oriented.
- Applications and components written in Visual Basic .NET have full access to the .NET Framework, an extensive class library that provides system and application services.
- All applications developed using Visual Basic .NET run within a managed runtime environment, the .NET common language runtime. In this introduction, I briefly discuss these changes and other changes before showing you three very simple, but complete, Visual Basic .NET applications.

1.1.1 What Is Visual Basic .NET?

Visual Basic .NET is the next generation of Visual Basic, but it is also a significant departure from previous generations. Experienced Visual Basic 6 developers will feel comfortable with Visual Basic .NET code and will recognize most of its constructs. However, Microsoft has made some changes to make Visual Basic .NET a better language and an equal player in the .NET world. These include such additions as a Class keyword for defining classes and an Inherits keyword for object inheritance, among others. Visual Basic 6 code can't be compiled by the Visual Basic .NET compiler without significant modification.

1.1.2 Why Should You Move to Visual Basic.NET?

One of the most common questions today is, "Why should I move to .NET?" .NET is new, and there are many questions about what it can do for you. From a Visual Basic standpoint, it's important to understand some of the dramatic benefits that can be achieved by moving to VB.NET.

1.1.3 What Can We Do with VB .NET?

With its language enhancements and its tight integration into the .NET Framework, Visual Basic is a thoroughly modernized language that will likely become the premier development tool for creating a wide range of .NET applications. In the past, Visual Basic was often seen as a "lightweight" language that could be used for particular kinds of tasks, but was wholly unsuitable for others. (It was often argued, sometimes incorrectly, that you couldn't create such things as Windows dynamic link libraries or shell extensions using Visual Basic.) In the .NET Framework, VB .NET emerges as an equal player.

Microsoft's claim of language independence?that programming language should be a lifestyle choice, rather than a choice forced on the developer by the character of a project?is realized in the .NET platform. This means that VB .NET can be used to create a wide range of applications and components, including the following:

Windows console mode applications
Standard Windows applications
Windows services
Windows controls and Windows control libraries
Web (ASP.NET) applications
Web services
Web controls and web control libraries
.NET classes and namespaces
Accessing application object models

Microsoft Office suite) using COM automation Most importantly, for the first time with the release of VB .NET, Visual Basic becomes an all-purpose development environment for building Internet applications, an area in which it has traditionally been weak. This means that the release of this newest version should revitalize Visual Basic, allowing it to remain the tool of choice for developing state-of-the-art software for the next generation of software development.

1.2 The Visual Basic .NET Language

This chapter discusses the syntax of the Visual Basic .NET language, including basic concepts such as variables, operators, statements, classes, etc. Some material that you'd expect to find in this chapter will seem to be missing. For example, mathematical functions, file I/O, and form declarations are all very much a part of developing Visual Basic .NET applications, yet they are not introduced in this chapter because they are not intrinsic to the Visual Basic .NET language. They are provided by the .NET Framework and will be discussed in subsequent chapters. Additionally, Visual Basic .NET functions that exist merely for backward compatibility with Visual Basic 6 are not documented in this chapter.

1.2.1 Source Files

Visual Basic .NET source code is saved in files with a .vb extension. The exception to this rule is when Visual Basic .NET code is embedded in ASP.NET web page files. Such files have an .aspx extension. Source files are plain-text files that can be created and edited with any text editor, including our old friend, Notepad. Source code can be broken into as many or as few files as desired. When you use Visual Studio .NET, source files are listed in the Solution Explorer window, and all source is included from these files when the solution is built. When you are compiling from the command line, all source files must appear as command-line arguments to the compile command. The location of declarations within source files is unimportant. As long as all referenced declarations appear somewhere in a source file being compiled, they will be found. Unlike previous versions of Visual Basic, no special file extensions are used to indicate various language constructs (e.g., .cls for classes, .frm for forms, etc.). Syntax has been

added to the language to differentiate various constructs. In addition, the pseudolanguage for specifying the graphical layout of forms has been removed. Form layout is specified by setting properties of form objects explicitly within code. Either this code can be written manually, or the WYSIWYG form designer in Visual Studio .NET can write it.

1.2.2 Identifiers

Identifiers are names given to namespaces (discussed later in this chapter), types (enumerations, structures, classes, standard modules, interfaces, and delegates), type members (methods, constructors, events, constants, fields, and properties), and variables. Identifiers must begin with either an alphabetic or underscore character (_), may be of any length, and after the first character must consist of only alphanumeric and underscore characters. Namespace declarations may be declared either with identifiers or qualified identifiers. Qualified identifiers consist of two or more identifiers connected with the dot character (.). Only namespace declarations may use qualified identifiers.

Consider this code fragment:

```
Imports System
Namespace ORelly.ProgVBNet
Public Class Hello
Public Shared Sub Neareast( )
Console.WriteLine("Near, East")
End Sub
End Class
End Namespace
```

1.2.3 Keywords

Keywords are words with special meaning in a programming language. In Visual Basic .NET, keywords are reserved; that is, they cannot be used as tokens for such purposes as naming variables and subroutines

1.2.4 Literals

Literals are representations of values within the text of a program. For example, in the following line of code, 10 is a literal, but x and y are not:

```
x = y * 10
```

Literals have data types just as variables do. The 10 in this code fragment is interpreted by the compiler as type Integer because it is an integer that falls within the range of the Integer type.

1.2.5 Numeric Literals

Any integer literal that is within the range of the Integer type (-2147483648 through 2147483647) is interpreted as type Integer, even if the value is small enough to be interpreted as type Byte or Short. Integer literals that are outside the Integer range but are within the range of the Long type (-9223372036854775808 through 9223372036854775807) are interpreted as type Long. Integer literals outside the Long range cause a compile-time error. Numeric literals can also be of one of the floating point types—Single, Double, and Decimal. For example, in this line of code, 3.14 is a literal of type Double: z = y * 3.14

In the absence of an explicit indication of type (discussed shortly), Visual Basic .NET interprets floating point literals as type Double. If the literal is outside the range of the Double type (- 1.7976931348623157E308 through 1.7976931348623157E308), a compile-time error occurs. Visual Basic .NET allows programmers to explicitly specify the types of literals. Table 2-2 (shown later in this chapter) lists Visual Basic .NET's intrinsic data types, along with the method for explicitly defining a literal of each type. Note that for some intrinsic types, there is no way to write a literal.

1.2.6 String Literals

Literals of type String consist of characters enclosed within quotation-mark characters. For example, in the following line of code, "hello, world" is a literal of type String:

Console.WriteLine("hello, world") String literals are not permitted to span multiple source lines. In other words, this is not permitted: ' Wrong Console.WriteLine("hello, world") To write a string literal containing quotation-mark characters, type the character twice for each time it should appear. For example: Console.WriteLine("So then Dave said, ""hello, world"".") This line produces the following output: So then Dave said, "hello, world".

1.2.7 Character Literals

Visual Basic .NET's Char type represents a single character. This is not the same as a one-character string; Strings and Chars are distinct types. Literals of type Char consist of a single character enclosed within quotation-mark characters, followed by the character c. For example, in the following code, "A"c is a literal of type Char: Dim MyChar As Char

MyChar = "A"c

To emphasize that this literal is of a different data type than a single-character string, note that this code causes a compile-time error if `Option Strict` is `On`:

' Wrong

Dim MyChar As Char

MyChar = "A"

The error is:

Option Strict On disallows implicit conversions from 'String' to 'Char'.

1.2.8 Date Literals

Literals of type Date are formed by enclosing a date/time string within number-sign characters. For example:

Dim MyDate As Date

MyDate = #11/15/2001 3:00:00 PM#

Date literals in Visual Basic .NET code must be in the format m/d/yyyy, regardless of the regional settings of the computer on which the code is written.

1.2.9 Boolean Literals

The keywords True and False are the only Boolean literals. They represent the true and false

Boolean states, respectively (of course!). For example:

```
Dim MyBoolean As Boolean
MyBoolean = True
```

1.2.10 Nothing

There is one literal that has no type: the keyword Nothing. Nothing is a special symbol that represents an uninitialized value of any type. It can be assigned to any variable and passed in any parameter. When used in place of a reference type, it represents a reference that does not reference any object. When used in place of a value type, it represents an empty value of that type. For numeric types, this is 0 or 0.0. For the String type, this is the empty string (""). For the Boolean type, this is False. For the Char type, this is the Unicode character that has a numeric code of 0. For

programmer-defined value types, Nothing represents an instance of the type that has been created but has not been assigned a value.

1.3 Fundamental Types

Visual Basic .NET has several built-in types. Each of these types is an alias for a type supplied by the .NET architecture. Because Visual Basic .NET types are equivalent to the corresponding underlying .NET-supplied types, there are no type-compatibility issues when passing arguments to components developed in other languages. In code, it makes no difference to the compiler whether types are specified using the keyword name for the type or using the underlying .NET type name. For example, the test in this code fragment succeeds:

```
Dim x As Integer
Dim y As System.Int32
If x.GetType() Is y.GetType( ) Then
Console.WriteLine("They're the same type!")
Else
Console.WriteLine("They're not the same type.")
```

End If

Boolean

The Boolean type is limited to two values: True and False. Visual Basic .NET includes many logical operators that result in a Boolean type. For example:

```
Public Shared Sub MySub(ByVal x As Integer, ByVal y As Integer)
Dim b As Boolean = x > y
' other code
End Sub ' MySub
```

The result of the greater-than operator (>) is of type Boolean. The variable b is assigned the

value True if the value in x is greater than the value in y and False if it is not. The underlying .NET type is System.Boolean.

Byte

The Byte type can hold a range of integers from 0 through 255. It represents the values that can be held in eight bits of data. The underlying .NET type is System.Byte.

Char

The Char type can hold any Unicode[1] character. The Char data type is new to Visual

Basic .NET. The underlying .NET type is System.Char.

Date

The Date type holds values that specify dates and times. The range of values is from midnight on January 1, 0001 (0001-01-01T00:00:00) through 1 second before midnight on December 31, 9999 (9999-12-31T23:59:59). The Date type contains many members for accessing, comparing, and manipulating dates and times. The underlying .NET type is System.DateTime.

Decimal

The Decimal type holds decimal numbers with a precision of 28 significant decimal digits. Its purpose is to represent and manipulate decimal numbers without the rounding errors of the Single and Double types. The Decimal type replaces Visual Basic 6's Currency type. The underlying .NET type is System.Decimal.

Double

The Double type holds a 64-bit value that conforms to IEEE standard 754 for binary floating point arithmetic. The Double type holds floating point numbers in the range

-1.7976931348623157E308 through 1.7976931348623157E308. The smallest nonnegative number (other than zero) that can be held in a Double is 4.94065645841247E-324. The underlying .NET type is System.Double. Integer

The Integer type holds integers in the range -2147483648 through 2147483647. The Visual Basic .NET Integer data type corresponds to the VB 6 Long data type. The underlying .NET type is System.Int32.

Long

The Long type holds integers in the range -9223372036854775808 through 922337203685477580. In Visual Basic .NET, Long is a 64-bit integer data type. The

underlying .NET type is System.Int64.

Object

The Object type is the base type from which all other types are derived. The Visual Basic .NET Object data type replaces the Variant in VB 6 as the universal data type. The underlying .NET type is System.Object.

Short

The Short type holds integers in the range -32768 through 32767. The Short data type corresponds to the VB 6 Integer data type. The underlying .NET type is System.Int16.

Single

The Single type holds a 32-bit value that conforms to IEEE standard 754 for binary floating

point arithmetic. The Single type holds floating point numbers in the range -3.40282347E38 through 3.40282347E38. The smallest nonnegative number (other than zero) that can be held in a Double is 1.401298E-45. The underlying .NET type is System.Single.

String

The String type holds a sequence of Unicode characters. The underlying .NET type is System.String. Of the fundamental types, Boolean, Byte, Char, Date, Decimal, Double, Integer, Long, Short, and Single (that is, all of them except Object and String) are value types. Object and String are reference types.

1.3.1 Custom Types

Visual Basic .NET provides rich syntax for extending the type system. Programmers can define both new value types and new reference types. Types declared with Visual Basic .NET's Structure and Enum statements are value types, as are all .NET Framework types that derive from System.ValueType. Reference types include Object, String, all types declared with Visual Basic .NET's Class, Interface, and Delegate statements, and all .NET Framework types that don't derive from System.ValueType.

1.3.2 Arrays

Array declarations in Visual Basic .NET are similar to those in Visual Basic 6 and other languages. For example, here is a declaration of an Integer array that has five elements:

Dim a(4) As Integer

The literal 4 in this declaration specifies the upper bound of the array. All arrays in Visual Basic .NET have a lower bound of 0, so this is a declaration of an array with five elements, having indexes 0, 1, 2, 3, and 4. The previous declaration is of a variable named a, which is of type "array of Integer." Array types implicitly inherit from the .NET Framework's Array type (defined in the System namespace) and, therefore, have access to the methods defined in that type. For example, the following code displays the lower and upper bounds of an array by calling the Array class's GetLowerBound and

GetUpperBound methods:

```
Dim a(4) As Integer
Console.WriteLine("LowerBound is " & a.GetLowerBound(0).ToString( ))
Console.WriteLine("UpperBound is " & a.GetUpperBound(0).ToString( ))
```

The output is:

```
LowerBound is 0
UpperBound is 4
```

Note that the upper bound of the array is dynamic: it can be changed by methods available in the Array type. Array elements are initialized to the default value of the element type. A type's default value is determined as follows:

- For numeric types, the default value is 0.
- For the Boolean type, the default value is False.
- For the Char type, the default value is the character whose Unicode value is 0.
- For structure types (described later in this chapter), the default value is an instance of the structure type with all of its fields set to their default values.
- For enumeration types (described later in this chapter), the default value is an instance of the enumeration type with its internal representation set to 0, which may or may not correspond to a legal value in the enumeration.
- For reference types (including String), the default value is Nothing. You can access array elements by suffixing the array name with the index of the desired element enclosed in parentheses, as shown here:

```
For i = 0 To 4
Console.WriteLine(a(i))
Next
```

Arrays can be multidimensional. Commas separate the dimensions of the array when used in declarations and when accessing elements. Here is the declaration of a three-dimensional array, where each dimension has a different size:

Dim a(5, 10, 15) As Integer As with single-dimensional arrays, array elements are initialized to their default values.

1.3.3 Namespaces

Thousands of types are defined in the .NET Framework. In addition, programmers can define new types for use in their programs. With so many types, name clashes are inevitable. To prevent name clashes, types are considered to reside inside of namespaces. Often, this fact can be ignored. For example, in Visual Basic .NET a class may be defined like this:

```
Public Class SomeClass
' ...
End Class
```

This class definition might be in a class library used by third-party customers, or it might be in the same file or the same project as the client code. The client code that uses this class might look something like this:

```
Dim x As New SomeClass( )
x.DoSomething( )
```

Now consider what happens if the third-party customer also purchases another vendor's class library, which also exposes a SomeClass class. The Visual Basic .NET compiler can't know which definition of SomeClass will be used. The client must therefore use the full name of the type, also known as its fully qualified name . Code that needs to use both types might look something like this: ' The namespace is "FooBarCorp.SuperFoo2100".

```
Dim x As New FooBarCorp.SuperFoo2100.SomeClass( )
x.DoSomething( )
' ...
' The namespace is "MegaBiz.ProductivityTools.WizardMaster".
```

```
Dim y As New MegaBiz.ProductivityTools.WizardMaster.SomeClass( )
y.DoSomethingElse( )
```

Note that a namespace name can itself contain periods (.). When looking at a fully qualified type name, everything prior to the final period is the namespace name. The name after the final period is the type name. Microsoft recommends that namespaces be named according to the format CompanyName.TechnologyName. For example, "Microsoft.VisualBasic".

1.3.4 The Namespace Statement

So how does a component developer specify a type's namespace? In Visual Basic .NET, this can be done several ways. One is to use the Namespace keyword, like this:

```
Namespace MegaBiz.ProductivityTools.WizardMaster
Public Class SomeClass
    ' ...
End Class
End Namespace
```

Note that it is permissible for different types in the same source file to have different namespaces.

A second way to provide a namespace is to use the /rootnamespace switch on the VisualBasic .NET command-line compiler. All types defined within the compiled file(s) then have the given namespace. If you're compiling in the Visual Studio .NET IDE, the root namespace is specified in the Project Property Pages dialog box, which can be reached by right-clicking the project name in the Solution Explorer window of the IDE, then choosing Properties (see Figure 2-1 for the resulting WizardMaster Property Pages dialog). By default, Visual Studio .NET sets the root namespace equal to the name of the project.

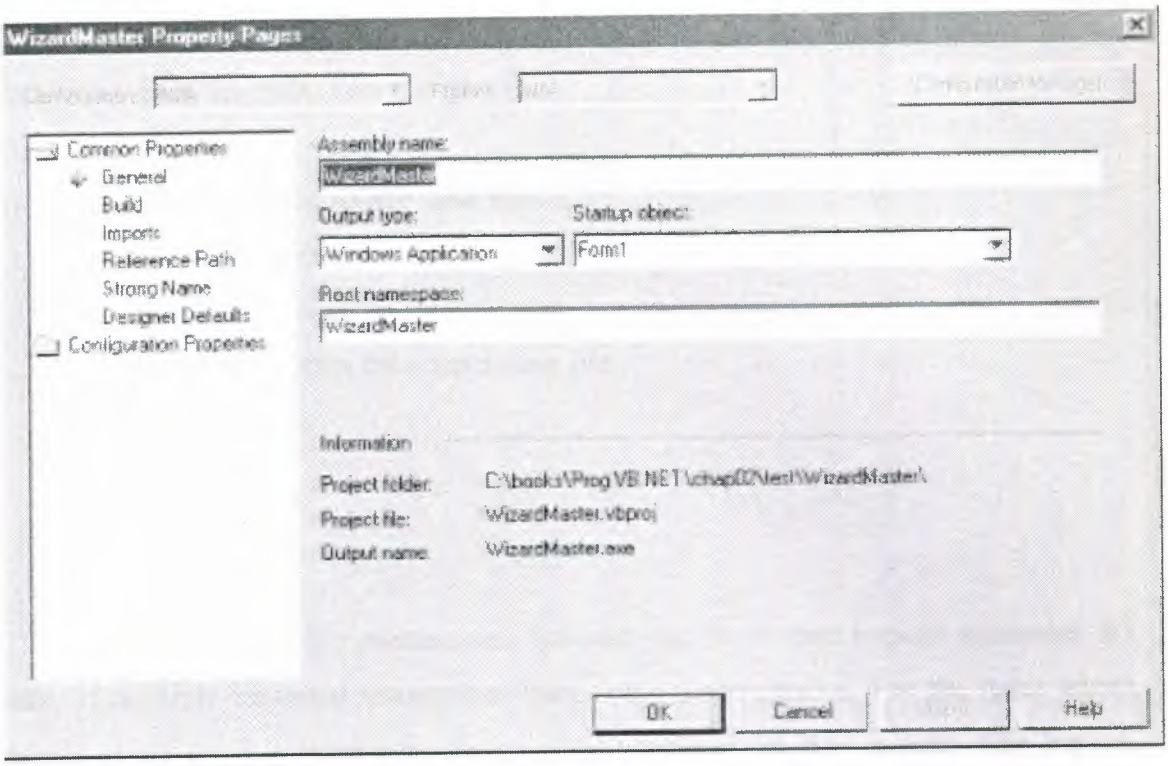

Figure 1-3. Setting the root namespace in the Visual Studio .NET

1.4 The Imports Statement

So far, the discussion has implied that it's not necessary for the user of a type to specify the type's full name unless there is a name clash. This isn't exactly true. The CLR deals with types only in terms of their full names. However, because humans don't like to deal with long names, Visual Basic .NET offers a shortcut. As an example, the .NET Framework provides a drawing library, in which a type called Point is defined. This type's namespace is called System.Drawing, so the type's fully qualified name is System.Drawing.Point. Code that uses this type might look like this:

```
Dim pt As System.Drawing.Point
pt.X = 10
pt.Y = 20
' ...
```

Typing the full name of every type whenever it is used would be too cumbersome, though, so Visual Basic .NET offers the Imports statement. This statement

indicates to the compiler that the types from a given namespace will appear without qualification in the code. For example:

```
' At the top of the source code file:
Imports System.Drawing
' ...
' Somewhere within the source code file:
Dim pt As Point
pt.X = 10
pt.Y = 20
' ...
```

To import multiple namespaces, list each one in its own Imports statement. It's okay if multiple imported namespaces have some name clashes. For the types whose names clash, the full name must be specified wherever the type is used. The Imports statement is just a convenience for the developer. It does not set a reference to the assembly in which the types are defined. See the discussion of assemblies in Chapter 3 to learn how to reference assemblies that contain the types you need.

Finally, note that namespaces, too, are just a convenience for the developer writing source code. To the runtime, a type is not "in" a namespace—a namespace is just another part of a type name. It is perfectly acceptable for any given assembly to have types in different namespaces, and more than one assembly can define types in a single namespace.

1.5 Variables

A variable is an identifier that is declared in a method and that stands for a value within that method. Its value is allowed to change within the method. Each variable is of a particular type, and that type is indicated in the declaration of the variable. For example, this line declares a variable named `i` whose type is Integer:

```
Dim i As Integer
```

The keyword `Dim` indicates a variable declaration. *Dim* is short for *dimension* and dates back to the original days of the BASIC programming language in the late 1960s. In that language, variables were not declared; they were just used where needed (except

for arrays). Because of how arrays were laid out in memory, the BASIC language interpreter had to be told of the dimensions of an array before the array was used. This was the purpose of the `Dim` statement. In later years, when declaration of all variables was agreed upon to be a *good thing*, the use of the `Dim` statement was broadened to include all variable declarations.

Variable identifiers may be suffixed with *type characters* that serve to indicate the variable's type. For example, this line declares a variable of type Integer:

```
Dim x%
```

The effect is precisely the same as for this declaration:

```
Dim x As Integer
```

The set of type characters is shown in Table 2-4; note that not all data types have a type character

1.5.1 Scope

Scope refers to the so-called *visibility* of identifiers within source code. That is, given a particular identifier declaration, the *scope* of the identifier determines where it is legal to reference that identifier in code. For example, these two functions each declare a variable `CoffeeBreaks`. Each declaration is invisible to the code in the other method. The scope of each variable is the method in which it is declared.

```
Public Sub MyFirstMethod( )
Dim CoffeeBreaks As Integer
' ...
End Sub
Public Sub MySecondMethod( )
Dim CoffeeBreaks As Long
' ...
End Sub
```

Unlike previous versions of Visual Basic, Visual Basic .NET has *block scope*. Variables declared withina set of statements ending with `End`, `Loop`, or `Next` are local to that block. For example:

```
Dim i As Integer
For i = 1 To 100
```

```
Dim j As Integer
For j = 1 To 100
' ...
Next
Next
' j is not visible here
```

Visual Basic .NET doesn't permit the same variable name to be declared at both the method level and the block level. Further, the life of the block-level variable is equal to the life of the method. This means that if the block is re-entered, the variable may contain an old value (don't count on this behavior, as it is not guaranteed and is the kind of thing that might change in future versions of Visual Basic).

1.6 Arithmetic Operators

The arithmetic operators perform the standard arithmetic operations on numeric values. The arithmetic operators supported by Visual Basic .NET are:

*** (Multiplication)**

The multiplication operator is defined for all numeric operands. The result is the product of the operands.

/ (Regular division)

The regular division operator is defined for all numeric operands. The result is the value of the first operand divided by the second operand.

\ (Integer division)

The integer division operator is defined for integer operands (Byte, Short, Integer, and Long). The result is the value of the first operand divided by the second operand, then rounded to the integer nearest to zero.

Mod (Modulo)

The modulo operator is defined for integer operands (Byte, Short, Integer, and Long). The result is the remainder after the integer division of the operands.

^ (Exponentiation)

The exponentiation operator is defined for operands of type Double. Operands of other numeric types are converted to type Double before the result is calculated. The result is the value of the first operand raised to the power of the second operand.

+ (Addition)

The addition operator is defined for all numeric operands and operands of an enumerated type. The result is the sum of the operands. For enumerated types, the sum is calculated on the underlying type, but the return type is the enumerated type. See the discussion of enumerated types in the "Enumerations" section later in this chapter for more information on the types that can underlie an enumerated type. See also Section 2.12.4 later in this section.

- (Subtraction)

The subtraction operator is defined for all numeric operands and operands of an enumerated type. The result is the value of the first operand minus the second operand. For enumerated types, the subtraction is calculated on the underlying type, but the return type is the enumerated type.

1.7 Relational Operators

The relational operators all perform some comparison between two operands and return a Boolean value indicating whether the operands satisfy the comparison. The relational operators supported by Visual Basic .NET are:

= (Equality)

The equality operator is defined for all primitive value types and all reference types. For primitive value types and for the String type, the result is `True` if the values of the operands are equal; `False` if not. For reference types other than String, the result is `True` if the references refer to the same object; `False` if not. If the operands are of type Object and they reference primitive value types, value comparison is performed rather than reference comparison.

<> (Inequality)

The inequality operator is defined for all primitive value types and for reference types. For primitive value types and for the String type, the result is `True` if the values of the operands are not equal; `False` if equal. For reference types other than String, the result is `True` if the references refer to different objects; `False` if they refer to the same object. If the operands are of type Object and they reference primitive value types, value comparison is performed rather than reference comparison.

< (Less than)

The less-than operator is defined for all numeric operands and operands of an enumerated type. The result is `True` if the first operand is less than the second; `False` if not. For enumerated types, the comparison is performed on the underlying type.

> (Greater than)

The greater-than operator is defined for all numeric operands and operands that are of an enumerated type. The result is `True` if the first operand is greater than the second; `False` if not. For enumerated types, the comparison is performed on the underlying type.

<= (lLess than or equal to)

The less-than-or-equal-to operator is defined for all numeric operands and operands of an enumerated type. The result is `True` if the first operand is less than or equal to the second operand; False if not.

>= (Greater than or equal to)

The greater-than-or-equal-to operator is defined for all numeric operands and operands of an enumerated type. The result is `True` if the first operand is greater than or equal to the second operand; `False` if not. TypeOf...Is The `TypeOf...Is` operator is defined to take a reference as its first parameter and the name of a type as its second parameter. The result is `True` if the reference refers to an object that is type-compatible with the given type-name; False if the reference is Nothing or if it refers to an object that is not type-compatible with the given type name.

Use the TypeOf...Is operator to determine whether a given object: Is an instance of a given class Is an instance of a class that is derived from a given class Exposes a given interface In any of these cases, the TypeOf expression returns True.Is

The Is operator is defined for all reference types. The result is True if the references refer to the same object; False if not.

Like

The Like operator is defined only for operands of type String. The result is True if the first operand matches the pattern given in the second operand; False if not. The rules for matching are: The ? (question mark) character matches any single character. The * (asterisk) character matches zero or more characters. The # (number sign) character matches any single digit.

A sequence of characters within [] (square brackets) matches any single character in the sequence.

Within such a bracketed list, two characters separated by a - (hyphen) signify a range of Unicode characters, starting with the first character and ending with the second character. A - character itself can be matched by placing it at the beginning or end of the bracketed sequence.

Preceding the sequence of characters with an `!` (exclamation mark) character matches any single character that does not appear in the sequence.

- The `?`, `*`, `#`, and `[` characters can be matched by placing them within `[]` in the pattern string. Consequently, they cannot be used in their wildcard sense within `[]`.
- The `]` character does not need to be escaped to be explicitly matched. However, it can't be used within `[]`.

1.8 Logical Operators

Logical operators are operators that require Boolean operands. They are:

And

The result is `True` if and only if both of the operands are `True`; otherwise, the result is `False`.

Or

The result is `True` if either or both of the operands is `True`; otherwise, the result is False.

The result is `True` if one and only one of the operands is `True`; otherwise, the result is `False`.

Not

This is a unary operator. The result is `True` if the operand is `False`; `False` if the operand is True.

1.9 Classes

Most Visual Basic developers are familiar with classes. Classes are definitions or blueprints of objects that will be created at runtime. Classes define the properties, methods, fields, and events of objects. If the term fields is new to you, it simply means public variables exposed by the class; fields are the "lazy way" to do properties. Together, properties, methods, fields, and events are generically called members of the class. If a class has one or more methods that do not contain any implementation, the class is said to be abstract. In VB.NET, you cannot instantiate abstract classes directly;

Instead, you must inherit from them. In VB6, it was possible to create a class that was just method definitions and then to use the Implements keyword to inherit the interface. You could actually instantiate the interface in VB6, but because it did not have any implementation code, there was no point in doing so. In VB.NET, you can create a class that has implementation code instead of just the interface, and then mark the class as abstract. Now, other classes can inherit from that abstract class and use the implementation in it or override the implementation as needed. These are new concepts to VB developers. In the past, VB had only interface inheritance, but VB.NET has "real" inheritance, known as implementation inheritance.

In VB.NET, interfaces are separate from classes. In VB6, you created interfaces by creating classes with method definitions, but no implementation code inside those methods. You will see more on interfaces in the next section, but realize that although a VB.NET class can implement any number of interfaces, it can inherit from only one base class. This will be examined in more detail throughout the book. Classes have a number of possible characteristics that can be set, and that are stored in the metadata. In addition, members can have characteristics. These characteristics include such items as whether or not the class or member is inheritable.

1.9.1 A class definition

```
Public Class Employee
Public EmployeeNumber As Integer
Public FamilyName As String
Public GivenName As String
```

```
Public DateOfBirth As Date
Public Salary As Decimal
Public Function Format( ) As String
Return GivenName & " " & FamilyName
End Function
End Class
```

Example Using a class

```
Dim emp As New Employee( )
emp.EmployeeNumber = 10
emp.FamilyName = "Rodriguez"
emp.GivenName = "Celia"
emp.DateOfBirth = #1/28/1965#
emp.Salary = 115000
Console.WriteLine("Employee Name: " & emp.Format( ))
Console.WriteLine("Employee Number: " & emp.EmployeeNumber)
Console.WriteLine("Date of Birth: " & emp.DateOfBirth.ToString("D",
Nothing))
Console.WriteLine("Salary: " & emp.Salary.ToString("C", Nothing)
```

1.10 Interfaces

Interfaces in VB.NET are like the interfaces in previous versions of VB: They are definitions of a class without the actual implementation. Because there is no implementation code, you cannot instantiate an interface, but must instead implement it in a class. There is one exception to the "no implementation code in an interface" rule: In VB.NET, you can define what are called *static* members.

1.11 Inheritance

Visual Basic allows true inheritance of objects. So, what does this really mean? *Inheritance* is a relationship where one object is derived from another object.When an object is inherited, all of its properties and methods are automatically included in the

new object. Let's take our car example a little further and create a new object for a specific type of vehicle, a truck:

Truck Object

Inherits Vehicle Object

BedLength

End Vehicle Object

This new truck object will now have the same properties as the vehicle object (number of wheels and doors as well as color), but we have now added how long the truck bed is without have to recreate the other properties. As you can see, this can be a powerful tool for code reuse.You don't have to rewrite code just to tweak it to your specific needs.

1.12 Method parameters

Methods can be defined to take arguments. As already shown, method definitions can take an optional parameter list. A parameter list looks like this: parameter { , parameter }

1.13 What About On Error?

Visual Basic 6 did not have exception objects and Try...Catch blocks. Instead, it used the On Error statement to specify a line within the current procedure to which execution should jump if an error occurred. The code at that point in the procedure could then examine the Err intrinsic object to determine the error that had occurred. For compatibility with previous versions, Visual Basic .NET continues to support the On Error and related statements, but they should not be used in new development, for the following reasons:

- Structured exception handling is more flexible.
- Structured exception handling does not use error codes. (Applicationdefined error codes often clashed with error codes defined by other applications.)
- Structured exception handling exists at the .NET Framework level, meaning that regardless of the language in which each component is written, exceptions can be thrown and caught across component boundaries.

1.14 Events

An *event* is a callback mechanism. With it, objects can notify users that something interesting has happened. If desired, data can be passed from the object to the client as part of the notification. Throughout this section, I use the terms *event producer*, *producer class*, and *producer object* to talk about a class (and its instances) capable of raising events. I use the terms *event consumer*, *consumer class*, and *consumer object* to talk about a class (and its instances) capable of receiving and acting on events raised by an event producer.

Chapter 2

2.1 History of Object Orientation and VB

Visual Basic has been best described as an *object-based* language, rather than an *object-oriented* one, because it did not support true inheritance from one object to another. Programmers have used different methods to simulate Inheritance since VB 5.0, specifically by using the *Implements* interface. Although this feature didn't actually bring functionality of a parent class, at least it defined a set of methods that would need to be coded. However, there was not an effective way to reuse business logic.This was a clumsy workaround, at best, and is far inferior to the overriding and overloading that are now available.

2.1.1 Object-Oriented Language

Previous versions of Visual Basic did not offer true object-oriented inheritance of code from a parent class to a child class. In VB.NET, propagating code from one module to another is now possible, while only overriding the behavior that needs changed in the child class, thus improving maintainability.

Because of the CLR, not only can a VB developer inherit a class from another VB module, he can also inherit from a module developed in another language, such as C#.

2.1.2 Why Learn Object-Oriented Techniques?

As you may know, Visual Basic has implemented some features of object-oriented programming since Version 4. However, in terms of object-orientation, the move from Version 6 to VB .NET has been dramatic. Many people did not consider VB 6 (or earlier versions) to be a truly object-oriented programming language. Whatever your thoughts may have been on this matter, it seems clear that VB .NET is an object-oriented programming language by any reasonable definition of the term. You may be saying to yourself: "I prefer not to use object-oriented techniques in my programming." This is something you could easily have gotten away with in VB 6. But in VB .NET, the structure of the .NET Framework specifically the .NET Base Class Library as well as the documentation, is so object-oriented that you can no longer avoid understanding the basics of object-orientation, even if you decide not to use them in your applications.

2.1.3 Principles of Object-Oriented Programming

It is often said that there are four main concepts in the area of object-oriented programming:

- Abstraction
- Encapsulation
- Inheritance
- Polymorphism

Each of these concepts plays a significant role in VB .NET programming at one level or another. Encapsulation and abstraction are "abstract" concepts providing motivation for object-oriented programming. Inheritance and polymorphism are concepts that are directly implemented in VB .NET programming.

2.1.4 Advantages of Object-Oriented Design

The true advantages to object-oriented design come when you can propagate behavior from one object to another. For example, if you were developing a sedan and a coupe, you might design few differences between the two cars other than the number of doors (four versus two). This is where inheritance comes in. If you already had a sedan designed, you could build a coupe just by inheriting all of the behavior of the sedan, except for overriding the number of doors. Observe the following VB pseudocode:

```
Public Class Coupe
Inherits Sedan
Overrides Sub BuildDoors()
Doors = Doors + 2
End Sub
End Class
```

Now, if you add new features to the sedan (such as side air bags, for example), they are automatically propagated to the coupe without adding any additional code. By contrast, overloading is when you want the methods of a single object to have different behaviors depending upon what parameters you pass to it.Then, VB is smart enough to determine which module to run depending upon the parameter list.

Type	Overriding	Overloading
Method Name	Same	Same
Argument List	Same	Different
Behavior	Replaces existing method	Supplements existing method

Table 2.1.4 Overriding versus Overloading

Chapter3

3.1 Major VB.NET Changes

VB.NET introduces major changes to the VB language. Some are modifications to existing ways of working, whereas others are brand new. This chapter will cover some of those changes, but this is by no means an exhaustive list of all changes from VB to VB.NET. First, you'll see some of the features that have changed. Then you will see some of the new features.

3.1.1 General Changes

There are a number of general changes to be aware of when moving from VB to VB.NET. Among them are topics such as the removal of default properties, subs and functions requiring parentheses, ByVal being the default method for passing parameters, and changes to the logical operators. These changes, and others, are detailed in this section.

3.1.2 Subs and Functions Require Parentheses

As you saw in the last chapter when you used the MsgBox function, you must now always use parentheses with functions, even if you are ignoring the return value. In addition, you must use parentheses when calling subs, which you did not do in VB6. For example, assume that you have this sub in both VB6 and VB.NET:

```
Sub foo(ByVal Greeting As String)
' implementation code here
End Sub
```

In VB6, you could call this sub in one of two ways:

```
foo "Hello"
Call foo("Hello")
```

In VB.NET, you also could call this sub in one of two ways:

```
Foo("Hello")
Call foo("Hello")
```

The difference, of course, is that the parentheses are always required in the VB.NET calls, even though you aren't returning anything. The Call statement is still supported, but it is not really necessary.

3.1.3 Changes to Boolean Operators

The And, Not, and Or operators were to have undergone some changes. Microsoft originally said that the operators would short-circuit, but now they are staying the way they worked in VB6. This means that in VB.NET, as in VB6, if you had two parts of an And statement and the first failed, VB6 still examined the second part. Examine the following code:

```
Dim x As Integer
Dim y As Integer
x = 1
y = 0
If x = 2 And y = 5/y Then
...
```

As a human, you know that the variable x is equal to 1. Therefore, when you look at the first part of the If statement, you know that x is not equal to 2, so you would logically think it should quit evaluating the expression. However, VB.NET examines the second part of the expression, so this code would cause a divide-by-zero error. If you want short-circuiting, VB.NET has introduced a couple of new operators: AndAlso and OrElse. In this case, the following code would not generate an error in VB.NET:

```
Dim x As Integer
Dim y As Integer
x = 1
y = 0
If x = 2 AndAlso y = 5/y Then
...
```

This code does not cause an error; instead, because x is not equal to 2, VB.NET does not even examine the second condition.

3.1.4 Declaration Changes

We can now initialize your variables when you declare them. You could not do this in VB6. In VB6, the only way to initialize a new variable was to do so on a separate line, like this:

```
Dim x As Integer
x = 5
```

In VB.NET, you can rewrite this into one line of code:

```
Dim x As Integer = 5
```

Another significant, and much-requested, change is that of declaring multiple variables, and what data type they assume, on one line. For example, you might have the following line:

```
Dim x, y As Integer
```

As you're probably aware, in VB6, y would be an Integer data type, but x would be a Variant. In VB.NET, this has changed, so both x and y are Integers. If you think, "It's about time," there are many who agree. This should remove a number of bugs and strange type conversions experienced by new VB developers. It should also make the code more efficient by making variables the expected type instead of using the Object type

3.1.5 Support for New Assignment Operators

VB.NET now supports shortcuts for performing certain assignment operations. In VB6, you incremented x by 1 with the following line of code:

```
x = x + 1
```

In VB.NET, you can type an equivalent statement like this:

```
x += 1
```

Not only can you use the plus sign, but VB.NET now also supports -=, *=, /=, \=, and ^= from a mathematical standpoint, and &= for string concatenation. If all this looks like C/C++, that's where it came from. However, the ++ operator is not supported.

Microsoft made a decision not to include the ++ operator because they felt it made the code more difficult to read.

Because VB.NET is in beta and has not yet been performance tuned, it is unclear whether these new assignment operators will be more efficient. These operators did tend to be more efficient in C/C++, due to a more efficient use of the CPU's registers. Therefore, it will be interesting to test these new operators when the final, tuned version of VB.NET is released.

3.1.6 ByVal Is Now the Default

In what many consider a strange decision, the default way to pass parameters in VB has always been by reference. The decision was actually made because passing by reference is faster within the same application, but can be costly if you are calling components across process boundaries. If you're a little rusty, *by reference* means that you are passing only the address of a variable into the called routine. If the called routine modifies the variable, it actually just updates the value in that memory location, and therefore the variable in the calling routine also changes.

```
Private Sub Command1_Click()
Dim x As Integer
x = 3
foo x
MsgBox x
End Sub
Sub foo(y As Integer)
y = 5
End Sub
```

The message box shows the value 5. That happens because in VB6, when you pass x to foo, you are just sending the memory address of the variable x, so when foo modifies y to 5, it is changing the value in the same memory location to which x points, and this causes the value of x to change as well. If you tried to type this example into VB.NET, you'd see something happen. First, of course, you'd have to add parentheses around x in your call to foo. However, when you tried to type the definition of foo, VB.NET would automatically add the word ByVal into the definition, so it would end up looking like this: Sub foo(ByVal y As Integer) If you wanted to pass by reference,

you would have to add the ByRef keyword yourself, instead of VB.NET using the new default of ByVal. This is a benefit to those of you calling procedures across process boundaries, something that is common in the world of distributed applications. In addition, this should cut down on errors like those seen by novice users who didn't understand the concept of passing by reference in previous versions of VB.

3.1.7 Block-Level Scope

VB.NET adds the ability to create variables that are visible only within a block. A block is any section of code that ends with one of the words End, Loop, or Next. This means that For...Next and If...End If blocks can have their own variables. Take a look at the following code:

```
While y < 5
Dim z As Integer
End While
```

The variable z is now visible only within the While loop. It is important to realize that although z is visible only inside the While loop, its lifetime is that of the procedure. That means if you re-enter the While statement, z will have the same value that it did when you left. Therefore, it is said that the scope of z is block level, but its lifetime is procedure level.

3.1.8 While...Wend Becomes While...End While

The While loop is still supported, but the closing of the loop is now End While instead of Wend. If you type Wend, the editor automatically changes it to End While. This change finally move the While loop into synch with most other VB "block" structures, which all end with an End <block> syntax.

3.2 Procedure Changes

VB.NET has changes that affect how you define and work with procedures. Some of those changes are mentioned in this section.

3.2.1 Optional Arguments Require a Default Value

In VB6, you could create an optional argument (or several optional arguments) when you defined a procedure. You could, optionally, give them a default value. That way, if someone chose not to pass in a value for an argument, you had a value in it. If you did not set a default value and the caller did not pass in a value, the only way you had to check was to call the IsMissing statement. IsMissing is no longer supported because VB.NET will not let you create an optional argument that does not have a default value. IsMissing is not needed because an optional argument is guaranteed to have a value. For example, your declaration might look like this:

```
Sub foo(Optional ByVal y As Integer = 1)
```

Notice that the Optional keyword is shown, just as it was in VB6. This means a parameter does not have to be passed in. However, if it is not passed in, y is given the default value of 1. If the calling routine does pass in a value, of course, y is set to whatever is passed in.

3.2.2 Static Not Supported on Subs or Functions

In VB6, you could put `Static` in the declaration of a sub or function. Doing so made every variable in that sub or function *static*, which meant that they retained their values between calls to the procedure. For example, this was legal in VB6:

```
Static Sub foo()
Dim x As Integer
Dim y As Integer
x = x + 1
y = y + 2
End Sub
```

In this example, `x` will retain its value between calls. So, the second time this procedure is called, `x` already has a value of 1, and the value of `x` will be incremented to

2. The variable y would have the value of 2, and therefore the second time in it would be incremented to 4.

VB.NET does not support the Static keyword in front of the sub or function declaration anymore, as you saw in the example above. In fact, if you want an entire sub or function to be static, you need to place Static in front of each variable for which you want to preserve the value. This is the only way to preserve values in variables inside a sub or function. In VB.NET, the equivalent sub would look like this:

```
Sub foo()
Static x As Integer
Static y As Integer
x = x + 1
y = y + 2
End Sub
```

3.3 Array Changes

Arrays have undergone some changes as well. Arrays could be somewhat confusing in previous versions of VB. VB.NET seeks to address any confusion by simplifying the rules and removing the capability to have nonzero lower boundaries.

In VB6, if you left the default for arrays to start at 0, declaring an array actually gave you the upper boundary of the array, not the number of elements. For example, examine the following code:

```
Dim y(2) As Integer
y(0) = 1
y(1) = 2
y(2) = 3
```

In this VB6 code, you declare that y is an array of type Integer, and the upper boundary is 2. That means you actually have three elements: 0–2. You can verify this by setting those three elements in code. In VB.NET, array declaration was going to change, so that the parameter was the number of elements. However, due to the possibility of breaking existing code in the upgrade process, this has been changed back to the way it worked in VB.

3.3.1 Lower Boundary Is Always Zero

VB6 allowed you to have a nonzero lower boundary in your arrays in a couple of ways. First, you could declare an array to have a certain range. If you wanted an array to start with 1, you declared it like this:

Dim y(1 To 3) As Integer

This would create an array with three elements, indexed 1–3. If you didn't like this method, you could use Option Base, which allowed you to set the default lower boundary to either 0 (the default) or 1. VB.NET removes those two options from you. You cannot use the 1 to x syntax, and Option Base is no longer supported. In fact, because the lower boundary of the array is always 0, the Lbound function is no longer supported.

3.4 Data Type Changes

There are several changes to data types that are important to point out. These changes can have an impact on the performance and resource utilization of your code. The data types in VB.NET correspond to the data types in the System namespace, which is important for cross-language interoperability.

3.4.1 All Variables Are Objects

Technically, in VB.NET, all variables are subclassed from the Object base class. This means that you can treat all variables as objects. For example, to find the length of a string, you could use the following code:

```
Dim x As String
x = "Hello, World"
MsgBox(x.Length)
```

This means that you are treating x as an object, and examining its Length property. Other variables have other properties or methods

3.4.2 Automatic String/Numeric Conversion Not Supported

In VB6, it was easy to convert from numbers to strings and vice versa. For example,

examine this block of code:

```
Dim x As Integer
Dim y As String
x = 5
y = x
```

In VB6, there is nothing wrong with this code. VB will take the value 5 and automatically convert it into the string "5". VB.NET, however, disallows this type of conversion by default. Instead, you would have to use the CStr function to convert a number to a string, or the Val function to convert a string to a number. You could rewrite the preceding code for VB.NET in this manner:

```
Dim x As Integer
Dim y As String
x = 5
y = CStr(x)
y = x.ToString ' This is equivalent to the previous line
```

3.4.3 Fixed-Length Strings Not Supported

In VB6, you could declare a fixed-length string by using a declaration like the one shown here:

```
Dim y As String * 30
```

This declared y to be a fixed-length string that could hold 30 characters. If you try this same code in VB.NET, you will get an error. All strings in VB.NET are variable length.

3.5 Structured Error Handling

Error handling has changed in VB.NET. Actually, the old syntax still works, but there is a new error handling structure called Try...Catch...Finally that removes the need to use the old On Error Goto structure. The overall structure of the Try...Catch...Finally syntax is to put the code that might cause an error in the Try portion, and then catch the error. Inside the Catch portion, you handle the error. The Finally portion runs code that happens after the

Catch statements are done, regardless of whether or not there was an error. Here is a simple example:

```
Dim x, y As Integer ' Both will be integers
Try
x \= y ' cause division by zero
Catch ex As Exception
msgbox(ex.Message)
End Try
```

Here, you have two variables that are both integers. You attempted to divide x by y, but because y has not been initialized, it defaults to 0. That division by zero raises an error, and you catch it in the next line. The variable ex is of type Exception, which holds the error that just occurred, so you simply print the Message property, much like you printed Err.Description in VB6. In fact, you can still use Err.Description, and the Err object in general. The Err object will pick up any exceptions that are thrown. For example, assume that your logic dictates that an error must be raised if someone's account balance falls too low, and another error is raised if the balance drops into the negative category. Examine the following code:

```
Try
If bal < 0 Then
Throw New Exception("Balance is negative!")
ElseIf bal > 0 And bal <= 10000 Then
Throw New Exception("Balance is low; charge interest")
End If
Catch
MessageBox.Show("Error: " & Err().Description)
Finally
MessageBox.Show("Executing finally block.")
End Try
```

Chapter 4

4. SQL Server 2000 Overview

4.1 Introduction

Microsoft SQL Server 2000 is more than a relational database management system; it is a complete database and analysis product that meets the scalability and reliability requirements of the most demanding enterprises. It is appropriate for a broad range of solution types, including e-commerce, data warehousing, and line-of-business applications. Of course, SQL Server 2000 contains many features that help businesses manage and analyze data, but one "feature" that might not be so obvious is the selection of SQL Server 2000 editions.

There are seven different editions to choose from. That might seem like a lot of different products to worry about, but understanding the differences and appropriate uses for these various editions is actually quite simple. The different editions are designed to accommodate the unique performance, runtime, and price requirements of organizations and individuals. For example, your organization may require not only that its database and analysis solution run on the largest, most powerful computers in your company's data center, but also that this solution be able to "scale down" to desktops, laptops, and even devices like the Pocket PC. SQL Server 2000 achieves this goal while maintaining maximum application compatibility across platforms.

Understanding these options allows organizations to make the most costeffective and technically appropriate choice for their particular needs. In this paper, you'll learn more specifically about the differences among the various editions of SQL Server 2000, and how you can save time and money by choosing the right one for the job.

4.2 SQL Server 2000 Editions for Special Uses

Besides the two server editions of SQL Server 2000, five editions exist for special uses. These are:

- SQL Server 2000 Personal Edition
- SQL Server 2000 Developer Edition
- SQL Server 2000 Evaluation Edition (also known as SQL Server 2000 Enterprise
- Evaluation Edition)

- SQL Server 2000 Windows CE Edition
- SQL Server 2000 Desktop Engine (also known as MSDE)

4.3 New and Enhanced Features of SQL Server 2000

SQL Server 2000 delivers a more mature RDBMS from Microsoft. The first release since the near-entire redesign of SQL Server that resulted in version 7.0, SQL Server 2000 builds on that version and the feedback that resulted in two service packs of enhancements and fixes to the SQL Server architecture. This latest release offers enhanced reliability, scalability, programmability, and services for SQL programmers and application developers. Delivering key new features allows

SQL Server to meet the demands of large-scale enterprise applications, including online transaction processing (OLTP), data warehousing, and electronic commerce, in which SQL Server continues to grow in market dominance. As a member of Microsoft's .NET Enterprise Server family, SQL Server 2000 provides native support for XML as well as standard Internet protocols such as HTTP and SSL. Numerous productivity enhancements are welcome additions for SQL programmers, including new data types, trigger enhancements, user-defined functions, and a supercharged Query Analyzer that includes a built-in debugger that doesn't require godlike talents to configure! SQL Server 7.0's OLAP Services have "grown up" and been renamed Analysis Services, offering OLAP and datamining capabilities native to SQL Server 2000. Having been designed for Windows 2000, SQL Server 2000 increases its scalability and availability levels, taking advantage of four-way fail-over clustering and support for up to 64GB of memory. A popular scalability enhancement in SQL Server 2000 is distributed partitioned views, which has allowed SQL Server to take over the Transaction Processing Council (TPC) leadership role in terms of price/performance measures and tremendously surpassing its rival, Oracle 8i, in scalability. Sharing and exchanging data are common tasks in distributed application environments, and SQL Server 2000 shines in this area. Replication enhancements in SQL Server 2000 allow for queued updating subscribers and easier setup and management of replication solutions. Since version 7.0 of SQL Server, Data Transformation Services (DTS) have enjoyed many new fans, and this latest release adds a few of the "missing" pieces of the previous version. If all this isn't enough to get you excited about this latest version of SQL Server, in the following sections we review the entire list of enhancements and additions to SQL Server 2000.

4.3.1 XML Support

XML, a subset of Standard Generalized Markup Language (SGML), is the latest addition to the arsenal of technologies available for building software solutions. Although not a new concept, the XML 1.0 recommendation was submitted to the World Wide Web Consortium (W3C) in 1998, many developers are finding it a strong and functional tool for communicating between heterogeneous systems and across the Internet. Microsoft has made such a strong commitment to XML that the company is touting it as a core technology component in its .NET architecture and services model. As its name implies, XML can be as simple or as complex as it needs to be to cater to a given situation, and its HTML-like format makes it easy for new XML developers to quickly begin being productive with the technology. Soon after Microsoft released SQL Server 7.0, the XML Technology Preview for SQL Server was released, providing insight into, and a draft of, what is now native XML support in SQL Server 2000. SQL Server 2000 offers native support for reading, writing, delivering, and using XML *documents*, the term for complete sets of XML tags and data. The following additions to SQL Server, along with the latest version of ActiveX Data Objects, version 2.6, provide complete support for using XML in your SQL Server-based applications:

4.3.2 User-Defined Functions

A significant addition to the programmability of SQL Server is the new User-Defined Function (UDF) object. We use dozens of the built-in SQL functions to manipulate and process data, define computed columns, and control logic flow in all our SQL applications, but now SQL development is no longer restricted to that list of predefined SQL functions from Microsoft. With support for returning single values such as integer or character data and the ability to return the new table data type, UDFs in SQL Server will increase developer productivity and code reuse in SQL applications.

The process of creating UDFs is similar to creating other objects in SQL Server. The CREATE FUNCTION statement allows you to define input parameters and the return parameter type of your custom function. The following example creates a new function to return a table data type that contains the product name and unit price from the Products table in the sample North-wind database for all products that have a "units

in stock" greater than the input parameter—essentially, a parameterized view. This function can be used in the FROM clause of a T-SQL statement as a rowset:

```
CREATE FUNCTION GetProductsAvail ( @intStockQty int )
RETURNS @ProductList TABLE
(
ProductName nvarchar(40),
UnitPrice money
)
AS
BEGIN
INSERT INTO @ProductList
SELECT productname, unitprice
FROM Products
WHERE UnitsInStock > @intStockQty
RETURN
END
```

4.4 How Will SQL Server 2000 Benefit into Organization?

SQL Server 2000 includes many new and enhanced features that have proven beneficial to all types of organizations and applications, including e-commerce, business intelligence, and line-of-business applications. Integrated technologies such as XML support, OLAP, and data-mining engines offer an unprecedented list of features, allowing SQL Server to play an integral role in every aspect of your organization—from business-to-business integration and electronic commerce to back-office data analysis and decision support. The importance of technologies such as XML continues to increase as organizations work toward greater integration with business partners, providing higher levels of efficiency and access to new customers. OLAP and data-mining capabilities result in more successful business decisions based on the discovery of new information among your piles of data. Whether your organization is a small business or a multinational corporation, SQL Server 2000 offers advantages such as improved self-tuning, automatic file growth, and configuration wizards through four-node fail-over clustering, federated servers, and support for up to 32 processors and

64GB of memory. SQL Server 2000 offers compatible platform support ranging from Windows CE to Windows 2000 Datacenter Server, allowing organizations to leverage existing SQL programming skills to deliver applications on every Windows platform.

4.4.1 Will SQL Server 2000 Fit into Organization?

Whether you are currently using SQL Server or have implemented a different RDBMS, SQL Server 2000 provides the tools for a successful migration or peaceful and productive coexistence. For organizations upgrading to SQL Server 2000 from SQL Server 6.5 and 7.0, backward compatibility and upgrade wizards make the migration seamless and efficient, with no modifications to your existing applications in most situations. A SQL Server 2000 named instance installation can even run concurrently with SQL Server 6.5 or 7.0 on the same server, so your database server can remain active during your migration. Tools such as the new Copy Database Wizard minimize database server downtime by copying databases and configuration information from SQL Server 7.0 servers without affecting their ability to service active applications.
Migrating from other RDBMSs or integrating SQL Server 2000 into heterogeneous environments is easily accomplished with its support for OLE DB providers for Oracle and Sybase, to name a couple. Nearly all popular database and information storage providers offer OLE DB drivers, allowing SQL Server to connect to other systems and perform tasks such as data import or export, execute distributed queries, and work with remote data sets using the OPENROWSET function. DTS in SQL Server allow you to attach to remote database systems and migrate objects and data. All SQL Server 2000 integration and migration capabilities will assist your organization in making a smooth transition to SQL Server 2000.

CHAPTER 5

5. SQL DATABASE DESIGN OF THE PROGRAM

6.

3.1 Database table Design of The Program

I have used six tables, six views and two stored procedures for my server asusa6vq. Which have this adress root on my database c.ConnectionString = "data source=asusa6vq;initial catalog=adem;integrated security=true" this database use the integrated security. İt means it's related to windows and sql server authontication.

The names of the tables are: login, company, customer, product, sale, stock

The name of thenames are: a,b,c, custotal, salepid and stockpid

I used the two data stored procedures for reporting. Because we can insert any dynamical data every time, which names are zrapor and zrapor2

Login table includes these fields

Column name	Data type	length
userid	int	4
uname	nvarchar	20
upass	nvarchar	15
access	nvarchar	15

Fig. 3.1.1 Login Table

Customer table includes these fields

Column name	Data type	length
Cusid	int	4
Cusname	nvarchar	50
Cussurname	nvarchar	50
Cuscompany	nvarchar	50
Cusfax	nvarchar	50
Cusphone	nvarchar	50
cusmail	nvarchar	50
cusaddress	nvarchar	50
custaxno	nvarchar	50

cusimage	nvarchar	70

Fig. 3.1.2 Customer Table

Company table includes these fields

Column name	Data type	length
Comid	int	4
Comname	nvarchar	50
Comphone	nvarchar	50
Comfax	nvarchar	50
Comweb	nvarchar	50
Commail	nvarchar	50
Comadress	nvarchar	70
comimage	char	50

Fig. 3.1.3 Company Table

Product table includes these fields

Column name	Data type	length
Pid	İnt	4
Pcode	Nvarchar	50
Pname	Nvarchar	100
Pprice	Money	8
Pguaranty	İnt	4
Pcompany	Nvarchar	50
pimage	nvarchar	80

Fig. 3.1.4 Product Table

Sale table includes these fields

Column name	Data type	length
salid	Int	4
customerid	Int	4
pid	Int	4
salcode	Nvarchar	50
salname	Nvarchar	100
saldate	Smalldatetime	4
salprice	Money	8

saltotal	Money	8
salvat	Int	4
salquantity	Int	4
salguaranty	Int	4

Fig. 3.1.5 Sale Table

Stock table includes these fields

Column name	**Data type**	**length**
Stid	Int	4
Pid	Int	4
Stcode	Nvarchar	50
Stname	Nvarchar	100
Stbuy	Money	8
Stcompany	Nvarchar	50
Stguaranty	Int	4
Stdate	Smalldatetime	4
Stquantity	Int	4
stimage	nvarchar	100

Fig. 3.1.6 Stock Table

I used six views to make aggregate functions.

Code of View A:

```
CREATE VIEW dbo.a
AS
SELECT    pid, salcode, salname, sum(salquantity) as total
FROM      dbo.sale
GROUP BY salcode, pid, salname
```

The above code make the aggregate funtion "summation" of salquantity.

Code of View B:

```
CREATE view dbo.b(pid,stcode,stname,stquantity)
```

as

select pid,stcode,stname,sum(stquantity)

from

stock

group by stcode,pid,stname

The view b mission is to make summation of stquantity

Code of View C:

```
CREATE VIEW dbo.c
AS
SELECT        dbo.b.pid, dbo.b.stcode, dbo.b.stname, dbo.b.stquantity, dbo.a.total,
dbo.b.stquantity - dbo.a.total AS kalan
FROM        dbo.a INNER JOIN
                 dbo.b ON dbo.a.pid = dbo.b.pid
```

After writing a view and b view. We made the subtruction of this two views result. This is writing like this above command dbo.b.stquantity - dbo.a.total. We use this in the stock control pages. This page shows the remaining product units. And the user can make order easly.

Code of View Custotal:

```
CREATE VIEW dbo.custotal
AS
SELECT     customerid, SUM(saltotal) AS total
FROM       dbo.sale
GROUP BY customerid
```

Custotal views used in the extras pages to show the total spending by customer id.

Code of View Salepid:

```
CREATE VIEW dbo.salepid
AS
```

SELECT pid, salcode, salname, SUM(salquantity) AS quantity, SUM(saltotal) AS total FROM dbo.sale GROUP BY salcode, salname, pid

Salpid created to take sum salquantity and saltotal colums.

Code of View Stockpid:

CREATE VIEW dbo.stockpid

AS SELECT pid, stcode, stname, SUM(stquantity) AS quantity

FROM dbo.stock

GROUP BY pid, stcode, stname, stquantity

Stockpid makes summation of stquantity as quantity by stokcode

We use the some relations between the tables to make consistent and reliable information from tables. Which are listed below:

- The stok table (stid) is primary key – the sale (stid) is foreign key
- The company (comname) is primary key – stock (stcompany) is foreign key
- The customer (cusid) is primary key – Sale (customerid) is foreign key

The program edimspeed stock control ask the username and password to access the main menü when the program starting.

Fig 3.1 Login Form

When the starting the program, we see this fig 3.1 transparancy form. This form require username and password to access to thr main menu. I used "giris" class to control the username and password. We used transparancy technic in this form. This form not have border, but we move to anywhere by mouse.

If the user enter wrong password the message box show fig 3.2

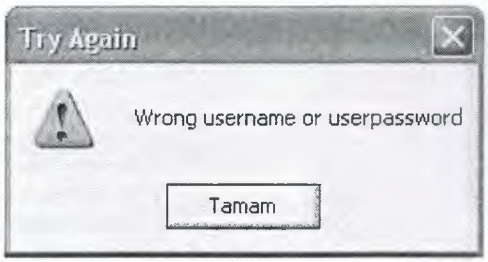

Fig 3.2 Wrong password messegebox

Fig 3.3 Main Page

The main menu is the main form of the edimspeed stock control. This menu contains the label. Which show the licence statement. If the product register, the label will be edimspeed stock control full version. Else the program will be trial period. Trials program allows 30 days. If the correct serial number not enter succesfully to the

program will not be open after 30 days. The program control this events with windows registry.

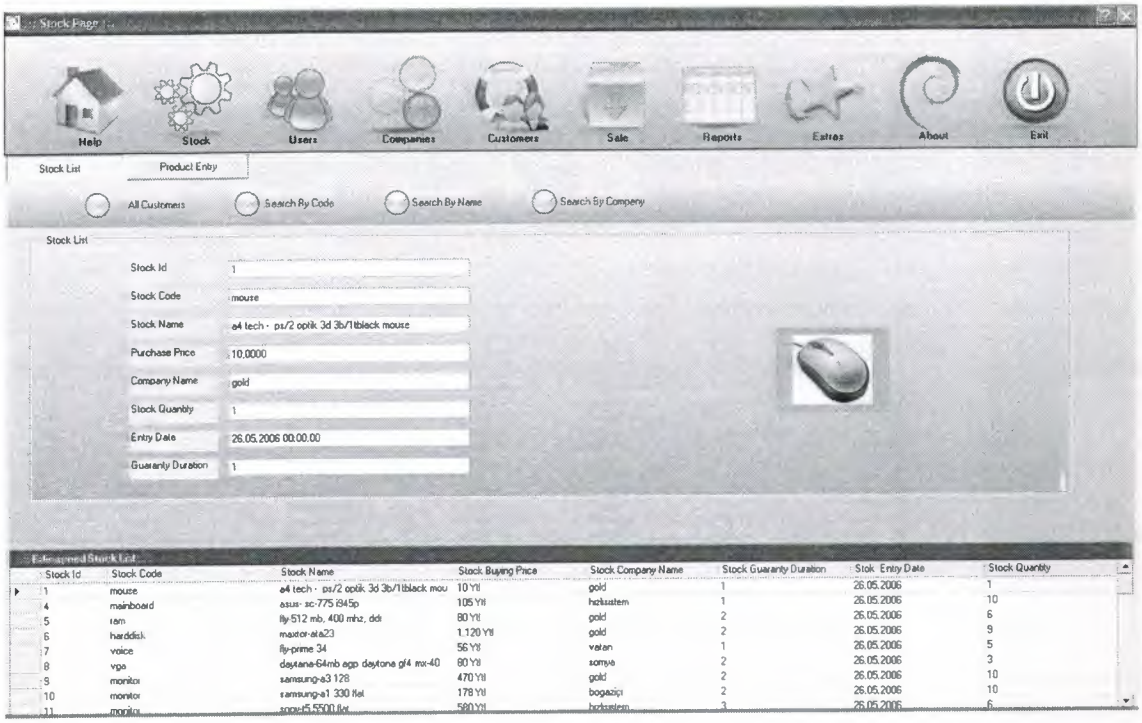

Fig 3.4 Stock List

Stok list shows the detail of the inserted product and product image. In this page we used currency manager and dataview. If we click the datagrid, clicked row information shows in the textboxes dynamically. In this form we find the records easly with above buttons. We search the record with by code, name, company. And than finded records number shows on the menu title menu.

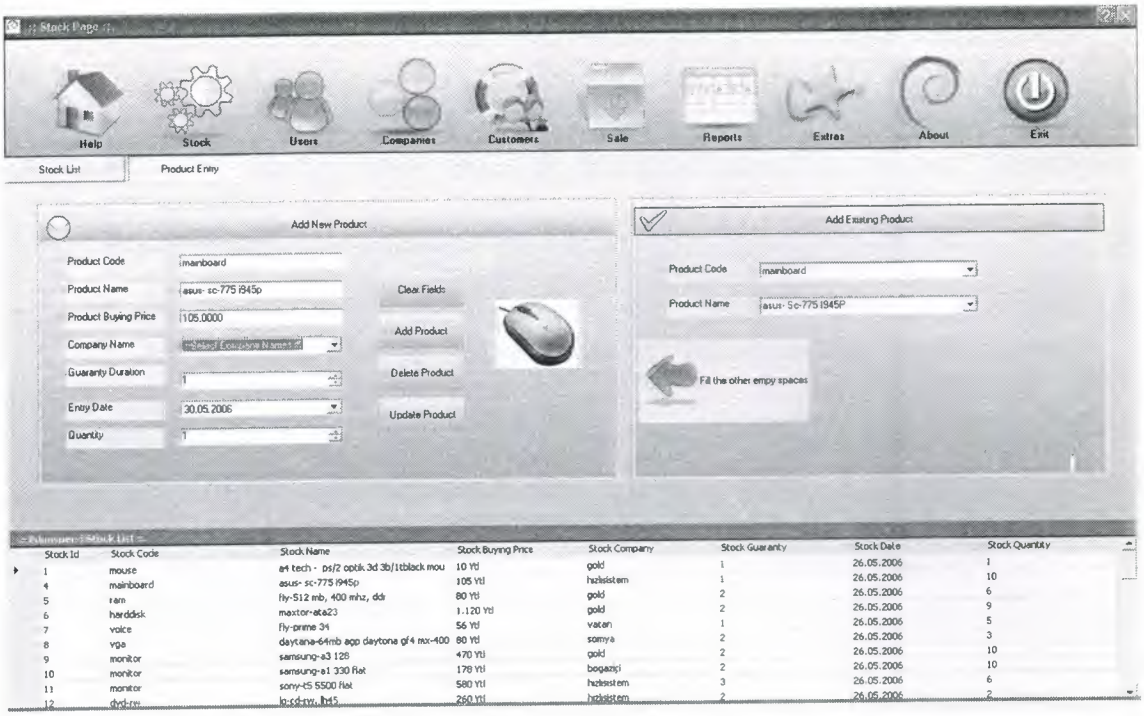

Fig 3.5 Stock Entry

In this form, we can add new product or existing product. And we declare the logo for new inserted product. This form control the data, makes this operation clear, update, delete and add

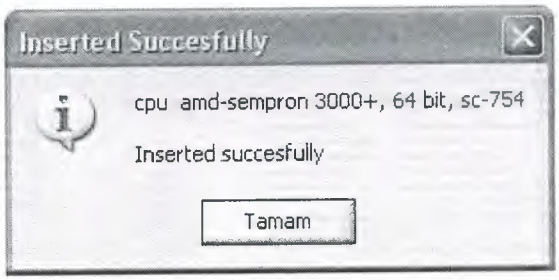

Fig 3.6 Succesfull Insertion

If record is added to database succesfully below fig 3.6 appear on the screen.

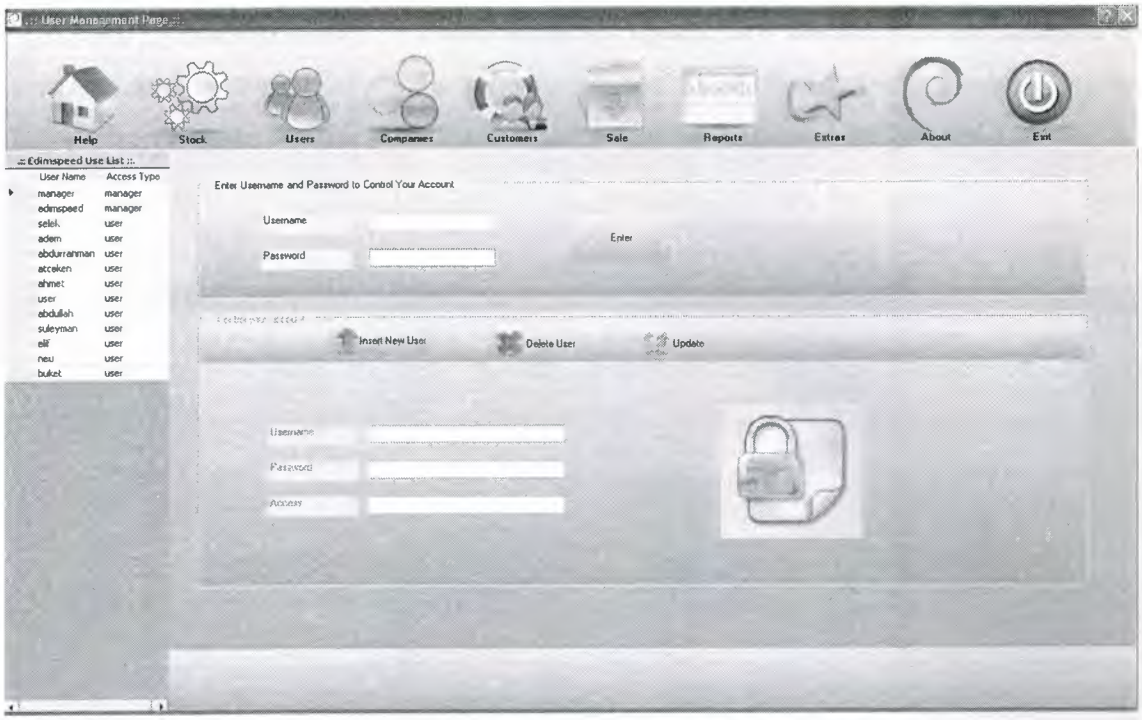

Fig 3.7 User Management Form

To update, delete and add our account, firstly entered username and password. After that we do these operation. The datagrid shows only usernames and access types.

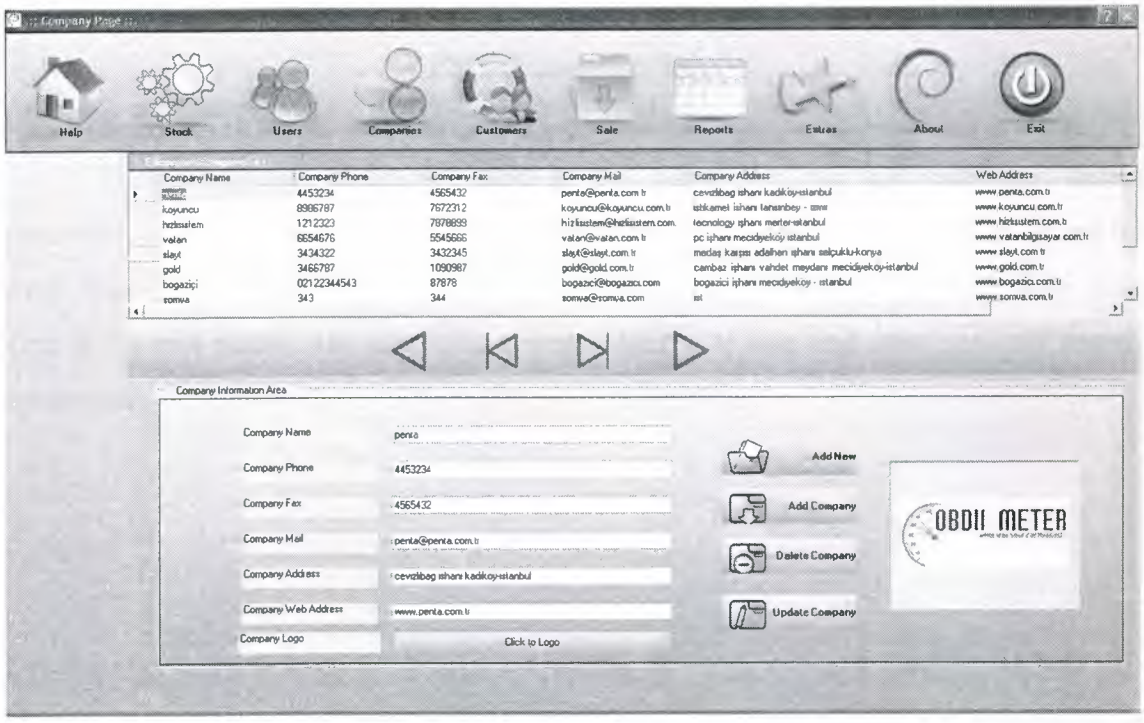

Fig 3.8 Company form

In "Company Form " we can add,delete,update new company to the database by pressing **add** button, after all the information are filled, from which company we are buying the product. And to clear the textbox it is needed to press **clear** button
We use in this form currency manager. First, previous,next and last operation.

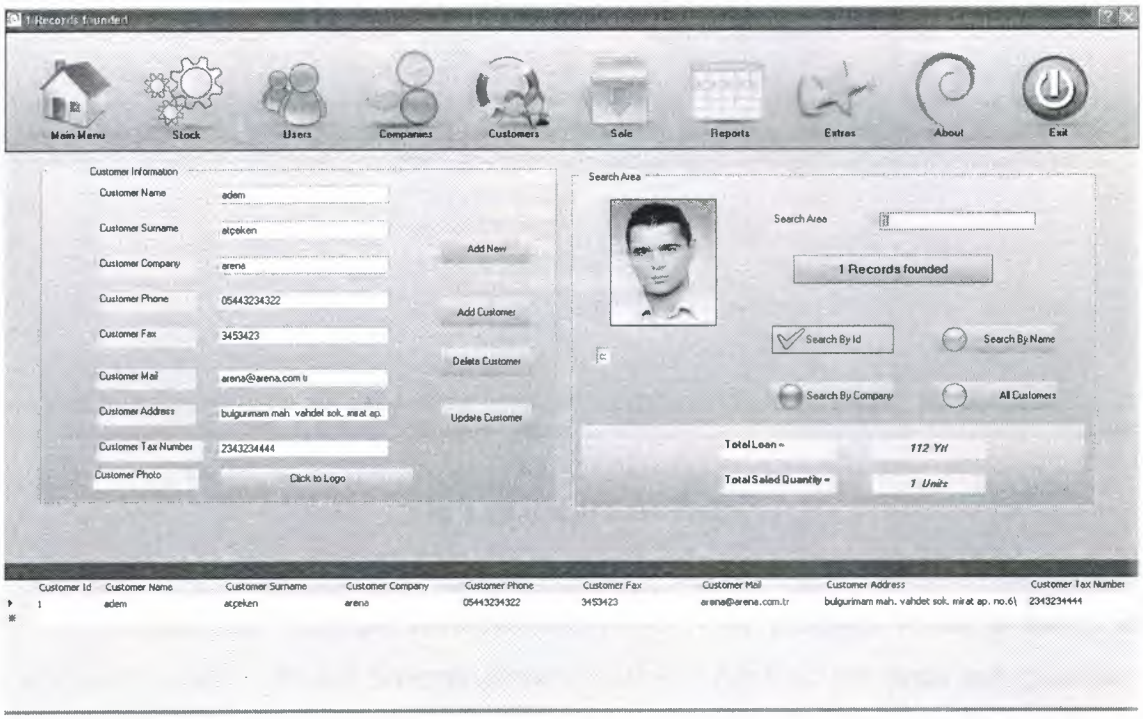

Fig 3.9 Customer form

Here Customer details are held in this form. We can add customer, delete customer, update customer in this form. Moreover we can search customer by Id and by First Name and Surname in the same form. When you select one customer then click one of the sale button the sale form appears. And total loan and saled quantity shown on secreen.

Fig 3.10 Retail Sale Form

In retail sale form the customer select one of the products which is shown in ComboBox, then customer determines the quantity. After that the price and guarantee are appeared automatically. Then price calculated automatically and "With VAT (18%)" are calculated defaultly but the user can change the vat percentage on the numericupdown control. When you press the "SELL" button you will see such as a "Please Confirm Sale" message: shown below fig 3.11 figure

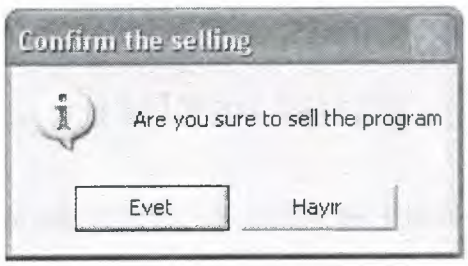

Fig 3.11 Confirm sale message

If user click evet the sale operation finished succesfully, else the operation terminated. And this messagebox shown on secreen

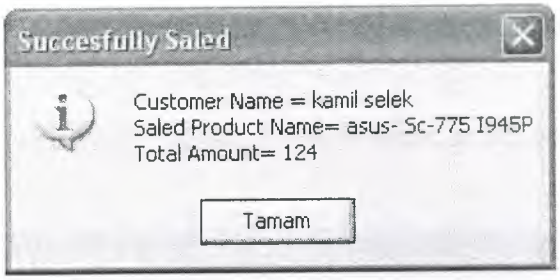

Fig 3.12 Succesfull sale message

This messages means the sale finished succesfully.

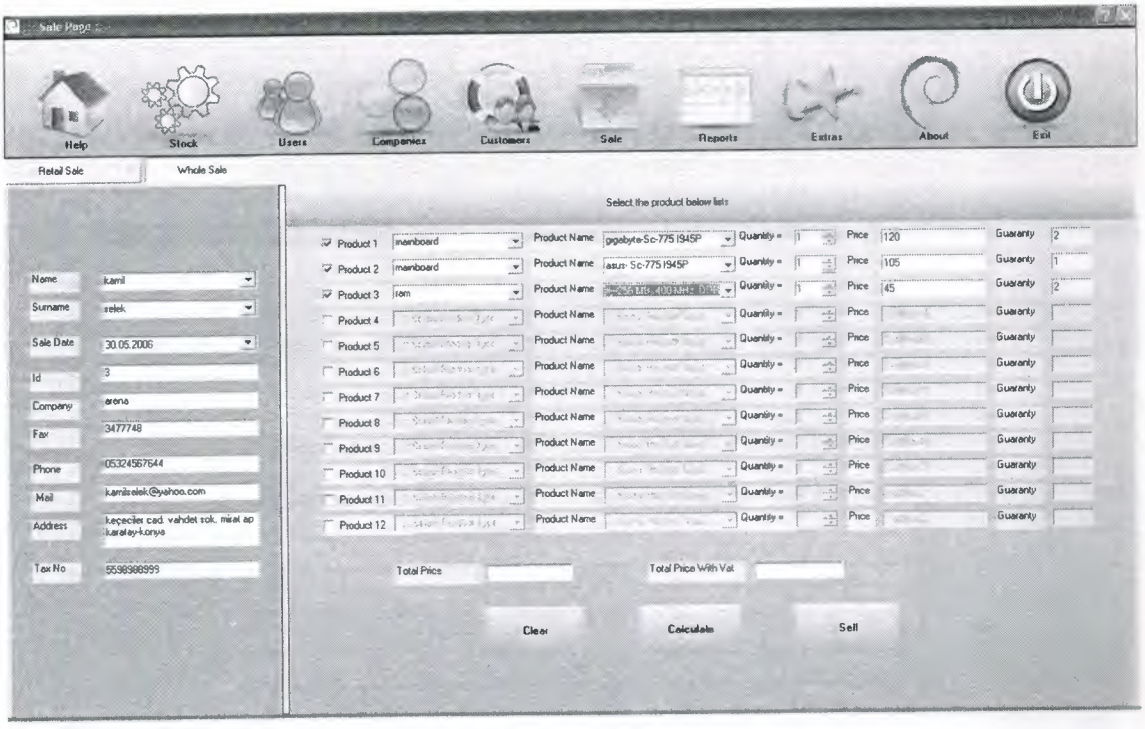

Fig 3.13 Whole Sale Form

If we sale more then one product we choose the whole sale form. ın this form we can sale twelve product. Firstly we should select username arena and than peoduct area. Normally the boxes is not active if we click the checkbox it will be active

In Wholesale form at the first clicked checkboxes then Comboboxes, Quantities, Prices, Guarantees will be activated. Then we can select the products in the Comboboxes and determine the quantity. Then when it is clicked the "CALCULATE" button "Total Price" and "With VAT (18%)" are calculated and displayed on the form.

When you press the "SELL" button you will see such as a "Please Confirm Sale" message"

If we click the yes confirm message, below message shown on screen

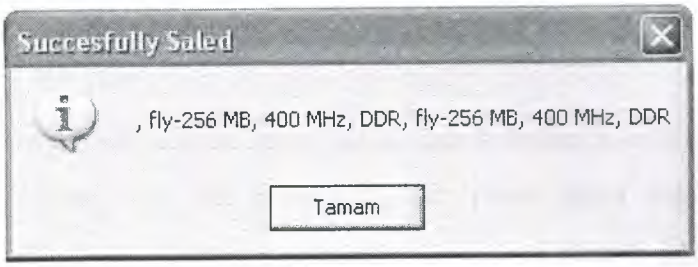

Fig 3.14 Succesfull Sale Message

This messages shows that, the sale operation finished succesfully.

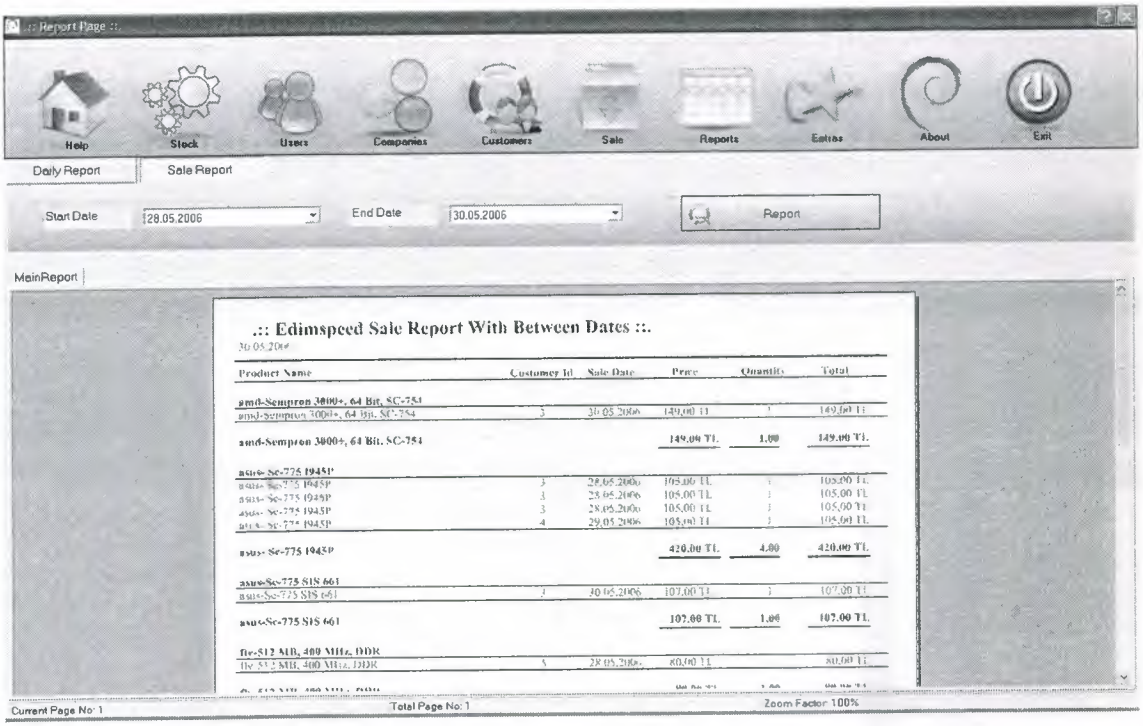

Fig 3.15 Daily and with Specific Date Sale Report

This form contains two part first part daily sale report, this is report page with system day date. We cannot add any parameter. We use in this report zrapor1. Which stored procedure code is like this

```
CREATE procedure zrapor
@tarih smalldatetime =saldate
as
select salid, customerid, salname, saldate, salprice, salquantity, saltotal from sale where
(saldate=@tarih)
GO
```

In this form we can see the report about sale between two specific dates. When we determine the Start Date and End Date, then press report button, then displays product name, date, price, quantity information as it is shown fig. below. We use zrapor2 for the specific date report. Zrapor2 contains this codes.

```
CREATE procedure zrapor2
(@tarih1 smalldatetime,
@tarih2 smalldatetime )
as
select salid, customerid, salname, saldate, salprice, salquantity, saltotal from sale where
(saldate>=@tarih1) and (saldate<=@tarih2)
GO
```

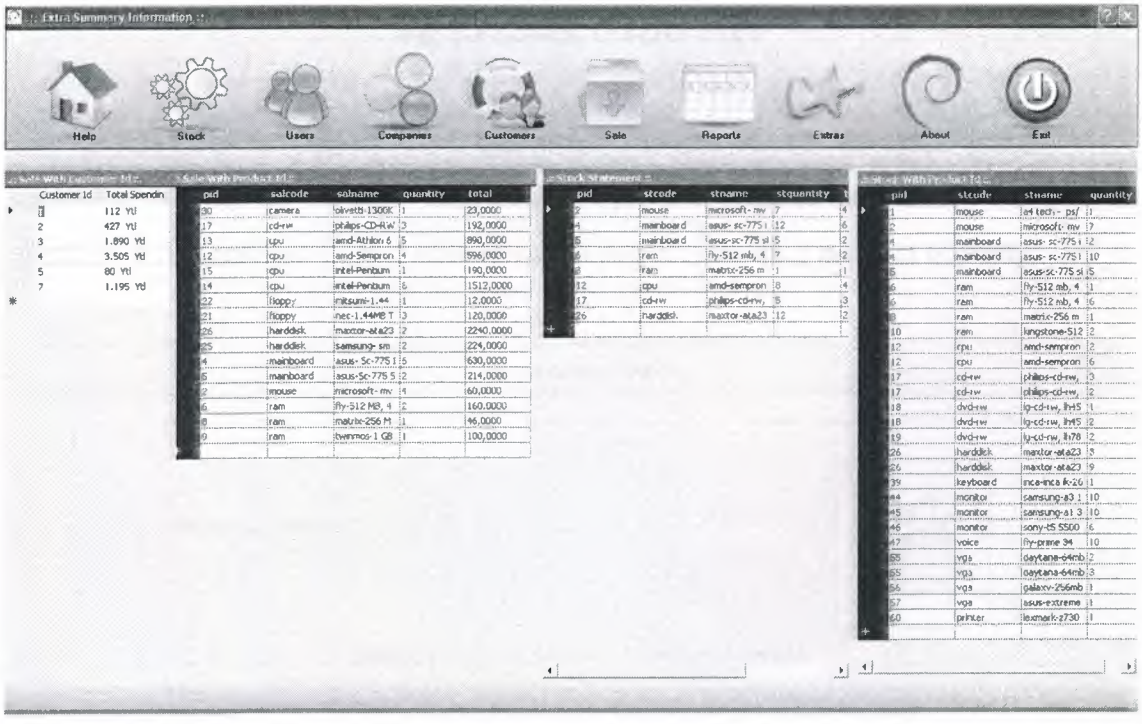

Fig 3.16 Extras Form

In the trial period extras button is enabled. If the user enter the correct serial number by registration form the extras buttons will be active. I mean this extras form for full version users. So advise that to users to see fig. 3.16 extras form, purchase the product.

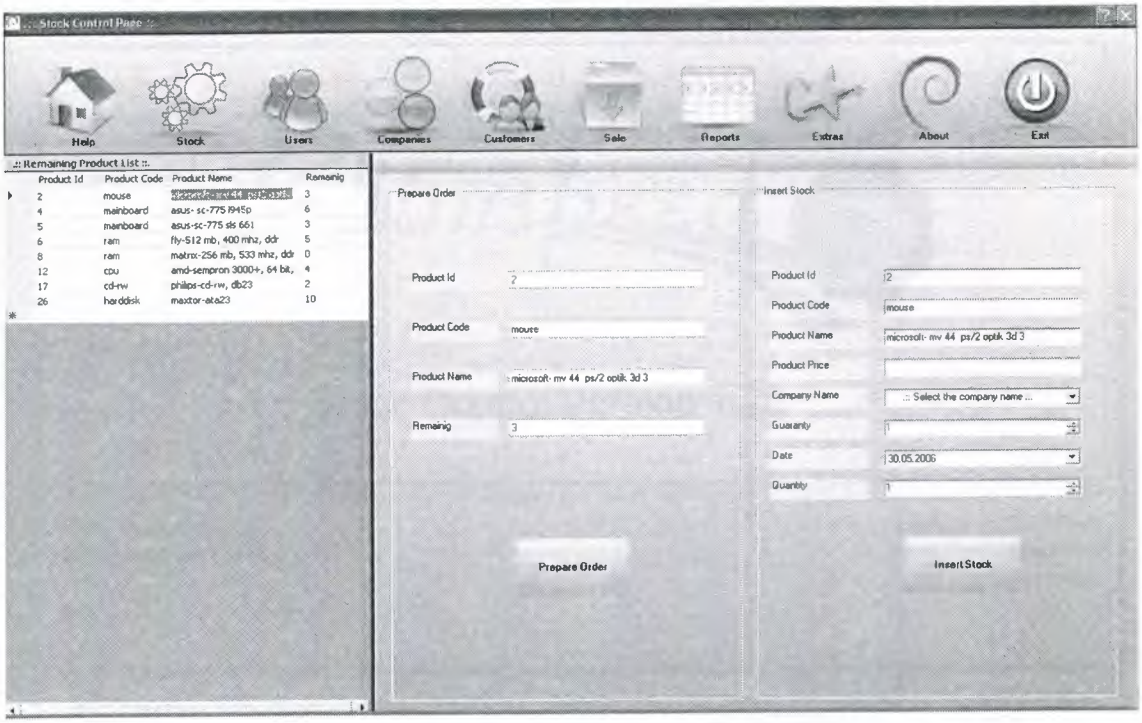

Fig 3.17 Stock Control Page

In this form controls the stock quantities. The user can see the stock quantity statement and then on the same form make the product order to increase the less product quantity. This form use c view table to calculate the remaining product quantity.

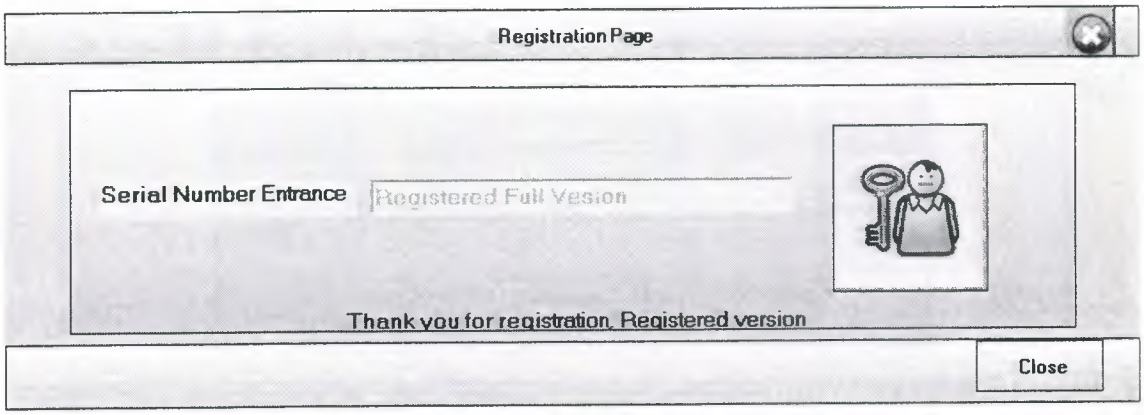

Fig 3.18 Registration Form

To get registered full version we should enter the serial number. If thee serial is correct the program mode change trial 30 to full vesion. To take the serial number the user should click www.ademateeken.ekibi.com or send email edimspeed@yahoo.com

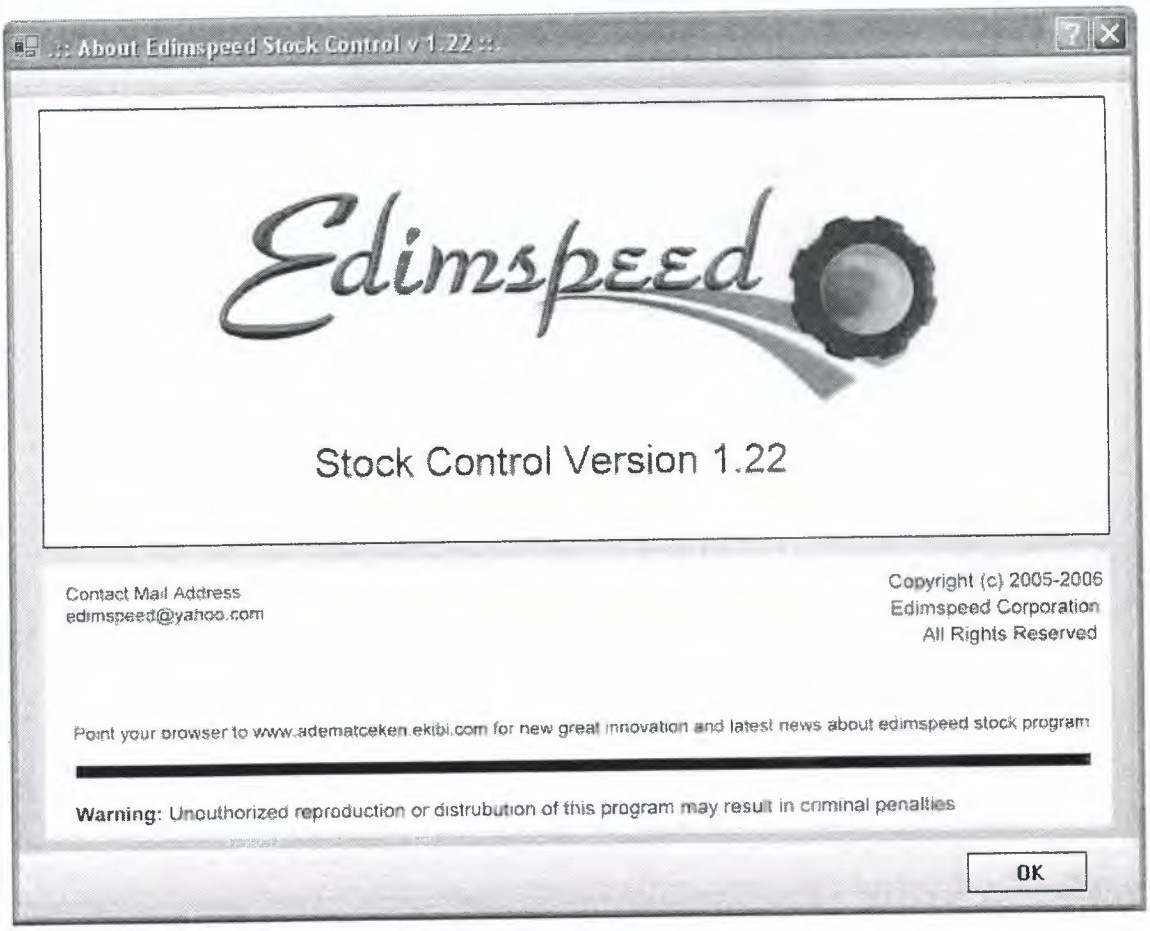

Fig 3.19 About Form

The about form give the necessary information contact address about edimspeed stock program. At the bottom of the page, there is a warning message to unouthorized reproduction or distrubution of edimspeed program.

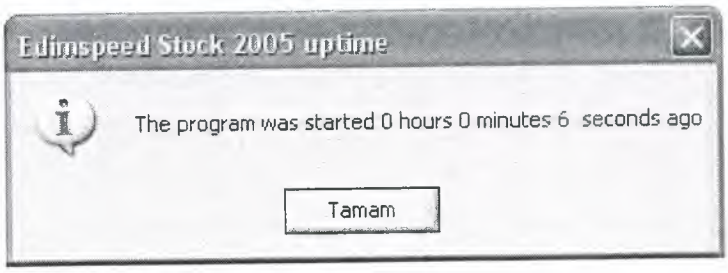

Fig 3.20 Program Uptime

Program uptimes starts, when the program starting and anytime if the user want to see how many time spent on this program. Fig 3.20 shows the spended time as hour and minutes and second. May usefull for users.

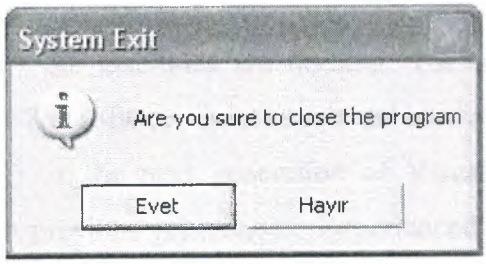

Fig 3.21 Program Exit

If the user want to close the program, click the exit "red" button. The program wants to confirm from the user to close, if the user click yes the program closed whole of the program. We used in this part this code: application.exit

CONCLUSION

Recently, the technology is developing a lot and started to use by everyone in the world. Because of placing the new tecnology to every platform in our life human needed to combine both software and hardware. We need software and hardware together Without software the machines are nothing. The new tecnologic machines needs software to operate. The software must supply the hardware for best operating.

Visual Basic .NET is the next generation of Visual Basic, but it is also a significant departure from previous generations. Experienced Visual Basic 6 will feel comfortable with Visual Basic .NET code and will recognize most of its instructions. Vb.net support many advantages, which The language itself is now fully object-oriented. Before versiones of visual basic do not supported object oriented tecnology.

Microsoft SQL Server 2000 is more userful than a relational database management system; it is a complete database and analysis product that meets the scalability and reliability requirements of the most demanding enterprises. We used sql server personel edition in my project.

Sql database management system is using record keeping system that stores, maintains and provides access to the information. A Database system consists of four major components that are Data, Hardware, Software, and Users. DBMS are used by any reasonably self contained commercial, scientific, technical or other organization for a single individual to a large company. Practically implementation of software for business though it is related to any field needs a devoted and complete life cycle. The most important idea is the attitude which has to be face during the life cycle of the Company or Organization. And according to this point of view the reason of most unsuccessful project is misunderstanding between the two parties.

The software created after a deep analysis, firstly we should determine the requirements, so that all-important requirements to the company dealing with computer product sales and purchase can be accomplished. Company and product, name and ID have been included in the program to overcome the mistakes, which can occur. Reports are also generated with the help of the Queries for the update purpose

REFERENCES

Reference to Books:

Palme publisher Visualbasic.net 2003

Palme publisher Sql Server Ado.Net

Reference to E-Book:

Dave Grundgeiger Programming Visual Basic .NET

Robert Patton, sql server databases for .net

Steven RomanVB .NET Language in a Nutshell

A Programmer's Introduction to v.b Paul Boger Publisher Sams Publishing

Vb.net developer guide Cameron Wakefield Henk, Evert Sonder, Wei Meng Lee

Reference to www

www.codeprojects.com

www.programmersheaven.com

www.vb_masterschool.com

www.vbmaster.com

www.programlama.com

APPENDIX A: PROGRAM CODES

CLASS "GIRIS" CODE

```
Public Class giris
    Private ad As String
    Private mail As String
    Private uname As String
    Private upass As String
    Public Sub logon(ByVal n As String, ByVal s As String)
        Dim c As New SqlClient.SqlConnection
        Dim co As New SqlClient.SqlCommand
        Dim dr As SqlClient.SqlDataReader
        Try
            c.ConnectionString = "data source=asusa6vq;initial catalog=adem;integrated
security=true"
        co.Connection = c
        co.CommandText = "select * from login"
        c.Open()
        Dim z As Integer = 0
        dr = co.ExecuteReader
        Do While dr.Read
            If LCase(n) = dr("uname") And LCase(s) = dr("upass") Then
                z = 1
                p1.Text = "dogru"
                Exit Do
            End If
        Loop
            If z = 0 Then
                p1.Text = "yanlıs"
```

```
            Throw New Exception("Wrong username or userpassword")
        End If
            Catch ex As Exception
        MessageBox.Show("Password veya ¸ifrenizde yanlı¸lık var", "Yanlı¸Giri¸", 
MessageBoxButtons.OK, MessageBoxIcon.Exclamation,
MessageBoxDefaultButton.Button1)
      Finally
        dr.Close()
        c.Close()
      End Try
   End Sub
End Class
```

MODUL "MODUL1" CODE

```
Module Module1
    Public p1 As New Form1 'entrance form
    Public p2 As New Form2 'stock form
    Public p3 As New Form3 'users form
    Public p4 As New Form4 'Companies form
    Public p5 As New Form5 'customers form
    Public p6 As New Form6 'sale form
    Public p7 As New Form7 'report form
    Public p8 As New Form8 'registration
    Public p9 As New Form9 'main
    Public p10 As New Form10 'stock control page
    Public p11 As New Form11 ' guaranty validate
    Public p12 As New Form12 ' extra summary information
    Public p13 As New Form13 ' about
    Public s1, s2, a1, a2 As DateTime
    Public sd, sd2, sd3, sd4, oku, sgir, sdate, fdate, son As String
```

```
    Public kgun As Integer
    Public pid As Integer  'product id
    Public tarih, tarih2 As DateTime
    Public www As String
    Public cusid, say1, say2, say3, say4 As Integer ' customer id
    'oku registry serial  number
    'sgir you enter the serial
End Module
```

FORM 1. ENTRANCE FORM

```
Imports Microsoft.Win32
    Private m As Point
    Private i As Boolean = False
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)
        Try
            Dim g As New giris
            If TextBox1.Text = "" Or TextBox2.Text = "" Then
                MessageBox.Show("Kullanıcı adı veya Ýfre boÝbırakılamaz", "Dikkat",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk, MessageBoxDefaultButton.Button1)
                Exit Sub
            End If
            g.logon(TextBox1.Text, TextBox2.Text)
        Catch ex As SqlClient.SqlException
            MsgBox(ex.Message & vbTab & ex.Number)
        End Try
    End Sub
        Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        TextBox1.Focus()
```

```
        oku = Registry.ClassesRoot.GetValue("edims")
    sdate = Registry.ClassesRoot.GetValue("sdate")
    a1 = sdate
    s1 = Now
    If oku Is Nothing Then  'if password not write then
      Registry.ClassesRoot.SetValue("edims", "edimspeed2005")
    End If
    If sdate Is Nothing Then
      Registry.ClassesRoot.SetValue("sdate", Now.ToShortDateString)
    End If
    fdate = DateAdd(DateInterval.Day, 30, a1)
    a2 = fdate
    kgun = DateDiff(DateInterval.Day, a1, a2)
    Dim ss As String
    ss = Registry.ClassesRoot.GetValue("edimd")
    If ss <> "edimspeed2005" Then
      p8.ButtonBand2.Text = "Remaining Trial Period= " & kgun & " days"
      p9.Label1.Text = "Remaining Trial Period= " & kgun & " days"
      If kgun <= 0 Then
        MessageBox.Show("Your trial 30 days program usage period finished", "Trial
Period finished", MessageBoxButtons.OK, MessageBoxIcon.Warning)
        Application.Exit()
      End If
    ElseIf ss = "edimspeed2005" Then
      p12.Button31.Enabled = True
    End If
  End Sub
        End If
      End If
      If p1.Text = "dogru" Then
        p1.Close()
```

```
            p9.Show()
        End If
    Catch ex As SqlClient.SqlException
        MsgBox(ex.Message & vbTab & ex.Number)
    Finally
    End Try
End Sub
Private Sub PictureBox1_MouseDown(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles PictureBox1.MouseDown
    Dim x As Integer
    Dim y As Integer
    If e.Button = MouseButtons.Left Then
        x = -e.X - SystemInformation.FrameBorderSize.Width
        y = -e.Y - SystemInformation.CaptionHeight -
SystemInformation.FrameBorderSize.Height
        m = New Point(x, y)
        i = True
    End If
End Sub
Private Sub PictureBox1_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles PictureBox1.MouseUp
    If e.Button = MouseButtons.Left Then
        i = False
    End If
End Sub
Private Sub PictureBox1_MouseMove(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles PictureBox1.MouseMove
    If i Then
        Dim p As Point = Control.MousePosition
        p.Offset(m.X, m.Y)
        Location = p
```

```
        End If
    End Sub
    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Timer1.Tick
        sdate = Registry.ClassesRoot.GetValue("edimd")
        If sdate <> "edimspeed2005" Then
            If kgun <= 0 Then
                MessageBox.Show("Your trial program, usage period finished", "Trial period
finished, program will be closed", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation)
                Application.Exit()
            End If
        End If
    End Sub
    End Class
```

FORM 2. STOCK LIST AND ADD FORM

```
Imports System.IO
Public Class Form2
    Inherits System.Windows.Forms.Form
    Public asa As CurrencyManager
    Public buy As Integer
    Dim img As String
    Dim tr As Date
    Dim a As DateTime
        End Sub
    Private Sub TabControl1_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles TabControl1.SelectedIndexChanged
        If TabPage1.Focus = True Then
```

```
            ds.Clear()
            da.Fill(ds.stock)
                End If
        If TabPage2.Focus = True Then
            ds.Clear()
            da2.Fill(Ds2.stock)
            Try
                Dim co As New SqlClient.SqlCommand
                Dim dr As SqlClient.SqlDataReader
                c.Open()
                co.CommandText = "select * from company"
                co.Connection = c
                dr = co.ExecuteReader
                Do While dr.Read
                    ComboBox1.Items.Add(dr("comname"))
                Loop
                c.Close()
                dr.Close()
            Catch ex As Exception
                MsgBox(ex.Message)
            Finally
                ComboBox1.Text = ".::Select Company Names ::."
                HoverGradientButton3.Enabled = True
                Button33.Selected = False
                da2.Update(ds.stock)
                DataGrid1.DataSource = DataView1
            End Try
        End If
    End Sub
    Private Sub Form2_Load(ByVal sender As Object, ByVal e As System.EventArgs)
Handles MyBase.Load
```

```
        TabPage1.Focus()
        ds.Clear()
        da.Fill(ds.stock)
        asa = CType(Me.BindingContext(DataView1), CurrencyManager)
        da.Fill(ds.stock)
    End Sub
    Private Sub HoverGradientButton3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles HoverGradientButton3.Click
        Try
            Dim d, m, y, t As String
            tarih = DateTimePicker1.Value
            www = tarih.Month & "." & tarih.Day & "." & tarih.Year
            buy = TextBox12.Text
            Dim c As New SqlClient.SqlConnection
            c.ConnectionString = "data source=asusa6vq;initial catalog=adem;integrated
security=true"
            Dim co As New SqlClient.SqlCommand
            Dim i As Integer = 0
            c.Open()
            sd4 = "C:\proje\pics\stock\" & TextBox10.Text & "." & "jpg"
            co.CommandText = "insert into
stock(pid,stcode,stname,stbuy,stcompany,stguaranty,stdate,stquantity,stimage) values(" &
pid & ",'" & TextBox10.Text & "','" & TextBox11.Text & "'," & buy & ",'" &
ComboBox1.Text & "'," & NumericUpDown2.Value & ",'" & www & "'," &
NumericUpDown1.Value & ",'" & sd4 & "')"
            co.Connection = c
            i = co.ExecuteNonQuery()
            If i > 0 Then
                MessageBox.Show(TextBox10.Text & "  " & TextBox11.Text & vbCrLf &
"Inserted succesfully", "Inserted Succesfully", MessageBoxButtons.OK,
MessageBoxIcon.Information)
```

```
        ElseIf i = 0 Then
            MessageBox.Show("Can't make Insert operation", "Not Succesfull",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
        End If
      Catch ex As SqlClient.SqlException
        MsgBox(ex.Message)
      Finally
        da.Fill(Ds2.stock)
        c.Close()
      End Try
      c.Close()
    End Sub
    Private Sub HoverGradientButton4_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles HoverGradientButton4.Click
      a = DateTimePicker1.Text
      Try
        If MessageBox.Show("Are you sure to delete Product? ", "Are You sure to delete?",
MessageBoxButtons.YesNo, MessageBoxIcon.Warning) = DialogResult.No Then
          Exit Sub
        End If
        Dim c As New SqlClient.SqlConnection
        c.ConnectionString = "data source=asusa6vq;initial catalog=adem;integrated
security=true"
        Dim co As New SqlClient.SqlCommand
        Dim i As Integer = 0
        c.Open()
        co.CommandText = "delete from stock  where stcode='" & TextBox10.Text & "'
and stname='" & TextBox11.Text & "' "
        co.Connection = c
        i = co.ExecuteNonQuery()
        If i > 0 Then
```

```
            MessageBox.Show(TextBox10.Text & "  " & TextBox11.Text & vbCrLf &
"Deleted succesfully", "Deleted Succesfully", MessageBoxButtons.OK,
MessageBoxIcon.Information)
        ElseIf i = 0 Then
            MessageBox.Show("Not succesfull operation", "Not Succesfull",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
        End If
        c.Close()
      Catch ex As SqlClient.SqlException
        MsgBox(ex.Message)
      Finally
        da.Fill(Ds2.stock)
      End Try
    End Sub
    End Class
```

FORM 3. USER MANAGEMENT FORM

```
Public Class Form3
    Inherits System.Windows.Forms.Form
    Private Sub Form3_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
      ds.login.Clear()
      dalogin.Fill(ds.login)
      Me.Text = ".:: User Management Page ::."
    End Sub
        Private Sub Button9_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button9.Click
      Dim ds As New DataSet
      Dim co As New SqlClient.SqlCommand
      Dim dr As SqlClient.SqlDataReader
```

```
        Dim i As Integer = 0
        Try
            co.Connection = c
            co.CommandText = "select * from login"
            c.Open()
            dr = co.ExecuteReader
            If TextBox4.Text = "" Or TextBox5.Text = "" Then
                MessageBox.Show("Kullanıcı ismi veya Ÿfre boŸbırakılamaz", "Lütfen
Doldurunuz", MessageBoxButtons.OK, MessageBoxIcon.Asterisk)
                Exit Sub
            End If
            Do While dr.Read
                If TextBox4.Text = dr("uname") And TextBox5.Text = dr("upass") Then
                    TextBox3.Enabled = False
                    i = 1
                    Exit Do
                End If
            Loop
            If i = 0 Then
                MessageBox.Show("Kullanıcı adı veya Ÿfre yanlıŸır", "YanlıŸGiriŸ",
MessageBoxButtons.OK, MessageBoxIcon.Warning)
            End If
        Catch ex As SqlClient.SqlException
            MsgBox(ex.Message)
        Finally
            dr.Close()
            c.Close()
        End Try
    End Sub
        Private Sub Button11_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button11.Click
```

```
        Dim co As New SqlClient.SqlCommand
        Dim co2 As New SqlClient.SqlCommand
        Dim dr As SqlClient.SqlDataReader
        Dim i As Integer = 0
        ds.login.Clear()
        If TextBox1.Text = "" And TextBox2.Text = "" Then
            MessageBox.Show("Username ve password kısımlarını boÝbırakmayınız", "BoÝ
Bırakmayınız", MessageBoxButtons.OK, MessageBoxIcon.Warning)
            Exit Sub
        End If
        Try
            co2.CommandText = "select * from login"
            co.CommandText = "insert into login(uname,upass) values('" & TextBox1.Text &
"','" & TextBox2.Text & "')"
            co.Connection = c
            co2.Connection = c
            c.Open()
            dr = co2.ExecuteReader
            Do While dr.Read
                If TextBox1.Text = dr("uname") Then
                    MessageBox.Show(TextBox1.Text & " isimli kullanıcıdan var. Farklı bir
kullanıcı ismi kullanınız", "Kullanıcı isminizi degiÝiriniz", MessageBoxButtons.OK,
MessageBoxIcon.Warning)
                    dr.Close()
                    c.Close()
                    Exit Sub
                End If
            Loop
            dr.Close()
            i = co.ExecuteNonQuery()
            Dim a As String
```

```
        a = "Username= " & TextBox1.Text & vbCrLf & "Password= " & TextBox2.Text
        If i > 0 Then MessageBox.Show(a & vbCrLf & "Succesfully added to database",
"Succesfully Added", MessageBoxButtons.OK, MessageBoxIcon.Information)
      Catch ex As SqlClient.SqlException
        MessageBox.Show(ex.Message)
      Finally
        dalogin.Fill(ds.login)
      End Try
      c.Close()
      dr.Close()
    End Sub
      Private Sub Button8_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button8.Click
      If MessageBox.Show("Are you sure to close the program", "System Exit",
MessageBoxButtons.YesNo, MessageBoxIcon.Asterisk,
MessageBoxDefaultButton.Button1) = DialogResult.Yes Then
        Application.Exit()
      End If
    End Sub
    End Class
```

FORM 4. COMPANY FORM

```
Imports System.IO
Public Class Form4

    Inherits System.Windows.Forms.Form
    Public cm As CurrencyManager
    Public a As String
Private Sub Button8_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button8.Click
```

```
        If MessageBox.Show("Are you sure to close the program", "System Exit",
MessageBoxButtons.YesNo, MessageBoxIcon.Asterisk,
MessageBoxDefaultButton.Button1) = DialogResult.Yes Then
            Application.Exit()
        End If
    End Sub
    Private Sub Form4_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        dacompany.Fill(ds.company)
        cm = CType(Me.BindingContext(DataView1), CurrencyManager)
        TextBox7.Hide()
    End Sub
        End Sub
    Private Sub Button9_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)
        dacompany.Fill(ds.company)
    End Sub
    Private Sub HoverGradientButton7_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles HoverGradientButton7.Click
        cm = CType(Me.BindingContext(DataView1), CurrencyManager)
        cm.EndCurrentEdit()
        MessageBox.Show(TextBox1.Text & " added your database successfully", "Added
successfully", MessageBoxButtons.OK)
        dacompany.Update(ds.company)
    End Sub
        Private Sub HoverGradientButton4_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles HoverGradientButton4.Click
        If cm.Position = 0 Then
            MsgBox("You are on first record", MsgBoxStyle.Information, "First Record")
        Else
            cm.Position = cm.Position - 1
```

```
        End If
    End Sub
    Private Sub HoverGradientButton8_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles HoverGradientButton8.Click
        If cm.Position = cm.Count - 1 Then
            MsgBox("You are on last record", MsgBoxStyle.Information, "Last Record")
        Else
            cm.Position = cm.Position + 1
        End If
    End Sub
            Private Sub DataGrid1_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles DataGrid1.Click
        Dim a As String
        a = DataGrid1(DataGrid1.CurrentRowIndex, 6)
        PictureBox1.Image = Image.FromFile(a)
    End Sub
    Private Sub HoverGradientButton10_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles HoverGradientButton10.Click
        Dim f As String
        Dim c As String
        OpenFileDialog1.InitialDirectory = "c:\"
        OpenFileDialog1.Filter = "Jpg files (*.jpg)|*.jpg"
        OpenFileDialog1.ShowDialog()
        If OpenFileDialog1.ShowDialog = DialogResult.OK Then
            f = OpenFileDialog1.FileName.ToString
            Me.Text = f
            Dim dosya As New FileInfo(f)
            c = "c:\proje\pics\logo\" & TextBox1.Text & "." & "jpg"
            TextBox7.Text = c
            dosya.CopyTo(c, True)
```

```
        MessageBox.Show("Your logo saved this " & c & " directory", " Succesfully added
to database", MessageBoxButtons.OK, MessageBoxIcon.Information)
      Else
        MessageBox.Show("Your logo not saved database succesfully", "Not successful",
MessageBoxButtons.OK, MessageBoxIcon.Warning)
      End If
    End Sub
    Private Sub TextBox7_TextChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles TextBox7.TextChanged
      a = TextBox7.Text.ToString
      PictureBox1.Image = Image.FromFile(a)
    End Sub
    End Class
```

FORM 5. CUSTOMER FORM

```
Imports System.IO

Public Class Form5
    Inherits System.Windows.Forms.Form
    Public cm As CurrencyManager
    Public a As String
Private Sub Form5_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
      cm = CType(Me.BindingContext(DataView1), CurrencyManager)
      da.Fill(ds.customer)
    End Sub
      Private Sub HoverGradientButton2_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button9.Click
      Try
        cm = CType(Me.BindingContext(DataView1), CurrencyManager)
```

```
            cm.AddNew()
        Catch ex As Exception
            MsgBox(ex.Message)
        End Try
    End Sub
        Private Sub HoverGradientButton19_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs)
        DataView1.RowFilter = ""
        If DataView1.Count = 0 Then
            MessageBox.Show("The search find anything", "There is no item",
MessageBoxButtons.OK, MessageBoxIcon.Information,
MessageBoxDefaultButton.Button1)
        ElseIf DataView1.Count > 0 Then
            Me.Text = DataView1.Count & " Records founded "
            Button33.Text = DataView1.Count & " Records founded "
        End If
    End Sub
    Private Sub HoverGradientButton25_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)
        Try
            If TextBox9.Text = "" Then
                MsgBox("Please fill the customer name field", MsgBoxStyle.Exclamation, "Fill
empty space")
                Exit Sub
            End If
            DataView1.RowFilter = "cusname='" & TextBox9.Text & "'"
            If DataView1.Count = 0 Then
                MessageBox.Show("The search find anything", "There is no item",
MessageBoxButtons.OK, MessageBoxIcon.Information,
MessageBoxDefaultButton.Button1)
                Button33.Text = DataView1.Count & " Records founded "
```

```
        ElseIf DataView1.Count > 0 Then
            Me.Text = DataView1.Count & " Records founded "
            Button33.Text = DataView1.Count & " Records founded "
        End If
    Catch ex As Exception
        MsgBox("Plase fill the not numeric values")
    End Try
End Sub
Private Sub HoverGradientButton8_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)
    Try
        If TextBox9.Text = "" Then
            MsgBox("Please fill the customer name field", MsgBoxStyle.Exclamation, "Fill
empty space")
            Exit Sub
        End If
        DataView1.RowFilter = "cuscompany='" & TextBox9.Text & "'"
        If DataView1.Count = 0 Then
            MessageBox.Show("The search find anything", "There is no item",
MessageBoxButtons.OK, MessageBoxIcon.Information,
MessageBoxDefaultButton.Button1)
            Button33.Text = DataView1.Count & " Records founded "
        ElseIf DataView1.Count > 0 Then
            Me.Text = DataView1.Count & " Records founded "
            Button33.Text = DataView1.Count & " Records founded "
        End If
    Catch ex As Exception
        MsgBox("Please fill the not numeric values")
    End Try
End Sub
```

```
    Private Sub HoverGradientButton10_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles HoverGradientButton10.Click
        Dim f As String
        Dim c As String
        OpenFileDialog1.InitialDirectory = "c:\"
        OpenFileDialog1.Filter = "Jpg files (*.jpg)|*.jpg"
        OpenFileDialog1.ShowDialog()
        If OpenFileDialog1.ShowDialog = DialogResult.OK Then
            f = OpenFileDialog1.FileName.ToString
            Me.Text = f
            Dim dosya As New FileInfo(f)
            c = "c:\proje\pics\customer\" & TextBox1.Text & TextBox2.Text & "." & "jpg"
            TextBox7.Text = c
            dosya.CopyTo(c, True)
            MessageBox.Show("Your Photo saved this " & c & " directory", " Succesfully
added to database", MessageBoxButtons.OK, MessageBoxIcon.Information)
        Else
            MessageBox.Show("Your Photo not saved database succesfully", "Not successful",
MessageBoxButtons.OK, MessageBoxIcon.Warning)
        End If
    End Sub
    Private Sub TextBox10_TextChanged_1(ByVal sender As System.Object, ByVal e As
System.EventArgs)
        a = TextBox10.Text.ToString
        PictureBox1.Image = Image.FromFile(a)
    End Sub
        Private Sub Button13_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)
        Try
            If TextBox9.Text = "" Then
```

```
            MsgBox("Please fill the customer id field", MsgBoxStyle.Exclamation, "Fill
empty space")
            Exit Sub
          End If
          DataView1.RowFilter = "cusid='" & TextBox9.Text & "'"
          If DataView1.Count = 0 Then
            MessageBox.Show("The search find anything", "There is no item",
MessageBoxButtons.OK, MessageBoxIcon.Information,
MessageBoxDefaultButton.Button1)
            Button33.Text = DataView1.Count & " Records founded "
          ElseIf DataView1.Count > 0 Then
            Me.Text = DataView1.Count & " Records founded "
            Button33.Text = DataView1.Count & " Records founded "
          End If
        Catch ex As Exception
          MsgBox("Plase fill the numerical values")
        End Try
      End Sub
      Private Sub Button14_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)
        Try
          If TextBox9.Text = "" Then
            MsgBox("Please fill the customer name field", MsgBoxStyle.Exclamation, "Fill
empty space")
            Exit Sub
          End If
          DataView1.RowFilter = "cusname='" & TextBox9.Text & "'"
          If DataView1.Count = 0 Then
            MessageBox.Show("The search find anything", "There is no item",
MessageBoxButtons.OK, MessageBoxIcon.Information,
MessageBoxDefaultButton.Button1)
```

```
            Button33.Text = DataView1.Count & " Records founded "
        ElseIf DataView1.Count > 0 Then
            Me.Text = DataView1.Count & " Records founded "
            Button33.Text = DataView1.Count & " Records founded "
        End If
    Catch ex As Exception
        MsgBox("Plase fill the not numeric values")
    End Try
End Sub
    c2.Open()
    Dim com1 As New SqlClient.SqlCommand
    Dim dr As SqlClient.SqlDataReader
    com1.CommandText = "select * from sale where customerid='" & TextBox9.Text &
com1.Connection = c2
    dr = com1.ExecuteReader
    say1 = 0
    say2 = 0
    Do While dr.Read
        say1 = say1 + dr("saltotal")
        say2 = say2 + dr("salquantity")
    Loop
    Band2.Visible = True
    Label11.Text = say1 & "  " & "Ytl"
    Label14.Text = say2 & "  " & "Units"
    c2.Close()
    dr.Close()
End Sub
Private Sub Button15_Click_2(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button15.Click
    Try
        If TextBox9.Text = "" Then
```

```
            MsgBox("Please fill the customer name field", MsgBoxStyle.Exclamation, "Fill
empty space")
            Exit Sub
          End If
          DataView1.RowFilter = "cuscompany='" & TextBox9.Text & "'"
          If DataView1.Count = 0 Then
            MessageBox.Show("The search find anything", "There is no item",
MessageBoxButtons.OK, MessageBoxIcon.Information,
MessageBoxDefaultButton.Button1)
            Button33.Text = DataView1.Count & " Records founded "
          ElseIf DataView1.Count > 0 Then
            Me.Text = DataView1.Count & " Records founded "
            Button33.Text = DataView1.Count & " Records founded "
          End If
        Catch ex As Exception
          MsgBox("Please fill the not numeric values")
        End Try
      End Sub
      Private Sub Button16_Click_2(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button16.Click
        DataView1.RowFilter = ""
        If DataView1.Count = 0 Then
          MessageBox.Show("The search find anything", "There is no item",
MessageBoxButtons.OK, MessageBoxIcon.Information,
MessageBoxDefaultButton.Button1)
        ElseIf DataView1.Count > 0 Then
          Me.Text = DataView1.Count & " Records founded "
          Button33.Text = DataView1.Count & " Records founded "
        End If
      End Sub
```

```
    Private Sub TextBox10_TextChanged_2(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles TextBox10.TextChanged
        a = TextBox10.Text.ToString
        PictureBox1.Image = Image.FromFile(a)
    End Sub
        Private Sub TextBox9_MouseDown(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles TextBox9.MouseDown
        TextBox9.ResetText()
    End Sub
End Class
```

FORM 6. SALE FORM

```
Public Class Form6
    Inherits System.Windows.Forms.Form
Public t, a, b, s As Integer
    Public total, guaranty As Integer
    Public v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12 As Integer
    Public stid(12) As Integer
    Public tot As Integer = 0
    Public totq As Integer = 0
    Public tot2 As Integer = 0
    Public cusid As Integer
 Private Sub TabControl1_SelectedIndexChanged(ByVal sender As System.Object, ByVal e
 As System.EventArgs) Handles TabControl1.SelectedIndexChanged
        If TabPage2.Focus = True Then
            Dim com As New SqlClient.SqlCommand
            Dim dr As SqlClient.SqlDataReader
            Try
                c.Open()
                com.CommandText = "select cusname from customer"
```

```
        com.Connection = c
        dr = com.ExecuteReader
        Do While dr.Read
            If ComboBox6.Items.Contains(dr("cusname")) = False Then
                ComboBox6.Items.Add(dr("cusname"))
            End If
        Loop
    Catch ex As Exception
        MsgBox(ex.Message)
    Finally
        c.Close()
        dr.Close()
    End Try
End If
If TabPage2.Focus = True Then
    Dim com As New SqlClient.SqlCommand
    Dim dr2 As SqlClient.SqlDataReader
    Try
        c.Open()
        com.CommandText = "select pcode from product"
        com.Connection = c
        dr2 = com.ExecuteReader
        Do While dr2.Read
            If ComboBox7.Items.Contains(dr2("pcode")) = False Then
                ComboBox7.Items.Add(dr2("pcode"))
            End If
            If ComboBox8.Items.Contains(dr2("pcode")) = False Then
                ComboBox8.Items.Add(dr2("pcode"))
            End If
            If ComboBox9.Items.Contains(dr2("pcode")) = False Then
                ComboBox9.Items.Add(dr2("pcode"))
```

```
        End If
        If ComboBox10.Items.Contains(dr2("pcode")) = False Then
            ComboBox10.Items.Add(dr2("pcode"))
        End If
        If ComboBox11.Items.Contains(dr2("pcode")) = False Then
            ComboBox11.Items.Add(dr2("pcode"))
        End If
        If ComboBox12.Items.Contains(dr2("pcode")) = False Then
            ComboBox12.Items.Add(dr2("pcode"))
        End If
        If ComboBox13.Items.Contains(dr2("pcode")) = False Then
            ComboBox13.Items.Add(dr2("pcode"))
        End If
        If ComboBox14.Items.Contains(dr2("pcode")) = False Then
            ComboBox14.Items.Add(dr2("pcode"))
        End If
        If ComboBox15.Items.Contains(dr2("pcode")) = False Then
            ComboBox15.Items.Add(dr2("pcode"))
        End If
        If ComboBox16.Items.Contains(dr2("pcode")) = False Then
            ComboBox16.Items.Add(dr2("pcode"))
        End If
        If ComboBox17.Items.Contains(dr2("pcode")) = False Then
            ComboBox17.Items.Add(dr2("pcode"))
        End If
        If ComboBox18.Items.Contains(dr2("pcode")) = False Then
            ComboBox18.Items.Add(dr2("pcode"))
        End If
    Loop
Catch ex As Exception
    MsgBox(ex.Message)
```

```
        Finally
            c.Close()
            dr2.Close()
        End Try
    End If
End Sub
Private Sub ComboBox1_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ComboBox1.SelectedIndexChanged
    Dim co As New SqlClient.SqlCommand
    Dim dr As SqlClient.SqlDataReader
    Try
        c.Open()
        co.CommandText = "select * from product"
        co.Connection = c
        dr = co.ExecuteReader
        Do While dr.Read
            If ComboBox2.Text = dr("pcode") And ComboBox1.Text = dr("pname") Then
                TextBox1.Text = dr("pprice")
                a = dr("pprice")
                s = dr("pid")
                TextBox1.Text = dr("pprice")
                total = dr("pprice") * (NumericUpDown2.Value / 100 + 1)
                guaranty = dr("pguaranty")
                TextBox2.Text = guaranty
                TextBox3.Text = total
            End If
        Loop
    Catch ex As Exception
        MsgBox(ex.Message)
    Finally
        c.Close()
```

```
        dr.Close()
      End Try
    End Sub
    Private Sub NumericUpDown1_ValueChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles NumericUpDown1.ValueChanged
      t = a * NumericUpDown1.Value
      TextBox3.Text = (a * (NumericUpDown2.Value / 100 + 1)) *
(NumericUpDown1.Value)
    End Sub
      End Sub
    Private Sub HoverGradientButton2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button14.Click
      NumericUpDown2.Value = 18
      NumericUpDown1.Value = 1
        End Sub
    Private Sub Form6_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
      Dim co As New SqlClient.SqlCommand
      Dim dr4 As SqlClient.SqlDataReader
      Try
        c.Open()
        co.CommandText = "select * from product"
        co.Connection = c
        dr4 = co.ExecuteReader
        Do While dr4.Read
          If ComboBox2.Items.Contains(dr4("pcode")) = False Then
            ComboBox2.Items.Add(dr4("pcode"))
          End If
        Loop
      Catch ex As Exception
        MsgBox(ex.Message)
```

```
    Finally
      c.Close()
      dr4.Close()
    End Try
    Dim com As New SqlClient.SqlCommand
    Dim dr5 As SqlClient.SqlDataReader
    Try
      c.Open()
      com.CommandText = "select cusname from customer"
      com.Connection = c
      dr5 = com.ExecuteReader
      Do While dr5.Read
        If ComboBox3.Items.Contains(dr5("cusname")) = False Then
          ComboBox3.Items.Add(dr5("cusname"))
        End If
      Loop
    Catch ex As Exception
      MsgBox(ex.Message)
    Finally
      c.Close()
      dr5.Close()
    End Try
  End Sub
        Dim co2 As New SqlClient.SqlCommand
      Dim i As Integer = 0
      tarih = DateTimePicker1.Value
      www = tarih.Month & "." & tarih.Day & "." & tarih.Year
      c.Open()
      co2.CommandText = "insert into
sale(customerid,pid,salcode,salname,saldate,salprice,salvat,salquantity,salguaranty)
values(" & t & "," & s & ",'" & ComboBox2.Text & "','" & ComboBox1.Text & "','" &
```

```
www & "'," & a & "," & NumericUpDown2.Value & "," & NumericUpDown1.Value & ","
& guaranty & ")"
        ' t is the customerid
        ' s is the stid
        co2.Connection = c
        i = co2.ExecuteNonQuery
        If i > 0 Then
          MessageBox.Show("Customer Name = " & ComboBox3.Text & " " &
ComboBox4.Text & vbCrLf & "Saled Product Name=" & " " & ComboBox1.Text &
vbCrLf & "Total Amount= " & total, "Succesfully Saled", MessageBoxButtons.OK,
MessageBoxIcon.Information)
        ElseIf i = 0 Then
          MessageBox.Show("olmadı")
        End If
      Catch ex As SqlClient.SqlException
        MessageBox.Show(ex.Message)
      Finally
        c.Close()
      End Try
    End Sub
    Private Sub NumericUpDown2_ValueChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles NumericUpDown2.ValueChanged
      TextBox3.Text = (a * (NumericUpDown2.Value / 100 + 1)) *
(NumericUpDown1.Value)
    End Sub
    Private Sub ComboBox4_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ComboBox4.SelectedIndexChanged
      Dim com As New SqlClient.SqlCommand
      Dim dr As SqlClient.SqlDataReader
      Try
        c.Open()
```

```
        com.CommandText = "select * from customer"
        com.Connection = c
        dr = com.ExecuteReader
        Do While dr.Read
          If ComboBox3.Text = dr("cusname") And ComboBox4.Text = dr("cussurname")
Then
            t = dr("cusid")
          End If
        Loop
      Catch ex As Exception
        MsgBox(ex.Message)
      Finally
        c.Close()
        dr.Close()
      End Try
    End Sub
    Private Sub ComboBox3_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ComboBox3.SelectedIndexChanged
      Dim com As New SqlClient.SqlCommand
      Dim dr As SqlClient.SqlDataReader
      Try
        c.Open()
        com.CommandText = "select * from customer"
        com.Connection = c
        dr = com.ExecuteReader
        Do While dr.Read
          If ComboBox3.Text = dr("cusname") Then
            ComboBox4.Items.Add(dr("cussurname"))
          End If
        Loop
      Catch ex As Exception
```

```
            MsgBox(ex.Message)
        Finally
            c.Close()
            dr.Close()
        End Try
    End Sub
    Private Sub ComboBox6_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ComboBox6.SelectedIndexChanged
        Try
            ComboBox5.Items.Clear()
            ComboBox5.Text = "       .:: Select Customer Surname ::."
            ComboBox5.Focus()
            Dim com As New SqlClient.SqlCommand
            Dim dr As SqlClient.SqlDataReader
            c.Open()
            com.CommandText = "select * from customer"
            com.Connection = c
            dr = com.ExecuteReader
            Do While dr.Read
                If ComboBox6.Text = dr("cusname") Then
                    ComboBox5.Items.Add(dr("cussurname"))
                End If
            Loop
        Catch ex As Exception
            MsgBox(ex.Message)
        Finally
            c.Close()
        End Try
    End Sub
    Private Sub ComboBox5_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ComboBox5.SelectedIndexChanged
```

```
        Dim co As New SqlClient.SqlCommand
        Dim dr As SqlClient.SqlDataReader
        Button20.Visible = False
        TextBox37.Visible = False
        Try
            c.Open()
            co.CommandText = "select * from customer"
            co.Connection = c
            dr = co.ExecuteReader
            Do While dr.Read
                If ComboBox6.Text = dr("cusname") And ComboBox5.Text = dr("cussurname")
Then
                    TextBox10.Text = dr("custaxno")
                End If
            Loop
        Catch ex As Exception
            MsgBox(ex.Message)
        Finally
            c.Close()
            dr.Close()
        End Try
    End Sub
    Private Sub CheckBox4_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles CheckBox4.CheckedChanged
        If CheckBox4.Checked = True Then
            ComboBox10.Enabled = True
        Else
            TextBox26.Enabled = False
        End If
    End Sub
```

```
    Private Sub ComboBox7_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ComboBox7.SelectedIndexChanged
        Dim com As New SqlClient.SqlCommand
        Dim dr As SqlClient.SqlDataReader
        Try
                    c.Open()
          com.CommandText = "select pcode, pname from product"
          com.Connection = c
          dr = com.ExecuteReader
          Do While dr.Read
            If ComboBox7.Text = dr("pcode") Then
              If ComboBox19.Items.Contains(dr("pname")) = False Then
                ComboBox19.Items.Add(dr("pname"))
              End If
            End If
          Loop
        Catch ex As Exception
          MsgBox(ex.Message)
        Finally
          c.Close()
          dr.Close()
        End Try
    End Sub
    Private Sub ComboBox8_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ComboBox8.SelectedIndexChanged
        Dim com As New SqlClient.SqlCommand
        Dim dr As SqlClient.SqlDataReader
        Try
          c.Open()
          com.CommandText = "select pcode, pname from product"
          com.Connection = c
```

```
        dr = com.ExecuteReader
        Do While dr.Read
            If ComboBox8.Text = dr("pcode") Then
                If ComboBox20.Items.Contains(dr("pname")) = False Then
                    ComboBox20.Items.Add(dr("pname"))
                End If
            End If
        Loop
    Catch ex As Exception
        MsgBox(ex.Message)
    Finally
        c.Close()
        dr.Close()
    End Try
End Sub
Private Sub ComboBox9_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ComboBox9.SelectedIndexChanged
    Dim com As New SqlClient.SqlCommand
    Dim dr As SqlClient.SqlDataReader
    Try
        c.Open()
        com.CommandText = "select pcode, pname from product"
        com.Connection = c
        dr = com.ExecuteReader
        Do While dr.Read
            If ComboBox9.Text = dr("pcode") Then
                If ComboBox21.Items.Contains(dr("pname")) = False Then
                    ComboBox21.Items.Add(dr("pname"))
                End If
            End If
        Loop
```

```
        Catch ex As Exception
            MsgBox(ex.Message)
        Finally
            c.Close()
            dr.Close()
        End Try
    End Sub
    Private Sub ComboBox10_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ComboBox10.SelectedIndexChanged
        Dim com As New SqlClient.SqlCommand
        Dim dr As SqlClient.SqlDataReader
        Try
            c.Open()
            com.CommandText = "select pcode, pname from product"
            com.Connection = c
            dr = com.ExecuteReader
            Do While dr.Read
                If ComboBox10.Text = dr("pcode") Then
                    If ComboBox22.Items.Contains(dr("pname")) = False Then
                        ComboBox22.Items.Add(dr("pname"))
                    End If
                End If
            Loop
        Catch ex As Exception
            MsgBox(ex.Message)
        Finally
            c.Close()
            dr.Close()
        End Try
    End Sub
```

```
Private Sub ComboBox11_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ComboBox11.SelectedIndexChanged
    Dim com As New SqlClient.SqlCommand
    Dim dr As SqlClient.SqlDataReader
    Try
      c.Open()
      com.CommandText = "select pcode, pname from product"
      com.Connection = c
      dr = com.ExecuteReader
      Do While dr.Read
        If ComboBox11.Text = dr("pcode") Then
          If ComboBox23.Items.Contains(dr("pname")) = False Then
            ComboBox23.Items.Add(dr("pname"))
          End If
        End If
      Loop
    Catch ex As Exception
      MsgBox(ex.Message)
    Finally
      c.Close()
      dr.Close()
    End Try
  End Sub
  Private Sub ComboBox12_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ComboBox12.SelectedIndexChanged
    Dim com As New SqlClient.SqlCommand
    Dim dr As SqlClient.SqlDataReader
    Try
      c.Open()
      com.CommandText = "select pcode, pname from product"
      com.Connection = c
```

```
            dr = com.ExecuteReader
            Do While dr.Read
                If ComboBox12.Text = dr("pcode") Then
                    If ComboBox24.Items.Contains(dr("pname")) = False Then
                        ComboBox24.Items.Add(dr("pname"))
                    End If
                End If
            Loop
        Catch ex As Exception
            MsgBox(ex.Message)
        Finally
            c.Close()
            dr.Close()
        End Try
    End Sub
    Private Sub ComboBox13_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ComboBox13.SelectedIndexChanged
        Dim com As New SqlClient.SqlCommand
        Dim dr As SqlClient.SqlDataReader
        Try
            c.Open()
            com.CommandText = "select pcode, pname from product"
            com.Connection = c
            dr = com.ExecuteReader
            Do While dr.Read
                If ComboBox13.Text = dr("pcode") Then
                    If ComboBox25.Items.Contains(dr("pname")) = False Then
                        ComboBox25.Items.Add(dr("pname"))
                    End If
                End If
            Loop
```

```
        Catch ex As Exception
            MsgBox(ex.Message)
        Finally
            c.Close()
            dr.Close()
        End Try
    End Sub
    Private Sub ComboBox14_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ComboBox14.SelectedIndexChanged
        Dim com As New SqlClient.SqlCommand
        Dim dr As SqlClient.SqlDataReader
        Try
            c.Open()
            com.CommandText = "select pcode, pname from product"
            com.Connection = c
            dr = com.ExecuteReader
            Do While dr.Read
                If ComboBox14.Text = dr("pcode") Then
                    If ComboBox26.Items.Contains(dr("pname")) = False Then
                        ComboBox26.Items.Add(dr("pname"))
                    End If
                End If
            Loop
        Catch ex As Exception
            MsgBox(ex.Message)
        Finally
            c.Close()
            dr.Close()
        End Try
    End Sub
```

```
Private Sub ComboBox15_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ComboBox15.SelectedIndexChanged
    Dim com As New SqlClient.SqlCommand
    Dim dr As SqlClient.SqlDataReader
    Try
      c.Open()
      com.CommandText = "select pcode, pname from product"
      com.Connection = c
      dr = com.ExecuteReader
      Do While dr.Read
        If ComboBox15.Text = dr("pcode") Then
          If ComboBox27.Items.Contains(dr("pname")) = False Then
            ComboBox27.Items.Add(dr("pname"))
          End If
        End If
      Loop
    Catch ex As Exception
      MsgBox(ex.Message)
    Finally
      c.Close()
      dr.Close()
    End Try
  End Sub
  Private Sub ComboBox16_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ComboBox16.SelectedIndexChanged
    Dim com As New SqlClient.SqlCommand
    Dim dr As SqlClient.SqlDataReader
    Try
      c.Open()
      com.CommandText = "select pcode, pname from product"
      com.Connection = c
```

```
        dr = com.ExecuteReader
        Do While dr.Read
            If ComboBox16.Text = dr("pcode") Then
                If ComboBox28.Items.Contains(dr("pname")) = False Then
                    ComboBox28.Items.Add(dr("pname"))
                End If
            End If
        Loop
    Catch ex As Exception
        MsgBox(ex.Message)
    Finally
        c.Close()
        dr.Close()
    End Try
End Sub
Private Sub ComboBox17_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ComboBox17.SelectedIndexChanged
    Dim com As New SqlClient.SqlCommand
    Dim dr As SqlClient.SqlDataReader
    Try
        c.Open()
        com.CommandText = "select pcode, pname from product"
        com.Connection = c
        dr = com.ExecuteReader
        Do While dr.Read
            If ComboBox17.Text = dr("pcode") Then
                If ComboBox29.Items.Contains(dr("pname")) = False Then
                    ComboBox29.Items.Add(dr("pname"))
                End If
            End If
        Loop
```

```
        Catch ex As Exception
            MsgBox(ex.Message)
        Finally
            c.Close()
            dr.Close()
        End Try
    End Sub
    Private Sub ComboBox18_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ComboBox18.SelectedIndexChanged
        Dim com As New SqlClient.SqlCommand
        Dim dr As SqlClient.SqlDataReader
        Try
            c.Open()
            com.CommandText = "select pcode, pname from product"
            com.Connection = c
            dr = com.ExecuteReader
            Do While dr.Read
                If ComboBox18.Text = dr("pcode") Then
                    If ComboBox30.Items.Contains(dr("pname")) = False Then
                        ComboBox30.Items.Add(dr("pname"))
                    End If
                End If
            Loop
        Catch ex As Exception
            MsgBox(ex.Message)
        Finally
            c.Close()
            dr.Close()
        End Try
    End Sub
```

```
    Private Sub ComboBox19_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ComboBox19.SelectedIndexChanged
        Dim com As New SqlClient.SqlCommand
        Dim dr As SqlClient.SqlDataReader
        Try
            NumericUpDown3.Enabled = True
            TextBox11.Enabled = True
            TextBox23.Enabled = True
            c.Open()
            com.CommandText = "select * from product"
            com.Connection = c
            dr = com.ExecuteReader
            Do While dr.Read
                If ComboBox7.Text = dr("pcode") And ComboBox19.Text = dr("pname") Then
                    NumericUpDown3.Value = 1
                    v1 = dr("pprice")
                    stid(1) = dr("pid")
                    TextBox23.Text = dr("pguaranty")
                End If
            Loop
            TextBox11.Text = v1
        Catch ex As Exception
            MsgBox(ex.Message)
        Finally
            c.Close()
            dr.Close()
        End Try
    End Sub
    Private Sub ComboBox20_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ComboBox20.SelectedIndexChanged
        Dim com As New SqlClient.SqlCommand
```

```
        Dim dr As SqlClient.SqlDataReader
        Try
            c.Open()
            com.CommandText = "select * from product"
            com.Connection = c
            dr = com.ExecuteReader
            Do While dr.Read
                If ComboBox8.Text = dr("pcode") And ComboBox20.Text = dr("pname") Then
                    NumericUpDown4.Value = 1
                    v2 = dr("pprice")
                    TextBox24.Text = dr("pguaranty")
                    stid(2) = dr("pid")
                End If
            Loop
            TextBox12.Text = v2
        Catch ex As Exception
            MsgBox(ex.Message)
        Finally
            c.Close()
            dr.Close()
        End Try
    End Sub
    Private Sub ComboBox21_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ComboBox21.SelectedIndexChanged
        Dim com As New SqlClient.SqlCommand
        Dim dr As SqlClient.SqlDataReader
        Try
            NumericUpDown5.Enabled = True
            c.Open()
            com.CommandText = "select * from product"
            com.Connection = c
```

```
            dr = com.ExecuteReader
            Do While dr.Read
                If ComboBox9.Text = dr("pcode") And ComboBox21.Text = dr("pname") Then
                    NumericUpDown5.Value = 1
                    v3 = dr("pprice")
                    TextBox25.Text = dr("pguaranty")
                    stid(3) = dr("pid")
                End If
            Loop
            TextBox13.Text = v3
        Catch ex As Exception
            MsgBox(ex.Message)
        Finally
            c.Close()
            dr.Close()
        End Try
    End Sub
    Private Sub ComboBox22_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ComboBox22.SelectedIndexChanged
        Dim com As New SqlClient.SqlCommand
        Dim dr As SqlClient.SqlDataReader
        Try
            c.Open()
            com.CommandText = "select * from product"
            com.Connection = c
            dr = com.ExecuteReader
            Do While dr.Read
                If ComboBox10.Text = dr("pcode") And ComboBox22.Text = dr("pname") Then
                    NumericUpDown6.Value = 1
                    v4 = dr("pprice")
                    TextBox26.Text = dr("pguaranty")
```

```
            stid(4) = dr("pid")
          End If
        Loop
        TextBox14.Text = v4
      Catch ex As Exception
        MsgBox(ex.Message)
      Finally
        c.Close()
        dr.Close()
      End Try
    End Sub
    Private Sub ComboBox23_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ComboBox23.SelectedIndexChanged
      Dim com As New SqlClient.SqlCommand
      Dim dr As SqlClient.SqlDataReader
      Try
        c.Open()
        com.CommandText = "select * from product"
        com.Connection = c
        dr = com.ExecuteReader
        Do While dr.Read
          If ComboBox11.Text = dr("pcode") And ComboBox23.Text = dr("pname") Then
            NumericUpDown7.Value = 1
            v5 = dr("pprice")
            TextBox27.Text = dr("pguaranty")
            stid(5) = dr("pid")
          End If
        Loop
        TextBox15.Text = v5
      Catch ex As Exception
        MsgBox(ex.Message)
```

```
        Finally
            c.Close()
            dr.Close()
        End Try
    End Sub
    Private Sub ComboBox24_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ComboBox24.SelectedIndexChanged
        Dim com As New SqlClient.SqlCommand
        Dim dr As SqlClient.SqlDataReader
        Try
            c.Open()
            com.CommandText = "select * from product"
            com.Connection = c
            dr = com.ExecuteReader
            Do While dr.Read
                If ComboBox12.Text = dr("pcode") And ComboBox24.Text = dr("pname") Then
                    NumericUpDown8.Value = 1
                    v6 = dr("pprice")
                    TextBox28.Text = dr("pguaranty")
                    stid(6) = dr("pid")
                End If
            Loop
            TextBox16.Text = v6
        Catch ex As Exception
            MsgBox(ex.Message)
        Finally
            c.Close()
            dr.Close()
        End Try
    End Sub
```

```
Private Sub ComboBox27_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ComboBox27.SelectedIndexChanged
    Dim com As New SqlClient.SqlCommand
    Dim dr As SqlClient.SqlDataReader
    Try
            c.Open()
      com.CommandText = "select * from product"
      com.Connection = c
      dr = com.ExecuteReader
      Do While dr.Read
        If ComboBox15.Text = dr("pcode") And ComboBox27.Text = dr("pname") Then
          NumericUpDown11.Value = 1
          v9 = dr("pprice")
          TextBox31.Text = dr("pguaranty")
          stid(9) = dr("pid")
        End If
      Loop
      TextBox19.Text = v9
    Catch ex As Exception
      MsgBox(ex.Message)
    Finally
      c.Close()
      dr.Close()
    End Try
  End Sub
  Private Sub ComboBox28_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ComboBox28.SelectedIndexChanged
    Dim com As New SqlClient.SqlCommand
    Dim dr As SqlClient.SqlDataReader
    Try
      c.Open()
```

```
            com.CommandText = "select * from product"
            com.Connection = c
            dr = com.ExecuteReader
            Do While dr.Read
                If ComboBox16.Text = dr("pcode") And ComboBox28.Text = dr("pname") Then
                    NumericUpDown12.Value = 1
                    v10 = dr("pprice")
                    TextBox32.Text = dr("pguaranty")
                    stid(10) = dr("pid")
                End If
            Loop
            TextBox20.Text = v10
        Catch ex As Exception
            MsgBox(ex.Message)
        Finally
            c.Close()
            dr.Close()
        End Try
    End Sub
    Private Sub ComboBox29_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ComboBox29.SelectedIndexChanged
        Dim com As New SqlClient.SqlCommand
        Dim dr As SqlClient.SqlDataReader
        Try
            c.Open()
            com.CommandText = "select * from product"
            com.Connection = c
            dr = com.ExecuteReader
            Do While dr.Read
                If ComboBox17.Text = dr("pcode") And ComboBox29.Text = dr("pname") Then
                    NumericUpDown13.Value = 1
```

```
                v11 = dr("pprice")
                TextBox33.Text = dr("pguaranty")
                stid(11) = dr("pid")
            End If
        Loop
        TextBox21.Text = v11
    Catch ex As Exception
        MsgBox(ex.Message)
    Finally
        c.Close()
        dr.Close()
    End Try
End Sub
Private Sub ComboBox30_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ComboBox30.SelectedIndexChanged
    Dim com As New SqlClient.SqlCommand
    Dim dr As SqlClient.SqlDataReader
    Try
        com.CommandText = "select * from product"
        com.Connection = c
        dr = com.ExecuteReader
        Do While dr.Read
            If ComboBox18.Text = dr("pcode") And ComboBox30.Text = dr("pname") Then
                NumericUpDown14.Value = 1
                v12 = dr("pprice")
                TextBox34.Text = dr("pguaranty")
                stid(12) = dr("pid")
            End If
        Loop
        TextBox22.Text = v12
    Catch ex As Exception
```

```
            MsgBox(ex.Message)
        Finally
            c.Close()
            dr.Close()
        End Try
    End Sub
    Private Sub TextBox35_TextChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles TextBox35.TextChanged
        tot2 = tot + tot * 0.18
        TextBox36.Text = tot2
    End Sub
    Private Sub Button13_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button13.Click
        tot = 0
        If CheckBox1.Checked = True Then
            tot = tot + v1 * NumericUpDown3.Value
            TextBox35.Text = tot
        End If
        If CheckBox2.Checked = True Then
            tot = tot + v2 * NumericUpDown4.Value
            TextBox35.Text = tot
        End If
        If CheckBox3.Checked = True Then
            tot = tot + v3 * NumericUpDown5.Value
            TextBox35.Text = tot
        End If
        If CheckBox4.Checked = True Then
            tot = tot + v4 * NumericUpDown6.Value
            TextBox35.Text = tot
        End If
        If CheckBox5.Checked = True Then
```

```
    tot = tot + v5 * NumericUpDown7.Value
    TextBox35.Text = tot
End If
If CheckBox6.Checked = True Then
    tot = tot + v6 * NumericUpDown8.Value
    TextBox35.Text = tot
End If
If CheckBox7.Checked = True Then
    tot = tot + v7 * NumericUpDown9.Value
    TextBox35.Text = tot
End If
If CheckBox8.Checked = True Then
    tot = tot + v8 * NumericUpDown10.Value
    TextBox35.Text = tot
End If
If CheckBox9.Checked = True Then
    tot = tot + v9 * NumericUpDown11.Value
    TextBox35.Text = tot
End If
If CheckBox10.Checked = True Then
    tot = tot + v10 * NumericUpDown12.Value
    TextBox35.Text = tot
End If
If CheckBox11.Checked = True Then
    tot = tot + v11 * NumericUpDown13.Value
    TextBox35.Text = tot
End If
If CheckBox12.Checked = True Then
    tot = tot + v12 * NumericUpDown14.Value
    TextBox35.Text = tot
End If
```

```
End Sub
Private Sub HoverButtonBand2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles HoverButtonBand2.Click
End Sub
Private Sub Button12_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button12.Click
    Dim mes As String
    tarih = DateTimePicker2.Value
    www = tarih.Month & "." & tarih.Day & "." & tarih.Year
    Dim i(12) As Integer
    If ComboBox5.Text = " .:: Select Customer Surname ::." Or ComboBox6.Text = "    .::
Select Customer Name ::." Then
      MessageBox.Show("Please, select or fill the empty fields carefully", "Attention
Please", MessageBoxButtons.OK, MessageBoxIcon.Information)
      Exit Sub
    End If
    Try
      c.Open()
      If CheckBox1.Checked = True Then
        Dim co1 As New SqlClient.SqlCommand

        co1.CommandText = "insert into
sale(customerid,pid,salcode,salname,saldate,salprice,salquantity,salguaranty) values(" &
cusid & "," & stid(1) & ",'" & ComboBox7.Text & "','" & ComboBox19.Text & "','" &
www & "'," & TextBox11.Text & "," & NumericUpDown3.Value & "," & TextBox23.Text
& ")"
        co1.Connection = c
        i(1) = co1.ExecuteNonQuery
        If i(1) > 0 Then
          mes = mes & ", " & ComboBox19.Text
        End If
```

```
            End If
            If CheckBox2.Checked = True Then
               Dim co1 As New SqlClient.SqlCommand
               co1.CommandText = "insert into
sale(customerid,pid,salcode,salname,saldate,salprice,salquantity,salguaranty) values(" &
cusid & "," & stid(2) & ",'" & ComboBox8.Text & "','" & ComboBox20.Text & "','" &
www & "'," & TextBox12.Text & "," & NumericUpDown4.Value & "," & TextBox24.Text
& ")"
               co1.Connection = c
               i(2) = co1.ExecuteNonQuery
               If i(2) > 0 Then
                  mes = mes & ", " & ComboBox20.Text
               End If
            End If
            If CheckBox3.Checked = True Then
               Dim co1 As New SqlClient.SqlCommand
               co1.CommandText = "insert into
sale(customerid,pid,salcode,salname,saldate,salprice,salquantity,salguaranty) values(" &
cusid & "," & stid(3) & ",'" & ComboBox9.Text & "','" & ComboBox21.Text & "','" &
www & "'," & TextBox13.Text & "," & NumericUpDown5.Value & "," & TextBox25.Text
& ")"
               co1.Connection = c
               i(3) = co1.ExecuteNonQuery
               If i(3) > 0 Then
                  mes = mes & ", " & ComboBox21.Text
               End If
            End If
                  If CheckBox4.Checked = True Then
               Dim co1 As New SqlClient.SqlCommand
               co1.CommandText = "insert into
sale(customerid,pid,salcode,salname,saldate,salprice,salquantity,salguaranty) values(" &
```

```
cusid & "," & stid(4) & ",'" & ComboBox10.Text & "','" & ComboBox22.Text & "','" &
www & "'," & TextBox14.Text & "," & NumericUpDown6.Value & "," & TextBox26.Text
& ")"
            co1.Connection = c
            i(4) = co1.ExecuteNonQuery
            If i(4) > 0 Then
               mes = mes & ", " & ComboBox22.Text
            End If
         End If
         If CheckBox5.Checked = True Then
            Dim co1 As New SqlClient.SqlCommand
            co1.CommandText = "insert into
sale(customerid,pid,salcode,salname,saldate,salprice,salquantity,salguaranty) values(" &
cusid & "," & stid(5) & ",'" & ComboBox11.Text & "','" & ComboBox23.Text & "','" &
www & "'," & TextBox15.Text & "," & NumericUpDown7.Value & "," & TextBox27.Text
& ")"
            co1.Connection = c
            i(5) = co1.ExecuteNonQuery
            If i(5) > 0 Then
               mes = mes & ", " & ComboBox23.Text
            End If
         End If
         If CheckBox6.Checked = True Then
            Dim co1 As New SqlClient.SqlCommand
            co1.CommandText = "insert into
sale(customerid,pid,salcode,salname,saldate,salprice,salquantity,salguaranty) values(" &
cusid & "," & stid(6) & ",'" & ComboBox12.Text & "','" & ComboBox24.Text & "','" &
www & "'," & TextBox16.Text & "," & NumericUpDown8.Value & "," & TextBox28.Text
& ")"
            co1.Connection = c
            i(6) = co1.ExecuteNonQuery
```

```
            If i(6) > 0 Then
                mes = mes & ", " & ComboBox23.Text
            End If
        End If
                If CheckBox7.Checked = True Then
            Dim co1 As New SqlClient.SqlCommand
            co1.CommandText = "insert into
sale(customerid,pid,salcode,salname,saldate,salprice,salquantity,salguaranty) values(" &
cusid & "," & stid(7) & ",'" & ComboBox13.Text & "','" & ComboBox25.Text & "','" &
www & "'," & TextBox17.Text & "," & NumericUpDown9.Value & "," & TextBox29.Text
& ")"
            co1.Connection = c
            i(7) = co1.ExecuteNonQuery
            If i(7) > 0 Then
                mes = mes & ", " & ComboBox25.Text
            End If
        End If
                If CheckBox8.Checked = True Then
            Dim co1 As New SqlClient.SqlCommand

            co1.CommandText = "insert into
sale(customerid,pid,salcode,salname,saldate,salprice,salquantity,salguaranty) values(" &
cusid & "," & stid(8) & ",'" & ComboBox14.Text & "','" & ComboBox26.Text & "','" &
www & "'," & TextBox18.Text & "," & NumericUpDown10.Value & "," &
TextBox30.Text & ")"
            co1.Connection = c
            i(8) = co1.ExecuteNonQuery
            If i(8) > 0 Then
                mes = mes & ", " & ComboBox26.Text
            End If
        End If
```

```
        If CheckBox9.Checked = True Then
            Dim co1 As New SqlClient.SqlCommand
            co1.CommandText = "insert into
sale(customerid,pid,salcode,salname,saldate,salprice,salquantity,salguaranty) values(" &
cusid & "," & stid(9) & ",'" & ComboBox15.Text & "','" & ComboBox27.Text & "','" &
www & "'," & TextBox19.Text & "," & NumericUpDown11.Value & "," &
TextBox31.Text & ")"
            co1.Connection = c
            i(9) = co1.ExecuteNonQuery
            If i(9) > 0 Then
                mes = mes & ", " & ComboBox26.Text
            End If
        End If
        If CheckBox10.Checked = True Then
            Dim co1 As New SqlClient.SqlCommand
            co1.CommandText = "insert into
sale(customerid,pid,salcode,salname,saldate,salprice,salquantity,salguaranty) values(" &
cusid & "," & stid(10) & ",'" & ComboBox16.Text & "','" & ComboBox28.Text & "','" &
www & "'," & TextBox20.Text & "," & NumericUpDown12.Value & "," &
TextBox32.Text & ")"
            co1.Connection = c
            i(10) = co1.ExecuteNonQuery
            If i(10) > 0 Then
                mes = mes & ", " & ComboBox28.Text
            End If
        End If
        If CheckBox11.Checked = True Then
            Dim co1 As New SqlClient.SqlCommand
            co1.CommandText = "insert into
sale(customerid,pid,salcode,salname,saldate,salprice,salquantity,salguaranty) values(" &
cusid & "," & stid(11) & ",'" & ComboBox17.Text & "','" & ComboBox29.Text & "','" &
```

```
www & "'," & TextBox21.Text & "," & NumericUpDown13.Value & "," &
TextBox33.Text & ")"
            co1.Connection = c
            i(11) = co1.ExecuteNonQuery
            If i(11) > 0 Then
               mes = mes & ", " & ComboBox29.Text
            End If
         End If
         If CheckBox12.Checked = True Then
            Dim co1 As New SqlClient.SqlCommand
            co1.CommandText = "insert into
sale(customerid,pid,salcode,salname,saldate,salprice,salquantity,salguaranty) values(" &
cusid & "," & stid(12) & ",'" & ComboBox18.Text & "','" & ComboBox30.Text & "','" &
www & "'," & TextBox22.Text & "," & NumericUpDown14.Value & "," &
TextBox34.Text & ")"
            co1.Connection = c
            i(12) = co1.ExecuteNonQuery
            If i(12) > 0 Then
               mes = mes & ", " & ComboBox30.Text
            End If
         End If
         MessageBox.Show(mes, "Succesfully Saled", MessageBoxButtons.OK,
MessageBoxIcon.Information)
      Catch ex As SqlClient.SqlException
         MessageBox.Show(ex.Message)
      Finally
         c.Close()
      End Try
   End Sub
End Class
```

FORM 7. REPORT FORM

```
Imports CrystalDecisions.CrystalReports.Engine
Imports CrystalDecisions.Shared

Public Class Form7
    Inherits System.Windows.Forms.Form
Public st, en As DateTime
    Public xx As New cr1
    Public yy As New cr2
    Private Sub Button8_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button8.Click
        If MessageBox.Show("Are you sure to close the program", "System Exit",
MessageBoxButtons.YesNo, MessageBoxIcon.Asterisk,
MessageBoxDefaultButton.Button1) = DialogResult.Yes Then
            Application.Exit()
        End If
    End Sub
    Private Sub TabControl1_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles TabControl1.SelectedIndexChanged
        If TabPage1.Focus = True Then
            CrystalReportViewer1.ReportSource = xx
            Dim z As New CrystalDecisions.Shared.ParameterValues
            Dim z1 As New CrystalDecisions.Shared.ParameterDiscreteValue
            Dim urun = tarih
            z1.Value = urun
            z.Add(z1)
            xx.DataDefinition.ParameterFields("@tarih").ApplyCurrentValues(z)
            CrystalReportViewer1.ReportSource = xx
        End If
```

```
        If TabPage2.Focus = True Then
            st = DateTimePicker1.Value
            en = DateTimePicker2.Value
            Dim z As New CrystalDecisions.Shared.ParameterValues
            Dim zz As New CrystalDecisions.Shared.ParameterValues
            Dim z1 As New CrystalDecisions.Shared.ParameterDiscreteValue
            Dim z2 As New CrystalDecisions.Shared.ParameterDiscreteValue
            Dim urun1 = st.ToShortDateString
            Dim urun2 = en.ToShortDateString
            z1.Value = urun1
            z2.Value = urun2
            z.Add(z1)
            zz.Add(z2)
            yy.DataDefinition.ParameterFields("@tarih1").ApplyCurrentValues(z)
            yy.DataDefinition.ParameterFields("@tarih2").ApplyCurrentValues(zz)
            CrystalReportViewer2.ReportSource = yy
        End If
    End Sub
    Private Sub Form7_Load(ByVal sender As Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        tarih = Now.ToShortDateString
        st = DateTimePicker1.Value
        en = DateTimePicker2.Value
    End Sub
        Private Sub HoverGradientButton1_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles Button12.Click
        st = DateTimePicker1.Value
        en = DateTimePicker2.Value
        Dim yy As New cr2
        Dim z As New CrystalDecisions.Shared.ParameterValues
        Dim zz As New CrystalDecisions.Shared.ParameterValues
```

```
        Dim z1 As New CrystalDecisions.Shared.ParameterDiscreteValue
        Dim z2 As New CrystalDecisions.Shared.ParameterDiscreteValue
        Dim urun1 = st.ToShortDateString
        Dim urun2 = en.ToShortDateString
        z1.Value = urun1
        z2.Value = urun2
        z.Add(z1)
        zz.Add(z2)
        yy.DataDefinition.ParameterFields("@tarih1").ApplyCurrentValues(z)
        yy.DataDefinition.ParameterFields("@tarih2").ApplyCurrentValues(zz)
        CrystalReportViewer2.ReportSource = yy
    End Sub
End Class
```

FORM 8. REGISTRATION FORM

```
Imports Microsoft.Win32
Public Class Form8
    Inherits System.Windows.Forms.Form
Private Sub Button11_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button11.Click
        If TextBox1.Text = "" Then
            MessageBox.Show("Please fill the serial number", "Fill the serial",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
            Exit Sub
        End If
        If TextBox1.Text <> "edimspeed2005" Then
            MessageBox.Show(TextBox1.Text & " is not valid serial number", "Enter correct
serial number", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
            TextBox1.Focus()
        ElseIf TextBox1.Text = "edimspeed2005" Then
```

```
        p1.Timer1.Dispose()
        MessageBox.Show("You registered successfully", "Thank you, to use full version",
MessageBoxButtons.OK, MessageBoxIcon.Information)
        Registry.ClassesRoot.SetValue("edimd", "edimspeed2005")
        ButtonBand2.Text = "Thank you for registration, Registered version"
        TextBox1.Enabled = False
        Button9.Visible = True
                p8.TextBox1.Text = "Registered Full Vesion"
    End If
  End Sub
  End Class
```

FORM 9. MAIN MENU

```
Public Class Form9
  Inherits System.Windows.Forms.Form
      End Class
```

FORM 10. STOCK CONTROL FORM

```
Public Class Form10
  Inherits System.Windows.Forms.Form
Private Sub Form10_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    da.Fill(ds.c)
        End Sub
      Private Sub HoverGradientButton3_Click_1(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles HoverGradientButton3.Click
    Dim buy As Integer
    buy = TextBox8.Text
```

```
        tarih = DateTimePicker1.Value
        www = tarih.Month & "." & tarih.Day & "." & tarih.Year
        sd3 = "C:\proje\pics\stock\" & TextBox6.Text & "." & "jpg"
        Try
            Dim c As New SqlClient.SqlConnection
            c.ConnectionString = "data source=asusa6vq;initial catalog=adem;integrated
security=true"
            Dim co As New SqlClient.SqlCommand
            Dim i As Integer = 0
            c.Open()
            co.CommandText = "insert into
stock(pid,stcode,stname,stbuy,stcompany,stguaranty,stdate,stquantity,stimage) values(" &
TextBox5.Text & ",'" & TextBox6.Text & "','" & TextBox7.Text & "'," & buy & ",'" &
ComboBox1.Text & "'," & NumericUpDown2.Value & ",'" & www & "'," &
NumericUpDown1.Value & ",'" & sd3 & "')"
            co.Connection = c
            If TextBox8.Text = "" Then
                MessageBox.Show("Please insert the product price", "Fill the product price",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
                TextBox8.Focus()
            End If
            i = co.ExecuteNonQuery()
            If i > 0 Then
                MessageBox.Show(TextBox6.Text & "  " & TextBox7.Text & vbCrLf &
"Inserted succesfully", "Inserted Succesfully", MessageBoxButtons.OK,
MessageBoxIcon.Information)
            ElseIf i = 0 Then
                MessageBox.Show("Can't make Insert operation", "Not Succesfull",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
            End If
```

```
    Catch ex As SqlClient.SqlException
      MsgBox(ex.Message)
    Finally
      c.Close()
      ds.Clear()
      da.Fill(ds.c)
    End Try
  End Sub
  Private Sub HoverGradientButton2_Click_1(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles HoverGradientButton2.Click
    Dim co As New SqlClient.SqlCommand
    Dim dr As SqlClient.SqlDataReader
    Try
      c.Open()
      co.CommandText = "select * from company"
      co.Connection = c
      dr = co.ExecuteReader
      Do While dr.Read
        ComboBox1.Items.Add(dr("comname"))
      Loop
    Catch ex As Exception
      MsgBox(ex.Message)
    Finally
      c.Close()
      dr.Close()
      ComboBox1.Text = "      .:: Select the company name ::."
      GroupBox2.Enabled = True
    End Try
  End Sub
End Class
```

FORM 11. GUARANTY VALIDATE FORM

```
Public Class Form11
   Inherits System.Windows.Forms.Form
Private Sub Form11_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
      dasale.Fill(dssale.sale)
   End Sub
      Private Sub HoverGradientButton1_Click(ByVal sender As System.Object, ByVal
As System.EventArgs) Handles HoverGradientButton1.Click
      Dim co As New SqlClient.SqlCommand
      Dim dr As SqlClient.SqlDataReader
      Dim i As Integer = 0
      Try
         co.Connection = c
         co.CommandText = "select * from sale"
         c.Open()
         dr = co.ExecuteReader
         If TextBox1.Text = "" Then
            MessageBox.Show("Please fill the Sale Id", "Please Fill the Sale Id",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk)
            Exit Sub
         End If
         Do While dr.Read
            If TextBox1.Text = dr("salid") Then
               tarih = dr("saldate")
               say1 = dr("salguaranty")
               sd2 = dr("salid")
               Exit Do
            End If
         Loop
```

```
            tarih = DateAdd(DateInterval.Year, say1, tarih)
            tarih2 = Now.ToShortDateString
            If tarih > Now.ToShortDateString Then
                sd3 = DateDiff(DateInterval.Day, tarih2, tarih)
                MessageBox.Show("Sale id= " & sd2 & " product guaranty is continue" &
vbCrLf & "The Remainig Quaranty Days = " & sd3, "The guaranty is valid",
MessageBoxButtons.OK, MessageBoxIcon.Information)
            Else
                MessageBox.Show("Sale id=" & sd2 & " have not guaranty", "The guaranty
duration finished", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
            End If
        Catch ex As SqlClient.SqlException
            MsgBox(ex.Message)
        Finally
            dr.Close()
            c.Close()
        End Try
    End Sub
End Class
```

FORM 12. EXTRAS FORM

```
Public Class Form12
    Inherits System.Windows.Forms.Form
Private Sub Form12_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Try
            dacus.Fill(dscus.custotal)
            dapid.Fill(dscus.salepid)
            dastock.Fill(dscus.stockpid)
            dafark.Fill(dscus.c)
```

```
    Catch ex As SqlClient.SqlException
      MsgBox(ex.Message)
    End Try
  End Sub
```

FORM 13. ABOUT FORM

```
Public Class Form13
  Inherits System.Windows.Forms.Form
  Private Sub MenuItem10_Click_1(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MenuItem10.Click
    s2 = Now
    sd = DateDiff(DateInterval.Hour, s1, s2)
    sd2 = DateDiff(DateInterval.Minute, s1, s2)
    sd3 = DateDiff(DateInterval.Second, s1, s2)
    sd4 = "The program was started " & sd & " hours" & " " & sd2 & " " & "minutes " &
sd3 & " " & " seconds ago"
    MessageBox.Show(sd4, "Edimspeed Stock 2005 uptime", MessageBoxButtons.OK,
MessageBoxIcon.Information)
  End Sub
End Class
```

APPENDIX B: SQL DATABASE TABLES

Column Name	Data Type	Length	Allow Nulls
comid	int	4	
comname	nvarchar	50	
comphone	nvarchar	50	✓
comfax	nvarchar	50	✓
comweb	nvarchar	50	✓
commail	nvarchar	50	✓
comadress	nvarchar	70	✓
comimage	char	50	✓

Fig. 4.1 Company Design Table

comid	comname	comphone	comfax	comweb	commail	comadress	comimage
6	penta	4453234	4565432	www.penta.com.tr	penta@penta.com.	cevizlibag ishanı ka	C:\proje\pics\logo\
7	koyuncu	8986787	7672312	www.koyuncu.com	koyuncu@koyuncu	istikamet ishanı tah	C:\proje\pics\logo\
8	hızlısistem	1212323	7878899	www.hizlisistem.co	hizlisistem@hizlisist	tecnology işhanı me	C:\proje\pics\logo\
11	vatan	6654676	5545666	www.vatanbilgisay	vatan@vatan.com.	pc işhanı mecidiyek	C:\proje\pics\logo\
12	slayt	3434322	3432345	www.slayt.com.tr	slayt@slayt.com.tr	medaş karşısı adalh	C:\proje\pics\logo\
13	gold	3466787	1090987	www.gold.com.tr	gold@gold.com.tr	cambaz işhanı vahc	C:\proje\pics\logo\
14	bogaziçi	02122344543	87878	www.bogazici.com.	bogazici@bogazici.c	bogazici işhanı meci	C:\proje\pics\logo\
15	somya	343	344	www.somya.com.tr	somya@somya.con	ist	c:\proje\pics\logo\s
16	ankara	121	211	www.ankara.com	ankara@ankara.co	kızılay	c:\proje\pics\logo\
17	kask	343	454	www.selek.com.tr	kask@kask.com	sandıklı - afyon	c:\proje\pics\logo\

Fig. 4.2 Company Table

Column Name	Data Type	Length	Allow Nulls
cusid	int	4	
cusname	nvarchar	50	
cussurname	nvarchar	50	
cuscompany	nvarchar	50	✓
cusfax	nvarchar	50	✓
cusphone	nvarchar	50	✓
cusmail	nvarchar	50	✓
cusaddress	nvarchar	50	✓
custaxno	nvarchar	50	✓

Fig. 4.3 Customer Design Table

cusid	cusname	cussurname	cuscompany	cusfax	cusphone	cusmail	cusaddress	custaxno	cusimage
1	adem	atçeken	arena	3453423	05443234322	arena@arena.com.	bulgurimam mah. v.	2343234444	C:\proje\pics\custo
2	alper	karakuş	penta	3453333	05324567345	alppower@yahoo.c	mehir cad. asa işha	5666677665	C:\proje\pics\custo
3	kamil	selek	arena	3477748	05324567644	kamilselek@yahoo.	keçeciler cad. vahd	5598988999	C:\proje\pics\custo
4	tevfik	poçanoglu	gold	6678688	05324567444	tserdar@hotmail.cc	selçuklu_konya	3434343434	C:\proje\pics\custo
5	abdurrahman	atçeken	arena	3434434	05553455432	apoatceken@yaho	bulgurimam mah. v.	6678554545	C:\proje\pics\custo
6	süleyman	atçeken	arena	7667777	05324567656	suleymanatc@yahc	www.suleymanatc.	6767676672	C:\proje\pics\custo
7	mustafa	arık	gold	03923434534	05338998944	mustafa_arik1@hot	marmara bölgesi 24	6654444333	C:\proje\pics\custo
8	adem	çelik	penta	6667788	02133455544	ademcelik@hotmail	izmir	4545454545	C:\proje\pics\custo

Fig. 4.4 Customer Table

	Column Name	Data Type	Length	Allow Nulls
▶	Z	int	4	
	uname	nvarchar	20	
	upass	nvarchar	15	
	access	nvarchar	15	

Fig. 4.5 Login Design Table

	userid	uname	upass	access
▶	6	manager	manager	manager
	8	edimspeed	9985306933	manager
	12	selek	selek	user
	13	adem	adem	user
	14	abdurrahman	atceken	user
	24	atceken	atceken	user
	25	ahmet	ahmet	user
	26	user	user	user
	27	abdullah	abdullah	user
	28	suleyman	suleyman	user
	29	elif	elif	user
	30	neu	neu	user
	31	buket	buket	user

Fig. 4.6 Login Table

	Column Name	Data Type	Length	Allow Nulls
▶	pid	int	4	
	pcode	nvarchar	50	✓
	pname	nvarchar	100	✓
	pprice	money	8	✓
	pguaranty	int	4	✓
	pcompany	nvarchar	50	✓
	pimage	nvarchar	80	✓

Fig. 4.7 Product Design Table

	pid	pcode	pname	pprice	pguaranty	pcompany	pimage
▶	1	mouse	a4 tech - aq12 PS/	10	1	arena	C:\proje\pics\stock
	2	mouse	microsoft- mv 44 P	15	1	penta	C:\proje\pics\stock
	3	mouse	logitech -lk 89 Optil	20	1	gold	C:\proje\pics\stock
	4	mainboard	asus- Sc-775 I945F	105	1	arena	C:\proje\pics\stock
	5	mainboard	asus-Sc-775 SIS 6€	107	1	slayt	C:\proje\pics\stock
	6	ram	fly-512 MB, 400 MH	80	2	slayt	C:\proje\pics\stock
	7	ram	fly-256 MB, 400 MH	45	2	gold	C:\proje\pics\stock
	8	ram	matrix-256 MB, 533	46	2	gold	C:\proje\pics\stock
	9	ram	twinmos-1 GB, 533	100	2	arena	C:\proje\pics\stock
	10	ram	kingstone-512 MB,	78	2	koyuncu	C:\proje\pics\stock
	11	cpu	amd-Athlon 64 FX-(	340	2	slayt	C:\proje\pics\stock

Fig. 4.8 Product Table

	Column Name	Data Type	Length	Allow Nulls
🔑	salid	int	4	
	customerid	int	4	
	pid	int	4	
	salcode	nvarchar	50	✓
	salname	nvarchar	100	✓
	saldate	smalldatetime	4	✓
	salprice	money	8	✓
	saltotal	money	8	✓
	salvat	int	4	✓
	salquantity	int	4	✓
	salguaranty	int	4	✓

Fig. 4.9 Sale Design Table

salid	customerid	pid	salcode	salname	saldate	salprice	saltotal	salvat	salquantity	salguaranty
56	3	14	cpu	intel-Pentium 4 531	26.05.2004	189	189	18	1	2
57	3	13	cpu	amd-Athlon 64 350	26.05.2003	178	178	18	1	2
58	3	4	mainboard	asus- Sc-775 i945F	26.05.2002	105	105	18	1	1
59	3	25	harddisk	samsung- sm 78 12	26.05.2001	112	112	18	1	2
60	3	13	cpu	amd-Athlon 64 350	26.05.2006	178	178	18	1	2
61	3	14	cpu	intel-Pentium 4 531	26.05.2006	189	189	18	1	2
62	3	15	cpu	intel-Pentium 4 511	26.05.2006	190	190	18	1	2
63	3	13	cpu	amd-Athlon 64 350	26.05.2006	178	178	18	1	2
64	7	14	cpu	intel-Pentium 4 531	26.05.2006	189	189	18	1	2
65	7	5	mainboard	asus-Sc-775 SIS 66	26.05.2006	107	107	18	1	1
66	7	30	camera	olivetti-1300K 1295	26.05.2006	23	23	18	1	1
67	7	21	floppy	nec-1.44MB Tekli	26.05.2006	40	40	18	1	1
68	7	6	ram	fly-512 MB, 400 MH	26.05.2006	80	80	18	1	2
69	7	8	ram	matrix-256 MB, 533	26.05.2006	46	46	18	1	2
70	7	2	mouse	microsoft- mv 44 R	26.05.2006	15	15	18	1	1

Fig. 4.10 Sale Table

	Column Name	Data Type	Length	Allow Nulls
🔑	stid	int	4	
	pid	int	4	
	stcode	nvarchar	50	✓
	stname	nvarchar	100	✓
	stbuy	money	8	✓
	stcompany	nvarchar	50	✓
	stguaranty	int	4	✓
	stdate	smalldatetime	4	✓
	stquantity	int	4	✓
	stimage	nvarchar	100	✓

Fig. 4.11 Stock Design Table

stid	pid	stcode	stname	stbuy	stcompany	stguaranty	stdate	stquantity	stimage
3	1	mouse	a4 tech - ps/2 opti	10	gold	1	26.05.2006	1	C:\proje\pics\stoc
4	4	mainboard	asus- sc-775 i945p	105	hizlisistem	1	26.05.2006	10	C:\proje\pics\stoc
5	6	ram	fly-512 mb, 400 mh	80	gold	2	26.05.2006	6	C:\proje\pics\stoc
6	26	harddisk	maxtor-ata23	1120	gold	2	26.05.2006	9	C:\proje\pics\stoc
7	47	voice	fly-prime 34	56	vatan	1	26.05.2006	5	C:\proje\pics\stoc
8	55	vga	daytana-64mb agp	80	somya	2	26.05.2006	3	C:\proje\pics\stoc
9	44	monitor	samsung-a3 128	470	gold	2	26.05.2006	10	C:\proje\pics\stoc
10	45	monitor	samsung-a1 330 fl	178	bogazici	2	26.05.2006	10	C:\proje\pics\stoc
11	46	monitor	sony-t5 5500 flat	580	hizlisistem	3	26.05.2006	6	C:\proje\pics\stoc
12	18	dvd-rw	lg-cd-rw, lh45	260	hizlisistem	2	26.05.2006	2	C:\proje\pics\stoc
13	19	dvd-rw	lg-cd-rw, lh78 dvd	70	hizlisistem	1	26.05.2006	2	C:\proje\pics\stoc
17	47	voice	fly-prime 34	56	gold	1	26.05.2006	5	C:\proje\pics\stoc

Fig. 4.12 Stock Table