Geekbrains

**Разработка консольного приложения для перемещения файлов на виртуальную машину, находящеюся вне локальной сети, по “FTP” протоколам**

Программа:

Разработчик - Веб-разработка на Java

Программист

Юрьев Михаил Денисович

Красноярск2024

**Содержание**

[Содержание](#_heading=h.7mt9x4b37ce)

[Введение](#_heading=h.1fob9te)

[Теоретическая и практическая главы](#_heading=h.hst8x8f1r8tz)

[Заключение](#_heading=h.dhwmmf3wq1kk)

[Список используемой литературы](#_heading=h.m8jvbwhv7ujg)

[Приложения](#_heading=h.s6k7t2lwcp0k)

Глава 1. Основы разработки консольного приложения.

1.1 Что такое консольное приложение, его особенности.

1.2 Что такое программирование, кто такой программист. Зачем нужно программирование и когда оно начинается.

1.3 Этапы разработки консольного приложения.

1.4 Источники требований: что такое техническое задание и как разрабатывать программный код, если техническое задание отсутствует.

Глава 2. Создание виртуальной Linux машины с FTP протоколами.

2.1 Установка инструментов для соз-2дания виртуальной Linux машины.

2.2 Создание виртуальной машины.

2.3 Настройка виртуальной машины.

2.4 Подключение SSH протоколов.

2.5 Проверка результата на основной машине.

Глава 3. Проверка работоспособности программного кода и подведение итогов.

Заключение.

Список используемой литературы.

Приложения.

# Введение

Проект представляет из себя программный код в купе с виртуальным компьютером на ОС Linux с дистрибутивом Ubuntu на котором подключён FTP протокол.

**Тема проекта:** Разработка консольного приложения для переноса файлов или их копий и их хранением на виртуальном компьютере с операционной системой Linux при помощи протоколов FTP.

На рабочих компьютеров часто хранятся важные файлы различного характера, возможность хранить копии или сами эти файлы на сервере безопаснее и практичнее из-за большей безопасности и труднодоступности из вне.

**Цель:** изучить и разработать консольное приложение обладающие функционалом для переноса файлов или их копий на виртуальный компьютер с операционной системой Linux на дистрибутиве Ubuntu.

**Какую проблему решает:** Отсутствия резервных копий нужных файлов и простого доступа к ним при отсутствии возможности или желания хранить н крупных и платных сервисах

**Задачи:**

1. Изучить литературу, касающуюся темы исследования.
2. Рассмотреть основные виды и методы разработки консольного приложения.
3. Разработать программный код
4. Поднять виртуальный компьютер на операционной системе Linux с дистрибутивом Ubuntu.
5. Изучить методы работы и установки FTP протоколов на компьютер с ОС Linux Ubuntu.
6. Прописать FTP протоколы внутри виртуального компьютера.
7. Объединить виртуальный компьютер и основной по протоколам FTP.
8. Дописать в программный код функцию переноса файлов.
9. Произвести проверочный перенос файлов.

**Инструменты:** Intellij IDEA — это интегрированная среда разработки (IDE) от компании JetBrains, предназначенная для разработки на многих языках программирования, в частности Java, JavaScript, Python.

Hyper-V — это платформа аппаратной виртуализации от компании Microsoft, которая позволяет создавать и запускать независимые изолированные виртуальные машины

PuTTY — это бесплатный клиент для различных протоколов удалённого доступа, включая SSH, Telnet, rlogin.

FARManager - консольный файловый менеджер для операционных систем семейства Microsoft Windows и Linux.

VSFTPD — FTP-сервер для UNIX-подобных систем, включая Linux.

**Состав команды**: Юрьев Михаил Денисович (разработчик)

**Глава 1. Основы разработки консольного приложения.**

Консольное приложение разрабатывается путём вывода возможного функционала в консоль (терминал) и последующей реализации работы предложенных функций воспринимая данные с устройства ввода

Для разработки этого проекта и главы использовался материал, данный преподавателями во время лекци и семинаров, а так же материалы общедоступной многоязычной универсальной интернет-энциклопедия со свободным контентом — “Википедия”, и ответы “ЯндексНейро” основаные на основе различных интернет ресурсов (по заявлению самого “ЯндексНейро”)

Основные методы разработки консольного приложения:

* Настройка среды разработки
* Работа с вводом и выводом
* Использование специальных сред разработки

Настройка среды разработки — это включает установку компилятора и выбор текстового редактора или интегрированной среды разработки (IDE). Правильная настройка упростит процесс написания и отладки кода.

Работа с вводом и выводом — для взаимодействия с пользователем в консольных приложениях используются функции ввода-вывода. Основные из них — это printf для вывода данных и scanf для ввода данных. Эти функции позволяют создавать интерактивные программы, которые могут запрашивать у пользователя данные и выводить результаты.

Использование специальных сред разработки — например, для разработки на языке Java можно использовать NetBeans IDE, Eclipse IDE, IntelliJ IDEA и другие среды. 3 Они предоставляют широкий набор инструментов для разработки, включая отладчики, профилировщики и интеграцию с системами контроля версий.

**1.1 Что такое консольное приложение, его особенности.**

Консольное приложение — это программа, которая работает с командной строкой.

Для взаимодействия с пользователем консольное приложение использует консоль — клавиатуру и монитор, работающий в режиме отображения символьной информации (буквы, цифры и специальные знаки). В консольном приложении зачастую не используется графический функционал, и программа представляет собой окно терминала или самой консоли на котором есть список функций и инструкция (в случае необходимости) доступа к ним. Пользователь использует эти функции путём ввода специальных команд указаных инструкциями в специальную строку или чего-то подобного предусмотренного внутри приложения.

В этом проекте используется встроенная консоль Intellij IDEA, из-за удобства проверки кода и быстро доступа, а также из-за её простой синхронизации со встроенным фреймворком Apache Maven.

Apache Maven — это фреймворк для автоматизации сборки проектов на основе описания их структуры в файлах POM (Project Object Model) на языке XML.

Проект Maven издаётся сообществом Apache Software Foundation, где формально является частью Jakarta Project.

Фреймворк (с англ. framework — «каркас, структура») — это набор правил, шаблонов и инструментов, которые используются для построения продуктов или процессов как в программировании, так и в других областях. Он задаёт структуру, определяет правила и предоставляет необходимый набор инструментов для создания проекта.

**1.2 Что такое программирование, кто такой программист. Зачем нужно программирование и когда оно начинается.**

Программирование — это процесс создания компьютерных программ или программного обеспечения, а также дисциплина, изучающая методы и приёмы создания и развития компьютерных программ.

Процесс программирования включает следующие этапы:

* Анализ
* Проектирование программы
* Кодирование
* Выявление дефектов и ошибок
* Развёртывание
* Сопровождение

Анализ предметной области. Выделение требований к программе и точная постановка решаемых задач.

Проектирование программы. Выделение компонентов или модулей на основе отдельных решаемых задач, определение способов взаимодействий между ними, определение алгоритмов работы и структур данных, используемых каждым модулем.

Кодирование. Написание отдельных модулей на определённых языках программирования.

Выявление дефектов и ошибок. Проверка программы на наборе заранее выбранных сценариев, позволяющем оценить её корректность, а также отладка программы.

Развёртывание. Размещение программы в её рабочем окружении, настройка и подготовка её к работе, обучение пользователей работе с программой.

Сопровождение. Поддержка работоспособности, настройка под изменяющееся окружение, выявление дефектов и новых задач, внесение исправлений и изменений.

Анализ предметной области в этом проекте составлял личный опыт и субъективное мнение, сложившееся в ходе обучения и выполнения семинарских и лекционных задач. Это мнение выражает то, что в большинстве случаев файлы или альтернативные ресурсы, создаваемые или изменяемые во время занятий, не имеют резервной копии и иногда возвращаться исключительно к базовым параметрам.

В этом проекте программа проектировалась на основе принципов SOLID:

SOLID — это аббревиатура от названий принципов объектно-ориентированного программирования (ООП), которые помогают разработчикам писать поддерживаемый и масштабируемый код.

Акроним SOLID расшифровывается так:

S — Single Responsibility Principle («Принцип единственной ответственности»);

O — Open/Closed Principle («Принцип открытости/закрытости»);

L — Liskov Substitution Principle («Принцип подстановки Барбары Лисков»);

I — Interface Segregation Principle («Принцип разделения интерфейса»);

D — Dependency Inversion Principle («Принцип инверсии зависимостей»).

SOLID-принципы разработал Роберт Мартин (также известный как Дядя Боб), американский инженер, программист и автор известной книги «Чистый код».

Весь код этого проекта написан и реализован на языке программирования Java из-за углублённого изучения именно этого языка программирования во время обучения на курсе после деления на специализации.

Во время написания этого проекта или любого другого невозможно без выявления ошибок, из-за того, что если не заниматься проверкой работоспособности кода во время его написания, в конечном итоге может появиться огромное количество ошибок выполнения программы, а их исправление займёт гораздо больше времени, чем своевременная проверка.

Развёртывание программы происходит благодаря фреймворку Apache Maven, окружение не имеет большого значения для запуска кода, если все данные и дерево иерархии наследования сохранены.

Сопровождение в данном проекте не предусматривается на постоянной основе, но сопровождение в смысле обновления функционала и развитие приложения безусловно будет происходить.

Программирование используется во многих сферах, от создания веб-сайтов и мобильных приложений до разработки игр и анализа данных.

Так же в программирование существует огромное количество языков, каждый из которых обладает различными характеристиками, такими как:

* Мощность
* Уровень.
* Концептуальная целостность.
* Надежность.
* Удобочитаемость
* Полнота.
* Гибкость.
* Простота.

Мощность — характеризуется количеством и разнообразием задач, алгоритмы решения которых можно записать, используя данный язык.

Уровень — характеризуется сложностью решения задач с помощью этого языка. Чем проще записывается решение задач, тем выше уровень языка.

Концептуальная целостность — характеризуется свойствами совокупности понятий, служащих для описания этого языка.

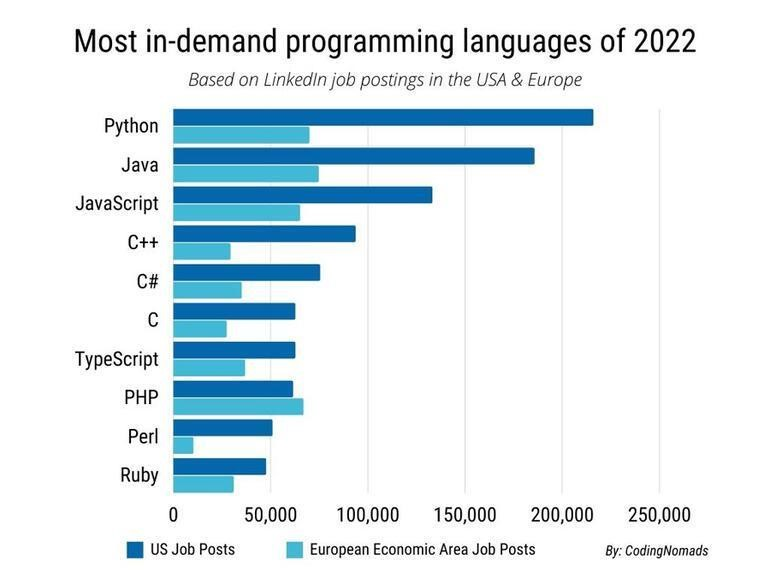
Надежность — обеспечивает минимум ошибок при написании программ.

Удобочитаемость — обеспечивает лёгкость восприятия программ человеком.

Полнота — обеспечивает описание на языке решения задач определённой предметной области.

Гибкость — обеспечивает лёгкость выражения на языке необходимых для решения задач действий.

Простота — характеризуется лёгкостью понимания семантики языковых конструкций и запоминания их синтаксиса.



Программист — специалист, который создаёт компьютерные программы, сайты, веб-сервисы и мобильные приложения с помощью языков программирования.

Основные обязанности программиста включают в себя:

* Написание, отладку и тестирование кода
* Анализ и написание алгоритмов
* Работа с системами контроля версий
* Обновление и написание технической документации
* Исправление и оптимизацию программ
* Разработка программного обеспечения

Написание кода — это этап программирования, на котором пишется программа.

Отладка — это процесс проверки работоспособности программы и исправления обнаруженных при этом ошибок. 3 Цель отладки — исправить проблемы таким образом, чтобы исправленная часть кода не вносила новых ошибок и соответствовала всем требованиям и спецификациям проекта.

Тестирование — это процесс выполнения программы на наборах исходных данных (тестах), для которых известны результаты, полученные другим методом. 3 Тестирование помогает обнаружить дефекты в программном обеспечении, проверяя его работоспособность в разнообразных условиях и сценариях использования.

Анализ и написание алгоритмов — это процессы, которые включают в себя ряд этапов:

* Постановка задачи
* Выбор метода решения
* Разработка алгоритма (алгоритмизация)
* Составление программы
* Отладка программы
* Анализ результатов

Постановка задачи — формулирование задачи с указанием входной и выходной информации и её цели.

Выбор метода решения — построение математической модели задачи, в которой связи между входными и выходными данными записываются в виде математических соотношений.

Разработка алгоритма (алгоритмизация) — построение алгоритма решения задачи выбранным методом.

Составление программы — запись составленного алгоритма на одном из языков программирования.

Отладка программы — исправление синтаксических и семантических (смысловых) ошибок в тексте программы, а также проверка её работоспособности на контрольном примере, то есть тестирование программы.

Анализ результатов — изучение результатов работы программы и их сопоставление с предполагаемыми (если таковые имеются)

Система контроля версий (СКВ, VCS) — это один из ключевых инструментов разработки, позволяющий наладить непрерывную интеграцию и развертывание ПО (CI/CD) в процессе работы над проектом. СКВ фиксирует все изменения в проекте, позволяет сохранять различные его версии, а при необходимости вернуться к нужному состоянию. Это защищает исходный код от непреднамеренного удаления или повреждения.

Обновление и написание технической документации — это типовой жизненный цикл, который включает следующие этапы:

* Планирование
* Сбор информации
* Проектирование
* Написание
* Редактирование
* Ревью
* Публикация
* Приёмка
* Сопровождение (поддержка)
* Мониторинг

Планирование — формируются состав и объём документов, разрабатываются требования, определяются сроки и критерии качества.

Сбор информации — изучаются, анализируются и систематизируются необходимые сведения.

Проектирование — разрабатывается общая структура, определяется содержание документов, устанавливаются требования к оформлению.

Написание — технические писатели на основе собранной информации и утверждённого задания разрабатывают проекты документов

Редактирование — документы редактируются на предмет грамматических и стилистических ошибок. Проверяется соблюдение единства используемой терминологии и общая согласованность документа

Ревью — эксперты и другие заинтересованные стороны изучают, проверяют на соответствие критериям качества и утверждают документы.

Публикация — документы размещаются в оговорённых форматах и передаются заказчику.

Приёмка — проводится работа с замечаниями внутреннего или внешнего заказчика, вносятся правки.

Сопровождение (поддержка) — по мере развития продукта документация улучшается и обновляется.

Мониторинг — проверяется и анализируется широкий спектр параметров: начиная от состояния сервера, на котором опубликована документация, и заканчивая оценкой посещаемости и качеством контента.

Исправление программ (отладка) — это процесс устранения ошибок в коде программы, найденных во время тестирования или при поступлении жалоб от пользователей.

Оптимизация программ — это преобразование программ для ЭВМ, направленное на улучшение их рабочих характеристик, чаще всего времени выполнения, реже — объёма, энергии, потребной для выполнения.

Разработка программного обеспечения (ПО) — это комплексный процесс создания программных продуктов и приложений, таких как операционные системы, офисные пакеты, игры, веб-сервисы и мобильные приложения.

Этот процесс включает в себя создание, тестирование и усовершенствование программных продуктов, а также их непрерывное техническое обслуживание.

Как правило, документацию в компании ведёт отдельный специалист — технический писатель. Он составляет внутренние документы для сотрудников и инструкции для пользователей.

Программисты создают и поддерживают продукты в различных сферах: бизнеса и финансов, медицины и здравоохранения, образования и науки, искусства и развлечений.

Программирование нужно, чтобы создавать программы, которые выполняют разнообразные задачи. Например, когда человек использует калькулятор на телефоне, за его работой стоит программа, написанная программистом.

Начинать программирование можно как дополнительный навык к основной профессии. В таком случае стоит начать с высокоуровневых языков программирования с большим комьюнити. Это, например, Python, JavaScript, Java, C#. Выбор языка зависит от того, какие задачи собирается решать человек.

Программирование также полезно для понимания технологий. Понимание основ программирования помогает лучше понять, как работают различные устройства и системы. Например, знание программирования может помочь понять, как работает алгоритм рекомендаций на YouTube или как функционирует система бронирования авиабилетов.

**1.3 Этапы разработки консольного приложения.**

В этом проекте, основываясь на принципах SOLID код разбит по нескольким классам:

* CertificationDiplomApplication
* Start
* Add\_file
* Create\_file
* FIleList
* FTP\_connect

Класс CertificationDiplomApplication — является так называемым “клиентским кодом” в этом классе реализован исключительно метод main() — который является необходимым методом для запуска Java программ. Внутри этого метода вызывается исключительно метод класс Srart go().

В классе Start реализовано 2 метода:

* go()
* go\_2()

Метод go() — создаёт новый объект класса Scanner (В языке программирования Java класс Scanner предоставляет удобный способ для считывания данных с консоли или из других источников в программе) с параметром System.in, благодаря этому параметру в дальнейшем будет возможность записывать данные введённые с клавиатуры в переменные. Следующим шагом создаётся новый объект класса File (Класс File в Java представляет файлы и пути каталогов в абстрактной манере. Этот класс используется для создания, поиска, удаления файлов и каталогов и т.д.) с названием “dir” с параметром “C:/Users/user/Desktop/demo/share” для того, чтобы создать по этой ссылке файл с название “share”. После создаётся переменная “created” типа boolean (boolean в Java — это логический тип данных. Переменная этого типа может принимать только два значения: true (истина) или false (ложь)) и приравнивается к методу “ dir.mkdir()” который преобразует файл “share” в одноимённую директорию. В последующих шагах в эту папку будут сохраняться файлы созданые пользователем внутри приложения и копироваться по ссылкам введённым пользователем. После создания пременной “created” следует проверка через if (If в Java — это основной оператор выбора, который позволяет задать условие, в соответствии с которым дальнейшая часть программы может быть выполнена) в условия проверки которого вписана эта переменная, в случае если директория успешно будет создана — в консоли появиться сообщение “Папка share создана”, в обратном случае в консоль не будет выведено ничего. Следующим шагом создаётся новый объект класса ArrayList<File> (ArrayList<File> в Java — это массив с возможностью изменения размера, который содержит файлы) с названием “file\_list” который методом “Arrays.asList(dir.listFiles())” добавляет все файлы которые есть в директории “share”. Благодоря этой строке при перезапуске программы, все файлы которые были созданы в прошлой сесси будут отображаться при использовании функции “Просмотр всех добавленных файлов”. Далее создаётся цикл while (While в Java — это оператор цикла с неизвестным заранее числом итераций) с условием выхода “true” для циклического вывода меню пользователю, пока он не воспользуеться функцией “Выход”, после чего программа завершит свою работу. В цикле методом “System.out.println()” (В языке программирования Java вывод на консоль осуществляется с использованием метода “println()” объекта “System.out”) прописаны следующие сообщения:

* \nИнструкция пользования: для выбора необходимой функции введите номер этой функции в консоль
* \nMenu:
* Создать файл
* Добавить файл
* Отправить файл
* Просмотр всех добавленных файлов
* Просмотр параметров подключения по FTP протоколу
* Выход

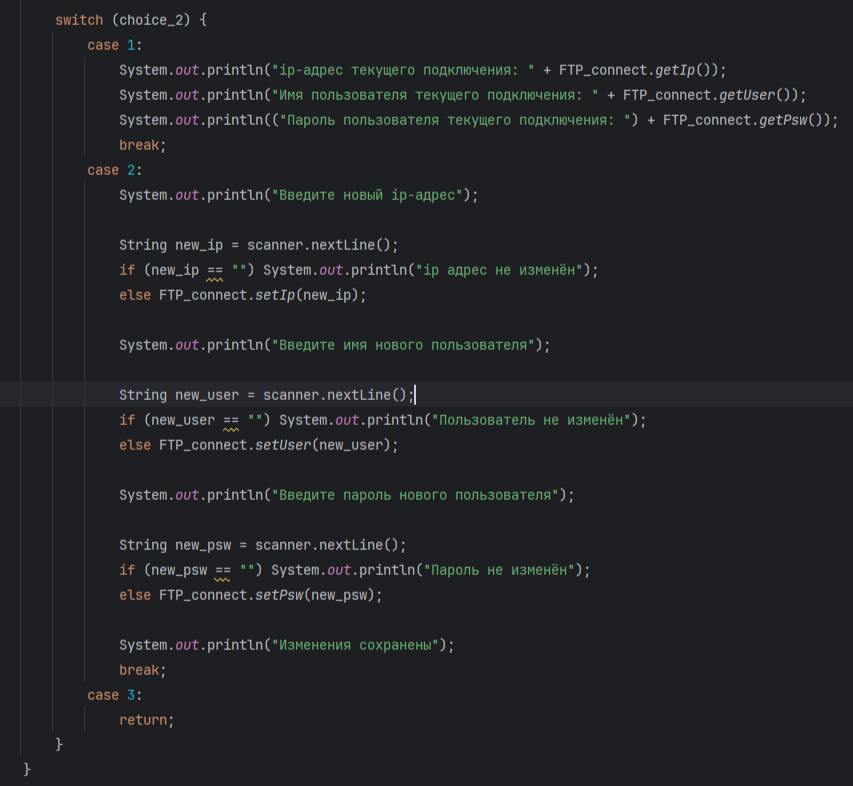
\n — это символ для перехода на новую строку. Эти сообщения будет видеть пользователь, когда запустит программу. После создаём переменную с названием “choice” типа int (int в Java — это 32-битное целочисленное значение) приравнённую к методу “nextInt()” объекта “scanner”. Благодаря этой строке в переменную “choice” будет возможно записать значение, введённое пользователем. После прописываем метод “nextLine()” того же объекта для того что бы переменная “choice” была значением, которое вписал пользователь. Далее прописывается конструкция “swith-case” которая позволит вызывать метод той функции, значение которой вписал пользователь предыдущим шагом. Так же в этом проекте прописано значение “default” для того, чтобы если пользователь введёт значение, за котором не закреплена ни одна из функций программы, в консоли появиться сообщение “Такой функции нет. Попробуйте ещё раз”, после чего опять появиться главное меню. Сами “case’ы” прописаны довольно просто, каждый из них вызывает метод определённого класса, для выполнения своей задачи, после чего в консоли появляется главное меню, кроме шестого “case’а” потому что в нём прописана логика завершения работы программы, которая возвращает значение в цикл “while” после чего работа как программы, так и цикла завершается, но перед этим в консоли появляется сообщение “Выход…”. Таким образом, методы в “case’ах” выглядят так:



Метод go\_2() — необходим для меню действий если пользователь воспользуется функцией “Просмотр параметров подключения по FTP протоколу”. Этот метод по структуре и выполняемой задаче схож с методом “go()” того же класса. Этот метод начинается так же с создания нового объекта класса “scanner” для того, чтобы программа могла воспринимать значения, введённые пользователем. После в консоль выводиться сообщения, с функциями и обозначающими их значениями. Следующим шагом создаётся переменная класса “int” с название “choice\_2” и метод класса “scanner” для записи значения, введённого пользователем в переменную. В этом методе пользователю предлагаться функции:

* Просмотр текущих параметров подключения
* Изменение параметров подключения
* Вернуться в основное меню

После идёт та же конструкция “swith-case”, за исключением изменения значений:



Методы getIp(), getUser(), getPsw() класса “FTP\_connect” — необходимы для работоспособности функции “Просмотр текущих параметров подключения” возможность воспользоваться которой появляться если пользователь выберет функцию “Просмотр параметров подключения по FTP протоколу”. Эти методы возвращают значения типа “String”. При выборе пользователем функции “Просмотр текущих параметров подключения” в консели появляется 3 сообщения, в которых отображаються текущие характеристики подключения по “FTP” протоколам, а именно: ip-адрес подключения, имя пользователя и пароль этого пользователя. Реализованна эта функци простым выводом сообщения в консоль методом “println()” класса “System.out” с параметрами :

"ip-адрес текущего подключения: " + FTP\_connect.getIp()

"Имя пользователя текущего подключения: " + FTP\_connect.getUser()

"Пароль пользователя текущего подключения: ") + FTP\_connect.getPsw()

Благодаря этим параметрам в консоли появляются сообщения с данными, которые используются в подключении на настоящий момент.

Методы setIp(), setUser(), setPsw() класса “FTP\_connect”— необходимы для работоспособности функции “Изменение параметров подключения”. При выборе пользователем этой функции в консоли появляеться сообщение “Введите новый ip-адрес” после чего создаётся переменная с названием “new\_ip” типа “String” приравненная к методу “nextLine()” объекта “scanner” для сохранения значения введённого пользователем в консоль. После запускаеься оператор “if” в условие которого указанно, что если переменная “new\_ip” это пустая переменная, то есть пользователь не отправил новое значение “ip” адреса, то в консоли появиться сообщение “ip адрес не изменён”, в обратном случае, в консоли не появиться сообщение.

При выборе пользователем функции “Вернуться в основное меню” в цикл вернёться значение “true” и цикл будет завершён, после чего пользователю в консоли будет выведенно сообщение с основным меню функций.

В случае если пользователь введёт значение, не предусмотренное программой, в консоли будет выведено сообщение “Такой функции нет. Попробуйте ещё раз” после чего будет выведено сообщение с текущем меню функций.

В классе Add\_file реализован 1 метод:

* add()

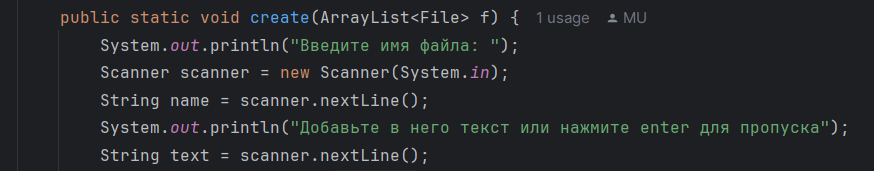
Метод add() — выводит в консоль сообщение “ведите путь к файлу: “, после чего создаётся новая переменна класса Scanner (В языке программирования Java класс Scanner предоставляет удобный способ для считывания данных с консоли или из других источников в программе.) с параметром System.in для того что бы программа считывала данные с клавиатуры введённые в консоль и могла записывать их в переменные. После этого создаётся переменная строчного типа String именуемая f\_path приравненная к значению введённого в консоль, затем создаться переменная типа File с именем add\_file в которую вкладывается значение переменной f\_path, так как переменная f\_path — является путём (абсолютной ссылкой) к файлу, который пользователь хочет добавить в папку share (созданную при запуске метода go класса Start) для дальнейшего переноса на виртуальный компьютер по FTP протоколам, а переменная add\_f — можно использовать как файл, находящийся по ссылке указанной в переменной f\_path. Следующим шагом идёт оператор выбора if – else (Конструкция if – else позволяет проверить одно или же несколько условий и в случае, если условие не будет верным, то выполнить другой код или проверить другое условие.) проверяющий, существует ли файл по пути (ссылке) который ввёл пользователем, если файл существует, то программа добавит его в папку share и выведет сообщение в консоль “ Файл найден и скопирован в папку share”, в обратном случае консоль выведет сообщение “Файл не найден” и вернётся в меню выбора функции.

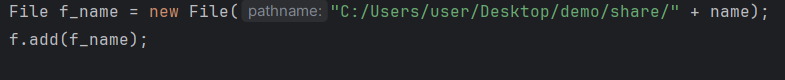


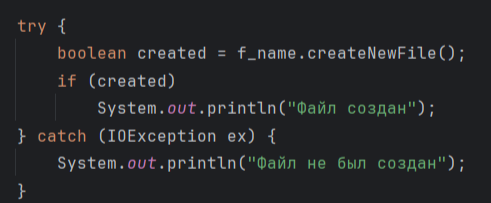
В классе Create\_file реализован 1 метод:

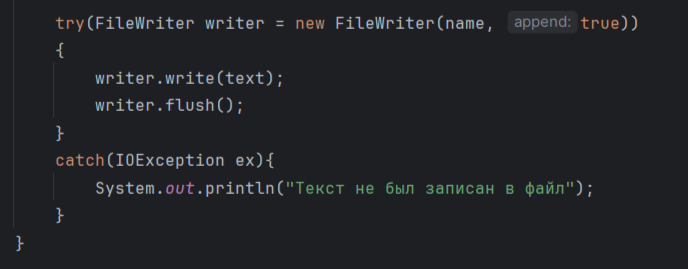
* create()

Метод create() — необходим для работы функционала создания файла внутри программы. В отличие от ранее от ранее описанных методов, метод “create()”. При запуски этой функции в консоли появляется сообщение “Введите имя файла: ” с помощью метода “println()” класса “System.out” после чего создаётся новый объект класса “Scanner” с параметром “System.in” для передачи значений введённых в консоль пользователем в переменные. Затем создаётся переменная “name” типа “String” (String в Java — это объект, представляющий последовательность символов) которая приравниваеться к значению, которое слудующим будет введено пользователем в консоль методом “nextLine()” объекта “scanner”. После в консоль выводиться сообщение “Добавьте в него текст или нажмите enter для пропуска”, после чего создаеться переменная “text” типа “String” приравниваеться к значению, которое слудующим будет введено пользователем в консоль таким же методом, что и с переменной “name”.

 После окончания этого блока следует блок создания файла с введёным пользователем именем и содержанием. Это происходит через создание нового объекта класса “File” с названием “f\_name” и параметром "C:/Users/user/Desktop/demo/share/" + “name” для создание нового файла с название “name” находящемся в директории “share” по вышеуказанному пути. После этого, новый файл “f\_name” добовляеться в “file\_list” созданый при запуске программы в методе “go()”. Это необходимо для того, что бы пользователь при запуске функции “ Просмотр всех добавленных файлов ” мог увидеть только что созданый файл.

В следующем блоке производиться проверка создания файла и запись в него текста конструкциями “try-catch” (конструкция try-catch в Java используется для обработки исключений в программе). Проверка происходит поиском файла в директории, в которую он был определён. Таким же способом, что и в методе “go()”, но проверяется был ли создан файл, а не директория. Так как в этом месте кода может возникнуть ошибка, то весь блок с условием “if” помещён в блок “try”. Так как ошибка, которая может возникнуть, это “IOException” (ошибка при вводе-выводе данных), то в блоке “catch” параметром указано название этой ошибки (“IOException”) и краткое наименование для последующих действие с этим параметром в блоке “catch” (“ex”). Далее в этом блоке прописано, что в случае, если при выполнении части кода, прописанного в блоке “try” появиться эта ошибка, то в консоли появиться сообщение ”Файл не был создан” и программа продолжит работу в штатном режиме. Если бы эта конструкция не была прописана, то в случае появления ошибки — программа прекращала работу, а в консоли появлялся список мест в коде, где появилась ошибка.

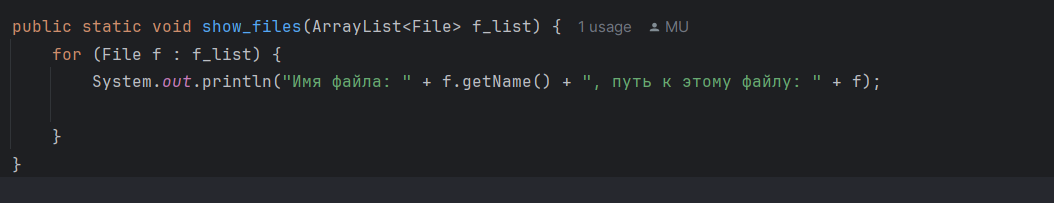
Далее прописана логика записи в созданый файл текст, указаный пользователем выше. Так как при выполнении этой части кода может возникнуть та же ошибка “IOException”, то следующий код то же записан в конструкции “try-catch”. Для записи текста в файл создаётся новый объект класса “FIleWriter” с параметрами “name” и “true”. “name” — имя файла в который нужно вписать текст, значение “true” в параметрах класса “FIleWriter” означает, что текст будет до записан к тексту который был уже в файле. После следуют строки, которые записывают текст в указанный файл, а после очищают содержимое потока “Writer”. После этого прописан блок срабатывания “catch”, который в случае ошибки выведит в консоль сообщение “Текст не был записан в файл”.



В классе FileList реализовано 1 метода:

* show\_files()

Метод show\_files() — необходим для работы функции “Просмотр всех добавленных файлов". В этом методе прописана логика вывода в консоль всех файлов, которые есть в папке “share” с именем и абсолютной ссылкой на эти файлы. Реализован этот метод через работу с “file\_list” который был создан при запуске программы в методе “go()” и который хранит в себе список всех файлов, которые есть в директории “share” которая была создана в том же методе. Этот метод выводит сообщением в консоль файлы через цикл “foreach” (цикл foreach позволяет перебирать элементы коллекции без необходимости использовать индексы или итераторы). Внутри цикла прописано единственно действие — отправка сообщения в консоль через метод “println()” класса “System.out”с параметрами "Имя файла: " + “f.getName()” + ", путь к этому файлу: " + “f”. “f” — название итерационной переменной для перебора объектов типа “File” внутри массива “f\_list”. “f.getName()”— метод для возращения имени запрошенного файла



В классе FTP\_connect реализовано 7 методов и 3 поля:

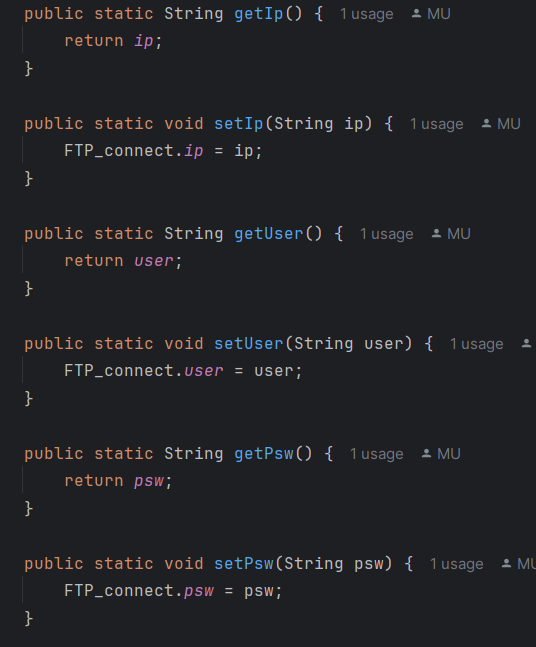
Методы:

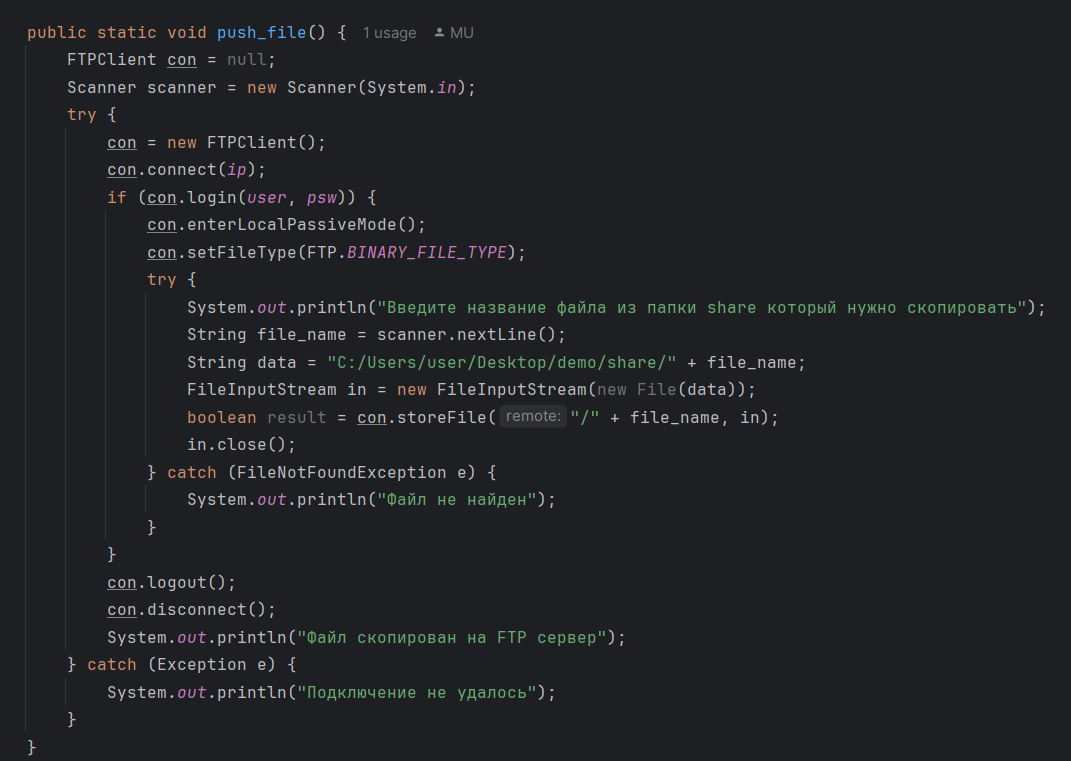
* getIp()
* setIp()
* getUser()
* setUser()
* getPsw()
* setPsw()
* push\_file()

Поля:

* ip
* user
* psw

Методы getIp(), getUser(), getPsw(), setIp(), setUser(), serPsw() — реализованы для отображения информации о переменных, связанных с текущем подключением по “FTP” протоколу, или для изменения этих самых переменных. Методы, возвращающие значение переменных в текущем подключении — getter’ы. Их реализация очень проста, единственное их действие “return” с именем переменной, которую необходимо вернуть. Методы, изменяющие значения переменных, связанных с подключением — setter’ы. Их реализация ненамного сложнее, эти методы ничего не возвращают, только изменяют значение переменной по введённому параметру.

Метод push\_file() — необходим для реализации цели этого проекта, переноса файлов на виртуальную машину по “FTP” протоколу. В этом методе создаётся объект “con” класс “FTPClint” (FTPClient в Java — это класс, который обеспечивает функциональность для сохранения и извлечения файлов с FTP-сервера) приравнённый к значению “null”, то есть этот объект пуст. После этого создаётся новый объект “scanner” класса “Scanner” с параметром для возможности приравнивания объектам и переменным значений, введённых пользователем. После следует конструкция “try-catch” где в блоке “try” объект “con” приравнивается к значению класса ”FTPClient”, после чего объект “con” вызывает метод “connect()” с параметром “ip” для подключения к “FTP” серверу находящемуся по этому адресу, но так как в “ip” адрес может быть введён неверный, блок “catch” отслеживает ошибку “Exeption” и в случае если какая-либо ошибка появиться при реализации этого или последующей части кода в блоке “try” в консоль будет выведено сообщение “Подключение не удалось”. После, в блоке “try” следует оператор “if” с параметром “con.login(user, psw)”, этот метод регистрируется на “FTP” сервере находящемся по указаному выше “ip” адресу, в случае удачного подключения запускается метод “enterLocalPassiveMode()” объекта “con”, это необходимо для поддержания подключения, даже когда не происходит обмен файлами. Следующим следует метод “setFileType()” того же объекта с параметром “FTP.BINARY\_FILE\_TYPE”, это константа, которая указывает, что передаваемый файл следует рассматривать как бинарное изображение, то есть не выполнять перевод. После следует ещё одна конструкция “try-catch”. После успешного запуска этого блока “try” в консоль выводиться сообщение “ Введите название файла из папки share который нужно скопировать” после создаётся переменная “file\_name” типа “String” приравненная к значению, введённым пользователем в терминал методом “nextLine()” объекта “scanner”. После создаёться переменная “data” типа “String” равная полю “share” с прибавлением переменной “file\_name”, таким образом, переменная “data” это файл, название которого ввёл пользователь, находящийся в папке “share”. Далее создаётся переменная “in” класса “FileInputStream” (FileInputStream в Java — это поток для чтения данных из файла) приравненный к значению нового потока для чтения данных из файла с параметром “new File(data)” необходимый для чтения файла, который указал пользователь. После создаётся переменная “result” типа “boolean” которая приравнена к значению метода “storeFile()” объекта “con” с параметрами “/” + “file\_name”, “in” которые необходимы для записи файла выбранного пользователем на “FTP” сервер. После этого блок “try” завершается методом “close()” объекта “in” закрывая потока для чтения файлов. После, в блоке “catch” отлавливается ошибка “FileNotFoundException” которая могла возникнуть в блоке “try” и в случае возникновения которой в консоли вывелось сообщение “Файл не найден”. Далее — переход в первый блок “try” в котором методом “logout()” объекта “con” подключение выходит из пользователя под котором было зарегистрированно во время подключения, а методом “disconnect()” того же объекта, подключение к виртуальному компьютеру по “FTP” протоколу завершаеться.



Поле ip — переменная типа “String” равная ip-адресу “FTP” сервера. Необходима для работоспособности метода “push\_file()” для подключения к “FTP”серверу находящемся на указанном адресе.

Поле user — переменная типа “String” равная имени пользователя, под котором производиться подключение на “FTP” сервер. Необходима для работоспособности метода “push\_file()” для подключения на “FTP” сервер под именем пользователя, указанном в переменной.

Поле psw — переменная типа “String” равная паролю для подключения на “FTP” под указаным выше именем пользователя. Необходима для работоспособности метода “push\_file()” при вводе пароля указанном в переменной для подключения на “FTP” сервер.

**1.4 Источники требований: что такое техническое задание и как разрабатывать программный код, если техническое задание отсутствует.**

Техническое задание (ТЗ) — это документ, в котором фиксируются требования к проекту. Условно ТЗ можно назвать любое поручение исполнителю, главное, чтобы в нём были ясно прописаны характеристики итогового продукта.

Основная цель создания технического задания — детально описать будущий проект. Это позволит команде, которая работает над проектом, и заказчику быть уверенными, что он получится таким, как задумано.

Техническое задание помогает решить ряд задач:

* фиксация пожеланий заказчика
* понимание бюджета, объёма и сроков работ
* разрешение разногласий

В техническом задании обычно содержатся следующие блоки:

* назначение проекта
* пользовательские группы
* обзор содержания (функции и сценарии использования)
* взаимодействие с другими компонентами
* обзор интерфейса (пожелания по структуре и дизайну страниц)
* безопасность (системы безопасности, которые обязаны быть в продукте)
* разработка (пожелания к технической части)
* системное окружение (технические характеристики серверов и устройств, на которых будет установлен продукт).

При заключении договора техническое задание становится неотъемлемой частью договора и является в ряде случаев юридическим документом.

В условиях отсутствия технического задания, как например в этом проекте, зачастую используется лишь идея исполнителя, а заказчик в целом отсутствует. При выполнении подобных проектов исполнитель отталкивается от тех же блоков, но интерпретирую под свою идею. Чаще всего проекты без технического задания не удовлетворяют ни сторону исполнителя, ни сторону заказчика (если он есть), так как подобные проекты изменчивы и их невозможно доделать до конца, из-за постоянного желания сделать его лучше и более продвинутым, в случае если нет заказчика, а если заказчик есть, то стороны врятли найдут реальные цели и договориться о реализации проекта, потому что без документального подтверждения слов, не возможно отталкиваться и работать, без ожидания скорого полного изменения проекта.

Этот проект реализовывался без технического задания, основываясь на цели и идеии. В основном это было проблемой для начала и завершения, так как сначала непонятно как начать разрабывать проект и начём сделать основание, а после не понятно, когда уже стоит заканчивать и не изменять всю реализацию из-зп одного придуманного только что метода, класса или чего-либо другого.

**Глава 2. Создание виртуальной Linux машины с FTP протоколами.**

В этой главе будет поэтапно рассмотрен процесс создания виртуального компьютера внутри своего рабочего компьютера. Виртуальный компьютер будет с операционной системой “Linux” и дистрибутивом “Ubuntu”, из-за удобства подключения к этому компьютеру в дальнейшем используя “SSH” протоколы. В последующих шагах будет рассматриваться именно терминал взаимодействия с виртуальным компьютером через “SSH” протоколы используя инструмент “PuTTY” который поддерживает стабильную работу и поддерживает полный функционал возможностей при подключении подобным образом. Так же для удобства последующей проверки работоспособности подключения по “FTP” протоколам будет использоваться “FAR Manager” из-за удобства синхронизация по “FTP” протоколам.

“Linux” — это семейство операционных систем (ОС) на базе одноимённого ядра. Нет одной операционной системы Linux, как, например, Windows или MacOS. Есть множество дистрибутивов (набор файлов, необходимых для установки ПО), выполняющих конкретные задачи.

Особенности Linux:

* Свободное использование
* Открытый исходный код системы/программы
* Актуальность и производительность
* Безопасность
* Универсальность

Свободное использование — использование большинства ОС Linux и большей части программ, основанных на ней, абсолютно бесплатно.

Открытый исходный код системы/программы — благодаря доступу к исходному коду можно изучать его, изменять, распространять, а также публиковать изменения в соответствии с лицензией.

Актуальность и производительности — по сравнению с Windows Linux не устаревает со временем.

Безопасность — Linux поддерживает разные параметры безопасности, защищающие от вирусов, замедлений, вредоносных программ, сбоев.

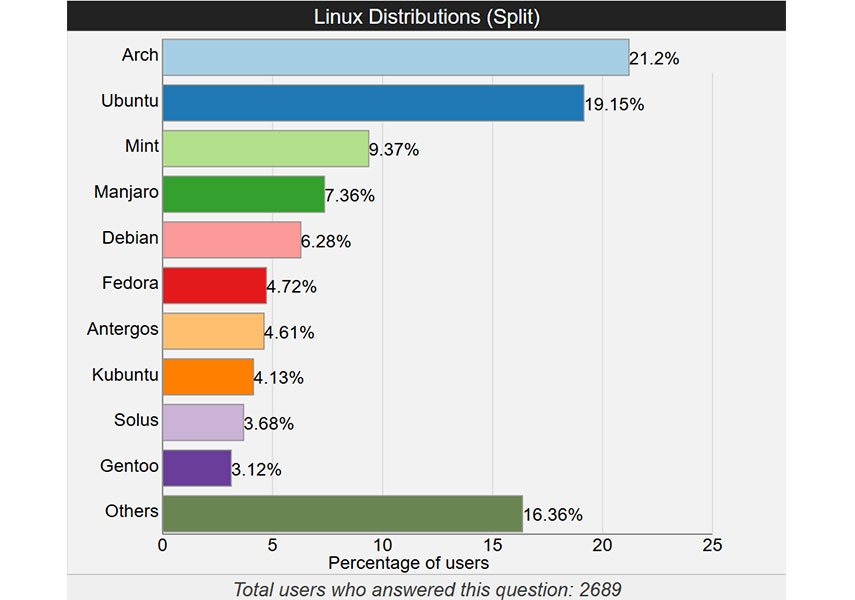
Универсальность — Linux поддерживает практически все популярные языки программирования: Java, C/C++, Python, Ruby, C# и другие.

Дистрибутив (форма распространения ПО) — это комплект (как правило, набор файлов), приспособленный для распространения программного обеспечения. Может включать вспомогательные инструменты для автоматической или автоматизированной начальной настройки ПО (установщик).

Дистрибутивы могут содержать:

* Для операционных систем
* Для программ

Для операционных систем — основные компоненты операционной системы (ядро, файловую систему и т. д.) и инструменты, необходимые для работы с ней. Также могут включать стандартные программы, такие как веб-браузеры, текстовые редакторы, мультимедийные приложения и т. д.



Для программ — все необходимые файлы для установки и запуска программы на компьютере. Список включает в себя исполняемый файл самой программы, библиотеки и зависимости, которые могут потребоваться для её работы, конфигурационные файлы, а также документацию или руководство пользователя.

Обычно дистрибутив упаковывается в один файл или архив для удобства распространения и установки на различных компьютерах.

Популярные дистрибутивы Linux: Debian, Ubuntu, Linux Mint, Arch Linux, MX Linux, Fedora, Manjaro, CentOS.

FTP (File Transfer Protocol) — протокол удалённой передачи данных через локальную или глобальную компьютерную сеть. С его помощью пользователь может подключиться к FTP-серверу для просмотра содержимого каталогов, а также чтобы загрузить или скачать файлы.

В процедуре передачи файлов участвуют два элемента — клиент и сервер. Первый отправляет запрос в виде команды, а второй обрабатывает её и возвращает ответ в виде файла или сообщения об ошибке.

FTP был разработан в 1971 году и относится к числу базовых протоколов передачи данных, которые используются в локальных и глобальных компьютерных сетях.

FTP-адреса обычно имеют формат ftp://hostname, где hostname — доменное имя или IP-адрес, например mysite.ru или 127.0.0.1

Принцип работы FTP заключается в следующем:

Клиентский компьютер открывает сессию с FTP-сервером, которая остается открытой на протяжении совершения всех требуемых операций.

Если пользовательское устройство не защищено файрволлом, дальнейшие действия ведутся в активном режиме, где сервер получает IP-адрес клиента и номер порта, куда будут приниматься пересылаемые файлы, после чего он открывает соединение с учетом полученных данных и параметров.

В ситуациях, где пользователь защищен файрволлом, используется пассивный режим FTP, когда входящее соединение от сервера не может быть принято по обычному алгоритму, а для решения используется управляющая команда PASV (перейти в пассивный режим).

Передача файлов с помощью FTP-протокола осуществляется в нескольких режимах:

* Блочный
* Поточный
* Режим сжатия

Блочный — материалы структурируются в виде «заголовок — объем — непосредственно данные».

Поточный — файлы передаются единым потоком без специальной структуры.

Режим сжатия — информация сжимается с помощью нужного алгоритма для оперативной и комфортной передачи.

SSH (Secure Shell — «безопасная оболочка») — это сетевой протокол для удалённого управления операционной системой с помощью командной строки и передачи данных в зашифрованном виде.

SSH-соединение состоит из двух компонентов:

* SSH-сервер
* SSH-клиент

SSH-сервер — это программа, которая устанавливает связь и производит аутентификацию с устройством пользователя. Он установлен на самом сервере.

SSH-клиент — используется для входа на удалённую машину и передачи ей команд. Он устанавливается на устройстве, с которого пользователь хочет подключиться к серверу.

SSH может использоваться:

* в удаленном системном администрировании локальных сетей
* в управлении работой почтовых служб (защите данных)
* для скрытой передачи файлов большого размера внутри сети
* для переноса проектов между серверами хостинг-провайдеров
* для подключения в сетевых многопользовательских играх и т. д.

Принцип работы SSH заключается в следующем:

* Установка TCP-соединения
* Открытие защищённого канала
* Аутентификация пользователя

Установка TCP-соединения — TCP — протокол управления передачей данных. С его помощью удалённый сервер и пользователь могут обмениваться данными.

Открытие защищённого канала — исходя из настроек соединения, обе стороны обмениваются информацией о способе шифрования и сжатия данных.

Аутентификация пользователя — она происходит при помощи открытого ключа. У сервера и клиента есть публичный и частный ключ. Публичный ключ доступен любому, а частный известен только владельцу.

**2.1 Установка инструментов для создания виртуальной Linux машины.**

В этом проекте для создания виртуального компьютера будут использоваться встроенный инструмент “Windows” — “Hyper-V”

Hyper-V является встроенной опцией в Windows Server 2008 и более поздних версиях. По умолчанию она не активна, поэтому её нужно настроить вручную.

Существует три способа включения Hyper-V в ОС Windows:

* Системные настройки Windows.
* Интерфейс командной строки PowerShell (CLI).
* Обслуживание образов развёртывания и управление ими (DISM).

Процесс установки Hyper-V с помощью панели управления Windows:

1. В строке поиска на панели задач введите «Настройки» и нажмите «Enter».
2. После открытия «Настройки» выберите «Приложения». В разделе «Связанные настройки» справа кликните на «Программы и компоненты».
3. Слева выберите «Включить или отключить компоненты Windows».
4. В диалоговом окне «Компоненты Windows» выберите «Hyper-V» и нажмите «ОК».
5. После завершения процесса установки Hyper-V перезагрузите компьютер, чтобы применить все необходимые изменения.

После необходимо скачать файлы ОС “Linux” с необходимым дистрибутивом, для последующей настройки виртуального компьютера. Для этого необходимо:

1. Перейти на сайт: <https://ubuntu.com>
2. В правом верхнем углу немного дальше центра экрана, выбрать раздел “products”
3. В этом разделе выбрать “Ubuntu Desktop”
4. После перехода нажать кнопку “Download Ubuntu Desktop”
5. После перехода выбрать версию для скачивания, затем нажать на кнопку “Download” рядом с выбранной версией

В этом проекте использовалась версия “Ubuntu” — “Ubuntu 24.04.1 LTS”.

**2.2 Создание виртуальной машины.**

Для этого проекта, виртуальная машина на Hyper-V создавалась по следующим этапам:

1. Запуск диспетчера Hyper-V
2. Настройка сети
3. Создание виртуальной машины
4. Параметры установки

Запуск диспетчера Hyper-V — его можно найти в поиске Windows 10 или в разделе «Средства администрирования» списка программ меню «Пуск».

Настройка сети — для реализации основной задачи этого проекта необходимо “FTP” подключение, которое не будет работоспособно без доступа к сети, поэтому в диспетчере Hyper-V слева в списке выберите «Диспетчер виртуальных коммутаторов» перед этим кликнув правой кнопки мышки по созданному для этого проекта компьютеру. В нём выберите «Создать виртуальный сетевой коммутатор», «Внешняя» (если нужен Интернет) и нажмите кнопку «Создать».

Создание виртуальной машины — кликните правой кнопкой мыши по имени компьютера в списке слева или нажмите пункт «Действие» меню, выберите «Создать» — «Виртуальная машина». Укажите размер памяти, необходимый для планируемой к установке ОС, и создайте виртуальный сетевой адаптер. Подключите или создайте виртуальный жёсткий диск, укажите его место расположения на диске и размер.

Рекомендованный объём оперативной памяти — 1024 МБ.

Рекомендованный объём выделенного дискового пространства — 10 ГБ.

Создать виртуальный жёсткий диск. Тип виртуального жёсткого диска можно оставить как есть — VDI (VirtualBox Disk Image). Формат хранения данных для данного проекта — динамический виртуальный жёсткий, так как объём диска растёт по мере заполнения, а фиксированный создаётся сразу того размера, который был указан на предыдущем шаге.

Параметры установки — например, установив пункт «Установить операционную систему с загрузочного компакт-диска или DVD», можно указать физический диск в приводе или файл образа ISO с дистрибутивом, который был скачен в предыдущем пункте.

Запустите виртуальную машину. Для этого дважды кликните по ней в списке диспетчера Hyper-V, а в окне подключения к виртуальной машине нажмите кнопку «Включить».

**2.3 Настройка виртуальной машины.**

После запуска только что созданной виртуальной машины на ОС “Linux Ubuntu Desktop” следует меню установки, для этого проекта выбор был тот же, что и для любой подобной виртуальной машины. По всем запросам во время установки были даны ответы, с пометкой “default”. После окончания загрузки ОС, виртуальная машина будет перезагружена. После этой перезагрузки виртуальная машина готова к работе и дальнейшей настройки внутренней среды.

**2.4 Подключение SSH протоколов.**

Для подключения SSH на Ubuntu Desktop необходимо:

* Установить пакет openssh-server
* Проверить статус сервера
* Разрешить SSH через брандмауэр
* Подключить к серверу

Установить пакет “openssh-server” — для этого нужно открыть терминал и ввести команду “sudo apt install openssh-server”. При появлении запроса ввести пароль и нажать “Enter” для продолжения установки.

Проверить статус сервера — после завершения загрузки и установки пакета служба SSH должна быть уже запущена. Проверить это можно с помощью команды “sudo systemctl status ssh”.

Разрешить SSH через брандмауэр — Ubuntu поставляется с утилитой межсетевого экрана под названием UFW (UncomplicatedFirewall). Если брандмауэр активен, он может помешать подключению к SSH-серверу. Чтобы настроить UFW так, чтобы он разрешал требуемый доступ, нужно выполнить команду “sudo ufw allow ssh”.

Подключить к серверу — для этого понадобятся его IP-адрес или доменное имя. В строку терминала нужно ввести команду “ssh имя\_пользователя@IP\_адрес” или “ssh имя\_пользователя@домен”.

**2.5 Подключение FTP протоколов.**

Для подключения “FTP” протоколов необходимо вводить команды через терминал. Вся последующая настройка и работа этого проекта будет происходить через терминальные запросы.

Сначала загружаем все доступные обновления командной: sudo apt full-upgrade.

После устанавливаем vsFTPd, необходимый для работы “FTP” протоколов командной: sudo apt install vsftpd.

Следующим шагом открываем конфигурационный файл командной: nano /etc/vsftpd.conf.

После снимаем коментарий со строк:

* write\_enable=YES
* chroot\_local\_user=YES

И дописываем:

* allow\_writeable\_chroot=YES
* pasv\_enable=YES
* pasv\_max\_port=60000
* pasv\_min\_port=65535

“write\_enable” — разрешить копировать файлы на сервер.

“chroot\_local\_user” — использовать для пользователей изолированное окружение.

“allow\_writeable\_chroot” разрешаем использовать домашние каталоги с правом на запись.

“pasv\_enable” включает пассивный режим работы FTP-сервера.

“pasv\_max\_port” и “pasv\_min\_port” определяют диапазон портов для пассивного режима.

Далее добавляется пользователь, для авторизации с использованием локальных пользователей. Это необходимо для работоспособности “FTP” протоколов и производиться командной: “useradd ftpuser -d /ftp -s /bin/false -m”

Где ftpuser — имя учетной записи; /ftp — домашний каталог (в него будем попадать при подключении); /bin/false — запрет пользователю на локальный вход в систему.

Следующим шагом задаём пароль для авторизации под этим пользователем командной: “passwd ftpuser”

После этого необходимо внести изменения в файл “/etc/shells” командной: “nano /etc/shells”

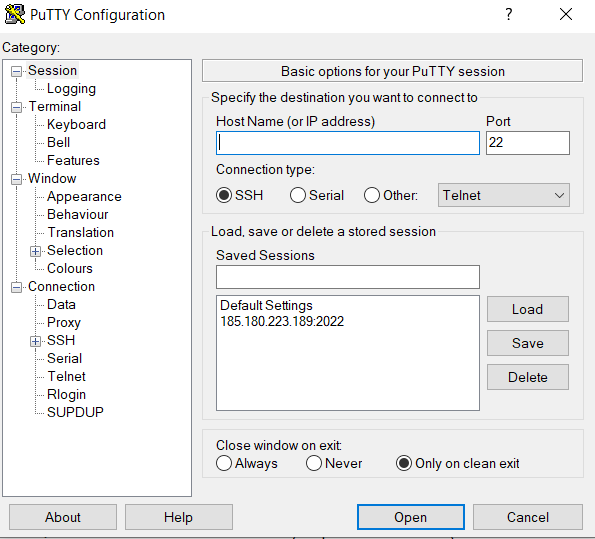
Необходимо добавить в этот файл строку “/bin/false” для добавления оболочки в список разрешённых оболочек, случае если не выполнить этот шаг может вернуться ошибка “530 Login incorrect”.

**2.6 Проверка результата на основной машине.**

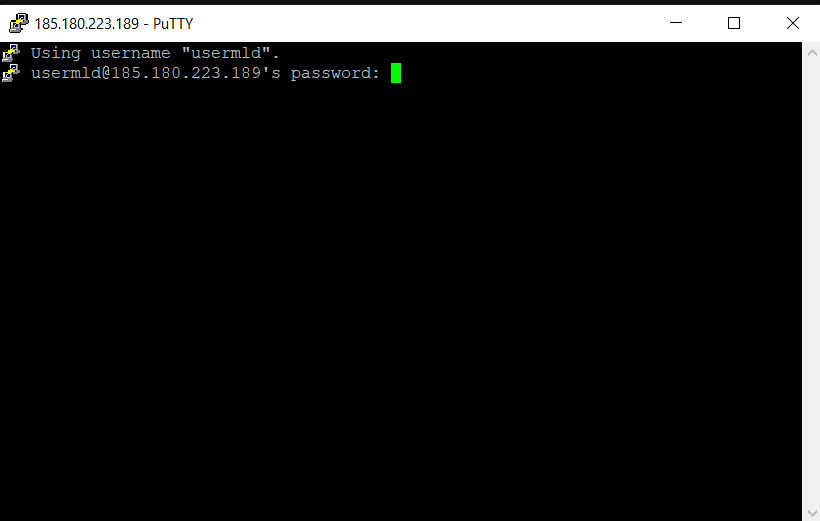
Для проверки результатов работы, проделанной в двух предыдущих пунктах, необходимо будет установить 2 программы:

* PuTTY
* FAR Manager

Для скачивания “PuTTY” — необходимо перейти на сайт “putty.org.ru”, перейти во вкладку “Скачать”, а после выбрать необходимую версию программы (в рамках этого проекта использовалась версия “PuTTY”) и нажать на ссылку напротив выбранной версии, после чего начнётся загрузка. После загрузки программа на рабочий экран выводит небольшое окно:



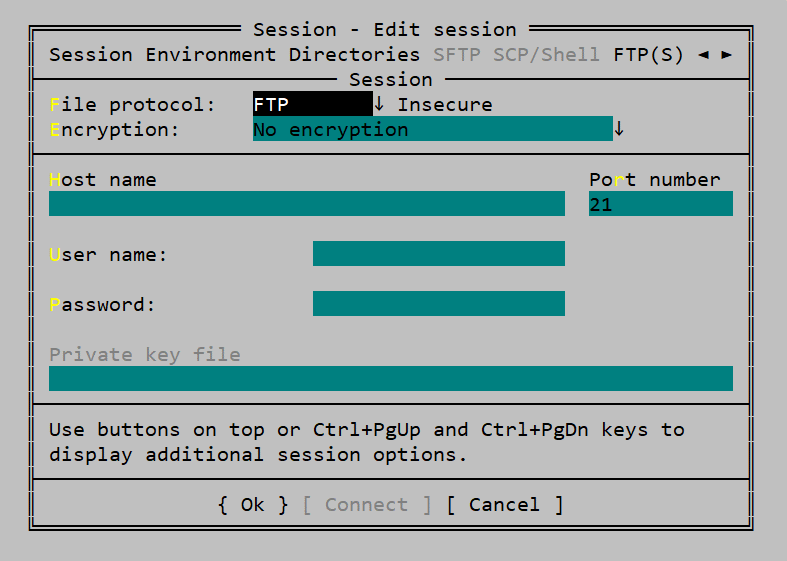
Для проверки “SSH” подключения необходимо ввести в строку “Host Name(or IP address)” “ip” адрес виртуальной машины, который можно узнать введя в терминал виртуальной машины команду “ifconfig”. После нажать кнопку “Open”. В случае если введён правильный “ip” адрес откроеться окно терминала со строками для авторизации, для продолжения подключения



Так как после ввода запрашаваемых данных в этом окное появился список директорий на виртуальной машине, значит все предыдущие шаги в пункте 2.4 выполнены верно.

Для скачивания “FAR Manager” — необходимо перейти на сайт “https://www.farmanager.com/download.php?l=ru”, после необходимо выбрать версию, в рамках этого проекта используется версия “3.0 x64”, после выбора версии необходимо нажать на кнопку с желаемым форматом загрузки, и программа начнёт скачиваться. После запуска программы открываться синее окно во весь экран рабочего стола, на котором в терминальном формате выведены все директории внутри диска “C” (по умолчанию).

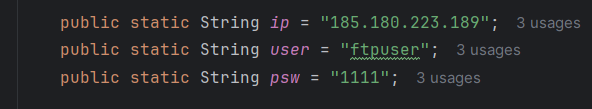
Для проверки работы, проделанной в пункте 2.5, необходимо провести обмен файлами с “FTP” сервером. Для этого в “FAR Manager” откроем 2ю панель с тем же диском использовав команду “ALT + F2”, откроеться маленькое окно, в котором необходимо выбрать диск “C”, после чего нажать “Enter” и в правой части экрана появиться паанель с директориями диска “C”. Послк этого, клавишей “TAB” переходим на правую панель и через директории доходим директорию “Desktop” в которой выбираем какой-нибудь файл (это не имеет значения). После этого, ещё раз нажимаем клавишу “TAB” и переходим на левую панель. Далее используем команду “ALT + F1” в появившемся окне необходимо выбрать “NetBox” и нажать клавишу “Enter”. В левой панели появиться список сетевых подключений. Для создания нового подключения необходимо использовать команду “SHIFT + F4” после ввода этой команды откроется окно параметров нового подключения.



В поле “File Protocol: “ необходимо изменить “SFTP” на “FTP”. В поле “Host name” необходимо вписать “ip” адрес виртуальной машины, а в поле “Port Number” необходимо вписать значение 21, так как по стандарту “FTP” подключение проходит по 21 порту. В полях “User name: “ и “Password: ” вводим данные вписанные после ввода команд для создания пользователя и его пароля в пункте 2.5. После нажимаем кнопку “Connect” и если все данные введены правильно, откроется директория, созданая на виртуальной машине, доступная из вне локального подключения. Для проверки работоспособности нажимаем клавишу “TAB” переходя на выбранный ранее файл на рабочем столе, а после нажимаем клавишу “F5” в следствии чего, выбранный файл должен успешно скопироваться в директорию “FTP” сервера.

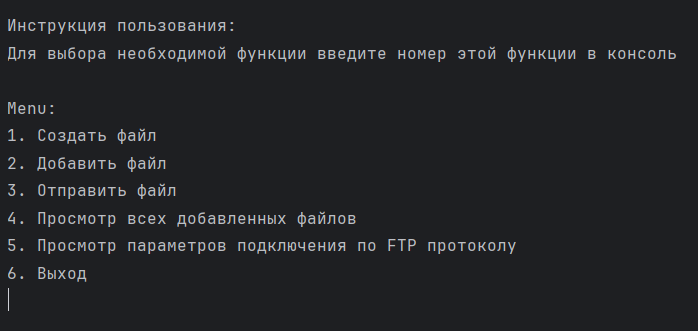
**Глава 3. Проверка работоспособности программного кода и подведение итогов.**

После проверки работоспособности подключения по “FTP” необходимо ввести данные виртуальной машины в поля класса “FTP\_connect” и проверить перенос файлов при помощи кода в консольном приложении.

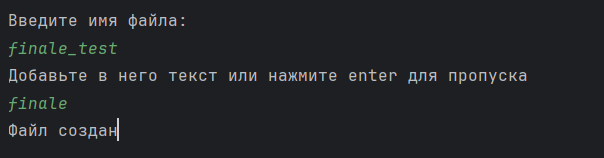


Это необходимо для создания подключения, которое точно работает и существует.

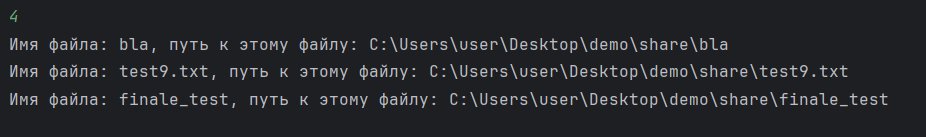
Далее необходимо только в работоспособности кода переносить файлы в рамках консольного приложения. Для этого запускаем приложение



Проверка функции “Создать файл”:

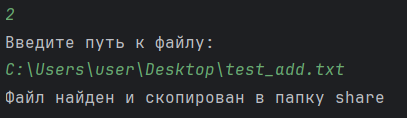


В консоли не появилось сообщений об ошибке, но для проверки создался ли файл, можно воспользоваться функцией “Просмотр всех файлов”. Таким образом будет проверено работоспособность двух функций.

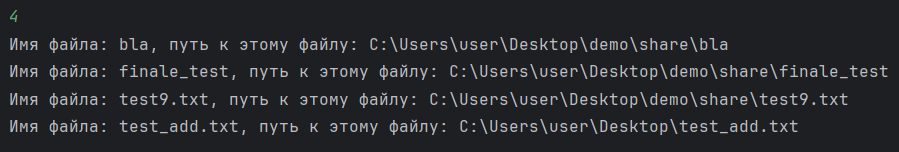


Функции “Просмотр всех файлов” и “Создать файл” полностью работоспособны.

Проверка функции “Добавить файл” так же от части будет производиться при помощи функции “Просмотр всех файлов”. Для проверки, будет добавлен файл с рабочего стола.

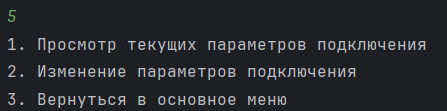


Сообщения в консоли работают в штатном режиме.

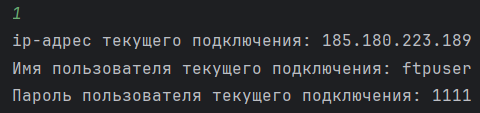


Функция “Добавить файл” работоспособна.

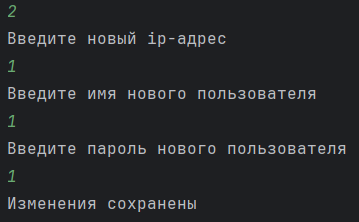
Проверка функции “Просмотр параметров подключения по FTP протоколу”.



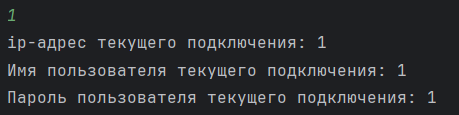
Вывод сообщениями в консоль подфункций работает в штатном режиме.



Вывод в консоль текущих параметров подключения успешно.



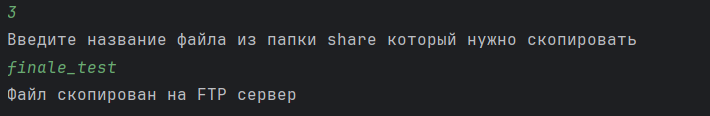
Вывод в консоль сообщений изменения параметров подключения успешен.



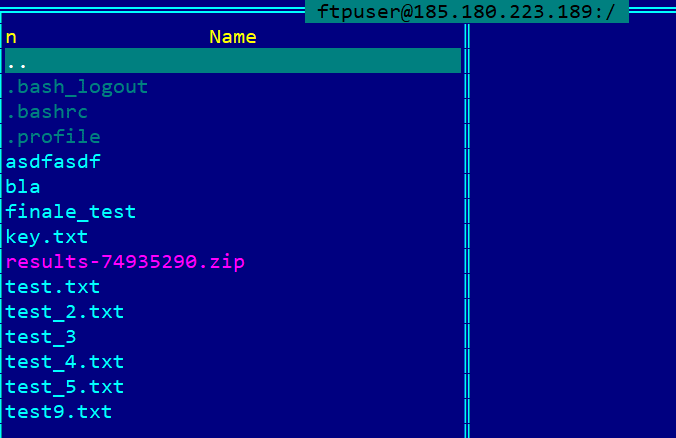
Сохранение параметров подключения, введённых пользователем успешно.

Функция “Просмотр параметров подключения по FTP протоколу” и подфункции полностью работоспособны.

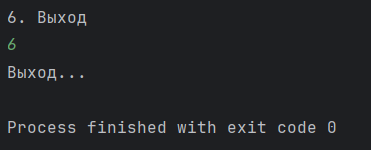
Для проверки функции “Отправить файл” необходимо вернуть стандартные параметры для подключения по “FTP” протоколу.



Вывод сообщений в консоль работает в штатном режиме. Для точной проверки работоспособности этой функции, необходимо просмотреть директорию “FTP” подключения, в рамках этого проекта, используется уже подключённый “FAR Manager”.



Функция “Отправить файл” полностью работоспособна.



Программа завершила свою работу, отработав в штатном режиме и доказав свою работоспособность.

# Заключение

1. Во время анализа практической базы, а после практического выполнения этого проекта, были получены новые и более углублённые знания в области сетевого перемещения файлов и данных.
2. Результат этого проекта полностью удовлетворяет его цели и поставленным задачам.
3. В дальнейшем этот проект можно развивать, приобретая новые познания и цели для его реализации.
4. Общим итогом является полностью работоспособное консольное приложение для перемещения файлов с физического компьютера на виртуальную машину по средствам “FTP” протоколов.
5. В дальнейшем это консольное приложение можно реализовать в компьютерное или “веб”. Реализовать подключение к серверу “SQL” для отслеживания перемещения файлов за всё время с указанием времени и файловых систем, а также реализовать “веб” составляющею для просмотра с любого устройства (при авторизации) истории перемещения файлов.

# Список используемой литературы

Nic.ru

Farmanager.com

Putty.org.ru

FirstVDS.ru

Lost.pro

Wiki.merionet.ru

Ubuntu.com

Skillbox.ru

ru.wikipedia.org

cnews.ru

roksis.ru

vdgb.ru

Kaspersky.ru

Habr.com

Picom.ru

Docks.oracle.com

Javarush.com

Codejava.net

Book2s.com

Infoworld.com

Jexp.ru

Ru.hexlet.io

Xpert.io

Geeksforgeeks.org

Ru.stackoverflow.com

# Приложения

Программный код этого проекта:

public class CertificationDiplomApplication {  
  
 public static void main(String[] args) {  
 Start.*go*();  
 }  
}

public class Start {  
 public static void go\_2() {  
 while (true) {  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.println("1. Просмотр текущих параметров подключения");  
 System.*out*.println("2. Изменение параметров подключения");  
 System.*out*.println("3. Вернуться в основное меню");  
  
 int choice\_2 = scanner.nextInt();  
 scanner.nextLine();  
  
 switch (choice\_2) {  
 case 1:  
 System.*out*.println("ip-адрес текущего подключения: " + FTP\_connect.*getIp*());  
 System.*out*.println("Имя пользователя текущего подключения: " + FTP\_connect.*getUser*());  
 System.*out*.println(("Пароль пользователя текущего подключения: ") + FTP\_connect.*getPsw*());  
 break;  
 case 2:  
 System.*out*.println("Введите новый ip-адрес");  
  
 String new\_ip = scanner.nextLine();  
 if (new\_ip == "") System.*out*.println("ip адрес не изменён");  
 else FTP\_connect.*setIp*(new\_ip);  
  
 System.*out*.println("Введите имя нового пользователя");  
  
 String new\_user = scanner.nextLine();  
 if (new\_user == "") System.*out*.println("Пользователь не изменён");  
 else FTP\_connect.*setUser*(new\_user);  
  
 System.*out*.println("Введите пароль нового пользователя");  
  
 String new\_psw = scanner.nextLine();  
 if (new\_psw == "") System.*out*.println("Пароль не изменён");  
 else FTP\_connect.*setPsw*(new\_psw);  
  
 System.*out*.println("Изменения сохранены");  
 break;  
 case 3:  
 return;  
 }  
 }  
 }  
  
  
 public static void go() {  
 Scanner scanner = new Scanner(System.*in*);  
  
 File dir = new File("C:/Users/user/Desktop/demo/share");  
 boolean created = dir.mkdir();  
 if (created)  
 System.*out*.println("Папка share создана");  
  
 ArrayList<File> file\_list = new ArrayList<File>(Arrays.*asList*(dir.listFiles()));  
  
 while (true) {  
 System.*out*.println("\nИнструкция пользования: \nДля выбора необходимой функции введите номер этой функции в консоль");  
 System.*out*.println("\nMenu:");  
 System.*out*.println("1. Создать файл");  
 System.*out*.println("2. Добавить файл");  
 System.*out*.println("3. Отправить файл");  
 System.*out*.println("4. Просмотр всех добавленных файлов");  
 System.*out*.println("5. Просмотр параметров подключения по FTP протоколу");  
 System.*out*.println("6. Выход");  
  
 int choice = scanner.nextInt();  
 scanner.nextLine();  
  
 switch (choice) {  
 case 1:  
 Create\_file.*create*(file\_list);  
 break;  
 case 2:  
 Add\_file.*add*(file\_list);  
 break;  
 case 3:  
 FTP\_connect.*push\_file*();  
 break;  
 case 4:  
 FileList.*show\_files*(file\_list);  
 break;  
 case 5:  
 Start.*go\_2*();  
 break;  
 case 6:  
 System.*out*.println("Выход...");  
 scanner.close();  
 return;  
 default:  
 System.*out*.println("Такой функции нет. Попробуйте ещё раз");  
 }  
 }  
 }  
}

public class Create\_file{  
 public static void create(ArrayList<File> f) {  
 System.*out*.println("Введите имя файла: ");  
 Scanner scanner = new Scanner(System.*in*);  
 String name = scanner.nextLine();  
 System.*out*.println("Добавьте в него текст или нажмите enter для пропуска");  
 String text = scanner.nextLine();  
 File f\_name = new File("C:/Users/user/Desktop/demo/share/" + name);  
 f.add(f\_name);  
  
 try {  
 boolean created = f\_name.createNewFile();  
 if (created)  
 System.*out*.println("Файл создан");  
 } catch (IOException ex) {  
 System.*out*.println("Файл не был создан");  
 }  
  
 try(FileWriter writer = new FileWriter(name, true))  
 {  
 writer.write(text);  
 writer.flush();  
 }  
 catch(IOException ex){  
 System.*out*.println("Текст не был записан в файл");  
 }  
 }  
  
}

public class Add\_file {  
 public static void add(ArrayList<File> file\_list

){  
 System.*out*.println("Введите путь к файлу: ");  
 Scanner scanner = new Scanner(System.*in*);  
 String f\_path = scanner.nextLine();  
 File add\_f = new File(f\_path);  
  
 if (add\_f.exists()) {  
 add\_f.renameTo(new File("C:/Users/user/Desktop/demo/share/" + add\_f.getName()));

file\_list.add(add\_f);  
 System.*out*.println("Файл найден и скопирован в папку share");  
 } else  
 System.*out*.println("Файл не найден");  
 }  
  
}

public class FTP\_connect {  
  
 public static String *ip* = "185.180.223.189";  
 public static String *user* = "ftpuser";  
 public static String *psw* = "1111";  
  
  
 FTPClient con = null;  
  
 public static String getIp() {  
 return *ip*;  
 }  
  
 public static void setIp(String ip) {  
 FTP\_connect.*ip* = ip;  
 }  
  
 public static String getUser() {  
 return *user*;  
 }  
  
 public static void setUser(String user) {  
 FTP\_connect.*user* = user;  
 }  
  
 public static String getPsw() {  
 return *psw*;  
 }  
  
 public static void setPsw(String psw) {  
 FTP\_connect.*psw* = psw;  
 }  
  
 public static void push\_file() {  
 FTPClient con = null;  
 Scanner scanner = new Scanner(System.*in*);  
 try {  
 con = new FTPClient();  
 con.connect(*ip*);  
 if (con.login(*user*, *psw*)) {  
 con.enterLocalPassiveMode();  
 con.setFileType(FTP.*BINARY\_FILE\_TYPE*);  
 try {  
 System.*out*.println("Введите название файла из папки share который нужно скопировать");  
 String file\_name = scanner.nextLine();  
 String data = "C:/Users/user/Desktop/demo/share/" + file\_name;  
 FileInputStream in = new FileInputStream(new File(data));  
 boolean result = con.storeFile("/" + file\_name, in);  
 in.close();  
 } catch (FileNotFoundException e) {  
 System.*out*.println("Файл не найден");  
 }  
 }  
 con.logout();  
 con.disconnect();  
 System.*out*.println("Файл скопирован на FTP сервер");  
 } catch (Exception e) {  
 System.*out*.println("Подключение не удалось");  
 }  
 }  
}

public class FileList {  
 public static void show\_files(ArrayList<File> f\_list) {  
 for (File f : f\_list) {  
 System.*out*.println("Имя файла: " + f.getName() + ", путь к этому файлу: " + f);  
  
 }  
 }  
}