Лабораторная работа № 2

по дисциплине “Объектно ориентированное программирование”

Выполнил студент

Группы БФИ1901

Бардюк Д.В.

Москва 2020

**Цель работы:** изучение подходов к командной разработке в системе контроля версий git.

В созданном в предыдущей работе либо вновь созданном репозитории:

1. Настроить игнорирование git’ом файлов определенного расширения с использованием glob-шаблонов

2. Вывести лог коммитов в измененном пользовательском формате

3. Переименовать ветку

4. Провести в ветке работу и слить с веткой master, удалить слитую ветку

5. Настроить удалённый репозиторий: получить изменения из репозитория и отправить изменения.

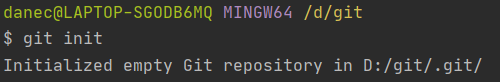
6. Организовать совместную работу над проектом в соответствии с выбранной идеологией

Возможна совместная работа студентов над одним проектом для изучения взаимодействия группы разработчиков и возможностей работы с СКВ.

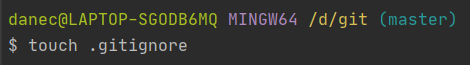
**Ход работы:**

Для того чтобы понять принципы командной работы в Git, мы создадим простую программу, основанную на взаимодействии двух классов. Один студент создает класс Food, а другой класс Human.

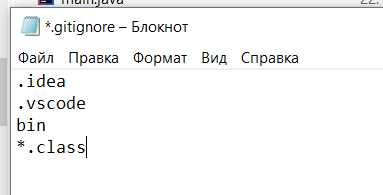
Добавим второго студента

  
Инициализируем Git командой “git init”.  


Создаем файл .gitignore, в который добавим те типы файлов, игнорируемые системой Git.



Добавляем в него папки .idea и .vscode, на случай если другой студент будет работать в данных редактора кода и его настройки не попадали в Git. Также добавляем папку bin и все файлы с расширением .class, чтобы скомпилированный код не попадал в репозиторий.



Создаем папку bin.



Создаем входную точку нашей программы.



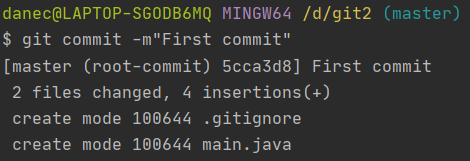
Подключаемся к удаленному репозиторию с помощью команды “git remote add origin”.



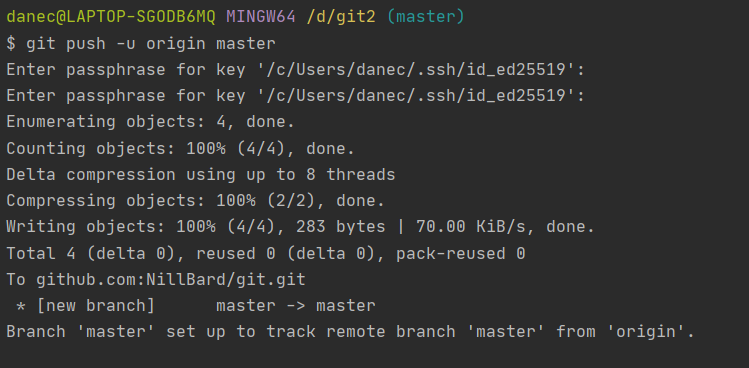
С помощью команды “git add .” добавляем этот файл в Git (аргумент точка обозначает, что добавить нужно все файлы, т.к. у нас он только один, данный вариант нам подходит).



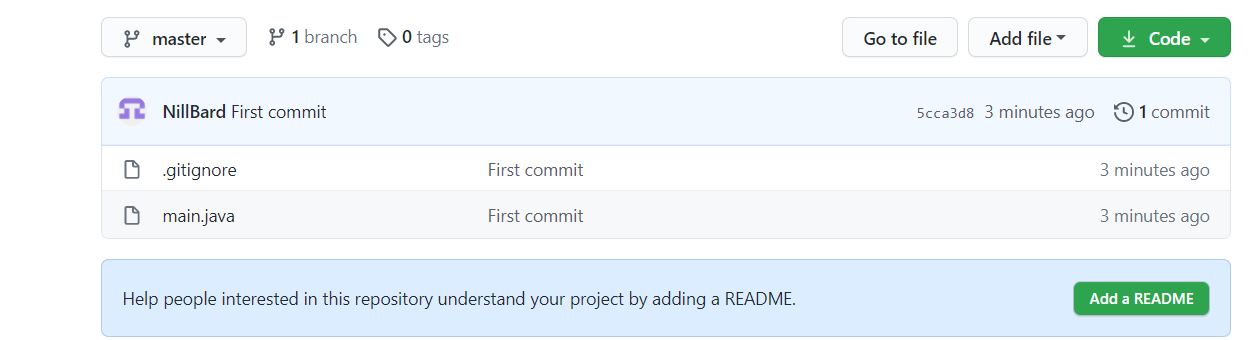
Делаем первый коммит с помощью команды “git commit -m” (флаг -m дает нам возможность сразу же написать пояснительное сообщение, если написать данную команду без флага, то сообщение нужно будет написать в открывшемся редакторе кода).



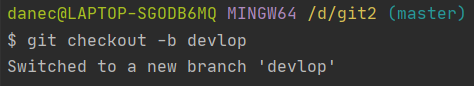
И отправляем изменения.



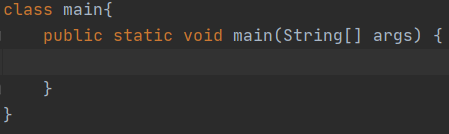
Как мы видим, в репозитории находятся только два файла. Все игнорируемые файлы в репозиторий не попадают.



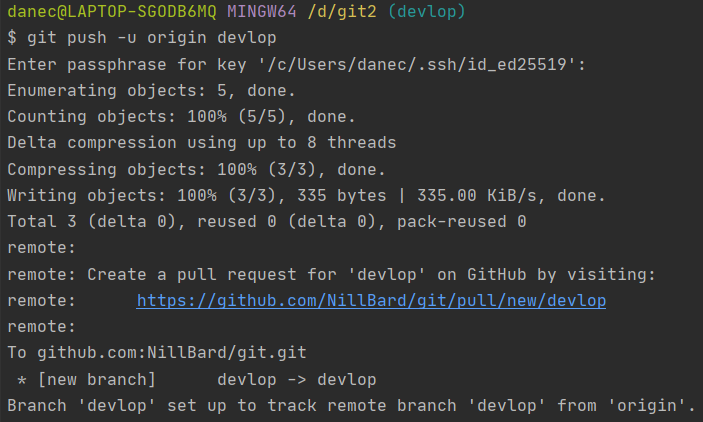
Переходим на ветку develop.



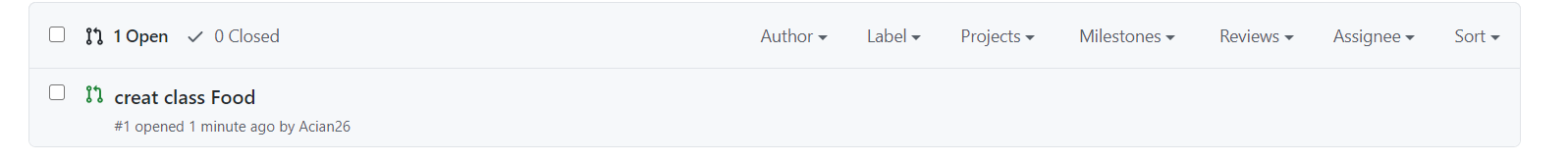
Создаем класс main.



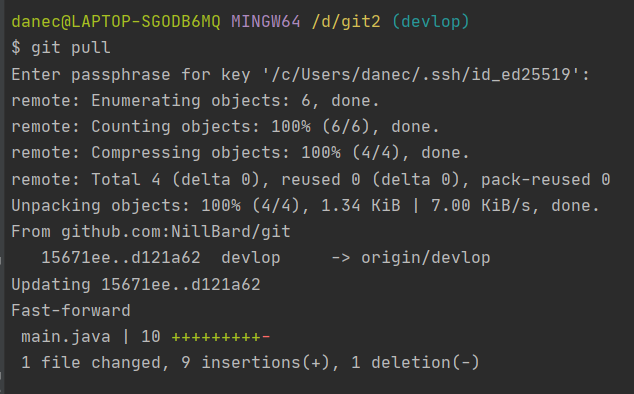
Отправляем изменения в репозиторий.



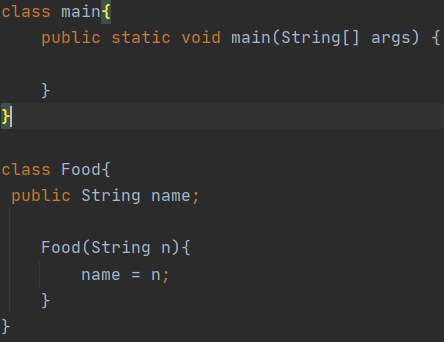
В истории изменений мы видим, что второй студент сделал один коммит, в котором создал класс Food.



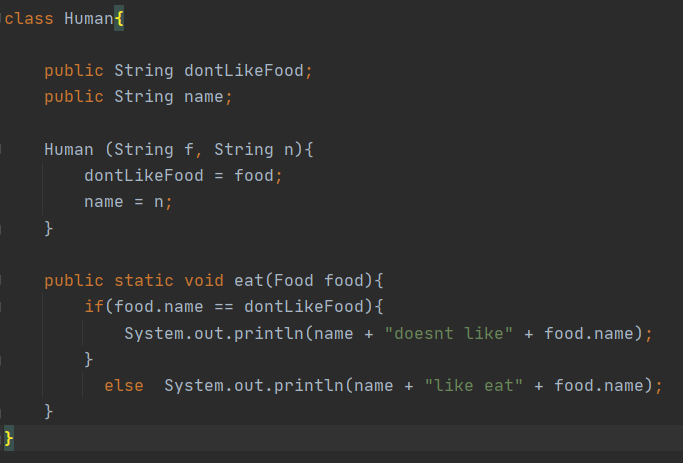
Используем комманду “git pull”, чтобы получить изменения.



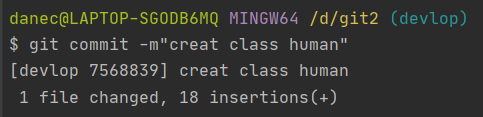
Класс Food добавился в файл Main.java.



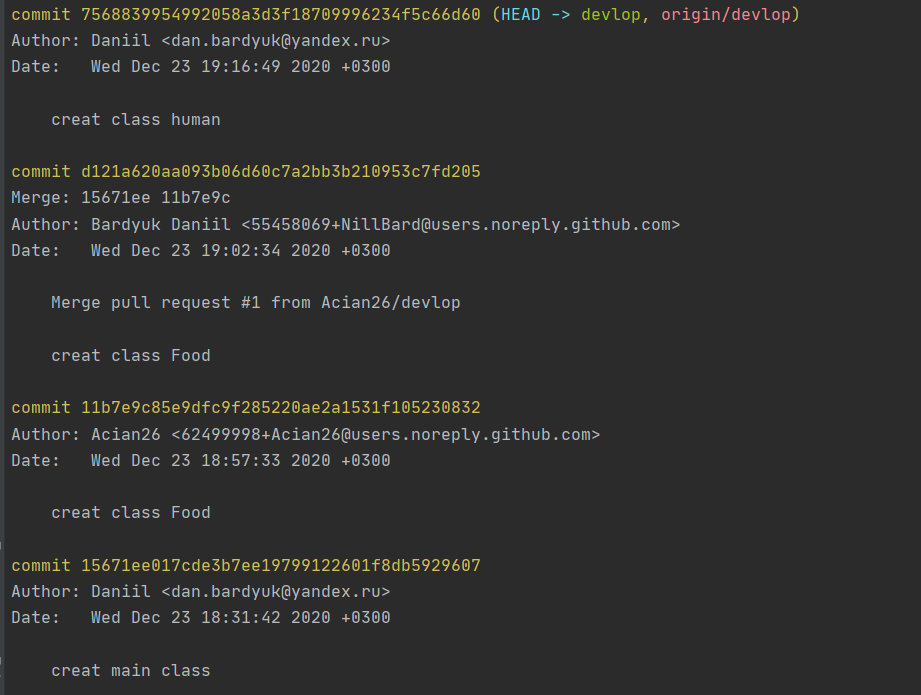
Создаем класс Human.



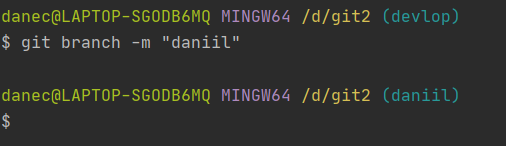
Делаем коммит.



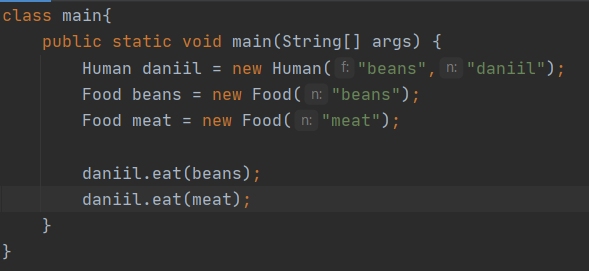
Вызвав команду “git log”, мы можем видеть коммиты обоих студентов.



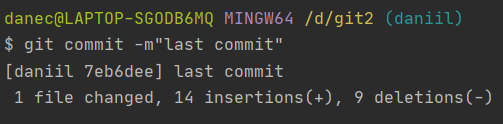
На случай если мы хотим переименовать ветку, существует команда “git branch” с флагом -m. Мы успешно переименовали ветку в локальном репозитории.



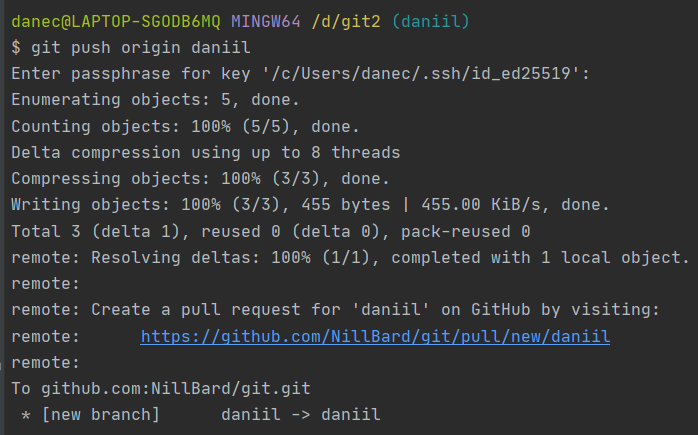
Реализуем взаимодействие экземпляров двух классов в методе main.



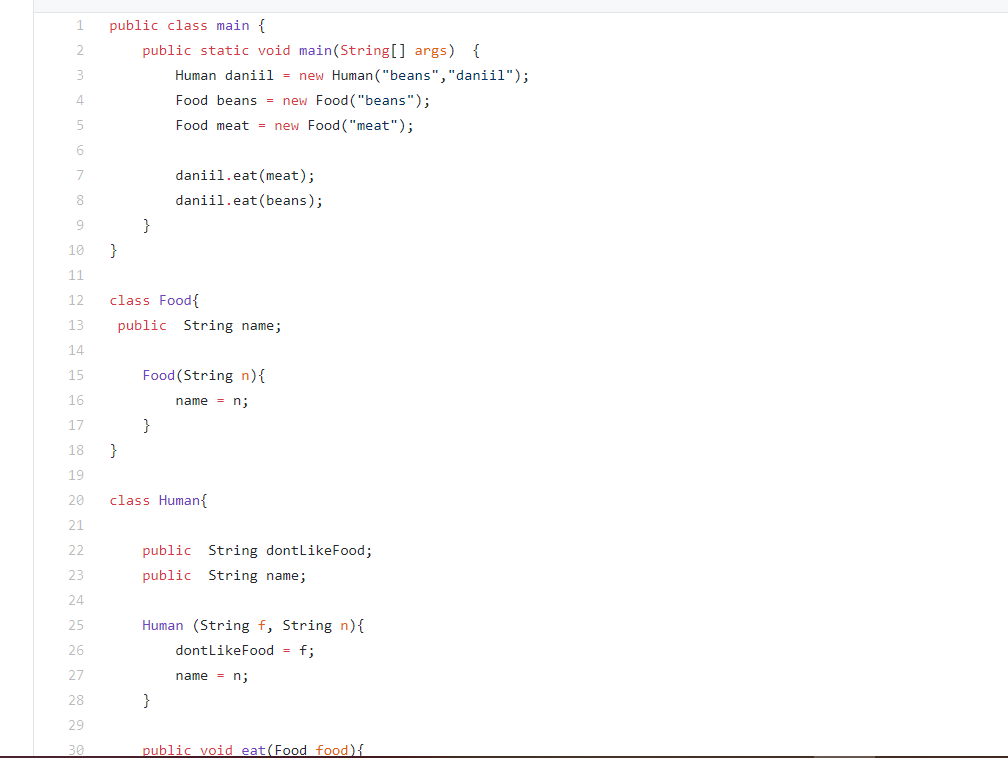
Делаем последний коммит.



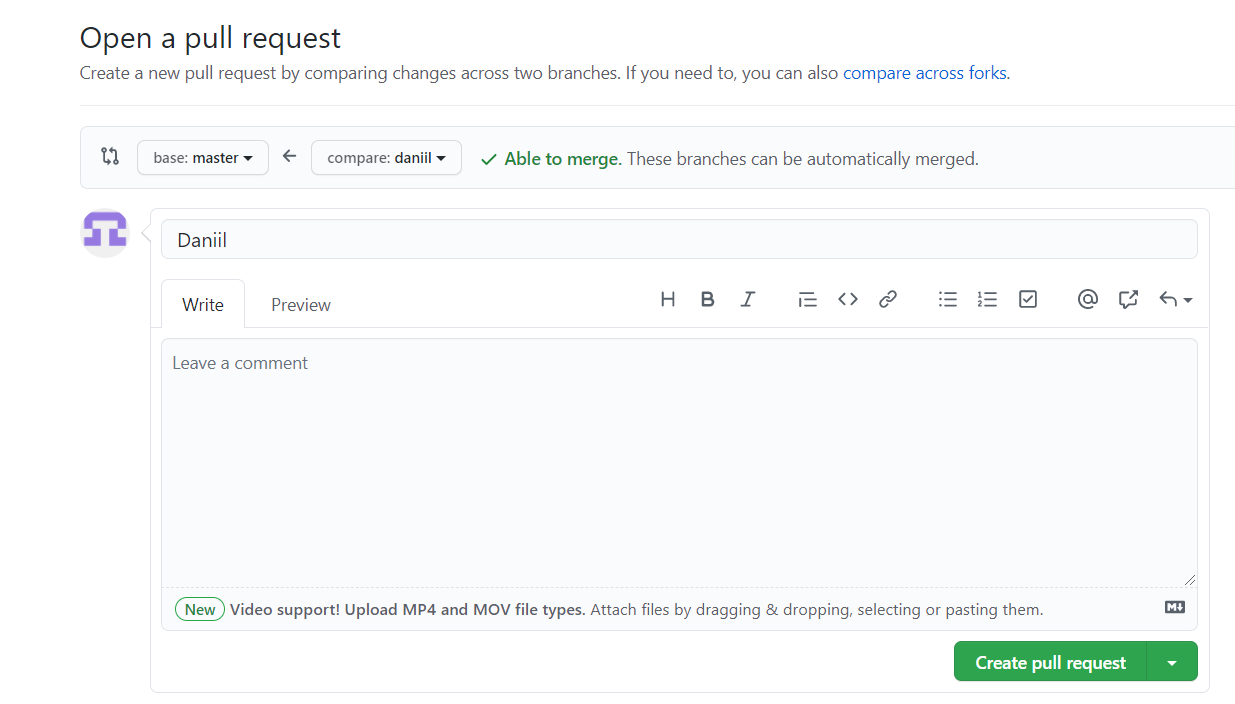
И отправляем изменения в удаленный репозиторий.

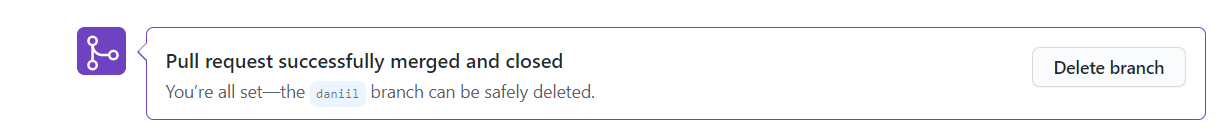


Весь код программы теперь есть в удаленном репозитории.

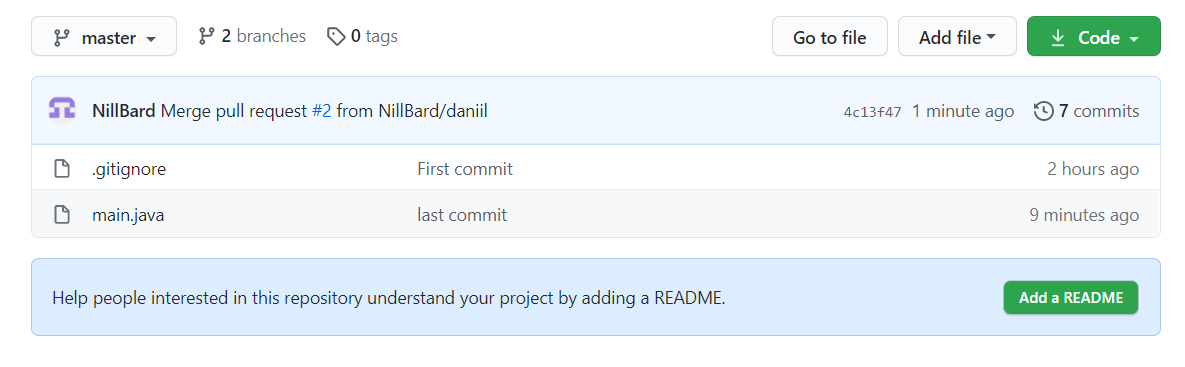


Создаем Merge Request на слияние веток и удаление ветки “daniil”.





Весь исходный код программы теперь доступен в master ветке репозитория.



**Вывод:** выполнив данную лабораторную работу, мы написали программу, на примере которой изучили подходы к командной разработке в системе контроля версий Git.

**Список использованных источников**

1) Документация Git: <https://git-scm.com/doc>

2) Документация GitLab: <https://docs.gitlab.com/>

3) Java. Полное руководство. Герберт Шилдт