**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ**

**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

Выполнил: Студент группы

БВТ2201

Бутарасов Дмитрий

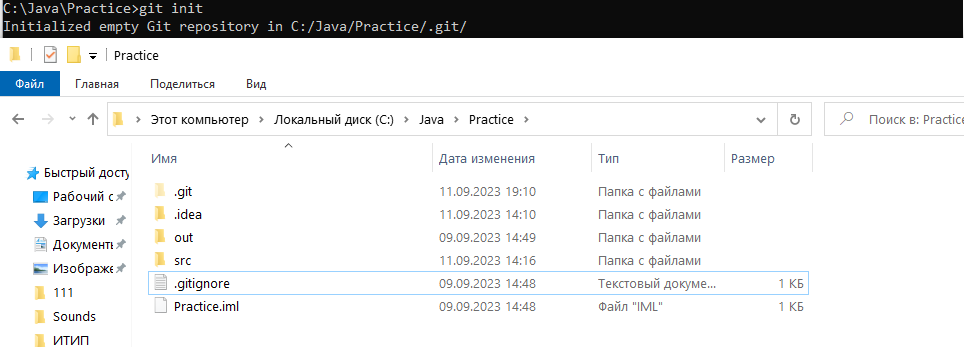
Алексеевич

Москва

2023

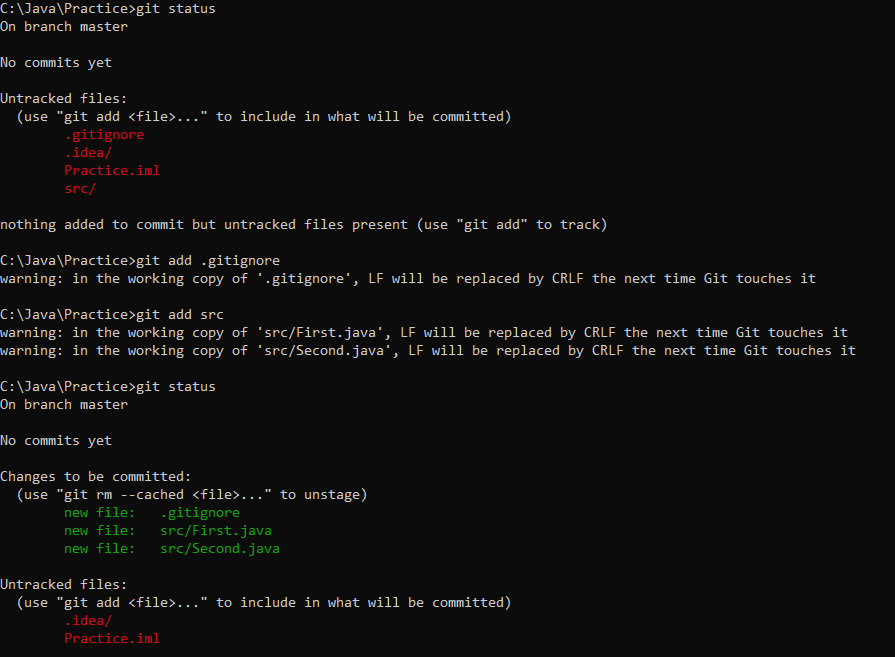
Для начала зададим наши данные в гит. Это можно сделать при помощи команды git config –global. Так мы вводим имя и почту как показано на рисунке

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



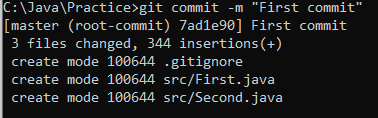
*Инициализация git*

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.



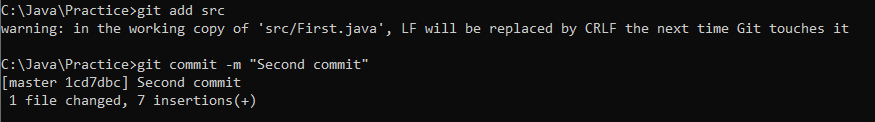
*Отслеживание файлов*

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



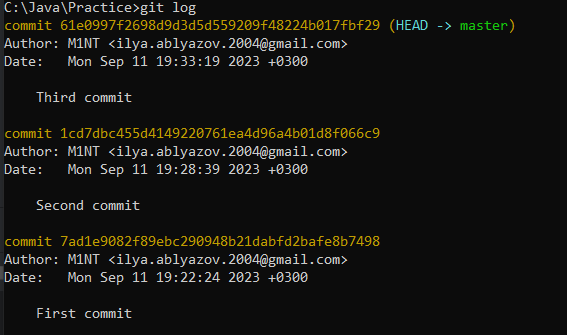
*Первый коммит*

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.



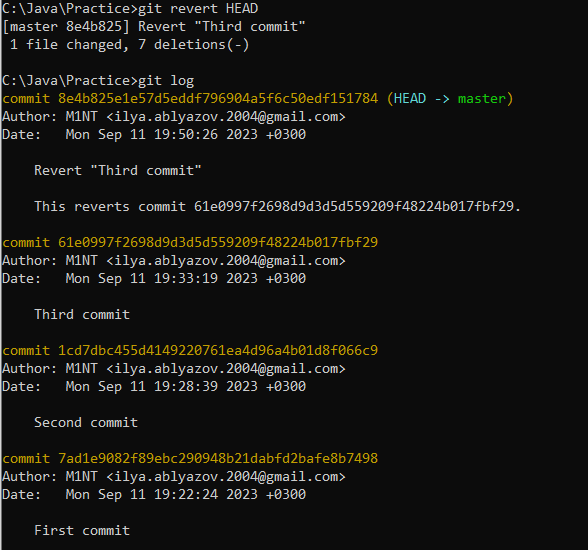
*Второй коммит*

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



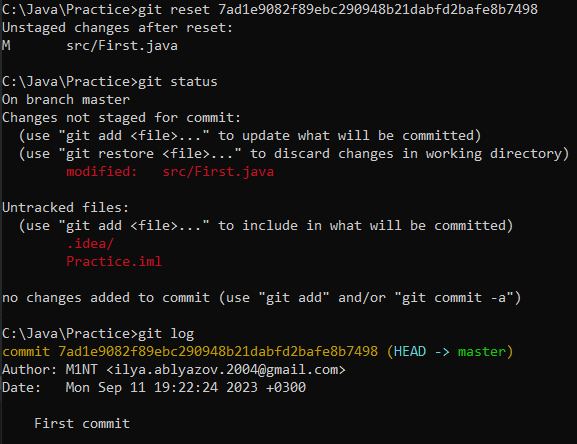
*Список коммитов*

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



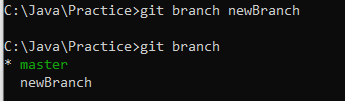
*Откат последнего коммита*

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.



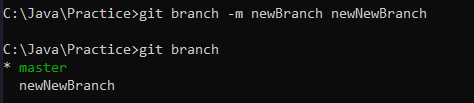
*Откат к коммиту с удалением изменений*

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



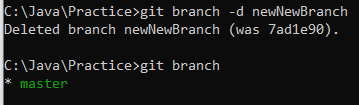
*Создание новой ветки*

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



*Переименование ветки*

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



*Удаление ветки*

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



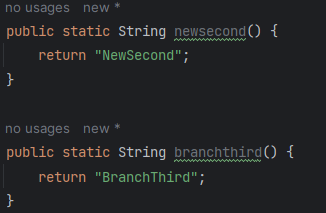
*Изначальный код*

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



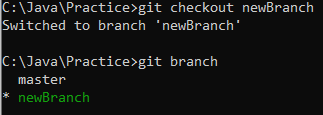
*Изменение в master ветке*

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



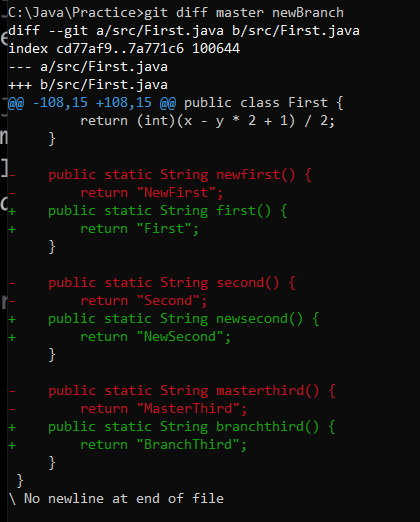
*Изменение в другой ветке*

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



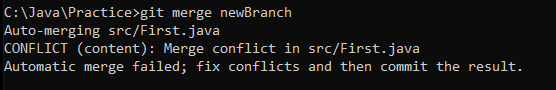
*Переключение к другой ветке*

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



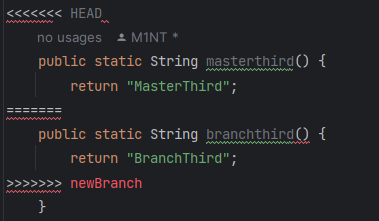
*Различия в ветках*

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



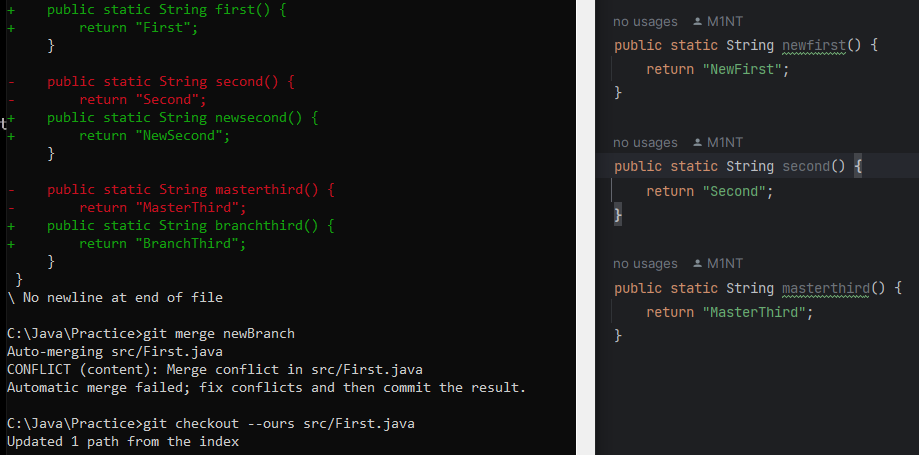
*Слияние ветки newBranch с веткой master*

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



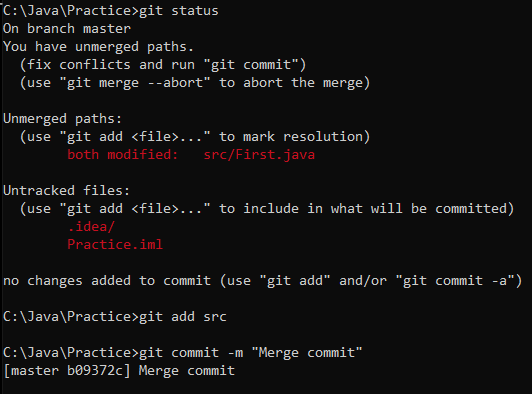
*Конфликт*

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.



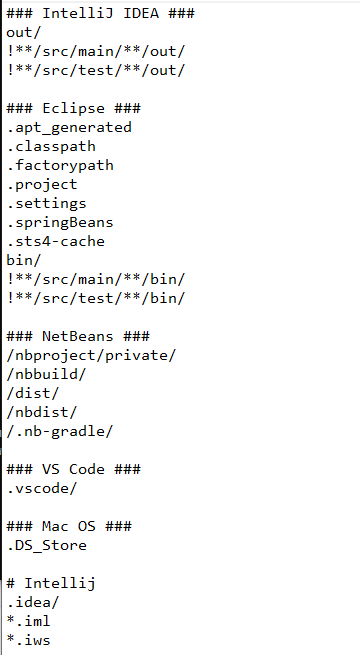
*Выбор нужного изменения*

После всех манипуляций необходимо провести новый коммит.

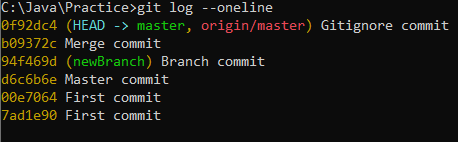


*Коммит слияния двух веток*

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.

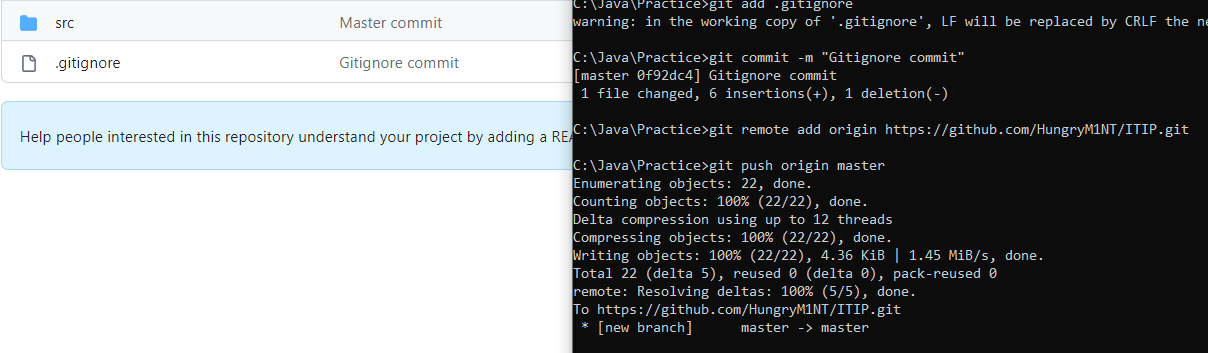


Помимо обычного вывода списка коммитов, мы можем выводить только необходимую для нас информацию



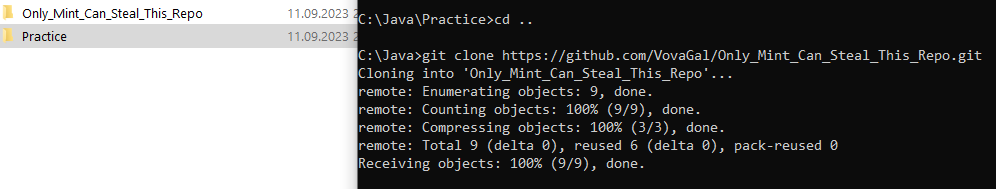
*Пользовательский список коммитов*

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.



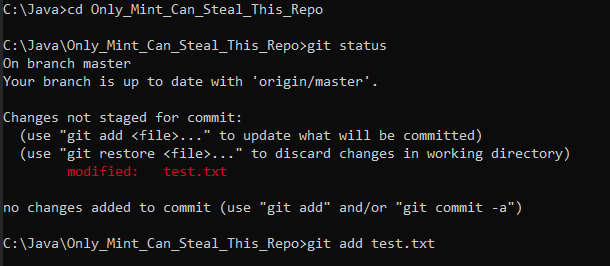
*Отправка файлов на удаленный репозиторий*

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



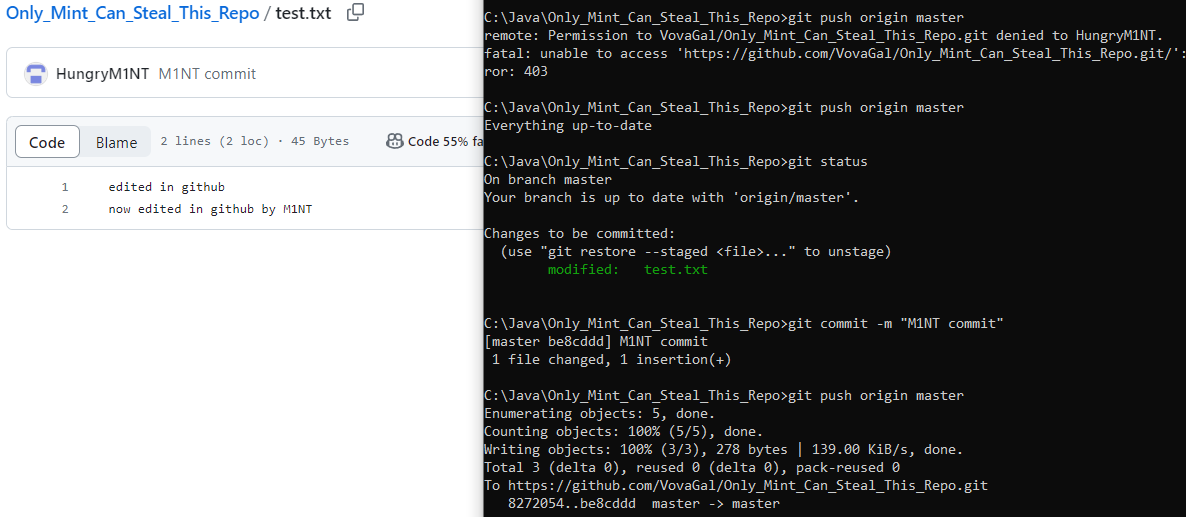
*Клонирование репозитория*

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



*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.



*Загрузка изменений на чужой удаленный репозиторий*

https://github.com/G1QSTAR/Laba-itip.git