

После изучения этого материала вы сможете:

- установить систему контроля версий Git на свой компьютер;
- подключить Git к среде разработки;
- сделать свой первый коммит и создать новую ветку;
- слить свои изменения из новой ветки в основную ветку master.

**Git** — система контроля версий, то есть хранилище истории разработки проекта. В нём можно:

- хранить код;
- отслеживать изменения в файлах;
- возвращаться к предыдущим версиям кода;
- организовывать совместную работу.

**Репозиторий (repository)** — место, где хранятся и поддерживаются какие-либо данные. В нашем случае это код приложения. Репозиторий может быть локальным и удалённым.

- **Локальный репозиторий** находится на компьютере разработчика.
- **Удалённый репозиторий** располагается на удалённом сервере. Он напоминает папку в облачном хранилище.

## Установка Git на компьютер

Установите дистрибутив Git на свой компьютер. На сайте [git-scm.com](https://git-scm.com) доступна инструкция по скачиванию и установке Git на компьютер для разных операционных систем.

Работа с системой контроля версий доступна в трёх вариантах:

- через командную строку;
- графический интерфейс (Git GUI);
- IntelliJ IDEA.

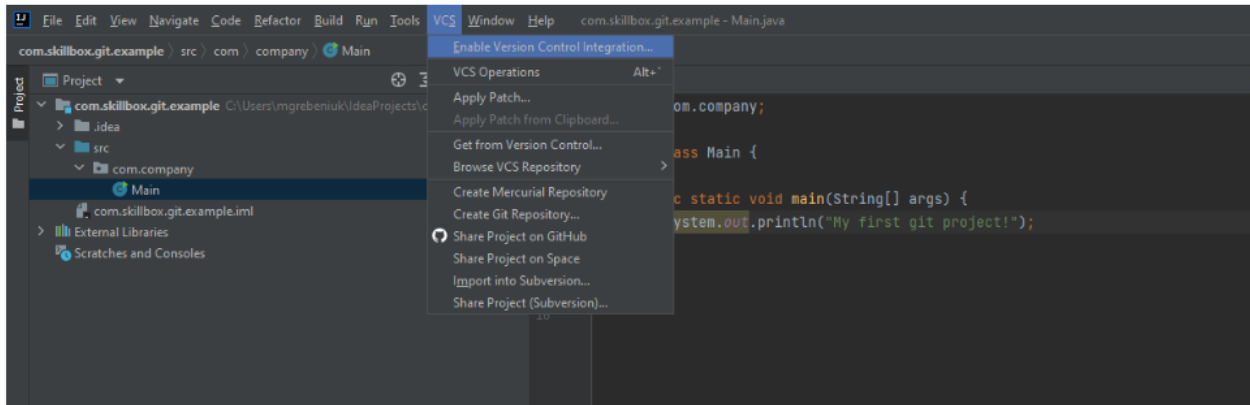
Первый вариант подходит опытным разработчикам, так как им требуется более расширенный инструментарий для работы с Git. При обучении и для начинающих разработчиков приоритетным будет вариант работы через IntelliJ IDEA, так как эта среда разработки предоставляет более удобный, понятный и достаточно функциональный интерфейс.

<https://git-scm.com/book/ru/v2/%D0%92%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5-%D0%A3%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BA%D0%B0-Git>

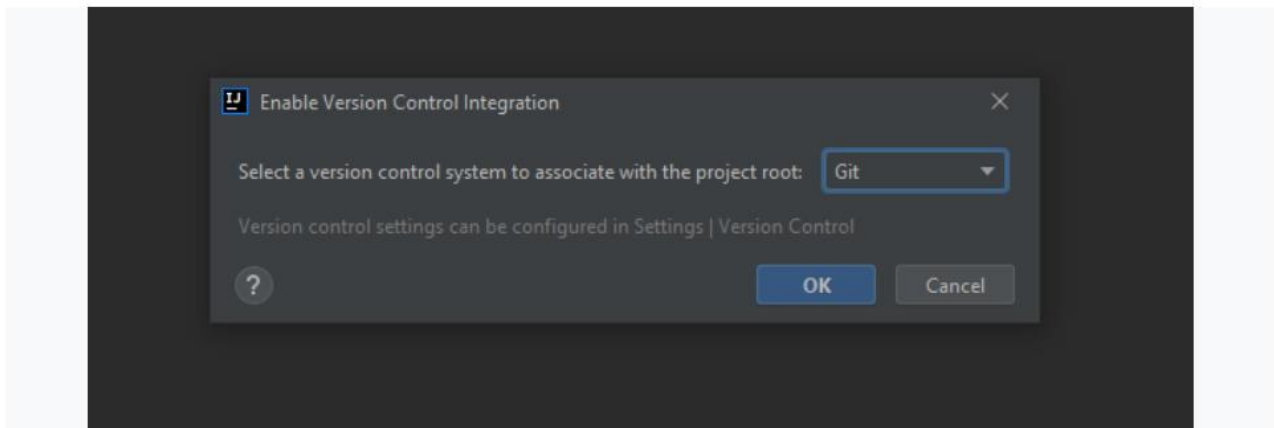
## Подключение Git к IntelliJ IDEA

Попрактикуемся работать с Git из среды разработки.

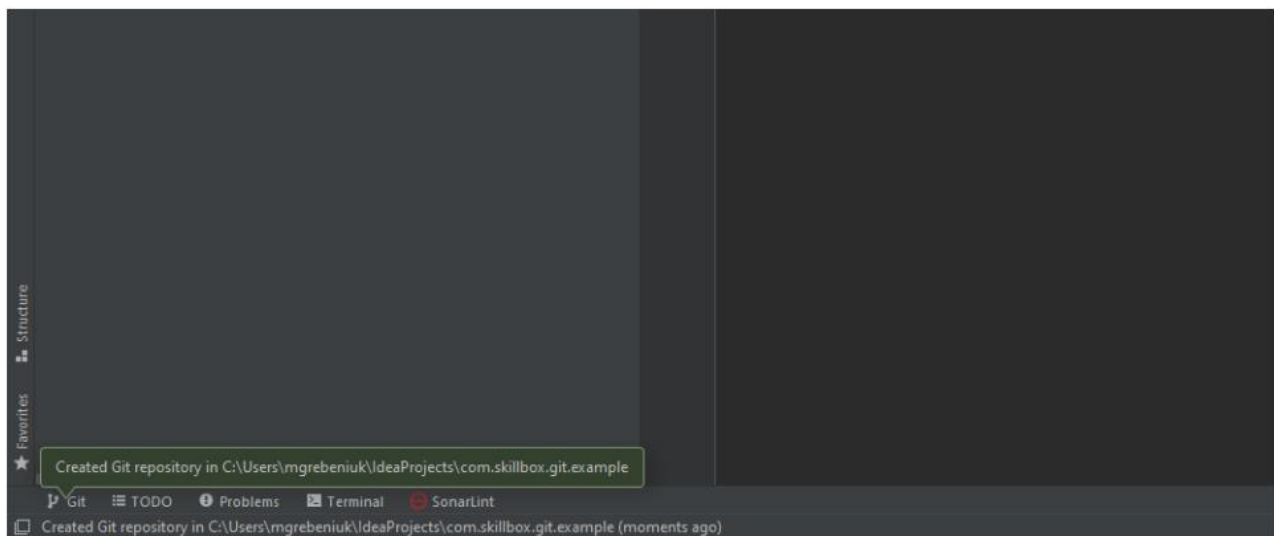
1. Откройте среду разработки IntelliJ IDEA.
2. Создайте новый проект.
3. Включите в среде разработки поддержку Git. Для этого выберите VCS → Enable Version Control Integration:



4. В появившемся окне выберите систему контроля версий Git:

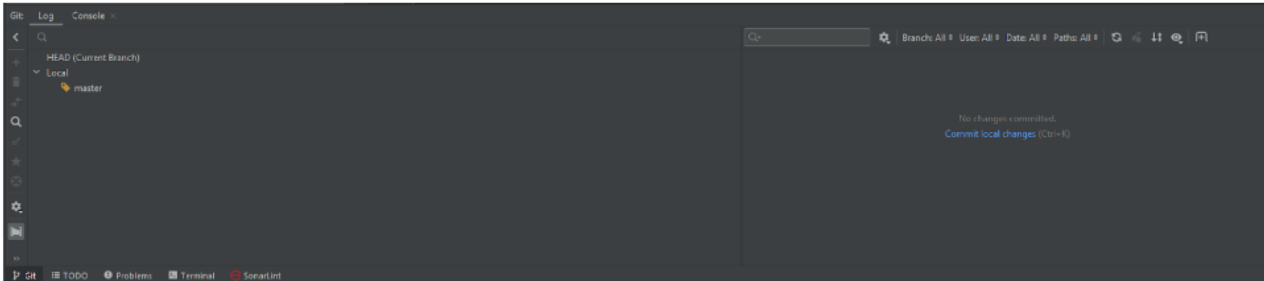


5. Убедитесь, что в нижнем трее (области состояния) появилась вкладка Git:

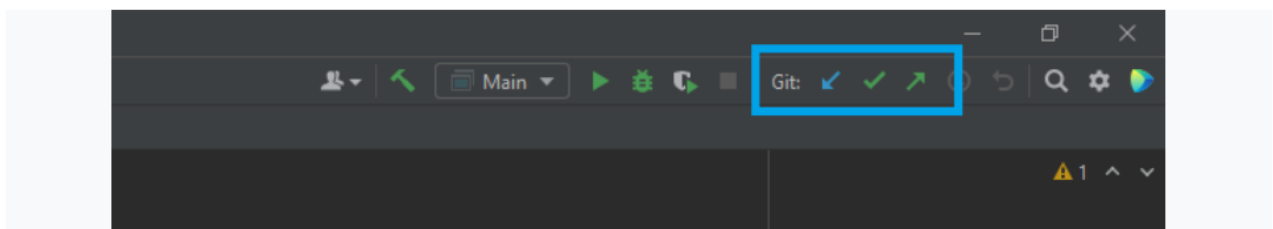


6. Раскройте вкладку Git и посмотрите, что находится в окне:

- В папке Local (так как текущий проект — локальный) мы видим расположение ветки master. Это основная ветка, которая есть в каждом проекте, сейчас мы находимся в ней.
- Справа — окно, в котором будет показана история ваших изменений, то есть история коммитов (как всё это будет выглядеть, вы увидите чуть позже).



- В правом верхнем углу появились три новые кнопки: **Git Pull, Git Commit, Git Push:**

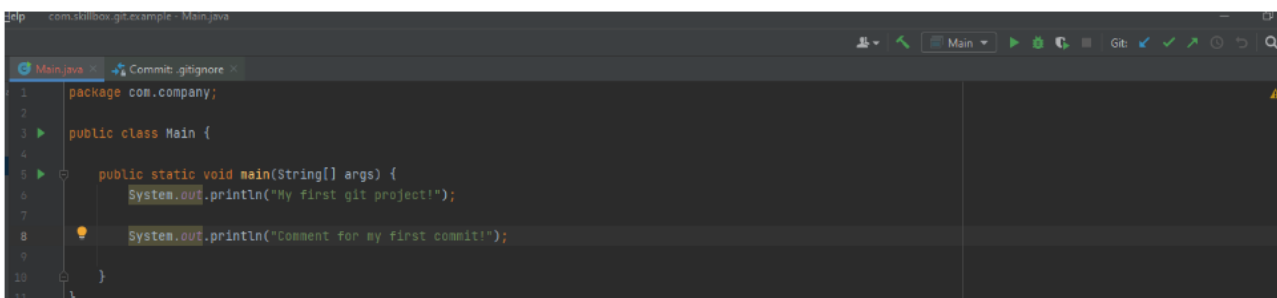


С командами Git Pull и Git Push мы поработаем в следующем модуле, а сейчас попрактикуемся с командой Git Commit.

## Первый коммит

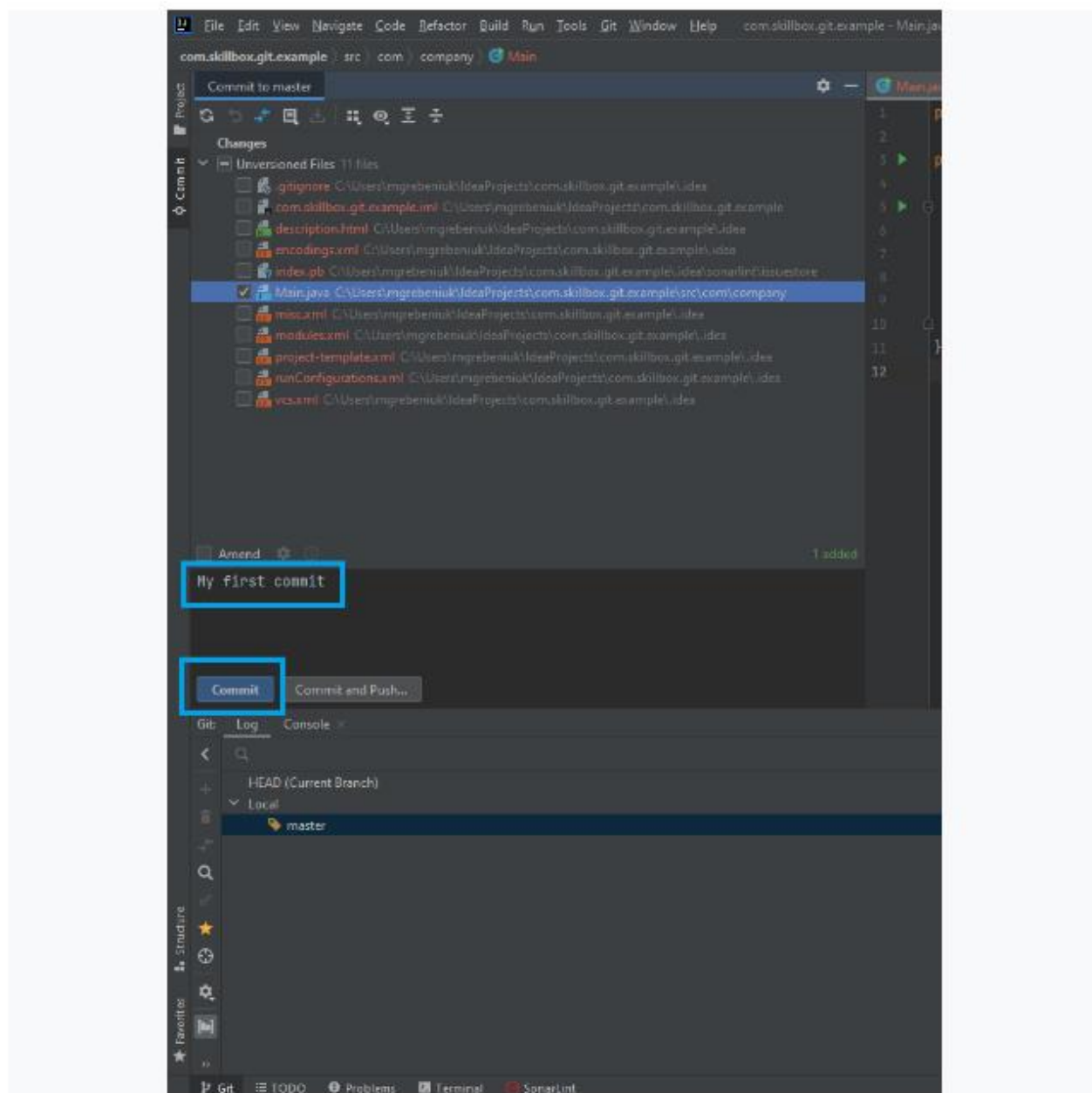
Внесём изменения в проект.

1. Добавьте новый вывод в консоль с комментарием Comment for my first commit и нажмите кнопку Commit в правом верхнем меню:



2. В открывшемся окне вы увидите файлы, которые нужно закоммитить, то есть сохранить их состояние на данный момент для текущей ветки (ветки master):

- Выберите класс Main, так как мы добавили в него изменения.
- Добавьте описание коммита: My first commit.
- Нажмите на кнопку Commit ниже.



3. Убедитесь, что первый коммит был сохранён, — в нижнем трее справа от ветки master должен появиться коммит и следующая информация:

- описание коммита;
- автор коммита;
- дата создания коммита.

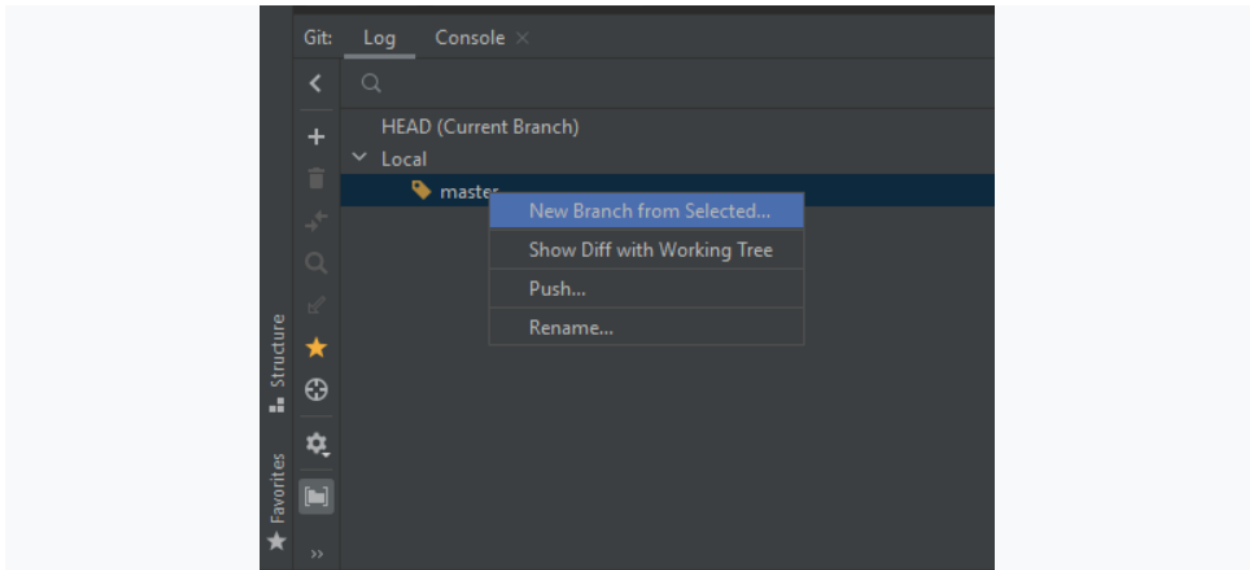


Здорово! Теперь вы знаете, что ваши изменения сохранены в локальном хранилище и с ними ничего не произойдёт.

## Создание новой ветки

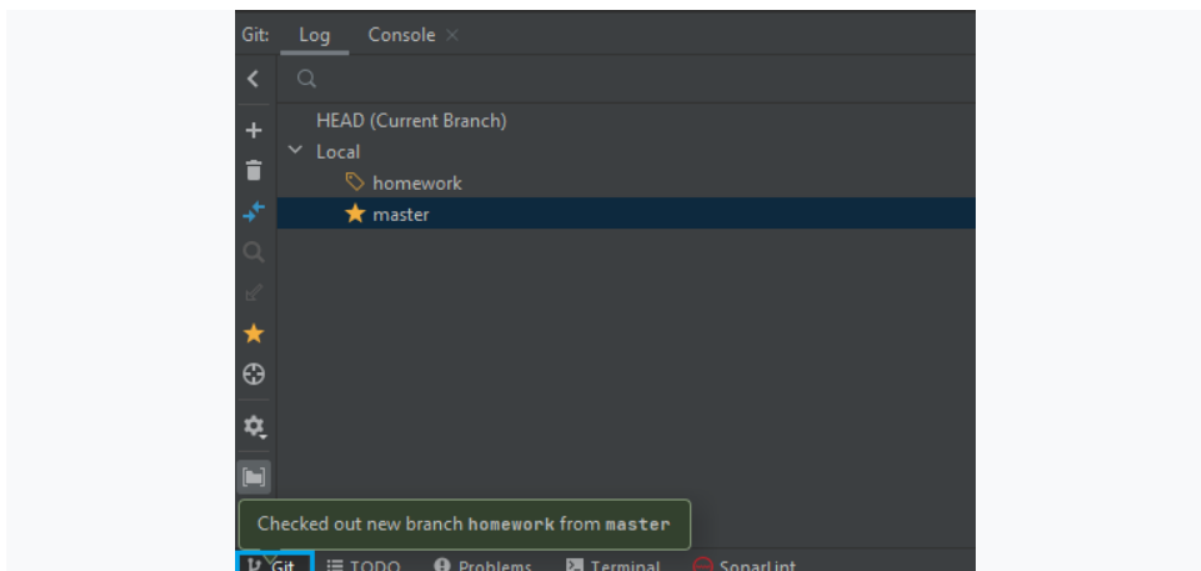
Создадим новую ветку для доработки. Представим, что мы хотим добавить в программу вычислительную операцию, но делать это в основной ветке master мы не хотим, чтобы сохранить исходную версию программы пока неизменной.

1. Для создания новой ветки кликните правой кнопкой мыши по master и выберите New Branch from Selected:



2. В появившемся окне укажите название новой ветки, например homework, и нажмите Create.

3. В нижнем меню появилась новая ветка homework. Значок бирки означает, что прямо сейчас мы находимся во вновь созданной ветке homework:



## Команда Checkout

Добавим в класс Main вычислительную операцию, именно для этого мы и создавали новую ветку:

```
1 package com.company;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         System.out.println("My first git project!");
7
8         System.out.println("Comment for my first commit!");
9
10        System.out.println(4 + 5);
11    }
12 }
13
14 }
```

Запускаем и проверяем код — всё работает отлично, в итоге возвращается нужное число 9. Наша задача выполнена, давайте сохраним результаты.

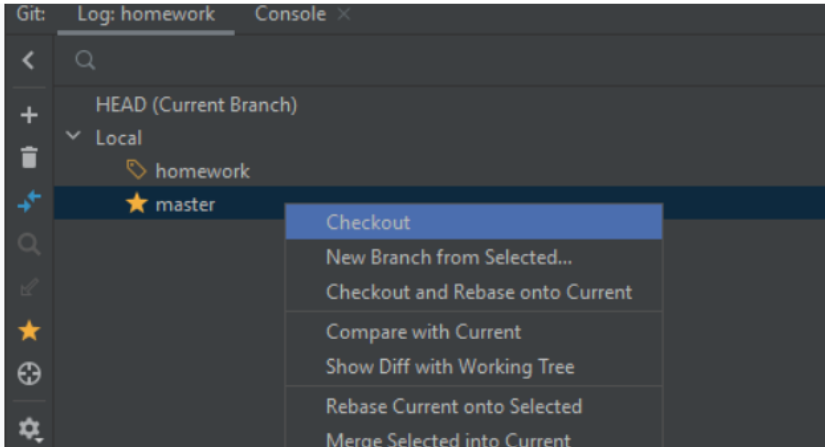
Действуйте по уже известной схеме:

1. Нажмите кнопку Git Commit в правом верхнем углу → выберите класс Main, добавьте описание коммита (например, My second commit) → нажмите Commit над описанием вашего коммита.
2. Проверьте, что второй коммит был создан и он строго последователен первому коммиту:



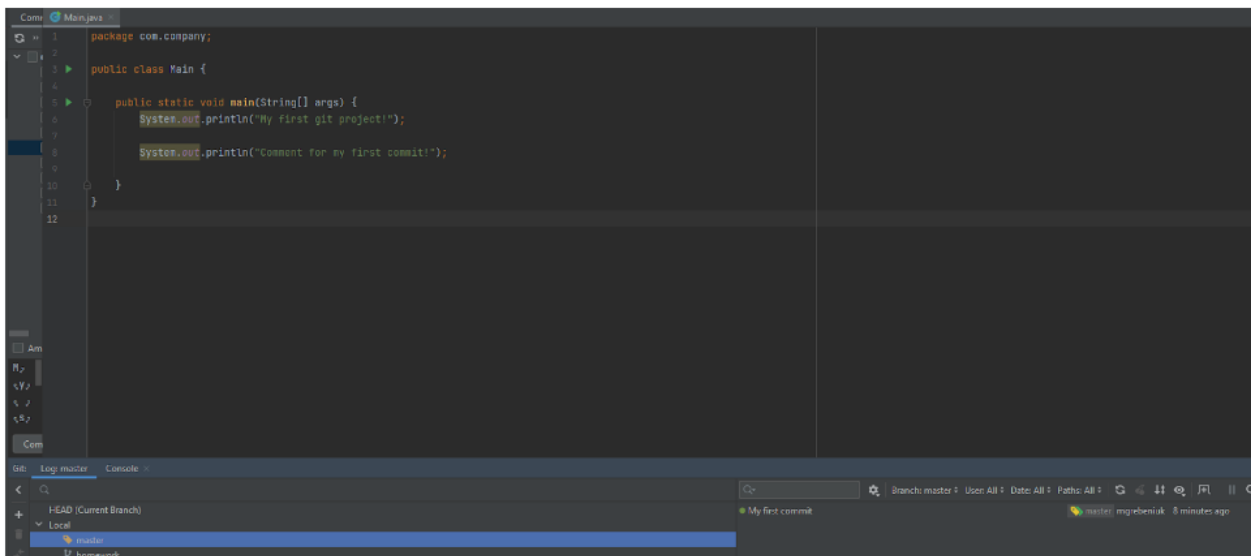
```
graph TD
    A[My second commit] --> B[My first commit]
```

Посмотрим, что происходит в ветке master. Для этого необходимо выполнить команду **Checkout**, которая позволяет перемещаться между ветками. Для этого кликните правой кнопкой мыши по ветке master → Checkout:



```
graph TD
    subgraph Git_GUI [Git: Log: homework Console x]
        subgraph Local_Branches [Local]
            homework[homework]
            master[master]
        end
        master --> Checkout[Checkout]
        master --> New_Branch[New Branch from Selected...]
        master --> Checkout_Rebase[Checkout and Rebase onto Current]
        master --> Compare[Compare with Current]
        master --> Show_Diff[Show Diff with Working Tree]
        master --> Rebase[Rebase Current onto Selected]
        master --> Merge[Merge Selected into Current]
    end
```

Перейдя в ветку master, видим, что последних доработок в ней нет. Это логично, ведь мы работали в другой ветке и ветка master осталась без изменений:

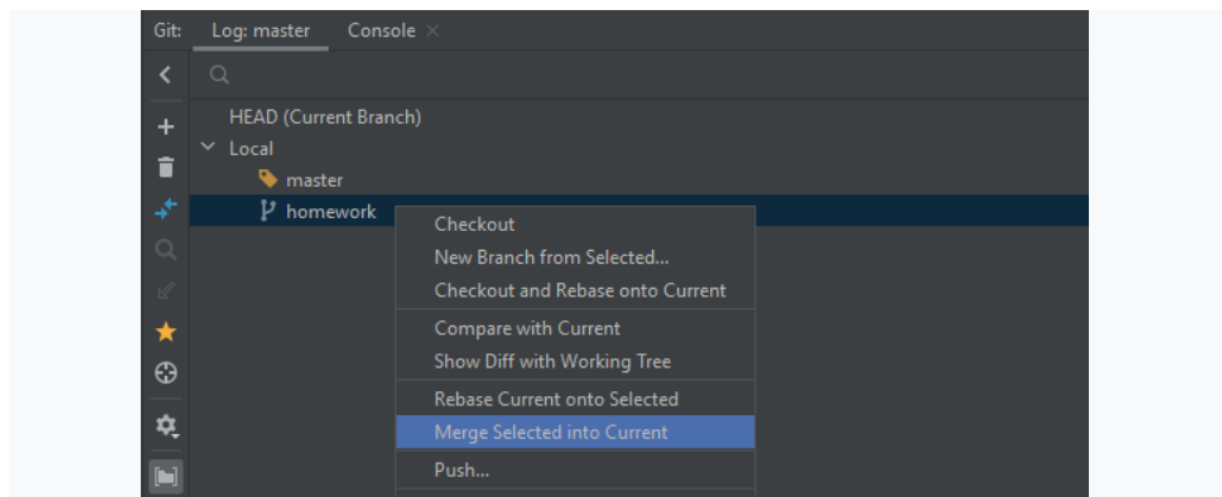


## Команда Merge

Мы внесли необходимые изменения в ветке homework, протестировали код и убедились, что всё работает корректно. Далее нашу доработку с вычислениями необходимо залить в основную ветку проекта — master.

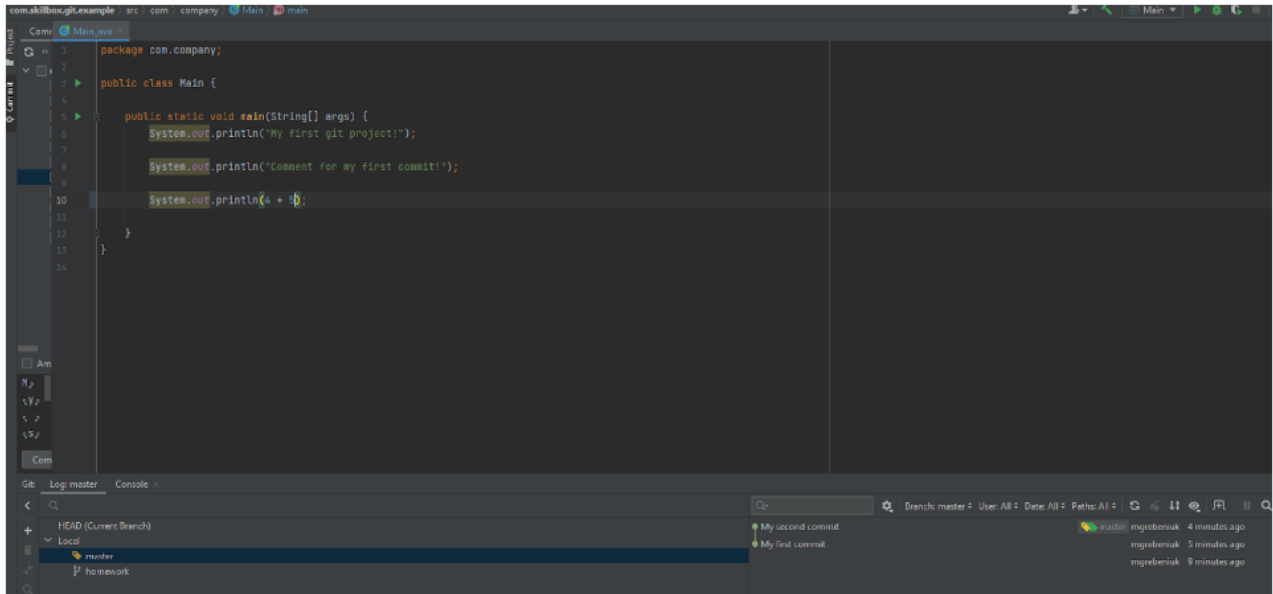
Для этого используем команду **Git Merge**, которая позволяет слить изменения из одной ветки в другую, в нашем случае homework → master.

Правой кнопкой мыши кликните по homework и выберите Merge Selected into Current:





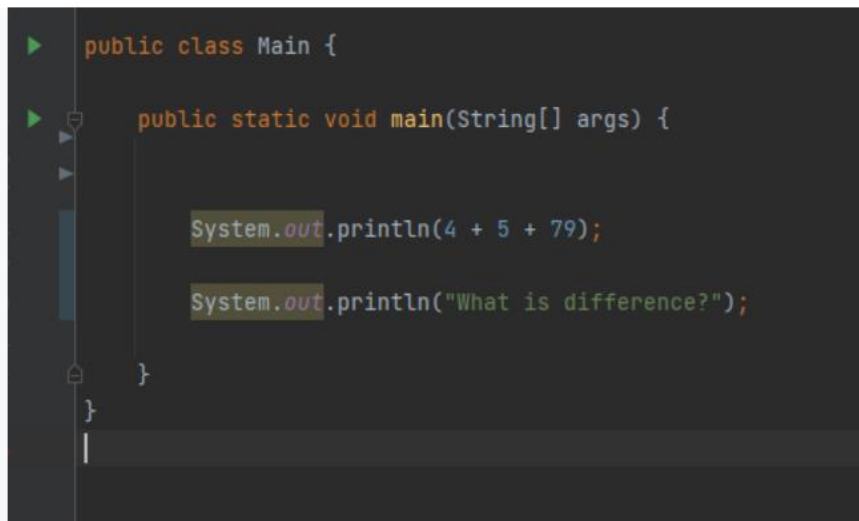
Теперь наши изменения из ветки homework попали в master. Чтобы убедиться в этом, открываем класс Main и видим код, который находился в ветке homework. В нижнем трее справа теперь также появилась история коммитов, включая второй коммит, который мы делали в ветке homework:



## Различия между версиями

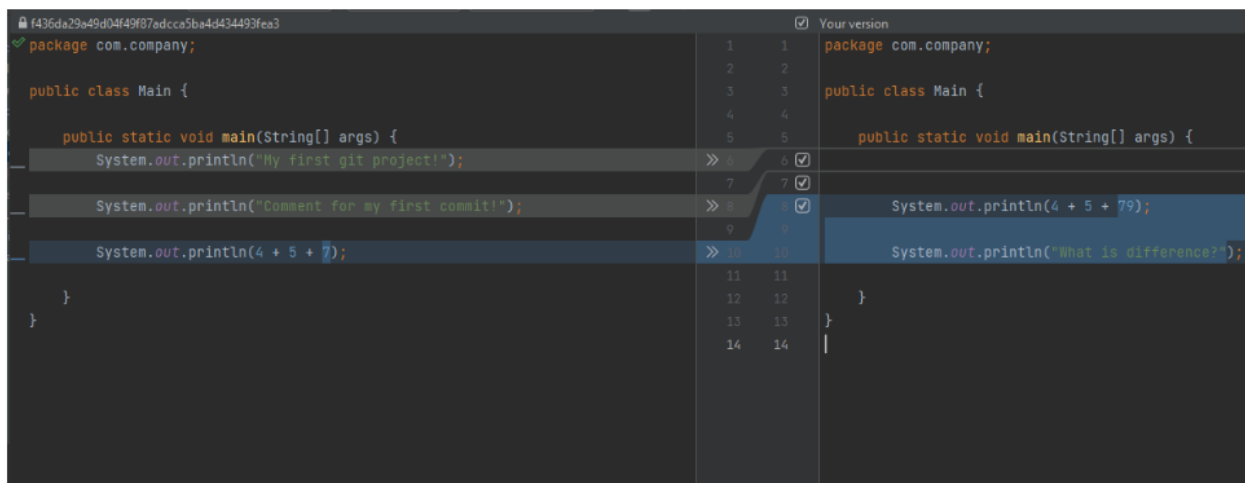
Как можно увидеть изменения, которые мы внесли в класс:

1. Удалите строку с первым коммитом `My first git project` и `Comment for my first commit`.
2. Добавьте к вычислениям ещё одно число и вставьте новую строку `«What is difference?»`:





3. Чтобы посмотреть изменения, нажмите в блоке с коммитом на класс Main. Серым цветом выделяются удалённые строки, зелёным — добавленные, синим — изменения в строках:



```
package com.company;

public class Main {
    public static void main(String[] args) {
        System.out.println("My first git project!");
        System.out.println("Comment for my first commit!");
        System.out.println(4 + 5 + 7);
    }
}
```

1 1 package com.company;

2 2

3 3 public class Main {

4 4

5 5 public static void main(String[] args) {

6 6

7 7

8 8

9 9

10 10

11 11

12 12

13 13

14 14

Так вы перед коммитом можете видеть то, что изменилось в ваших файлах.