**Самостоятельная работа**

**Тема:** Управление Git-репозиторием при помощи Git Bash и при помощи веб-интерфейса GitHub.

**Теория:**

**Git** — распределённая система управления версиями. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux, первая версия выпущена 7 апреля 2005 года.

**Система контроля версий** знает о нескольких состояниях кода (как бы по вертикали). Это как современный смартфон, который сохраняет не только фотографию, но и то, что было немного до и немного после нажатия на кнопку, — можно «отмотать» время и выбрать лучший момент. Аналогично в Git хранятся все версии проекта — они называются коммитами. Можно откатить текущую версию кода до предыдущей или сразу на несколько шагов назад. Удобно, если после изменения коммита допущена ошибка.

**Основные команды:**

* git init – инициализация локального репозитория
* git status – проверить статус репозитория и файлов в нём
* git add <имя файла> – добавить конкретный файл
* git add -A – добавить все файлы
* git add . – альтернативная команда для добавления всех файлов
* git commit -m <сообщение коммита> – создание коммита
* git log – вывод списка всех коммитов репозитория
* git remote add <название> <ссылка на удалённый репозиторий> – соединение локального и удалённого репозитория (обычно для названия используют слово origin)
* git push -u <название удалённого репозитория> <название ветки> – отправить изменения в удалённый репозиторий
* git pull – получить все изменения из удалённого репозитория, если такие имеются
* git branch –a – вывести список веток
* git checkout <название ветки> – переключиться на ветку
* git checkout -b <название ветки> – создать ветку на основе текущей
* git checkout -b <название ветки> <идентификатор коммита> – создать ветку на основе определённого коммита

**Работа с .gitignore.** Обычно в репозитории мы хотим хранить только исходный код и ничего другого. Но вместе с кодом, в репозиторий могут попасть скомпилированные классы и/или файлы, которые создают среды разработки. Чтобы Git их игнорировал, есть специальный файл, который нужно создать. Делаем это: создаем файл в корне проекта с названием .gitignore, и в этом файле каждая строка будет шаблоном для игнорирования.

**Пример содержания файла .gitignore:**

\*.class

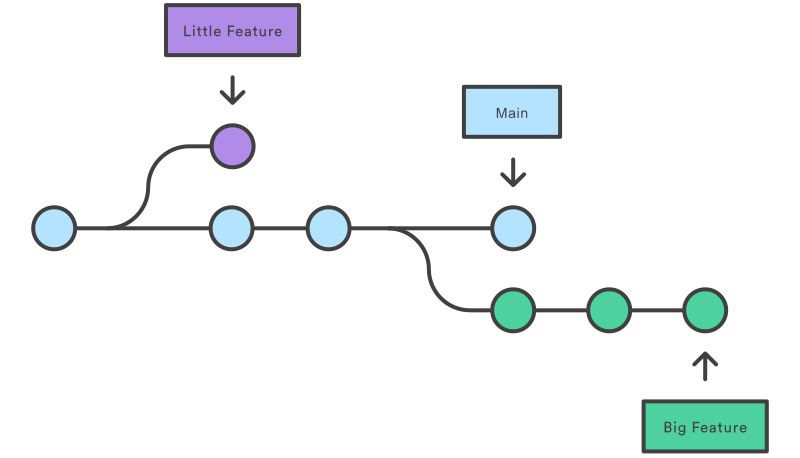
target/

\*.iml

.idea/

Первая и третья строка означают, что **все файлы с расширением** .class и .iml будут игнорироваться. А вторая и четвёртая строки означают, что **папки** target и .idea и **все файлы внутри** них будут игнорироваться.

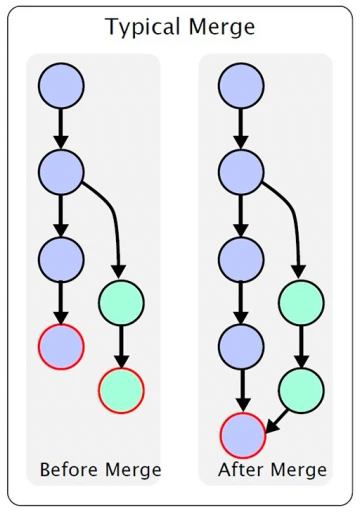
**Ветки** являются указателями на коммит. Если нужно добавить новую возможность или исправить ошибку вы создаете новую ветку, в которой будут размещаться эти изменения.



**Работа с ветками.** Чтобы посмотреть список всех веток в репозитории и понять, на какой мы находимся, нужно написать команду **git branch –a**.Чтобы создать ветку на основе конкретного коммита, необходимо:  
узнать идентификатор коммита с помощью **git log** и ввести команду **git checkout -b <название ветки> <идентификатор коммита>**.

Чтобы создать ветку на основе текущей, необходимо ввести команду **git checkout <название ветки>** для перехода на нужную ветку, после чего ввести команду **git checkout -b <название ветки>**.

**Слияние веток.**



Чтобы слить ветки необходимо ввести команду **git checkout <название ветки>** для перехода на нужную ветку, после чего ввести команду **git merge <название ветки для слияния>**.

**Задание. Для выполнения используйте Git Bash.**

1. Создайте локальный репозиторий.
2. Создайте удалённый репозиторий на GitHub.
3. Соедините репозитории при помощи команды **git remote add**.
4. В локальном репозитории смените ветку на main при помощи команды **git branch -M main**.
5. Добавьте в локальный репозиторий файл README.md и напишите туда что-либо. Сделайте коммит с любым осмысленным комментарием.
6. Отправьте изменения в удалённый репозиторий на GitHub.
7. Создайте любой другой новый файл и сделайте коммит.
8. Создайте новую ветвь на основе текущей. В новой ветви измените содержимое файла, созданного в прошлом шаге, сделайте коммит и отправьте изменения в удалённый репозиторий.
9. Переключитесь на основную ветку (main) и сделайте слияние с веткой, созданной в прошлом шаге. Отправьте изменения в удалённый репозиторий.
10. Создайте папку build. Создайте файл .gitignore и добавьте в него ранее созданную папку. Сделайте коммит и отправьте изменения в удалённый репозиторий.