**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**

Факультет РиТ

**ЛАБОРАТОРНАЯ РАБОТА №1**

по дисциплине «Java-программирование»

на тему:

«Основы работы с системой контроля версий»

Выполнил: студ. гр. БПЗ1901

Неживлева Ксения

Проверил: Харрасов Камиль Раисович

Москва 2021

**Цель работы**: изучение базовых функций системы контроля версий на примере 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. Создание веток и выполнение ряда изменений в проекте в созданных ветках

4. Слияние ветки с веткой мастер без конфликтов

5. Слияние ветки с веткой мастер с решением возникающих конфликтов

6. Откат изменений до одного из предыдущих коммитов

**Перечень основных используемых команд ($ git …):**

git init – создание нового репозитория

git add – добавление указанных файлов в индекс

git status - отображение список измененных, добавленных и удаленных файлов git merge – слияние веток, разрешение возможных конфликтов

git branch - создание, перечисление и удаление веток

git checkout - переключение между ветками, извлечение файлов

git diff - отличия между деревьями проекта, коммитами и т.д.

git commit – фиксирование добавленных в индекс изменений,  совершение коммита

git log - разнообразная информация о коммитах в целом

git clone- создание копии (удаленного) репозитория

git fetch-забирает изменения удаленной ветки из репозитория по умолчания, основной ветки

git pull-забирает изменения и проводит слияние с активной веткой

git push - вносим изменения в удаленный репозиторий

git reset - возврат к определенному коммиту, откат изменений

**Ход работы:**

1. Создаем директорию (mk dir laba) и перейдем в нее (cd laba) при помощи команд в командной строке.
2. Создаем репозиторий.

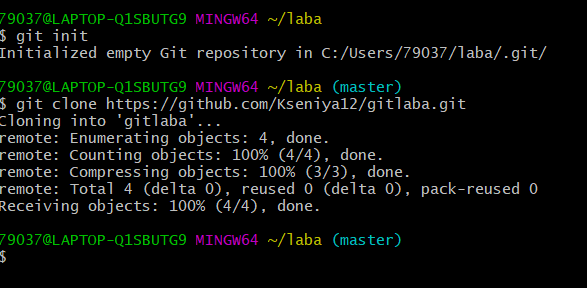


Рис.1 – Создание репозитория

1. Добавляем содержимое рабочей директории в индекс для последующего коммита (git add).
2. Проверим состояние файлов в рабочей директории и индексе (git status), совершим коммит(git commit –m “Add Main” и проверим информацию об истории коммитов(git log).

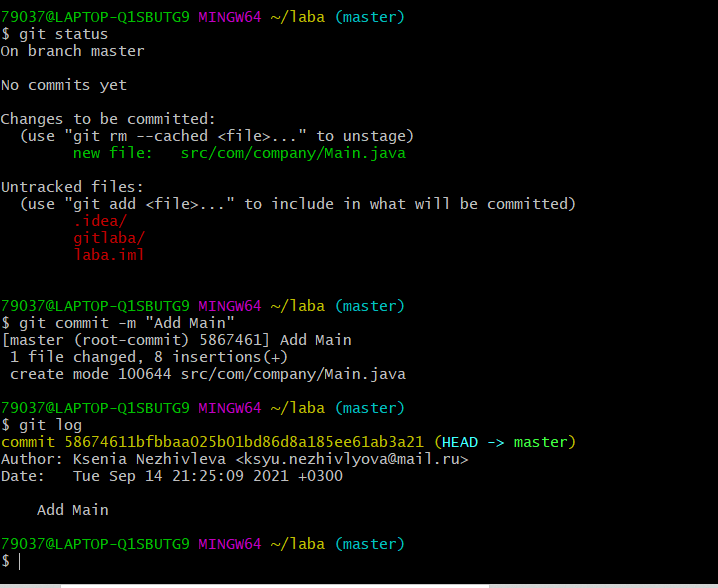


Рис. 2 – Иллюстрация 4 пункта

1. Повторим операцию и аналогично создадим второй коммит.
2. Создадим еще один файл func2.java и закоммитим его.

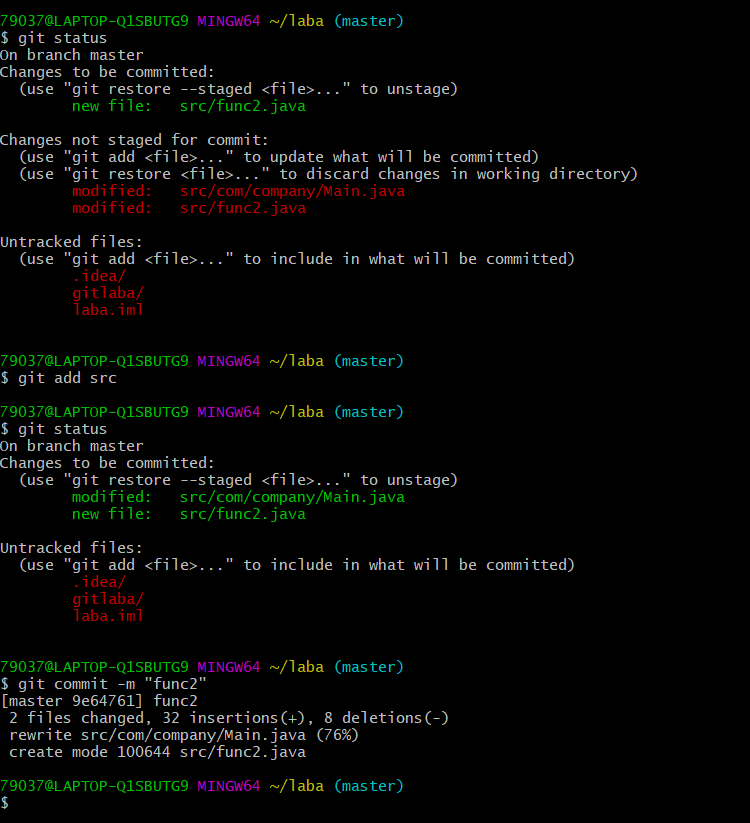


Рис. 3 – Иллюстрация 6 пункта

1. Проверим информацию об истории коммитов.



Рис.4 - История коммитов

1. Создадим новую ветку (git checkout –b branch2).

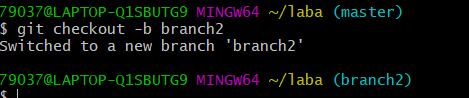


Рис.5 - Создание ветки

1. Внесем изменения в проекте, находясь на второй ветке.

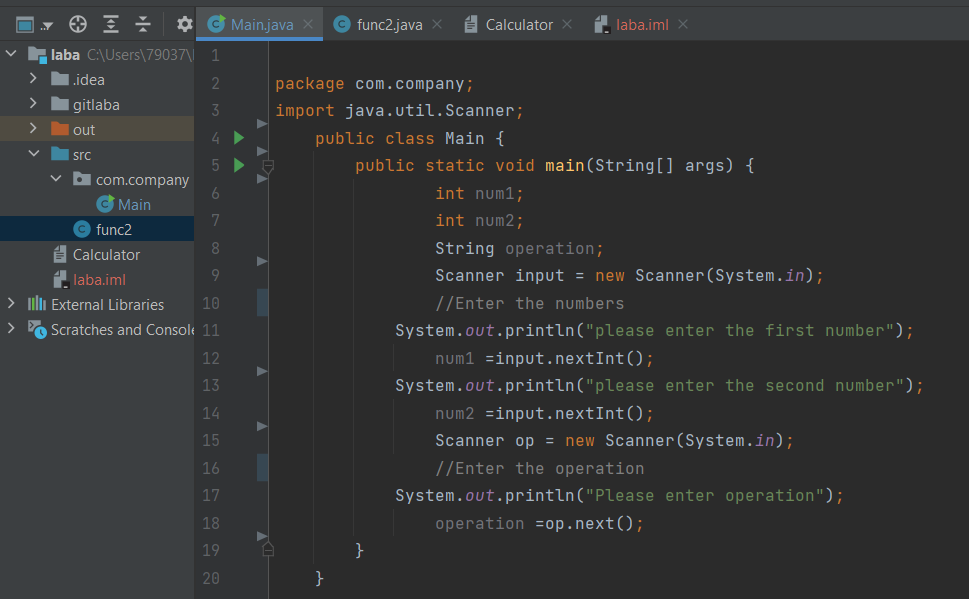


Рис.6 - Внесение изменений в коде

1. Закоммитим измененные файлы на второй ветке (git commit –m “Add new comments”), затем повторим операцию и аналогично создадим еще одну ветку (git checkout –b branch3). Внесем изменения в проекте на новой ветке и закоммитим файлы.
2. Вернемся на главную ветку и сольем ее с 3 веткой без конфликтов.

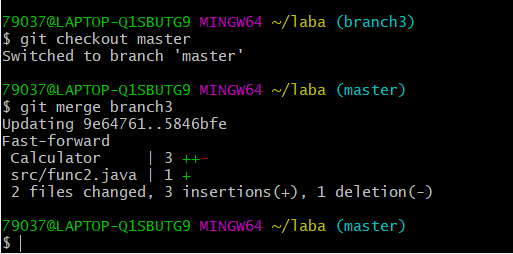


Рис.7 – Слияние веток без конфликтов

1. Добавим наши файлы в удаленный репозиторий (git push gitlaba)

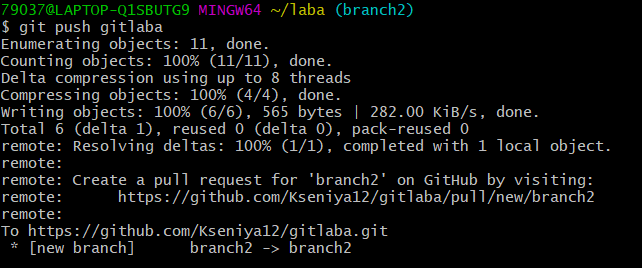


Рис.8 – Добавление ветки 2

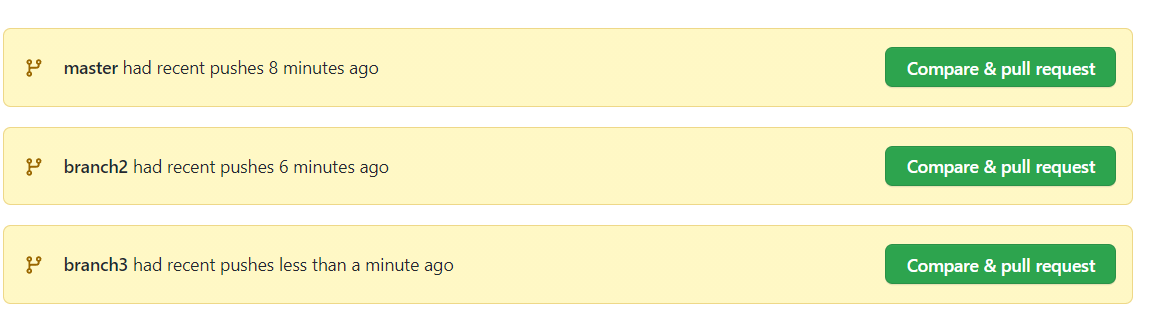


Рис. 9 – Ветки успешно добавлены

1. Сольем ветки, решая возникшие конфликты в удаленном репозитории. Изменим код у ветки, чтобы возник конфликт при слиянии. При возникновении конфликта запросим разрешение на слияние, решим возникшие конфликты и сольем ветки.

Аналогично это можно сделать в командной строке, применяя git checkout master и решая возникшие конфликты:

Произошла неудачная попытка слияния:

git merge

Смотрим на проблемные места:

git status

Разрешаем проблемы:

edit TROUBLE

Индексируем наши изменения, тем самым снимая метки:

git add .

Совершаем коммит слияния: git commit

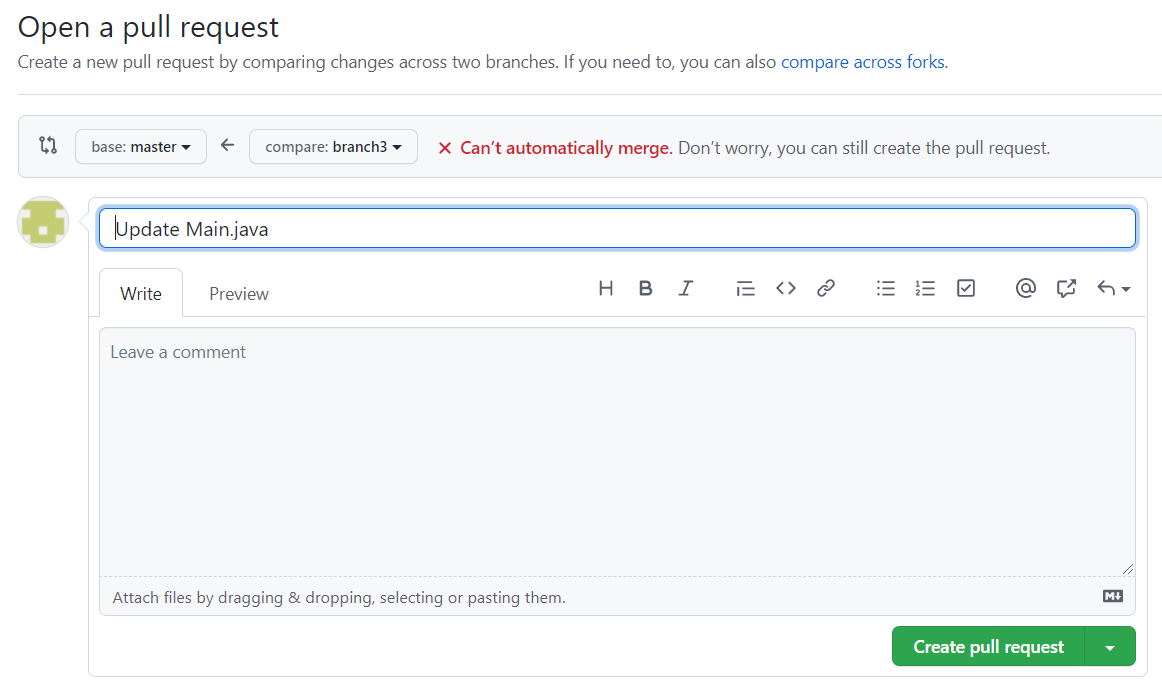


Рис.10 - Запрос на слияние



Рис.11 - Решение конфликта

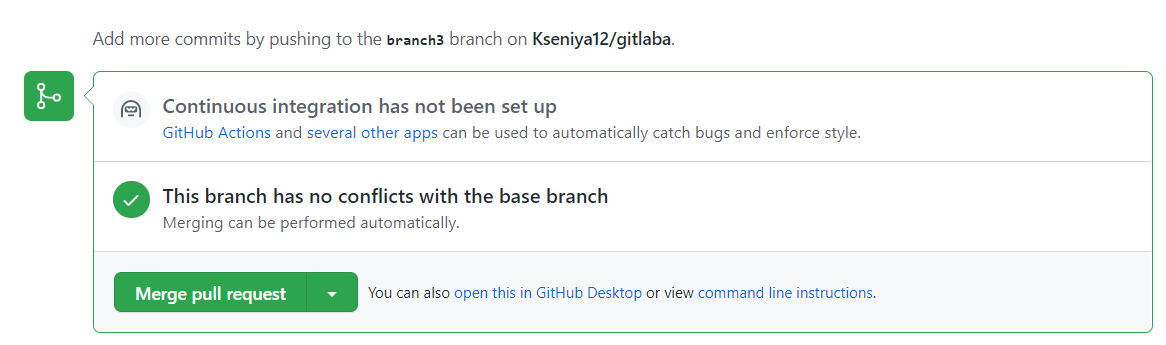


Рис.12 - Слияние веток

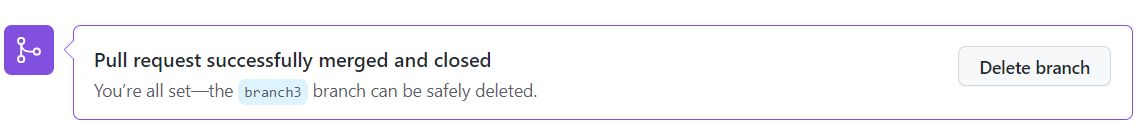


Рис. 13 – Сообщение об успешном слиянии веток

1. Создадим новую ветку, создадим коммит, затем откатимся назад при помощи команды (get reset --hard HEAD)



Рис.14 – Откат изменений

**Выводы:**

В данной работе были изучены базовые функции системы контроля версий на примере Git. Git - это консольная утилита, для отслеживания и ведения истории изменения файлов, в нашем проекте. С помощью Git-a мы можем откатить свой проект до более старой версии, сравнивать, анализировать или сливать свои изменения в репозиторий. Репозиторием называют хранилище нашего кода и историю его изменений. Git работает локально и все наши репозитории хранятся в определенных папках на жестком диске. Так же наши репозитории можно хранить и в интернете.

**Литература:**

1. Камаев В.А., Костерин В.В. Технологии программирования. М.: Высшая школа,

2006.

1. Жоголев Е.А.Технология программирования. – М.: Научный мир, 2004.

Git:

1. Scott Chacon, Ben Straub «Pro Git»
2. [git-scm.com](https://git-scm.com/)