MINISTRY FOR DEVELOPMENT OF INFORMATION TECHNOLOGIES AND COMMUNICATIONS OF THE REPUBLIC OF UZBEKISTAN

TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES

«Allowed to Defense»
Head of the Department
Abdurahmanova Yu.M.
_____
«____» _____ 2015 y.

BACHELOR'S FINAL WORK

Theme: "Creating Email Client service in order to work with Windows OS without browsers""

Graduate _____ M.E.Narmatov
Supervisor _____ B.M.Nurmetov
Reviewer _____ H.N. Zaynidinov
SVA and E
consultant _____ S.M.Abdullayeva

Tashkent – 2015

CONTENT

INTRODUCTION

Implementation and development e-government in Uzbekistan

"E-government" - a new form of organization of public authorities, providing due to the wide use of information and communication technologies qualitatively new level of efficiency and convenience of receiving organizations and citizens of public services, as well as information on the performance of government agencies.

The Decree of the President of the Republic of Uzbekistan on June 27, 2013 PP-1989 "On measures for further development of national information and communication system of the Republic of Uzbekistan" approved projects and activities for the development of e-government in the Republic of Uzbekistan for the period from 2013 to 2014, tasking authorities public authorities a number of key tasks for creating and mutual integration of information systems, automation of public authorities that improve the efficiency and quality performance of their functions[1]. The ultimate goal of the implementation of "e-government" is the creation of an entirely electronic state apparatus capable of significantly improve the ability to provide interactive services.

When "E-government" will be created, it can be achieved the significant results, such as:

- improve the accountability and transparency of government agencies,
- improve availability and individualized services,
- information and effective citizen participation in the political process,
- Free exchange of information,
- optimization of government services to the public and businesses,
- Support and empowerment of citizens self-service, as well as increased participation of all voters in the processes of governance and management of the country.

On the Single portal there were launched 30 new interactive services (now the total number of services - more than 220 services). Business entities and public today can use some of the following services:

- submit electronic statement of the schedule of inspections;
- register data on foreign trade contracts in the single electronic information system of foreign trade operations;
- submit electronic application for issuance of technical conditions for connection to gas supply networks of businesses;
- pay for certain types of services through the payment system «SMS-To`lov»;
- submit electronic application to equip payment terminals, for the purchase of domestic vehicles for the needs of businesses;
- submit electronic application for individual types of licenses and permits, and others.

Developed and launched on the Single portal for business is allowing for interactive design and presentation of tax, statistical and other reporting, payment of various fees, as well as obtaining certificates and other forms of online government services. In this context, it was instructed to take the necessary measures for the development of interactive public servants, in particular with cooperation ministries and agencies to ensure the implementation of priority of interactive services for businesses and the public:

- submit application for approval of materials for selection of land with authorized organizations, as well as the execution of issuing permits in the construction sector;
- submit application for certification produced and imported products in the national system of certification;
- submit application for a permit for temporary import (export) of goods;
- submit application for a permit for placement of goods under the customs regime of re-export, and others.

Advance in ICT(Information and Communication Technologies) plays an indispensable role on progress of every county, as President Islam Karimov stated[1]: “Once again I want to stress that today implementing modern ICT, digital

and large-format telecommunication, internet is obtaining tremendous significance not only in schools, lyceum, colleges and high institutions, but and in every family". Today, free and open source software (FOSS) is considered as reasonable way of implementing software potential for developing countries. Because, first and foremost, FOSS is cost-effective and has ever-increasing functionalities. For that reasons, FOSS gives great opportunities for our country. Following arguments will prove topicality of using FOSS in ICT sector of Uzbekistan.

In 2004, August 27, Oliy Majlis of Uzbekistan adopted resolution about joining to the Bern convention of copyright protection[2]. According to the convention each member country must maintain international commitment about organizing copyright protection of commercial software authority. Thus, it is becoming vital importance to use only licensed software for users. Although licensed software offers convenient features, it has several drawbacks related with high prices and customizing issues. In many countries, especially, developing countries one of the main problems in the growth of ICT sector is gigantic payments for licenses of commercial software which significantly damage the government budget. To understate outgoing national currency we must develop own software. But it demands sufficient development potential, namely source code and documentation resources. Evidently, such potential is not formed yet in Uzbekistan. In our case, development and introduction free and open source software would be a perfect solution for shortage of software potential.

Actuality of the topic. Nowadays it is almost impossible to imagine the world without information technologies. These days the role of personal computers and internet is irreplaceable.

For these purpose, nowadays our government pays a great attention to creating, controlling, and developing interactive services. The fastest and best way of creating interactive services is to implement them with the use of global network.

These days email service is developing quickly in our country. Modern Information technologies has helped a lot email system. To be specific, old mail system has been replaced with new, comfortable, fast email system. There are a number of services in .UZ which can support email services. For example “inbox.uz”, “mail.uz”, and some other email services. Furthermore, the role of email services is irreplaceable in the age of Electron Government which is implementing in our government system. To prove this fact, each organization has been educated with computer skills and supplied with internet by the support of president of Uzbekistan I.A.Karimov.

The aim of the Bachelor’s final work is to create a software of the email client service. The result of the thesis will be creation of the program that can work with all email servers in Windows OS.

To implement Bachelor’s final work there some tasks that are required:

1) To know more about email servers.
2) Analyzing and Understanding different types of email clients.
3) Choosing the right programming language.
4) Protection of emails.
5) Choosing appropriate protocols and advantages, disadvantage of protocols.
6) To create a comfortable interface of the program.
7) To write a manual for users.

This Bachelor’s final work includes, Introduction, 4 chapters and conclusion, reference, appendix as well.

Introduction is devoted to the topicality of the subject, its goals and functions, information about chapters as well.

The first chapter includes, analyzing and understanding various types of email clients, benefits and drawbacks of protocols, understanding encryptions.

The second chapter gives information about using appropriate programming language and its structure, two types of encryption and working with file streams.

As for the third chapter, it explains adding an account to the program, testing of the program and manual of the program.

The last chapter is Life Safety, here is information about Safety of work, Working conditions, Principles and technologies cleaner production.

CHAPTER I. ANALYZING AND UNDERSTANDING DIFFERENT TYPES OF EMAIL CLIENTS

1.1. Functionality of different types of modern Email Clients

Nowadays there a lot of email client softs however generally most of them works with only one server. We can see lots of programs like that for example: “Ms Outlook”, ”Apple Mail”, ”Foxmail”, ”The Bat”, ”Windows Live”. Nevertheless, a program which I have created namely “Tuitmailer” not only supports all kinds of servers but also it can decrypt and encrypt emails directly to a server. What is important encrypting mails is not supported in other top-rated email client applications.

Email as a technology has been around for decades, and thanks to wide spread adoption and popularity, it isn't in danger of disappearing. Check out the five most popular email clients to help you wrangle your email.

Earlier this week we asked your to share your favorite email client. We didn't restrict the voting to only stand-alone email applications or web-based email clients, but we did specify that if you voted for a web-based tool it had to have distinctly client-like features—such as Gmail's ability to fetch and sort email from other sources. The email Call for Contenders was one of the most popular we've ever had, with over 1,000 votes logged. Here are the five most popular clients used by Life hacker readers:

Outlook (Windows, $399 for Office Standard Suite)

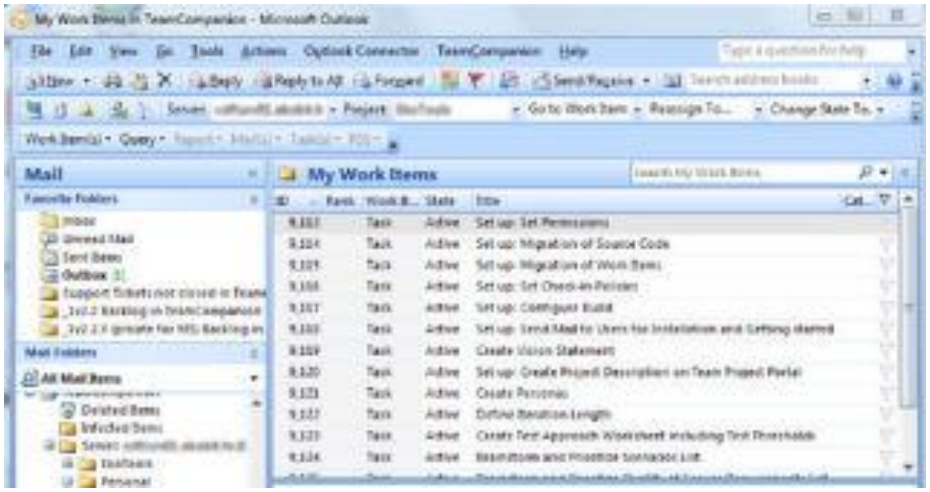


Figure 1.1. Microsoft Outlook

Outlook has been around since the 1990s, and by virtue of being part of the Microsoft Office suite and having been entrenched in the business environment, it enjoys an enormous popularity. Even if many people use Outlook because it's the email client provided—and often required!—by their place of employment, that doesn't mean it can't stand on its own merits. Integration with Windows Desktop Search gives you the ability to quickly search through your entire Outlook workflow, and Outlook can handle everything from your email to your calendar and easily transfer tasks, contacts, and more between the two.

Apple Mail (Mac, Free)

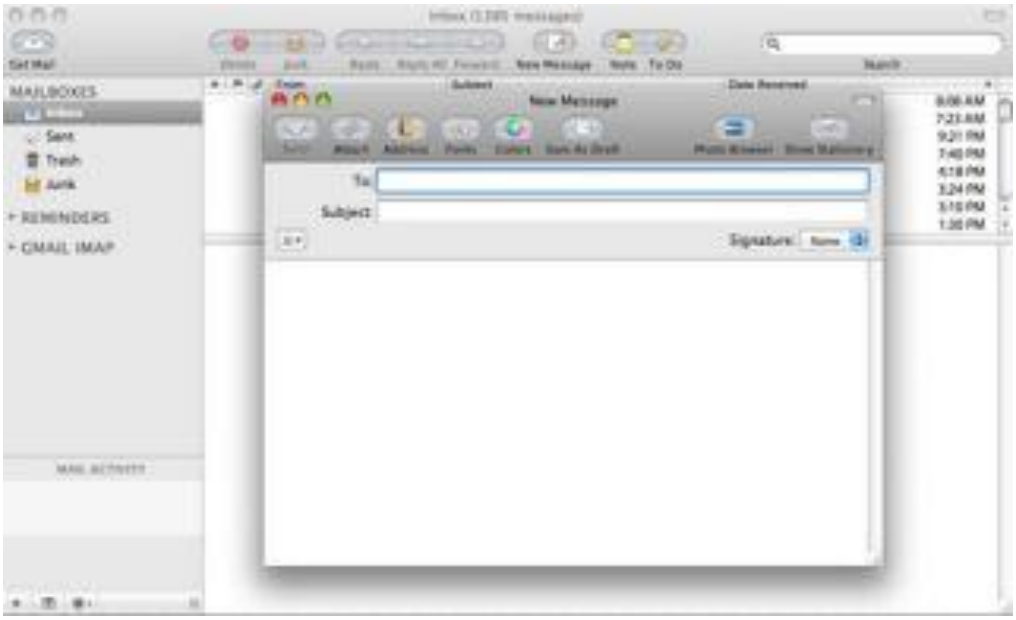


Figure 1.2. Apple Mail

Apple's Mail application, also known as Mail.app or simply Mail, unsurprisingly continues the tradition of Apple applications following the "it just works" method of design. Mail allows you to collect all your email from across the web and various email servers in one place, and it actively engages your email as you read it. For example, if you get an email with an invitation to a meeting next Thursday, Mail will detect it and make it simple to kick that appointment right over to iCal. Like the integration between Windows Desktop Search and Outlook, Mail is integrated with Spotlight to make deep massaging your messages easy.

Thunderbird (Windows/Mac/Linux, Free)

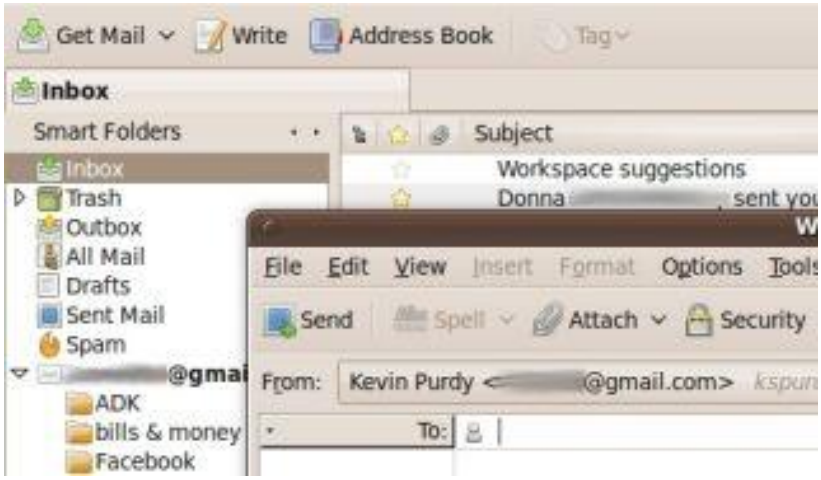


Figure 1.3. Thunderbird.

Thunderbird is an open-source offering from Mozilla—the company behind the beloved open-source browser Firefox. Thunderbird is a solid email application that sports the same extensibility of its code-sibling Firefox. Many readers voted for not just Thunderbird but Thunderbird with the addition of Lightning, a Thunderbird extension that adds scheduling and task management functionality to Thunderbird.

Gmail (Web-based, Free)

Figure 1.4. Gmail

Google has had quite a hit on their hands with Gmail, their extremely popular web-based email client. Not only do you get a feature-packed email account when you sign up for Gmail—you also get an email client that's is very adept at pulling in email from other services and organizing it with a robust system of filters and tags. You can check out how to manage multiple inboxes here if you'd like to use Gmail as a central hub for managing all your email. Many of the features in Gmail aren't necessarily revolutionary—like the ability to filter messages, flag, or label them—but the features are implemented in such a way that makes them effortless to use. And, surprising as it may seem, its much-loved threaded conversations are still relatively unique to Gmail.

Postbox (Windows/Mac, $39.95)

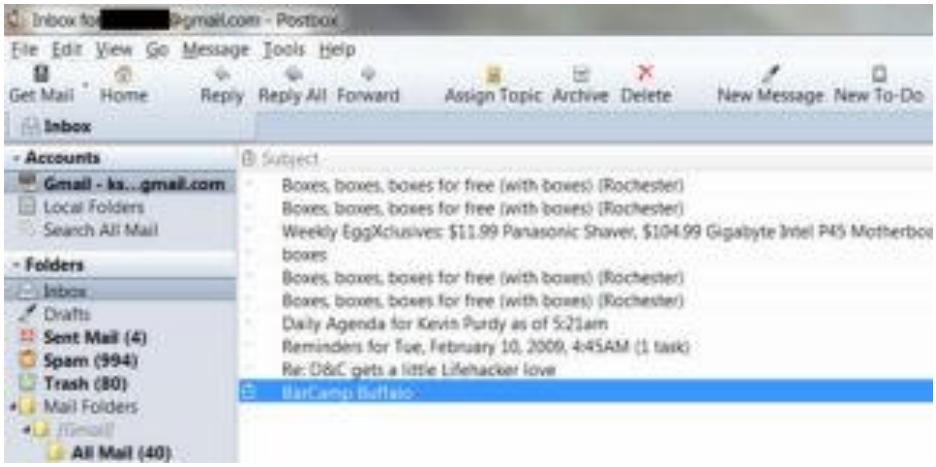


Figure 1.5. Postbox

Postbox is stand-alone email client for Windows and Mac operating systems. Postbox is based on Mozilla-code, so the Postbox team has been able to tweak quite a few Thunderbird extensions, including Lightning, to work with Postbox. In addition to its extensibility, Postbox's default interface is powerful. You can look up an email address, search for a previous attachment, and check an old email for information all in the sidebar while working on your current email. Postbox also provides email summaries as you read through and search your email, showing you not just the sender and subject line but the attachments and any important information inside the email like addresses, appointments, and URLs.

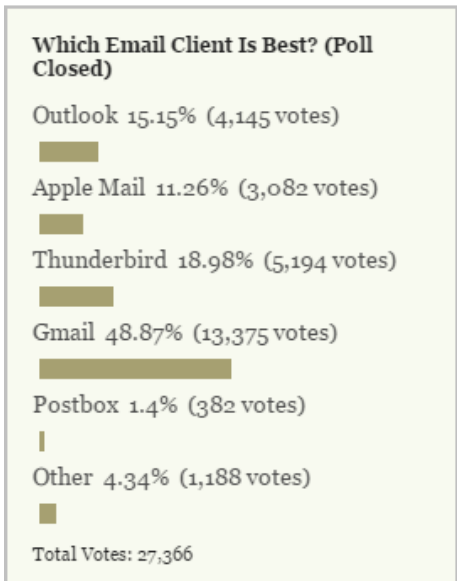


Figure 1.6. Ranking of email client applications

1.2. Choosing the right programming language

These days there are a variety of programming languages and their capability is developing significantly. However, I have chosen C# 5.0 to do my final project since my Bachelor's final work should be for Windows OS.

It is a clear fact that there is a framework called .NET by Microsoft that give a great opportunities to developers

The structure of C# and .NET (DOT NET)

C# is a simple, modern, general-purpose, object-oriented programming language developed by Microsoft within its.

C# is a modern, general-purpose, object-oriented programming language developed by Microsoft and approved by European Computer Manufacturers Association (ECMA) and International Standards Organization (ISO).

C# is designed for Common Language Infrastructure (CLI), which consists of the executable code and runtime environment that allows use of various high-level languages on different computer platforms and architectures.

The following reasons make C# a widely used professional language:

- It is a modern, general-purpose programming language
- It is object oriented.
- It is component oriented.
- It is easy to learn.
- It is a structured language.
- It produces efficient programs.
- It can be compiled on a variety of computer platforms.
- It is a part of .Net Framework.

Strong Programming Features of C#

Although C# constructs closely follow traditional high-level languages, C and C++ and being an object-oriented programming language. It has strong resemblance with Java, it has numerous strong programming features that make it endearing to a number of programmers worldwide.

Following is the list of few important features of C#:

- Boolean Conditions
- Automatic Garbage Collection
- Standard Library
- Assembly Versioning
- Properties and Events
- Delegates and Events Management
- Easy-to-use Generics
- Indexers
- Conditional Compilation
- Simple Multithreading
- LINQ and Lambda Expressions
- Integration with Windows

In this chapter, we will discuss the tools required for creating C# programming. We have already mentioned that C# is part of .Net framework and is used for writing .Net applications. Therefore, before discussing the available tools for running a C# program, let us understand how C# relates to the .Net framework.

The .Net Framework

The .Net framework is a revolutionary platform that helps you to write the following types of applications:

- Windows applications
- Web applications
- Web services

The .Net framework applications are multi-platform applications. The framework has been designed in such a way that it can be used from any of the following languages: C#, C++, Visual Basic, Jscript, COBOL, etc. All these languages can access the framework as well as communicate with each other[4].

The .Net framework consists of an enormous library of codes used by the client languages such as C#. Following are some of the components of the .Net framework:

- Common Language Runtime (CLR)

- The .Net Framework Class Library
- Common Language Specification
- Common Type System
- Metadata and Assemblies
- Windows Forms
- ASP.Net and ASP.Net AJAX
- ADO.Net
- Windows Workflow Foundation (WF)
- Windows Presentation Foundation
- Windows Communication Foundation (WCF)
- LINQ

Operator Precedence in C#

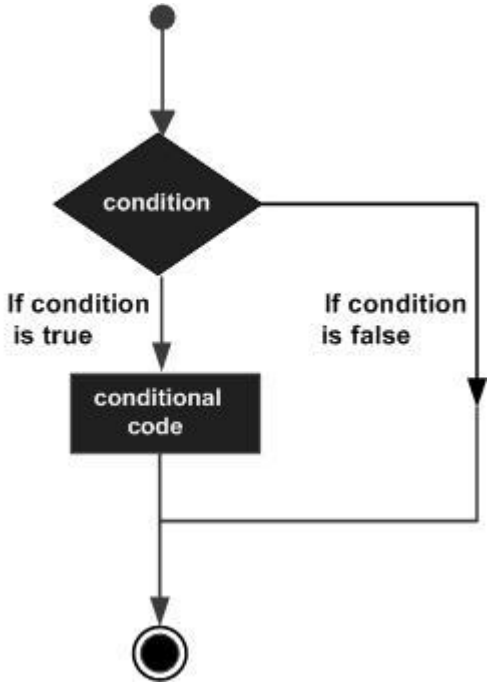
Operator precedence determines the grouping of terms in an expression. This affects evaluation of an expression. Certain operators have higher precedence than others; for example, the multiplication operator has higher precedence than the addition operator.

For example x = 7 + 3 * 2; here, x is assigned 13, not 20 because operator * has higher precedence than +, so the first evaluation takes place for 3*2 and then 7 is added into it.

Here, operators with the highest precedence appear at the top of the table, those with the lowest appear at the bottom. Within an expression, higher precedence operators are evaluated first[5].

Decision making structures requires the programmer to specify one or more conditions to be evaluated or tested by the program, along with a statement or statements to be executed if the condition is determined to be true, and optionally, other statements to be executed if the condition is determined to be false.

Following is the general from of a typical decision making structure found in most of the programming languages:



C# provides following types of decision making statements. Click the following links to check their detail.

Statement	Description
if statement	An **if statement** consists of a boolean expression followed by one or more statements.
if...else statement	An **if statement** can be followed by an optional **else statement**, which executes when the boolean expression is false.
nested if statements	You can use one **if** or **else if** statement inside another **f**or **else if** statement(s).
switch statement	A **switch** statement allows a variable to be tested for equality against a list of values.

nested switch statements	You can use one **switch** Statement inside another **switch** statement(s).

The ? : Operator:

We have covered **conditional operator ? :** in previous chapter which can be used to replace **if...else** statements. It has the following general form:

```
Exp1 ? Exp2 : Exp3;
```

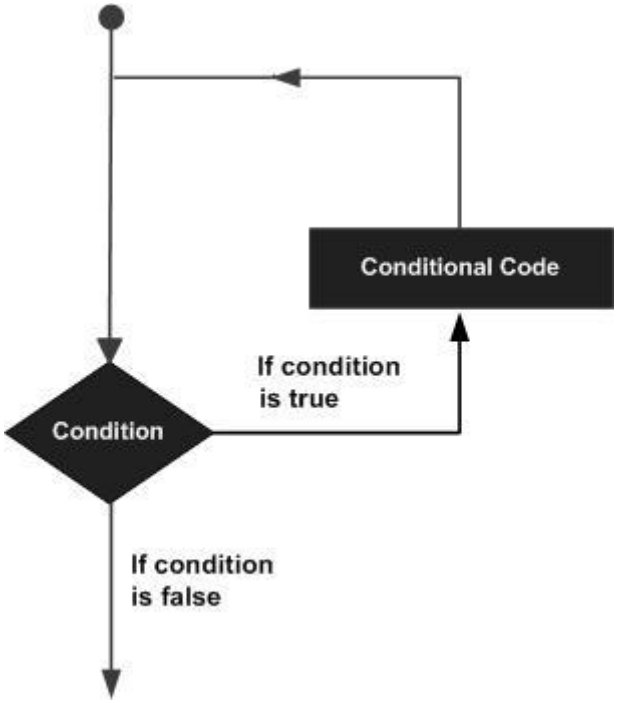
Where Exp1, Exp2, and Exp3 are expressions. Notice the use and placement of the colon.

The value of a ? expression is determined as follows: Exp1 is evaluated. If it is true, then Exp2 is evaluated and becomes the value of the entire ? expression. If Exp1 is false, then Exp3 is evaluated and its value becomes the value of the expression.

There may be a situation, when you need to execute a block of code several number of times. In general, the statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on.

Programming languages provide various control structures that allow for more complicated execution paths.

A loop statement allows us to execute a statement or a group of statements multiple times and following is the general from of a loop statement in most of the programming languages:



C# provides following types of loop to handle looping requirements. Click the following links to check their detail.

Loop Type	Description
while loop	It repeats a statement or a group of statements while a given condition is true. It tests the condition before executing the loop body.
for loop	It executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.
do...while loop	It is similar to a while statement, except that it tests the condition at the end of the loop body
nested loops	You can use one or more loop inside any another while, for or do..while loop.

Loop Control Statements

Loop control statements change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed.

C# provides the following control statements. Click the following links to check their details.

Control Statement	Description
break statement	Terminates the **loop** or **switch** statement and transfers execution to the statement immediately following the loop or switch.
continue statement	Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.

Encapsulation is defined 'as the process of enclosing one or more items within a physical or logical package'. Encapsulation, in object oriented programming methodology, prevents access to implementation details.

Encapsulation is implemented by using **access specifiers**. An **access specifier** defines the scope and visibility of a class member. C# supports the following access specifiers:

- Public
- Private
- Protected
- Internal
- Protected internal

1.3 Choosing appropriate protocols and advantages, disadvantages of protocols

Interactions between email servers and clients are governed by email protocols. The three most common email protocols are POP, IMAP and MAPI. Most email software operates under one of these (and many products support more than one). The most important reason for knowing of their existence? To understand that the correct protocol must be selected, and correctly configured, if you want your email account to work.

The Post Office Protocol (currently in version 3, hence POP3) allows email client software to retrieve email from a remote server. The Internet Message Access Protocol (now in version 4 or IMAP4) allows a local email client to access email messages that reside on a remote server. There's a related protocol called SMTP, which we also discuss below.

The Messaging Application Programming Interface (MAPI) is a proprietary email protocol of Microsoft, that can be used by Outlook (Microsoft's email client software) to communicate with Microsoft Exchange (its email server software). It provides somewhat more functionality than an IMAP protocol; unfortunately, as a proprietary protocol, it works only for Outlook-Exchange interactions.

In computing, the **Post Office Protocol** (**POP**) is an application-layer Internet standard protocol used by local e-mail clients to retrieve e-mail from a remote server over a TCP/IP connection.[1] POP has been developed through several versions, with version 3 (**POP3**) being the current standard.

Virtually all modern e-mail clients and servers support POP3, and it along with IMAP (Internet Message Access Protocol) are the two most prevalent Internet standard protocols for e-mail retrieval, with many webmail service providers such as Gmail, Outlook command Yahoo! Mail also providing support for either IMAP or POP3 to allow mail to be downloaded.

POP supports simple download-and-delete requirements for access to remote mailboxes (termed mail drop in the POP RFC's). Although most POP clients have

an option to leave mail on server after download, e-mail clients using POP generally connect, retrieve all messages, store them on the user's PC as new messages, delete them from the server, and then disconnect. Other protocols, notably IMAP, (Internet Message Access Protocol) provide more complete and complex remote access to typical mailbox operations. In the late 1990s and early 2000s, fewer Internet Service Providers (ISPs) supported IMAP due to the storage space that was required on the ISP's hardware. Contemporary e-mail clients supported POP, then over time popular mail client software added IMAP support.

A POP3 server listens on well-known port 110. Encrypted communication for POP3 is either requested after protocol initiation, using the STLS command, if supported, or by POP3S, which connects to the server using Transport Layer Security (TLS) or Secure Sockets Layer (SSL) on well-known TCP port 995.

Available messages to the client are fixed when a POP session opens the mail drop, and are identified by message-number local to that session or, optionally, by a unique identifier assigned to the message by the POP server. This unique identifier is permanent and unique to the mail drop and allows a client to access the same message in different POP sessions. Mail is retrieved and marked for deletion by message-number. When the client exits the session, the mail marked for deletion is removed from the mail drop.

The original POP3 specification supported only an unencrypted USER/PASS login mechanism or Berkeley .rhosts access control. POP3 currently supports several authentication methods to provide varying levels of protection against illegitimate access to a user's e-mail. Most are provided by the POP3 extension mechanisms. POP3 clients support SASLauthentication methods via the AUTH extension. MIT Project Athena also produced a Kerberized version. RFC 1460 introduced APOP into the core protocol. APOP is achallenge/response protocol which uses the MD5 hash function in an attempt to avoid replay attacks and disclosure of the shared secret. Clients implementing APOP include Mozilla Thunderbird, Opera Mail, Eudora, KMail, Novell Evolution, RimArts' Becky!, Windows Live Mail, PowerMail, Apple Mail,

and Mutt. RFC 1460 was obsoleted by RFC 1725, which was in turn obsoleted by RFC 1939.

Simple Mail Transfer Protocol (**SMTP**) is an Internet standard for electronic mail (e-mail) transmission. First defined by RFC 821in 1982, it was last updated in 2008 with the Extended SMTP additions by RFC 5321 - which is the protocol in widespread use today.

SMTP by default uses TCP port 25. The protocol for mail submission is the same, but uses port 587. SMTP connections secured by SSL, known as SMTPS, default to port 465 (nonstandard, but sometimes used for legacy reasons).

Although electronic mail servers and other mail transfer agents use SMTP to send and receive mail messages, user-level client mail applications typically use SMTP only for sending messages to a mail server for relaying. For receiving messages, client applications usually use either POP3 or IMAP.

Although proprietary systems (such as Microsoft Exchange and Lotus Notes/Domino) and webmail systems (such as Hotmail, Gmail and Yahoo! Mail) use their own non-standard protocols to access mail box accounts on their own mail servers, all use SMTP when sending or receiving email from outside their own systems.

Email is submitted by a mail client (MUA, mail user agent) to a mail server (MSA, mail submission agent) using SMTP on TCP port 587. Most mailbox providers still allow submission on traditional port 25. From there, the MSA delivers the mail to its mail transfer agent (MTA, mail transfer agent). Often, these two agents are just different instances of the same software launched with different options on the same machine. Local processing can be done either on a single machine, or split among various appliances; in the former case, involved processes can share files; in the latter case, SMTP is used to transfer the message internally, with each host configured to use the next appliance as a smart host. Each process is an MTA in its own right; that is, an SMTP server.

The **Hypertext Transfer Protocol** (**HTTP**) is an application protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web.

Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text. HTTP is the protocol to exchange or transfer hypertext.

The standards development of HTTP was coordinated by the Internet Engineering Task Force (IETF) and the World Wide Web Consortium (W3C), culminating in the publication of a series of Requests for Comments (RFCs), most notably RFC 2616 (June 1999), which defined HTTP/1.1, the version of HTTP most commonly used today. In June 2014, RFC 2616 was retired and HTTP/1.1 was

HTTP functions as a request-response protocol in the client-server computing model. A web browser, for example, may be the client and an application running on a computer hosting a web site may be the server. The client submits an HTTP request message to the server. The server, which provides resources such as HTML files and other content, or performs other functions on behalf of the client, returns a response message to the client. The response contains completion status information about the request and may also contain requested content in its message body.

A web browser is an example of a user agent (UA). Other types of user agent include the indexing software used by search providers (web crawlers), voice browsers, mobile apps, and other software that accesses, consumes, or displays web content.

HTTP is designed to permit intermediate network elements to improve or enable communications between clients and servers. High-traffic websites often benefit from web cache servers that deliver content on behalf of upstream servers to improve response time. Web browsers cache previously accessed web resources and reuse them when possible to reduce network traffic. HTTP proxy

servers at private network boundaries can facilitate communication for clients without a globally routable address, by relaying messages with external servers.

HTTP is an application layer protocol designed within the framework of the Internet Protocol Suite. Its definition presumes an underlying and reliable transport layer protocol, and Transmission Control Protocol (TCP) is commonly used. However HTTP can use unreliable protocols such as the User Datagram Protocol (UDP), for example in Simple Service Discovery Protocol (SSDP).

HTTP resources are identified and located on the network by Uniform Resource Identifiers (URIs)—or, more specifically, Uniform Resource Locators (URLs)—using the http or https URI schemes. URIs and hyperlinks in Hypertext Markup Language (HTML) documents form webs of inter-linked hypertext documents[6].

1.4 Analyzing different types of Encription

RSA is an algorithm used by modern computers to encrypt and decrypt messages. It is an asymmetric cryptographic algorithm. Asymmetric means that there are two different keys. This is also called public key cryptography, because one of them can be given to everyone. The other key must be kept private. It is based on the fact that finding the factors of a integer is hard (the factoring problem). RSA stands for Ron Rivest, Adi Shamir and Leonard Adleman, who first publicly described it in 1978. A user of RSA creates and then publishes the product of two large prime numbers, along with an auxiliary value, as their public key. The prime factors must be kept secret. Anyone can use the public key to encrypt a message, but with currently published methods, if the public key is large enough, only someone with knowledge of the prime factors can feasibly decode the message[8].

RSA involves a public key and private key. The public key can be known to everyone, it is used to encrypt messages. Messages encrypted using the public key can only be decrypted with the private key. The keys for the RSA algorithm are generated the following way:

1. Choose two different large random prime numbers p and q

- Calculate $n = pq$ n is the modulus for the public key and the private keys

2. Calculate the totient: $\phi(n) = (p-1)(q-1)$.

3. Choose an integer e such that $1 < e < \phi(n)$, and e is coprime to $\phi(n)$ **ie:** e and $\phi(n)$ share no factors other than 1; gcd(e,$\phi(n)$) = 1.

- e is released as the public key exponent

4. Compute d to satisfy the congruence relation $de \equiv 1 \pmod{\phi(n)}$ **ie:** $de = 1 + k\phi(n)$ for some integer k.

- d is kept as the private key exponent

Notes on the above steps:

- Step 1: Numbers can be probabilistically tested for primality.
- Step 3: changed in PKCS#1 v2.0 to $\lambda(n) = \operatorname{lcm}(p-1, q-1)$ instead of $\phi(n) = (p-1)(q-1)$.
- Step 4: A popular choice for the public exponents is e = 2^{16} + 1 = 65537. Some applications choose smaller values such as e = 3, 5, or 35 instead. This is done to make encryption and signature verification faster on small devices like smart cards but small public exponents may lead to greater security risks.
- Steps 4 and 5 can be performed with the extended Euclidean algorithm; see modular arithmetic.

The **public key** is made of the modulus n and the public (or encryption) exponent e

The **private key** is made of the modulus n and the private (or decryption) exponent d which must be kept secret.

- For efficiency a different form of the **private key** can be stored:
- p and q: the primes from the key generation,
- $d \mod (p-1)$ and $d \mod (q-1)$: often called dmp1 and dmq1.
- $q^{-1} \mod (p)$: often called iqmp
- All parts of the private key must be kept secret in this form. p and q are sensitive since they are the factors of n, and allow computation of d given e. If p and q are not stored in this form of the private key then they are securely deleted along with other intermediate values from key generation[7].
- Although this form allows faster decryption and signing by using the Chinese Remainder Theorem (CRT) it is considerably less secure since it enables side channel attacks. This is a particular problem if implemented on smart cards, which benefit most from the improved efficiency. (Start with $y = x^e \pmod n$ and let the card decrypt that. So it computes $y^d \pmod p$ or $y^d \pmod q$ whose results give some value z. Now, induce an error in one of the computations. Then $\gcd(z-x, n)$ will reveal $p$or q.)

Encrypting messages

Alice gives her public key (n & e) to Bob and keeps her private key secret. Bob wants to send message **M** to Alice.

First he turns **M** into a number smaller than n by using an agreed-upon reversible protocol known as a padding scheme. He then computes the cipher text c corresponding to:

$$c = m^e \mod n$$

This can be done quickly using the method of exponentiation by squaring. Bob then sends c to Alice[9].

Decrypting messages

Alice can recover m from c by using her private key d in the following procedure:

$$m = c^d \mod n$$

Given m, she can recover the original message **M**.

The decryption procedure works because first

$$c^d \equiv (m^e)^d \equiv m^{ed} \pmod{n}.$$

Now, since

$$ed \equiv 1 \pmod{p-1}$$ and

$$ed \equiv 1 \pmod{q-1}$$

Fermat's little theorem yields

$$m^{ed} \equiv m \pmod{p}$$ and

$$m^{ed} \equiv m \pmod{q}.$$

Since p and q are distinct prime numbers, applying the Chinese remainder theorem to these two congruences yields

$$m^{ed} \equiv m \pmod{pq}.$$

Thus,

$$c^d \equiv m \pmod{n}.$$

A working example

Here is an example of RSA encryption and decryption. The parameters used here are artificially small, but you can also use OpenSSL to generate and examine a real keypair.

1. Choose two random prime numbers
2. :$p = 61$ and $q = 53$
3. Compute $n = pq$
4. :$n = 61 * 53 = 3233$
5. Compute the totient $\phi(n) = (p-1)(q-1)$
6. :$\phi(n) = (61-1)(53-1) = 3120$
7. Choose $e > 1$ coprime to 3120
8. :$e = 17$
9. Choose d to satisfy $de \equiv 1 \pmod{\phi(n)}$
10. :$d = 2753$
11. :17 * 2753 = 46801 = 1 + 15 * 3120.

The **public key** is ($n = 3233$, $e = 17$). For a padded message m the encryption function is:

$$c = m^e \mod n = m^{17} \mod 3233.$$

The **private key** is ($n = 3233$, $d = 2753$). The decryption function is:

$$m = c^d \mod n = c^{2753} \mod 3233.$$

For example, to encrypt $m = 123$, we calculate

$$c = 123^{17} \mod 3233 = 855$$

To decrypt $c = 855$, we calculate

$$m = 855^{2753} \mod 3233 = 123.$$

Both of these calculations can be computed efficiently using the square-and-multiply algorithm for modular exponentiation.

Padding schemes

When used in practice, RSA must be combined with some form of padding scheme, so that no values of M result in insecure cipher texts. RSA used without padding may have some problems:

- The values m = 0 or m = 1 always produce cipher texts equal to 0 or 1 respectively, due to the properties of exponentiation.
- When encrypting with small encryption exponents (e.g., e = 3) and small values of the m, the (non-modular) result of m^e may be strictly less than the modulus n. In this case, cipher texts may be easily decrypted by taking the eth root of the cipher text with no regard to the modulus.
- RSA encryption is a deterministic encryption algorithm. It has no random component. Therefore, an attacker can successfully launch a chosen plaintext attack against the cryptosystem. They can make a dictionary by encrypting likely plaintexts under the public key, and storing the resulting cipher texts. The attacker can then observe the communication channel. As soon as they see cipher texts that match the ones in their dictionary, the attackers can then use this dictionary in order to learn the content of the message.

Signing messages

Suppose Alice uses Bob's public key to send him an encrypted message. In the message, she can claim to be Alice but Bob has no way of verifying that the message was actually from Alice since anyone can use Bob's public key to send him encrypted messages. So, in order to verify the origin of a message, RSA can also be used to sign a message.

Suppose Alice wishes to send a signed message to Bob. She produces a hash value of the message, raises it to the power of d mod n (just like when decrypting a message), and attaches it as a "signature" to the message. When Bob receives the signed message, he raises the signature to the power of e mod n (just like encrypting a message), and compares the resulting hash value with the message's actual hash value. If the two agree, he knows that the author of the message was in

possession of Alice's secret key, and that the message has not been tampered with since.

Note that secure padding schemes such as RSA-PSS are as essential for the security of message signing as they are for message encryption, and that the same key should never be used for both encryption and signing purposes.

The **Advanced Encryption Standard** (**AES**), also referenced as **Rijndael** (its original name), is a specification for theen cryption of electronic data established by the U.S. National Institute of Standards and Technology (NIST) in 2001.

AES is based on the Rijndael cipher developed by two Belgian cryptographers, Joan Daemen and Vincent Rijmen, who submitted a proposal to NIST during the AES selection process. Rijndael is a family of ciphers with different key and block sizes.

For AES, NIST selected three members of the Rijndael family, each with a block size of 128 bits, but three different key lengths: 128, 192 and 256 bits.

AES has been adopted by the U.S. government and is now used worldwide. It supersedes the Data Encryption Standard (DES), which was published in 1977. The algorithm described by AES is a symmetric-key algorithm, meaning the same key is used for both encrypting and decrypting the data[10].

In the United States, AES was announced by the NIST as U.S. FIPS PUB 197 (FIPS 197) on November 26, 2001. This announcement followed a five-year standardization process in which fifteen competing designs were presented and evaluated, before the Rijndael cipher was selected as the most suitable (see Advanced Encryption Standard process for more details).

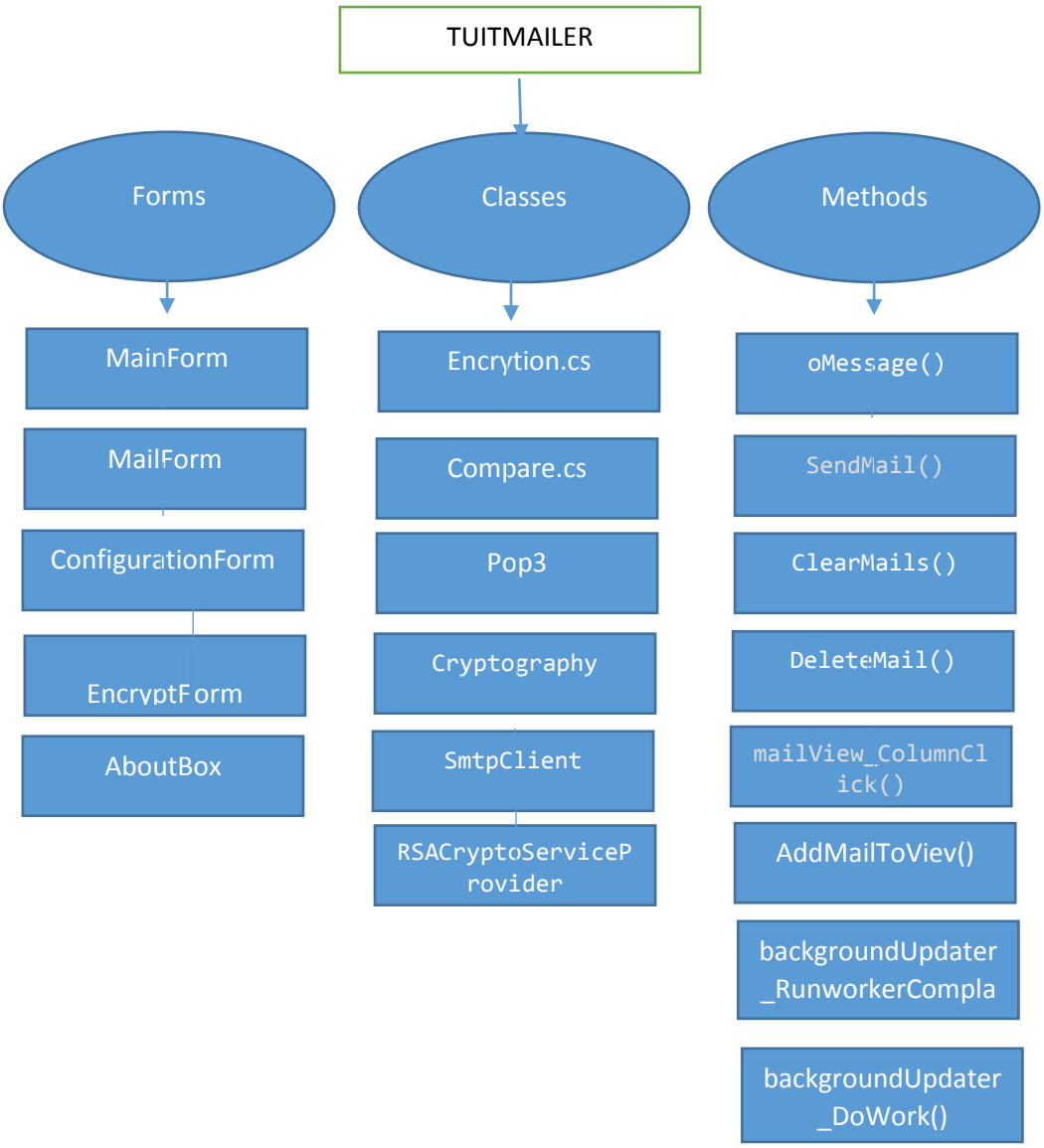
AES became effective as a federal government standard on May 26, 2002 after approval by the Secretary of Commerce. AES is included in the ISO/IEC 18033-3 standard. AES is available in many different encryption packages, and is the first publicly accessible and open cipher approved by the National Security Agency (NSA) for top secret information when used in an NSA approved cryptographic module (see Security of AES, below).

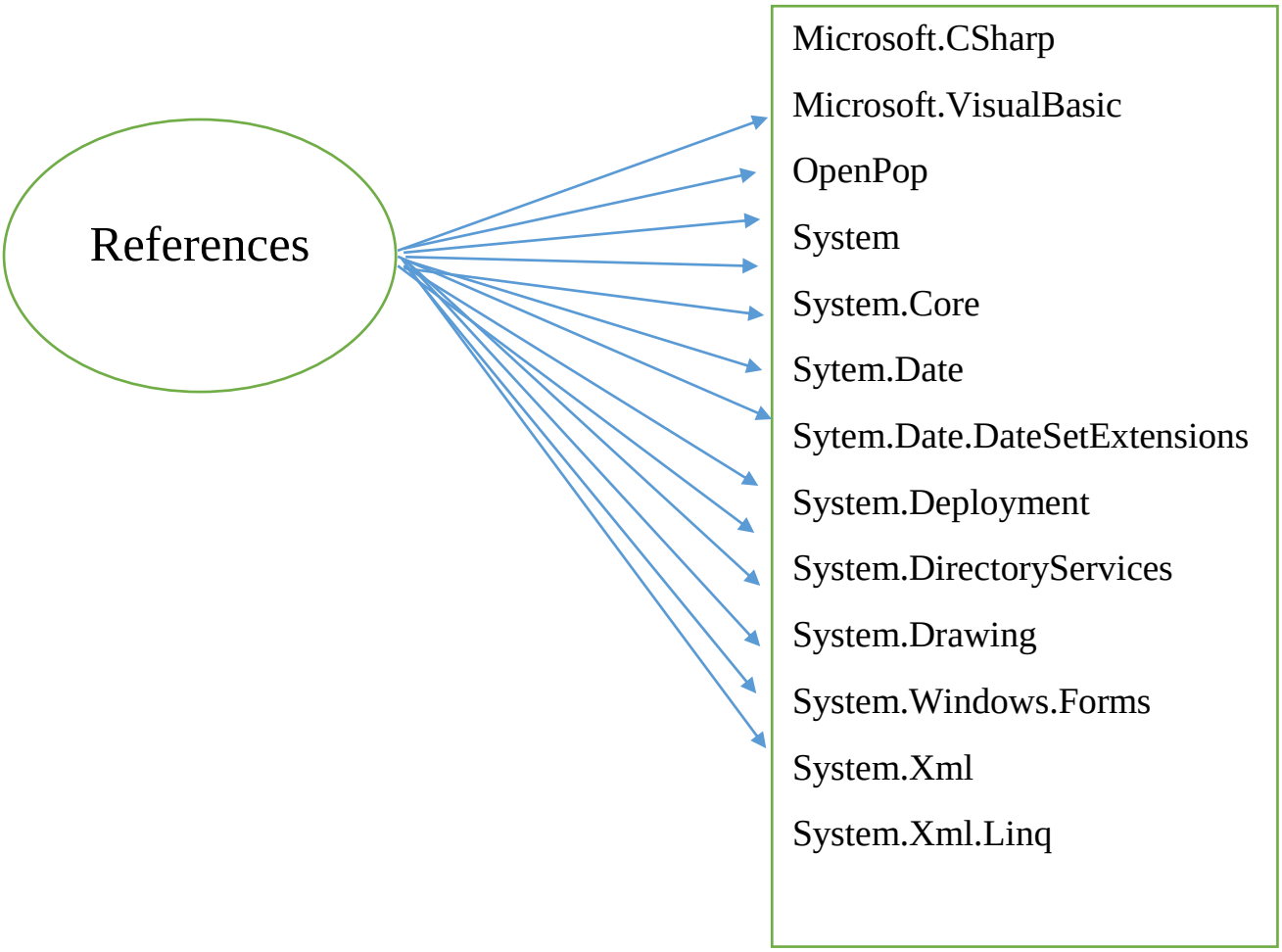
CHAPTER II. USING PROGRAMMING LANGUAGE IN ORDER TO CREATING A SOFTWARE

2.1. The structure of software

This scheme shows the architecture of software. As shown below, I have used

five forms, number of classes, methods and some other standard references.




References
Microsoft.CSharp
Microsoft.VisualBasic
OpenPop
System
System.Core
Sytem.Date
Sytem.Date.DateSetExtensions
System.Deployment
System.DirectoryServices
System.Drawing
System.Windows.Forms
System.Xml
System.Xml.Linq

MainForm is the main form of the program that means from this window we can use other windows of the program. There a lot of methods, classes and objects. Main forms generally is for reading inbox messages. First of all, I have created structure namely “oMessage” which contains four properties.

```
struct oMessage
  {
    public oMessage(string r, string s, string b,ListViewItem l)
    {
      recipients = r;
      subject = s;
      body = b;
      listItem = l;
    }

    public string recipients;
    public string subject;
    public string body;
    public ListViewItem listItem;
  }
```

In **ConfigurationForm,** we can connect to other email servers from this window. There are two popular email servers: Hotmail, Gmail. However there is a section called **“other”** which means that you can add other email servers.

```
private void accountSelecter_SelectedIndexChanged(object sender, EventArgs e)
    {
      switch (accSelecter.Text)
      {
        case "Select account...":
          inGroupBox.Visible = false;
          outGroupBox.Visible = false;
          accGroupBox.Visible = false;
          break;
        case "Gmail":
          inGroupBox.Visible = false;
          outGroupBox.Visible = false;
          accGroupBox.Visible = true;

          accUser.Text = "@gmail.com";
          inServer.Text = "pop.gmail.com";
          inPort.Text = "995";
          inSSL.Checked = true;
          outServer.Text = "smtp.gmail.com";
          outPort.Text = "25";
          outSSL.Checked = true;
```

```
            break;
        case "Hotmail":
            inGroupBox.Visible = false;
            outGroupBox.Visible = false;
            accGroupBox.Visible = true;

            accUser.Text = "@hotmail.com";
            inServer.Text = "pop3.live.com";
            inPort.Text = "995";
            inSSL.Checked = true;
            outServer.Text = "smtp.live.com";
            outPort.Text = "25";
            outSSL.Checked = true;
            break;
        case "Other":
            inGroupBox.Visible = true;
            outGroupBox.Visible = true;
            accGroupBox.Visible = false;
            break;
    }

}
```

Now I will give information about my classes that created and standard.

Constructor	**Description**
Pop3Client	Initializes a new instance of the **Pop3Client** class.
Static Method	**Description**
CertificateValidationCallback	Validates a server certificate.
Instance Method	**Description**
AuthenticateClearText	Authenticates to the POP3 server using clear text.
AuthenticateNtlm	Authenticates to the POP3 server using NTLM.
Connect	Connects to the POP3 server.
DeleteMessage	Deletes a message.
Disconnect	Disconnects from the POP3 server.
Dispose	Disposes the **Pop3Client** object instance.
DownloadHeaders	Downloads message headers.
DownloadMessage	Downloads the entire message (body and header).
RetrieveMessageCount	Retrieves the message count.
Instance Property	**Description**
Pop3Port	Gets the connection state.

Pop3Server	Gets the POP3 server port.
WelcomeBanner	Gets the POP3 server name.
RetrieveMessageCount	Gets the POP3 server welcome banner.

```
SmtpClient smtpClient = new SmtpClient();

  smtpClient.Host = outServer;
            smtpClient.Port = outPort;
            smtpClient.Credentials = new System.Net.NetworkCredential(outUser, outPass);
            smtpClient.EnableSsl = outSSL;
```

Here using smtp class I have created smtp object which sends messages.

The next class is **"comparer"** which sorts inbox messages. To be specific, it sorts read and unread messages.

The next part are for methods that I created. I have created a number of methods and used some standard ones. The first is **SendMail()** which send mails to other email clients and we can send messages at the same time number of emails.

```
void SendMail(string recipients, string subject)
    {
      if (outServer != null)
      {
        MailForm mailForm = new MailForm();

        mailForm.setRecipients(recipients);
        mailForm.setSubject(subject);

        if (mailForm.ShowDialog() == System.Windows.Forms.DialogResult.OK)
        {
          ListViewItem newItem = new ListViewItem(mailForm.getRecipients());
          newItem.SubItems.Add(mailForm.getSubject());
          newItem.SubItems.Add(DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss"));
          newItem.Group = mailView.Groups[0];
          newItem.ImageIndex = 2;

          outbox.Add(new oMessage(mailForm.getRecipients(), mailForm.getSubject(),
mailForm.getMessage(), mailView.Items.Add(newItem)));

          backgroundSender.RunWorkerAsync();
        }
      }
            }
```

AddMailToView() adds messages from inbox to MainForm.

```
void AddMailToView(ListViewItem item)
    {
      if (mailView.InvokeRequired)
      {
        mailView.Invoke(new AddMailToViewDelegate(AddMailToView), new object[] { item });
```

```
        }
        else
        {
           mailView.Items.Add(item);
        }
              }
```

DeleteMail() method is for deleting inbox messages.

```
void DeleteMail(ListViewItem item)
      {
        if (mailView.InvokeRequired)
        {
           mailView.Invoke(new DeleteMailDelegate(DeleteMail), new object[] { item });
        }
        else
        {
           item.Remove();
        }
              }
```

ClearMails() method allows to clear main window but still it will not delete inbox messages.

```
void ClearMails()
      {
        if (mailView.InvokeRequired)
        {
           mailView.Invoke(new ClearMailsDelegate(ClearMails), new object[] { });
        }
        else
        {
           messages.Clear();
           mailView.Items.Clear();
        }
              }
```

2.2. Using encryption RSA and RIJNDAEL in email client

To encrypt sending messages I have used two different types of encryption types: Rijndail and RSA. To encrypt in Rijndail I have created function namely EncryteStringToBytes which has three parameters. The first is for receiving string, the second is for key and the third for encryption combination.

```
public static byte[] EncryptStringToBytes(string plainText, byte[] Key, byte[] IV)
    {
        // Check arguments.
        if (plainText == null || plainText.Length <= 0)
            throw new ArgumentNullException("plainText");
        if (Key == null || Key.Length <= 0)
            throw new ArgumentNullException("Key");
        if (IV == null || IV.Length <= 0)
            throw new ArgumentNullException("Key");
        byte[] encrypted;
        // Create an RijndaelManaged object
        // with the specified key and IV.
        using (RijndaelManaged rijAlg = new RijndaelManaged())
        {
            rijAlg.Key = Key;
            rijAlg.IV = IV;

            // Create a decrytor to perform the stream transform.
            ICryptoTransform encryptor = rijAlg.CreateEncryptor(rijAlg.Key, rijAlg.IV);

            // Create the streams used for encryption.
            using (MemoryStream msEncrypt = new MemoryStream())
            {
                using (CryptoStream csEncrypt = new CryptoStream(msEncrypt, encryptor,
CryptoStreamMode.Write))
                {
                    using (StreamWriter swEncrypt = new StreamWriter(csEncrypt))
                    {

                        //Write all data to the stream.
                        swEncrypt.Write(plainText);
                    }
                    encrypted = msEncrypt.ToArray();
                }
            }
        }


        // Return the encrypted bytes from the memory stream.
        return encrypted;

            }
```

As for decryption, I have created function called “DecryptStringFromBytes” which has three parameters: First adds cipher text, second is a key, and the third one is for combination.

```
public static string DecryptStringFromBytes(byte[] cipherText, byte[] Key, byte[] IV)
    {
        // Check arguments.
        if (cipherText == null || cipherText.Length <= 0)
            throw new ArgumentNullException("cipherText");
        if (Key == null || Key.Length <= 0)
            throw new ArgumentNullException("Key");
        if (IV == null || IV.Length <= 0)
            throw new ArgumentNullException("Key");

        // Declare the string used to hold
        // the decrypted text.
        string plaintext = null;

        // Create an RijndaelManaged object
        // with the specified key and IV.
        using (RijndaelManaged rijAlg = new RijndaelManaged())
        {
            rijAlg.Key = Key;
            rijAlg.IV = IV;

            // Create a decrytor to perform the stream transform.
            ICryptoTransform decryptor = rijAlg.CreateDecryptor(rijAlg.Key, rijAlg.IV);

            // Create the streams used for decryption.
            using (MemoryStream msDecrypt = new MemoryStream(cipherText))
            {
                using (CryptoStream csDecrypt = new CryptoStream(msDecrypt, decryptor,
CryptoStreamMode.Read))
                {
                    using (StreamReader srDecrypt = new StreamReader(csDecrypt))
                    {

                        // Read the decrypted bytes from the decrypting stream
                        // and place them in a string.
                        plaintext = srDecrypt.ReadToEnd();
                    }
                }
            }

        }

        return plaintext;

            }
```

Encryption in RSA

When is clicked “Send” button it automatically encrypts to RSA and sends to server.

```
        private void sendButton_Click(object sender, EventArgs e)
    {
        if (recipientsTextbox.Text == "" | subjectTextbox.Text == "" | messageTextbox.Text == "")
        {
            MessageBox.Show(this,"Something is missing","Ops",MessageBoxButtons.OK);
        }
        else
        {
            if (rijndaelToolStripMenuItem.Checked)
            {
                if (encryptionKey.Text.Length == 16 | encryptionKey.Text.Length == 24 |
encryptionKey.Text.Length == 32)
                {
                    messageTextbox.Text =
Convert.ToBase64String(Encryption.EncryptStringToBytes(messageTextbox.Text,
Encoding.ASCII.GetBytes(encryptionKey.Text), Encoding.ASCII.GetBytes("1234567890123456")));
                    this.DialogResult = System.Windows.Forms.DialogResult.OK;
                    this.Close();
                }
                else
                {
                    MessageBox.Show(this, "Encryption key has to be 16, 24 or 32 chars in length.", "Ops",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
            else if (rSAToolStripMenuItem.Checked)
            {
                try {
                    RSACryptoServiceProvider RSA = new RSACryptoServiceProvider();
                    RSA.FromXmlString(encryptionKey.Text);
                    messageTextbox.Text =
Convert.ToBase64String(RSA.Encrypt(Encoding.Unicode.GetBytes(messageTextbox.Text), false));
                    this.DialogResult = System.Windows.Forms.DialogResult.OK;
                    this.Close();
                }
                catch (Exception err)
                {
                    MessageBox.Show(err.Message);
                }
            }
            else
            {
                this.DialogResult = System.Windows.Forms.DialogResult.OK;
                this.Close();
            }
        }
    }
```

Cryptography

One of the biggest challenges when dealing with the security and encryption for a system, is the determination of the correct ciphering paradigm. In .NET, there is a copious amount of libraries available for use in the **System.Cryptography** namespace. A significant amount of these libraries have been deprecated, usually, due to vulnerabilities being subsequently exposed, so it is very easy to use something that may be as watertight as a sieve.

This is further compounded by the fact that the cryptography API's are very detailed and low level – they are not easy to use for a novice – the consequences of setting a single parameter incorrectly results in a security implementation that may as well not exist. Consequently, it is imperative that this subject never be approached in a typical agile/sprint manner – security should definitely be approached using a waterfall model. Have no hesitation to advise any manager or architect that your solution "will be ready, when it is ready". The agile methodology is typically about adding units of functionality in a YAGNI way, accruing technical debt that can be paid back later, and refactoring applied, this just simply not a correct or acceptable approach when dealing with the security of a system. Do ensure you take the time to do a lot of research, understanding the pitfalls of various implementations is vital to a robust security implementation.

Rijndael Class

.NET Framework 4.6 and 4.5

Represents the base class from which all implementations of the Rijndael symmetric encryption algorithm must inherit.

Inheritance Hierarchy

System.Object

System.Security.Cryptography.SymmetricAlgorithm

System.Security.Cryptography.Rijndael

System.Security.Cryptography.RijndaelManaged

Namespace:

System.Security.Cryptography

Assembly: mscorlib (in mscorlib.dll)

Methods

Name	Description
Clear	Releases all resources used by the SymmetricAlgorithm class. (Inherited from SymmetricAlgorithm.)
Create()	Creates a cryptographic object to perform the Rijndael algorithm.
CreateDecryptor()	Creates a symmetric decryptor object with the current Key property and initialization vector (IV). (Inherited fromSymmetricAlgorithm.)
CreateEncryptor()	Creates a symmetric encryptor object with the current Key property and initialization vector (IV). (Inherited fromSymmetricAlgorithm.)
Dispose()	Releases all resources used by the current instance of the SymmetricAlgorithm class. (Inherited from SymmetricAlgorithm.)
Equals(Object)	Determines whether the specified object is equal to the current object. (Inherited from Object.)
Finalize	Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.(Inherited from Object.)
GetType	Gets the Type of the current

	instance. (Inherited from Object.)
ValidKeySize	Determines whether the specified key size is valid for the current algorithm. (Inherited from SymmetricAlgorithm.)

Remarks

This algorithm supports key lengths of 128, 192, or 256 bits.

The Rijndael class is the predecessor of the Aes algorithm. You should use the Aes algorithm instead of Rijndael. For more information, see the entry The Differences Between Rijndael and AES in the .NET Security blog.

RSACryptoServiceProvider Class

Performs asymmetric encryption and decryption using the implementation of the RSA algorithm provided by the cryptographic service provider (CSP). This class cannot be inherited.

Inheritance Hierarchy

System.Object

System.Security.Cryptography.AsymmetricAlgorithm

System.Security.Cryptography.RSA

System.Security.Cryptography.RSACryptoServiceProvider

Namespace:

System.Security.Cryptography

Assembly: mscorlib (in mscorlib.dll)

Remarks

This is the default implementation of RSA.

The RSACryptoServiceProvider supports key sizes from 384 bits to 16384 bits in increments of 8 bits if you have the Microsoft Enhanced Cryptographic Provider installed. It supports key sizes from 384 bits to 512 bits in increments of 8 bits if you have the Microsoft Base Cryptographic Provider installed.

Valid key sizes are dependent on the cryptographic service provider (CSP) that is used by the RSACryptoServiceProvider instance. Windows CSPs enable

keys sizes of 384 to 16384 bits for Windows versions prior to Windows 8.1, and key sizes of 512 to 16384 bits for Windows 8.1. For more information, see CryptGenKey function in the Windows documentation.

Interoperation with the Microsoft Cryptographic API (CAPI)

Unlike the RSA implementation in unmanaged CAPI, the RSACryptoServiceProvider class reverses the order of an encrypted array of bytes after encryption and before decryption. By default, data encrypted by the RSACryptoServiceProvider class cannot be decrypted by the CAPI **CryptDecrypt** function and data encrypted by the CAPI**CryptEncrypt** method cannot be decrypted by the RSACryptoServiceProvider class.

If you do not compensate for the reverse ordering when interoperating between APIs, the RSACryptoServiceProvider class throws a CryptographicException.

To interoperate with CAPI, you must manually reverse the order of encrypted bytes before the encrypted data interoperates with another API. You can easily reverse the order of a managed byte array by calling the Array.Reverse method.

2.3. File stream

To work with messages I have used file stream. Here this function downloads messages through browser provider from server and saves them in ”.eml”.

```
if (!System.IO.Directory.Exists(Application.UserAppDataPath + "\\Inbox"))
        {
           System.IO.Directory.CreateDirectory(Application.UserAppDataPath + "\\Inbox");
        }
        if (!System.IO.Directory.Exists(Application.UserAppDataPath + "\\Outbox"))
        {
           System.IO.Directory.CreateDirectory(Application.UserAppDataPath + "\\Outbox");
        }
        if (!System.IO.Directory.Exists(Application.UserAppDataPath + "\\RSA"))
        {
           System.IO.Directory.CreateDirectory(Application.UserAppDataPath + "\\RSA");
        }
```

After that, if files are exist in this path this function opens new message

```
if (!File.Exists(Application.UserAppDataPath + "\\Inbox\\" + message.Headers.MessageId + ".eml"))
      {
         message.Save(new FileInfo(Application.UserAppDataPath + "\\Inbox\\" + message.Headers.MessageId + ".eml"));
      }
```

If “browse” button is clicked, it will open all downloaded messages in new default window.

```
private void browseButton_Click(object sender, EventArgs e)
   {
     Process.Start("explorer.exe" ,"\"" + Application.UserAppDataPath + "\\Inbox\"");
   }
```

Use the FileStream class to read from, write to, open, and close files on a file system, and to manipulate other file-related operating system handles, including pipes, standard input, and standard output. You can use the Read, Write, CopyTo, and Flush methods to perform synchronous operations, or the ReadAsync, WriteAsync, CopyToAsync, andFlushAsync methods to perform asynchronous operations. Use the asynchronous methods to perform resource-

intensive file operations without blocking the main thread. This performance consideration is particularly important in a Windows Store app or desktop app where a time-consuming stream operation can block the UI thread and make your app appear as if it is not working. FileStream buffers input and output for better performance.

This type implements the IDisposable interface. When you have finished using the type, you should dispose of it either directly or indirectly. To dispose of the type directly, call its Dispose method in a **try**/**catch** block. To dispose of it indirectly, use a language construct such as **using** (in C#) or **Using** (in Visual Basic). For more information, see the "Using an Object that Implements IDisposable" section in the IDisposable interface topic.

The IsAsync property detects whether the file handle was opened asynchronously. You specify this value when you create an instance of the FileStream class using a constructor that has an *isAsync*, *useAsync*, or *options* parameter. When the property is **true**, the stream utilizes overlapped I/O to perform file operations asynchronously. However, theIsAsync property does not have to be **true** to call the ReadAsync, WriteAsync, or CopyToAsync method. When the IsAsync property is **false** and you call the asynchronous read and write operations, the UI thread is still not blocked, but the actual I/O operation is performed synchronously.

CHAPTER III. WORKING WITH “TUITMAILER” AND FUNCTIONALITY OF THIS PROGRAMM

3.1. The results after testing process

As you can see, there is a list of inbox messages(Figure).

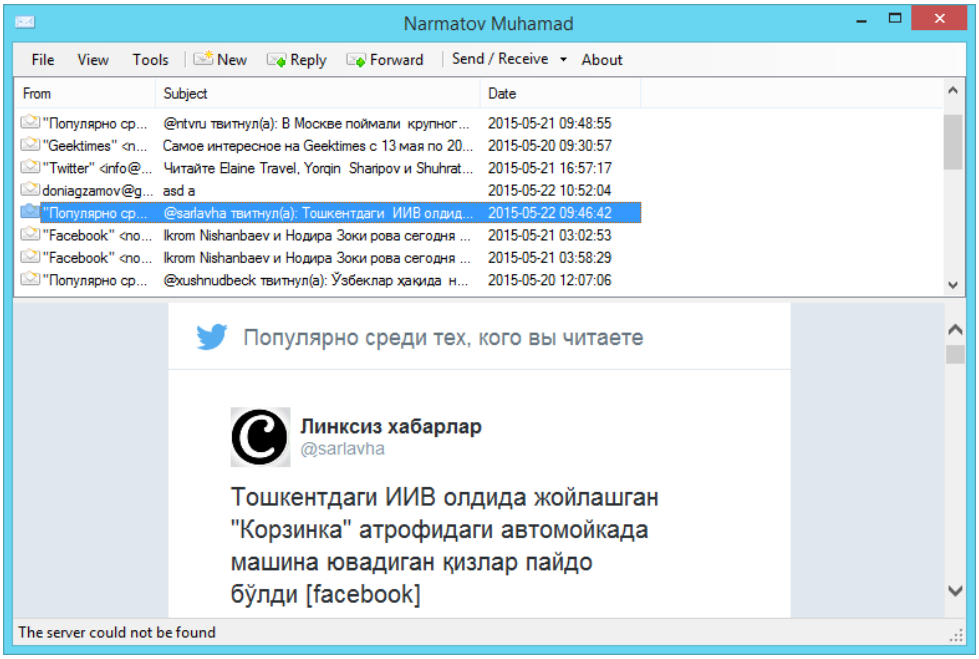


Figure 3.1. First testing “Inbox”

This is a window that sends messages to receivers (Figure -).

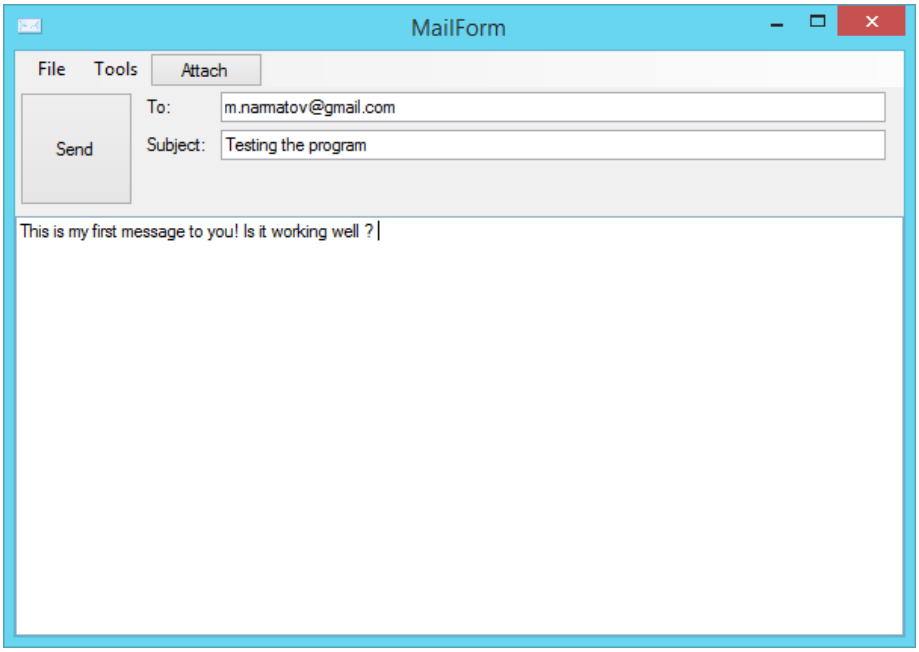


Figure 3.2. Testing “Outbox”

So I received the email that I send to me from my email client app(Figure -).

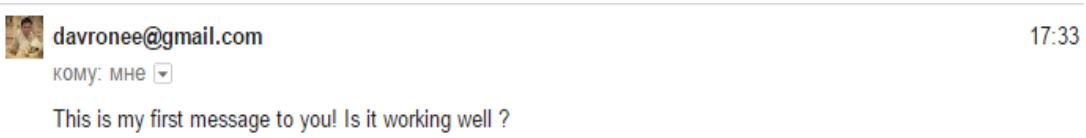


Figure 3.3. Received from my email app

3.2. Manual of the program.

Before beginning my project, I have analyzed a number of popular softwares which work with email client. After mastering some visual knowledge about email clients I have tried to avoid some sophisticated parts which can be difficult for ordinary users. First of all I have paid a great attention to the comfortability of the program namely “TuitMailer”. To be more accurate, the interface of the program is very simple and functional that can understand counting from ordinary users to expert programmers.

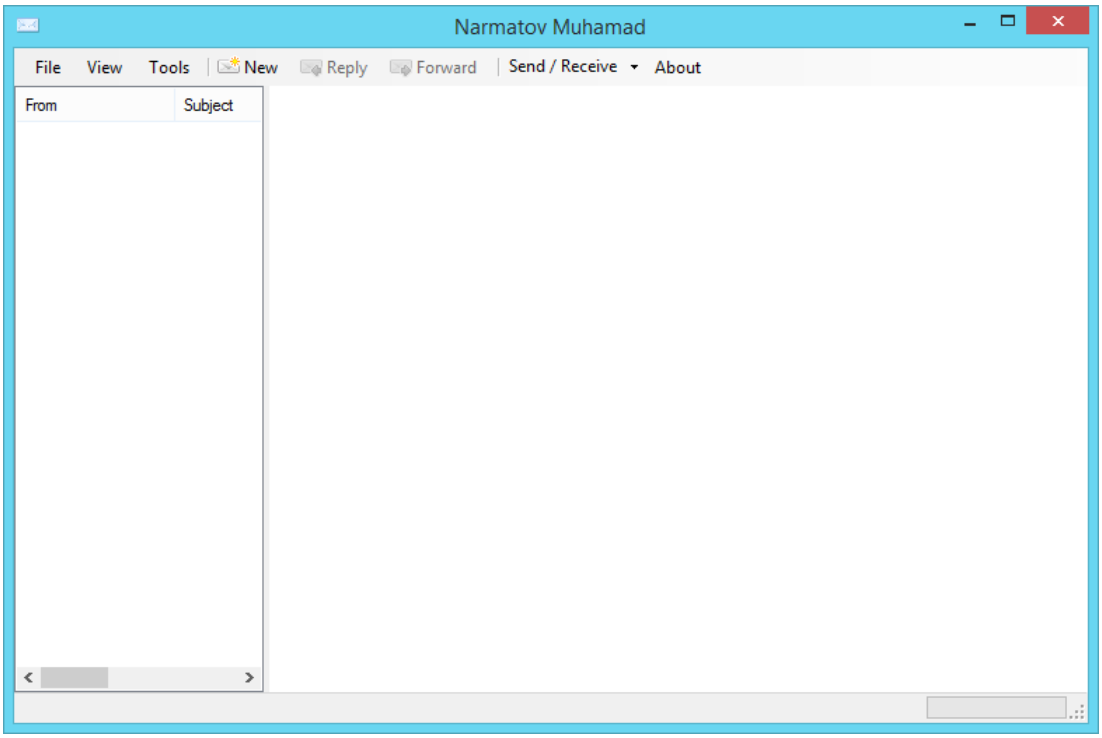


Figure 3.4. The general view of the program.

Majority of the tasks we can achieve by using common tools. The view of the common tools we can see in 3.5. – Picture.

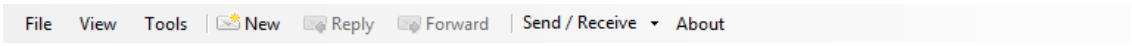


Figure 3.5. Common tools

As you can see there the first menu is for "File" and here you can find 3 buttons namely **"File > Configuration", "File > About", "File > Exit**". In **"File > Configuration**" we can change some options like connecting to servers manually. As for **"File > About"** you could get information about author(3.6. – Picture).

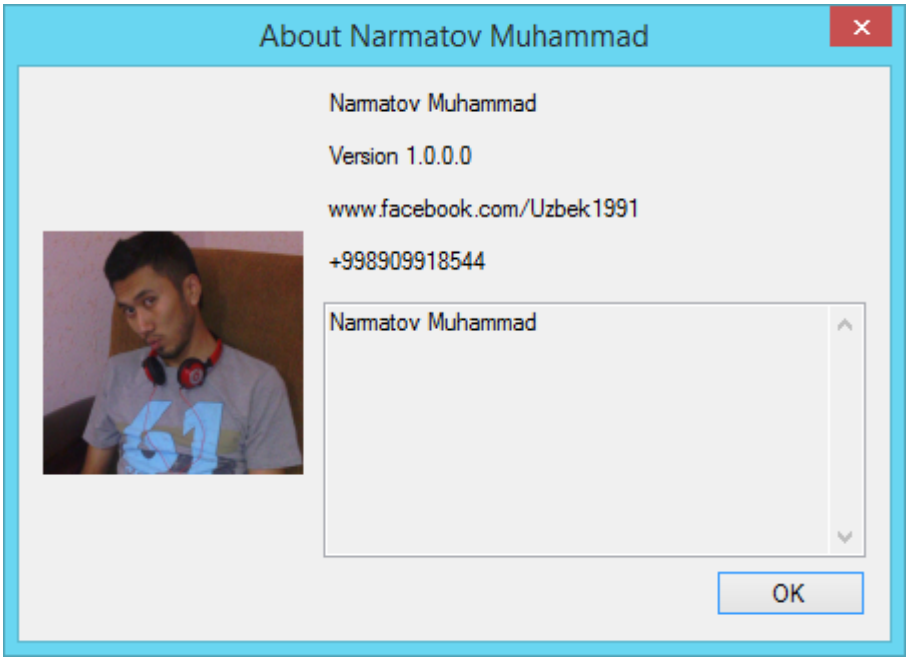


Figure 3.6. Window "About"

And the last button of the "File" menu is **"File > Exit"** for closing the program.

After that here is the second menu called “View”, here are three types of interface view (3.7. – Picture).

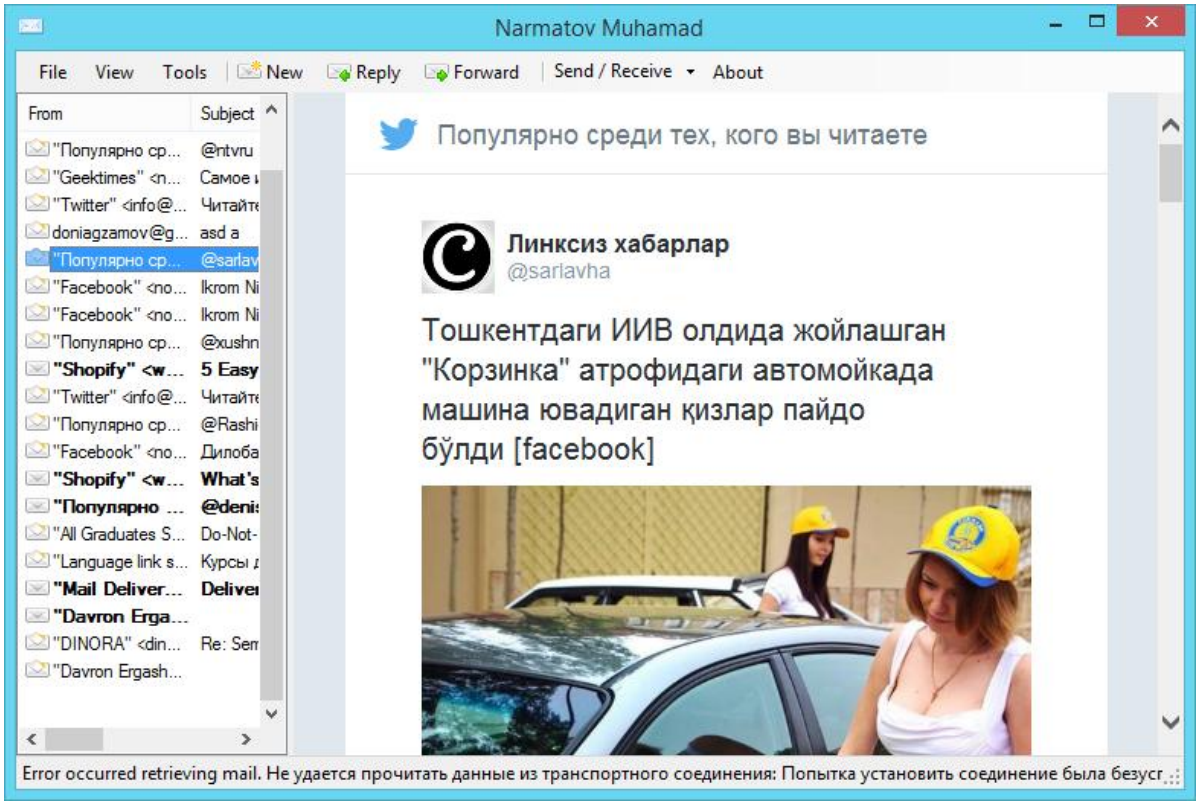


Figure 3.7. The view of “View” menu(Vertical)

The second variety of the reading pane is horizontal view. Here the interface of the program divided by two side(3.8 – Picture)

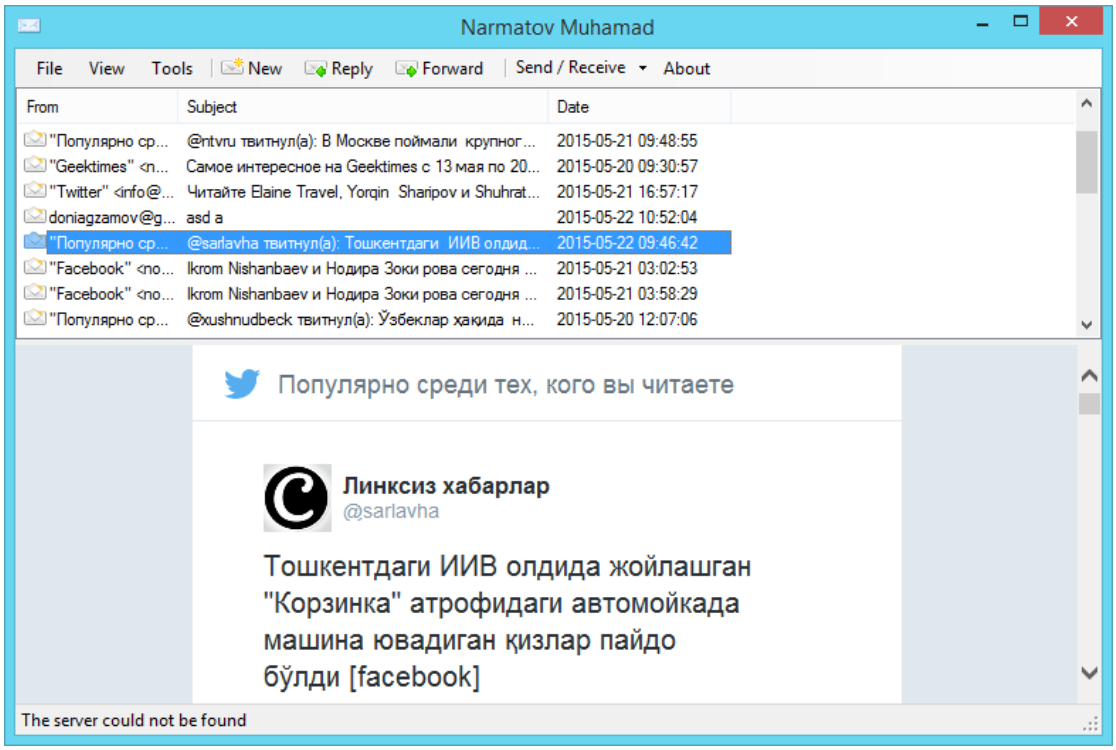


Figure 3.8. Horizontal view

The next menu is “Tools” which contains two buttons **“File > Browse”** and **“File > Decrypt”**. The first button is for expert user that shows the path of downloaded message files(3.9 - Picture).

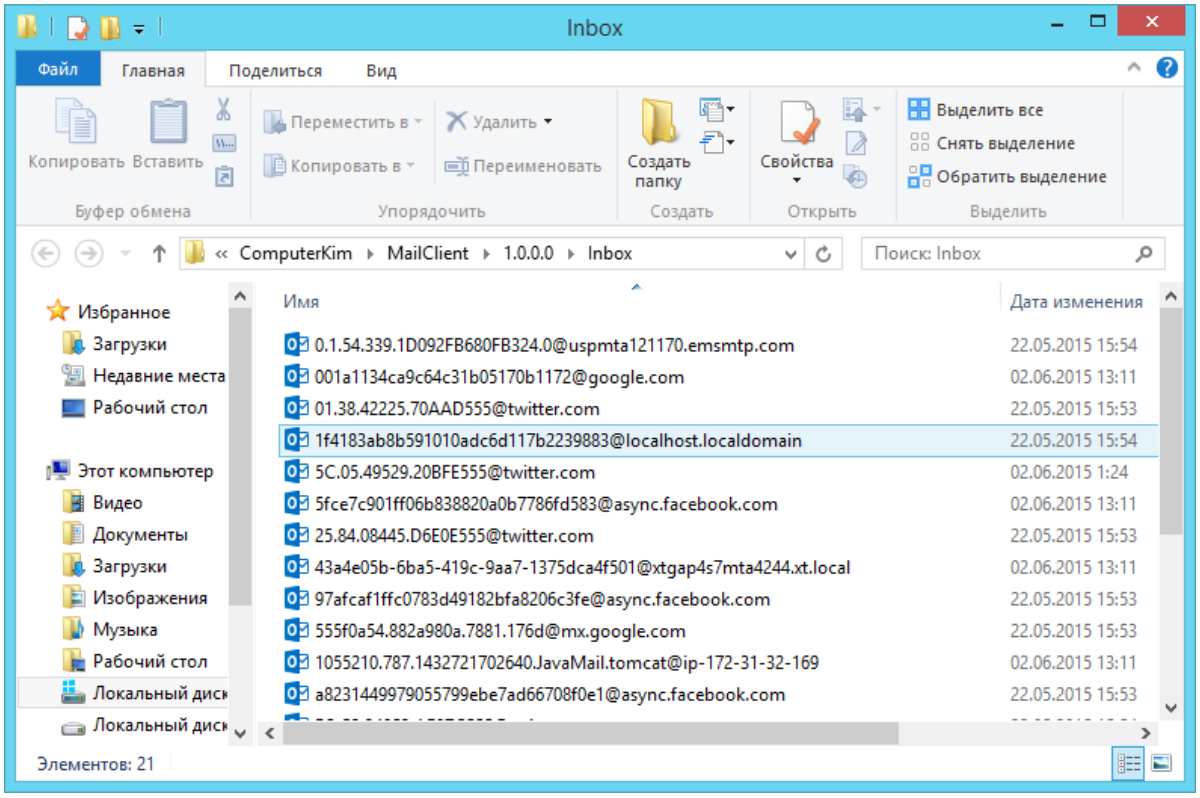


Figure 3.9. Downloaded messages

The second menu is for decryption that can be decrypted in two different ways of downloaded inbox messages (3.10. - Picture).

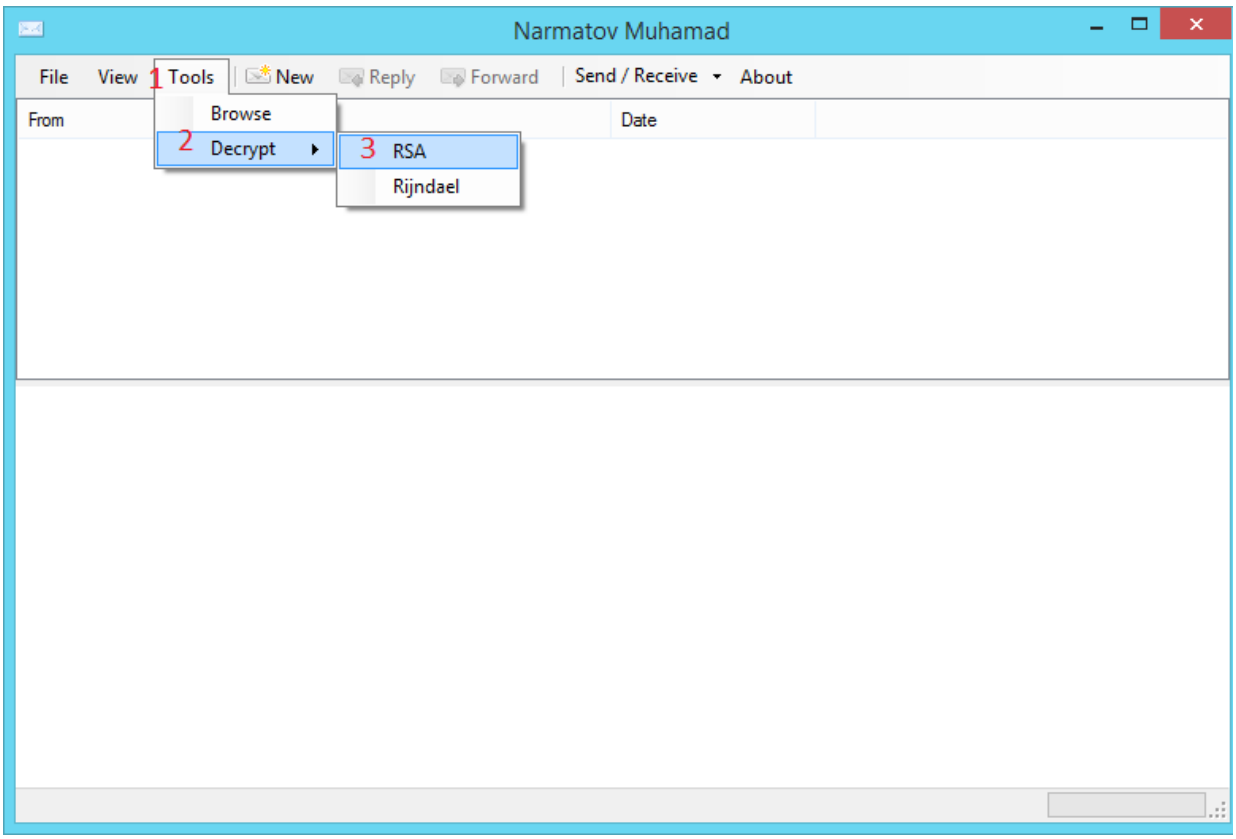


Figure 3.10. Decryption view.

One of the most important menu is “New” that can sent new messages to contacts (3.11. - Picture).

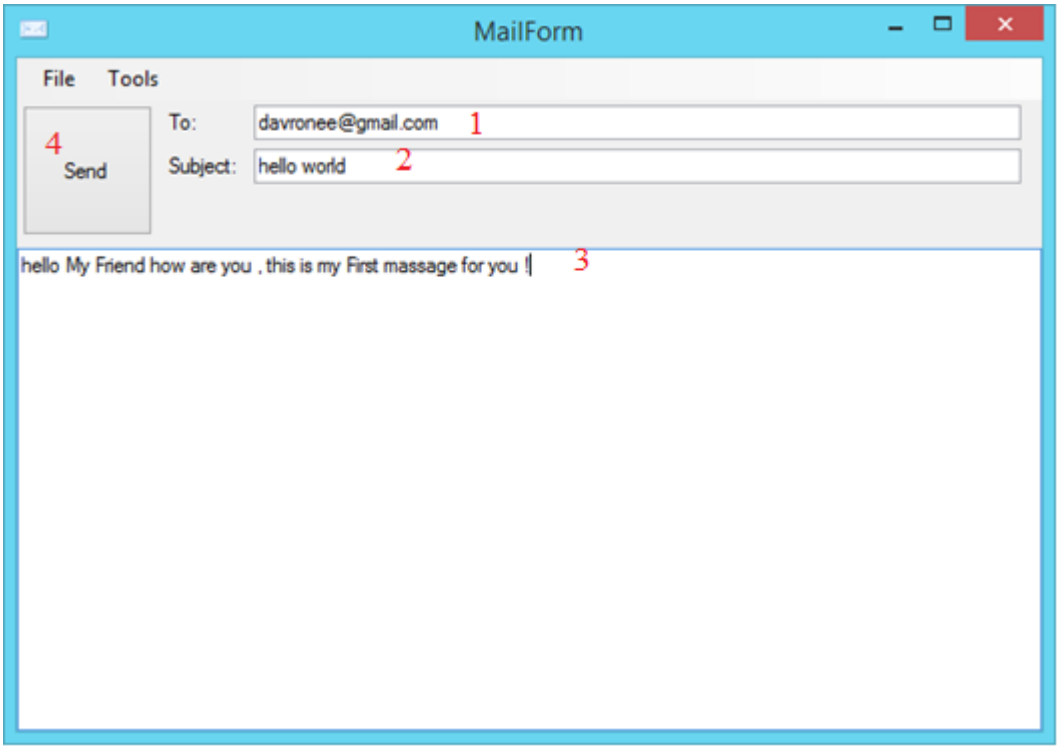


Figure 3.11. ”New” button view.

Follow menu is “Reply” that can answer the received emails (3.12. - Picture).

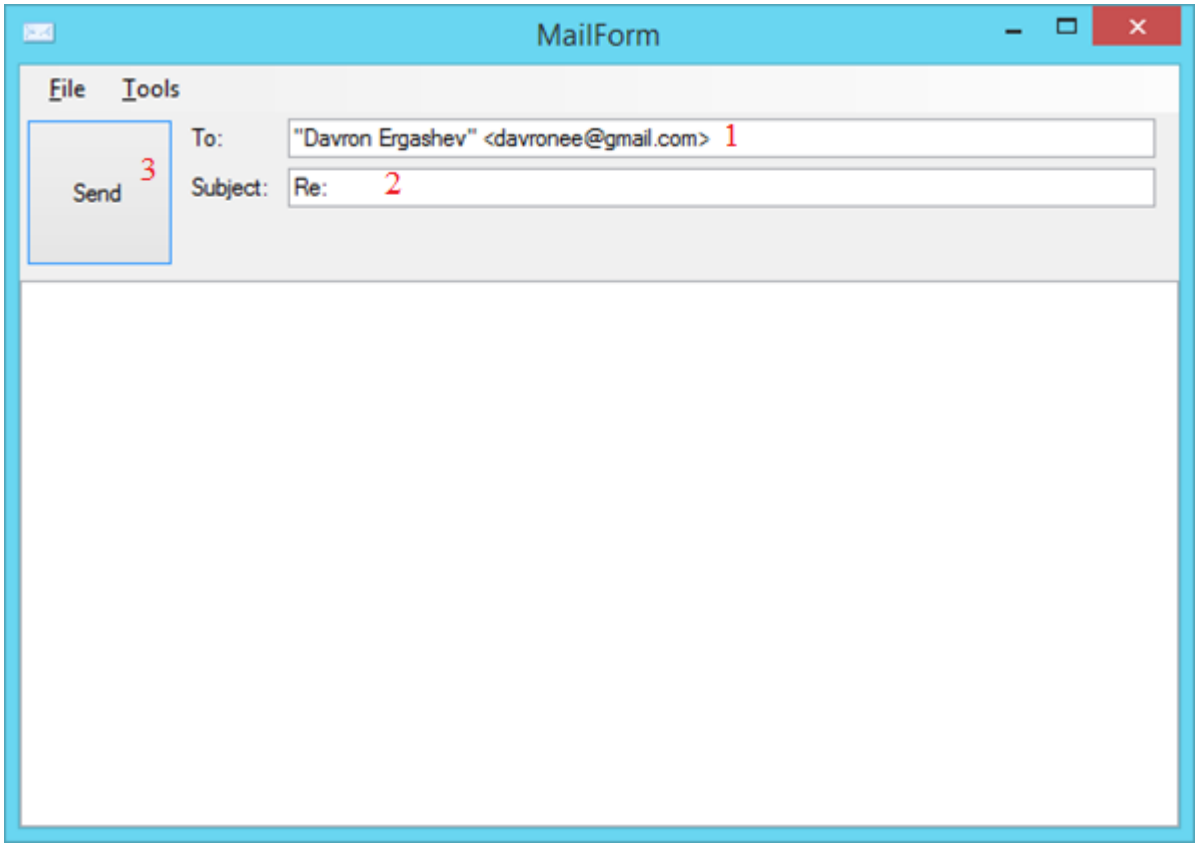


Figure 3.12. “Reply” window.

If you would like to send the message that received you can use “Forward” menu (3.9 - Picture)

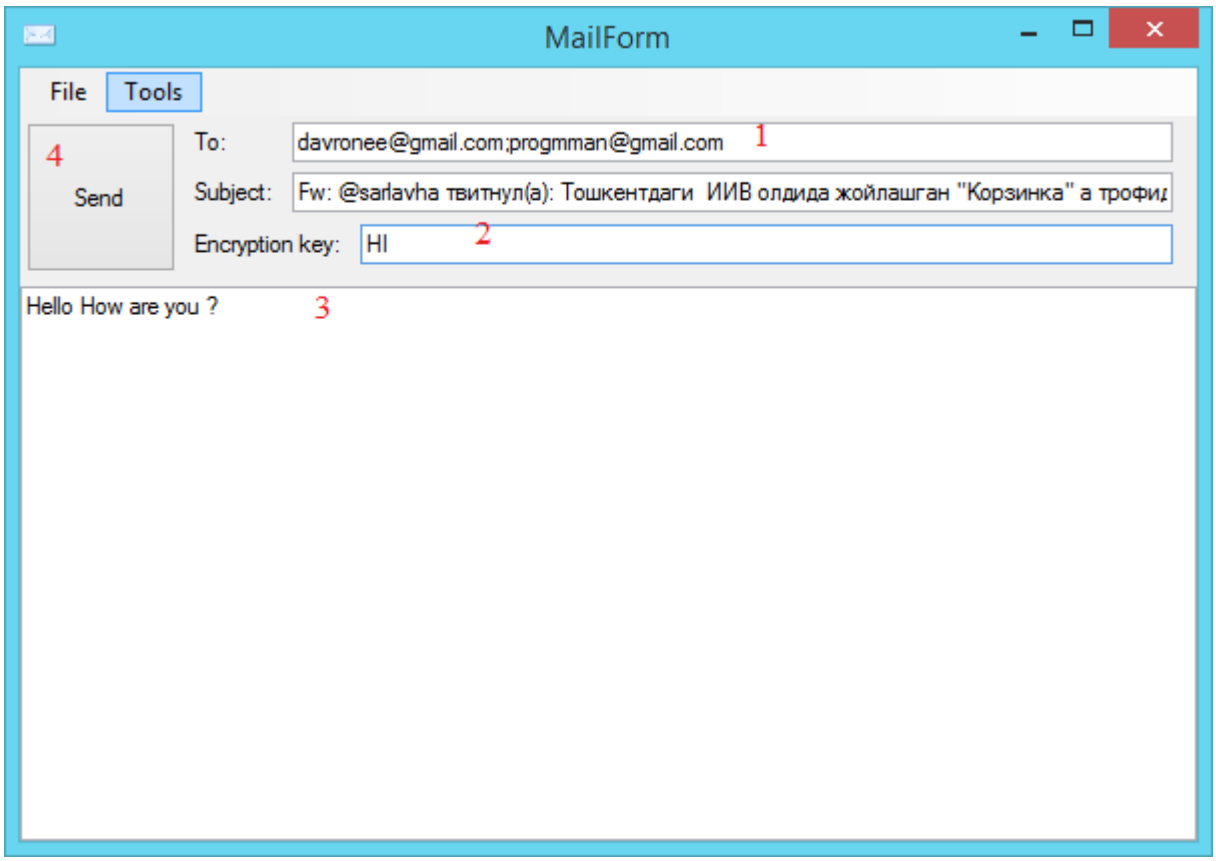


Figure 3.13. “Forward” window

In the next step called “Send/Receive” menu you can find two button namely **“Send/Receive > Send Mails”** and **“Send/Receive > Update mails”. “Send/Receive > Send Mails”** button show the sent messages and **“Send/Receive > Update Mails”** button refreshes inbox messages.

Finally, the last button devoted to the author’s information namely **“About”** (3.10 - Picture).

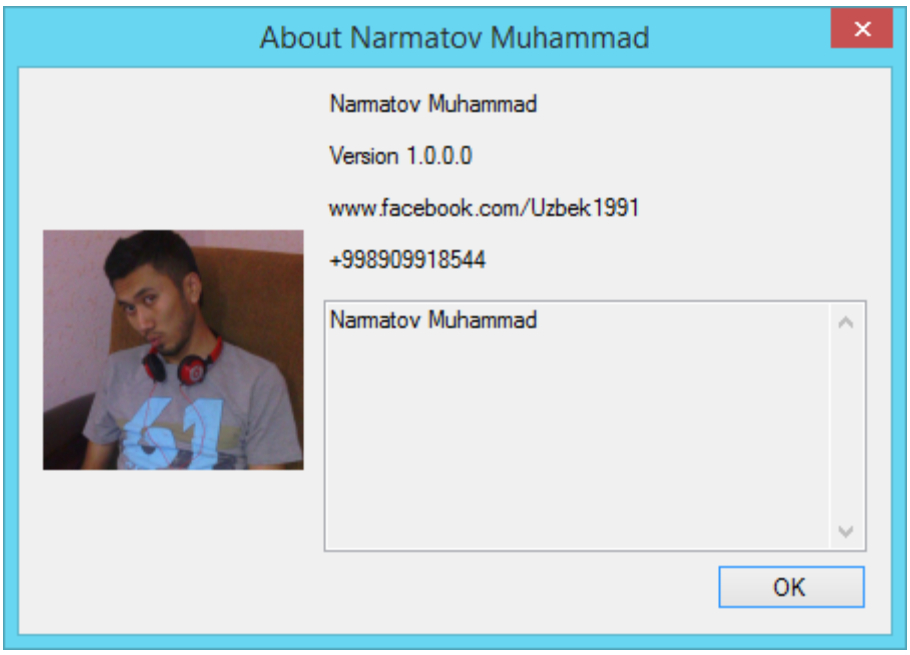


Figure 3.14. “About” window.

3.2. Adding an account and server ports

“TuitMailer” not only works with “Gmail”, “YAHOO” or “Hotmail” but also any kind of email servers. In order to add other email server you should know server’s POP3, SMTP and port. If you want to add another email server you should click on **“File > Configuration > Other.** As you can see the “Configuration” window divided by two part: “**Incoming server**” and “**Outgoing server**”. Generally, “Incoming server” works with SMTP server since this server responsible for sending messages. As for “**Outgoing server**” works with POP3 server which receives emails. Here is example how to fill the labels:

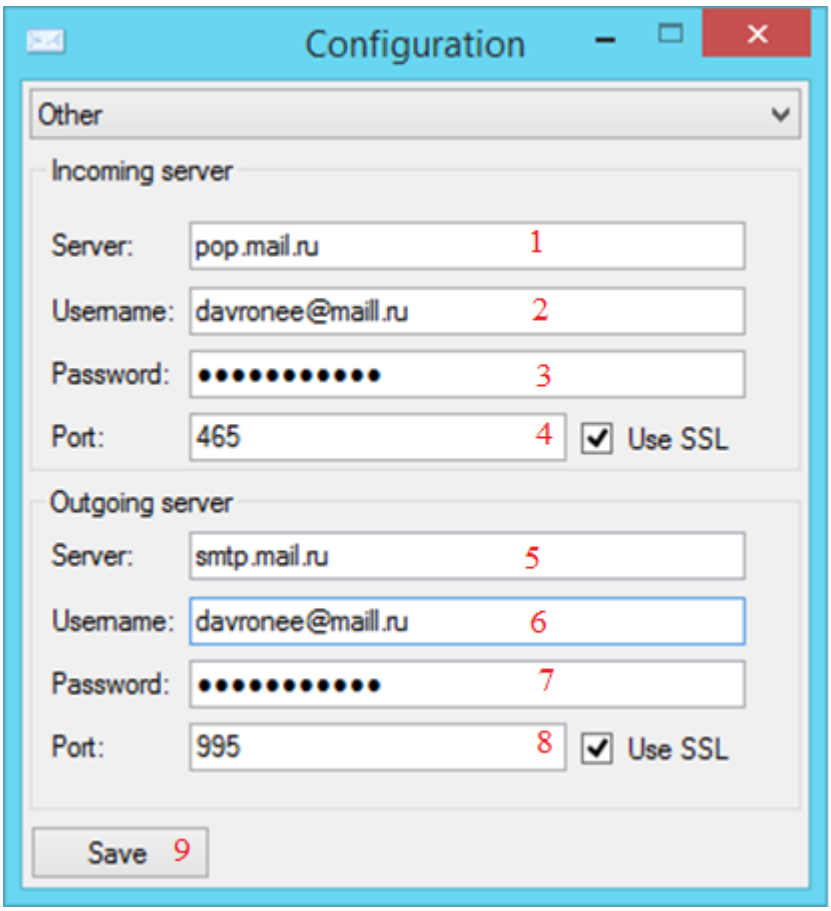


Figure 3.15 Configuration window

Here is some information about servers POP3, PORT and SMTP

INBOX.UZ

POP3: mail.inbox.uz, PORT: 110

SMTP: mail.inbox.uz, PORT: 25

MAIL.RU

POP3: pop.mail.ru, PORT: 995

SMPT: smtp.mail.ru, PORT: 465

YANDEX.RU

POP3: pop.yandex.ru, PORT: 995

SMPT: smtp.yandex.ru, PORT: 465

CHAPTER IV. LIFE SAFETY

The working at personal computer is characterized by effecting of the following harmful factors to human's organism:

- The boosted value of a voltage;
- X-radiation originating at braking of an electron ray on an inside surface of a kinescope of a monitor;
- The dark blue luminophore of the monitor screen has partial radiation in ultra-violet domain of a spectrum;
- Electromagnetic waves of a low frequencies concerned with the operation of sweep circuits of a cathode ray tube of the monitor;
- Electromagnetic fields (effect of reflection);
- The intensive noise level;
- Electric waves (radiofrequency);
- Electrostatic field.

For provision of best conditions for effective and safe operation at PC it's necessary to establish such working conditions, which would be comfortable and reduce the effects of the given harmful factors as much as possible. It is essential, that the listed harmful factors match with the set rules and norms.

4.1. Safety of work

The noise is an aggregate of sounds effecting on human organism, and interfering to its operation and rest.

The researches show that in conditions of noise aural functions suffer first of all. But the effect of noise is not limited to influence only on hearing. It provokes noticeable shifts of a number of physiological and mental functions. The noise harmfully influences a nervous system and reduces velocity and accuracy of sense-motor processes, the number of errors at the solving of the intellectual tasks increases. The noise renders noticeable influence on human's attention and provokes negative emotions.

The fundamental noisemaker in rooms where computer is situated is the air

conditioning equipment, print and copy techniques, and fans of cooling systems in computers. According to the specifications the noise level on a work station should not exceed 59 dB. The normalized noise levels are provided by usage of sound-proof materials for rooms facing.

The fundamental measures of noise control are:

- Elimination or attenuation of noise factors in its source during design and maintenance of an equipment;
- Isolation of noisemakers from an environment particularly by a means of deafening and acoustical absorption;
- Rational design of product engineering.

Protection from electric radiation

According to the specifications, normalized parameters in a frequency band 60 kHz – 300 MHz are E and H intensities of an electromagnetic field. At frequency from 60 kHz up to 3 MHz the electric field intensity may be up to 50 V/m, and magnetic intensity can be up to 5A/m.

During the operation of the monitor the electric waves of very low frequencies are generated. The intensity of an electromagnetic field near the monitor is from 4 up to 7 milligauss. The magnetic intensity higher than 4 milligauss is harmful to human. Such radiation is a reason of anomalies at pregnancy and causes cellular level variations. Magnifying of quantity of positively charged ions in air near to the switched on monitor also negatively affects the human organism. The medical research, which have been carried out in USA, shows that long-time staying in deionized atmosphere effects on metabolism and results in variation of biochemical reaction in blood on cellular level, which quite often results in stresses.

The charge of a static electricity, resulting in originating of an electrostatic discharge is accumulated on the surface of a monitor during its work. The protection from electromagnetic effect may be provided using screens with electro conductive surface. In a construction of these screens possibility of grounding, as a rule, is stipulated. These screens provide reliable protection

against electric waves. There are following types of screens:

- "Polaroid" type screens;
- Screens of a film type;
- Glass screens;
- Cancellated screens.

Among the representatives of above listed types there are screens with a conducting surface and grounding possibility. These screens protect the owner from electrostatic and electric waves of the monitor.

Protection from the effect of an electrostatic field

The fact of that the cathode ray tube in the monitor is charged negatively is a reason of electrostatic charge originating, which results in accumulation of positive aero ions. At the surplus of positive aero ions their positive charge starts to repel micro particles, which always present in air. They become dispersed and bombard the human's face and eyes. It makes very negative effect (irritation of a nervous system and skin).

The contents of easy aero ions of both signs in a breath area of the working man practically can change in limits from 1,5-10 up to 5-10 in 1 cm of an air. The intensity of an electrostatic field should not increase to exceed these values.

It's rather preferable to use grounded protective filter for the protection from effect of an electrostatic charge.

Also, as protection from static electricity in rooms with PC it is possible to use neutralizers and humidifiers, and floors should have an antistatic coating.

For maintenance of normalized values of positive and negative ions concentration, it is recommended to install conditioners, devices for air ionization in work rooms with PCs, or to carry out natural airing lasting not less than 10 minutes after every 2 work hours.

For the purposes of preventing the harmful influence of motes with aero ions to an organism of the operating personnel it is necessary to carry out daily damp cleanup of rooms, and not less often than 1 time in scheduled work period delete a dust from screens at switched off monitor.

4.2. Working conditions

Appropriate working conditions in workplaces are significant factor of efficiency of business.

The most significant physical factor is the industrial microclimate, which is characterized by a level of temperature and air humidity, and also intensity of a radiation level.

The used computers do not require the creation of the special microclimatic conditions for operation and normally functions within the limits of values of temperature and humidity, allowed for the man. In sort that computers are sources of heat releases, there is a possibility of rising of temperature and descending of air humidity on work stations promoting a skin irritation. The microclimatic conditions in a room with a computer should meet the following requirements:

- Temperature of an environment in cold period of year 20 - 22°C, and 22 - 25°C in warm period;
- Relative humidity of air 30 - 60 %;
- The contents of a dust - max. 0.0001 kg/m at the dimension of particles max. 3 microns.

Another one of conditions of healthy and high-efficiency work is the provision of cleanness of air. Atmospheric air contains in its structure these in percentage terms:

- Nitrogen……. ….. 78,8 %
- Oxygen ……. …… 20,25 %
- Argon, neon and other inert gases …… 0,93 %
- Carbonic gas ……. 0,03 %

Air of such structure is most favorable for breath of the man.

Network equipment and the workstations, considered in the given degree work do not produce any harmful substances during their operation. Thus, aerial environment in a room where they work does not render harmful effects on human organism and meets the requirements of first category of works.

The optimum norms of temperature, relative humidity and rate of

movement of air in a working area of industrial rooms are normalized as given in the Table.

Normalized parameters of microclimate in industrial rooms

Year season	Category of works	Temperature, °C	Relative humidity, %	Rate of air movement, m/s
Cold period	I	22 - 24	40 – 60	0,1
Warm period	I	23 - 25	40 – 60	0,1

Industrial lighting

The lighting is one of the major factors influencing to the productivity of work. The rationally arranged lighting on work stations of operators is an essential metric of high level labor culture, integral part of scientific organization of work and aesthetics of production. The requirements to rational room lighting are reduced to the following:

- Correct choice of light sources and lighting systems;
- Creation of a necessary level of lighting of working surfaces;
- Limiting of blinding action of light, elimination of patches of reflected light;
- Provision of uniform lighting.

The acceptable level of lighting in a room can be found if we sequentially solve two tasks:

1. To determine a required level of lighting of operator’s work station by external light sources.

2. If the required level of lighting appears unacceptable for other operators working in considered room, it is necessary to find a way of saving of required contrast of the representation by other means. For example, it is possible to arrange light flow taking into account the location of workstations and means of displaying of information.

At designing and organization of computer operator's workstation it is necessary to undertake actions on preventing the direct and reflected patches of light. Direct patches of light occur as a result of presence of light sources directly in sight of the operator, reflected patches of light appear as a result of presence of reflecting surfaces inside of field of view. The direct patches of light can be reduced by any of the following ways: to apply reflected lighting; to use several light sources of smaller power instead of one of high power; to use means of screening of direct light from eyes of the operator.

The reflected patches of light can be reduced by the following ways:

- to use diffused light;
- to apply matted surfaces;
- to allocate direct light sources so that a visual angle of working square by the operator do not concur with an angle of incidence of light from source.

The important task is the choosing the sort of lighting (natural or artificial) and choosing a working room according to it (with windows or without windows). Natural lighting is most favorable for the working personnel. The productivity of work at natural lighting is higher than at artificial one. For sufficient natural lighting the square of windows should take not less than 1/3 part of the total square of walls. However, it is necessary to take into account that the application of natural lighting has many disadvantages: as a rule arrival of light from only one side, the space non-uniformity of illumination, etc. For elimination of these disadvantages is necessary to apply extras. The application of double light (combination of natural and artificial lighting) physiologically non-effectively and negatively affects the sight, promotes early tiredness.

Accordingly to standards, the value of lighting by luminescent lamps should not be below 300 lx in a horizontal plane for aggregate system of lighting. In view of visual operation of high accuracy the value of illuminance can be increased up to 1000 lx. Apart from the illuminance, color of coloring of a room and spectral characteristics of used light renders the large influence on the operator's activity. It is recommended, that the ceiling should reflect 80-90 %,

wall - 50-60 %, and floor - 15-30 % of light, falling on them. A room, where PC is located should be light and clean. The ceilings and walls are recommended to be colored tint. In rooms where the computer equipment is placed, the conditions fit to the given requirements must be created.

Technical measures of protection from electric shock

All technical measures can be conditionally divided into two groups. The technical protective measures of the first group provide protection of the personnel from electric shock in case of their touch to current-carrying parts. These include:

- The supervision over conditions of isolation of electro technical devices and sections of power supply network;
- Blocking and protective guards;
- Optimum arrangement of equipment, providing severance between current-carrying parts;
- Trouble signaling (light, acoustic), marking and preventive placards;
- Protection against junction of a high voltage to the side of a low voltage;
- Application of low (42 V and 12 V) voltages;
- Use of an individual protective insulating means.

The technical measures of the second group provide protection from electric shock at a touch to carcass of electro installation in case of breakdown of insulation of current-carrying parts, followings concern to them:

- Protective grounding;
- Protective zeroing;
- Protective cutoff (disconnection);
- Double isolation;
- Application of isolation transformers.

As a rule, the electro technical equipment has operational isolation, which should endure extreme mechanical, electric and thermal loads, which are possible under operational conditions.

The protective ground is a deliberate junction of noncurrent-carrying metal parts of electric equipment, lighting rods and dischargers with ground. The designation of protective grounding is to lower to safe value the voltage, which originates on noncurrent-carrying parts of electro installations in case of a fault to field at an insulation failure of conductors, carrying an operational current for the equipment.

Conclusion

Safety engineering is one of the most important parts of the finally work. Because of utilization given project must be safety and security. In general every human should learn rules of occupation and safety engineering. Here is given how to use equipment and said that hand shouldn't be bare when working with electricity devices.

4.3. Principles and technologies cleaner production

Industrial Environmental Management through Cleaner Production
Throughout the second half of the 20th century a growing World wide movement has attempted to change the way industry interacts with the environment. Governments and industry

A like have contributed to this movement. The focus has been toreduce environmental impacts from industry through changes in industrial behaviour and technology.

The background is a common recognition that human activities have contributed to the deterioration of the environment and the loss of natural resources. Many significant steps have been taken towards restoring the natural environment.

Still pollution of air, water and soil remains one of the largest environmental challenges facing today's world.

Over the period Industrial Environmental Management (IEM) practices have developed gradually by the evolution of strategies for mitigating the environmental problems. Practising IEM could be understood as walking in a staircase. The concepts and strategies for pollution abatement make up the steps.

Concepts higher up the staircase include the concepts below, and add additional elements of scope and complexity. The art and science of management expands as one moves up

the staircase. Below some of these steps – concepts and strategies – will be introduced. All of them are relevant. Some are however in themselves insufficient to solve the environmental problems of an industrial activity. For many of the more efficient strategies the problem is rather that they have not been fully used and implemented.

The relationship between these terms is one of subsets and supersets. The lower terms are part of the higher terms. The highest term, sustainable development, also includes other "staircases" of concepts such as social responsibility, natural resource management, and economic development, as well.

The staircase of industrial environmental management.

There are three types of concepts on the staircase. The macro-scale concepts of sustainable development and industrial ecology extend far beyond the firm and include relationships between companies, social institutions, the public and the environment in all its facets. The company-wide concepts of environmental management systems and cleaner production address all aspects of the firm's operations in a life cycle approach, from the use of natural resources via suppliers, production, marketing and product use to product disposal.

Pollution Control

In the past, pollution control was seen as the key to a cleaner environment. Pollution control refers to the measures taken to manage pollution after it has been generated. One example is the extensive investment in the building of sewage or wastewater treatment plants, both in industries and in municipalities. This took place in Western Europe typically during 1960s and 70s, while in Central and Eastern Europe it was not until after the systems change around 1990 that WWTPs were built on a significant scale. Another example is the installation of flue gas cleaning equipment, for instance different types of filters for separation of dust and particles from industrial flue gases produced by

incineration of oil and solvent wastes. Also here equipment for gas cleaning was being installed in Western Europe long before it was in Central and Eastern Europe. The operational concepts also include the strategies of waste minimisation and recycling. Waste minimisation includes both waste avoidance and waste utilisation. Waste avoidance refers to the actions taken by producers to avoid generating hazardous waste, while waste utilisation includes a variety of actions which make waste a useful input into the production processes.The overall concept of recycling can also be broken down into a number of subsets, with terms as reuse, recycling, and recovery. Reuse, or closed loop recycling, refers to the repeated use of a "waste" material in the production process. Recycling occurs when one producer is able to utilise the waste from another production process. Recovery refers to the extraction of certain components of a "waste" material for use in a production process.

Pollution Prevention and Cleaner Production

In recent decades we have witnessed a paradigm shift from pollution control to pollution prevention (sometimes referred to as P2). Pollution prevention is the use of materials, processes, or practices that reduce or eliminate the creation of pollutants or wastes at the source. It includes practices that diminish the use of hazardous materials, energy, water, or other resources, and practices that protect natural resources through conservation or more efficient use. Most recently, the concept of cleaner production (CP) has entered the global environmental arena. CP fits within P2's broader commitment towards the prevention, rather than the control, of pollution. Cleaner production refers to the continuous application of an integrated preventive environmental strategy to processes and products to reduce risks to humans and the environment. For production processes, cleaner production includes 1) conserving raw materials and energy, 2) eliminating toxic raw materials, and 3) reducing the quantity and toxicity of all emissions and wastes before they leave a process. For products, the strategy focuses on reducing impacts along the entire life cycle of the product, from raw material extraction to the ultimate disposal of the product. Cleaner

production is achieved by applying know-how, by improving technology, and by changing attitudes.

Eco-efficiency

The concept of eco-efficiency was introduced by the World Business Council for Sustainable Development, WBCSD, in 1992 and since then has been widely adopted. Many businesses in all continents have been pursuing ways of reducing their impact on the environment while continuing to grow and develop. According to the definition given by the WBCSD Eco-efficiency is a management philosophy that encourages business to search for environmental improvements that yield parallel economic benefits. This concept describes a vision for the production of economically valuable goods and services while reducing the ecological impacts of production. The reduction in ecological impacts translates into an increase in resource productivity, which in turn can create competitive advantage. Inother words eco-efficiency means producing more with less. However, the concepts of Eco-efficiency and Cleaner Production are almost synonymous. The slight difference between them is that Eco-efficiency starts from issues of economic efficiency which have positive environmental benefits, while Cleaner Production starts from issues of environmental efficiency which have positive economic benefits. Industrial Ecology Industrial ecology can be considered the “production” component of sustainable development. The most important aspect of industrial ecology is the idea of industry as a system in which there is no waste at any step because all “waste” is a resource for another part of the industry network. Individual firms participate in industrial ecology by considering how their activities fit into the larger industrial system. For example, they have to consider what other industries can use the company’s wastes as inputs, and how they can work with them. This concept is thus one of relationships and dynamics between companies. To make industrial ecology work, of course, requires conscious application of the lower level concepts on the staircase as well as a motivation to support sustainable development. Using the same definition approach as that of the Brundtland Commission industrial

ecology may be defined as follows: “Industrial ecology is the means by which humanity can deliberately and rationally approach and maintain a desirable carrying capacity, given continued economic, cultural and technological evolution. The concept requires that an industrial system be viewed not in isolation from its surrounding systems, but in concert with them. It is a system view in which one seeks to optimise the total materials cycle from virgin material, to finished material, to product, to waste product, and to ultimate disposal. Factors to be optimised include resources, energy and capital.”

In this definition, the emphasis on deliberate and rational differentiates the industrial-ecology path from unplanned, precipitous, and perhaps quite costly and disruptive alternatives. By the same token, desirable indicates the goal that industrial ecology practices will support a sustainable world with a high quality of life for all, as opposed to, for example, an alternative where population levels are controlled by famine. Industrial ecology is therefore a more realizable macro-scale goal for industrial enterprises. Eco-efficiency and industrial ecology approaches can be used as examples of wise management of raw materials-products-waste streams.

Conclusion

In today's world, in the age of IT there is a mass of interactive programs are being created in the purpose of easing people's life. Therefore, using ICT sphere would be advisable in order to develop email services.

The main goal of Bachelor's final work – "Creating email client service in order to work with emails without browsers".

Software was created in C# using .Net framework, and to have interactive interface added XAML.

To implement Bachelor's final work there some tasks that are required:

- Mastered skills in email servers.
- Analyzed and understood different types of email clients.
- Chosen the right programming language.
- Emails were protected.
- Created a comfortable interface of the program.
- The program was tested.
- Wrote a manual for users.

REFERENCE

1. The Decree of the President of the Republic of Uzbekistan on June 27, 2013 PP-1989 "On measures for further development of national information and communication system of the Republic of Uzbekistan".

2. Speech of President Islam Karimov in ceremonial meeting, devoted to 18 years anniversary of Constitution. December 7, 2010.

3. http://lifehacker.com/

4. Herbert Schildt. The complete reference C# 4.0. Publishing house – Williams 2011.

5. Andrew Troelsen. PRO C# 2010 and the .NET 4 Platform, fifth edition. Publishing house – Williams 2011.

6. Mathias Bartoll. Design pattern in C#. Department of Computer Science, Malardalen University Vasteras, Sweden 2004.

7. Karli Watson, Kristian Negal, John D.Reid. Beginning Visual Studio C# 2008. Publishing house Williams

8. http://msdn.com/

9. Joan Daemen, Vincent Rijmen The Design of Rijndael. November 26, 2001

10. RSA Laboratories. Publishing house - RSA Security Inc. 2000

11. http:// professorweb.ru

12. http:// homeandlearn.co.uk

13. http:// metanit.com

14. http:// csharp.net-tutorials.com

15. http:// w3schools.com

Appendix

```
namespace MailClient
{
   struct oMessage
   {
      public oMessage(string r, string s, string b,ListViewItem l)
      {
         recipients = r;
         subject = s;
         body = b;
         listItem = l;
      }
}
   public partial class MainForm : Form
   {
      Dictionary<string, Message> messages = new Dictionary<string, Message>();
      Pop3Client pop3Client;
      SmtpClient smtpClient = new SmtpClient();
      List<oMessage> outbox = new List<oMessage>();
      string inServer, inUser, inPass, outServer, outUser, outPass;
      int inPort, outPort;
      bool inSSL, outSSL, loadSaved = true;
      string result;
      public MainForm()
      {
         InitializeComponent();
         this.Icon = Properties.Resources.mail;
         pop3Client = new Pop3Client();
         lvwColumnSorter = new Comparer.ListViewColumnSorter();
         this.mailView.ListViewItemSorter = lvwColumnSorter;
      }
      delegate void AddMailToViewDelegate(ListViewItem item);
      void AddMailToView(ListViewItem item)
```

```
            {
                if (mailView.InvokeRequired)
                {
                    mailView.Invoke(new AddMailToViewDelegate(AddMailToView), new
object[] { item });
                }
                else
                {
                    mailView.Items.Add(item);
                }
            }
            delegate void ClearMailsDelegate();
            void ClearMails()
            {
                if (mailView.InvokeRequired)
                {
                    mailView.Invoke(new ClearMailsDelegate(ClearMails), new object[] { });
                }
                else
                {
                    messages.Clear();
                    mailView.Items.Clear();
                }
            }
            void SendMail(string recipients, string subject)
            {
                if (outServer != null)
                {
                    MailForm mailForm = new MailForm();
                    mailForm.setRecipients(recipients);
                    mailForm.setSubject(subject);
        }
        }
            private void backgroundSender_DoWork(object sender, DoWorkEventArgs e)
            {
```

```
        if (outbox.Count != 0)
        {
            int max = outbox.Count-1;
            for (int i = max; i>=0; i--)
            {
                //Create message
                MailMessage message = new MailMessage(
                    outUser,
                    outbox[i].recipients,
                    outbox[i].subject,
                    outbox[i].body
                    );
                message.IsBodyHtml = false;
                // Send message
                /*try
                {*/
                    smtpClient.Send(message);
                    DeleteMail(outbox[i].listItem);
                    outbox.RemoveAt(i);
                /*}
                catch (Exception err)
                {
                    MessageBox.Show(err.Message);
                }*/
            }
        }
    }
void AddMessage(Message message, bool unread, int group, int image)
    {
        // Add the message to the dictionary from the messageNumber to the Message
        messages.Add(message.Headers.MessageId, message);

        ListViewItem newItem = new ListViewItem();

        newItem.Text = message.Headers.From.ToString();
```

```
                newItem.SubItems.Add(message.Headers.Subject);
                newItem.SubItems.Add(message.Headers.DateSent.ToString("yyyy-MM-dd
        HH:mm:ss"));
                newItem.Tag = message.Headers.MessageId;
                newItem.Group = mailView.Groups[group];
                newItem.ImageIndex = image;
                if (unread)
                {
                   newItem.Font = new Font(newItem.Font, FontStyle.Bold);
                }

                if (!File.Exists(Application.UserAppDataPath + "\\Inbox\\" +
message.Headers.MessageId + ".eml"))
                {
                   message.Save(new FileInfo(Application.UserAppDataPath + "\\Inbox\\" +
message.Headers.MessageId + ".eml"));
                }
             }
             private void rijndaelToolStripMenuItem_Click(object sender, EventArgs e)
             {
                DecryptForm newForm = new DecryptForm();

                if (newForm.ShowDialog() == System.Windows.Forms.DialogResult.OK)
                {
                   using (RijndaelManaged myRijndael = new RijndaelManaged())
                   {
                      try
                      {
                         contentView.DocumentText =
Encryption.DecryptStringFromBytes(Convert.FromBase64String(contentView.DocumentText.
Replace("<br />", "\n").Replace("&#60;", "<").Replace("&#62;", ">")),
Encoding.ASCII.GetBytes(newForm.getKey()),
Encoding.ASCII.GetBytes("1234567890123456")).Replace("<", "&#60;").Replace(">",
"&#62;").Replace("\n", "<br />");
                      }
```

```
                    catch (Exception)
                    {
                        MessageBox.Show("FAIL!");
                    }
                }
            }
        }

    }
}
        private void sendButton_Click(object sender, EventArgs e)
        {
            if (recipientsTextbox.Text == "" | subjectTextbox.Text == "" |
messageTextbox.Text == "")
            {
                MessageBox.Show(this,"Something is
missing","Ops",MessageBoxButtons.OK);
            }
            else
            {
                if (rijndaelToolStripMenuItem.Checked)
                {
                    if (encryptionKey.Text.Length == 16 | encryptionKey.Text.Length == 24 |
encryptionKey.Text.Length == 32)
                    {
                        messageTextbox.Text =
Convert.ToBase64String(Encryption.EncryptStringToBytes(messageTextbox.Text,
Encoding.ASCII.GetBytes(encryptionKey.Text),
Encoding.ASCII.GetBytes("1234567890123456")));
                        this.DialogResult = System.Windows.Forms.DialogResult.OK;
                        this.Close();
                    }
                    else
                    {
```

```
                    MessageBox.Show(this, "Encryption key has to be 16, 24 or 32 chars in length.", "Ops", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
            else if (rSAToolStripMenuItem.Checked)
            {
                try {
                    RSACryptoServiceProvider RSA = new RSACryptoServiceProvider();
                    RSA.FromXmlString(encryptionKey.Text);
                    messageTextbox.Text = Convert.ToBase64String(RSA.Encrypt(Encoding.Unicode.GetBytes(messageTextbox.Text), false));
                    this.DialogResult = System.Windows.Forms.DialogResult.OK;
                    this.Close();
                }
                catch (Exception err)
                {
                    MessageBox.Show(err.Message);
                }
            }
            else
            {
                this.DialogResult = System.Windows.Forms.DialogResult.OK;
                this.Close();
            }
        }
    }
public string getInUser()
    {
        switch (accSelecter.Text)
        {
            case "Gmail":
            case "Hotmail":
                return accUser.Text;
            case "Other":
                return inUser.Text;
            default:
                return "";
```

```
    }
}

public string getInPass()
{
    switch (accSelecter.Text)
    {
        case "Gmail":
        case "Hotmail":
            return accPass.Text;
        case "Other":
            return inPass.Text;
        default:
            return "";
    }
}
```