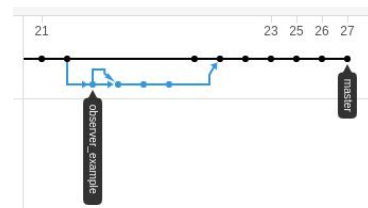


# Git и совместная работа

Марк Заславский, [mark.zaslavskiy@gmail.com](mailto:mark.zaslavskiy@gmail.com)

# Вспоминаем про Git

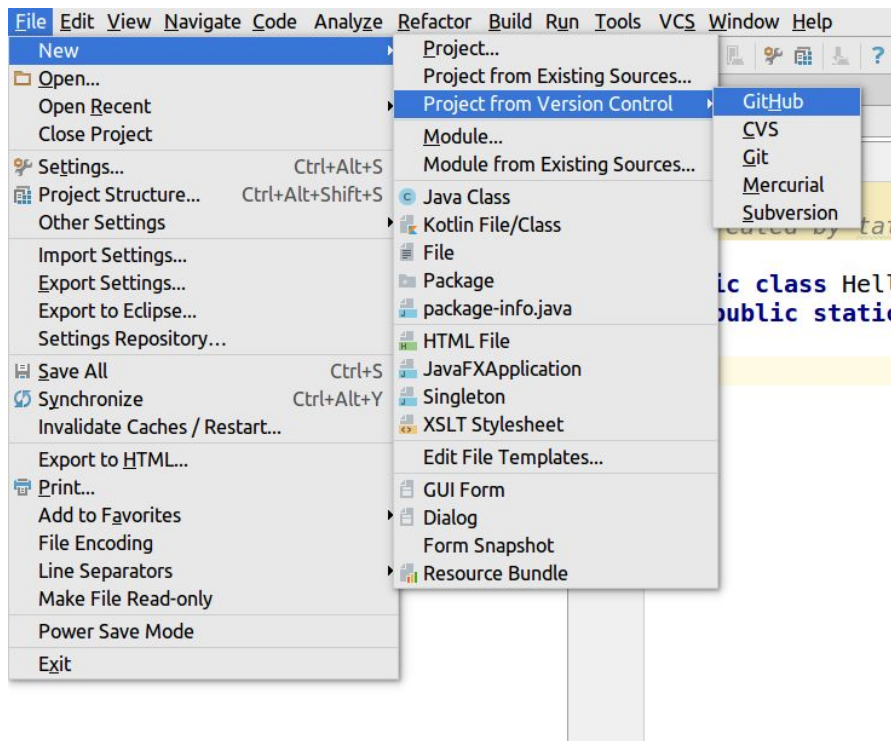
- Кто делал коммиты?
- Система контроля версий
  - Возможность откатиться
  - Возможность удобно интегрировать результаты работы нескольких человек
- Основные термины
  - Репозиторий (локальный/удаленный)
  - Ветка
  - Коммит
  - Теги
- Github.org = хостинг Git репозиториев



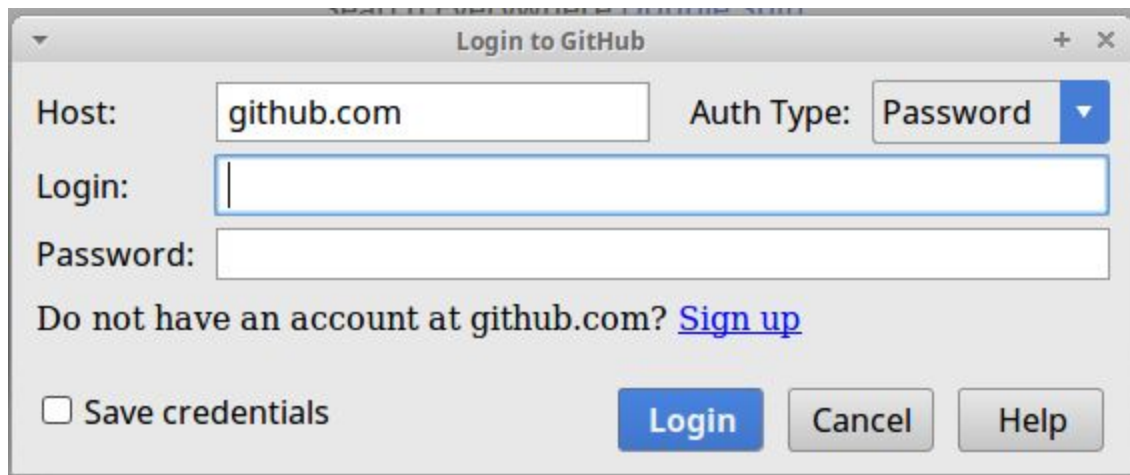
# Git - смотрим вглубь

- **Рабочая директория** - каталог, в котором лежит репозиторий. Файлы в РД могут появляться и исчезать при изменении текущей отметки.
- **Индекс** - изменения, которые войдут в коммит.
- **Текущая отметка** - версия исходного кода, с которой идет работа в данный момент (номер коммита).
- **Отслеживаемые и не отслеживаемые файлы**

# Открываем проект из репозитория - 0



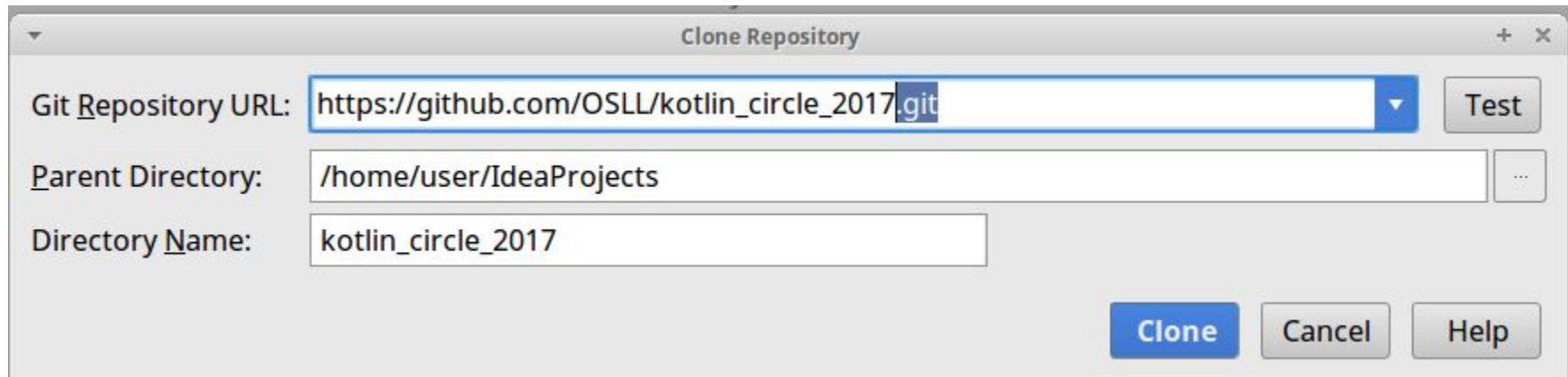
# Открываем проект из репозитория - 1



A screenshot of a 'Login to GitHub' dialog box. The dialog has a title bar with a minus sign, a plus sign, and a close button. It contains the following fields and controls:

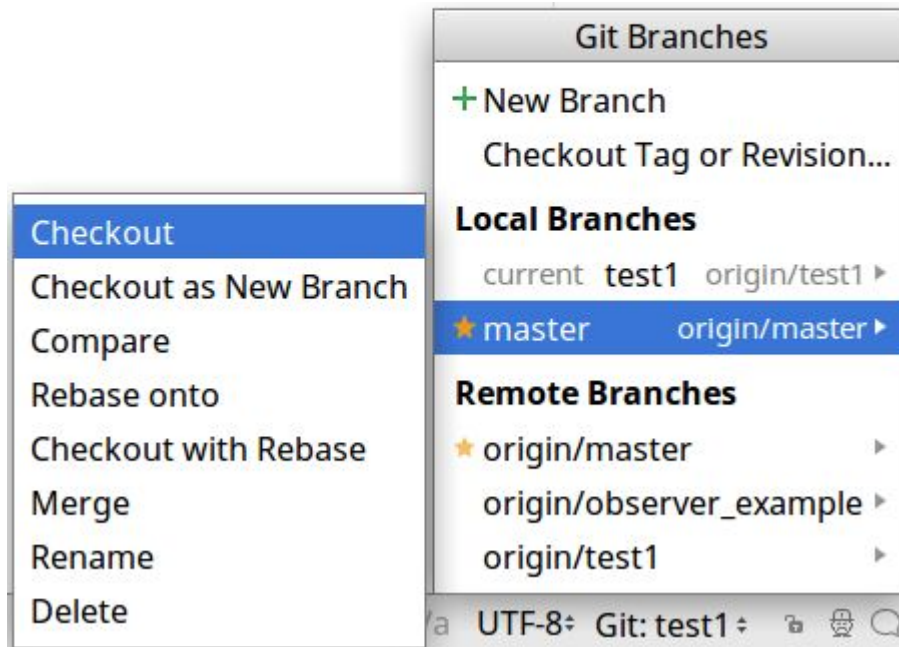
- Host:** A text field containing 'github.com'.
- Auth Type:** A dropdown menu showing 'Password' with a blue arrow icon.
- Login:** An empty text field with a blue border.
- Password:** An empty password field.
- Text:** 'Do not have an account at github.com? [Sign up](#)'.
- Checkbox:** 'Save credentials' with an unchecked checkbox.
- Buttons:** 'Login' (blue), 'Cancel' (gray), and 'Help' (gray).

# Открываем проект из репозитория - 2



# Самая важная кнопка (1) - ветки

- **New branch** - создать новую ветку.
- **Checkout ...** - переключение на определенную отметку.
- **Local Branches** - ветки, с которыми ведется локальная работа
- **Remote Branches** - ветки из удаленного репозитория.

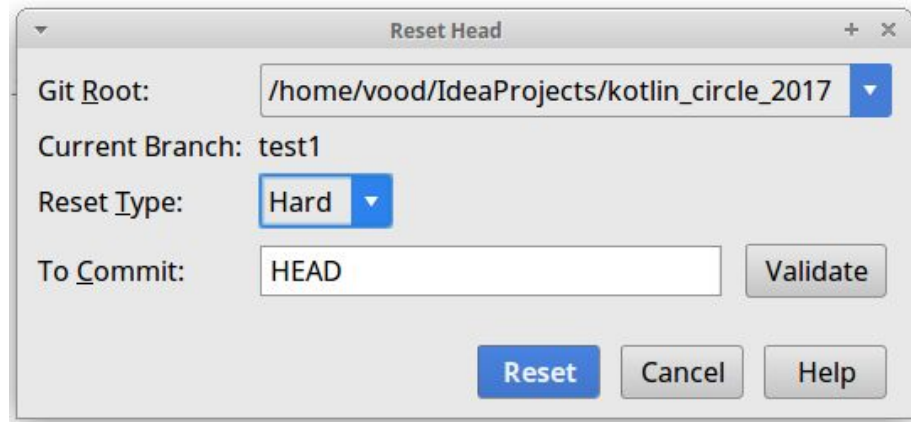
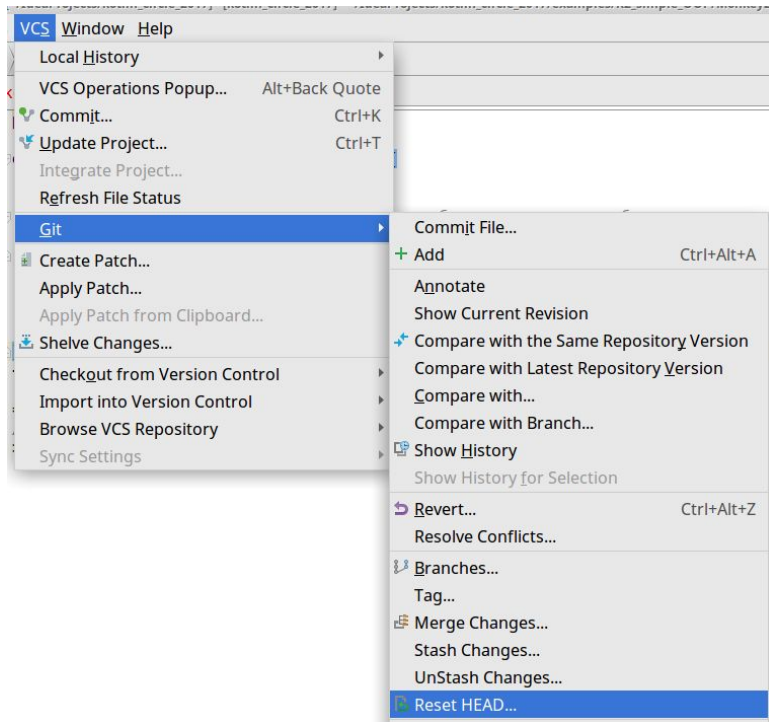


# Самая частая последовательность команд

- <https://github.com/.../invitations> // Добавление в репозиторий (если потеряли письмо)
- Checkout master // Переключаемся на мастер
- Pull // Подкачиваем новые изменения
- New branch // Создаем новую ветку под задачу
- Commit & push // Коммитим и пушим нашу ветку
- Commit & push
- Commit & push
- Create pull-request // По готовности задачи создаем Pull-Request

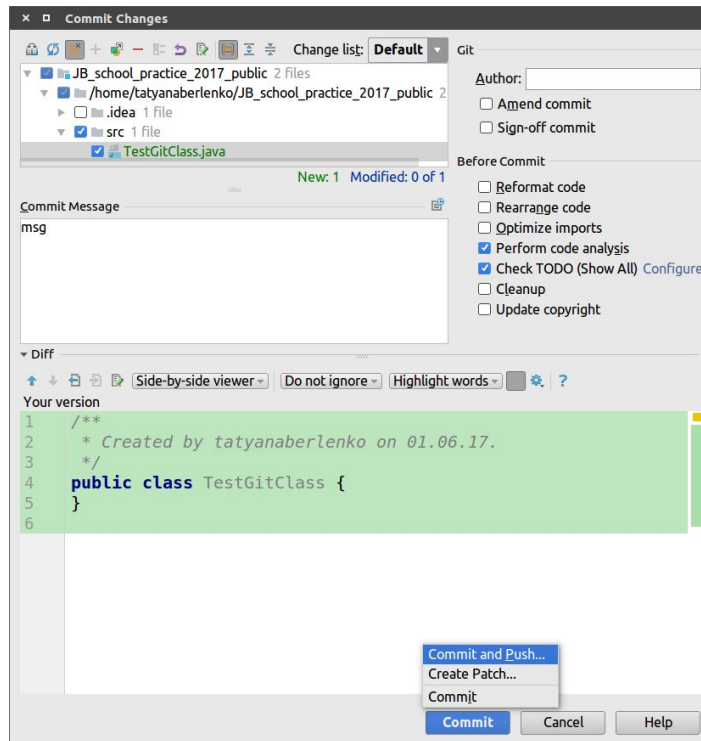


# Если хотим убрать незакомиченные изменения (навсегда)



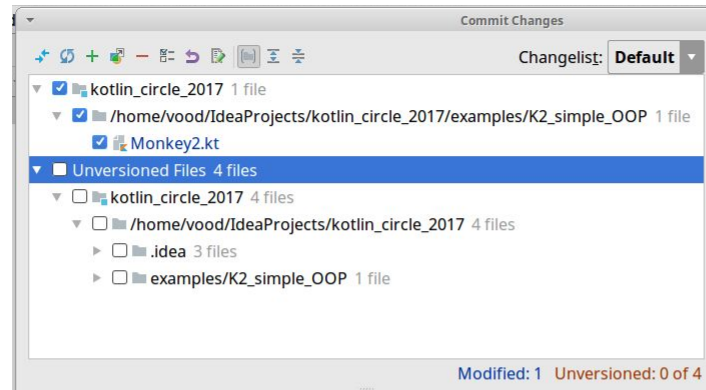
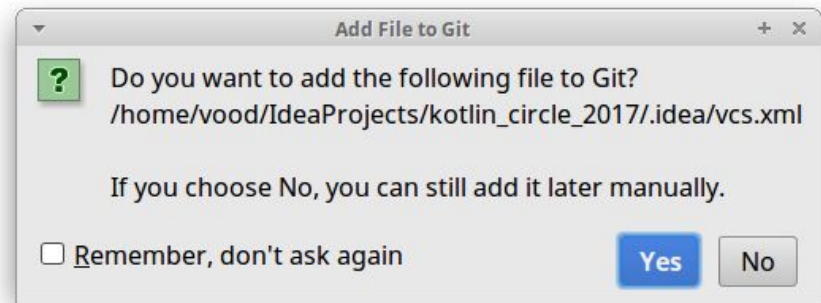
# Как сделать коммит

- Можно коммитить один или несколько файлов / всю директорию - это влияет на объем изменений.
- Старайтесь писать понятные Commit Message - потом будет сложно понять, зачем вы делали этот коммит.
- Чем **меньше и чаще** коммиты - тем лучше!



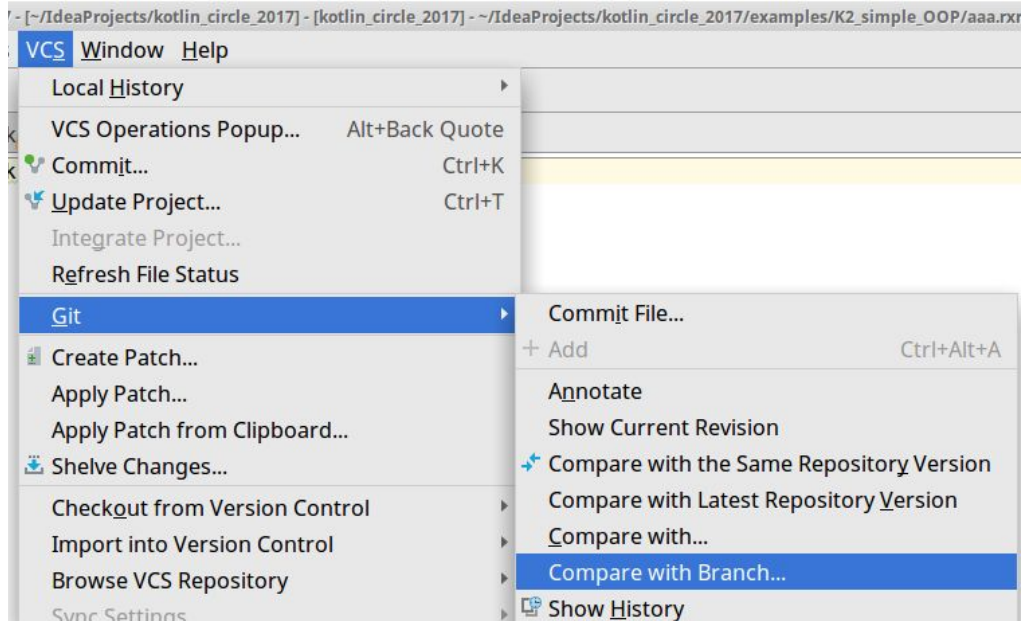
# Добавление файлов в под контроль Git

- При создании новых файлов вы будете видеть сообщение (1).
- Не добавленные файлы не войдут в коммит (**unversioned**) - сообщение (2).



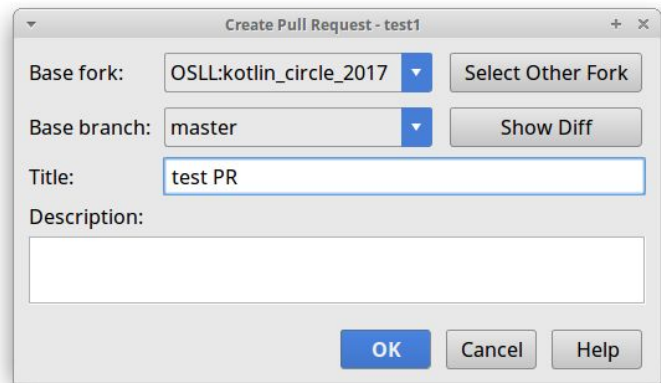
# Как сравнить две ветки

- Зачем?
- Зачем нужен и что делает **Compare with ...?**



# Как сделать пулл-реквест

- **Пулл-реквест** - заявка на мерж вашей ветки в другую (обычно master).
- **Base branch** - в какую ветку вы хотите залить свою.
- **Title** - название



Create Pull Request - test1

Base fork: OSLL:kotlin\_circle\_2017 Select Other Fork

Base branch: master Show Diff

Title: test PR

Description:

OK Cancel Help

Successfully created pull request  
[Pull request #2](#)

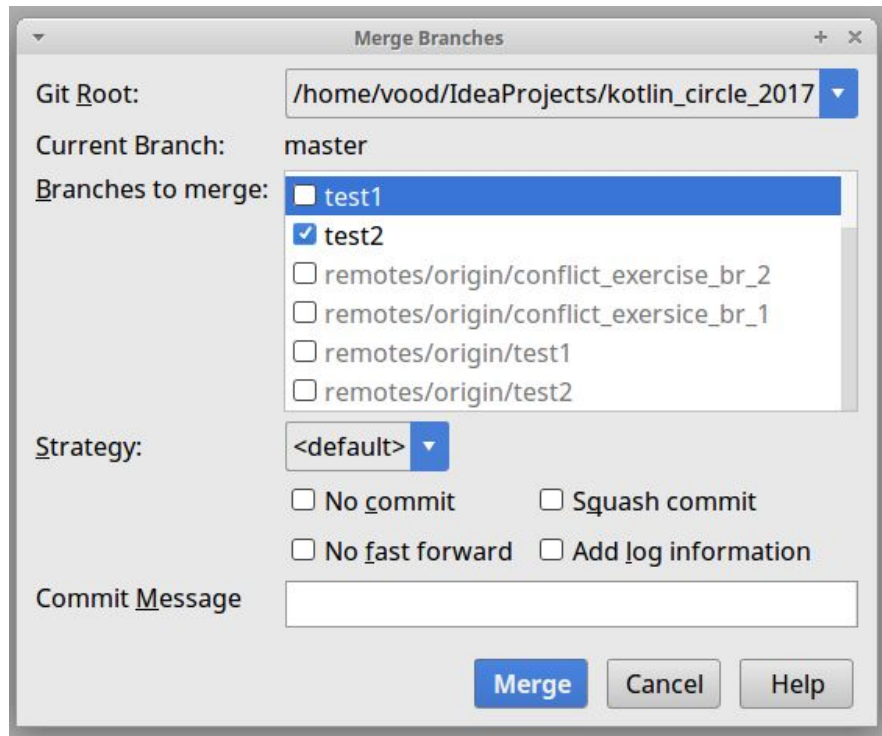
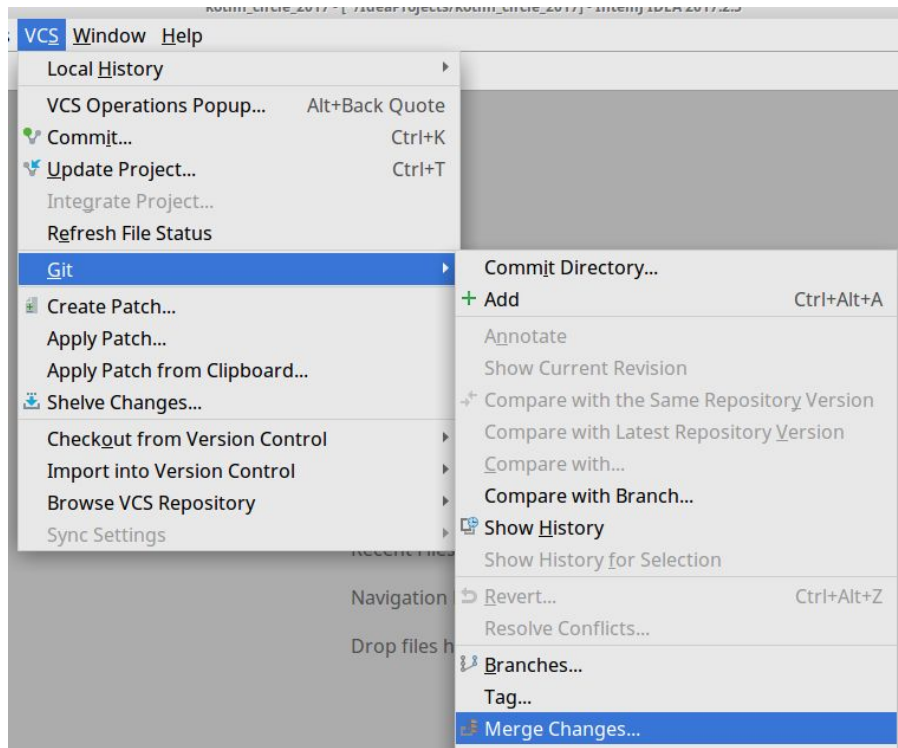
# Задача #1

1. Клонировать локально свой репозиторий в Idea.
2. Создать в репозитории своего проекта новую ветку **<Фамилия>\_1** с коммитами и **Pull Request** для нее в **master**.

# Как мержить ветки и фиксировать конфликты - Idea

- **Мерж (merge) или слияние** - объединение коммитов двух веток в одной из них.
- Слияние не всегда сопровождается конфликтами.
- **Конфликт** - ситуация при мерже, когда git не может понять, как именно сделать мерж.
- Конфликты возникают, когда изменения равноправны, например, не произошла синхронизация двух веток через мастер.
- Конфликт исправляет автор вливаемой ветки.

# Как мержить ветки и фиксить конфликты - 1





# Как выглядит конфликт (test1 и test2)

Run Tools VCS Window Help

Monkey2.kt

Monkey2.kt x

```
1 package K2_simple_OOP
2
3 open class Monkey2 : MammalInterface {
4     private val name: String? = null
5
6     override fun sleep() { // Этот метод обязательно
7         println("zzzzzzzz")
8     }
9
10    // все поля интерфейса доступны наследникам
11    val isCheeksPresent: Boolean
12    get() = isCheeksPresent
13 }
14 <<<<<< HEAD
15
16 =====
17 ///Good day, sir. I am mr. Conflict
18 >>>>>> test2
19
```

Files Merged with Conflicts

Name ^	Type	Yours	Theirs
Monkey2.kt (/home/vood/	Text	Modified	Modified

Accept Yours

Accept Theirs

Merge...

# Как фиксировать конфликт

- Поправьте исходник,
- Удалите оставшиеся указатели на места конфликта
  - >>>>>
  - =====
  - <<<<<
- Убедитесь, что работоспособна как функциональность вашей ветки, так и функциональность другой ветки.
- Сделайте коммит.

## Задача #2

Клонировать в идее репозиторий с примерами

[https://github.com/OSLL/kotlin\\_circle\\_2017](https://github.com/OSLL/kotlin_circle_2017)

1. Смержить в мастер ветку **conflict\_exersice\_br\_1**
2. Смержить ветку **conflict\_exersice\_br\_1** в **conflict\_exersice\_br\_2**, разрешить возникшие конфликты так, чтобы в `conflict.kt` остались наработки обеих веток.

# DEMO

## MASTER PROTECTION IN GITHUB

# Как защитить master от прямых коммитов - 1

The screenshot shows the GitHub repository settings for `OSLL / kotlin_circle_2017`. The repository has 6 watchers, 0 stars, and 0 forks. The navigation bar includes links for Code, Issues (0), Pull requests (1), Projects (0), Wiki, Insights, and Settings (which is highlighted).

**Left sidebar:**

- Options
- Collaborators & teams
- Branches**
- Webhooks
- Integrations & services
- Deploy keys

**Default branch**

The default branch is considered the "base" branch in your repository, against which all pull requests and code commits are automatically made, unless you specify a different branch.

Buttons: `master` (dropdown), `Update`

**Protected branches**

Protect branches to disable force pushing, prevent branches from being deleted, and optionally require status checks before merging. New to protected branches? [Learn more.](#)

Buttons: `master` (dropdown)

**Branches dropdown menu:**

- Filter branches
- ✓ `master`
- `observer_example`
- `test1`

**Footer:**

© 2017 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Contact GitHub](#) [API](#) [Training](#) [Shop](#) [Blog](#) [About](#)

# Как защитить master от прямых коммитов - 2

[Options](#)  
[Collaborators & teams](#)  
**Branches**  
[Webhooks](#)  
[Integrations & services](#)  
[Deploy keys](#)

## Branch protection for **master**

☒ **Protect this branch**  
Disables force-pushes to this branch and prevents it from being deleted.

☒ **Require pull request reviews before merging**  
When enabled, all commits must be made to a non-protected branch and submitted via a pull request with at least one approved review and no changes requested before it can be merged into **master**.

☐ **Dismiss stale pull request approvals when new commits are pushed**  
New reviewable commits pushed to a branch will dismiss pull request review approvals.

☐ **Require review from Code Owners**  
Require an approved review in pull requests including files with a designated code owner.

☐ **Restrict who can dismiss pull request reviews**  
Specify people or teams allowed to dismiss pull request reviews.

☐ **Require status checks to pass before merging**  
Choose which [status checks](#) must pass before branches can be merged into **master**. When enabled, commits must first be pushed to another branch, then merged or pushed directly to **master** after status checks have passed.

☐ **Restrict who can push to this branch**  
Specify people or teams allowed to push to this branch. Required status checks will still prevent these people from merging if the checks fail.

☐ **Include administrators**  
Enforce all configured restrictions for administrators.

Save changes

22

# Пулл-реквесты и ревью

- Код-ревью (code-review) - процедура проверки изменений участниками проекта друг у друга.
- Ревьюер просматривает пулл-реквест на Github и комментирует в нем проблемные участки.
- После комментирования возможны следующие варианты вердикта:
  - **Comment** - прокомментировать и не дать явного одобрения на мерж PR.
  - **Approve** - PR хороший, его можно мержить.
  - **Request changes** - PR требует доработок и повторного ревью.

# DEMO

## PULL-REQUEST REVIEW IN GITHUB




# Процедура Review на гитхабе - 0

[Code](#) [Issues 0](#) [Pull requests 1](#) [Projects 0](#) [Wiki](#) [Insights](#) [Settings](#)

## test PR #2 [Edit](#)

[Open](#) zmm wants to merge 1 commit into master from test1

[Conversation 0](#) [Commits 1](#) [Files changed 2](#) +3 -1



zmm commented 3 hours ago

Owner + 🗨️ ✎️


No description provided.

test commit

4905b46

Reviewers

Suggestions

 rosshkaa

Assignees

25

# Процедура Review на гитхабе - 1

The screenshot shows a GitHub Pull Request (PR) titled "test PR #2". The PR is created by user "zmm" and is currently in a "Pending" state. The PR description indicates that "zmm wants to merge 1 commit into master from test1".

The interface includes a navigation bar with links to Code, Issues, Pull requests, Projects, Wiki, Insights, and Settings. Below the navigation bar, there are tabs for Conversation, Commits, and Files changed. The "Files changed" tab is selected, showing a diff view of the code changes.

The diff view shows changes to the file "examples/K2\_simple\_OOP/Monkey2.kt". The changes include adding a new class "Monkey2" that implements the "MammalInterface". The diff shows lines 10, 11, and 12 being added, and line 13 being removed. The changes are highlighted with a green background.

On the right side of the diff view, there is a "Review changes" button. Below this button, there is a "Submit your 1 pending comment" dialog box. The dialog box contains a "Review summary" section with a text area for leaving a comment. Below the text area, there are three radio buttons for selecting the type of review: "Comment", "Approve", and "Request changes". The "Comment" option is selected. Below the radio buttons, there is a "Submit review" button.

At the bottom of the diff view, there are two buttons: "Start a new conversation" and "Finish your review".

# Рекомендуемый подход к совместной работе

- **master** под защитой.
- Все работают в отдельных ветках, начатых от мастера.
  - Одна задача == одна ветка.
  - Ветки максимально ортогональны (не пересекаются).
- Когда задача готова, исполнитель:
  - **мержит master в вашу ветку** (кто скажет зачем?) и сам решает конфликты (почему?),
  - создает **Pull-request** и назначает **reviewer**
  - **Reviewer** проверяет реквест и мержит/пишет комментарии.
- Часто в команде есть специальный человек для проверки и мержа PR.

# Задача #3

Для репозитория вашего проекта

1. Добавить комментарий в чужой PR.
2. Отметиться как reviewer.
3. Поставить вердикт **Request changes**.
4. \*Отреагировать на такой же статус вашего PR, добавить коммиты.
5. \*\*Убедиться, что коллега из пункта 1 выполнил пункт 4, поставить его **PR** статус **Approve**.

# Домашнее задание

- Включить в репозиториях **Branch protection** для **master**.
- Определить, как будут проверяться **PR** (все проверяют друг за другом или есть за это отвечает один человек).

# Материалы

- [ru] Онлайн-курс по Git <https://stepik.org/course/3145>
- [ru] Волшебство Git  
<http://www-cs-students.stanford.edu/~blynn/gitmagic/intl/ru/>
- [ru] Pro Git <https://git-scm.com/book/ru/v2>
- [en] Using Git Integration  
<https://www.jetbrains.com/help/idea/using-git-integration.html>
- [en] About pull request reviews  
<https://help.github.com/articles/about-pull-request-reviews/>