

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего
профессионального образования

«Вятский государственный университет»

(ФГБОУ ВО «ВятГУ»)

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Разработка программных систем

Организация процесса разработки ПО с применением системы контроля версий

Вариант 4

Выполнил студент группы ИВТ-31 _____/Кудяшев Я.Ю./

Проверил преподаватель _____/Чистяков Г.А./

Киров 2022

1. Задание

Для выполнения лабораторной работы необходимо решить следующие задачи:

- Установить распределенную систему контроля версий Git
- Зарегистрировать аккаунт на GitHub
- Синхронизировать репозиторий с Git
- Добавить ссылку на созданный репозиторий в IntelliJ IDEA
- Изменить функционал проекта
- Обновить файлы в репозитории
- Откатить сделанные изменения с помощью информации из репозитория
- Создать побочную ветвь проекта и произвести конфликтующие изменения в стволовой и побочной ветвях
- Разрешить возникший конфликт посредством встроенных инструментов IntelliJ IDEA

2. Ход работы

2.1. Установка Git

Git был установлен как отдельно ПО. Работа с репозиторием велась через интерфейс командной строки и IntelliJ IDEA.

2.2. Регистрация аккаунта на GitHub

В качестве репозитория используется крупнейший веб-сервис для хостинга IT-проектов GitHub. После регистрации аккаунта можно загружать проект в репозиторий. Профиль пользователя созданного аккаунта приведен на рисунке 1.

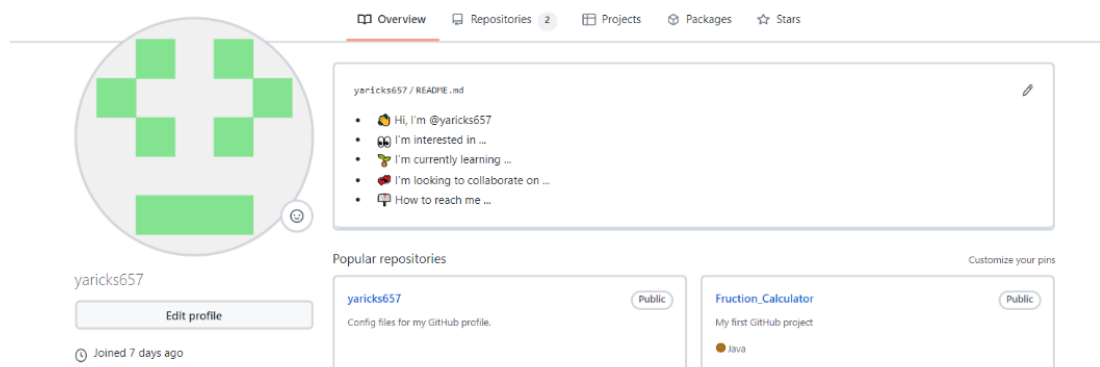
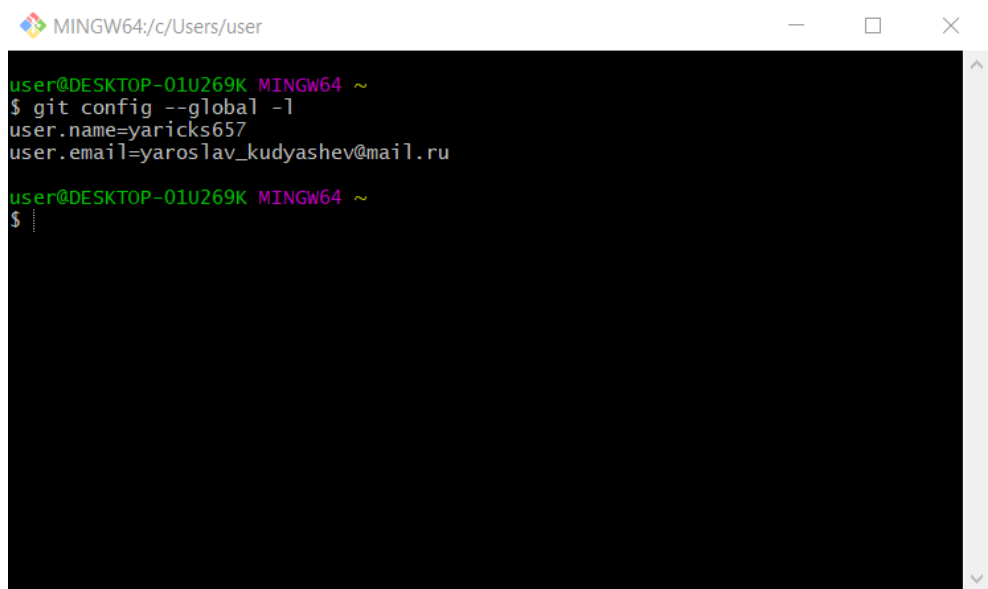


Рисунок 1 – Профиль пользователя

2.3. Синхронизация репозитория с Git и добавление ссылки на созданный репозиторий в IntelliJ IDEA

Для синхронизации аккаунта GitHub с Git необходимо прописать команды «*git config --global user.name ...*» и «*git config --global user.email ...*», благодаря которым можно будет загружать проекты напрямую в GitHub. Пример ввода команды приведен на рисунке 2.

A screenshot of a terminal window titled "MINGW64:/c/Users/user". The terminal shows the following commands and output:

```
user@DESKTOP-01U269K MINGW64 ~  
$ git config --global -l  
user.name=yaricks657  
user.email=yaroslav_kudyashev@mail.ru  
  
user@DESKTOP-01U269K MINGW64 ~  
$
```

Рисунок 2 – Проверка введенных данных в Git

После проделанных в Git действий было необходимо синхронизировать Git со средой IntelliJ IDEA и добавить аккаунт. Добавление аккаунта осуществлялось путём нажатия комбинации клавиш «Ctrl + Alt + S» и выбора вкладки GitHub. При добавлении аккаунта было необходимо ввести данные от своей учётной записи. В результате проделанных действий аккаунт стал привязан к среде. Результат показан на рисунке 3.

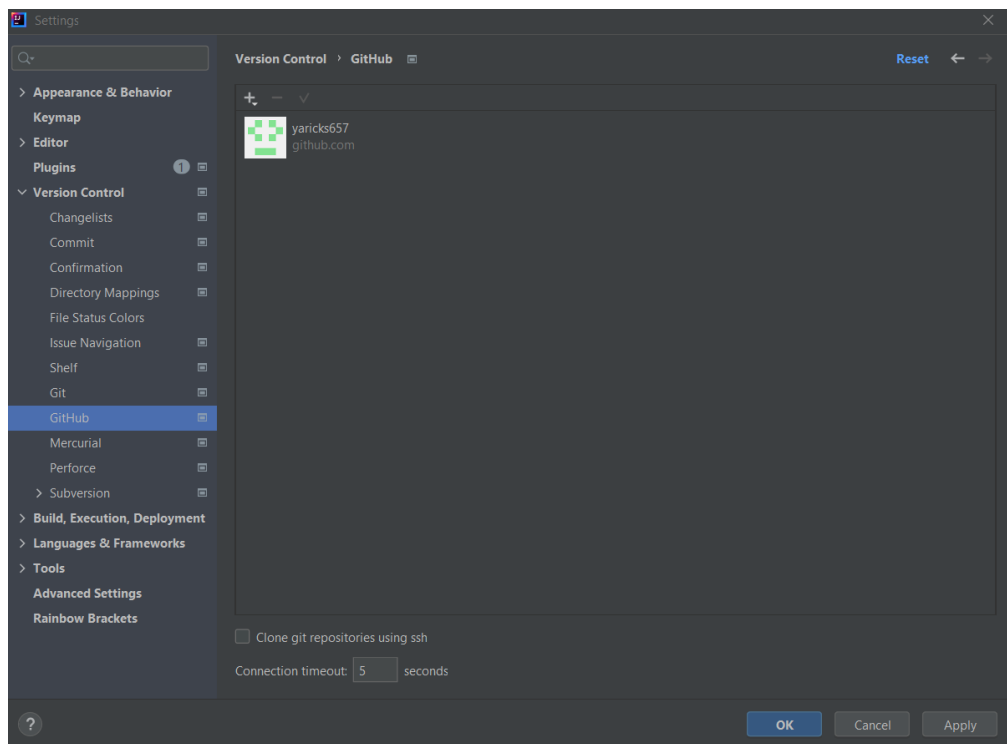


Рисунок 3 – Синхронизированный аккаунт GitHub

2.4. Обновление файлов в репозитории

Были сделаны изменения в классе `BigFractionCalculatorGUI`: закомментирована функция, отвечающая за отображение некоторых кнопок калькулятора. Если класс был изменен, то он фиксируется синим цветом в левой панели, что можно видеть на рисунке 4. После изменений необходимо было закоммитить изменения и загрузить обновленную версию проекта на GitHub путем нажатия на кнопки `Commit` и `Push`. Пример загрузки проекта приведен на рисунке 5.

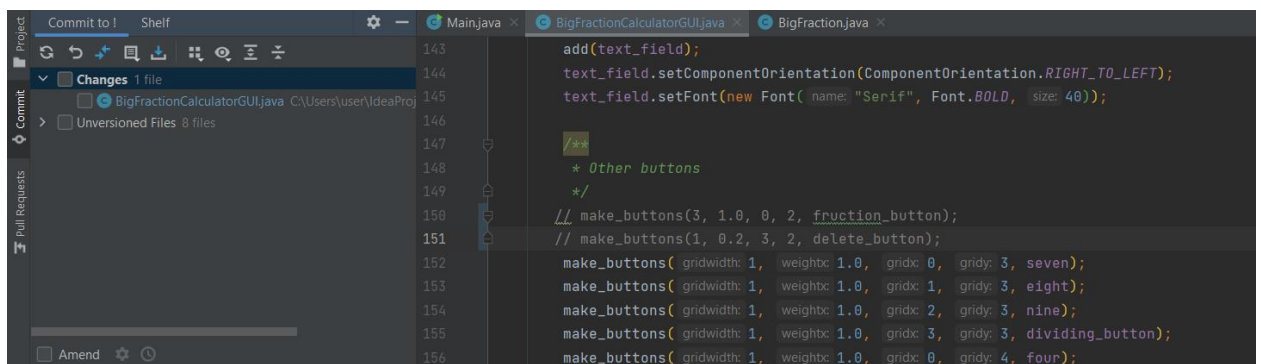


Рисунок 4 – Пример фиксации изменения класса

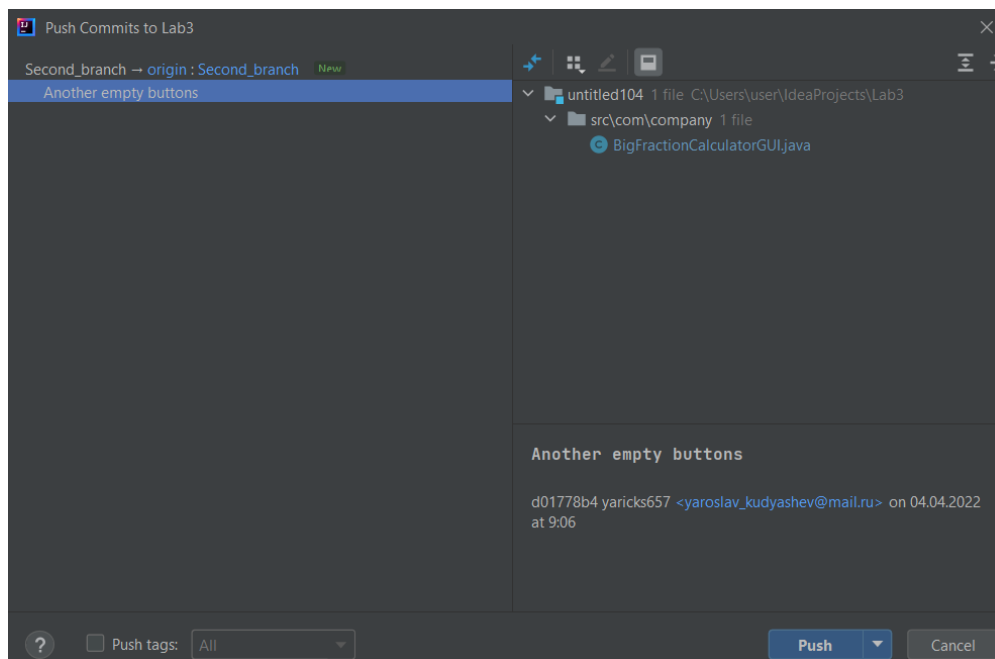


Рисунок 5 – Пример загрузки проекта на GitHub

После проделанных действий в GitHub появится обновленная версия проекта. В этом можно убедиться благодаря рисунку 6.

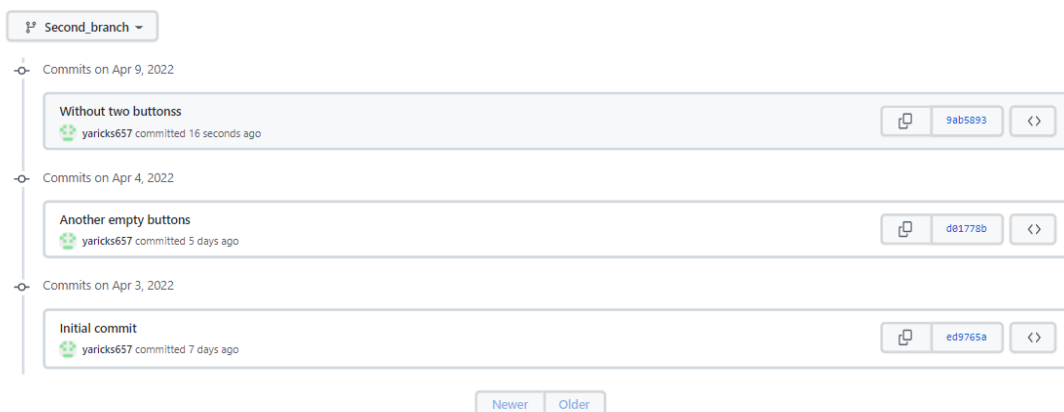


Рисунок 6 – Новая версия проекта на GitHub

2.5. Откат действий и возврат к предыдущей версии проекта

Благодаря встроенным средствам IntelliJ IDEA можно легко перейти к любой версии проекта на любой ветке. Сначала нужно открыть панель Git, в которой отображены все ветки проекта и изменения, а затем нажать на нужную версию проекта правой кнопкой и мыши и выбрать вкладку «Checkout Revision». Пример приведен на рисунке 7. После проделанных действий будет загружена нужная нам версия, в данном случае – исходная версия проекта.

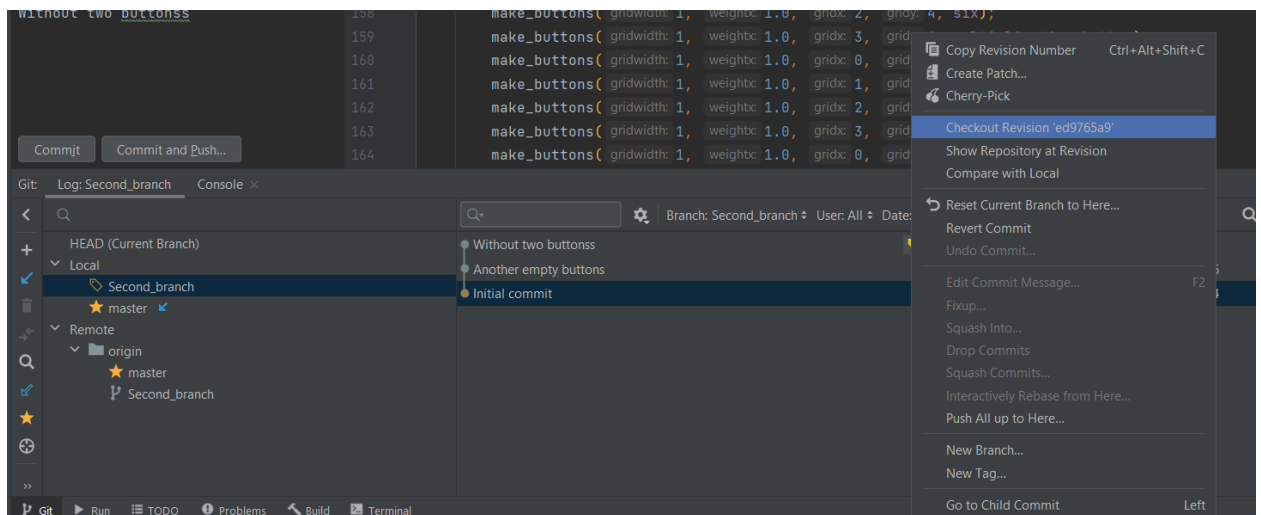


Рисунок 7 – Загрузка исходной версии проекта

2.6. Разрешения конфликта между ствовой и побочной ветвями

В качестве примера была создана побочная ветка, которая содержала изменения, связанные с кнопками калькулятора: были закомментированы все кнопки. Из-за этого возник конфликт версий, т.к. был в одной ветке функция не закомментирована, а в другой наоборот. IntelliJ IDEA имеет средство для решения данных проблем под названием Merge Revision, рисунок 8. Благодаря данному инструменту проблема была решена путём закомментирования строк в исходной ветке, красные маркеры на рисунке 8. Затем измененный проект был отправлен в репозиторий.

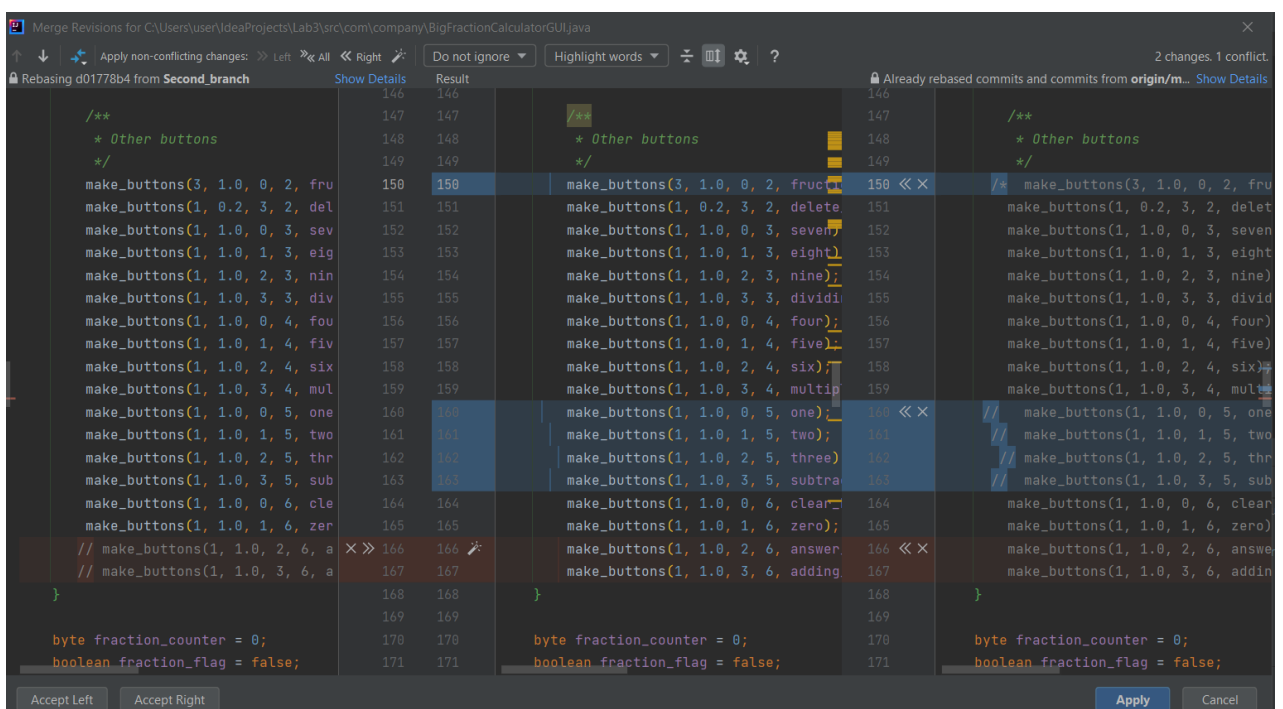


Рисунок 8 – Средство для решения проблем, возникших на разных ветках

3. Вывод

В ходе выполнения лабораторной работы был изучен базовый функционал Git для работы с системой контроля версий. Используя проект, созданный в ходе предыдущей лабораторной работы, были отработаны следующие сценарии работы с репозиторием: подключение удалённого репозитория к локальному проекту и загрузка исходных кодов, фиксация изменений, откат сделанных изменений, отправка изменений в репозиторий, просмотр истории версий, решение конфликтов ветвей.