



NEAR EAST UNIVERSITY

ENGINEERING FACULTY

COMPUTER ENGINEERING DEPARTMENT

COM400 GRADUATION PROJECT

ECZANEM

The Pharmacy Automation using Microsoft Access XP Database

Student Name: Arda GÜLERER

Student Number: 20020287

Supervisor: Asst. Prof Dr. Elbrus IMANOV

NICOSIA 2006



ACKNOWLEDGMENT

First of all I want to thanks to my supervisor Asst. Prof. Dr. Elrus IMANOV for his help valuable advices.

Secondly to Mr. Okan DONANGİL for excellent teaching of first year lectures on my student life. And his valuable advices while I was writing program.

Thirdly I want to thanks to Near East University for good education and chance to be an computer engineer.

Fourthly I want to thanks to my parents who give me chance to be a computer engineer in Northern Cyprus Turkish Republic.

Fifthly, I want to thanks to my brother Alper who is an engineer has a master degree and his wife Burçin who is pharmacist. They prepare a requirement document for me and I learned much information about pharmacy with their support.

Then I want to thanks all my room mates in Girne for their patient when I was studying until late hours.

ABSTRACT

This project is a pharmacy automation which I called ECZANEM. ECZANEM is programmed at Microsoft Visual Basic 6.0 Professional Edition with Service Pack 3. Also I used Microsoft Access XP for creating Database and its tables. And I used some simple SQL queries to manage databases.

Even a small pharmacy shop has thousands of medicine and many customers of these medicines. There are about one hundreds of customer of a small pharmacy shop in a day. Every customer can buys more than one medicine in a day. And some of the customer use cash money, some of them uses credit card, some of them has society health insurance and some of them wants to buy medicine on credit. So the pharmacist need manage the selling operation. By using ECZANEM Sales form pharmacist easily manage this selling operation. Some of the medicines like aspirin can found in any pharmacy shop. There are some medicines Cancer Drug can found only big pharmacy shop because of their high price. Also there are 50-100 in stock aspirin even in a small shop but many of the medicine have only one in stock. In a day a pharmacy shop can sell more than hundred medicines. When a medicine is exhausted in stock pharmacist must order to firm to buy some. ECZANEM help to pharmacist to manage order operation.

ECZANEM is a user friendly program easy to use and helpful for the pharmacist.

CHAPTER 3: MICROSOFT ACCESS	27
3.1 Uses	27
3.2 Features	28
3.3 Development	29
CHAPTER FOUR: VISUAL BASIC	30
4.1. Derivative languages	31
4.2. Language features	32
4.3. Controversy	33
4.3.1 Programming constructs not present in Visual Basic	35
4.3.2 Behaviors present in Visual Basic	35
4.3. Visual Basic and VB.NET	36
4.4. Evolution of Visual Basic	37
CHAPTER 5: DESCRIPTION ABOUT MY PROJECT	40
5.1. Software Requirement Document	40
5.2 Starting a Visual Basic 6.0 Profession Edition with service pack 3 Project	40
5.2.1 Designing of forms	41
5.3 Working with ECZANEM	47
5.3.1 Main form of the Program	48
5.3.2 Sales Form of the Program	50
5.3.3 Order Form of the ECZANEM	57
5.3.4. Medicine Information Forms	65
5.3.5. The STATICS Form of the ECZANEM	68
5.3.6 User Accounts	70
5.3.7 Default Variables	72
5.3.8 The Search Form of the ECZANEM	73
5.3.9 Tools of ECZANEM	74
5.3.9.1 Calendar	74
5.3.9.2 Calculator	75
5.3.10 the About Form of the ECZANEM	75
Chapter 6: TABLES	75
Conclusion	76
Appendix	77

TABLE OF CONTENTS

Acknowledgement	2
Abstract	3
Table of contents	4
Introduction	6
CHAPTER ONE: INTRODUCTION THE DATABASE	
1.1 History	8
1.2 Database models	9
1.2.1 Flat model	10
1.2.2 Hierarchical model	10
1.2.3 Network model	10
1.2.4 Relational model	11
1.2.4.1 Relational operations	12
1.2.5 Dimensional model	13
1.2.6 Object database models	13
1.3 Database Internals	14
1.3.1 Indexing	14
1.3.2. Transactions and concurrency	15
1.3.3. Replication	15
1.4. Applications of databases	16
1.5. Database Brands	16
CHAPTER TWO: SQL	17
2.1 History	17
2.1.1 Standardization	19
2.2. Scope	19
2.3. SQL keywords	21
2.3.1 Data retrieval	21
2.3.2. Data manipulation	22
2.3.3 Data transaction	22
2.3.4 Data definition	23
2.3.5. Data control	24
2.4. Criticisms of SQL	24
2.5 Logical Operators	25

INTRODUCTION

My project is a pharmacy automation which uses Access and SQL queries. I write this program at Microsoft Visual Basic 6.0 Professional Edition with Service Pack 3. And I used Microsoft Access XP to create database tables. And also I used sometimes SQL on my code.

I tell the subjects chapter by chapter so let us go through the overview the chapters in brief.

CHAPTER 1: I discuss about the database general information, database models, relational operations, database brands.

CHAPTER 2: I discuss about SQL. Its history, keywords and some of the commands of it

CHAPTER 3: I discuss about Microsoft Access and its features.

CHAPTER 4: I discuss about Visual Basic Language Features and past and future of Visual Basic

CHAPTER 5: I discuss about my program, how I create it, its forms and using my program

CHAPTER 6: I give a graphics about ECZANEM tables.

CHAPTER ONE: INTRODUCTION TO DATABASE

A **database** is an organized collection of data. The term originated within the computer industry, but its meaning has been broadened by popular use, to the extent that the European Database Directive (which creates intellectual property rights for databases) includes non-electronic databases within its definition. This article is confined to a more technical use of the term; though even amongst computing professionals, some attach a much wider meaning to the word than others.

One possible definition is that a database is a collection of records stored in a computer in a systematic way, so that a computer program can consult it to answer questions. For better retrieval and sorting, each record is usually organized as a set of data elements (facts). The items retrieved in answer to queries become information that can be used to make decisions. The computer program used to manage and query a database is known as a database management system (DBMS). The properties and design of database systems are included in the study of information science.

The central concept of a database is that of a collection of records, or pieces of knowledge. Typically, for a given database, there is a structural description of the type of facts held in that database: this description is known as a **schema**. The schema describes the objects that are represented in the database, and the relationships among them. There are a number of different ways of organizing a schema, that is, of modeling the database structure: these are known as database models (or data models). The model in most common use today is the relational model, which in layman's terms represents all information in the form of multiple related tables each consisting of rows and columns (the true definition uses mathematical terminology). This model represents relationships by the use of values common to more than one table. Other models such as the hierarchical model and the network model use a more explicit representation of relationships.

Strictly speaking, the term *database* refers to the collection of related records, and the software should be referred to as the *database management system* or DBMS. When the context is unambiguous, however, many database administrators and programmers use the term *database* to cover both meanings.

Many professionals would consider a collection of data to constitute a database only if it has certain properties: for example, if the data is managed to ensure its integrity and quality, if it allows shared access by a community of users, if it has a schema, or if it supports a query language. However, there is no agreed definition of these properties.

Database management systems are usually categorized according to the data model that they support: relational, object-relational, network, and so on. The data model will tend to determine the query languages that are available to access the database. A great deal of the internal engineering of a DBMS, however, is independent of the data model, and is concerned with managing factors such as performance, concurrency, integrity, and recovery from hardware failures. In these areas there are large differences between products.

1.1 History

The earliest known use of the term '*data base*' was in June 1963, when the System Development Corporation sponsored a symposium under the title *Development and Management of a Computer-centered Data Base*. **Database** as a single word became common in Europe in the early 1970s and by the end of the decade it was being used in major American newspapers. The first database management systems were developed in the 1960s. A pioneer in the field was Charles Bachman. Bachman's early papers show that his aim was to make more effective use of the new direct access storage devices becoming available: until then, data processing had been based on punched cards and magnetic tape, so that serial processing was the dominant activity. Two key data models arose at this time: CODASYL developed the network model based on Bachman's ideas, and (apparently independently) the hierarchical model was used in a system developed by North American Rockwell, later adopted by IBM as the cornerstone of their IMS product.

The relational model was proposed by E. F. Codd in 1970. He criticized existing models for confusing the abstract description of information structure with descriptions of physical access mechanisms. For a long while, however, the relational model remained of academic interest only. While CODASYL systems and IMS were conceived as practical engineering solutions taking account of the technology as it existed at the time, the relational model took a much more theoretical perspective, arguing (correctly) that hardware and software technology would catch up in time. Among the first implementations were Michael Stonebraker's Ingres at Berkeley, and the System R project at IBM. Both of these were research prototypes, announced during 1976. The first commercial products, Oracle and DB2, did not appear until

around 1980. The first successful database product for microcomputers was dBASE for the CP/M and PC-DOS/MS-DOS operating systems.

During the 1980s, research activity focused on distributed database systems and database machines, but these developments had little effect on the market. Another important theoretical idea was the Functional Data Model, but apart from some specialized applications in genetics, molecular biology, and fraud investigation, the world took little notice.

In the 1990s, attention shifted to object-oriented databases. These had some success in fields where it was necessary to handle more complex data than relational systems could easily cope with, such as spatial databases, engineering data (including software engineering repositories), and multimedia data. Some of these ideas were adopted by the relational vendors, who integrated new features into their products as a result.

In the 2000s, the fashionable area for innovation is the XML database. As with object databases, this has spawned a new collection of startup companies, but at the same time the key ideas are being integrated into the established relational products. XML databases aim to remove the traditional divide between documents and data, allowing all of an organization's information resources to be held in one place, whether they are highly structured or not.

1.2 Database models

Various techniques are used to model data structure. Most database systems are built around one particular data model, although it is increasingly common for products to offer support for more than one model. For any one logical model various physical implementations may be possible, and most products will offer the user some level of control in tuning the physical implementation, since the choices that are made have a significant effect on performance. An example of this is the relational model: all serious implementations of the relational model allow the creation of indexes which provide fast access to rows in a table if the values of certain columns are known.

A data model is not just a way of structuring data: it also defines a set of operations that can be performed on the data. The relational model, for example, defines operations such as selection, projection, and join. Although these operations may not be explicit in a particular query language, they provide the foundation on which a query language is built.

This gives excellent retrieval performance, at the expense of operations such as database loading and reorganization.

1.2.3. Relational model

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

The products that are generally referred to as relational databases in fact implement a model that is only an approximation to the mathematical model defined by Codd. The data structures in these products are tables, rather than relations: the main differences being that tables can contain duplicate rows, and that the rows (and columns) can be treated as being ordered. The same criticism applies to the SQL language which is the primary interface to these products. There has been considerable controversy, mainly due to Codd himself, as to whether it is correct to describe SQL implementations as "relational": but the fact is that the world does so, and the following description uses the term in its popular sense.

A relational database contains multiple tables, each similar to the one in the "flat" database model. Relationships between tables are not defined explicitly; instead, *keys* are used to match up rows of data in different tables. A key is a collection of one or more columns in one table whose values match corresponding columns in other tables: for example, an *Employee* table may contain a column named *Location* which contains a value that matches the key of a *Location* table. Any column can be a key, or multiple columns can be grouped together into a single key. It is not necessary to define all the keys in advance; a column can be used as a key even if it was not originally intended to be one.

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

A key that has an external, real-world meaning (such as a person's name, a book's ISBN, or a car's serial number), is sometimes called a "natural" key. If no natural key is suitable (think of the many people named *Brown*), an arbitrary key can be assigned (such as by giving employees ID numbers). In practice, most databases have both generated and

natural keys, because generated keys can be used internally to create links between rows that cannot break, while natural keys can be used, less reliably, for searches and for integration with other databases. (For example, records in two independently developed databases could be matched up by social security number, except when the social security numbers are incorrect, missing, or have changed.)

1.2.4.1. Relational operations

Users (or programs) request data from a relational database by sending it a query that is written in a special language, usually a dialect of SQL. Although SQL was originally intended for end-users, it is much more common for SQL queries to be embedded into software that provides an easier user interface. Many web sites, such as wikipedia, perform SQL queries when generating pages.

In response to a query, the database returns a result set, which is just a list of rows containing the answers. The simplest query is just to return all the rows from a table, but more often, the rows are filtered in some way to return just the answer wanted. Often, data from multiple tables is combined into one, by doing a join. Conceptually, this is done by taking all possible combinations of rows (the Cartesian product), and then filtering out everything except the answer. In practice, relational database management systems rewrite ("optimize") queries to perform faster, using a variety of techniques.

There are a number of relational operations in addition to join. These include project (the process of eliminating some of the columns), restrict (the process of eliminating some of the rows), union (a way of combining two tables with similar structures), difference (which lists the rows in one table that are not found in the other), intersect (which lists the rows found in both tables), and product (mentioned above, which combines each row of one table with each row of the other). Depending on which other sources you consult, there are a number of other operators - many of which can be defined in terms of those listed above. These include semi-join, outer operators such as outer join and outer union, and various forms of division. Then there are operators to rename columns, and summarizing or aggregating operators, and if you permit relation values as attributes (RVA - relation-valued attribute), then operators such as group and ungroup. The SELECT statement in SQL serves to handle all of these except for the group and ungroup operators.

The flexibility of relational databases allows programmers to write queries that were not anticipated by the database designers. As a result, relational databases can be used by multiple applications in ways the original designers did not foresee, which is especially important for databases that might be used for decades. This has made the idea and implementation of relational databases very popular with businesses.

1.2.5. Dimensional model

The dimensional model is a specialized adaptation of the relational model used to represent data in data warehouses in a way that data can be easily summarized using OLAP queries. In the dimensional model, a database consists of a single large table of facts that are described using dimensions and measures. A dimension provides the context of a fact (such as who participated, when and where it happened, and its type) and is used in queries to group related facts together. Dimensions tend to be discrete and are often hierarchical; for example, the location might include the building, state, and country. A measure is a quantity describing the fact, such as revenue. It's important that measures can be meaningfully aggregated - for example, the revenue from different locations can be added together.

In an OLAP query, dimensions are chosen and the facts are grouped and added together to create a summary.

The dimensional model is often implemented on top of the relational model using a star schema, consisting of one table containing the facts and surrounding tables containing the dimensions. Particularly complicated dimensions might be represented using multiple tables, resulting in a snowflake schema.

1.2.6. Object database models

In recent years, the object-oriented paradigm has been applied to database technology, creating a new programming model known as object databases. These databases attempt to bring the database world and the application programming world closer together, in particular by ensuring that the database uses the same type system as the application program. This aims to avoid the overhead (sometimes referred to as the *impedance mismatch*) of converting

information between its representation in the database (for example as rows in tables) and its representation in the application program (typically as objects). At the same time object databases attempt to introduce the key ideas of object programming, such as encapsulation and polymorphism, into the world of databases.

A variety of these ways have been tried for storing objects in a database. Some products have approached the problem from the application programming end, by making the objects manipulated by the program persistent. This also typically requires the addition of some kind of query language, since conventional programming languages do not have the ability to find objects based on their information content. Others have attacked the problem from the database end, by defining an object-oriented data model for the database, and defining a database programming language that allows full programming capabilities as well as traditional query facilities.

Object databases suffered because of a lack of standardization: although standards were defined by ODMG, they were never implemented well enough to ensure interoperability between products. Nevertheless, object databases have been used successfully in many applications: usually specialized applications such as engineering databases or molecular biology databases rather than mainstream commercial data processing. However, object database ideas were picked up by the relational vendors and influenced extensions made to these products and indeed to the SQL language.

1.3 Database Internals

1.3.1 Indexing

All of these kinds of database can take advantage of indexing to increase their speed, and this technology has advanced tremendously since its early uses in the 1960s and 1970s. The most common kind of index is a sorted list of the contents of some particular table column, with pointers to the row associated with the value. An index allows a set of table rows matching some criterion to be located quickly. Various methods of indexing are commonly used; B-trees, hashes, and linked lists are all common indexing techniques.

Relational DBMSs have the advantage that indices can be created or dropped without changing existing applications making use of it. The database chooses between many different strategies based on which one it estimates will run the fastest.

Relational DBMSs utilize many different algorithms to compute the result of an SQL statement. The RDBMS will produce a plan of how to execute the query, which is generated by analysing the run times of the different algorithms and selecting the quickest. Some of the key algorithms that deal with joins are Nested Loops Join, Sort-Merge Join and Hash Join.

1.3.2. Transactions and concurrency

In addition to their data model, most practical databases ("transactional databases") attempt to enforce a database transaction model that has desirable data integrity properties. Ideally, the database software should enforce the ACID rules, summarized here:

Atomicity - Either all the tasks in a transaction must be done, or none of them. The transaction must be completed, or else it must be undone (rolled back).

Consistency - Every transaction must preserve the integrity constraints -- the declared consistency rules -- of the database. It cannot place the data in a contradictory state.

Isolation - Two simultaneous transactions cannot interfere with one another. Intermediate results within a transaction are not visible to other transactions.

Durability - Completed transactions cannot be aborted later or their results discarded. They must persist through (for instance) restarts of the DBMS after crashes.

In practice, many DBMS's allow most of these rules to be selectively relaxed for better performance.

Concurrency control is a method used to ensure that transactions are executed in a safe manner and follow the ACID rules. The DBMS must be able to ensure that only serializable, recoverable schedules are allowed, and that no actions of committed transactions are lost while undoing aborted transactions.

1.3.3. Replication

Replication of databases is closely related to transactions. If a database can log its individual actions, it is possible to create a duplicate of the data in real time. The duplicate can be used to improve Performance or Availability of the whole database system. Common replication concepts include:

Master/Slave Replication: All write requests are performed on the master and then replicated to the slaves

Quorum: The result of Read and Write requests is calculated by querying a "majority" of replicas.

Multimaster: Two or more replicas sync each other via a transaction identifier.

1.4. Applications of databases

Databases are used in many applications, spanning virtually the entire range of computer software. Databases are the preferred method of storage for large multiuser applications, where coordination between many users is needed. Even individual users find them convenient, though, and many electronic mail programs and personal organizers are based on standard database technology. Software database drivers are available for most database platforms so that application software can use a common application programming interface (API) to retrieve the information stored in a database. Two commonly used database APIs are JDBC and ODBC.

1.5. Database Brands

(In alphabetical order)

4D

Adabas

Adaptive Server Enterprise

Corel Paradox

Dataflex

Dataphor

DB2

Filemaker

Firebird

Information Management System

Informix

Ingres

Intersystem Cache

Kx

Microsoft Access

Microsoft SQL Server
MySQL
Netezza
OpenOffice.org Base
Oracle
PostgreSQL
Progress
Rel (DBMS)
SQLite
SQL Anywhere Studio
Teradata
VistaDB

CHAPTER TWO: SQL

SQL (commonly expanded to **Structured Query Language** — see *History* for the term's derivation) is the most popular computer language used to create, modify, retrieve and manipulate data from relational database management systems. The language has evolved beyond its original purpose to support object-relational database management systems. It is an ANSI/ISO standard.

2.1 History

An influential paper, "A Relational Model of Data for Large Shared Data Banks", by Dr. Edgar F. Codd, was published in June, 1970 in the Association for Computing Machinery (ACM) journal, *Communications of the ACM*. Codd's model became widely accepted as the definitive model for *relational* database management systems (RDBMS or RDMS).

During the 1970s, a group at IBM's San Jose research center developed a database system "System R" based upon, but not strictly faithful to, Codd's model. **Structured English Query Language** ("*SEQUEL*") was designed to manipulate and retrieve data stored in System R. The acronym *SEQUEL* was later condensed to **SQL** because the word 'SEQUEL'

was held as a trademark by the Hawker-Siddeley aircraft company of the UK. Although SQL was influenced by Codd's work, Donald D. Chamberlin and Raymond F. Boyce at IBM were the authors of the SEQUEL language design.. Their concepts were published to increase interest in SQL.

The first non-commercial, relational, non-SQL database, Ingres, was developed in 1974 at U.C. Berkeley.

In 1978, methodical testing commenced at customer test sites. Demonstrating both the usefulness and practicality of the system, this testing proved to be a success for IBM. As a result, IBM began to develop commercial products that implemented SQL based on their System R prototype, including the System/38 (announced in 1978 and commercially available in August 1979), SQL/DS (introduced in 1981), and DB2 (in 1983).

At the same time Relational Software, Inc. (now Oracle Corporation) saw the potential of the concepts described by Chamberlin and Boyce and developed their own version of a RDBMS for the Navy, CIA and others. In the summer of 1979 Relational Software, Inc. introduced Oracle V2 (Version2) for VAX computers as the first commercially available implementation of SQL. Oracle is often incorrectly cited as beating IBM to market by two years, when in fact they only beat IBM's release of the System/38 by a few weeks. Considerable public interest then developed; soon many other vendors developed versions, and Oracle's future was ensured.

It is often suggested that IBM was slow to develop SQL and relational products, possibly because it wasn't available initially on the mainframe and Unix environments, and that they were afraid it would cut into lucrative sales of their IMS database product, which used navigational database models instead of relational. But at the same time as Oracle was being developed, IBM was developing the System/38, which was intended to be the first relational database system, and was thought by some at the time, because of its advanced design and capabilities, that it might have become a possible replacement for the mainframe and Unix systems.

2.1.1 Standardization

SQL was adopted as a standard by ANSI (American National Standards Institute) in 1986 and ISO (International Organization for Standardization) in 1987. ANSI has declared that the official pronunciation for SQL is /ɛs kju: ɛl/, although many English-speaking database professionals still pronounce it as *sequel*, and with gaining popularity has received the alias 'SQuirreL'.

The SQL standard has gone through a number of revisions:

Year	Name	Alias	Comments
1986	SQL-86	SQL-	First published by ANSI. Ratified by ISO in 1987.
		87	
1989	SQL-89		Minor revision.
1992	SQL-92	SQL2	Major revision.
1999	SQL:1999	SQL3	Added regular expression matching, recursive queries, triggers, non-scalar types and some object-oriented features. (The last two are somewhat controversial and not yet widely supported.)
2003	SQL:2003		Introduced XML-related features, <i>window functions</i> , standardized sequences and columns with auto-generated values (including identity-columns).

2.2. Scope

Although SQL is defined by both ANSI and ISO, there are many extensions to and variations on the version of the language defined by these standards bodies. Many of these extensions are of a proprietary nature, such as Oracle Corporation's PL/SQL or Sybase, IBM's SQL PL (SQL Procedural Language) and Microsoft's Transact-SQL. It is also not uncommon for commercial implementations to omit support for basic features of the standard, such as the DATE or TIME data types, preferring some variant of their own. As a result, in contrast to ANSI C or ANSI Fortran, which can usually be ported from platform to platform without major structural changes, SQL code can rarely be ported between database systems without major modifications. There are several reasons for this lack of portability between database systems:

the complexity and size of the SQL standard means that most databases do not implement the entire standard.

the standard does not specify database behavior in several important areas (e.g. indexes), leaving it up to implementations of the standard to decide how to behave.

the SQL standard precisely specifies the syntax that a conformant database system must implement. However, the standard's specification of the semantics of language constructs is less well-defined, leading to areas of ambiguity.

many database vendors have large existing customer bases; where the SQL standard conflicts with the prior behavior of the vendor's database, the vendor may be unwilling to break backward compatibility.

some believe the lack of compatibility between database systems is intentional in order to ensure vendor lock-in.

SQL is designed for a specific, limited purpose — querying data contained in a relational database. As such, it is a set-based, declarative computer language rather than an imperative language such as C or BASIC which, being programming languages, are designed to solve a much broader set of problems. Language extensions such as PL/SQL are designed to address this by turning SQL into a full-fledged programming language while maintaining the advantages of SQL. Another approach is to allow programming language code to be embedded in and interact with the database. For example, Oracle and others include Java in the database, while PostgreSQL allows functions to be written in a wide variety of languages, including Perl, Tcl, and C.

SQL contrasts with the more powerful database-oriented fourth-generation programming languages such as Focus or SAS in its relative functional simplicity and simpler command set. This greatly reduces the degree of difficulty involved in maintaining SQL source code, but it also makes programming such questions as 'Who had the top ten scores?' more difficult, leading to the development of procedural extensions, discussed above. However, it also makes it possible for SQL source code to be produced (and optimized) by software, leading to the development of a number of natural language database query languages, as well as 'drag and drop' database programming packages with 'object oriented' interfaces. Often these allow the resultant SQL source code to be examined, for educational purposes, further enhancement, or to be used in a different environment.

2.3. SQL keywords

SQL keywords fall into several groups.

2.3.1 Data retrieval

The most frequently used operation in transactional databases is the data retrieval operation. When restricted to data retrieval commands, SQL acts as a functional language. `SELECT` is used to retrieve zero or more rows from one or more tables in a database. In most applications, `SELECT` is the most commonly used DML command. In specifying a `SELECT` query, the user specifies a description of the desired result set, but they do *not* specify what physical operations must be executed to produce that result set. Translating the query into an efficient query plan is left to the database system, more specifically to the query optimizer.

Commonly available keywords related to `SELECT` include:

`FROM` is used to indicate from which tables the data is to be taken, as well as how the tables join to each other.

`WHERE` is used to identify which rows to be retrieved, or applied to `GROUP BY`. `WHERE` is evaluated before the `GROUP BY`.

`GROUP BY` is used to combine rows with related values into elements of a smaller set of rows.

`HAVING` is used to identify which of the "combined rows" (combined rows are produced when the query has a `GROUP BY` keyword or when the `SELECT` part contains aggregates), are to be retrieved. `HAVING` acts much like a `WHERE`, but it operates on the results of the `GROUP BY` and hence can use aggregate functions.

`ORDER BY` is used to identify which columns are used to sort the resulting data.

Example 1:

```
SELECT * FROM books
WHERE price > 100.00
ORDER BY title
```

This example retrieves the records from the `books` table that have a price field which is greater than 100.00. The result is sorted alphabetically by book title. The asterisk (*) means to show all columns of the `books` table. Alternatively, specific columns could be named.

Example 2:

```
SELECT books.title, count(*) AS Authors
FROM books
JOIN book_authors ON books.book_number = book_authors.book_number
GROUP BY books.title
```

Example 2 shows both the use of multiple tables in a join and aggregation (grouping). This example shows how many authors there are per book. Example output may resemble:

Title	Authors
SQL Examples and Guide	3
The Joy of SQL	1
How to use Wikipedia	2
Pitfalls of SQL	1
How SQL Saved my Dog	1

2.3.2. Data manipulation

First there are the standard Data Manipulation Language (DML) elements. DML is the subset of the language used to add, update and delete data.

INSERT is used to add zero or more rows (formally tuples) to an existing table.

UPDATE is used to modify the values of a set of existing table rows.

MERGE is used to combine the data of multiple tables. It is something of a combination of the INSERT and UPDATE elements. It is defined in the SQL:2003 standard; prior to that, some databases provided similar functionality via different syntax, sometimes called an "upsert".

TRUNCATE deletes all data from a table (non-standard, but common SQL command).

DELETE removes zero or more existing rows from a table.

Example:

```
INSERT INTO my_table (field1, field2, field3) VALUES ('test', 'N', NULL);
```

```
UPDATE my_table SET field1 = 'updated value' WHERE field2 = 'N';
```

```
DELETE FROM my_table WHERE field2 = 'N';
```

2.3.3 Data transaction

Transaction, if available, can be used to wrap around the DML operations.

START TRANSACTION (or BEGIN WORK, depending on SQL dialect) can be used to mark the start of a database transaction, which either completes completely or not at all.

COMMIT causes all data changes in a transaction to be made permanent.

ROLLBACK causes all data changes since the last COMMIT or ROLLBACK to be discarded, so that the state of the data is "rolled back" to the way it was prior to those changes being requested.

COMMIT and ROLLBACK interact with areas such as transaction control and locking. Strictly, both terminate any open transaction and release any locks held on data. In the absence of a START TRANSACTION or similar statement, the semantics of SQL are implementation-dependent.

Example:

```
START TRANSACTION;
```

```
UPDATE inventory SET quantity = quantity - 3 WHERE item = 'pants';
```

```
COMMIT;
```

2.3.4 Data definition

The second group of keywords is the Data Definition Language (DDL). DDL allows the user to define new tables and associated elements. Most commercial SQL databases have proprietary extensions in their DDL, which allow control over nonstandard features of the database system.

The most basic items of DDL are the CREATE and DROP commands.

CREATE causes an object (a table, for example) to be created within the database.

DROP causes an existing object within the database to be deleted, usually irretrievably.

Some database systems also have an ALTER command, which permits the user to modify an existing object in various ways -- for example, adding a column to an existing table.

Example:

```
CREATE TABLE my_table (  
  my_field1 INT      UNSIGNED,  
  my_field2 VARCHAR (50),  
  my_field3 DATE     NOT NULL,  
  PRIMARY KEY (my_field1, my_field2)
```

2.3.5. Data control

The third group of SQL keywords is the Data Control Language (DCL). DCL handles the authorization aspects of data and permits the user to control who has access to see or manipulate data within the database.

Its two main keywords are:

GRANT — authorises one or more users to perform an operation or a set of operations on an object.

REVOKE — removes or restricts the capability of a user to perform an operation or a set of operations.

Example:

```
GRANT SELECT, UPDATE ON my_table TO some_user, another_user
```

Other

ANSI-standard SQL supports `--` as a single line comment identifier (some extensions also support curly brackets or C-style `/*` comments `*/` for multi-line comments).

Example:

```
SELECT * FROM inventory -- Retrieve everything from inventory table
```

Database systems using SQL

List of relational database management systems

List of object-relational database management systems

2.4. Criticisms of SQL

Technically, SQL is a declarative computer language for use with "SQL databases". Theorists and some practitioners note that many of the original SQL features were inspired by, but in violation of, the relational model for database management and its tuple calculus realisation. Recent extensions to SQL achieved relational completeness, but have worsened the violations, as documented in *The Third Manifesto*.

In addition, there are also some criticisms about the practical use of SQL:

The language syntax is rather complex (sometimes called "COBOL-like").

It does not provide a standard way, or at least a commonly-supported way, to split large commands into multiple smaller ones that reference each other by name. This tends to result in "run-on SQL sentences" and may force one into a deep hierarchical nesting when a graph-like (reference-by-name) approach may be more appropriate and better repetition-factoring.

Implementations are inconsistent and, usually, incompatible between vendors.

For larger statements, it is often difficult to factor repeated patterns and expressions into one or fewer places to avoid repetition and avoid having to make the same change to different places in a given statement.

Confusion about the difference between value-to-column assignment in UPDATE and INSERT syntax.

2.5 Logical Operators

Logical constructs consist of other logical constructs, formulas and values, which are connected through logical operators. For example, comparisons with equal, unequal and so on.

Operator	Description	Example	Description
AND	Means that the values at both sides of the operator must be TRUE, otherwise this operator returns FALSE. The execution priority is (in SQL) bigger than OR, but smaller than NOT.	preis > 10 AND preis < 100	Price is bigger than 10 and smaller than 100.
NOT	Logical negation, makes from FALSE a TRUE value and vice versa. Biggest execution priority of all logical operators.	NOT preis=0	Price is not zero.
OR	Means that at least one of the values at both sides of the operator must be TRUE, otherwise this operator returns FALSE. This operator is executed after NOT and AND.	preis > 10 AND preis < 100 OR preis > 1000	Price must be bigger than 10 and smaller than 100 or bigger than 1000.
=	Is TRUE when the values at both sides of the operator are equal. The execution priority is bigger than these of NOT, AND and OR.	preis = 10	Price is equal to 10.
>	Is TRUE when the value at the left side of the operator is greater than the value at the right side. The execution priority is bigger than these of NOT, AND and OR.	preis > 0	Price is bigger than 0.

\geq	Is TRUE when the value at the left side of the operator is greater or equal to the value at the right side. The execution priority is bigger than these of NOT, AND and OR.	$\text{preis} \geq 300$	Price is bigger than or equal to 300.
$<$	Is TRUE when the value at the left side of the operator is smaller than the value at the right side. The execution priority is bigger than these of NOT, AND and OR.	$\text{preis} < 300$	Price is smaller than 300.
\leq	Is TRUE when the value at the left side of the operator is smaller or equal to the value at the right side. The execution priority is bigger than these of NOT, AND and OR.	$\text{preis} \leq 300$	Price is smaller than or equal to 300 is.
\neq	Is TRUE when the value at the left side of the operator is not equal to the value at the right side. The execution priority is bigger than these of NOT, AND and OR.	$\text{preis} \neq 0$	Price is not equal to 0.

CHAPTER 3: MICROSOFT ACCESS

Maintainer:	Microsoft
Latest	2003 for Windows / October 2003
release:	(Windows)
OS:	Microsoft Windows
Use:	RDBMS
Website:	office.microsoft.com

Microsoft Access (full name **Microsoft Office Access**) is a relational database management system from Microsoft, packaged with Microsoft Office Professional which combines the relational Microsoft Jet Database Engine with a graphical user interface. It can use data stored in Access/Jet, SQL Server, Oracle, or any ODBC-compliant data container. Skilled software developers and data architects use it to develop powerful, complex application software. Relatively unskilled programmers and non-programmer "power users" can use it to build simple applications without having to deal with features they don't understand. It supports substantial object-oriented (OO) techniques but falls short of being a fully OO development tool.

Microsoft Access was also the name of a communications program from Microsoft, meant to compete with ProComm and other programs. It proved a failure and was dropped. Years later they reused the name for their database software.

3.1 Uses

Access is widely used by small businesses, within departments of large corporations, and hobby programmers to create ad hoc customized systems for handling the creation and manipulation of data. Its ease of use and powerful design tools give the non-professional programmer a lot of power for little effort. However, this ease of use can be misleading. This sort of developer is often an office worker with little or no training in application or data design. Because Access makes it possible even for such developers to create usable systems, many are misled into thinking that the tool itself is limited to such applications.

Some professional application developers use Access for rapid application development, especially for the creation of prototypes and standalone applications that serve as tools for on-the-road salesmen. Access does not scale well if data access is via a network, so applications that are used by more than a handful of people tend to rely on a Client-Server based solution such as Oracle, DB2, Microsoft SQL Server, PostgreSQL, MySQL, MaxDB, or Filemaker. However, an Access "front end" (the forms, reports, queries and VB code) can be used against a host of database backends, including Access itself, SQL Server, Oracle, and any other ODBC-compliant product. This approach allows the developer to move a matured application's data to a more powerful server without sacrificing the development already in place.

Many developers who use Microsoft Access use the Leszynski naming convention, though this is not universal; it is a programming convention, not a DBMS-enforced rule.

3.2 Features

One of the benefits of Access from a programmer's perspective is its relative compatibility with SQL – queries may be viewed and edited as SQL statements, and SQL statements can be used directly in Macros and VBA Modules to manipulate Access tables. Users may mix and use both VBA and "Macros" for programming forms and logic and offers object-oriented possibilities.

The report writer in Access, while capable and up to the task of sophisticated report creation, is not as full-featured and powerful as another popular database report writer – Crystal Reports. MSDE (Microsoft SQL Server Desktop Engine) 2000, a mini-version of MS SQL Server 2000, is included with the developer edition of Office XP and may be used with Access as an alternative to the Jet Database Engine. (*Early versions of MSDE and Microsoft Exchange Server actually use the Jet engine to handle huge volumes of data and placed a "fake" application layer for those applications on top of it. Lack of knowledge about this fact has contributed to an undeserved disrespect for Access/Jet family of software products, particularly as regards "large" projects.)

Access' cut and paste functionality can make it a useful tool for connecting between other databases (for example, Oracle and Microsoft SQL Server during data or database conversions). Access comes with various import and export features that allow integration with Windows and other platform applications, several of which can be executed on demand from within applications or manually by the user. For example the very compact SNP format for sharing perfectly formatted reports with people who don't have the full Access software. It can also easily be upgraded to Microsoft SQL Server.

Unlike complete RDBMS's, it lacks database triggers and stored procedures. It does allow forms to contain code that is triggered as changes are made to the underlying table, and it is common to use pass-through queries and other techniques in Access to run stored procedures in RDBMSs that support these.

3.3 Development

The programming language available in Access is, as in other products of the Microsoft Office suite, Microsoft Visual Basic for Applications. Two database access libraries of COM components are provided: the legacy Data Access Objects (DAO), only available with Access, and the new ActiveX Data Objects (ADO).

Microsoft Access is easily applied to small projects but scales inefficiently to large projects if applications are designed poorly.

All database queries, forms, and reports are stored in the database, and in keeping with the ideals of the relational model, there is no possibility of making a physically structured hierarchy with them.

One design technique is to divide an Access application between data and programs. One database should contain only tables and relationships, while another would have all programs, forms, reports and queries, and links to the first database tables. Unfortunately, Access allows no relative paths when linking, so the development environment should have the same path as the production environment (Although you can write your own "dynamic-linker" routine in VBA that can search out a certain back-end file by searching through the directory tree, if it can't find it in the current path).

This technique also allows the developer to divide the application among different files, so some structure is possible.

CHAPTER FOUR: VISUAL BASIC



Paradigm:	Event-driven
Developer:	Microsoft
Typing discipline:	Static, strong
Influenced by:	QuickBASIC
Influenced:	Visual Basic .NET

Visual Basic (VB) is an event driven programming language and associated development environment prototyped by Alan Cooper as Project Ruby, then bought and vastly improved upon by Microsoft. It is derived heavily from BASIC and enables rapid application development (RAD) of graphical user interface (GUI) applications, access to databases using DAO, RDO, or ADO, and creation of ActiveX controls and objects.

A programmer can put together an application using the components provided with Visual Basic itself. Programs written in Visual Basic can also use the Windows API, but doing so requires external function declarations. Like all other Turing complete programming languages, it can be used to create arbitrarily complex applications.

In business programming, Visual Basic has one of the largest user bases. According to some sources, as of 2003, 52 percent of software developers used Visual Basic, making it the most popular programming language at that time. However, research done by Evans Data

found that forty three percent of those Visual Basic developers planned to move to other languages.

4.1. Derivative languages

Microsoft has developed derivatives of Visual Basic for use in scripting. It is derived heavily from BASIC and host applications, and has replaced the original Visual Basic language with a .NET platform version:

Visual Basic for Applications (VBA) is included in many Microsoft applications (like Microsoft Office), and also in several third-party products like WordPerfect Office 2002. There are small inconsistencies in the way VBA is implemented in different applications, but it is largely the same language as VB6.

VBScript is the default language for Active Server Pages and can be used in Windows scripting and client-side web page scripting. Although it resembles VB in syntax, it is a separate language and it is executed by the Windows Script Host as opposed to the VB runtime. These differences can affect the performance of an ASP web site (namely inefficient string concatenation and absence of short-cut evaluation). ASP and VBScript must not be confused with ASP.NET which uses Visual Basic.Net or any other language that targets the .NET Common Language Runtime.

Visual Basic .NET is the successor to Visual Basic 6.0, and is part of Microsoft's .NET platform. The VB.NET programming language is a true object-oriented language that compiles and runs on the .NET Framework. VB.NET is a totally new tool from the ground up, not backwards compatible with VB6. It ships with a basic converter to upgrade legacy VB6 code although the inefficient nature of the resulting VB.NET code (due to major differences between the two languages) often leads programmers to prefer manual conversion instead. This usually involves re-writing much of the code although in doing so, the programmer can simplify and improve the code through use of the extensive .NET framework and the more powerful constructs offered by the newer language.

4.2. Language features

Visual Basic was designed to be easy to learn and use. The language not only allows programmers to easily create simple GUI applications, but also has the flexibility to develop fairly complex applications as well. Programming in VB is a combination of visually arranging components or controls on a form, specifying attributes and actions of those components, and writing additional lines of code for more functionality. Since default attributes and actions are defined for the components, a simple program can be created without the programmer having to write many lines of code. Performance problems were experienced by earlier versions, but with faster computers and native code compilation this has become less of an issue.

Although programs can be compiled into native code executables from version 5 onwards, they still require the presence of runtime libraries of approximately 2 MB in size. This runtime is included by default in Windows 2000 and later, but for earlier versions of Windows it must be distributed together with the executable.

Forms are created using drag and drop techniques. A tools palette is used to place controls (e.g., text boxes, buttons, etc.) on the form (window). Controls have attributes and event handlers associated with them. Default values are provided when the control is created, but may be changed by the programmer. Many attribute values can be modified during run time based on user actions or changes in the environment, providing a dynamic application. For example, code can be inserted into the form resize event handler to reposition a control so that it remains centered on the form, expands to fill up the form, etc. By inserting code into the event handler for a keypress in a text box, the program can automatically translate the case of the text being entered, or even prevent certain characters from being inserted.

A Visual Basic application can consist of one or more windows, or a single window that contains MDI child windows, as provided by the operating system. Dialog boxes with less functionality (e.g., no maximize/minimize control) can be used to provide pop-up capabilities. Controls provide the basic functionality of the application, while programmers can insert additional logic within the appropriate event handlers. For example, a drop-down combination box will automatically display its list and allow the user to select any element. An event handler is called when an item is selected, which can then execute additional code created by the programmer to perform some action based on which element was selected, such as populating a related list.

Alternatively, a Visual Basic component can have no user interface, and instead provide ActiveX objects to other programs via Component Object Model (COM). This allows for server-side processing or an add-in module.

The language is garbage collected using reference counting, has a large library of utility objects, and has basic object oriented support. Since the more common components are included in the default project template, the programmer seldom needs to specify additional libraries. Unlike many other programming languages, Visual Basic is generally not case sensitive, although it will transform keywords into a standard case configuration and force the case of variable names to conform to the case of the entry within the symbol table entry. String comparisons are case sensitive by default, but can be made case insensitive if so desired.

4.3. Controversy

Visual Basic is a controversial language; many programmers have strong feelings regarding the quality of Visual Basic and its ability to compete with newer languages. It was designed to be a simple language. In the interest of convenience and rapid development, some features like compile time type-checking and variable declaration are turned off by default. This leads to some programmers praising Visual Basic for how simple it is to use, but can also lead to frustration when programmers encounter problems that the features would have detected (e.g., a typo generating an "undefined variable" error message).

Many critics of Visual Basic explain that the simple nature of Visual Basic is harmful in the long run. Many people have learned VB on their own without learning good programming practices. Even when VB is learned in a formal classroom, the student may not be introduced to many fundamental programming techniques and constructs, since much of the functionality is contained within the individual components and not visible to the programmer. Since it is possible to learn how to use VB without learning standard programming practices, this often leads to unintelligible code and workarounds. Second, having many of the checks and warnings that a compiler implements turned off by default may lead to difficulties in finding bugs. Experienced programmers working in VB tend to turn such checks on.

Many of the criticisms fired at Visual Basic are in fact criticisms of its ancestor, BASIC. A famous formulation by Edsger Dijkstra was, "It is practically impossible to teach good programming to students that have had a prior exposure to BASIC: as potential programmers they are mentally mutilated beyond hope of regeneration ." (Dijkstra was no less scathing about FORTRAN, PL/I, COBOL and APL.)

However, many proponents of Visual Basic explain that the simple nature of Visual Basic is its main strength, allowing very rapid application development to experienced Visual Basic coders and a very slight learning curve for programmers coming from other languages. Additionally, Visual Basic applications can easily be integrated with databases, a common requirement. For example, by using controls that are bound to a database, it is possible to write a VB application that maintains information within the database without writing any lines of VB code.

Visual Basic is also a conglomerate of language features and syntax, with less consistency, but more tolerance, than many modern programming languages. Many language features like GoSub, On Error, and declaring the type of a variable by the last character in the name (i.e. str\$) are legacies from Visual Basic's BASIC roots, and are included for backward-compatibility. The syntax of VB is different than most other languages, which can lead to confusion for new VB programmers. For example, the statement "Dim a, b, c As Integer" declares "c" as integer, but "a" and "b" are declared as Variant. Other characteristics include the entry of keyword, variable and subroutine names that are not case sensitive, and an underscore "_" must be used for a statement to span multiple lines. Some Visual Basic programmers perceive these as strengths needed to avoid case-sensitive compiler errors, and accidentally omitting line-termination characters some languages require (usually semicolons). For example, the ability to enter variable and subroutine names in any case can be used to the programmer's advantage: by declaring all names in mixed case, but entering them in lower case elsewhere, allows the programmer to type faster and to detect typos when a token remains in lower case.

13. The language continues to attract much praise and criticism, and it continues to cater to a large base of users and developers. The language is well suited for certain kinds of GUI applications (e.g., front end to a database), but less suited for others (e.g., compute-bound programs). Its simplicity and ease of use explain its popularity as a tool for solving business

problems — most business stakeholders do not care about technical elegance and effectiveness, and concentrate instead on the cost effectiveness of Visual Basic.

4.3.1 Programming constructs not present in Visual Basic

Inheritance. Visual Basic versions 5 and 6 are not quite object oriented languages as they do not include implementation inheritance. VB5 and 6 do, however include specification of interfaces. That is, a single class can have as many distinct interfaces as the programmer desires. VB.NET implements the full set of object-oriented features. Visual Basic provides a specific syntax for access to attributes called Property methods, and this is often implemented using getters and setters in C++ or Java. Python has an equivalent notation to VB6's property Let and Get.

Threading support not present prior to Visual Basic .NET.

Structured exception handling prior to Visual Basic .NET. Error handling is controlled by an "On Error" statement.

Built-in support for bit shifting. This feature appears only in Visual Basic .NET 2003 (7.1) or higher.

Typecasting. VB instead has conversion functions.

Equivalents to C-style pointers are very limited.

Visual Basic is limited to unsigned 8-bit integers and signed integers of 16 to 64 bits. Many other languages provide wider range of signed and unsigned integers.

32-bit and 64-bit Visual Basic is internally limited to UTF-16 strings, although it provides conversion functions to other formats (16-bit Visual Basic is internally limited to ASCII strings).

Visual Basic doesn't allow *constant* variables to contain an array. Therefore extra processing is required to emulate this.

While Visual Basic does not naturally support these features, programmers can construct work-arounds to give their programs similar functionality if they desire.

4.3.2 Behaviors present in Visual Basic

Visual Basic has the following uncommon traits:

Boolean constant True has numeric value -1. In most other languages, True is mapped to numeric value 1. This is because the Boolean data type is stored in the same way as a 16 bit signed integer. In this construct -1 evaluates to 16 binary 1s (the Boolean value True), and 0 as 16 0s (the Boolean value False). This is apparent when performing a Not operation on a 16 bit signed integer value 0 which will return the integer value -1. This inherent functionality becomes especially useful when performing logical operations on the individual bits of an integer such as And, Or, Xor and Not.

Logical and bitwise operators are unified. This is unlike all the C-derived languages (such as Java or Perl), which have separate logical and bitwise operators.

Variable array base. Arrays are declared by specifying the upper and lower bounds in a way similar to Pascal. It is also possible to use the Option Base statement to set the default lower bound. Use of the Option Base statement can lead to confusion when reading Visual Basic code and is best avoided by always explicitly specifying the lower bound of the array. This lower bound is not limited to 0 or 1, because it can also be set by declaration. In this way, both the lower and upper bounds are programmable. In more subscript-limited languages, the lower bound of the array is not variable. This uncommon trait doesn't exist in Visual Basic .NET and VBScript.

Ability to run the application without performing a full compile or making an executable, allowing for edit-and-continue changes.

Relatively strong integration with the Windows operating system.

Banker's rounding as the default behavior when converting real numbers to integers.

Integers are automatically promoted to reals in expressions involving the normal division operator (/) so that division of an odd integer by an even integer produces the intuitively correct result. There is a specific integer divide operator (\) which does truncate.

By default, if a variable has not been declared or if no type declaration character is specified, the variable is of type Variant. However this can be changed with Deftype statements such as DefInt, DefBool, DefVar, DefObj, DefStr to name a few. There are 12 Deftype statements in total offered by Visual Basic 6.0

4.3. Visual Basic and VB.NET

Visual Basic .NET, VB.NET, is a backwards-incompatible redesign of Visual Basic to Microsoft's .NET platform. Almost all of the above criticisms have been addressed with many of the missing features added. VB.NET has support for threading, advanced object oriented

code, Try-Catch-Finally blocks. Many new features (mainly from the .NET framework) have also appeared, like remoting, web services, 64-bit integers and ADO.NET.

VB.NET is also a fully-compiled language (as opposed to previous versions which could both compile and interpret the language). Programs require compilation even if the program is only to be debugged. This resulted in a number of convenient features being removed from Visual Basic, including the quick execution of programs, and the famous edit-and-continue feature (this feature has been restored in Visual Basic 2005). Visual Basic 2005 also includes the "MY" namespace, which gives the developer a vast amount of commonly used functionality.

Note: Microsoft has officially dropped the ".net" within the Visual Studio product name for 2005.

Many of the original critics of Visual Basic now praise VB.NET for providing a "complete" language, while a few supporters of Visual Basic claim VB.NET has made the language too complicated and too hard to use for simple application development. Another criticism of VB.NET is the incompatibility and lack of similarity in syntax. VB.NET provides a wizard to help upgrade code, but many features are not converted properly. The wizard produces a list of places in the code where the upgrade is incomplete, and large projects have many thousands of such places requiring significant programmer time to complete the upgrade. In particular the *Variant* data type, which was the default data type, is no longer supported. It is replaced by the Object type. Programs written in VB.NET must be manually converted to VB6 code if this is desired.

Some believe VB.NET support will diminish, with C# becoming the preferred language for .NET programming. This is despite the fact that both languages compile to the same .NET Common Intermediate Language, with the programming language choice merely a matter of syntax preference.

4.4. Evolution of Visual Basic

VB 1.0 was introduced in 1991. The approach for connecting the programming language to the graphical user interface is derived from a system called *Tripod* (sometimes

also known as *Ruby*), originally developed by Alan Cooper, which was further developed by Cooper and his associates under contract to Microsoft.

Timeline of Visual Basic before Visual Basic .NET

Visual Basic 1.0 (May 1991) was released for Windows.

Visual Basic for MS-DOS

Visual Basic 1.0 for DOS was released in September 1992. The language itself was not quite compatible with Visual Basic for Windows, as it was actually the next version of Microsoft's DOS-based BASIC compilers, QuickBASIC and BASIC Professional Development System. The interface was barely graphical, using extended ASCII characters to simulate the appearance of a GUI.

Visual Basic 2.0 was released in November 1992. The programming environment was easier to use, and its speed was improved.

Visual Basic 3.0 was released in the summer of 1993 and came in Standard and Professional versions. VB 3 included version 1.1 of the Microsoft Jet database engine that could read and write Jet (or Access) 1.x databases.

Visual Basic 4.0 (August 1995) was the first version that could create 32-bit as well as 16-bit Windows programs. It also introduced the ability to write classes in Visual Basic.

With version 5.0 (February 1997), Microsoft released Visual Basic exclusively for 32-bit versions of Windows. Programmers who preferred to write 16-bit programs were able to import programs written in Visual Basic 4.0 to Visual Basic 5.0, and Visual Basic 5.0 programs can easily be converted with Visual Basic 4.0. Visual Basic 5.0 also introduced the ability to create custom user controls, as well as the ability to compile to native Windows executable code, speeding up runtime code execution.

Visual Basic 6.0 (Mid 1998) improved in a number of areas, including the ability to create web-based applications. VB6 is currently scheduled to enter Microsoft's "non-supported phase" starting March 2008.

In April 2005 Microsoft announced that support for non .NET versions of Visual Basic would end within a few years. The Visual Basic community instantly expressed its concern and lobbied users to sign a petition to keep the product alive. Microsoft has so far refused to change their position on the matter. Ironically, around this time, it was exposed that Microsoft's new anti-spyware offering, Microsoft AntiSpyware, was coded in Visual Basic 6.0 (although this can be explained by the fact that the product was "inherited" with Microsoft's acquisition of GIANT). Windows Defender Beta 2 was rewritten as C++/CLI code, as mentioned in Paul Thurrott's review of this product.

Timeline of Visual Basic .NET

Visual Basic .NET was launched in 2002 along with the .NET Framework. Its language features are much richer than previous versions, although it is more complex. VB .Net is not backwards compatible, so many older VB programs must be modified to remove features incompatible with VB .Net (e.g., non-zero base arrays, the use of Variant, etc.)

Visual Basic .NET 2003 was launched in 2003 along with the .NET Framework 1.1.

In 2004 Microsoft released a beta version of Visual Studio.NET 2005 (codename Whidbey). This included a beta of version 2.0 of Visual Basic .NET

Also in 2004, Microsoft announced a return to offering support for Visual Basic hobbyists with the announcement of Visual Basic Express, and Visual Web Developer Express. Both are reduced feature versions of Visual Studio 2005 and support Visual Basic.NET 2.0.

On November 7 2005 Visual Studio 2005 was released, which includes Visual Basic .NET 2005 along with the .NET Framework 2.0. Microsoft also introduced Visual Basic 2005 Express Edition, a cut-down free edition designed to introduce people to the Visual Basic .NET environment.

In early 2006, Microsoft started making available community technology preview releases of Visual Basic 9.0 ("Orcas"), primarily focusing on the addition of Language integrated query, but also adding several new language features such as anonymous types, nullable types, and nested functions.

CHAPTER 5: DESCRIPTION ABOUT MY PROJECT

Now, I want describe my project in details step by step. My project is pharmacy automation. I create this automation by using Microsoft Visual Basic 6.0 Profession Edition with service pack 3 and Microsoft Access XP. Also I used some SQL.

5.1. Software Requirement Document

First Step of my project was read software requirement document which is prepared by my brother who is wife is a pharmacist. I learn about pharmacy shop operations more. What they need to manage a pharmacy shop. They need sales operations and manage orders. See the earning of shop. After reading this document I decide want can be the main forms of project. These forms are:

- Sales
- Order
- Medicine Information
- User Information

5.2 Starting a Basic 6.0 Profession Edition with service pack 3 Project

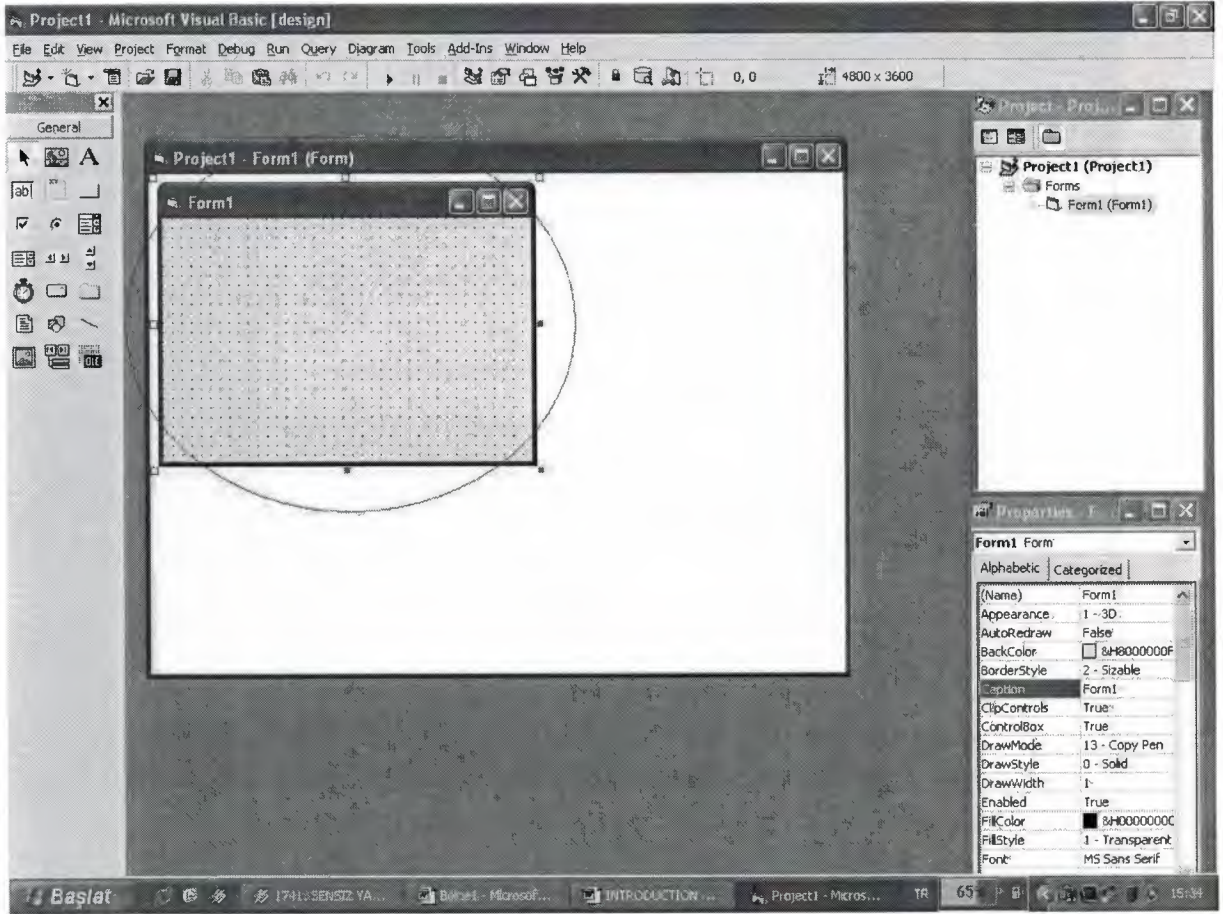
I mention about Visual Basic before but here I mention it an easy way. Microsoft Visual Basic 6.0 Profession Edition with service pack 3 has many special tools to create a project. After opening Visual Basic it ask to user to select what kind of project user wants to do. I clicked Standard Exe then Aç(open) button.



Starting a New Project

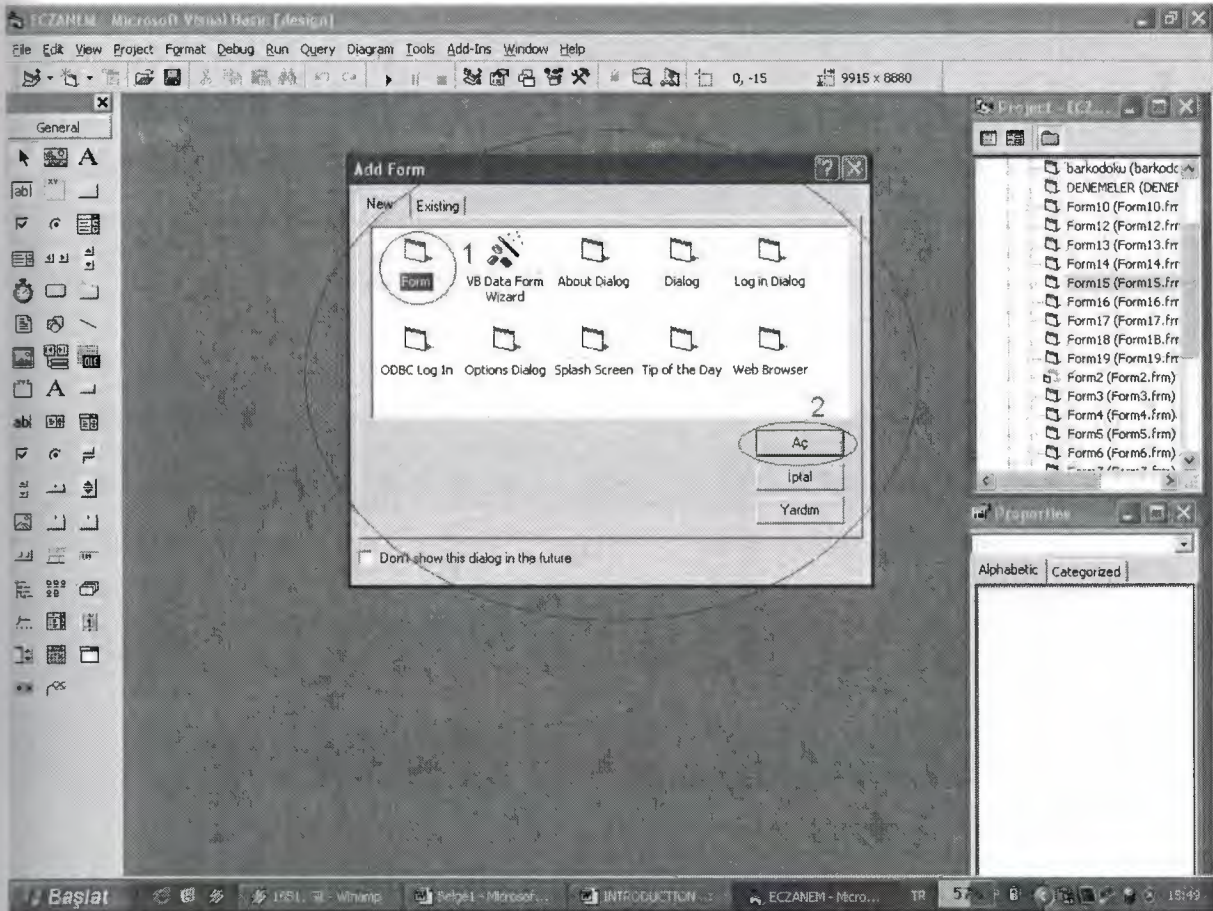
5.2.1 Designing of forms

Designing a form is very easy with Visual Basic. After starting a project it has already a form which is called form1 ready to use.



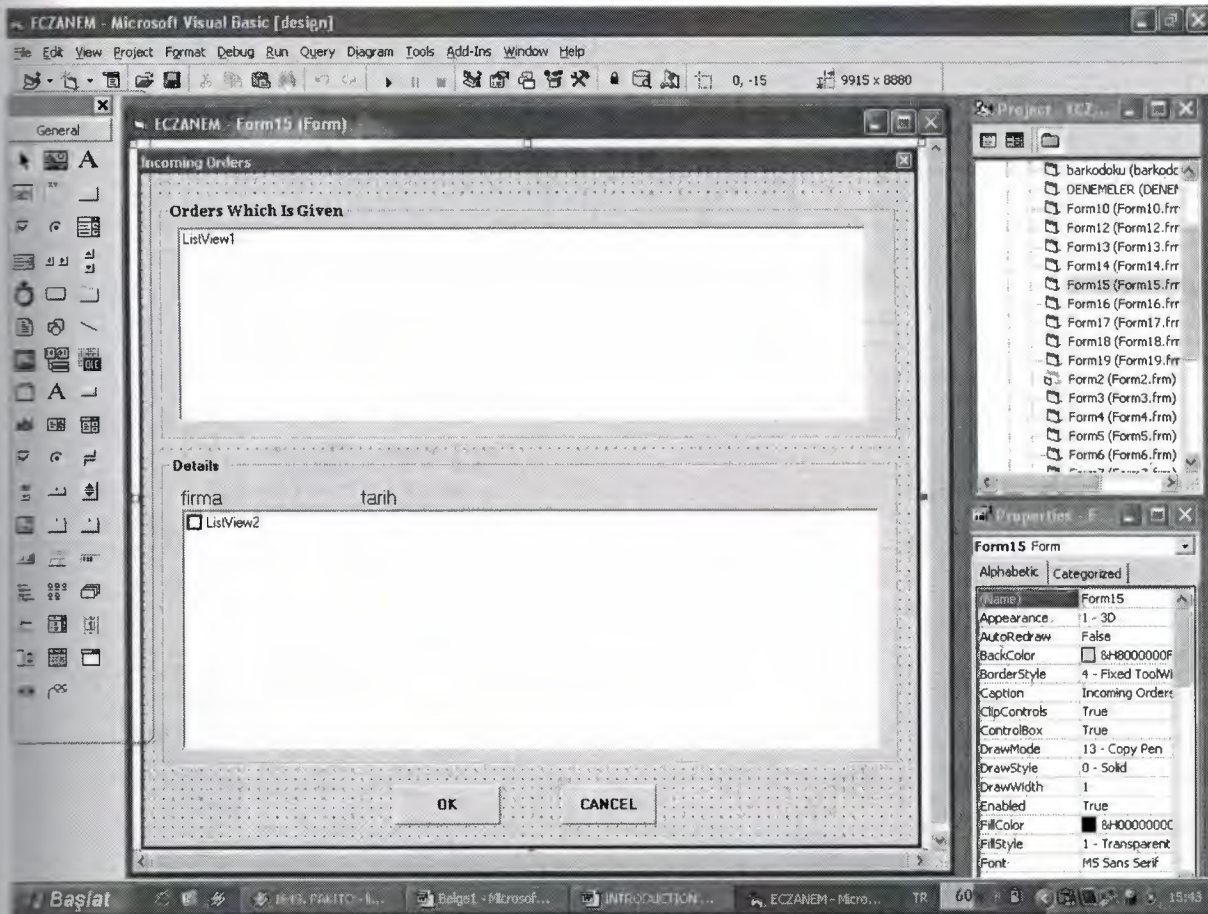
Form1

This Form1 has a name form1 and has a caption Form1. To add a new form I select project menu of Visual Basic and click add form. Then below figure appears. I click the form then aç(open) button. New form appears on the screen and I can easily add control to it and change its properties. I can rename it, remove it or add it to my project.



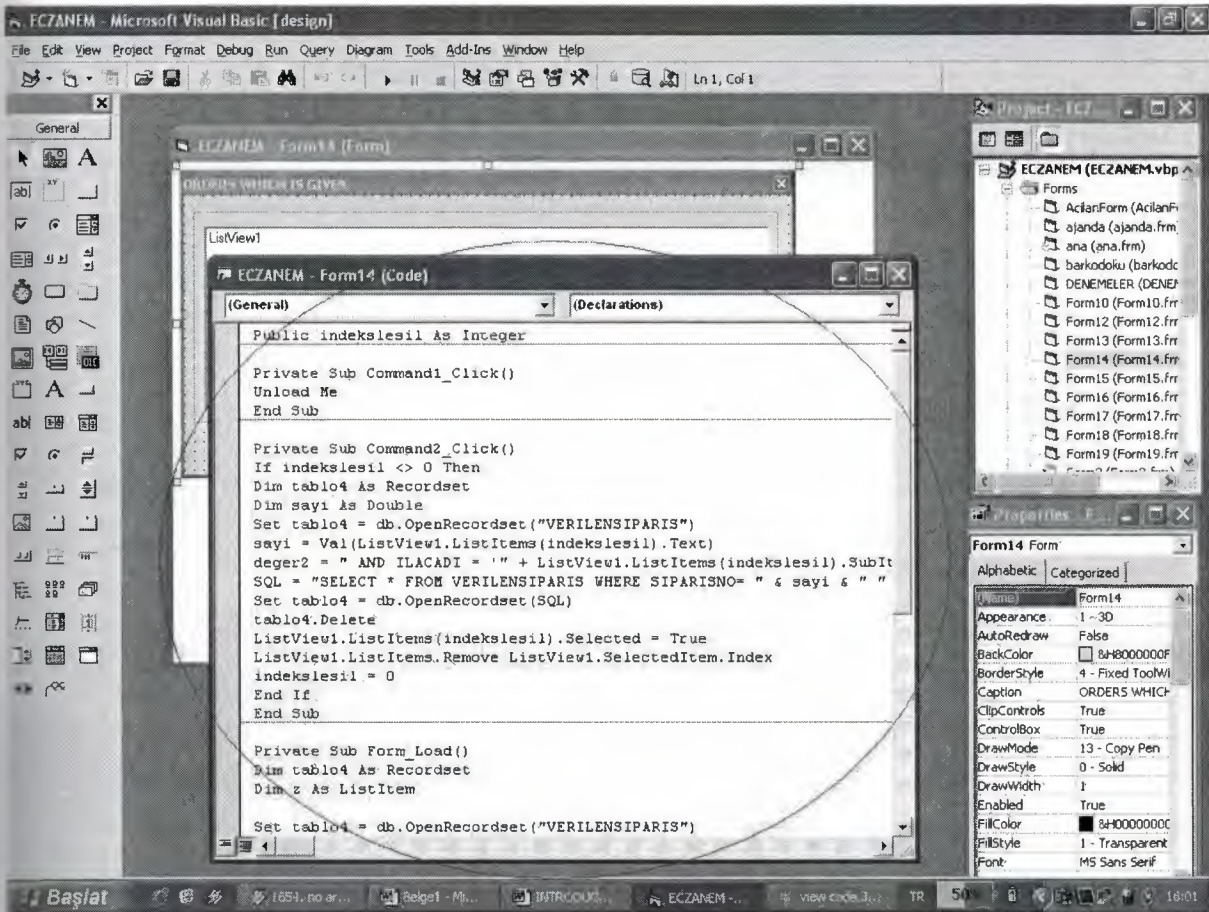
Adding a form to a project

By using tools at the left hand side I can add controls to my form. In below figure you can see the tools which I used in my project.

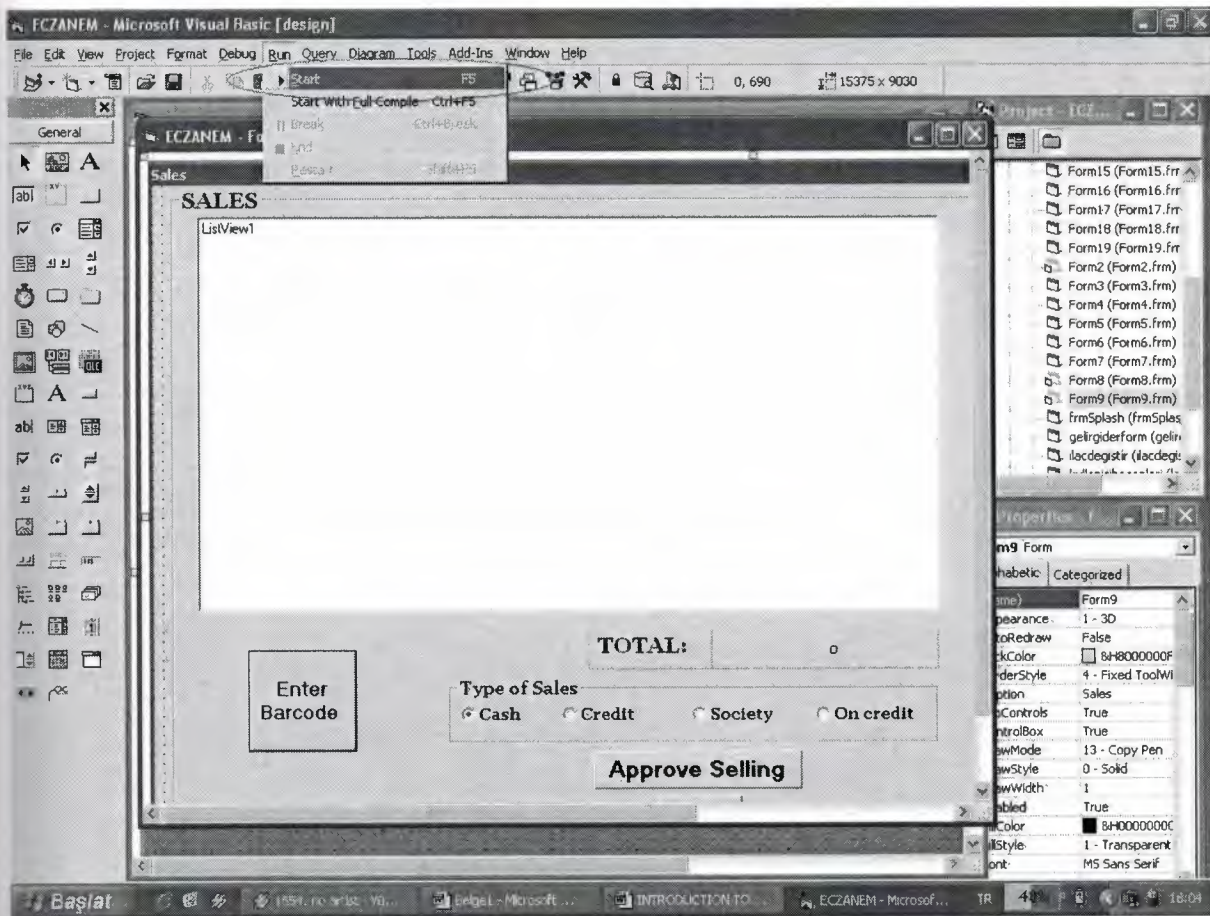


Tools

To see and the change the codes of a form I right click the mouse and I click the view code. Then I can easily change the codes of that form or that control.

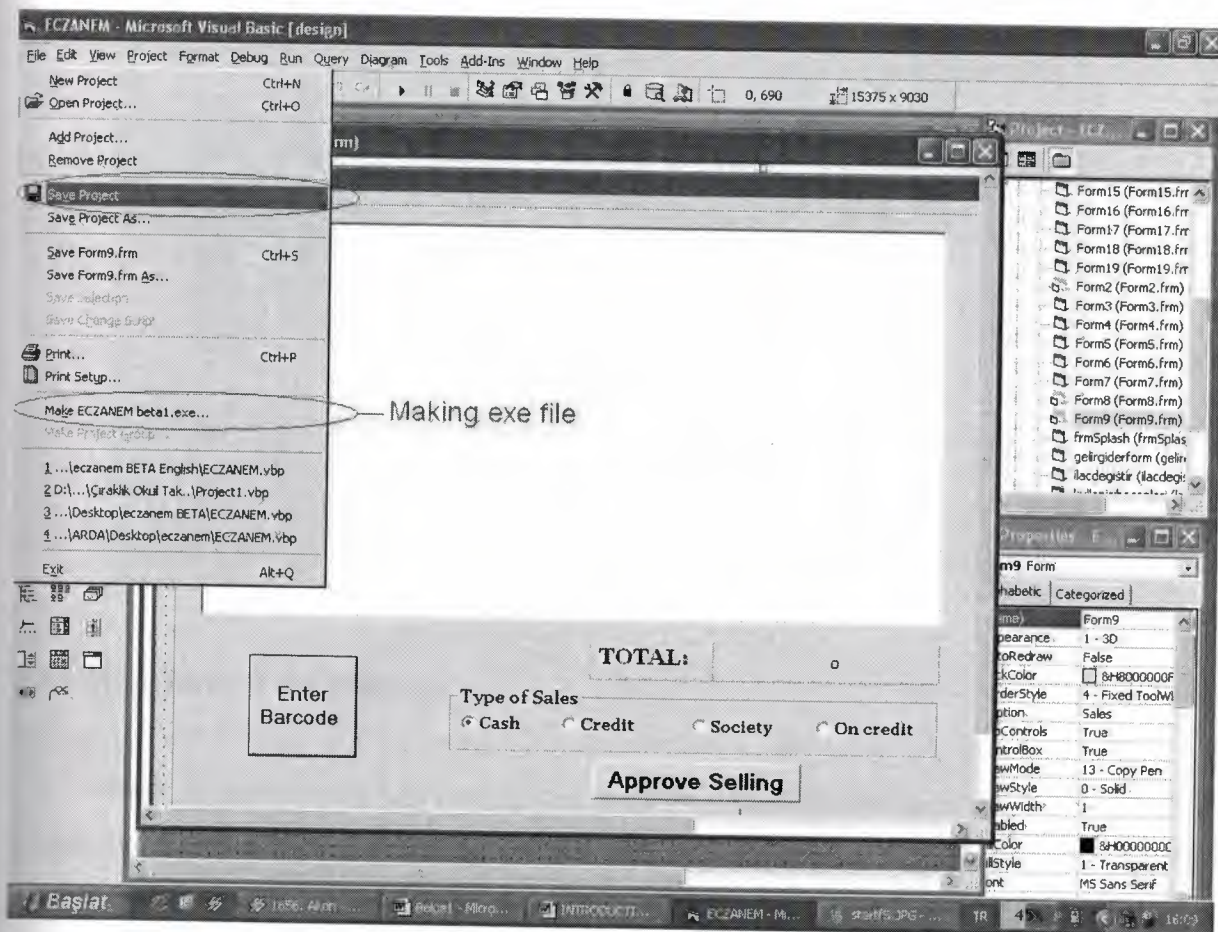


After change the code or add new feature to my program I click the run menu of Visual Basic and the Start Button. Or easily press F5 button on keyboard. The project starts to run if there is no error on code.



Start running of a project

After test the project if everything okay about the changing code I click the Visual Basic Menu File and then Save Project button. All the changes will be saved.

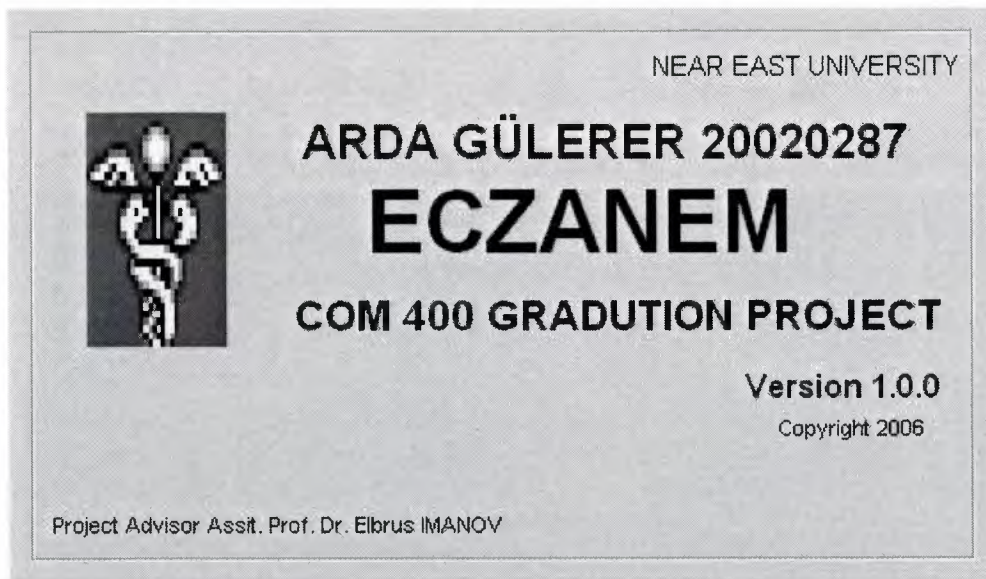


Saving a project and Making EXE file

Making Executable file of the project is very easy. By clicking File Menu of Visual Basic there is Make EXE selection here.

5.3 Working with ECZANEM

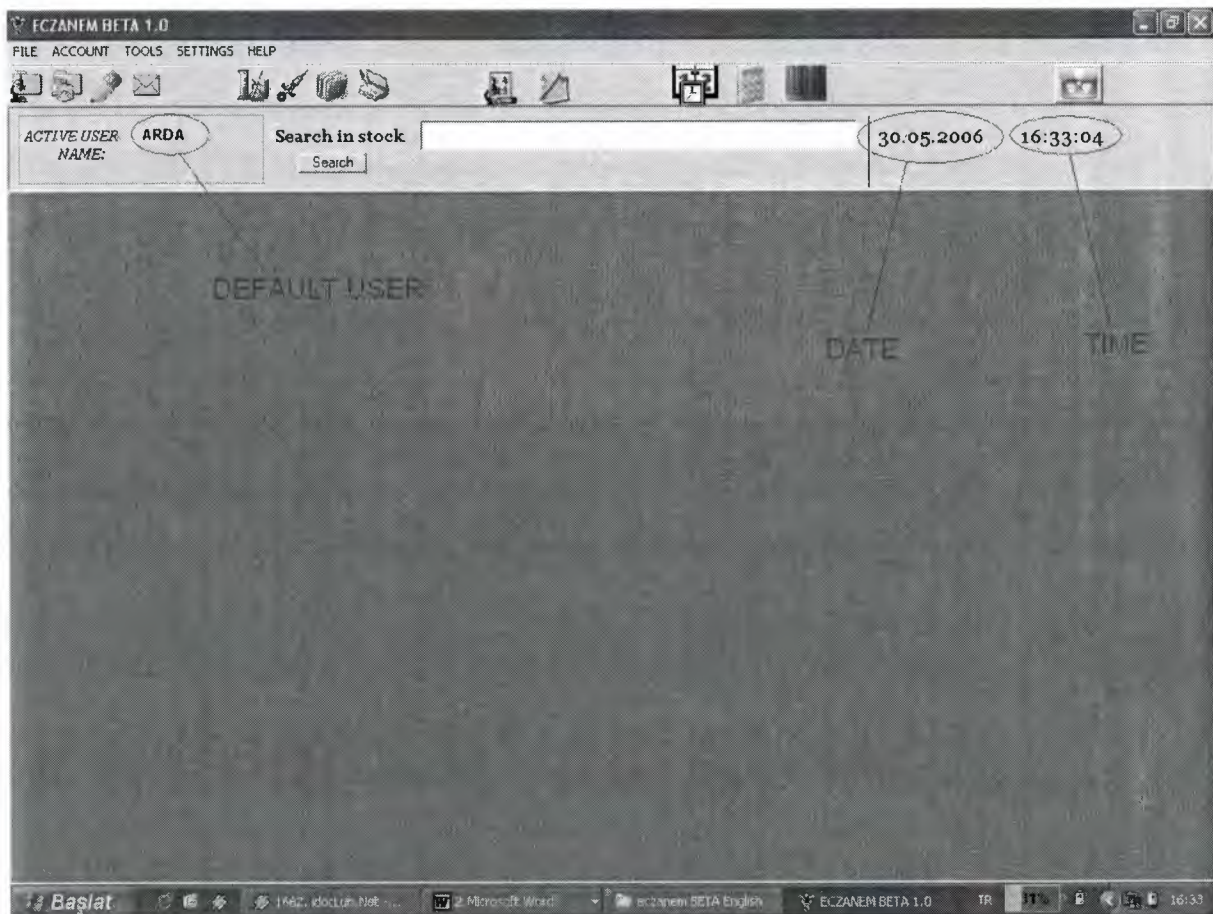
To start the use the ECZANEM Program Click the executable file icon of the ECZANEM. A splash screen welcomes to you. This splash screen show the program icon, name of university, program name, program version, when it built, project programmer and the project advisor information. This splash screen is show only two seconds. The main program starts with default variable. Default user name is Arda and the default firm is Selçuk Ecza you can change the default variables in the main program by clicking its menu. There is more information about default variable in its chapter.



Splash Screen of the ECZANEM

5.3.1 Main form of the Program

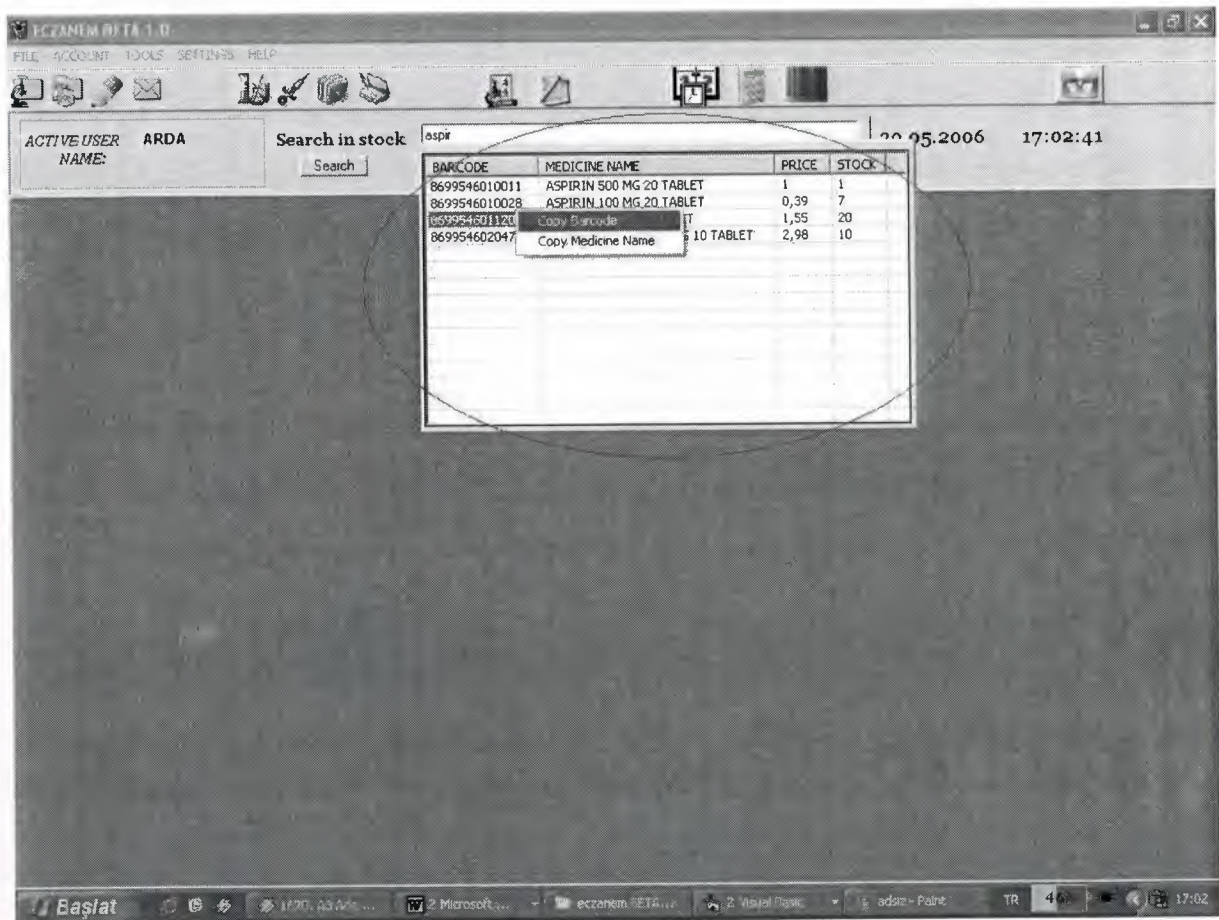
After splash screen Main Form of the program Show. This is the main screen of the program and it is always show when you are executing the program. My program uses MDI form structure. So you can minimize the some of the other important forms of the program. The figure of the main form of the program is bellow:



The Main Screen of the Program

You can see the user of the program and the current date and the current time in the screen every time. Also there is a text box which is using the search a medicine in the stock. And there is a button which is called SEARCH to make medicine search in details. And there is toolbar to easy access to sub forms of the program.

When you enter a text to this text box an opening form appears immediately and show the medicine details which medicine name starts with this text. You can see medicine details such as medicine barcode, medicine name, price and stock information of those medicines. You can easy copy their barcode or their medicine name information to use at another form of the program by clicking right mouse button while your mouse pointer is on the medicine information. Then this information copied to the clipboard and ready to paste any where you want.



Opening Form for Easy Search Medicine by Name

5.3.2 Sales Form of the Program

The Sales Form of the program is the most using part of the program because there are about six thousand of medicine and many customers of these medicines. There are about one hundreds of customer of a small pharmacy shop in a day. Every customer can buys more than one medicine in a day. And some of the customer use cash money, some of them uses credit card, some of them has society health insurance and some of them wants to buy medicine on credit. So the pharmacist need manage the selling operation. By using ECZANEM Sales form pharmacist easily manage this selling operation. Pharmacist can see medicine details medicine price, total amount of selling, how many medicine he/she is selling. Also pharmacist can see total amount of cash money earnings and total earnings. Pharmacist can cancel selling, take return back a medicine now or later. And manage on credits easily. The figure of the Sales Form is bellow.

ECZANEM BETA 1.0 - [Sales]

FILE ACCOUNT TOOLS SETTINGS HELP

ACTIVE USER: ARDA NAME: Search in stock 30.05.2006 17:12:35

SALES

BARCODE	MEDICINE NAME	PRICE
8699546011209	ASPIRIN FORT 20 TABLET	1.55
8699560010349	PANALGINE 300 MG 50 TABLET	1.82
8699560010745	PAROL 500 MG 30 TABLET	1.48

Medicines which is selling now

How many medicines selling: 3 MEDICINE(S) Total price of this selling: 4.85

Enter Barcode

To add medicine to this selling

Type of Sales: ☒ Cash ☐ Credit ☐ Society ☐ On credit

Approve Selling **CANCEL**

RETURN

RETURN NOW

AFTERWARD RETURN

PAYMENTS

ON CREDITS

SOCIETY PAY DETAILS

CASH: 5.6 YTL

TOTAL EARNINGS: 14.51 YTL

Başlat 1673. www.ump3.com 2 Microsoft Word 2 Visual Basic Bilgisayar TR 56 17:12

Sales form of the ECZANEM

Pharmacist can sell medicine by pressing Enter Barcode button. When he/she clicked the button new form appears and asks for medicine barcode. The pharmacist enters the barcode of the medicine which is selling to the text box of Barcode form and then clicks the BY BARCODE button. Then automatically the medicine added to the list and shows its details there. And Total value and how many medicine value changes. If the user wants to add same medicine again he/she add it just pressing Enter button again without entering the barcode again. If the user wants to add different medicine all he/she must do is entering the barcode to the text box and click the BY BARKODE button. Barcode form figure is bellow:

ICZANIM HİTA 1.0 [Sales]

FILE ACCOUNT TOOLS SETTINGS HELP

ACTIVE USER: ARDA
NAME: Search in stock Search

30.05.2006 17:42:37

SALES

BARCODE	MEDICINE NAME	PRICE
9699546011209	ASPIRIN FORT 20 TABLET	1,55
9699560010349	PANALGINE 300 MG 50 TABLET	1,82
9699560010745	PAROL 500 MG 30 TABLET	1,48

Barcode

ADD MEDICINE

Barcode: 9699560010745 BY BARCODE

CANCEL CLOSE

SOCIETY PAY DETAILS

3 MEDICINE(S) **TOTAL:** 4.85

Type of Sales

☒ Cash ☐ Credit ☐ Society ☐ On credit

Approve Selling

CANCEL

CASH: 5.6 YTL

TOTAL EARNINGS: 14.51 YTL

Enter Barcode

Başlat 1453, Salarad - Yast... Microsoft Word Visual Basic Paint TR 83% 17:42

Barcode Form to Add Medicine to the Selling

While entering barcode the stock level of this medicine is checking and if it is bellow the Critical Level or it is bellow the 0 the user informed about it by message boxes. But when you added the medicine to the selling list the stock, cash and total earning value does not updated because the user did not approve the selling yet. To approve a selling the pharmacist must click the Approve Selling button. After clicking the Approve selling button the stock value of the medicines updated and the selling added to the GELIRGIDER table of the database. And the Cash and Total Earning values are updated.

The pharmacist can select the sales type clicking the option buttons. If the selling is cash then click cash, if the selling is credit then click credit and so on. Default value is cash.

Type of Sales

☒ Cash ☐ Credit ☐ Society ☐ On credit

Type of Sales

If the credit option button clicked it means that the customers pay by credit card. This selling adds to GELIRGIDER table and its type is credit. Cash value does not change but the Total Income value is updated.

If the society button is clicked it means that the customer pay by society insurance. Then the new form appears to collect customer information. After filling required information user clicks OK button. And this information added to KURUMSATIS table when the user approves selling. Society Customer form figure is bellow:

Date 30.05.2006 18:00:30

Total: 4.85 YTL

Society: ☒ Emekli Sandığı ☐ SSK ☐ İş Bank

Name:

Surname:

Telephone number:

OK **CANCEL**

Society Customer Information Form

If the On Credit option button is clicked it means that the customer pay by On Credit. Then the new form appears to collect information. After filling required information user click OK button. And this information added to VERESIYESATIS table when the user approves selling. On Credit Customer form is below:

Date 30.05.2006 18:09:18

Medicine: ASPIRIN FORT 20 TABLET

Total: 4.85 YTL

Name:

Surname:

Telephone number:

OK **CANCEL**

On Credit Form

If the customer does not want to buy a medicine which he/she wanted before, pharmacist clicks the medicine on the list and then “RETURN NOW” button. Medicine delete form list and total and how many medicine values are updated.

RETURN

RETURN NOW

AFTERWARD RETURN

PAYMENTS

ON CREDITS

SOCIETY PAY DETAILS

BUTTONS



If the customer does not want to buy any medicine which he/she wanted before, pharmacist clicks the “CANCEL” button any everything on the sales screen cleared.

If the customer bought some medicine before and some time later he/she wants to give back to shop, pharmacist enters barcodes of medicines and click the “AFTERWARDS RETURN” button. The total becomes minus. Then pharmacist clicks type of sales. When the pharmacist approves selling these medicines added to stock and this return added to “GELIRGIDER” table and total earning updates.

When pharmacist clicks the “ON CREDITS” button new form appears and show the On Credit Customers details. Pharmacist can see On Credits name, surname, telephone number, medicine barcode, medicine name, medicine price, payment and additional information. The payment information shows that on credit customer pay or not. Additional information shows that payment type. There two option button on this form. One of them is cash and the other one is credit. If the on credit customer pays his/her dept by cash user click the cash option button, if the on customer pays his/her dept by credit card user click the credit option button. Default value for these option buttons is cash. After selecting payment type user clicks the “DONE” button and payment information and additional information for the customer is changed. But the database does not change until the “OK” button is clicked. If the “OK” button is clicked KISIVERESIYE and GELIRGIDER table updated.

The screenshot shows a software window titled "On Credit". Inside, there is a table with the following data:

NAME	SURNAME	TELEPHONE	BARCODE	MEDICINE NA...	PRICE	DATE	PAYMENT	ADDITION
ÖZGÜR	ÇIKI	3332211	8699525092472	AMOKLAVIN BI...	17,1	29.05.2006	YES	CASH
ÖZGÜR	ÇIKI	3332211	8699514010241	APRANAX FD...	3,78	29.05.2006	YES	CREDIT
ARDA	GÜLERER	05339999999	8699548031601	KLACID MR 50...	66,08	30.05.2006	NO	

Below the table, there are two radio buttons under the heading "Collected":

- ☒ Cash
- ☐ Credit

There are two buttons at the bottom: "Done" and "OK". The "Done" button is circled in red.

On Credit Form

If the user click the “SOCIETY PAY DETAILS” button new form appears, and show the society payment details. Pharmacist can see the customer who has society insurance in details such as name, surname, telephone, date, time, barcode, medicine name, price, society and the additional information. When the society paid the money of the medicine, user clicks customer name and then click “DONE” button. Then additional information of that customer becomes “SOCIETYPAYED”. If the society does not pay the money of medicine user select customer name and then click the “Society Not Pay” button. Then additional information of that customer becomes “SOCIETYNOTPAID”. So pharmacist can contact the customer by telephone number and ask him/her to pay money himself/herself. If the pharmacist wants to delete a record then he/she clicks “Delete Selected” button. But database does not change until the “OK” button is clicked. When the “OK” button is clicked “KURUMSATIS” table is updated. If user wants to cancel process easily clicks the “CANCEL” button and database does not change.

NAME	SURNAME	TELEPHONE	DATE	TIME	BARCODE	MEDICINE NA...	PRICE	SOCIETY	ADDITIONAL IN...
ALI	VELI	3332732	29.05.2006	15:03:31	8699531...	ASEPAR 100 ...	1.66	"EMEKLİ..."	SOCIETYNOTP...
ALI	VELI	3332732	29.05.2006	15:03:31	8699531...	ASEPAR 100 ...	1.66	"EMEKLİ..."	SOCIETYPAYED
ALI	VELI	3332732	29.05.2006	15:03:31	8699531...	ASEPAR 100 ...	1.66	"EMEKLİ..."	SOCIETYPAYED
ARDA	GÜLERER	05339998877	30.05.2006	21:15:14	8699505...	SUPRADYN 30...	5.39	"SSK"	
ARDA	GÜLERER	05339998877	30.05.2006	21:15:14	8699505...	SUPRADYN 30...	5.39	"SSK"	

Buttons: Done, Society not Pay, Delete Selected, OK, CANCEL

Society Pay Detail Form

If the user wants the cancel sales form clicks “CANCEL” button of sales form and program exits form sales form.

5.3.3 Order Form of the ECZANEM

In a small pharmacy shop there are thousands of medicines. Some of the medicines like aspirin can be found in any pharmacy shop. There are some medicines Cancer Drug can be found only in big pharmacy shops because of their high price. Also there are 50-100 in stock aspirin even in a small shop but many of the medicines have only one in stock. In a day a pharmacy shop can sell more than hundred medicines. When a medicine is exhausted in stock pharmacist must order to firm to buy some. Usually pharmacist does not wait the stock to exhaust pharmacist decides a critical level for each medicine and when a medicine stock is under the critical level pharmacist orders to firm to buy some. Sometimes a medicine can not be found in firm stock because of not producing more these medicines called as Absent Medicine. Sometimes firm sends more than pharmacist wants or sends wrong medicine. So it is not easy to manage this without a help.

ECZANEM has a form to manage orders, this form called "ORDERS". In Orders form user can see a list for medicine that their stock level is under the critical level. Pharmacist can see order which is given, absent orders, incoming orders, return orders and easily he/she can create an order. The list of the medicines is updating in every opening of form by the program which is checking medicine by medicine stock level. This is done in less than a second. And at the list pharmacist sees the medicine barcode, medicine name, stock, critical level, price and the firm which is working with.

ECZANEM BETA 1.0 - [Orders]

FILE ACCOUNT TOOLS SETTINGS HELP

ACTIVE USER: ARDA NAME: Search in stock 30.05.2006 21:55:40

Search

ORDERS

Add Order(s)

BARCODE	MEDICINE NAME	STOCK	CRITICAL	PRICE	FIRM	ADDITIONAL
8699514010203	APRANAX 275 MG 10 TABL	1	2	1.76	SELÇUK ECZA	
8699546010011	ASPIRIN 500 MG 20 TABLET	1	2	1	SELÇUK ECZA	
8699505121192	SUPRADYN 30 DRAJE	1	2	5.39	SELÇUK ECZA	

ORDER WHICH IS GIVEN

ABSENT ORDERS

INCOMING ORDERS

RETURN ORDERS

Create ORDER

CANCEL

The Orders Form of the ECZANEM

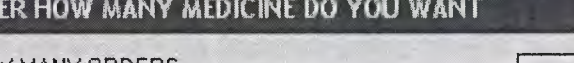
When the user want to create a new order he/she clicks the “CREATE ORDER” button at the bottom of the form and then a new form appears on the screen.

ORDER NOW

BARCODE	MEDICINE NA...	STOCK	CRITICAL LEVEL	PRICE	FIRM	HOW MANY
8699514010203	APRANAX 275 ...	1	2	1.76	SELÇUK ECZA	1
8699546010011	ASPIRIN 500 ...	1	2	1	SELÇUK ECZA	1
8699505121192	SUPRADYN 30...	1	2	5.39	SELÇUK ECZA	1

ADD ORDER NOW CANCEL

The Order Now Form of the ECZANEM



ENTER HOW MANY MEDICINE DO YOU WANT

HOW MANY ORDERS

1

OK Cancel

User can write how many of this medicine wants and click the “OK” button. If the user clicks the cancel button default value does not change.

The screenshot shows a window titled "ORDER NOW" with a close button in the top right corner. Inside the window is a table with the following columns: BARCODE, MEDICINE NA..., STOCK, CRITICAL LEVEL, PRICE, FIRM, and HOW MANY. The table contains three rows of data:

BARCODE	MEDICINE NA...	STOCK	CRITICAL LEVEL	PRICE	FIRM	HOW MANY
8699514010203	APRANAX 275 ...	1	2	1,76	SELÇUK ECZA	1
8699546010011	ASPIRIN 500 ...	1	2	1	SELÇUK ECZA	1
8699505121192	SUPRADYN 30...	1	2	1,5 29	SELÇUK ECZA	1

A context menu is open over the third row, with two options: "delete from this list" and "send it to the absent list". At the bottom of the window, there is a text input field, a button labeled "ADD", a button labeled "ORDER NOW", and a button labeled "CANCEL".

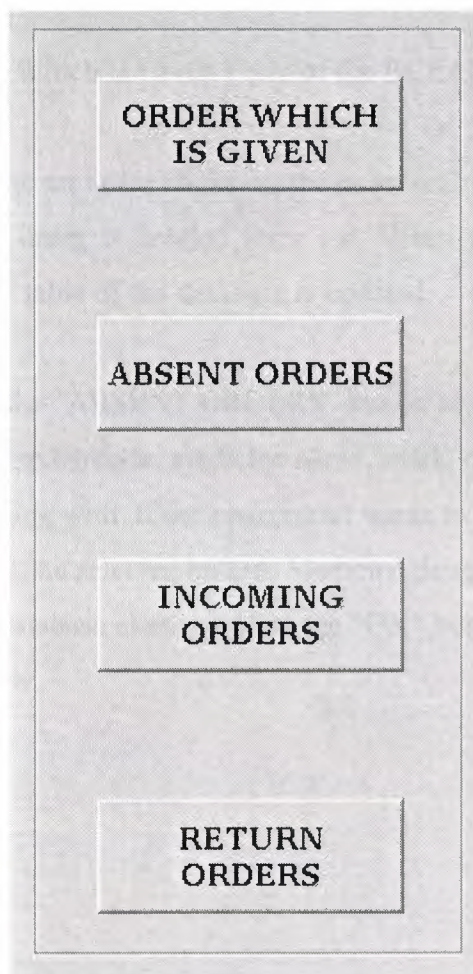
59

When the user clicks the “send it to the absent list” menu option the medicine information is add the “YOKTAKISIPARIS” table of the database of program. If the user clicks the “delete from this list” menu option then this medicine is only deleted from this list and does not send to anywhere.

Pharmacist can want to order any medicine which is not in the list he/she can add it by writing its barcode to the textbox. And then user clicks the “ADD” button on the form. If the barcode is not valid user inform about it by a message box. And if the medicine is already in the list user inform about it by a message box.

If pharmacist want to cancel this order clicks “CANCEL” button. And anything changes on the database.

When the pharmacist clicks the “ORDER NOW” button this orders add to the “VERILENSIPARIS” table of the database. By time and date information also.

A vertical rectangular panel with a light gray background and a thin black border. Inside the panel, there are four rectangular buttons stacked vertically, each with a thin black border and black text. The buttons are labeled: "ORDER WHICH IS GIVEN", "ABSENT ORDERS", "INCOMING ORDERS", and "RETURN ORDERS".

The Order Form Buttons of the ECZANEM

When the pharmacist wants to see orders which are given before, clicks to “ORDER WHICH IS GIVEN” button at the right. And the new form appears on the screen and shows the list of the orders by ordering number, ordered medicine name, firm and the time information. The figure of the orders which is given is below.

ORDER NUMBER	ORDER	FIRM	TIME
1	APRANAX 275 MG...	SELÇUK ECZA	29.05.2006 08:29:49
2	APRANAX 275 MG...	SELÇUK ECZA	30.05.2006 22:31:28
2	ASPIRIN 500 MG ...	SELÇUK ECZA	30.05.2006 22:31:28
2	SUPRADYN 30 D...	SELÇUK ECZA	30.05.2006 22:31:28

DELETE SELECTED OK

The Orders Which Is Given Form of the ECZANEM

If the pharmacist wants the delete an order clicks on the order and then clicks the “DELETE SELECTED” button. Then the order is deleted form list. When the user clicks the “OK” button the “VERILENSIPARIS” table of the database is updated.

When pharmacist clicks the “ABSENT ORDERS” button see the absent order medicine details. Such as medicine barcode, medicine name, stock, critical level, medicine price and the firm which is working with. If the pharmacist wants to delete a record from this list clicks on the medicine and delete selected button. Medicine delete from absent order list. But database does not change. Database changes when the “OK” button is clicked. The figure of the absent orders form is below.

BARCODE	MEDICINE NAME	STOCK	CRITICAL LEVEL	MEDICINE PRI...	FIRM
8699505121192	SUPRADYN 30 DRAJE	1	2	5,39	ŞELCUK ECZA
8699514010203	APRANAX 275 MG 1...	1	2	1,76	ŞELCUK ECZA

The Absent Orders Form of the ECZANEM

When the orders incomes to the pharmacy shop user clicks the “INCOMING ORDER” button. New form appears on the screen and it shows the details of the orders which is given. When user clicks any medicine in orders which is given list at above, any medicine has same ordering number appears in below list. The below list show the medicine information in detail, such as medicine barcode, medicine name, how many medicine, date and time information. There are checkboxes at the below list’s left hand side. These check boxes for the check the medicine is correct or not. If user clicks “OK” button , program check the below list. If the medicine check box is checked the stock of the medicine is updated by adding new incoming total amount of that medicine and delete from “VERILENSIPARIS” table on the database. If the medicine check box is not checked the medicine stays waiting at the “VERILENSIPARIS” table.

Incoming Orders

Orders Which Is Given

ORDER ...	ORDER	FIRM	ORDER TIME
1	APRANAX 275 MG 10 ...	SELÇUK ECZA	29.05.2006 08:29:49
2	APRANAX 275 MG 10 ...	SELÇUK ECZA	30.05.2006 22:31:28
2	ASPIRIN 500 MG 20 T...	SELÇUK ECZA	30.05.2006 22:31:28
2	SUPRADYN 30 DRAJE	SELÇUK ECZA	30.05.2006 22:31:28

Details

firma tarih

ORDER NUMB...	BARCODE	MEDICINE NA...	HOW MANY	DATE	TIME
<input type="checkbox"/> 2	8699514010203	APRANAX 275 ...	1	30.05.2006	22:31:28
<input checked="" type="checkbox"/> 2	8699548010011	ASPIRIN 500 ...	1	30.05.2006	22:31:28
<input type="checkbox"/> 2	8699505121192	SUPRADYN 30...	1	30.05.2006	22:31:28

Check Boxes

OK CANCEL

The Incoming Orders Form of the ECZANEM

The right mouse button menu of the incoming orders

When the user learned about absent medicine click the right mouse button and a sub menu appears. The menu has two options one of them is the “send to the absent orders” and the other option is the “send to the return orders”. If the user clicks the “send to the absent orders” option medicine information is added to the “YOKTAKISIPARIS” table on the database.

If the pharmacist wants to a medicine back to firm click the right mouse button on this medicine and submenu appears and pharmacist click the “send to the return orders”. So this medicine information is added to the “IADESIPARIS” table on the database.

User can cancel the incoming orders form by clicking “CANCEL” button. When “CANCEL” button is clicked there is no changing on the database.

When the user click “RETURN ORDERS” button, new form appears on the screen and shows the details of the return orders medicines. This form has list of return order medicine details, such as medicine barcode, medicine name, order information, return back and the additional information. When the firm has accepted the pharmacist return medicine user clicks on that medicine and then clicks the “ACCEPTED” button. After clicking the “ACCEPTED” button, the return back and the additional information of that medicine has changed to “YES” and time information of accepting firm. User can delete any medicine from the list by clicking on the medicine and then clicking “DELETE SELECTED” button. The “IADESIPARIS” table on the database does not change until the “OK” button is clicked. When the “OK” button is clicked, the “IADESIPARIS” table on the database is updated.

[illegible]

The user can search any medicine while on the orders menu by writing medicine name to the text box on the ECZANEM main form. It does not affect the order menu. But order menu affects the opening form of ECZANEM main form.

The user can cancel Order form of the ECZANEM by clicking “CANCEL” button on the order form bottom.

5.3.4. Medicine Information Forms

There are about six thousands of medicines on the “FIYAT” table on the database. The price of these medicines is deciding by the government and pharmacist can not sell medicines at different price. Government announce the new price for a medicine its web site. The pharmacist must update the price information of that medicine.

Sometimes new medicine is found and giving to the market. So the pharmacist should add this medicine to the database. And sometimes a medicine is forbidden to sell at the market because of the bad effect on the human or the medicine won't produce no more can be delete at the database.

To manage these operations ECZANEM has a form with there tab on it. Each tabs for different processes. First tab of the form is “UPDATE MEDICINE”. The pharmacist can find a medicine by entering its barcode to the text box than click “BY BARCODE” button. If it exist in database the medicine information show at the “MEDICINE INFORMATION” frame. This information is medicine barcode, medicine name, price, stock and the critical level. The medicine barcode and the medicine name can not change. Because, this is information is using on the almost all of the tables of database. And changing it can be dangerous. The user inform about this dangerous by showing images near these information. User can change price, stock and critical level here and then click “UPDATE” button. “FIYAT” table on the database is updated. To update a medicine user must fill all information text boxes. If user does not fill all of them, a message box appears on the screen and informs the user.

MEDICINE

UPDATE MEDICINES ADD MEDICINE DELETE MEDICINE

PLEASE ENTER MEDICINE BARCODE WHICH IS YOU WANT TO UPDATE

1

BARCODE: 8699546011209 FIND BY BARCODE

MEDICINE INFORMATION

BARCODE: 8699546011209

MEDICINE NAME: ASPIRIN FORT 20 TABLET

MEDICINE PRICE: 1.55

STOCK: 20

CRITICAL LEVEL: 2

2 UPDATE CANCEL

CLOSE

The Medicine Information Form Update Medicine Tab of the ECZANEM

When the new medicine is on the market pharmacist can enter its information from The Add Medicine Tab of The Medicine Information Form. User must fill all the information to add a medicine to "FIYAT" table of database. The new barcode and new medicine name can not be the same any medicine in the database. This causes an error a message box appears on the screen and informs the user. When the new medicine added to the database it is ready the use on the other forms of the ECZANEM program.

The screenshot shows a window titled "MEDICINE" with three tabs: "UPDATE MEDICINES", "ADD MEDICINE" (which is selected and highlighted), and "DELETE MEDICINE". Below the tabs, a message reads "PLEASE FILL ALL TO ENTER MEDICINE". Underneath this is a section titled "MEDICINE INFORMATION" containing five text input fields labeled "BARCODE:", "MEDICINE NAME:", "MEDICINE PRICE:", "STOCK:", and "CRITICAL LEVEL:". At the bottom of this section are two buttons: "ADD" and "CANCEL". A "CLOSE" button is located at the bottom right of the window.

The Medicine Information Form Add Medicine Tab of the ECZANEM

When pharmacist wants to delete a medicine from database clicks the delete medicine tab of the medicine information form of the ECZANEM. To delete a medicine user must enter its barcode to the text box and click "FIND BY BARCODE" button. If the barcode exists in database its information shows to the user at the medicine information frame. User control if it is true to sure to delete correct one. Then click the "DELETE" button and this medicine delete from "FIYAT" on the database. But to delete the medicine its stock must be greater than the critical level. This is because of the program stability.

MEDICINE

UPDATE MEDICINES ADD MEDICINE **DELETE MEDICINE**

PLEASE ENTER BARCODE OF MEDICINE WHICH IS YOU WANT TO DELETE

BARCODE: FIND BY BARCODE 1

MEDICINE INFORMATION

BARCODE:	8699522095711
MEDICINE NAME:	AUGMENTIN BID 1000 MG 10 FILMTABLET
MEDICINE PRICE:	17,1
STOCK:	10
CRITICAL LEVEL:	2

2

DELETE CANCEL

CLOSE

Delete Medicine tab of the Medicine Information Form

The user can cancel delete operation by clicking “CANCEL” button. If the “CANCEL” button is clicked nothing changes on database.

And user can close this form clicking “CLOSE” button at any tab of the form.

5.3.5. The STATICS Form of the ECZANEM

The pharmacist wants to see all the money operations on the screen. Pharmacist can learn about what is going on at the program by click ECZANEM menu * account * STATICS. User can see the all the details here such as seller, date, time, type of sale,

barcode, medicine name, price and additional information at the list. To see all the operation user clicks the “SHOW ALL” button.

SELLER	DATE	TIME	TYPE OF SALE	BARCODE	MEDICINE NAME	PRICE	ADDITIONAL INFO...
ARDA	30.05.2006	17:11:39	CASH	8699514010241	APRANAX FORT...	3.78	
ARDA	30.05.2006	17:11:39	CASH	8699560010349	PANALGINE 300 ...	1.82	
ARDA	30.05.2006	17:11:53	CREDIT	8699502700722	ASIDAL 150 ML S...	2.97	
ARDA	30.05.2006	17:11:53	CREDIT	8699502700722	ASIDAL 150 ML S...	2.97	
ARDA	30.05.2006	17:11:53	CREDIT	8699502700722	ASIDAL 150 ML S...	2.97	
ARDA	30.05.2006	20:54:54	ONCREDIT	8699548031601	KLACID MR 500 ...	0	
ARDA	30.05.2006	21:13:57	SOCIETY	8699505121192	SUPRADYN 30 D...	5.39	
ARDA	30.05.2006	21:13:57	SOCIETY	8699505121192	SUPRADYN 30 D...	5.39	
ARDA	30.05.2006	21:14:11	CASHRETURN	8699505121192	SUPRADYN 30 D...	-5.39	
ARDA	30.05.2006	21:14:11	CASHRETURN	8699505121192	SUPRADYN 30 D...	-5.39	
ARDA	30.05.2006	21:14:24	CASHRETURN	8699505121192	SUPRADYN 30 D...	-5.39	
ARDA	30.05.2006	21:15:14	SOCIETY	8699505121192	SUPRADYN 30 D...	5.39	
ARDA	30.05.2006	21:15:14	SOCIETY	8699505121192	SUPRADYN 30 D...	5.39	

☐ Cash
 ☐ Credit
 ☐ Cash Return
 ☐ Credit Return
 ☐ On Credit
 ☐ Society
 ☐ Society Return

Show All Show Selected Delete All Close

STATICS form of the ECZANEM

The pharmacist can want to see only the selected information on the screen. This is doing by “Show Selected” button. First select the types form check boxes then click the “Show Selected” button. User can select more than one type of sales.

☐ Cash
 ☐ Credit
 ☐ Cash Return
 ☐ Credit Return
 ☐ On Credit
 ☐ Society
 ☐ Society Return

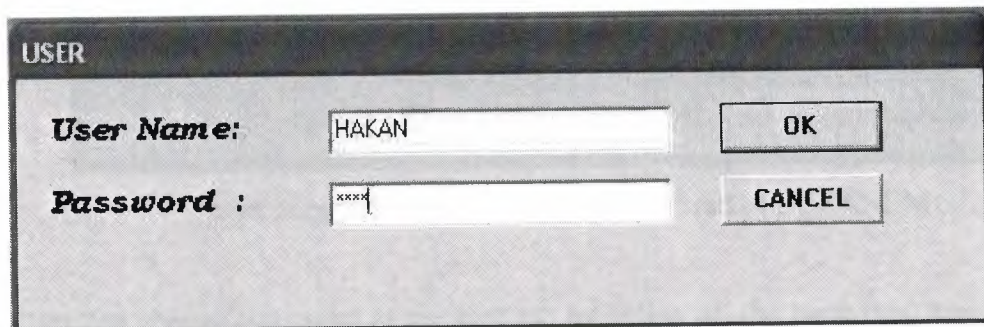
Check boxes of the STATICS form

User can delete any time all the “GELIRGIDER” table by clicking “Delete All” button. The “GELIRGIDER” table becomes empty.

The “CLOSE” button is for close the form. When user is clicked the “CLOSE” button the form is closed.

5.3.6 User Accounts

There more than one employee even at a small pharmacy shop. Sometimes it is important to learn who made a sale. Because of the find if there is wrong operation. This is managing by creating user account and using it at the program. But this causes a speed problem in real problem every time of sales change the users spend time and every opening of the program entering a username and password also spend time. So ECZANEM uses a default user which is called "ARDA". This user name is ready when the program is opened. User can change this by menu * setting * defaults. But at a important selling user enter his/her user name and password to change the user name which is adding the database. User can click menu * change user. Then the below figure appears:



User entrance form of the ECZANEM

The user name and password textboxes must be filled both and then user must click "OK" button. If this information is correct user name of the program is changes. But if it is wrong user inform about this by a message box and user name of the program does not change.

We can add new account, change password of existing account and delete an account by user accounts form. To open this form; user clicks menu * account * user accounts. The user accounts form appears on the screen and it has there tabs on it.

USER ACCOUNTS

CHANGE PASSWORD **CREATE NEW ACCOUNT** **DELETE ACCOUNT**

USER NAME:

PASSWORD:

NEW PASSWORD:

NEW PASSWORD AGAIN:

CHANGE **CANCEL**

CLOSE

User accounts form change password tab of ECZANEM

User can change password at the first tab by filling all the textboxes. User name and old password must be correct. And new password and new password again data should be match each other. This is very important to change password. Then user clicks the "CHANGE" button. Then the "KULLANICI" table of the database is updated. If the user wants to cancel changing password clicks to the "CANCEL" button. if the "CANCEL" button clicked nothing change on the database.

User can add a new user account at the second tab of the user accounts form. User should fill all the text boxes. Password and Password Again data should be match each other. If the user name is already using user inform about this by a message box. If the user name is not using before it adds to the "KULLANICI" table of the database.

The image shows a Windows-style dialog box titled "USER ACCOUNTS". It has three tabs: "CHANGE PASSWORD", "CREATE NEW ACCOUNT" (which is selected and highlighted with a dotted border), and "DELETE ACCOUNT". Inside the dialog, there are three text input fields labeled "USER NAME:", "PASSWORD:", and "PASSWORD AGAIN:". Below these fields are two buttons: "ADD" (which is highlighted with an oval) and "CANCEL". At the bottom right of the dialog is a "CLOSE" button. The dialog box has standard Windows window controls (minimize, maximize, close) in the top right corner.

Create new account tab of the user account form of the ECZANEM

Pharmacist can delete any user account by entering its user name and password. Then click to the "DELETE" button. If the user in the database it is deleted from "KULLANICI" table. If it is not in the database an error occur and pharmacist is inform about it.

At any tab user can close the form by clicking "CLOSE" button.

5.3.7 Default Variables

Pharmacist can change the default variable of the program by menu * setting * defaults. In this form the firm and user default variables show to pharmacist. These variables saved at the "kullanici.txt" and "firma.txt" files. Pharmacist can enter new value for default variable and then click "CHANGE". The file then updated. When the program is opening again it starts with the new variables.

DEFAULT VARIABLES

CHANGE DEFAULT VARIABLES

DEFAULT FIRM: ŞELCUK ECZA

DEFAULT USER: ARDA

Default Variables Form of the ECZANEM

5.3.8 The Search Form of the ECZANEM

This form for make search operations. The user can select search criteria by a combo box. This combo box has two value one of them medicine name and the other one is medicine barcode. Default value is medicine name. If you find a medicine by its name select the medicine name enter medicine name to the text box and then click the search button. All the medicine which is includes that text is shown at the list box. You can see how many item found in this search at the bottom.

If you want to make search by barcode select the medicine barcode from combo box and write barcode to the text box and then click the search button. Every medicine which includes that text is shown at the list box. If there no record at database which includes this text "Not found" message box appears and inform the user.

User can select the one of the found medicine to copy its information. When the user selects a medicine at the bottom text box its information appears and ready to copy from there. User can select the text than right click the mouse and copy option. It is copied to the clipboard.

SEARCH

Search Criteria

MEDICINE NAME

aspirin

8699546010011 ASPIRIN 500 MG 20 TABLET FIYATI= 1 STOK= 1 KRITIKDUZEY= 2

8699546010028 ASPIRIN 100 MG 20 TABLET FIYATI= .39 STOK= 7 KRITIKDUZEY= 2

8699546011209 ASPIRIN FORT 20 TABLET FIYATI= 1.55 STOK= 20 KRITIKDUZEY= 2

8699546020478 ASPIRIN PLUS-C 400 MG 10 TABLET FIYATI= 2.98 STOK= 10 KRITIKDUZEY= 2

8699546010028 ASPIRIN 100 MG 20 TABLET FIYATI= .39 STOK= 7 KRITIKDUZEY= 2

4 find!

5.3.9 Tools of ECZANEM

5.3.9.1 Calendar

When the pharmacist want to see a calendar, he/she clicks menu * tools * calendar. And a month view appears on the screen and show month and today.

Mayıs 2006							
Paz	Sal	Car	Per	Cum	Cmt	Paz	
	24	25	26	27	28	29	30
1	2	3	4	5	6	7	
8	9	10	11	12	13	14	
15	16	17	18	19	20	21	
22	23	24	25	26	27	28	
29	30	31	1	2	3	4	
Today: 31.05.2006							

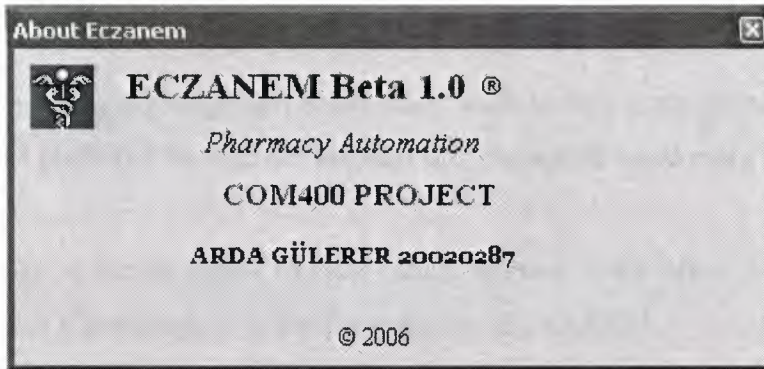
Calendar of ECZANEM

5.3.9.2 Calculator

The pharmacist can open windows calculator for calculations by clicking menu * tools * calculator.

5.3.10 the About Form of the ECZANEM

This form is give information about ECZANEM pharmacy automation program.



About form of the ECZANEM

Chapter 6: TABLES

İYAT BARKOD İLACADI İLACFIYATI STOK KRITIKDUZEY	GELİRGİDER SATISYAPAN SATISTARİH SATISSAAT SATISTIPI BARKOD İLACADI İLACFIYATI BİLGİ	İADESİPARİS BARKOD İLACADI SİPARİSBİLGİSİ İADEEDİLDİ İADETARİHİ	KİSİVERESİYE İLACTARİH ALANİNADI ALANİNSOYADI ALANİNTELEFONU BARKOD İLACADI İLACFIYATI TAHSİLEDİLDİ BİLGİ	KULLANICI KULLANICIADI KULLANICIPAROLA	KURUMSALİS İLACTARİH İLACSAAT BARKOD İLACADI İLACFIYATI KURUM ALANİNADI ALANİNSOYADI ALANİNTELEFONU BİLGİ
İHTİHASİPARİS SİPARİSNO BARKOD İLACADI SİPARİSADEDİ SİPARİSTARİHİ SİPARİSSAATI FİRMA SİPARİSVEREN BİLGİ	SİPARİSGENEL BARKOD İLACADI FİRMA SİPARİSNO SİPARİSTARİHİ	YÜKTAKSİPARİS BARKOD İLACADI FİRMA			

All the tables of the ECZANEM PROJECT

CONCLUSION

The ECZANEM is pharmacy automation. I prepared this software by using Microsoft Visual Basic 6.0 Professional Edition and Microsoft Access XP. Both of them are powerful software and they help me very much.

Visual Basic 6.0 is very easy language. It has many tools to help programmer. Newly editions on the market but I preferred version 6.0 because it is enough to made a big program.

Access is very easy to create tables I create tables without much effort with Access. It has some limitations but it is enough to make program like ECZANEM.

SQL helped me very much. In one line of program code I solved many problems with it.

ECZANEM is for pharmacy shops which is small but has to manage lot of data. It is easy to use and user friendly.

APPENDIX

THE CODES OF THE MY PROJECT

THE CODES OF THE ANA FORM

```
Public db As Database
Public tb As Recordset
Public tb2 As Recordset
Public tb3 As Recordset
```

```
Private Sub ajan_Click()
Load ajanda
ajanda.Show
End Sub
```

```
Private Sub ara_Click()
Load Form19
Form19.Show
End Sub
```

```
Private Sub bar_Click()
barkodoku.Show
End Sub
```

```
Private Sub cik_Click()
End
End Sub
```

```
Private Sub Command2_Click()
Form19.Show
'DENEMELER.Show
End Sub
```

```
Private Sub hak_Click()
Load Form7
Form7.Show
End Sub
```

```
Private Sub hes_Click(Index As Integer)
Shell ("c:\windows\system32\calc.exe")
End Sub
```

```
Private Sub kul_Click()
Unload Form8
Unload Form9
Unload Form14
Unload Form4
Unload Form5
Unload Form6
```



```

Unload Form10
Unload Form12
Unload Form13
Unload Form15
Unload Form3
Unload Form16
Unload siparis_ayar
Unload Form19
Load Form2
Form2.Show
End Sub

```

```

Private Sub MDIForm_Load()
If Dir("firma.txt") <> "" Then
Open "firma.txt" For Input As #1
While Not EOF(1)
Input #1, firma
Wend
Close #1
End If
If Dir("kullanici.txt") <> "" Then
Open "kullanici.txt" For Input As #2
While Not EOF(2)
Input #2, kullanici
Wend
Close #2
End If
ana.Label2.Caption = kullanici
End Sub

```

```

Private Sub muh_Click()
gelirgiderform.Show
End Sub

```

```

Private Sub ozel_Click()
varsayilan.Show
End Sub

```

```

Private Sub sat_Click()
Unload Form8
Load Form9
Form9.WindowState = 2
Form9.Show
Unload Form14
Unload Form4
Unload Form5
Unload Form6
Unload Form10
Unload Form12

```

```

Unload Form13
Unload Form15
Unload Form3
Unload Form16
Unload siparis_ayar
Unload Form19

```

```
End Sub
```

```
Private Sub sip_Click()
```

```

Unload Form9
Load Form8
Form8.Show
Unload Form3
Unload Form14
Unload Form4
Unload Form5
Unload Form6
Unload Form10
Unload Form12
Unload Form13
Unload Form15
Unload Form16
Unload siparis_ayar
Unload Form19
End Sub

```

```
Private Sub stok_Click()
```

```

ilacdegistir.Show
End Sub

```

```
Private Sub Text1_Change()
```

```

If Len(Text1.Text) = 0 Then
    Unload AcilanForm
End If

```

```
If Len(Text1.Text) >= 1 Then
```

```

    AcilanForm.Show , ana
    AcilanForm.Left = ana.Text1.Left
    AcilanForm.Top = ana.Text1.Height + ana.Text1.Top + 1100
End If

```

```
IlacBul (Text1.Text)
```

```
End Sub
```

```
Private Sub Timer1_Timer()
```

```

Label8.Caption = Date
Label9.Caption = Time

```

```
End Sub
```

```

Private Sub Toolbar1_ButtonClick(ByVal Button As MSComctlLib.Button)
If Button.Index = 1 Then
    MsgBox "a"
End If
If Button.Index = 2 Then
    MsgBox "b"
End If
End Sub

```

```

Private Sub yeni_Click()
Unload Form3
Unload Form4
Unload Form5
Unload Form6
Unload Form7
Unload Form8
Unload Form9
Unload siparis_ayar
Unload ajanda
Unload Form10
Unload Form12
Unload Form13
Unload Form14
Unload Form15
Unload Form16
Unload Form17
Unload Form18
Unload Form19
kullanicihesaplari.Show
End Sub

```

THE CODE OF THE ACILAN FORM

```

Dim ind As Integer

```

```

Private Sub ListView1_ItemClick(ByVal Item As MSComctlLib.ListItem)
ind = Item.Index
End Sub

```

```

Private Sub ListView1_MouseDown(Button As Integer, Shift As Integer, x As Single, y As Single)
If Button = 2 Then
    PopupMenu mnuMenu, , x, y
End If
End Sub

```

```

Private Sub mnullacAdı_Click()
Dim a As Clipboard
Set a = Clipboard

```



```

a.SetText (ListView1.ListItems(ind).SubItems(1))
ana.Text1.Text = ""
End Sub

```

```

Private Sub mnuKopyala_Click()
On Error GoTo hata
Dim a As Clipboard
Set a = Clipboard
a.SetText ListView1.ListItems(ind).Text
ana.Text1.Text = ""
hata:
If Err.Number > 0 Then
MsgBox "PLEASE SELECT MEDICINE"
End If
End Sub

```

THE CODE OF THE BARKODEOKU FORM

```

Public toplam As Double
Public ilacsayisi As Double
Dim Uyari_KritikSeviye As Boolean

```

```

Private Sub Combarlaekle_Click()
'On Error GoTo hata
Control = 0
sayac = 1
If Text1lekle.Text <> "" Then

```

```

Set db = OpenDatabase(App.Path + "\db1.mdb")
Set tb = db.OpenRecordset("FIYAT")
Set tb2 = db.OpenRecordset("SIPARISGENEL")

```

```

y = LTrim(Text1lekle.Text)
SQL = "SELECT * FROM FIYAT"
Set tb = db.OpenRecordset(SQL)

```

```

'While Not tb.EOF
' If y = Str(tb.Fields("BARKOD")) Then
'dusur = tb.Fields("STOK")
'GoTo devam
'End If
'tb.MoveNext
'Wend
'devam:
tb.MoveLast
tb.MoveFirst

```

```

While Not tb.EOF

```

```

If y = LTrim(tb.Fields("BARKOD")) Then

```

```

Control = 1
dusur = tb.Fields("STOK")
For i = 1 To Form9.ListView1.ListItems.Count
If (Form9.ListView1.ListItems(i).Text) = y Then
sayac = sayac + 1
End If
Next i

If sayac > dusur Then
a = MsgBox("THERE IS NO IN STOCK PLEASE ORDER THIS MEDICINE",
vbInformation)
End If

If dusur > 0 Then
dusur = dusur - 1
End If
If dusur <= 0 Then
a = MsgBox("THERE IS NO IN STOCK PLEASE ORDER THIS MEDICINE",
vbInformation)
End If
If dusur > 0 Then
Set x = Form9.ListView1.ListItems.Add(, LTrim(tb.Fields("barkod")))
x.SubItems(1) = tb.Fields("ILACADI")
x.SubItems(2) = tb.Fields("ILACFIYATI")

'List1.AddItem (tb.Fields("ILACADI") + " FIYATI=" + Str(tb.Fields("ILACFIYATI")))
toplam = toplam + tb.Fields("ILACFIYATI")
Form9.Label12.Caption = Str(toplam)
ilacsayisi = Form9.ListView1.ListItems.Count
Form9.Label1.Caption = Str(ilacsayisi) + " MEDICINE(S)"

'tb.Edit
'tb.Fields("STOK") = dusur
'tb.Update
If dusur < tb.Fields("KRITIKDUZEY") Then
' Şipariş Veriliyor
If Uyari_KritikSeviye = False Then
MsgBox tb.Fields("ILACADI") + " PRICE=" + Str(tb.Fields("ILACFIYATI")) &
Chr(13) & Chr(10) & "MEDICINE STOCK IS UNDER CRITICAL LEVEL", 48, "Kritik
Seviye"
End If

Uyari_KritikSeviye = True
'tb2.AddNew
'tb2.Fields("BARKOD") = Str(tb.Fields("BARKOD"))
'tb2.Fields("ILACADI") = tb.Fields("ILACADI")
'tb2.Update
End If

'Else

```

```

'a = MsgBox("STOKTA BU ILACTAN KALMADI SIPARIS EDINIZ!", vbInformation)
End If

End If
tb.MoveNext

Wend
If Control = 1 Then
' MsgBox ("tamamdir")
Else
MsgBox ("bulunamadı!")
End If

End If
hata:
If Err.Number = 3022 Then
Err.Number = 0
End If
End Sub

Private Sub Command1_Click()
Unload Me
End Sub

Private Sub Command3_Click()
Unload Me
End Sub

Private Sub Form_Load()
Uyari_KritikSeviye = False
End Sub

```

THE CODE OF THE FORM10 (ORDER NOW)

```

Public tus As Integer
Public indeks As Integer

Private Sub Command1_Click()
Set tb2 = db.OpenRecordset("SIPARISGENEL")
Dim tablo3 As Recordset
Set tablo3 = db.OpenRecordset("VERILENSIPARIS")
If tablo3.RecordCount > 0 Then
tablo3.MoveLast
siparisnosu = tablo3.Fields("SIPARISNO")
Else
siparisnosu = 0
End If
siparisnosu = siparisnosu + 1
For i = 1 To ListViewSiparisVer.ListItems.Count

```



```

tablo3.AddNew
tablo3.Fields("SIPARISNO") = siparisosu
tablo3.Fields("BARKOD") = ListViewSiparisVer.ListItems(i).Text
tablo3.Fields("ILACADI") = ListViewSiparisVer.ListItems(i).SubItems(1)
tablo3.Fields("SIPARISADEDI") = ListViewSiparisVer.ListItems(i).SubItems(6)
tablo3.Fields("SIPARISTARIHI") = Date
tablo3.Fields("SIPARISSAATI") = Time
tablo3.Fields("FIRMA") = ListViewSiparisVer.ListItems(i).SubItems(4)
'tablo.Fields ("SIPARISVEREN")=
'tablo.Fields( "BILGI")=
tablo3.Update
Next i

i = 1
sonlistviewsiparisver = ListViewSiparisVer.ListItems.Count
While i <= sonlistviewsiparisver
j = 1
sonlistview1 = Form8.ListView1.ListItems.Count
While j <= sonlistview1
If (ListViewSiparisVer.ListItems(i).Text) = (Form8.ListView1.ListItems(j).Text) Then
Form8.ListView1.ListItems(j).Selected = True
Form8.ListView1.ListItems.Remove Form8.ListView1.SelectedItem.Index
sonlistview1 = sonlistview1 - 1
End If
j = j + 1
Wend
i = i + 1
Wend
'guncelle
If tb2.RecordCount > 0 Then
tb2.MoveLast
tb2.MoveFirst
End If

While Not tb2.EOF
tb2.Delete
tb2.MoveNext
Wend

For j = 1 To Form8.ListView1.ListItems.Count
tb2.AddNew
tb2.Fields("BARKOD") = Form8.ListView1.ListItems(j).Text
tb2.Fields("ILACADI") = Form8.ListView1.ListItems(j).SubItems(1)
tb2.Update
Next j

For j = 1 To ListViewSiparisVer.ListItems.Count
tb2.AddNew
tb2.Fields("BARKOD") = ListViewSiparisVer.ListItems(j).Text

```

```
tb2.Fields("ILACADI") = ListViewSiparisVer.ListItems(j).SubItems(1)
tb2.Update
Next j
```

```
ListViewSiparisVer.ListItems.Clear
```

```
End Sub
```

```
Private Sub Command2_Click()
Unload Me
End Sub
```

```
Private Sub Command3_Click()
'On Error GoTo hata
zatenvar = 0
If Text1.Text <> "" Then
Set db = OpenDatabase(App.Path + "\db1.mdb")
Set tb = db.OpenRecordset("FIYAT")
Dim x As ListItem
y = Text1.Text
For i = 1 To ListViewSiparisVer.ListItems.Count
If LTrim(ListViewSiparisVer.ListItems(i).Text) = LTrim(y) Then
b = MsgBox("THIS MEDICINE IS ALREADY IN LIST", vbExclamation)
zatenvar = 1
End If
Next i
```

```
If zatenvar <> 1 Then
tb.MoveLast
tb.MoveFirst
While Not tb.EOF
If y = (tb.Fields("BARKOD")) Then
Control = 1
Set x = ListViewSiparisVer.ListItems.Add(, , tb.Fields("BARKOD"))
x.SubItems(1) = tb.Fields("ILACADI")
x.SubItems(2) = tb.Fields("STOK")
x.SubItems(3) = tb.Fields("KRITIKDUZEY")
x.SubItems(4) = tb.Fields("ILACFIYATI")
x.SubItems(5) = firma
x.SubItems(6) = 1
```

```
End If
tb.MoveNext
Wend
If Control <> 1 Then
MsgBox ("MEDICINE NOT FOUND")
Text1.SetFocus
End If
```

```
End If
End If
```

```
hata:
```

```
If Err.Number = 13 Then
a = MsgBox("PLEASE ENTER BARCODE, NOT USE CHARACTERS", vbExclamation)
Text1.SetFocus
End If
```

```
End Sub
```

```
Private Sub Form_Load()
```

```
For i = 1 To Form8.ListView1.ListItems.Count
Set x = ListViewSiparisVer.ListItems.Add(, Form8.ListView1.ListItems(i).Text)
x.SubItems(1) = Form8.ListView1.ListItems(i).SubItems(1)
x.SubItems(2) = Form8.ListView1.ListItems(i).SubItems(2)
x.SubItems(3) = Form8.ListView1.ListItems(i).SubItems(3)
x.SubItems(4) = Form8.ListView1.ListItems(i).SubItems(4)
x.SubItems(5) = Form8.ListView1.ListItems(i).SubItems(5)
x.SubItems(6) = 1
```

```
Next i
```

```
End Sub
```

```
Private Sub ListViewSiparisVer_ItemClick(ByVal Item As MSComctlLib.ListItem)
indeksi = Item.Index
If tus = 1 Then
kactane = InputBox("HOW MANY ORDERS", "ENTER HOW MANY MEDICINE DO YOU WANT", 1)
If kactane = "" Then
kactane = 1
End If
Item.SubItems(6) = kactane
Else
PopupMenu mnumenu2
End If
```

```
End Sub
```

```
Private Sub ListViewSiparisVer_MouseDown(Button As Integer, Shift As Integer, x As Single, y As Single)
If Button = 1 Then
```



```
tus = 1
Else
tus = 2
End If
```

```
End Sub
```

```
Private Sub mnubulistedencikar_Click()
ListViewSiparisVer.ListItems.Remove ListViewSiparisVer.SelectedItem.Index
End Sub
```

```
Private Sub mnuyoktakisipariseaktar_Click()
indeksi = ListViewSiparisVer.SelectedItem.Index
Control = 1
Dim tablo5 As Recordset
Set tablo5 = db.OpenRecordset("YOKTAKISIPARIS")
While Not tablo5.EOF
If (LTrim(ListViewSiparisVer.ListItems(indeksi).Text)) =
(LTrim(tablo5.Fields("BARKOD"))) Then
Control = 0
End If
tablo5.MoveNext
Wend
If Control = 1 Then
tablo5.AddNew
tablo5.Fields("BARKOD") = ListViewSiparisVer.ListItems(indeksi).Text
tablo5.Fields("ILACADI") = Form8.ListView1.ListItems(indeksi).SubItems(1)
tablo5.Fields("FIRMA") = firma
tablo5.Update
a = MsgBox("ADDED TO ABSENT MEDICINE LIST", vbInformation)
Else
a = MsgBox("IT IS ALREADY IN ABSENT LIST", vbExclamation)
End If
```

```
End Sub
```

THE CODE OF THE FORM11 (ABSENT ORDERS)

```
Public indekslesil As Integer
```

```
Private Sub Command1_Click()
Unload Me
End Sub
```

```
Private Sub Command2_Click()
Unload Me
End Sub
```

```

Private Sub Command3_Click()
If indekslesil <> 0 Then
Dim tablo4 As Recordset
Set tablo4 = db.OpenRecordset("YOKTAKISIPARIS")
deger1 = "BARKOD=" + ListView1.ListItems(indekslesil).Text + ""
deger2 = " AND ILACADI=" + ListView1.ListItems(indekslesil).SubItems(1) + ""
deger = deger1 + deger2
SQL = "SELECT * FROM YOKTAKISIPARIS WHERE " + deger + " ;"
Set tablo4 = db.OpenRecordset(SQL)
If tablo4.RecordCount > 0 Then
tablo4.Delete
ListView1.ListItems(indekslesil).Selected = True
ListView1.ListItems.Remove ListView1.SelectedItem.Index
End If
indekslesil = 0
End If
End Sub

Private Sub Form_Load()
Dim tablo As Recordset
Dim x As ListItem
Set tablo = db.OpenRecordset("YOKTAKISIPARIS")
If Not tablo.RecordCount = 0 Then
While Not tablo.EOF
SQL = "SELECT * FROM FIYAT WHERE BARKOD=" & tablo.Fields("BARKOD") & ""
and ILACADI=" & tablo.Fields("ILACADI") & "" ;"
Set tb = db.OpenRecordset(SQL)
Set x = ListView1.ListItems.Add(, , tablo.Fields("BARKOD"))
x.SubItems(1) = tablo.Fields("ILACADI")
x.SubItems(2) = tb.Fields("STOK")
x.SubItems(3) = tb.Fields("KRITIKDUZEY")
x.SubItems(4) = tb.Fields("ILACFIYATI")
x.SubItems(5) = tablo.Fields("FIRMA")
tablo.MoveNext
tb.MoveNext
Wend
End If
End Sub

Private Sub ListView1_ItemClick(ByVal Item As MSComctlLib.ListItem)
indekslesil = Item.Index
End Sub

```

THE CODE OF THE FORM 14 (ORDERS WHICH IS GIVEN)

```

Public indekslesil As Integer

Private Sub Command1_Click()
Unload Me

```

End Sub

```
Private Sub Command2_Click()
If indekslesil <> 0 Then
Dim tablo4 As Recordset
Dim sayi As Double
Set tablo4 = db.OpenRecordset("VERILENSIPARIS")
sayi = Val(ListView1.ListItems(indekslesil).Text)
deger2 = " AND ILACADI = " + ListView1.ListItems(indekslesil).SubItems(1) + ""
SQL = "SELECT * FROM VERILENSIPARIS WHERE SIPARISNO= " & sayi & " " +
deger2 + " ;"
Set tablo4 = db.OpenRecordset(SQL)
tablo4.Delete
ListView1.ListItems(indekslesil).Selected = True
ListView1.ListItems.Remove ListView1.SelectedItem.Index
indekslesil = 0
End If
End Sub
```

```
Private Sub Form_Load()
Dim tablo4 As Recordset
Dim z As ListItem
Set tablo4 = db.OpenRecordset("VERILENSIPARIS")
If tablo4.RecordCount > 0 Then
While Not tablo4.EOF
Set z = ListView1.ListItems.Add(, , tablo4.Fields("SIPARISNO"))
z.SubItems(1) = tablo4.Fields("ILACADI")
z.SubItems(2) = "SELÇUK ECZA"
z.SubItems(3) = tablo4.Fields("SIPARISTARIHI") + tablo4.Fields("SIPARISSAATI")
tablo4.MoveNext
Wend
Else
c = MsgBox("THERE IS NO ORDER WHICH IS GIVEN", vbInformation)
End If
End Sub
```

```
Private Sub ListView1_ItemClick(ByVal Item As MSComctlLib.ListItem)
indekslesil = Item.Index
End Sub
```

THE CODE OF THE FORM 15 (INCOMING ORDERS)

```
Public tus As Integer
Public indeks As Integer
```

```
Private Sub Command1_Click()
i = 1
j = 1
x = 1
sonlistview1 = ListView1.ListItems.Count
```



```

sonlistview2 = ListView2.ListItems.Count
cik = sonlistview1
While i <= sonlistview1 And x < cik
While j <= sonlistview2
While i <= sonlistview1
If ListView2.ListItems(j).Checked Then
If (ListView1.ListItems(i).Text = ListView2.ListItems(j).Text) And
(ListView1.ListItems(i).SubItems(1) = ListView2.ListItems(j).SubItems(2)) Then
ListView1.ListItems(i).Selected = True
ListView1.ListItems.Remove ListView1.SelectedItem.Index
sonlistview1 = sonlistview1 - 1
End If
End If
i = i + 1
Wend
j = j + 1
Wend
i = 1
x = x + 1
Wend

```

```

Dim tablo4 As Recordset

```

```

Set tablo4 = db.OpenRecordset("VERILENSIPARIS")
Set tb = db.OpenRecordset("FIYAT")

```

```

sonu = ListView2.ListItems.Count
'MsgBox (sonu)
If tablo4.RecordCount > 0 Then
j = 1
While (j <= sonu) And (tablo4.RecordCount > 0)
'While Not tablo4.EOF
'j = 1
'sonu = ListView2.ListItems.Count
' While j <= sonu
tablo4.MoveFirst
While Not tablo4.EOF
If ListView2.ListItems(j).Checked Then
If (tablo4.Fields("SIPARISNO") = (ListView2.ListItems(j).Text)) And
((tablo4.Fields("ILACADI") = (ListView2.ListItems(j).SubItems(2)))) Then
'MsgBox (tablo4.Fields("ILACADI"))
tb.MoveFirst
While Not tb.EOF
If LTrim((tb.Fields("BARKOD"))) =
LTrim(ListView2.ListItems(j).SubItems(1)) Then
tb.Edit
tb.Fields("STOK") = tb.Fields("STOK") +
(ListView2.ListItems(j).SubItems(3))
tb.Update

```

```

        MsgBox (tb.Fields("ILACADI") + " STOCK INFORMATION IS
UPDATED")
        End If
        tb.MoveNext
    Wend
    tablo4.Delete
    End If
End If
j = j + 1
'tablo4.Delete
'If tablo4.RecordCount > 0 Then
tablo4.MoveNext
'End If
Wend
'tablo4.Delete
'If tablo4.RecordCount > 0 Then
'tablo4.MoveNext
'End If
j = j + 1
Wend

Else
    c = MsgBox("THERE IS NO ORDER WHICH IS GIVEN", vbInformation)
End If

i = 1
sonlistview1 = ListView1.ListItems.Count
For j = 1 To ListView2.ListItems.Count
    While i <= sonlistview1
        If ListView2.ListItems(j).Checked Then
            If ListView1.ListItems(i) = ListView2.ListItems(j) Then
                ListView1.ListItems(i).Selected = True
                ListView1.ListItems.Remove ListView1.SelectedItem.Index
                sonlistview1 = sonview1 - 1
            End If
        End If
        i = i + 1
    Wend
Next j

x = 1
sonux = ListView2.ListItems.Count
sonlistview2 = ListView2.ListItems.Count
While x <= sonlistview2
    j = 1
    While j <= sonlistview2
        If ListView2.ListItems(j).Checked Then
            ListView2.ListItems(j).Selected = True
            ListView2.ListItems.Remove ListView2.SelectedItem.Index
            sonlistview2 = sonview2 - 1
        End If
        j = j + 1
    Wend
Next j

```

```

End If
j = j + 1
Wend
x = x + 1
Wend
For i = 1 To 10
stokkontrollu ("a")
Next i
Unload Me
End Sub

Private Sub Command2_Click()
Unload Me
End Sub

Private Sub Form_Load()
Dim tablo4 As Recordset
Dim z As ListItem

Set tablo4 = db.OpenRecordset("VERILENSIPARIS")
If tablo4.RecordCount > 0 Then
    While Not tablo4.EOF
        Set z = ListView1.ListItems.Add(, , tablo4.Fields("SIPARISNO"))
        z.SubItems(1) = tablo4.Fields("ILACADI")
        z.SubItems(2) = "SELÇUK ECZA"
        z.SubItems(3) = tablo4.Fields("SIPARISTARIHI") + tablo4.Fields("SIPARISSAATI")
        tablo4.MoveNext
    Wend
Else
c = MsgBox("THERE IS NO ORDER WHICH IS GIVEN", vbInformation)
End If
End Sub

Private Sub ListView1_ItemClick(ByVal Item As MSComctlLib.ListItem)
ListView2.ListItems.Clear
numara = Item
Dim tablo5 As Recordset
Dim k As ListItem
Set tablo5 = db.OpenRecordset("VERILENSIPARIS")
If tablo5.RecordCount > 0 Then
    While Not tablo5.EOF
        kontrol = (Str(tablo5.Fields("SIPARISNO")))
        If Str(numara) = kontrol Then
            Set k = ListView2.ListItems.Add(, , tablo5.Fields("SIPARISNO"))
            k.SubItems(1) = tablo5.Fields("BARKOD")
            k.SubItems(2) = tablo5.Fields("ILACADI")
            k.SubItems(3) = tablo5.Fields("SIPARISADEDI")
            k.SubItems(4) = tablo5.Fields("SIPARISTARIHI")
            k.SubItems(5) = tablo5.Fields("SIPARISSAATI")
        End If
    Wend
End If

```



```

tablo5.MoveNext
Wend
End If
End Sub

```

```

Private Sub ListView2_ItemClick(ByVal Item As MSComctlLib.ListItem)
indeksi = Item.Index
If tus = 2 Then
PopupMenu menu3
End If
End Sub

```

```

Private Sub ListView2_MouseDown(Button As Integer, Shift As Integer, x As Single, y As Single)
If Button = 1 Then
tus = 1
Else
tus = 2
End If
End Sub

```

```

Private Sub mnuiade_Click()
indeksi = ListView2.SelectedItem.Index
Control = 1
Dim tablo6 As Recordset
Dim tablo7 As Recordset
Set tablo6 = db.OpenRecordset("IADESIPARIS")

```

```

tablo6.AddNew
tablo6.Fields("BARKOD") = ListView2.ListItems(indeksi).SubItems(1)
tablo6.Fields("ILACADI") = ListView2.ListItems(indeksi).SubItems(2)

```

```

bilgisi = (ListView2.ListItems(indeksi).SubItems(4)) + " " +
(ListView2.ListItems(indeksi).SubItems(5))
If bilgisi <> "" Then
tablo6.Fields("SIPARISBILGISI") = bilgisi
End If
tablo6.Update
a = MsgBox("ADDED TO RETURN ORDERS LIST", vbInformation)
End Sub

```

```

Private Sub mnuyokta_Click()
indeksi = ListView2.SelectedItem.Index
Control = 1
Dim tablo5 As Recordset
Set tablo5 = db.OpenRecordset("YOKTAKISIPARIS")
While Not tablo5.EOF
If (LTrim(ListView2.ListItems(indeksi).SubItems(1))) = (LTrim(tablo5.Fields("BARKOD")))
Then
Control = 0

```

```

End If
tablo5.MoveNext
Wend
If Control = 1 Then
tablo5.AddNew
tablo5.Fields("BARKOD") = ListView2.ListItems(indeksi).SubItems(1)
tablo5.Fields("ILACADI") = ListView2.ListItems(indeksi).SubItems(2)
tablo5.Fields("FIRMA") = firma
tablo5.Update
a = MsgBox("ADDED TO ABSENT ORDERS LIST", vbInformation)
Else
a = MsgBox("IT IS ALREADY IN ABSENT ORDERS LIST", vbExclamation)
End If
End Sub

```

THE CODE OF THE FORM 16 (RETURN ORDERS)

```

Public indeks As Integer
Public tus As Integer

```

```

Private Sub Command1_Click()
Unload Me
End Sub

```

```

Private Sub Command2_Click()
If (indeks <> 0) Then
If (ListView1.ListItems(indeks).SubItems(3) <> "YES") Then
ListView1.ListItems(indeks).SubItems(3) = "YES"
ListView1.ListItems(indeks).SubItems(4) = Str(Date) + " " + Str(Time)
Set tb3 = db.OpenRecordset("IADESIPARIS")
If Not tb3.RecordCount = 0 Then
While Not tb3.EOF
tb3.Delete
tb3.MoveNext
Wend
End If

```

```

For i = 1 To ListView1.ListItems.Count
tb3.AddNew
tb3.Fields("BARKOD") = ListView1.ListItems(i).Text
tb3.Fields("ILACADI") = ListView1.ListItems(i).SubItems(1)
tb3.Fields("SIPARISBILGISI") = ListView1.ListItems(i).SubItems(2)
If Not ListView1.ListItems(i).SubItems(3) = "" Then
tb3.Fields("IADEEDILDI") = ListView1.ListItems(i).SubItems(3)
End If
If Not ListView1.ListItems(i).SubItems(4) = "" Then
tb3.Fields("IADETARIHI") = ListView1.ListItems(i).SubItems(4)
End If
tb3.Update

```

```

Next i
Else
a = MsgBox("THIS MEDICINE IS ALREADY RETURNED", vbExclamation)
End If
End If

```

```

End Sub

```

```

Private Sub Command3_Click()
If indeks <> 0 Then
Set tb3 = db.OpenRecordset("IADESIPARIS")
If Not tb3.RecordCount = 0 Then
While Not tb3.EOF
tb3.Delete
tb3.MoveNext
Wend
End If
ListView1.ListItems(indeks).Selected = True
ListView1.ListItems.Remove ListView1.SelectedItem.Index
For i = 1 To ListView1.ListItems.Count
tb3.AddNew
tb3.Fields("BARKOD") = ListView1.ListItems(i).Text
tb3.Fields("ILACADI") = ListView1.ListItems(i).SubItems(1)
tb3.Fields("SIPARISBILGISI") = ListView1.ListItems(i).SubItems(2)
If Not ListView1.ListItems(i).SubItems(3) = "" Then
tb3.Fields("IADEEDILDI") = ListView1.ListItems(i).SubItems(3)
End If
If Not ListView1.ListItems(i).SubItems(4) = "" Then
tb3.Fields("IADETARIHI") = ListView1.ListItems(i).SubItems(4)
End If
tb3.Update
Next i
End If
End Sub

```

```

Private Sub Form_Load()
Dim x As ListItem

Set tb2 = db.OpenRecordset("IADESIPARIS")
While Not tb2.EOF
Set x = ListView1.ListItems.Add(, , tb2.Fields("BARKOD"))
x.SubItems(1) = tb2.Fields("ILACADI")
x.SubItems(2) = tb2.Fields("SIPARISBILGISI")
If tb2.Fields("IADEEDILDI") <> "" Then
x.SubItems(3) = tb2.Fields("IADEEDILDI")
End If
If tb2.Fields("IADETARIHI") <> "" Then
x.SubItems(4) = tb2.Fields("IADETARIHI")
End If
tb2.MoveNext

```



```
Wend
End Sub
```

```
Private Sub ListView1_ItemClick(ByVal Item As MSComctlLib.ListItem)
indeksi = Item.Index
End Sub
```

THE CODE OF THE FORM 19 (SEARCH)

```
Private Sub Command1ARA_Click()
Set db = OpenDatabase(App.Path + "\db1.mdb")
Set tb = db.OpenRecordset("FIYAT")
x = Text1ARA.Text
x = UCase(x)
SQL = "SELECT * FROM FIYAT"
Set tb = db.OpenRecordset(SQL)
List1.Clear
While Not tb.EOF
If Combo1.ListIndex = 0 Then
y = tb.Fields("ILACADI")
If InStr(y, x) Then
Control = 1
List1.AddItem ((tb.Fields("BARKOD")) + " " + tb.Fields("ILACADI") + " FIYATI=" +
Str(tb.Fields("ILACFIYATI")) + " STOK=" + Str(tb.Fields("STOK")) + " KRITIKDUZEY="
+ Str(tb.Fields("KRITIKDUZEY"))))
End If
Else
z = tb.Fields("BARKOD")
If InStr(z, x) Then
List1.AddItem ((tb.Fields("BARKOD")) + " " + tb.Fields("ILACADI") + " FIYATI=" +
Str(tb.Fields("ILACFIYATI")) + " STOK=" + Str(tb.Fields("STOK")) + " KRITIKDUZEY="
+ Str(tb.Fields("KRITIKDUZEY"))))
Control = 1
End If
End If
tb.MoveNext
Wend
If Control = 1 Then
Label2.Caption = Str(List1.ListCount) + " find!"
Else
MsgBox ("NOT FOUND")
Label2.Caption = "not found"
End If
End Sub

Private Sub Form_Load()
Combo1.AddItem "MEDICINE NAME"
Combo1.AddItem "MEDICINE BARCODE"
Combo1.ListIndex = 0
```

End Sub

```
Private Sub List1_Click()  
Textbarkodal = List1.Text  
End Sub
```

THE CODE OF THE FORM 2 (USER ENTERANCE)

```
Private Sub Command1_Click()  
Control = 0  
yap = 0  
Set db = OpenDatabase(App.Path + "\db1.mdb")  
Set tb4 = db.OpenRecordset("KULLANICI")  
If (Text1.Text <> "") And (Text2.Text <> "") Then  
yap = 1  
If tb4.RecordCount > 0 Then  
yap = 1  
While Not tb4.EOF  
If (tb4.Fields("KULLANICIADI") = UCase(Text1.Text)) And  
tb4.Fields("KULLANICIPAROLA") = Text2.Text Then  
MsgBox("ACCESS CONFIRMED", vbInformation)  
Control = 1  
kullanici = UCase(Text1.Text)  
na.Label2.Caption = kullanici  
Unload Me  
Exit Sub  
End If  
tb4.MoveNext  
Wend  
End If  
Else  
MsgBox("PLEASE FILL ALL BLANKS", vbExclamation)  
End If  
If UCase(Text1.Text) = "ARDA" And Text2.Text = "1234" Then  
Unload Me  
na.Label2.Caption = kullanici  
Control = 1  
Exit Sub  
Else  
MsgBox("WRONG USER NAME OR PASSWORD", vbExclamation)  
Text1.SetFocus  
End If  
If Control = 0 And yap = 1 Then  
MsgBox("WRONG USER NAME", vbExclamation)  
Text1.SetFocus  
End If  
End Sub
```

```
Private Sub Command2_Click()
Unload Me
End Sub
```

THE CODE OF THE FORM 3 (VERESIYE)

```
Public veresiyeadi As String
Public veresiyesoyadi As String
Public veresiyetelefon As String
```

```
Private Sub Command1_Click()
If Text1.Text <> "" Then
veresiyeadi = UCase(Text1.Text)
veresiyesoyadi = UCase(Text2.Text)
veresiyetelefon = UCase(Text3.Text)
Unload Me
Else
a = MsgBox("PLEASE ENTER NAME INFORMATION", vbExclamation)
Text1.SetFocus
End If
End Sub
```

```
Private Sub Command3_Click()
veresiyeadi = ""
veresiyesoyadi = ""
veresiyetelefon = ""
Unload Me
End Sub
```

```
Private Sub Form_Load()
Label6.Caption = Date
Label8.Caption = Time
Label7.Caption = Str(barkodoku.toplam) + " YTL"
Combo1.Clear
For i = 1 To Form9.ListView1.ListItems.Count
Combo1.AddItem Form9.ListView1.ListItems(i).SubItems(1)
Next i
Combo1.ListIndex = 0
End Sub
```

THE CODE OF THE FORM 4 (SOCIETY)

```
Public kurumalaninadi As String
Public kurumalaninsoyadi As String
Public kurumalanintelefonu As String
Public kurumkurum As String
```

```
Private Sub Command1_Click()
```



```

kurumalaninadi = UCase(Text1.Text)
kurumalaninsoyadi = UCase(Text2.Text)
kurumalanintelefonu = UCase(Text3.Text)
For i = 0 To 2
If Not Option1(i).Value = 0 Then
    kurumkurum = Option1(i).Tag
End If
Next i
Unload Me
End Sub

Private Sub Command3_Click()
Unload Me
End Sub

Private Sub Form_Load()
Label6.Caption = Date
Label2.Caption = Time
Label7.Caption = Str(barkodoku.toplam) + " YTL"
End Sub

```

THE CODE OF THE FORM 5 (ON CREDIT)

```

Public indeksinial As Integer
Private Sub Command1_Click()
j = indeksinial
If ListView1.ListItems(indeksinial).SubItems(7) = "YES" Then
a = MsgBox("ITS ALREADY PAID", vbExclamation)
End If

If (indeksinial <> 0) And (Not ListView1.ListItems(indeksinial).SubItems(7) = "YES") Then
If Not Option1.Value = 0 Then
Set tb3 = db.OpenRecordset("GELIRGIDER")
tb3.AddNew
tb3.Fields("SATISYAPAN") = kullanici
tb3.Fields("SATISTARIH") = Date
tb3.Fields("SATISSAAT") = Time
tb3.Fields("SATISTIPI") = "CASH"
tb3.Fields("BARKOD") = ListView1.ListItems(j).SubItems(3)
tb3.Fields("ILACADI") = ListView1.ListItems(j).SubItems(4)
tb3.Fields("ILACFIYATI") = ListView1.ListItems(j).SubItems(5)
tb3.Fields("BILGI") = "ONCREDITCASH"
tb3.Update
ListView1.ListItems(j).SubItems(7) = "YES"
ListView1.ListItems(j).SubItems(8) = "CASH"
End If

If Not Option2.Value = 0 Then
Set tb3 = db.OpenRecordset("GELIRGIDER")
tb3.AddNew

```

```

tb3.Fields("SATISYAPAN") = kullanici
tb3.Fields("SATISTARIH") = Date
tb3.Fields("SATISSAAT") = Time
tb3.Fields("SATISTIPI") = "CREDIT"
tb3.Fields("BARKOD") = ListView1.ListItems(j).SubItems(3)
tb3.Fields("ILACADI") = ListView1.ListItems(j).SubItems(4)
tb3.Fields("ILACFIYATI") = ListView1.ListItems(j).SubItems(5)
tb3.Fields("BILGI") = "ONCREDITCREDIT"
tb3.Update
ListView1.ListItems(j).SubItems(7) = "YES"
ListView1.ListItems(j).SubItems(8) = "CREDIT"
End If

```

```

End If
End Sub

```

```

Private Sub Command2_Click()
Unload Me
End Sub

```

```

Private Sub Form_Load()
indeksinal = 0
Set db = OpenDatabase(App.Path + "\db1.mdb")
Set tb3 = db.OpenRecordset("KISIVERESIYE")
Dim x As ListItem
While Not tb3.EOF
'List1.AddItem Str(tb3.Fields("ILACTARIH")) + " " + tb3.Fields("ALANINADI") + " " +
Str(tb3.Fields("ILACFIYATI"))
Set x = ListView1.ListItems.Add(, tb3.Fields("ALANINADI"))
x.SubItems(1) = tb3.Fields("ALANINSOYADI")
x.SubItems(2) = tb3.Fields("ALANINTELEFONU")
x.SubItems(3) = tb3.Fields("BARKOD")
x.SubItems(4) = tb3.Fields("ILACADI")
x.SubItems(5) = tb3.Fields("ILACFIYATI")
x.SubItems(6) = tb3.Fields("ILACTARIH")
x.SubItems(7) = tb3.Fields("TAHSILEDILDI")
If tb3.Fields("BILGI") <> "" Then
x.SubItems(8) = tb3.Fields("BILGI")
End If
tb3.MoveNext
Wend
End Sub

```

```

Private Sub Form_Unload(Cancel As Integer)
Set db = OpenDatabase(App.Path + "\db1.mdb")
Set tb3 = db.OpenRecordset("KISIVERESIYE")
While Not tb3.EOF
tb3.Delete
tb3.MoveNext
Wend

```

```

For j = 1 To ListView1.ListItems.Count
tb3.AddNew
tb3.Fields("ALANINADI") = ListView1.ListItems(j).Text
tb3.Fields("ALANINSOYADI") = ListView1.ListItems(j).SubItems(1)
tb3.Fields("ALANINTELEFONU") = ListView1.ListItems(j).SubItems(2)
tb3.Fields("BARKOD") = ListView1.ListItems(j).SubItems(3)
tb3.Fields("ILACADI") = ListView1.ListItems(j).SubItems(4)
tb3.Fields("ILACFIYATI") = ListView1.ListItems(j).SubItems(5)
tb3.Fields("ILACTARIH") = ListView1.ListItems(j).SubItems(6)
tb3.Fields("TAHSILEDILDI") = ListView1.ListItems(j).SubItems(7)
tb3.Fields("BILGI") = ListView1.ListItems(j).SubItems(8)
tb3.Update
Next j
End Sub

Private Sub ListView1_ItemClick(ByVal Item As MSCComctlLib.ListItem)
indeksinial = Item.Index
End Sub

```

THE CODE OF THE FORM 6 (SOCIETY IN DETAILS)

```

Public indeksolist As Integer

Private Sub Command1_Click()
ListView1.ListItems(indeksolist).SubItems(9) = "SOCIETYPAY"
End Sub

Private Sub Command2_Click()
Unload Me
End Sub

Private Sub Command3_Click()
Unload Me
End Sub

Private Sub Command4_Click()
ListView1.ListItems(indeksolist).SubItems(9) = "SOCIETYNOTPAY"
End Sub

Private Sub Command5_Click()
ListView1.ListItems(indeksolist).SubItems(9) = " "
End Sub

Private Sub Form_Load()
Dim tablo5 As Recordset
Set db = OpenDatabase(App.Path + "\db1.mdb")
Set tablo5 = db.OpenRecordset("KURUMSATIS")
Dim x As ListItem
If tablo5.RecordCount > 0 Then
While Not tablo5.EOF

```



```

Set x = ListView1.ListItems.Add(, , tablo5.Fields("ALANINADI"))
x.SubItems(1) = tablo5.Fields("ALANINSOYADI")
x.SubItems(2) = tablo5.Fields("ALANINTELEFONU")
x.SubItems(3) = tablo5.Fields("ILACTARIH")
x.SubItems(4) = tablo5.Fields("ILACSAAT")
x.SubItems(5) = tablo5.Fields("BARKOD")
x.SubItems(6) = tablo5.Fields("ILACADI")
x.SubItems(7) = tablo5.Fields("ILACFIYATI")
x.SubItems(8) = tablo5.Fields("KURUM")
If tablo5.Fields("BILGI") <> "" Then
x.SubItems(9) = tablo5.Fields("BILGI")
End If
tablo5.MoveNext
Wend
Else
a = MsgBox("NO RECORD!", vbExclamation)
End If
End Sub

```

```

Private Sub Form_Unload(Cancel As Integer)
Set db = OpenDatabase(App.Path + "\db1.mdb")
Set tb4 = db.OpenRecordset("KURUMSATIS")
While Not tb4.EOF
tb4.Delete
tb4.MoveNext
Wend
For i = 1 To ListView1.ListItems.Count
tb4.AddNew
tb4.Fields("ILACTARIH") = ListView1.ListItems(i).SubItems(3)
tb4.Fields("ILACSAAT") = ListView1.ListItems(i).SubItems(4)
tb4.Fields("BARKOD") = ListView1.ListItems(i).SubItems(5)
tb4.Fields("ILACADI") = ListView1.ListItems(i).SubItems(6)
tb4.Fields("ILACFIYATI") = ListView1.ListItems(i).SubItems(7)
tb4.Fields("KURUM") = ListView1.ListItems(i).SubItems(8)
tb4.Fields("ALANINADI") = ListView1.ListItems(i).Text
tb4.Fields("ALANINSOYADI") = ListView1.ListItems(i).SubItems(1)
tb4.Fields("ALANINTELEFONU") = ListView1.ListItems(i).SubItems(2)
If Not (ListView1.ListItems(i).SubItems(9)) = "" Then
tb4.Fields("BILGI") = ListView1.ListItems(i).SubItems(9)
End If
tb4.Update
Next i
End Sub

```

```

Private Sub ListView1_ItemClick(ByVal Item As MSComctlLib.ListItem)
indeksoflist = Item.Index
End Sub

```

THE CODE OF THE FORM 7 (ABOUT)

```
Private Sub Form_Click()  
Unload Me  
End Sub
```

THE CODE OF THE FORM 8 (ORDERS)

```
Private Sub Command1_Click()  
Unload Me  
End Sub
```

```
Private Sub Command10_Click()  
Load Form14  
Form14.Show  
Unload Form3  
Unload Form4  
Unload Form5  
Unload Form6  
Unload Form10  
Unload Form12  
Unload Form13  
Unload Form15  
Unload siparis_ayar  
Unload Form16  
End Sub
```

```
Private Sub Command11_Click()  
Load Form15  
Form15.Show  
Unload Form3  
Unload Form4  
Unload Form5  
Unload Form6  
Unload Form10  
Unload Form12  
Unload Form14  
Unload Form13  
Unload siparis_ayar  
Unload Form16  
End Sub
```

```
Private Sub Command12_Click()  
Load Form13  
Form13.Show  
Unload Form3  
Unload Form4  
Unload Form5  
Unload Form6  
Unload Form10  
Unload Form12  
Unload Form14
```

```
Unload Form15
Unload siparis_ayar
Unload Form16
End Sub
```

```
Private Sub Command13_Click()
Load Form16
Form16.Show
Unload Form3
Unload Form4
Unload Form5
Unload Form6
Unload Form14
Unload Form12
Unload Form13
Unload Form15
Unload Form10
Unload siparis_ayar
End Sub
```

```
Private Sub Command9_Click()
Load Form10
Form10.Show
Unload Form3
Unload Form4
Unload Form5
Unload Form6
Unload Form14
Unload Form12
Unload Form13
Unload Form15
Unload Form16
Unload siparis_ayar
End Sub
```

```
Private Sub Form_Load()
'On Error GoTo hata
```

```
Set db = OpenDatabase(App.Path + "\db1.mdb")
Set tb = db.OpenRecordset("FIYAT")
Set tb2 = db.OpenRecordset("SIPARISGENEL")
Dim x As ListItem
```

```
List2.Clear
```

```
If tb.RecordCount > 0 Then
tb.MoveLast
tb.MoveFirst
End If
```



```

If tb2.RecordCount > 0 Then
tb2.MoveLast
tb2.MoveFirst
End If
'MsgBox tb2.RecordCount

```

```

For i = 0 To tb2.RecordCount - 1

```

```

SQL = "SELECT * FROM FIYAT WHERE BARKOD='" & LTrim(tb2.Fields("BARKOD"))
& "' and ILACADI='" & tb2.Fields("ILACADI") & "' ;"

```

```

Set tb = db.OpenRecordset(SQL)

```

```

List2.AddItem "Barkod: " & tb2.Fields("BARKOD") & " İlac Adı: " &
tb2.Fields("ILACADI") & " Stokda: " & tb.Fields("STOK") & " Kiritik Düzey: " &
tb.Fields("KRITIKDUZEY")

```

```

tb2.MoveNext
tb.MoveNext

```

```

Next i
stokkontrollu ("a")
'stok kontrolü
If tb.RecordCount > 0 Then
tb.MoveLast
tb.MoveFirst
End If
While Not tb.EOF
If tb.Fields("STOK") < tb.Fields("KRITIKDUZEY") Then
Set x = ListView1.ListItems.Add(, , tb.Fields("BARKOD"))
x.SubItems(1) = tb.Fields("ILACADI")
x.SubItems(2) = tb.Fields("STOK")
x.SubItems(3) = tb.Fields("KRITIKDUZEY")
x.SubItems(4) = tb.Fields("ILACFIYATI")
x.SubItems(5) = firma
End If
tb.MoveNext
Wend
'stok kontrolü sonu

```

```

'stokgu kritik duzeyden buyuk olanlarin cikarilmasi
If tb2.RecordCount > 0 Then
tb2.MoveLast
tb2.MoveFirst
End If

```

```

For i = 0 To tb2.RecordCount - 1

```

```
SQL = "SELECT * FROM FIYAT WHERE BARKOD=" & tb2.Fields("BARKOD") & ""  
and ILACADI=" & tb2.Fields("ILACADI") & "" ;"
```

```
Set tb = db.OpenRecordset(SQL)
```

```
If (tb.Fields("STOK")) >= (tb.Fields("KRITIKDUZEY")) Then
```

```
'MsgBox ("VAR")
```

```
tb2.Delete
```

```
End If
```

```
tb2.MoveNext
```

```
tb.MoveNext
```

```
Next i
```

```
'stogu kritik duzeyden buyuk olanlarin cikarilmasi sonu
```

```
'veri tabanindakilerin listeye alınmasi
```

```
If tb.RecordCount > 0 Then
```

```
tb.MoveLast
```

```
tb.MoveFirst
```

```
End If
```

```
If tb2.RecordCount > 0 Then
```

```
tb2.MoveLast
```

```
tb2.MoveFirst
```

```
End If
```

```
If tb2.RecordCount > 0 Then
```

```
For i = 0 To tb2.RecordCount - 1
```

```
SQL = "SELECT * FROM FIYAT WHERE BARKOD=" & tb2.Fields("BARKOD") & ""  
and ILACADI=" & tb2.Fields("ILACADI") & "" ;"
```

```
Set tb = db.OpenRecordset(SQL)
```

```
'List2.AddItem "Barkod: " & tb2.Fields("BARKOD") & " İlac Adı: " &
```

```
tb2.Fields("ILACADI") & " Stokda: " & tb.Fields("STOK") & " Kiritik Düzey: " &
```

```
tb.Fields("KRITIKDUZEY")
```

```
Set x = ListView1.ListItems.Add(, , tb2.Fields("BARKOD"))
```

```
x.SubItems(1) = tb2.Fields("ILACADI")
```

```
x.SubItems(2) = tb.Fields("STOK")
```

```
x.SubItems(3) = tb.Fields("KRITIKDUZEY")
```

```
x.SubItems(4) = tb.Fields("ILACFIYATI")
```

```
x.SubItems(5) = firma
```

```
tb2.MoveNext
```

```
tb.MoveNext
```

```
Next i
```

```
End If
```

```
'veri tabanindakilerin listeye alınmasi sonu
```

```
'listede aynı olanları çıkar
```

```
sonsayi = ListView1.ListItems.Count
```

```
h = sonsayi
```

```
c = 1
```

```
i = 1
```

```
While (i <= sonsayi) And (c <= h)
```

```
    i = 1
```

```

While i <= sonsayi
j = 1
While (j <= sonsayi) And (i <= sonsayi)
If i <> j Then
    esit = (LTrim(ListView1.ListItems(i).Text))
    esitmi = (LTrim(ListView1.ListItems(j).Text))
    If esit = esitmi Then
        ListView1.ListItems(j).Selected = True
        ListView1.ListItems.Remove ListView1.SelectedItem.Index
        sonsayi = sonsayi - 1
        'MsgBox (esit + " cikarildi")
    End If
End If
j = j + 1
Wend
i = i + 1
Wend
c = c + 1
Wend
'listede ayni olanlari cikarmanın sonu

'veritabanini sil
If tb2.RecordCount > 0 Then
tb2.MoveLast
tb2.MoveFirst
End If

While Not tb2.EOF
tb2.Delete
tb2.MoveNext
Wend
'veritabani silmenin sonu

'listenin son halini veri tabanına kaydet
For i = 1 To ListView1.ListItems.Count
tb2.AddNew
tb2.Fields("BARKOD") = ListView1.ListItems(i).Text
tb2.Fields("ILACADI") = ListView1.ListItems(i).SubItems(1)
tb2.Update
Next i
'listenin son halini veri tabanına kaydetmenin sonu

hata:
If Err.Number = 3021 Then
    MsgBox "NO RECORD", 48, "Exit  "
End If
End Sub

Private Sub List2_DblClick()
Load siparis_ayar

```



```

siparis_ayar.Show
Unload Form3
Unload Form4
Unload Form5
Unload Form6
Unload Form10
Unload Form12
Unload Form13
Unload Form14
Unload Form15
Unload Form16
End Sub

```

THE CODE OF THE FORM 9 (SALES)

```

Public kasa As Double
Public ciro As Double
Public toplam2 As Double

```

```

Private Sub Check1_Click()
    toplam = 0
    If Check1.Value = 1 Then
        For i = 1 To ListView1.ListItems.Count
            toplam = toplam + ListView1.ListItems(i).SubItems(2)
        Next i
        barkodoku.toplam = -toplam
        Label12.Caption = barkodoku.toplam
    Else
        For i = 1 To ListView1.ListItems.Count
            toplam = toplam + ListView1.ListItems(i).SubItems(2)
        Next i
        barkodoku.toplam = toplam
        Label12.Caption = barkodoku.toplam
    End If
End Sub

```

```

Private Sub Combarkodoku_Click()
    barkodoku.Show
End Sub

```

```

Private Sub Command1_Click()
    Form6.Show
End Sub

```

```

Private Sub Command3_Click()
    Unload Form3
    Unload Form4
    Unload Form5
    Unload Form6
    Unload Form10

```

```
Unload Form12
Unload Form13
Unload Form15
Unload siparis_ayar
Unload Form16
```

```
If ListView1.ListItems.Count > 0 Then
toplam2 = 0
'barkodoku.toplam = barkodoku.toplam - (ListView1.SelectedItem.SubItems(2))
'Label12.Caption = barkodoku.toplam
ListView1.ListItems.Remove ListView1.SelectedItem.Index
barkodoku.ilacsayisi = (barkodoku.ilacsayisi) - 1
Label1.Caption = Str(barkodoku.ilacsayisi) + "MEDICINE(S)"
For i = 1 To ListView1.ListItems.Count
toplam2 = toplam2 + ListView1.ListItems(i).SubItems(2)
Next i
If Check1.Value = 1 Then
barkodoku.toplam = -toplam2
Label12.Caption = barkodoku.toplam
Else
barkodoku.toplam = toplam2
Label12.Caption = barkodoku.toplam
End If
```

```
End If
```

```
End Sub
```

```
Private Sub Command4_Click()
Unload Form3
Unload Form4
Unload Form5
Unload Form6
Unload Form10
Unload Form12
Unload Form13
Unload Form15
Unload siparis_ayar
Unload Form16
sonradaniadeform.Show
```

```
End Sub
```

```
Private Sub Command5_Click()
Load Form5
Form5.Show
Unload Form3
Unload Form4
Unload Form14
Unload Form6
```

```

Unload Form10
Unload Form12
Unload Form13
Unload Form15
Unload siparis_ayar
Unload Form16
End Sub

```

```

Private Sub Command7_Click()
'skjadjk
'On Error GoTo hata

```

```

Set db = OpenDatabase(App.Path + "\db1.mdb")
Set tb = db.OpenRecordset("FIYAT")
SQL = "SELECT * FROM FIYAT"
Set tb = db.OpenRecordset(SQL)
Set tb2 = db.OpenRecordset("SIPARISGENEL")
Set tb3 = db.OpenRecordset("GELIRGIDER")
Set tb4 = db.OpenRecordset("KISIVERESIYE")
Set tb5 = db.OpenRecordset("KURUMSATIS")

```

```

ekle = 1
cirohesapla (1)
Label18.Caption = Str(ciros) + " YTL"
kasaheapla (1)
Label17.Caption = Str(kasas) + " YTL"
tb.MoveLast
tb.MoveFirst
'tb2.MoveLast
'tb2.MoveFirst

```

```

'BARKODDAN GELEN KODUN BASI

```

```

If Not Check1.Value = 1 Then

```

```

For i = 1 To ListView1.ListItems.Count
While Not tb.EOF
If ((ListView1.ListItems(i).Text)) = ((tb.Fields("BARKOD"))) Then
'Control = 1
eklemetext = (tb.Fields("BARKOD"))

```

```

'If (tb.Fields("STOK")) > 0 Then
dusur = tb.Fields("STOK")
dusur = dusur - 1
tb.Edit
tb.Fields("STOK") = dusur
tb.Update

```

```

' Set x = Form9.ListView1.ListItems.Add(, , tb.Fields("barkod"))

```



```

'x.SubItems(1) = tb.Fields("ILACADI")
'x.SubItems(2) = tb.Fields("ILACFIYATI")
>List1.AddItem (tb.Fields("ILACADI") + " FIYATI=" + Str(tb.Fields("ILACFIYATI")))
'toplam = toplam + tb.Fields("ILACFIYATI")
'Form9.Label12.Caption = Str(toplam)

If dusur < tb.Fields("KRITIKDUZEY") Then
' Şipariş Veriliyor
' If Uyari_KritikSeviye = False Then
' MsgBox tb.Fields("ILACADI") + " FIYATI=" + Str(tb.Fields("ILACFIYATI")) &
Chr(13) & Chr(10) & "İlaç Kritik seviyenin altında!", 48, "Kritik Seviye"
'End If
' Uyari_KritikSeviye = True
'If Not tb2.RecordCount > 0 Then
'tb2.MoveLast
'tb2.MoveFirst
'End If

While Not tb2.EOF
If Not tb2.RecordCount = 0 Then
If eklemetext = (tb2.Fields("BARKOD")) Then
ekle = 0
MsgBox ("listede zaten var!")
End If
tb2.MoveNext
End If
Wend

If ekle <> 0 Then
tb2.AddNew
tb2.Fields("BARKOD") = LTrim(tb.Fields("BARKOD"))
tb2.Fields("ILACADI") = tb.Fields("ILACADI")
tb2.Update
End If

End If
' Else
'a = MsgBox("STOKTA BU ILACTAN KALMADI SIPARIS EDINIZ!", vbInformation)
End If

'End If
tb.MoveNext
'Next k
Wend
tb.MoveFirst
Next i

'BARKODDAN GELENIN SONU
Else

```

```

For i = 1 To ListView1.ListItems.Count
While Not tb.EOF
If ((ListView1.ListItems(i).Text)) = ((tb.Fields("BARKOD")))) Then
tb.Edit
tb.Fields("STOK") = tb.Fields("STOK") + 1
tb.Update
End If
tb.MoveNext
Wend
tb.MoveFirst
Next i
'Check1.Value = 0
End If

```

```

If Not OptionButton3.Value = 0 Then
'Load Form3
'Form3.Show
Unload Form14
Unload Form4
Unload Form5
Unload Form6
Unload Form10
Unload Form12
Unload Form13
Unload Form15
Unload Form16
Unload siparis_ayar

```

```

For z = 1 To ListView1.ListItems.Count
tb4.AddNew
tb4.Fields("ILACTARIH") = Date
tb4.Fields("ALANINADI") = Form3.veresiyeadı
tb4.Fields("ALANINSOYADI") = Form3.veresiyesoyadı
tb4.Fields("ALANINTELEFONU") = Form3.veresiyetelefon
tb4.Fields("BARKOD") = ListView1.ListItems(z).Text
If Not Check1.Value = 1 Then
tb4.Fields("ILACADI") = ListView1.ListItems(z).SubItems(1)
tb4.Fields("ILACFIYATI") = ListView1.ListItems(z).SubItems(2)
tb4.Fields("TAHSILEDILDI") = "NO"
'Tb2.Fields("BILGI")=
tb4.Update
Else
tb4.Fields("ILACADI") = ListView1.ListItems(z).SubItems(1)
tb4.Fields("ILACFIYATI") = -ListView1.ListItems(z).SubItems(2)
tb4.Fields("TAHSILEDILDI") = "RETURNED"
'Tb2.Fields("BILGI")=
tb4.Update
End If

```

```

'gelirgidere ekle

```

```

tb3.AddNew
tb3.Fields("SATISYAPAN") = kullanici
tb3.Fields("SATISTARIH") = Date
tb3.Fields("SATISSAAT") = Time
If Not Check1.Value = 1 Then
tb3.Fields("SATISTUPI") = "ONCREDIT"
tb3.Fields("BARKOD") = ListView1.ListItems(z).Text
tb3.Fields("ILACADI") = ListView1.ListItems(z).SubItems(1)
'tb3.Fields("ILACFIYATI") = ListView1.ListItems(z).SubItems(2)
'Tb2.Fields("BILGI")=
tb3.Update
Else
tb3.Fields("SATISTUPI") = "ONCREDITRETURN"
tb3.Fields("BARKOD") = ListView1.ListItems(z).Text
tb3.Fields("ILACADI") = ListView1.ListItems(z).SubItems(1)
'tb3.Fields("ILACFIYATI") = -(ListView1.ListItems(z).SubItems(2))
'Tb2.Fields("BILGI")=
tb3.Update
End If

Next z
ListView1.ListItems.Clear
barkodoku.toplam = 0
Label12.Caption = 0
Check1.Value = 0
OptionButton1.Value = 1
End If

If Not OptionButton1.Value = 0 Then

'kasa = kasa + barkodoku.toplam
'ciro = ciro + barkodoku.toplam
For j = 1 To ListView1.ListItems.Count
tb3.AddNew
tb3.Fields("SATISYAPAN") = kullanici
tb3.Fields("SATISTARIH") = Date
tb3.Fields("SATISSAAT") = Time
If Not Check1.Value = 1 Then
tb3.Fields("SATISTUPI") = "CASH"
tb3.Fields("BARKOD") = ListView1.ListItems(j).Text
tb3.Fields("ILACADI") = ListView1.ListItems(j).SubItems(1)
tb3.Fields("ILACFIYATI") = ListView1.ListItems(j).SubItems(2)
'Tb2.Fields("BILGI")=
tb3.Update
Else
tb3.Fields("SATISTUPI") = "CASHRETURN"
tb3.Fields("BARKOD") = ListView1.ListItems(j).Text
tb3.Fields("ILACADI") = ListView1.ListItems(j).SubItems(1)
tb3.Fields("ILACFIYATI") = -(ListView1.ListItems(j).SubItems(2))
'Tb2.Fields("BILGI")=

```



```

tb3.Update
End If
Next j
cirohesapla (1)
kasaheapla (1)
Label17.Caption = Str(kasas) + " YTL"
Label18.Caption = Str(ciros) + " YTL"
Label12.Caption = 0
barkodoku.toplam = 0
ListView1.ListItems.Clear
Check1.Value = 0
End If

```

```

If Not OptionButton5.Value = 0 Then
'ciro = ciro + barkodoku.toplam
'Label18.Caption = Str(ciro)
For j = 1 To ListView1.ListItems.Count
tb3.AddNew
tb3.Fields("SATISYAPAN") = kullanici
tb3.Fields("SATISTARIH") = Date
tb3.Fields("SATISSAAT") = Time
If Not Check1.Value = 1 Then
tb3.Fields("SATISTUPI") = "CREDIT"
tb3.Fields("BARKOD") = ListView1.ListItems(j).Text
tb3.Fields("ILACADI") = ListView1.ListItems(j).SubItems(1)
tb3.Fields("ILACFIYATI") = ListView1.ListItems(j).SubItems(2)
'Tb2.Fields("BILGI")=
tb3.Update
Else
tb3.Fields("SATISTUPI") = "CREDITRETURN"
tb3.Fields("BARKOD") = ListView1.ListItems(j).Text
tb3.Fields("ILACADI") = ListView1.ListItems(j).SubItems(1)
tb3.Fields("ILACFIYATI") = -ListView1.ListItems(j).SubItems(2)
'Tb2.Fields("BILGI")=
tb3.Update
End If

```

```

Next j
Label12.Caption = 0
barkodoku.toplam = 0
ListView1.ListItems.Clear
Check1.Value = 0
OptionButton1.Value = 1
cirohesapla (1)
Label18.Caption = Str(ciros) + " YTL"

```

```

End If

```

```

'SOCIETY satis basi
If Not OptionButton4.Value = 0 Then

```

```

For z = 1 To ListView1.ListItems.Count
tb5.AddNew
tb5.Fields("ILACTARIH") = Date
tb5.Fields("ILACSAAT") = Time
tb5.Fields("BARKOD") = ListView1.ListItems(z).Text
tb5.Fields("ILACADI") = ListView1.ListItems(z).SubItems(1)
tb5.Fields("KURUM") = Form4.kurumkurum
tb5.Fields("ALANINADI") = Form4.kurumalaninadi
tb5.Fields("ALANINSOYADI") = Form4.kurumalaninsoyadi
tb5.Fields("ALANINTELEFONU") = Form4.kurumalanintelefonu

```

```

If Not Check1.Value = 1 Then
tb5.Fields("ILACFIYATI") = ListView1.ListItems(z).SubItems(2)
'Tb5.Fields("BILGI")=
tb5.Update
Else
tb5.Fields("ILACFIYATI") = -ListView1.ListItems(z).SubItems(2)
tb5.Fields("BILGI") = "SOCIETYRETURN"
tb5.Update
End If

```

```

'gelirgidere ekle
tb3.AddNew
tb3.Fields("SATISYAPAN") = kullanici
tb3.Fields("SATISTARIH") = Date
tb3.Fields("SATISSAAT") = Time
If Not Check1.Value = 1 Then
tb3.Fields("SATISTUPI") = "SOCIETY"
tb3.Fields("BARKOD") = ListView1.ListItems(z).Text
tb3.Fields("ILACADI") = ListView1.ListItems(z).SubItems(1)
tb3.Fields("ILACFIYATI") = ListView1.ListItems(z).SubItems(2)
'Tb3.Fields("BILGI")=
tb3.Update
Else
tb3.Fields("SATISTUPI") = "SOCIETYRETURN"
tb3.Fields("BARKOD") = ListView1.ListItems(z).Text
tb3.Fields("ILACADI") = ListView1.ListItems(z).SubItems(1)
tb3.Fields("ILACFIYATI") = -(ListView1.ListItems(z).SubItems(2))
'tb3.Fields("BILGI") =
tb3.Update
End If

```

```

Next z
ListView1.ListItems.Clear
barkodoku.toplam = 0
Label12.Caption = 0
Check1.Value = 0
OptionButton1.Value = 1
cirohesapla (1)

```

```
Label18.Caption = Str(ciros) + " YTL"  
End If  
'SOCIETY satis sonu  
Label1.Caption = ""
```

```
hata:  
If Err.Number = 3021 Then  
    MsgBox "NO RECORD", 48, "Exit  "  
End If
```

```
'skjdksd
```

```
End Sub
```

```
Private Sub Command8_Click()  
barkodoku.toplam = 0  
Unload Me  
End Sub
```

```
Private Sub Form_Load()  
toplamlam = 0  
kasa = 0  
ciro = 0  
toplamlam2 = 0  
kasahehapla (1)  
Label17.Caption = Str(kasas) + " YTL"  
cirohehapla (1)  
Label18.Caption = Str(ciros) + " YTL"  
End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)  
barkodoku.toplam = 0  
End Sub
```

```
Private Sub OptionButton3_Click()  
If Not OptionButton3.Value = 0 Then  
Form3.Show  
End If  
End Sub
```

```
Private Sub OptionButton4_Click()  
Form4.Show  
End Sub
```

THE CODE OF FORM FRMSPLASH

```
Option Explicit  
Dim sayac As Integer
```



```
Private Sub Form_KeyPress(KeyAscii As Integer)
    Unload Me
End Sub
```

```
Private Sub Form_Load()
    lblVersion.Caption = "Version " & App.Major & "." & App.Minor & "." & App.Revision
    lblProductName.Caption = App.Title
End Sub
```

```
Private Sub Frame1_Click()
    Unload Me
End Sub
```

```
Private Sub Timer1_Timer()
    sayac = sayac + 1
    If sayac > 1 Then
        Load ana
        ana.Show
        Unload frmSplash
    End If
End Sub
```

THE CODE OF THE FORM GELIRGIDER

```
Private Sub Command1_Click()
    Dim tablo6 As Recordset
    Set db = OpenDatabase(App.Path + "\db1.mdb")
    Set tablo6 = db.OpenRecordset("GELIRGIDER")
    While Not tablo6.EOF
        tablo6.Delete
        tablo6.MoveNext
    Wend
    ListView1.ListItems.Clear
    a = MsgBox("ALL OF THEM IS DELETED", vbInformation)
End Sub
```

```
Private Sub Command2_Click()
    ListView1.ListItems.Clear
    Dim ciros As Double
    Dim tablo2 As Recordset
    ListView1.View = lvwReport
    Set db = OpenDatabase(App.Path + "\db1.mdb")
    Dim x As ListItem
    toplamsayis = 0
```

```
CASH = ""
```

```
For i = 0 To 6
```

```

If Check1(i).Value = 1 Then
If CASH <> "" Then CASH = CASH + " or "
CASH = CASH + "SATISTIPI = " + Check1(i).Tag + ""
End If
Next i
If CASH = "" Then
MsgBox "PLEASE SELECT TYPE OF SEARCH", 48, "Bul"
Exit Sub
End If

```

```

SQL = "SELECT * FROM GELIRGIDER WHERE " + CASH + ";"
Set tablo2 = db.OpenRecordset(SQL)
If Not tablo2.RecordCount = 0 Then
tablo2.MoveLast
tablo2.MoveFirst

```

```

While Not tablo2.EOF
Set x = ListView1.ListItems.Add(, , tablo2.Fields("SATISYAPAN"))
x.SubItems(1) = tablo2.Fields("SATISTARIH")
x.SubItems(2) = tablo2.Fields("SATISSAAT")
x.SubItems(3) = tablo2.Fields("SATISTIPI")
x.SubItems(4) = tablo2.Fields("BARKOD")
x.SubItems(5) = tablo2.Fields("ILACADI")
x.SubItems(6) = tablo2.Fields("ILACFIYATI")
If tablo2.Fields("BILGI") <> "" Then
x.SubItems(7) = tablo2.Fields("BILGI")
End If
tablo2.MoveNext
Wend
Exit Sub
Else
a = MsgBox("NO RECORD!", vbInformation)
End If

End Sub

```

```

Private Sub Command4_Click()
Unload Me
End Sub

```

```

Private Sub CommandButton1_Click()
ListView1.ListItems.Clear

```

```

Dim ciros As Double

```

```

Dim tablo2 As Recordset
ListView1.View = lvwReport
Set db = OpenDatabase(App.Path + "\db1.mdb")
Set tablo2 = db.OpenRecordset("GELIRGIDER")
If tablo2.RecordCount > 0 Then
Dim x As ListItem
toplamsayis = 0
If Not tablo2.RecordCount = 0 Then
tablo2.MoveLast
tablo2.MoveFirst
While Not tablo2.EOF
Set x = ListView1.ListItems.Add(, , tablo2.Fields("SATISYAPAN"))
x.SubItems(1) = tablo2.Fields("SATISTARIH")
x.SubItems(2) = tablo2.Fields("SATISSAAT")
x.SubItems(3) = tablo2.Fields("SATISTIP")
x.SubItems(4) = tablo2.Fields("BARKOD")
x.SubItems(5) = tablo2.Fields("ILACADI")
x.SubItems(6) = tablo2.Fields("ILACFIYATI")
If tablo2.Fields("BILGI") <> "" Then
x.SubItems(7) = tablo2.Fields("BILGI")
End If
tablo2.MoveNext
Wend
Exit Sub
End If
Else
a = MsgBox("NO RECORD!", vbInformation)
End If
End Sub

```

THE CODE OF THE FORM ILACDEGISTIR

```

Private Sub Command1_Click()
Set db = OpenDatabase(App.Path + "\db1.mdb")
Set tb = db.OpenRecordset("FIYAT")
tb.MoveLast
tb.MoveFirst

If Text1.Text <> "" And Text2.Text <> "" And Text3.Text <> "" Then
While Not tb.EOF
If (tb.Fields("BARKOD")) = Label2.Caption Then
Image1.Visible = True
Image2.Visible = True
tb.Edit
tb.Fields("ILACFIYATI") = Text1.Text
tb.Fields("STOK") = Text2.Text
tb.Fields("KRITIKDUZEY") = Text3.Text
tb.Update
a = MsgBox("MEDICINE UPDATETED", vbInformation)

```



```

Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Control = 1
End If
tb.MoveNext
Wend
If Control <> 1 Then
a = MsgBox("MEDICINE NOT FOUND", vbExclamation)
End If
Else
a = MsgBox("PLEASE FILL ALL BLANKS", vbExclamation)
End If

End Sub

```

```

Private Sub Command2_Click()
Unload Me
End Sub

```

```

Private Sub Command3_Click()
If Text4.Text <> "" Then
y = Text4.Text
Set db = OpenDatabase(App.Path + "\db1.mdb")
Set tb = db.OpenRecordset("FIYAT")
tb.MoveLast
tb.MoveFirst

```

```

While Not tb.EOF
If y = (tb.Fields("BARKOD")) Then
Label2.Caption = tb.Fields("BARKOD")
Image1.Visible = True
Label3.Caption = tb.Fields("ILACADI")
Image2.Visible = True
Text1.Text = tb.Fields("ILACFIYATI")
Text2.Text = tb.Fields("STOK")
Text3.Text = tb.Fields("KRITIKDUZEY")
Control = 1
End If
tb.MoveNext
Wend
If Control <> 1 Then
a = MsgBox("MEDICINE NOT FOUND", vbExclamation)
End If
End If

End Sub

```

```
Private Sub Command4_Click()
Unload Me
End Sub
```

```
Private Sub Command5_Click()
Set db = OpenDatabase(App.Path + "\db1.mdb")
Set tb = db.OpenRecordset("FIYAT")
tb.MoveLast
tb.MoveFirst
ekle = 1
If Text5.Text <> "" And Text6.Text <> "" And Text7.Text <> "" And Text8.Text <> "" And
Text9.Text <> "" Then
While Not tb.EOF
If tb.Fields("BARKOD") = Text5.Text Then
a = MsgBox("THIS BARCODE IS ALREADY USED", vbExclamation)
ekle = 0
End If
tb.MoveNext
Wend
If ekle = 1 Then
tb.AddNew
tb.Fields("BARKOD") = Text5.Text
tb.Fields("ILACADI") = Text6.Text
tb.Fields("ILACFIYATI") = Text7.Text
tb.Fields("STOK") = Text8.Text
tb.Fields("KRITIKDUZEY") = Text9.Text
tb.Update
a = MsgBox("MEDICINE ADDED", vbInformation)
Text5.Text = ""
Text6.Text = ""
Text7.Text = ""
Text8.Text = ""
Text9.Text = ""
End If
Else
a = MsgBox("PLEASE FILL ALL BLANKS", vbExclamation)
End If
End Sub
```

```
Private Sub Command7_Click()
If Text10.Text <> "" Then
y = Text10.Text
Set db = OpenDatabase(App.Path + "\db1.mdb")
Set tb = db.OpenRecordset("FIYAT")
tb.MoveLast
tb.MoveFirst
Image3.Visible = False
Image4.Visible = False
While Not tb.EOF
```

```

If y = (tb.Fields("BARKOD")) Then
Label23.Caption = tb.Fields("BARKOD")
Label24.Caption = tb.Fields("ILACADI")
Label25.Caption = tb.Fields("ILACFIYATI")
Label26.Caption = tb.Fields("STOK")
Label27.Caption = tb.Fields("KRITIKDUZEY")
If tb.Fields("STOK") < tb.Fields("KRITIKDUZEY") Then
Image3.Visible = True
Image4.Visible = True
Command8.Enabled = False
a = MsgBox("YOU CAN'T DELETE THIS MEDICINE", vbExclamation)
End If
Control = 1
End If
tb.MoveNext
Wend
If Control <> 1 Then
a = MsgBox("MEDICINE NOT FOUND", vbExclamation)
End If
End If
End Sub

Private Sub Command8_Click()
Set db = OpenDatabase(App.Path + "\db1.mdb")
Set tb = db.OpenRecordset("FIYAT")
tb.MoveLast
tb.MoveFirst

If Label23.Caption <> "" Then
While Not tb.EOF
If (tb.Fields("BARKOD")) = Label23.Caption And (tb.Fields("ILACADI") =
Label24.Caption) Then
tb.Delete
Control = 1
Label23.Caption = ""
Label24.Caption = ""
Label25.Caption = ""
Label26.Caption = ""
Label27.Caption = ""
MsgBox ("MEDICINE DELETED")
End If
tb.MoveNext
Wend
If Control <> 1 Then
a = MsgBox("MEDICINE NOT FOUND", vbExclamation)
End If
Else
a = MsgBox("PLEASE SELECT A MEDICINE", vbExclamation)
End If
End Sub

```



```
Private Sub Command9_Click()  
Unload Me  
End Sub
```

```
Private Sub Image1_Click()  
MsgBox ("YOU CAN'T CHANGE MEDICINE BARCODE")  
End Sub
```

```
Private Sub Image2_Click()  
MsgBox ("YOU CAN'T CHANGE MEDICINE NAME")  
End Sub
```

```
Private Sub Label2_Click()  
MsgBox ("YOU CAN'T CHANGE MEDICINE NAME")  
End Sub
```

```
Private Sub Label3_Click()  
MsgBox ("YOU CAN'T CHANGE MEDICINE NAME")  
End Sub
```

THE CODE OF THE KULLANICIHESAPLARI

```
Private Sub Command1_Click()  
Unload Me  
End Sub
```

```
Private Sub Command2_Click()  
Set db = OpenDatabase(App.Path + "\db1.mdb")  
Set tb4 = db.OpenRecordset("KULLANICI")  
If (Text1.Text <> "") And (Text2.Text <> "") And (Text3.Text <> "") And (Text4.Text <> "")  
Then  
yap = 1  
If (Text3.Text <> Text4.Text) Then  
a = MsgBox("YOUR NEW PASSWORDS DOESN'T MATCHES PLEASE TRY AGAIN",  
vbExclamation)  
Else  
If tb4.RecordCount > 0 Then  
yap = 1  
While Not tb4.EOF  
If (tb4.Fields("KULLANICIADI") = UCase(Text1.Text)) And  
(tb4.Fields("KULLANICIPAROLA") = Text2.Text) Then  
tb4.Edit  
tb4.Fields("KULLANICIPAROLA") = Text3.Text  
tb4.Update  
Control = 1  
a = MsgBox("PASSWORD CHANGED", vbInformation)
```

```

Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""

End If
tb4.MoveNext
Wend
End If
End If
Else
a = MsgBox("PLEASE FILL ALL BLANKS", vbExclamation)
End If
End Sub

Private Sub Command3_Click()
Unload Me
End Sub

Private Sub Command4_Click()
Set db = OpenDatabase(App.Path + "\db1.mdb")
Set tb4 = db.OpenRecordset("KULLANICI")
If tb4.RecordCount > 0 Then
While Not tb4.EOF
If tb4.Fields("KULLANICIADI") = Text5.Text Then
MsgBox ("THIS USER NAME IS ALREADY USED")
Exit Sub
End If
tb4.MoveNext
Wend
End If
If (Text5.Text <> "") And (Text6.Text <> "") And (Text7.Text <> "") Then
If Text6.Text = Text7.Text Then
tb4.AddNew
tb4.Fields("KULLANICIADI") = UCase(Text5.Text)
tb4.Fields("KULLANICIPAROLA") = Text6.Text
tb4.Update
a = MsgBox("USER NAME IS ADDED", vbInformation)
Text5.Text = ""
Text6.Text = ""
Text7.Text = ""

Else
a = MsgBox("YOUR PASSWORDS DOESN'T MATCHES PLEASE TRY AGAIN",
vbExclamation)
End If
Else
a = MsgBox("PLEASE FILL ALL BLANKS", vbExclamation)
End If

```

End Sub

```
Private Sub Command5_Click()
Control = 0
yap = 0
Set db = OpenDatabase(App.Path + "\db1.mdb")
Set tb4 = db.OpenRecordset("KULLANICI")
If (Text8.Text <> "") And (Text9.Text <> "") Then
yap = 1
If tb4.RecordCount > 0 Then
yap = 1
While Not tb4.EOF
If (tb4.Fields("KULLANICIADI") = UCase(Text8.Text)) And
(tb4.Fields("KULLANICIPAROLA") = Text9.Text) Then
a = MsgBox("USER DELETED", vbInformation)
tb4.Delete
If kullanici = UCase(Text8.Text) Then
kullanici = "ARDA"
ana.Label2.Caption = kullanici
End If
Text8.Text = ""
Text9.Text = ""
Control = 1
End If
tb4.MoveNext
Wend
End If
Else
a = MsgBox("PLEASE FILL ALL BLANKS", vbExclamation)
End If
```

```
If Control = 0 And yap = 1 Then
a = MsgBox("THIS USER NAME DOESN'T USING", vbExclamation)
Text8.SetFocus
End If
End Sub
```

```
Private Sub Command6_Click()
Unload Me
End Sub
```

```
Private Sub Command7_Click()
Unload Me
End Sub
```

THE CODE OF THE FORM VARSİYILAN

```
Private Sub Command1_Click()
If Text1.Text <> "" Then
firma = Text1.Text
```



```

MsgBox ("DEFAULT FIRM CHANGED")
If firma = "" Then
firma = "SELÇUK ECZA"
End If

```

```

Open "firma.txt" For Output As #1
Write #1, firma
Close #1
End If
End Sub

```

```

Private Sub Command2_Click()
If Text2.Text <> "" Then
kullanici = Text2.Text
MsgBox ("DEFAULT USER CHANGED")
If kullanici = "" Then
kullanici = "ARDA"
End If

```

```

Open "kullanici.txt" For Output As #2
Write #2, kullanici
Close #2
End If
End Sub

```

```

Private Sub Command4_Click()
Unload Me
End Sub

```

```

Private Sub Form_Load()
Text1.Text = firma
Text2.Text = kullanici
End Sub

```

THE CODE OF THE MODULE MODULE1

```

Global db As Database
Global db2 As Database
Global tb As Recordset
Global tb2 As Recordset
Global tb3 As Recordset
Global tb4 As Recordset
Global tb5 As Recordset
Global tablo2 As Recordset
Global ciros As Double
Global kasas As Double
Global firma As String
Global kullanici As String

```

```
Public Function IlacBul(bul As String)
```

```
Dim SQL As String
```

```
Dim dosya As Database
```

```
Dim tablo As Recordset
```

```
Set dosya = OpenDatabase(App.Path + "\db1.mdb")
```

```
Set tablo = dosya.OpenRecordset("FIYAT")
```

```
Dim x As ListItem
```

```
If tablo.RecordCount = 0 Then
```

```
    tablo.Close
```

```
    dosya.Close
```

```
    ana.Text1.SetFocus
```

```
    Exit Function
```

```
End If
```

```
SQL = "SELECT ILACADI, BARKOD, ILACFIYATI, STOK FROM FIYAT WHERE  
ILACADI Like '" & UCase(bul) & "';"
```

```
Set tablo = dosya.OpenRecordset(SQL)
```

```
If tablo.RecordCount = 0 Then
```

```
    ana.Text1.SetFocus
```

```
    Exit Function
```

```
End If
```

```
tablo.MoveLast
```

```
tablo.MoveFirst
```

```
With AcilanForm
```

```
    .ListView1.ListItems.Clear
```

```
For i = 0 To tablo.RecordCount - 1
```

```
    Set x = .ListView1.ListItems.Add(, , tablo.Fields("barkod"))
```

```
    x.SubItems(1) = tablo.Fields("ILACADI")
```

```
    x.SubItems(2) = tablo.Fields("ILACFIYATI")
```

```
    x.SubItems(3) = tablo.Fields("STOK")
```

```
    tablo.MoveNext
```

```
Next i
```

```
End With
```

```
ana.Text1.SetFocus
```

```
End Function
```

```
Public Function cirohesapla(b As Integer)
```

```
    ciros = 0
```

```
    Dim a As Double
```

```
    Set db = OpenDatabase(App.Path + "\db1.mdb")
```

```
    Set tablo2 = db.OpenRecordset("GELIRGIDER")
```

```
    While Not tablo2.EOF
```

```

If tablo2.Fields("SATISTIP") = "CASH" Or tablo2.Fields("SATISTIP") = "CREDIT" Or
tablo2.Fields("SATISTIP") = "SOCIETY" Or tablo2.Fields("SATISTIP") =
"SOCIETYIADE" Or tablo2.Fields("SATISTIP") = "CASHRETURN" Or
tablo2.Fields("SATISTIP") = "CREDITRETURN" Then
sayac = sayac + 1
a = tablo2.Fields("ILACFIYATI")
ciros = ciros + a
'tablo2.Delete
End If
tablo2.MoveNext
Wend
If (ciros > 0) And ciros < (0.000001) Then
ciros = 0
End If
If (ciros < 0) And (ciros > -0.000001) Then
ciros = 0
End If
'MsgBox (ciros)
End Function

```

```

Public Function stokkontrollu(b As String)
'stok kontrolü
Set db = OpenDatabase(App.Path + "\db1.mdb")
Set tb = db.OpenRecordset("FIYAT")
Set tb2 = db.OpenRecordset("SIPARISGENEL")
Dim x As ListItem
With Form8
If tb2.RecordCount > 0 Then
tb2.MoveLast
tb2.MoveFirst
End If

```

```

If tb.RecordCount > 0 Then
tb.MoveLast
tb.MoveFirst
End If
While Not tb.EOF
If tb.Fields("STOK") < tb.Fields("KRITIKDUZEY") Then
Set x = .ListView1.ListItems.Add(, , tb.Fields("BARKOD"))
x.SubItems(1) = tb.Fields("ILACADI")
x.SubItems(2) = tb.Fields("STOK")
x.SubItems(3) = tb.Fields("KRITIKDUZEY")
x.SubItems(4) = tb.Fields("ILACFIYATI")
x.SubItems(5) = "SELÇUK ECZA"
End If
tb.MoveNext
Wend
'stok kontrolü sonu

```

'stokgu kritik duzeyden buyuk olanlarin cikarilmasi


```

If tb2.RecordCount > 0 Then
tb2.MoveLast
tb2.MoveFirst
For i = 0 To tb2.RecordCount - 1
SQL = "SELECT * FROM FIYAT WHERE BARKOD=" & tb2.Fields("BARKOD") & ""
and ILACADI=" & tb2.Fields("ILACADI") & "" ;"
Set tb = db.OpenRecordset(SQL)
If (tb.Fields("STOK")) >= (tb.Fields("KRITIKDUZEY")) Then

```

```

tb2.Delete
End If
tb2.MoveNext
tb.MoveNext
Next i
End If
'stogu kritik duzeyden buyuk olanlarin cikarilmasi sonu

```

'veri tabanindakilerin listeye alınması

```

If tb.RecordCount > 0 Then
tb.MoveLast
tb.MoveFirst
End If
If tb2.RecordCount > 0 Then
tb2.MoveLast
tb2.MoveFirst
End If

```

```

If tb2.RecordCount > 0 Then
For i = 0 To tb2.RecordCount - 1
SQL = "SELECT * FROM FIYAT WHERE BARKOD=" & tb2.Fields("BARKOD") & ""
and ILACADI=" & tb2.Fields("ILACADI") & "" ;"
Set tb = db.OpenRecordset(SQL)
>List2.AddItem "Barkod: " & tb2.Fields("BARKOD") & " İlac Adı: " &
tb2.Fields("ILACADI") & " Stokda: " & tb.Fields("STOK") & " Kiritik Düzey: " &
tb.Fields("KRITIKDUZEY")
Set x = .ListView1.ListItems.Add(, , tb2.Fields("BARKOD"))
x.SubItems(1) = tb2.Fields("ILACADI")
x.SubItems(2) = tb.Fields("STOK")
x.SubItems(3) = tb.Fields("KRITIKDUZEY")
x.SubItems(4) = tb.Fields("ILACFIYATI")
x.SubItems(5) = "SELÇUK ECZA"
tb2.MoveNext
tb.MoveNext
Next i
End If
'veri tabanindakilerin listeye alınması sonu

```

'listede aynı olanları çıkar

```

sonsayi = .ListView1.ListItems.Count
h = sonsayi

```

```

c = 1
i = 1
While (i <= sonsayi) And (c <= h)
    i = 1
    While i <= sonsayi
        j = 1
        While (j <= sonsayi) And (i <= sonsayi)
            If i <> j Then
                esit = (LTrim(.ListView1.ListItems(i).Text))
                esitmi = (LTrim(.ListView1.ListItems(j).Text))
                If esit = esitmi Then
                    .ListView1.ListItems(j).Selected = True
                    .ListView1.ListItems.Remove .ListView1.SelectedItem.Index
                    sonsayi = sonsayi - 1
                End If
            End If
            j = j + 1
        Wend
        i = i + 1
    Wend
    i = 1
    c = c + 1
Wend
'listede ayni olanlari cikarmanın sonu

'veritabanini sil
If tb2.RecordCount > 0 Then
    tb2.MoveLast
    tb2.MoveFirst
End If

While Not tb2.EOF
    tb2.Delete
    tb2.MoveNext
Wend
'veritabani silmenin sonu

'listenin son halini veri tabanına kaydet
For i = 1 To .ListView1.ListItems.Count
    tb2.AddNew
    tb2.Fields("BARKOD") = LTrim(.ListView1.ListItems(i).Text)
    tb2.Fields("ILACADI") = .ListView1.ListItems(i).SubItems(1)
    tb2.Update
Next i
'listenin son halini veri tabanına kaydetmenin sonu

'stok kontrolü sonu
End With

End Function

```

```

Public Function kasaHesapla(b As Integer)
    kasas = 0
    Dim a As Double
    Set db = OpenDatabase(App.Path + "\db1.mdb")
    Set tablo2 = db.OpenRecordset("GELIRGIDER")
    While Not tablo2.EOF
        If tablo2.Fields("SATISTIPI") = "CASH" Or tablo2.Fields("SATISTIPI") =
"CASHRETURN" Then
            sayac = sayac + 1
            a = tablo2.Fields("ILACFIYATI")
            kasas = kasas + a
            'tablo2.Delete
        End If
        tablo2.MoveNext
    Wend
    If (kasas > 0) And kasas < (0.000001) Then
        kasas = 0
    End If
    If (kasas < 0) And (kasas > -0.000001) Then
        kasas = 0
    End If
End Function

```


REFERENCES

Books:

Visual Basic Pro 6.0 Ihsan Karagülle and Zeydin Pala

Internet:

<http://www.Wikipedia.com>

<http://www.google.com.tr>