МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ

КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Математическая кибернетика и информационные технологии»

Лабораторная работа №0

**Основы работы с системой контроля версий**

**по дисциплине**

**«Информационные технологии и программирование»**

Выполнил: студент гр. БВТ2201

Аблязов И.В.

Проверил:

Москва, 2023 г.

# ВВЕДЕНИЕ

Цель работы: изучение базовых функций и подходов к командной разработке в системе контроля версий Git.

Задание: определить проект (состоящий из нескольких файлов), который будет добавлен под версионный контроль, и произвести ряд операций в СКВ, включающих:

1. Создание репозитория или клонирование существующего
2. Внесение ряда изменений в файл(ы) и выполнение коммитов (кол-во не меньше 2)
3. Откат изменений до одного из предыдущих коммитов двумя разными способами
4. Создание, переименование и удаление ветки
5. Слияние ветки с веткой мастер с решением конфликта содержания файлов
6. Настройка игнорирования git’ом файлов определенного расширения с использованием glob-шаблонов
7. Вывод лога коммитов в измененном пользовательском формате
8. Настройка удалённого репозитория: получение и отправка изменений в репозиторий
9. Организация совместной работы над проектом: клонирование чужого репозитория, выполнение коммитов, отправка изменений

# ОСНОВНАЯ ЧАСТЬ

Для начала зададим наши данные в гит. Это можно сделать при помощи команды git config –global. Так мы вводим имя и почту как показано на рисунке 1.

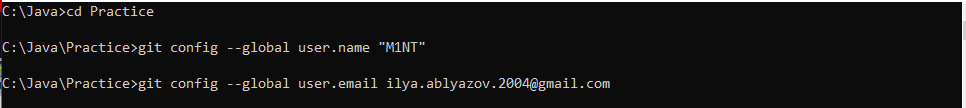


Рисунок 1 Ввод личных данных

Далее в папке нашего проекта мы инициализируем git командой git init.

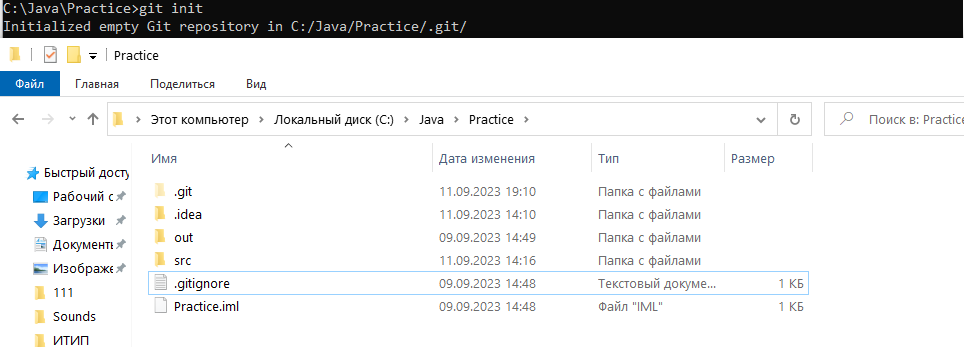


Рисунок 2 Инициализация git

Теперь с помощью команды git status мы можем узнать о состоянии файлов в данном репозитории (отслеживаются ли они или нет). Отслеживание файлов происходит с помощью команды git add <название файла>. Таким образом мы выбираем все необходимые нам файлы.

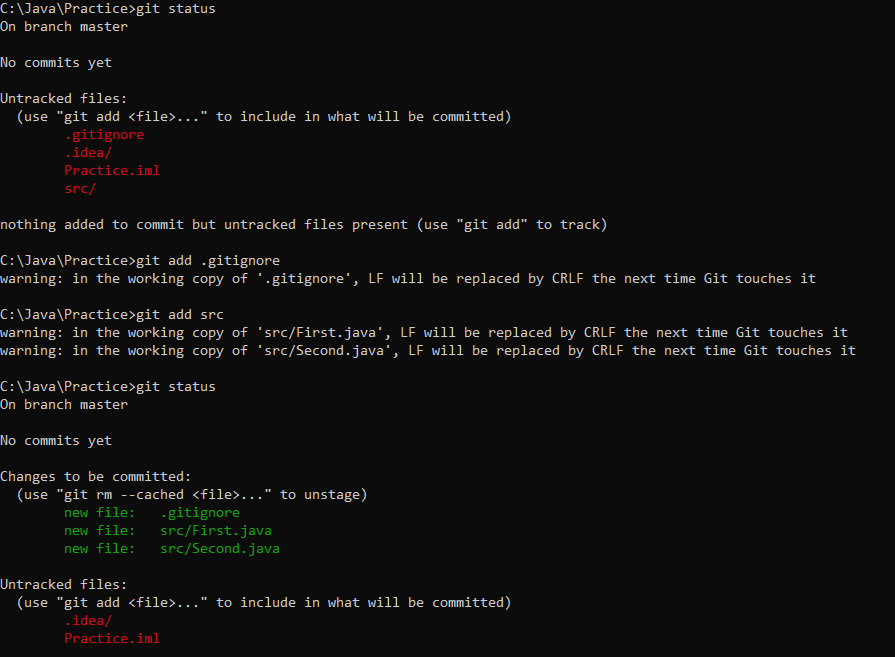


Рисунок 3 Отслеживание файлов

После этого мы можем сохранить наши изменения командой git commit –m “комментарий”.

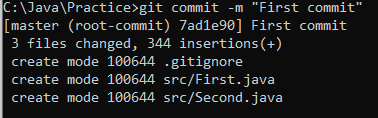


Рисунок 4 Первый коммит

После очередных изменений в нашем проекте мы должны заново прописать команду git add на те файлы, что были изменены. После чего мы можем сохранить изменения в виде нового коммита.

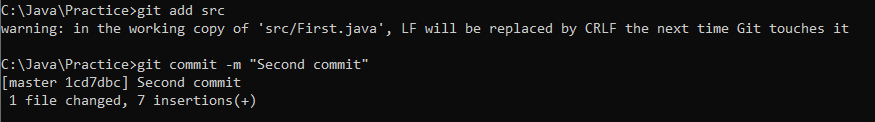


Рисунок 5 Второй коммит

Так мы можем создавать любое количество коммитов. Список всех коммитов можно изучить командой git log. Она также покажет хеш кажого коммита.

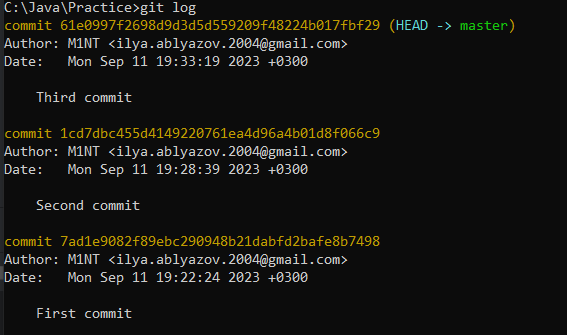


Рисунок 6 Список коммитов

Если нам необходимо откатить изменения, совершенные последним коммитом, мы можем воспользоваться командой git revert HEAD. Она воссоздаст новый коммит, основанный на предпоследнем.

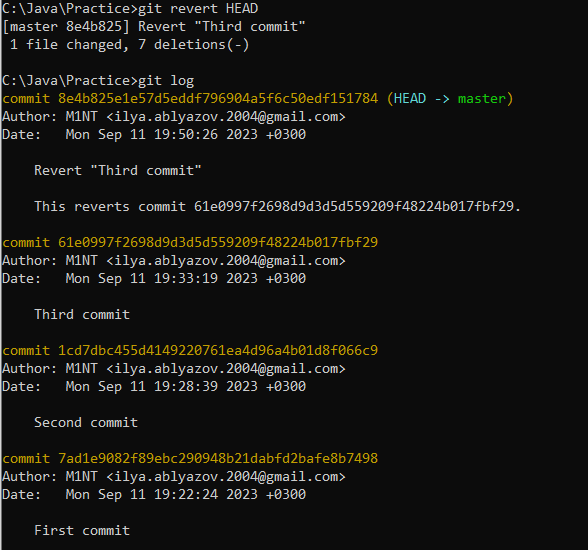


Рисунок 7 Откат последнего коммита

Если же нам необходимо вернуться к определенному коммиту, удаляя все произведенные после него изменения, существует команда git reset <hash коммита>. На рисунке 8 изображен откат до самого первого коммита.

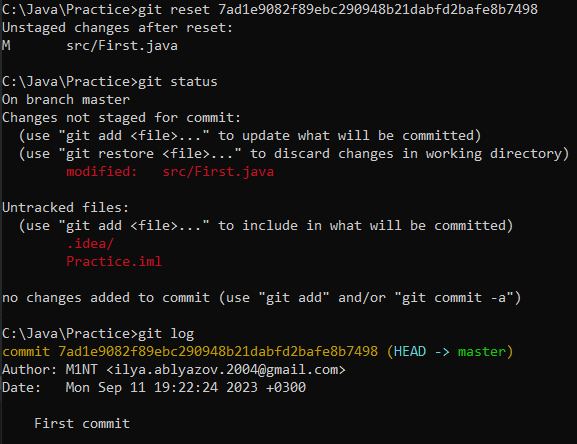


Рисунок 8 Откат к коммиту с удалением изменений

Также git позволяет создавать новые ветки командой git branch <название ветки>.

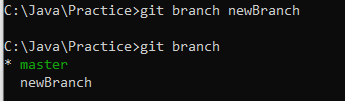


Рисунок 9 Создание новой ветки

Командой git branch –m <старое имя ветки> <новое имя ветки> можно изменить название ветки.

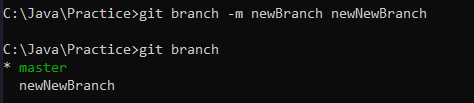


Рисунок 10 Переименование ветки

Удаление ветки происходит командой git branch –d <название ветки>.

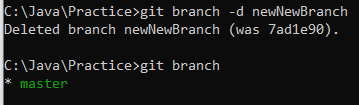


Рисунок 11 Удаление ветки

Теперь представим ситуацию, что мы имеем код, показанный на рисунке 12.



Рисунок 12 Изначальный код

После этого мы внесли изменения в ветке master как показано на рисунке 13.



Рисунок 13 Изменение в master ветке

После чего мы также внесли изменения в другой ветке как показано на рисунке 14.

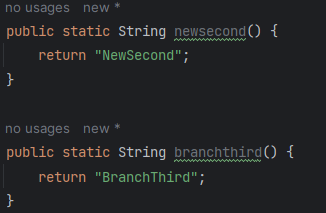


Рисунок 14 Изменение в другой ветке

Для того, чтобы переходить по веткам используется команда git checkout <название ветки>.

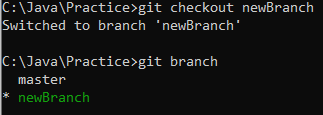


Рисунок 15 Переключение к другой ветке

Проследить разницу в ветках возможно командой git diff <название первой ветки> <название второй ветки>.

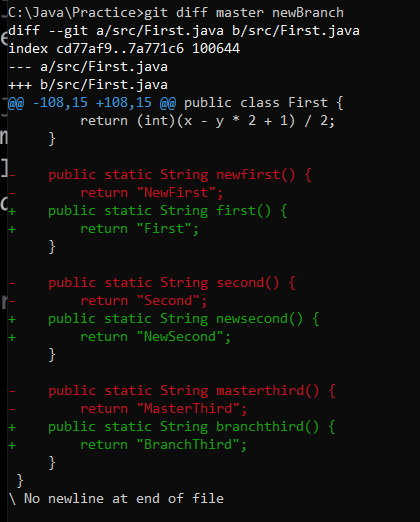


Рисунок 16 Различия в ветках

Слияние веток происходит командой git merge <название ветки> при нахождении в ветке, с которой будет происходить слияние.

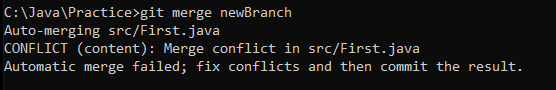


Рисунок 17 Слияние ветки newBranch с веткой master

Как можно увидеть из рисунка 17, у нас возник конфликт при слиянии веток. Перейдя в конфликтный файл, мы увидим пометки git’a.

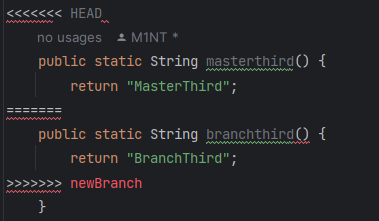


Рисунок 18 Конфликт

В пункте HEAD указано изменения основной ветки, а уже после изменения сливаемой ветки. Мы можем самостоятельно выбрать, какое из изменений нам подходит. Для этого прописывается команда git checkout --ours или --theirs <название конфликтного файла>. При выборе --ours, слияние оставит изменения ветки, в которую пытаются слить другую ветку. При выборе --theirs – противоположная ситуация.

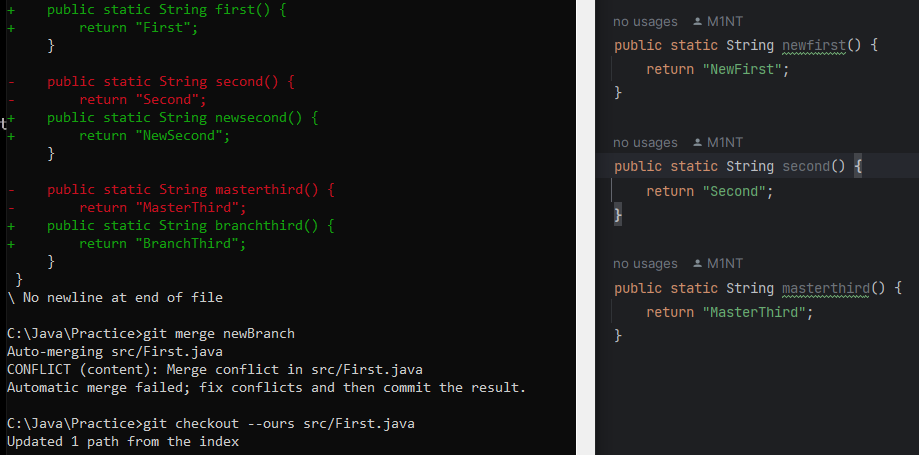


Рисунок 19 Выбор нужного изменения

После всех манипуляций необходимо провести новый коммит.

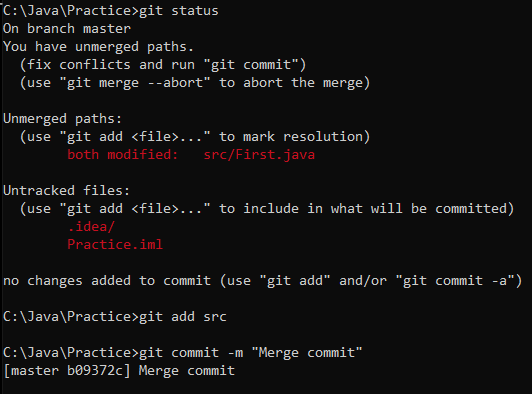


Рисунок 20 Коммит слияния двух веток

Теперь заставим git игнорировать лишние файлы. Для этого в .gitignore мы прописываем игнорирование ненужных нам файлов. В данном случае – папка .idea и файл расширения .iml.

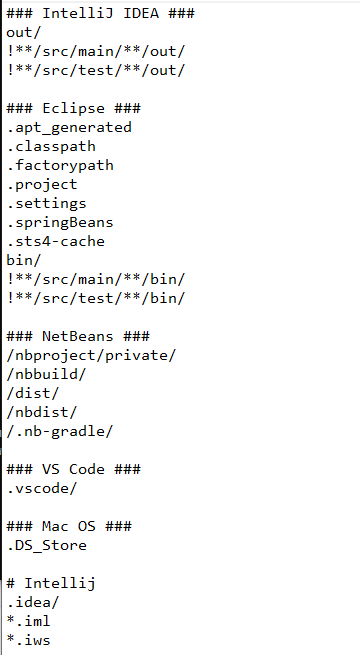


Рисунок 21 .gitignore

Помимо обычного вывода списка коммитов, мы можем выводить только необходимую для нас информацию

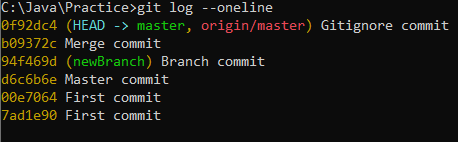


Рисунок 22 Пользовательский список коммитов

Когда все необходимые задачи с локальным репозиторием были выполнены, мы можем залить наш проект на удаленный репозиторий. Создав удаленный репозиторий на GitHub, копируем ссылку на него и привязываем ее к git командой git remote add origin <ссылка>. После этого все сохранения передаются на удаленный репозиторий командой git push origin master.

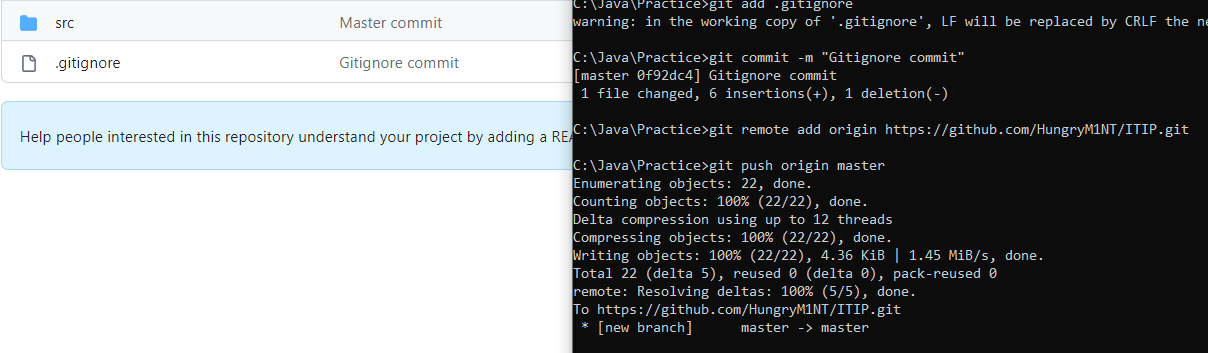


Рисунок 23 Отправка файлов на удаленный репозиторий

Если же у нас появилась необходимость клонировать чужой репозиторий, прописываем команду git clone <ссылка>.

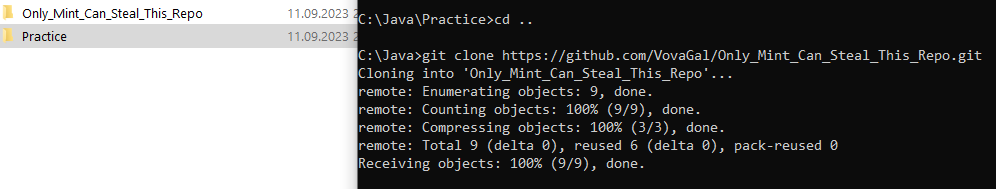


Рисунок 24 Клонирование репозитория

После чего мы можем работать с клонированным проектом и создавать новые коммиты.

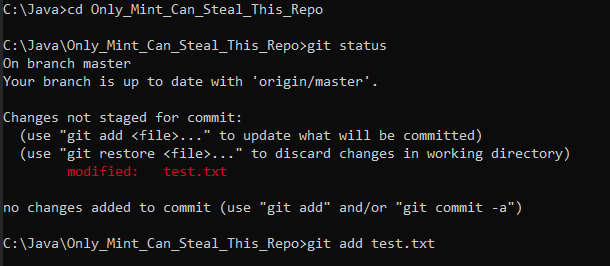


Рисунок 25 Создание коммита в клонированном репозитории

После внесения и сохранения изменений мы можем залить их в тот же удаленный репозиторий командой git push origin mater. Только в данном случае нам необходимо получить разрешение от владельца репозитроем.

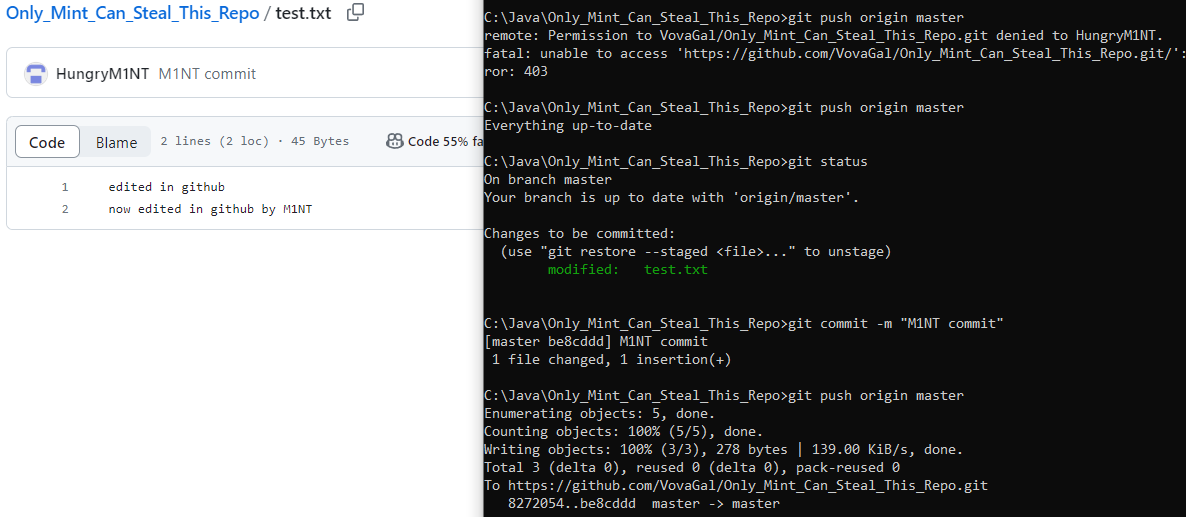


Рисунок 26 Загрузка изменений на чужой удаленный репозиторий

# ВЫВОД

Данная лабораторная работа помогла лучше освоить git. Помимо обычного создания локального репозитория и переноса изменений на удаленный репозиторий была проделана работа с коммита и ветками. Также была затронута работа с клонированием чужого репозитория, что очень необходимо при работе в команде. Все знания были закреплены на практике и сохранены на GitHub: <https://github.com/HungryM1NT/ITIP>