Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Военный Факультет

Кафедра электронных вычислительных машин

Дисциплина: Операционные системы и системное программирование

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

«Программа обмена данных поверх ethernet(mac2mac)»

БГУИР КП 1-40 02 01 112 ПЗ

Студент: гр. 130501 Новиков Н. Д.

Руководитель: Басак Д. В.

Минск 2023

Учреждение образования

«Белорусский государственный университет информатики   
и радиоэлектроники»

Военный факультет

УТВЕРЖДАЮ

Заведующий кафедрой

*.*

(подпись)

2023 г.

ЗАДАНИЕ

по курсовому проектированию

Студенту *Новикову Никите Дмитриевичу*

1. Тема проектаПрограмма обмена данных поверх ethernet(mac2mac)

2. Срок сдачи студентом законченного проекта *24 мая 2023 г.*

3. Исходные данные к проекту *класс ChatServer котором находится реализация чата для сервера также есть класс с таким же функционалом и для клиента ChatClient, класс serverIPAdress который возвращает IP-адресс сервера и классы FileServer и ClientServer которые отвечают за обмен файлов.*

4. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

*1. Диаграмма классов*

*2. Схема алгоритма метода:*

private static void downloadFile(File file)

*3. Схема алгоритма метода:*

private List<String> getFileNamesOnServer()

4. Консультант по проекту *Басак Д. В*.

5. Дата выдачи задания *24 февраля 2023 г.*

6. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и трудоемкости отдельных этапов):

*разделы 1, 2 к 1 марта 2023 г. – 20 %;*

*разделы 3, 4 к 1 апреля 2023 г. – 30 %;*

*разделы 5, 6 ,7 к 1 мая 2023 г. – 30 %;*

*оформление пояснительной записки и графического материала к 14 мая 2023 г. – 20 %*

*Защита курсового проекта с 23 мая 2023 г. по 12 июня 2023 г.*

РУКОВОДИТЕЛЬ– –––––––––*Басак Д. В.*

Задание принял к исполнению –Х *Новиков Н.Д.*

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc121470076)

[1 ПОСТАНОВКА ЗАДАЧИ 7](#_Toc121470077)

[2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ 8](#_Toc121470079)

[3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ 10](#_Toc121470080)

[3.1 Вычеслительные сети 10](#_Toc121470080)

[3.2 Распределенные программы 10](#_Toc121470080)

[3.3 Линии связи 12](#_Toc121470080)

[4ОБЗОР МЕТОДОВ И АЛГОРИТМОВ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ 14](#_Toc121470086)

[5 ОПИСАНИЕ АЛГОРИТМОВ РЕШЕНИЯ ЗАДАЧИ 15](#_Toc121470086)

[5.1 Алгоритм по шагам 15](#_Toc121470087)

[5.2 Схемы алгоритмов 17](#_Toc121470087)

[6 ТЕСТИРОВАНИЕ 18](#_Toc121470088)

[7 СПИСОК ЛИТЕРАТУРЫ 26](#_Toc121470088)

[ЗАКЛЮЧЕНИЕ 27](#_Toc121470090)

[Приложение А 28](#_Toc121470091)

[Приложение Б 29](#_Toc121470092)

[Приложение В 30](#_Toc121470093)

[Приложение Г 31](#_Toc121470094)

[Приложение Д 32](#_Toc121470095)

# **ВВЕДЕНИЕ**

В современном информационном обществе обмен данных является неотъемлемой частью нашей повседневной жизни. Различные системы и устройства не только собирают и обрабатывают данные, но и передают их друг другу для совместного использования и синхронизации. Одним из наиболее распространенных и эффективных способов обмена данных является использование сети Ethernet.

Ethernet - это широко применяемая технология локальной вычислительной сети (LAN), предназначенная для передачи данных между компьютерами и другими сетевыми устройствами. Она обеспечивает надежную и быструю передачу данных с использованием физической среды связи, такой как медные или оптоволоконные кабели.

Целью данной курсовой работы является изучение и разработка программы обмена данных поверх Ethernet. Мы будем исследовать протоколы, стандарты и методы передачи данных, а также разработаем программное обеспечение, которое позволит устройствам обмениваться данными в сети Ethernet.

В ходе работы мы рассмотрим основные компоненты программы обмена данных по Ethernet, включая протоколы сетевого уровня, управление доступом к среде передачи, архитектуру клиент-сервер и механизмы передачи данных. Мы также рассмотрим вопросы безопасности и надежности обмена данных по Ethernet.

Кроме того, в работе будут рассмотрены современные технологии и тенденции в области обмена данными по Ethernet, такие как виртуализация сетей, облачные вычисления и Интернет вещей (IoT). Мы также проанализируем применение программы обмена данных по Ethernet в различных областях, включая промышленность, телекоммуникации и системы автоматизации.

В итоге нашей работы будет разработана и протестирована программа обмена данных по Ethernet, которая будет демонстрировать эффективность и надежность обмена данных в сети Ethernet. Эта программа может быть использована в различных приложениях и системах, где требуется быстрый и безопасный обмен данными между устройствами.

Результаты данной работы не только помогут расширить наше понимание принципов работы Ethernet и программного обмена данными, но и найдут применение в реальных сценариях, где требуется эффективный обмен информацией.

# **1 ПОСТАНОВКА ЗАДАЧИ**

Темой данного курсового проекта является «Программа обмена данных поверх ethernet(mac2mac)».

Данная сиcтема содержит программное обеспечение для персонального компьютера.

*Цели* курсового проектирования: овладеть практическими навыками проектирования и разработки законченного, отлаженного и протестированного программного продукта с использованием языка высокого уровня в моем случае Java;

*Задачами* курсового проекта (работы) как этапа подготовки к дипломному проектированию являются:

– освоение, углубление и обобщение знаний, полученных студентами в процессе обучения;

– приобретение практических навыков и развитие творческих подходов к решению конкретной инженерной или инженерно-экономической задачи;

– формирование умений использовать справочную литературу, нормативную, правовую, нормативно-техническую документацию, осуществлять патентный поиск;

– приобретение навыков по оформлению текстовой и графической документации согласно требованиям государственных стандартов и стандарта предприятия Дипломные проекты (работы).

# 

# **2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

**Java** - это высокоуровневый, объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems (в настоящее время принадлежит Oracle Corporation).

**IP адрес** — это уникальный адрес (идентификатор) устройства в глобальной сети интернет или в локальной сети, в переводе на русский означает интернет протокол. Он используется для адресации и передачи данных по сети, без него устройство не могло бы определить куда именно стоит передавать данные. Каждому устройству, работающему по сети (телефон, компьютер, сетевой принтер, сервер и т.д.), необходим свой сетевой адрес. На сегодняшний день существует два вида IP адресов, IPv4 и IPv6.

**IPv4** — это 32-битное число, всего таких адресов существует 4,22 миллиарда. Это максимальное число, которое может сохраниться в типе данных INT (integer).

**IPv6** — это уже 128-битное число и в отличии от IPv4, таких адресов настолько много, что их явно хватит на всех, 3.4×10^38 (10 в 38 степени), попробуйте прикинуть цифру.[1]

# **Со́кет** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *socket* — разъём) — название [программного интерфейса](https://ru.wikipedia.org/wiki/API) для обеспечения обмена данными между [процессами](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)). Процессы при таком обмене могут исполняться как на одной [ЭВМ](https://ru.wikipedia.org/wiki/%D0%AD%D0%92%D0%9C), так и на различных ЭВМ, связанных между собой только [сетью](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C). Сокет — [абстрактный](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%86%D0%B8%D1%8F) объект, представляющий конечную точку соединения.[2]

**Ethernet** — семейство технологий пакетной передачи данных между устройствами для компьютерных и промышленных сетей.

**Протокол TCP/IP** – это целая сетевая модель, описывающая способ передачи данных в цифровом виде. На правилах, включенных в нее, базируется работа интернета и локальных сетей независимо от их назначения и структуры.

**Протокол UDP** - это протокол транспортного уровня в сетевой модели TCP/IP. UDP является простым протоколом без установления соединения, предоставляющим ненадежную доставку данных между устройствами в сети.

**Процесс** — это совокупность кода и данных, разделяющих общее виртуальное адресное пространство. Чаще всего одна программа состоит из одного процесса, но бывают и исключения (например, браузер Chrome создает отдельный процесс для каждой вкладки, что дает ему некоторые преимущества, вроде независимости вкладок друг от друга). Процессы изолированы друг от друга, поэтому прямой доступ к памяти чужого процесса невозможен (взаимодействие между процессами осуществляется с помощью специальных средств).

**Поток** – это одна единица исполнения кода. Каждый поток последовательно выполняет инструкции процесса, которому он принадлежит, параллельно с другими потоками этого процесса.

# **3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ**

### **3.1** **Вычислительные сети**

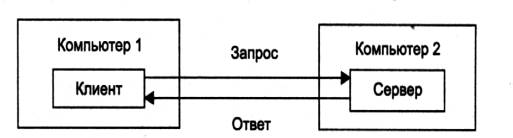
В вычислительных сетях программные и аппаратные связи являются еще более слабыми, а автономность обрабатывающих блоков проявляется в наибольшей степени - основными элементами сети являются стандартные компьютеры, не имеющие ни общих блоков памяти, ни общих периферийных устройств. Связь между компьютерами осуществляется с помощью специальных периферийных устройств - сетевых адаптеров, соединенных относительно протяженными каналами связи. Каждый компьютер работает под управлением собственной операционной системы, а какая-либо «общая» операционная система, распределяющая работу между компьютерами сети, отсутствует. Взаимодействие между компьютерами сети происходит за счет передачи сообщений через сетевые адаптеры и каналы связи. С помощью этих сообщений один компьютер обычно запрашивает доступ к локальным ресурсам другого компьютера. Такими ресурсами могут быть как данные, хранящиеся на диске, так и разнообразные периферийные устройства - принтеры, модемы, факс-аппараты и т. д. Разделение локальных ресурсов каждого компьютера между всеми пользователями сети - основная цель создания вычислительной сети.

Каким же образом сказывается на пользователе тот факт, что его компьютер подключен к сети? Прежде всего, он может пользоваться не только файлами, дисками, принтерами и другими ресурсами своего компьютера, но аналогичными ресурсами других компьютеров, подключенных к той же сети. Правда, для этого недостаточно снабдить компьютеры сетевыми адаптерами и соединить их кабельной системой. Необходимы еще некоторые добавления к операционным системам этих компьютеров. На тех компьютерах, ресурсы которых должны быть доступны всем пользователям сети, необходимо добавить модули, которые постоянно будут находиться в режиме ожидания запросов, поступающих по сети от других компьютеров. Обычно такие модули называются программными серверами (server), так как их главная задача - обслуживать (serve) запросы на доступ к ресурсам своего компьютера. На компьютерах, пользователи которых хотят получать доступ к ресурсам других компьютеров, также нужно добавить к операционной системе некоторые специальные программные модули, которые должны вырабатывать запросы на доступ к удаленным ресурсам и передавать их по сети на нужный компьютер. Такие модули обычно называют программными клиентами (client). Собственно же сетевые адаптеры и каналы связи решают в сети достаточно простую задачу - они передают сообщения с запросами и ответами от одного компьютера к другому, а основную работу по организации совместного использования ресурсов выполняют клиентские и серверные части операционных систем.

Пара модулей «клиент - сервер» обеспечивает совместный доступ пользователей к определенному типу ресурсов, например к файлам. В этом случае говорят, что пользователь имеет дело с файловой службой (service). Обычно сетевая операционная система поддерживает несколько видов сетевых служб для своих пользователей - файловую службу, службу печати, службу электронной почты, службу удаленного доступа и т. п.

#### **3.2** **Распределенные программы**

Сетевые службы всегда представляют собой распределенные программы. Распределенная программа - это программа, которая состоит из нескольких взаимодействующих частей (в приведенном на рис. 1.5 примере - из двух), причем каждая часть, как правило, выполняется на отдельном компьютере сети.



**Рисунок 3.1.** Взаимодействие частей распределенного приложения

До сих пор речь шла о системных распределенных программах. Однако в сети могут выполняться и распределенные пользовательские программы - приложения. Распределенное приложение также состоит из нескольких частей, каждая и которых выполняет какую-то определенную законченную работу по решению прикладной задачи. Например, одна часть приложения, выполняющаяся на компьютере пользователя, может поддерживать специализированный графический интерфейс вторая - работать на мощном выделенном компьютере и заниматься статистической обработкой введенных пользователем данных, а третья - заносить полученные результаты в базу данных на компьютере с установленной стандартной СУБД. Распределенные приложения в полной мере используют потенциальные возможности распределенной обработки, предоставляемые вычислительной сетью, и поэтому часто называются сетевыми приложениями.

Следует подчеркнуть, что не всякое приложение, выполняемое в сети, является сетевым. Существует большое количество популярных приложений, которые не являются распределенными и целиком выполняются на одном компьютере сети. Тем не менее и такие приложения могут использовать преимущества сети за счет встроенных в операционную систему сетевых служб. Значительная часть истории локальных сетей связана как раз с использованием таких нераспределенных приложений. Рассмотрим, например, как происходила работа пользователя с известной в свое время СУБД dBase. Обычно файлы базы данных, с которыми работали все пользователи сети, располагались на файловом сервере. Сама же СУБД хранилась на каждом клиентском компьютере в виде единого программного модуля.

Программа dBase была рассчитана на обработку только локальных данных, то есть данных, расположенных на том же компьютере, что и сама программа. Пользователь запускал dBase на своем компьютере, и она искала данные на локальном диске, совершенно не принимая во внимание существование сети. Чтобы обрабатывать с помощью dBase данные на удаленном компьютере, пользователь обращался к услугам файловой службы, которая доставляла данные с сервера на клиентский компьютер и создавала для СУБД эффект их локального хранения.

Большинство приложений, используемых в локальных сетях в середине 80-х годов, являлись обычными, нераспределенными приложениями. И это понятно - они были написаны для автономных компьютеров, а потом просто были перенесены в сетевую среду. Создание же распределенных приложений, хотя и сулило много преимуществ (уменьшение сетевого трафика, специализация компьютеров), оказалось делом совсем не простым. Нужно было решать множество дополнительных проблем - на сколько частей разбить приложение, какие функции возложить на каждую часть, как организовать взаимодействие этих частей, чтобы в случае сбоев и отказов оставшиеся части корректно завершали работу, и т. д., и т. п. Поэтому до сих пор только небольшая часть приложений является распределенными, хотя очевидно, что именно за этим классом приложений будущее, так как они в полной мере могут использовать потенциальные возможности сетей по распараллеливанию вычислений.

## **3.3. Линии связи**

Линия связи (рис. 2.1) состоит в общем случае из физической среды, по которой передаются электрические информационные сигналы, аппаратуры передачи данных и промежуточной аппаратуры. Синонимом термина *линия связи (line)* является термин *канал связи(channel)*.



Рисунок 3.2. Состав линии связи

Скрученная пара проводов называется *витой парой (twisted pair)*. Витая пара существует в экранированном варианте *(Shielded Twistedpair, STP),* когда пара медных проводов обертывается в изоляционный экран, и неэкранированном *(Unshielded Twistedpair, UTP)*, когда изоляционная обертка отсутствует. Скручивание проводов снижает влияние внешних помех на полезные сигналы, передаваемые по кабелю. *Коаксиальный кабель (coaxial)* имеет несимметричную конструкцию и состоит из внутренней медной жилы и оплетки, отделенной от жилы слоем изоляции. Существует несколько типов коаксиального кабеля, отличающихся характеристиками и областями применения - для локальных сетей, для глобальных сетей, для кабельного телевидения и т. п. *Волоконно-оптический кабель (optical fiber)* состоит из тонких (5-60 микрон) волокон, по которым распространяются световые сигналы. Это наиболее качественный тип кабеля - он обеспечивает передачу данных с очень высокой скоростью (до 10 Гбит/с и выше) и к тому же лучше других типов передающей среды обеспечивает защиту данных от внешних помех.

# **4 ОБЗОР МЕТОДОВ И АЛГОРИТМОВ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ**

Поскольку в данной работе используется объектно-ориентрованный подход программирования, то ниже приведены некоторые из классов, использованных в программе (диаграмма классов представлена в приложении А):

1. Класс ChatClient для реализации клиента-чата:

Методы:

-Chat()– основной метод программы который реализует подключение к серверу и обмен сообщениями.

-run()–создание потока для чтения сообщения из сервера.

1. Класс ChatServer для реализации сервера-чата:

Методы:

-ChatServ()– основной метод программы который реализует подключение к клиенту и обмен сообщениями.

-run()–создание потока для чтения сообщения из клиента.

1. Класс ServerIPAdress для нахождения ip-адреса сервера:

Методы:

-IpAddress()– метод который реализует нахождение ip-адреса сервераю

1. Класс FileClient для обмена данными между клиентом и сервером:

Методы:

- createAndShowGUI()– метод который реализует интерфейс клиента для обмена данными.

- uploadFile() – метод загрузки файла на сервер.

- downloadFile () – метод скачки файла из сервера.

- viewFilesOnServer () – метод просмотра файлов на сервере.

1. Класс FileServer для обмена данными между клиентом и сервером:

Методы:

- getFileNamesOnServer()– метод который возвращает имена всех файлов находящихся на сервере.

- Accept() () –установка соединения между клиентом и сервером.

- run () – обработка действий отправленных клиентом серверу.

# **5 ОПИСАНИЕ АЛГОРИТМОВ РЕШЕНИЯ ЗАДАЧИ**

# **5.1 Алгоритмы по шагам**

Алгоритм метода downloadFile()

private static void downloadFile(File file)

1. Начало
2. Входные данные
   1. File file – файл.
3. Socket socket = new Socket(serverIp, serverPort); - установка соединения с сервером.
4. OutputStream outputStream = socket.getOutputStream(); - получаем выходной поток для отправки данных на сервер.
5. dataOutputStream.writeUTF("DOWNLOAD"); - отправляем команду серверу.
6. dataOutputStream.writeUTF(file.getName()); - отправляем имя файла для скачивания.
7. InputStream inputStream = socket.getInputStream(); - получаем входной поток с сервера
8. Условие receivedFileName - Проверяется, что переменная "receivedFileName" не пуста, то есть было успешно получено имя файла от сервера.
9. File savedFile = new File(SAVE\_FOLDER, receivedFileName); - Создается новый объект "File" с указанным именем файла и путем сохранения. В данном случае, папка для сохранения файлов предварительно указана в переменной "SAVE\_FOLDER".
10. Цикл while c условием((bytesRead = dataInputStream.read(buffer)) != -1) - Цикл while выполняется до тех пор, пока данные считываются из "dataInputStream" (предполагается, что это поток ввода данных от сервера) и не достигнут конец файла (-1). В каждой итерации цикла считанные данные записываются в "fileOutputStream"
11. Конец цикла while
12. System.out.println("Файл сохранен на клиенте: " + savedFile.getAbsolutePath()); - информацио о успешном сохранении файла
13. fileOutputStream.close(); - закрываем потоки.
14. Конец

Алгоритм поиска getFileNamesOnServer ()

private List<String> getFileNamesOnServer()

1. Начало
2. List<String> fileNames = new ArrayList<>();Node<T> \*\_rab = beg – Создается новый объект "ArrayList<String>" под названием "fileNames", который будет использоваться для хранения имен файлов на сервере.
3. File folder = new File(SAVE\_FOLDER); - Создается объект "File" с указанным путем к папке сохранения файлов на сервере. Затем вызывается метод "listFiles()" для получения массива файлов в данной папке.
4. Условие if(files != null) – если условие true то прохожим дальше по методу.
5. for (File file : files) - Обход файлов в папке:
6. Выполняется цикл "for-each" для каждого файла в массиве "files". Для каждого файла выполняются следующие шаги:
7. Проверка, что элемент массива является файлом:
8. Выполняется проверка, что текущий элемент "file" является файлом, а не папкой.
9. Добавление имени файла в список:
10. Если текущий элемент является файлом, то его имя добавляется в список "fileNames" с помощью метода "getName()", который возвращает имя файла.rab = rab->next; - переносим указатель на след. значение.
11. Конец цикла while
12. Возврат списка имен файлов:
13. Конец

# **5.2 Схемы алгоритмов**

Схема алгоритма метода downloadFile()

private static void downloadFile(File file)

приведена в приложении Б – метод установки файлов клиентом с сервера.

Схема алгоритма метода getFileNamesOnServer ()

private List<String> getFileNamesOnServer()

приведена в приложении В – метод получения клиентом имен файлов на сервере.

# **6 ТЕСТИРОВАНИЕ**

При запуске программы открывается окно

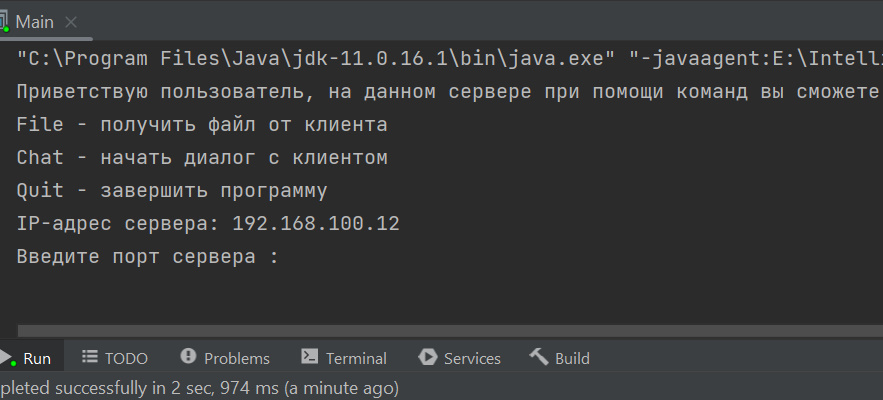


Рисунок 5.1 – Главное меню Сервера

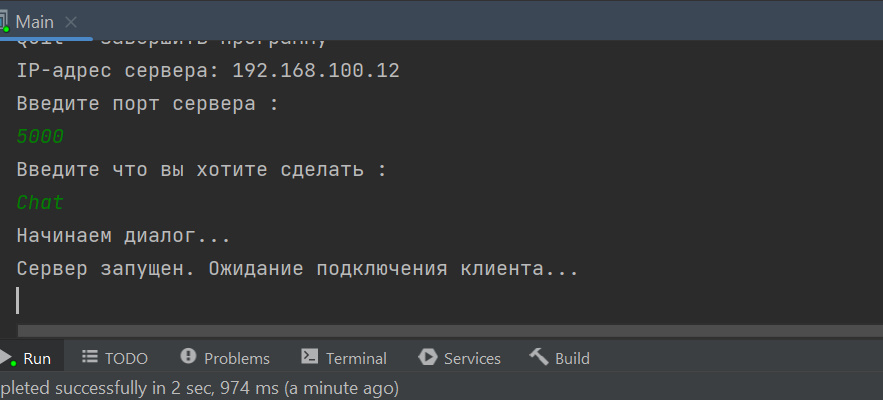


Рисунок 5.2 – Главное меню Клиента

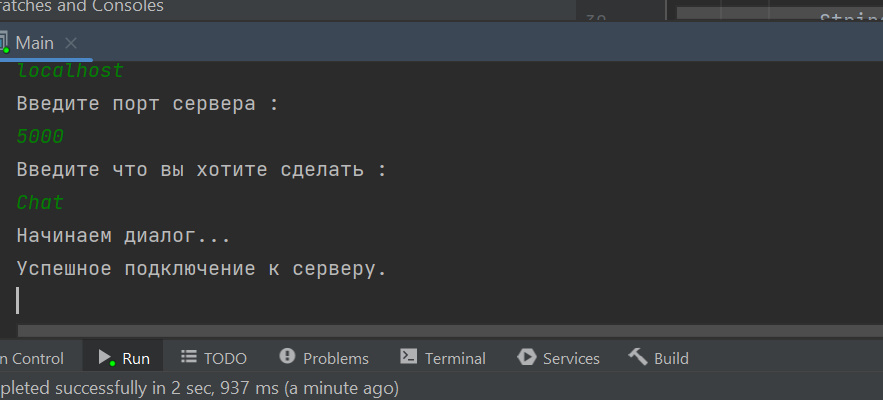


Рисунок 4.3 –Выбор операций Chat

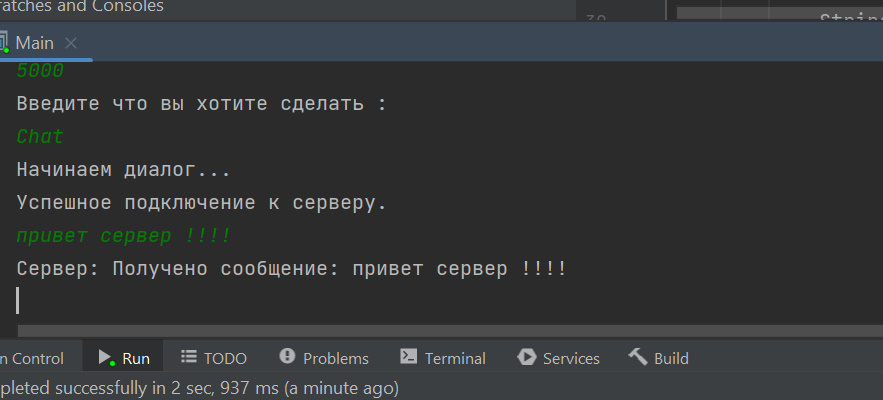


Рисунок 4.4 –Диалог между клиентом и сервером вид клиента

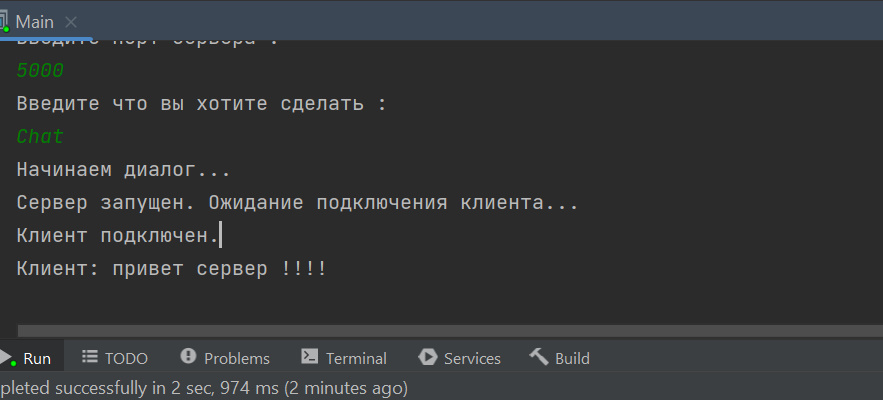


Рисунок 4.5 – Диалог между клиентом и сервером вид сервера

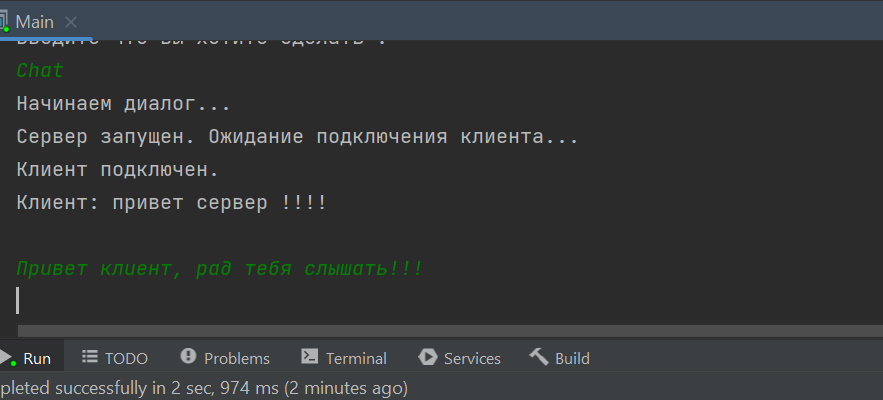


Рисунок 4.6 – Диалог между клиентом и сервером вид сервера

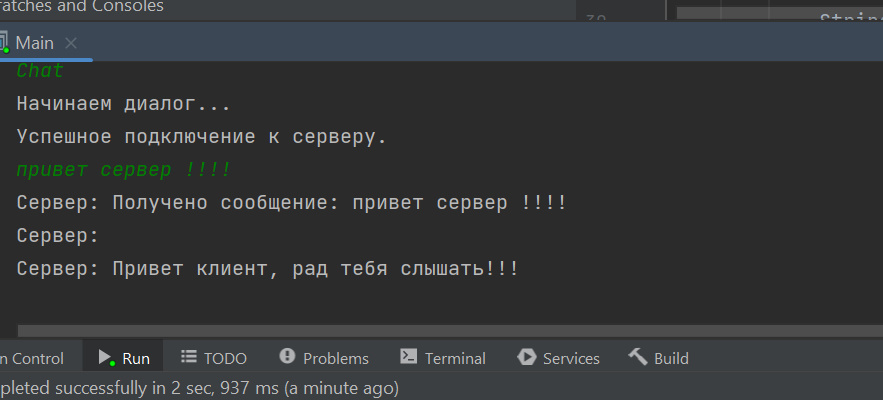


Рисунок 4.7 – Диалог между клиентом и сервером вид клиента

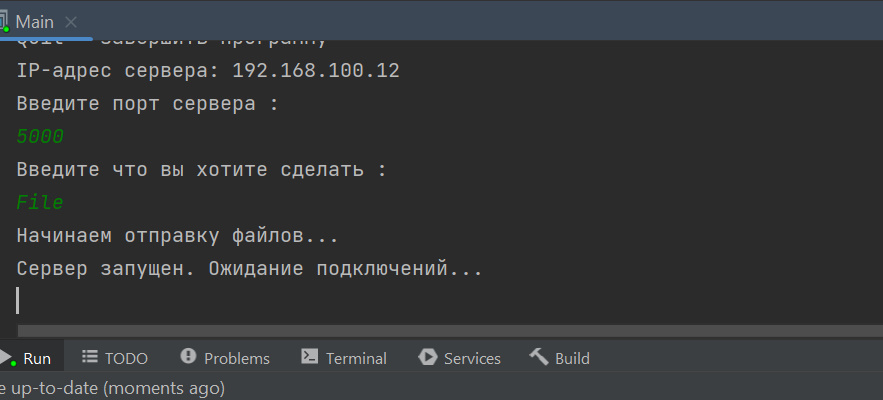


Рисунок 4.8 – Выбор операций File

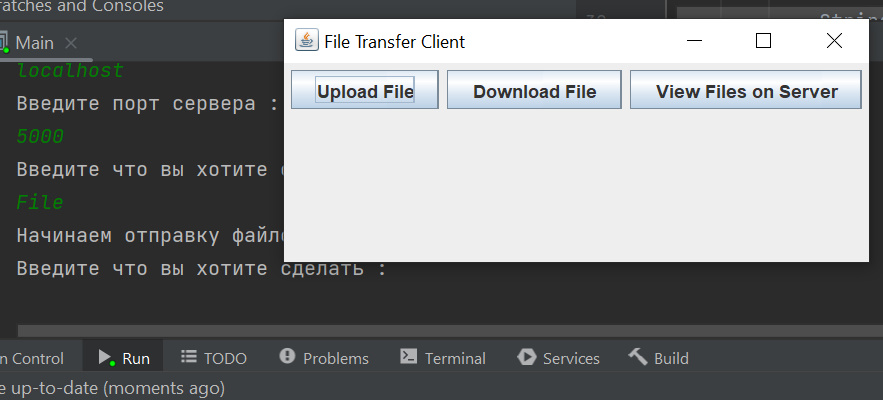


Рисунок 4.9 –Меню с 3-мя кнопками

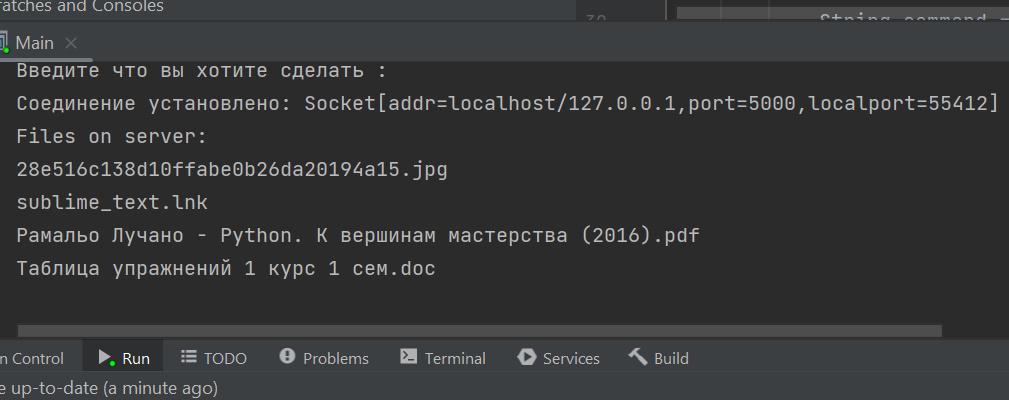


Рисунок 4.10 – кнопка View Files on Server

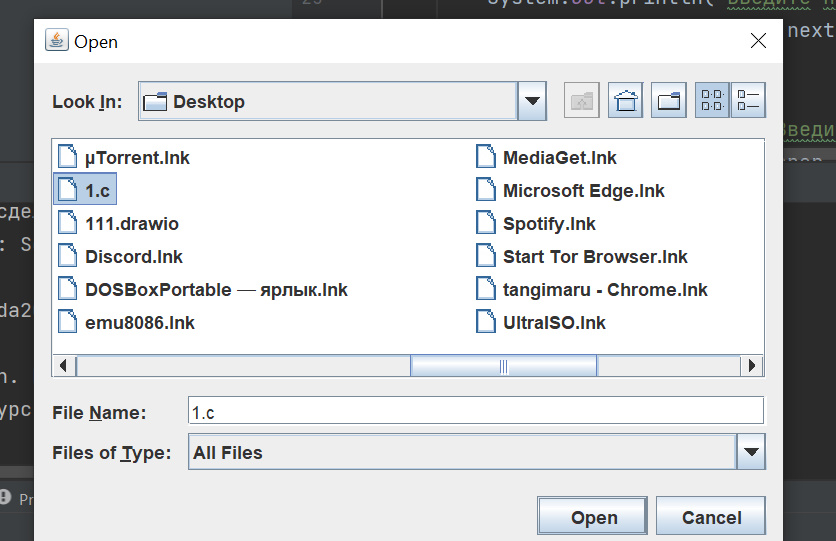


Рисунок 4.11 – Отправка файлов на сервер

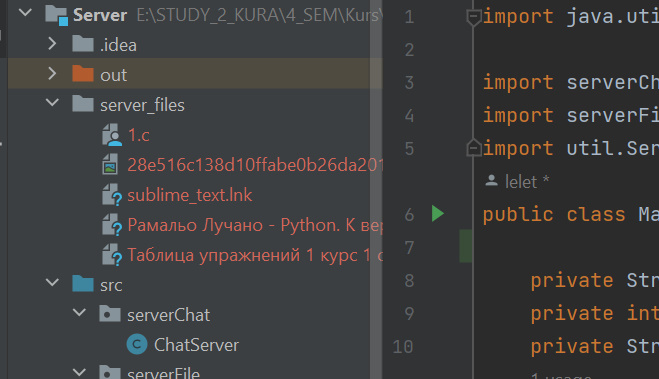


Рисунок 4.12 –Сервер принял файл

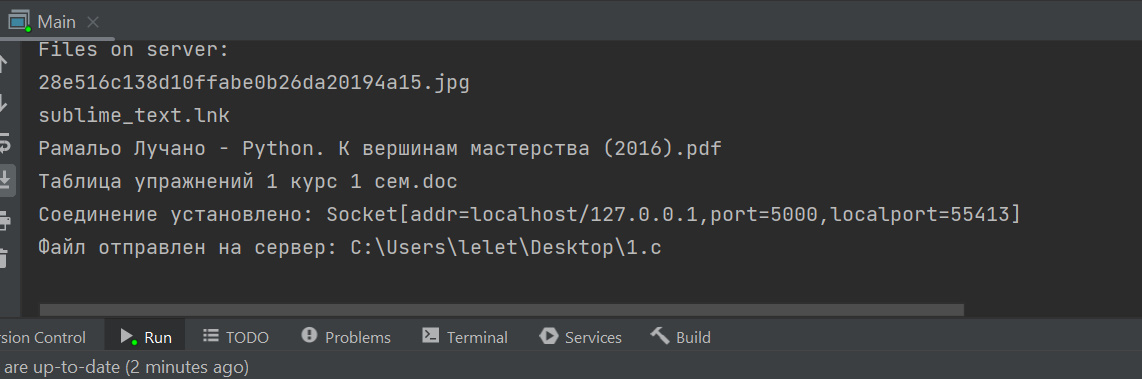


Рисунок 4.13 –Вывод консоли о передачи файла

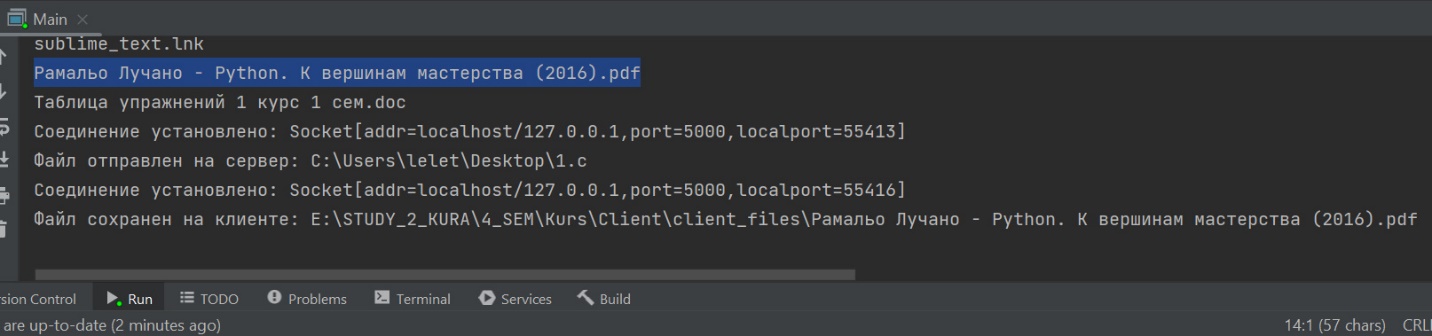


Рисунок 4.14.1 –Скачиваем файлы с сервера

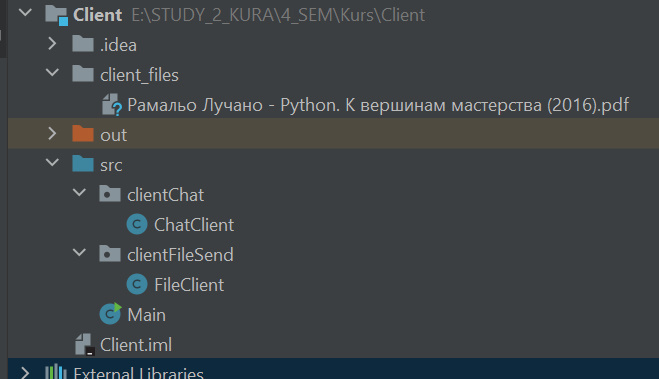


Рисунок 4.14.1 – Клиент успешно скачал файлы с сервера

# 

# **7. СПИСОК ЛИТЕРАТУРЫ**

[1].Что такое IP - адрес [Электронный ресурс]. – Режим доступа <https://hostgid.net/baza-znanii/hosting/chto-takoe-ip-adres.html>. – Дата доступа 10.05.2023.

[2].Сокеты [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Сокет\_(программный\_интерфейс) – Дата доступа 10.05.2023.

[3].Технологии программирования [Электронный ресурс]. – Режим доступа: [http://info-master.su/programming/profi/programming-technologies.php – Дата доступа 12.05.2023](http://info-master.su/programming/profi/programming-technologies.php%20–%20Дата%20доступа%2012.05.2023).

[4].Структура программного обеспечения ПК [Электронный ресурс]. – Режим доступа: [https://studopedia.ru/4\_4312\_tema--struktura-programmnogo-obespecheniya-pk.html – Дата доступа 12.05.2023](https://studopedia.ru/4_4312_tema--struktura-programmnogo-obespecheniya-pk.html%20–%20Дата%20доступа%2012.05.2023).

[5].Руководство по разработке на компонентах [Электронный ресурс]. – Режим доступа: https://habr.com/ru/company/ruvds/blog/461661/ – Дата доступа 15.05.2023.

[6]. TCP/IP Sockets in Java, Second Edition: Practical Guide for Programmers, Kenneth L. Calvert, Michael J. Donahoo– Morgan Kaufmann, 2008. – 192 с.:ил.

[7]. Java Network Programming: Developing Networked Applications (2013), Elliotte Rusty Harold – O'Reilly Media, 2013. - 762 c.:ил.

# **ЗАКЛЮЧЕНИЕ**

В результате выполнения курсового проекта была разработано программное средство для обмена данных поверх ethernet. Во время реализации данного проекта были использован язык программирования Java. При разработке программы стало понятно, что классы и объекты в Java являются основными концепциями объектно-ориентированного программирования — ООП. А объектно-ориентированное программирование — расширение структурного программирования, в котором основными концепциями являются понятия классов и объектов. В программе удалось реализовать удобный пользовательский интерфейс, хороший дизайн. Программа проста в использовании и имеет большой функционал, который помогает при работе с обменом данных поверх ethernet.

**Приложение А**

*(обязательное)*

Диаграмма классов

**Приложение А**

*(обязательное)*

Диаграмма классов

**Приложение Б**

*(обязательное)*

Схема алгоритма метода downloadFile (File file)

private static void downloadFile(File file)

**Приложение В**

*(обязательное)*

Схема алгоритма метода getFileNamesOnServer()

private List<String> getFileNamesOnServer()

**Приложение Г**

(*обязательное*)

Код программы

**Приложение Д**

*(обязательное)*

Ведомость документов