Для локальных действий

[(2) #2 GIT для тестировщика / Установка GIT / Регистрация в GitHub / Первый репозиторий - YouTube](https://www.youtube.com/watch?v=zbg39ink85Q)

1. Зайти в папку «market»
2. Стоя в папке нажать правую кнопку мыши, выбрать и запустить «Git Bahe Here».

Также можно использовать консоль (на заходя в папку) – необходимо в консоли набрать команду «cd “C:\OpenServer\domains\market”»

1. Команды:

Git –version

Git init

git add . – добавляет все файлы в репазиторий для отслеживания.

**Git status –** показывает что файлы были добавлены в репозиторий, и теперь GIT будет за ними следить.

git commit -m "\*\*\*\*\*" - добавить комментарий.

Git lod – выводит историю коммитов

Git branch +название – создаёт новую ветку

Git branch – показывает какие ветки существуют

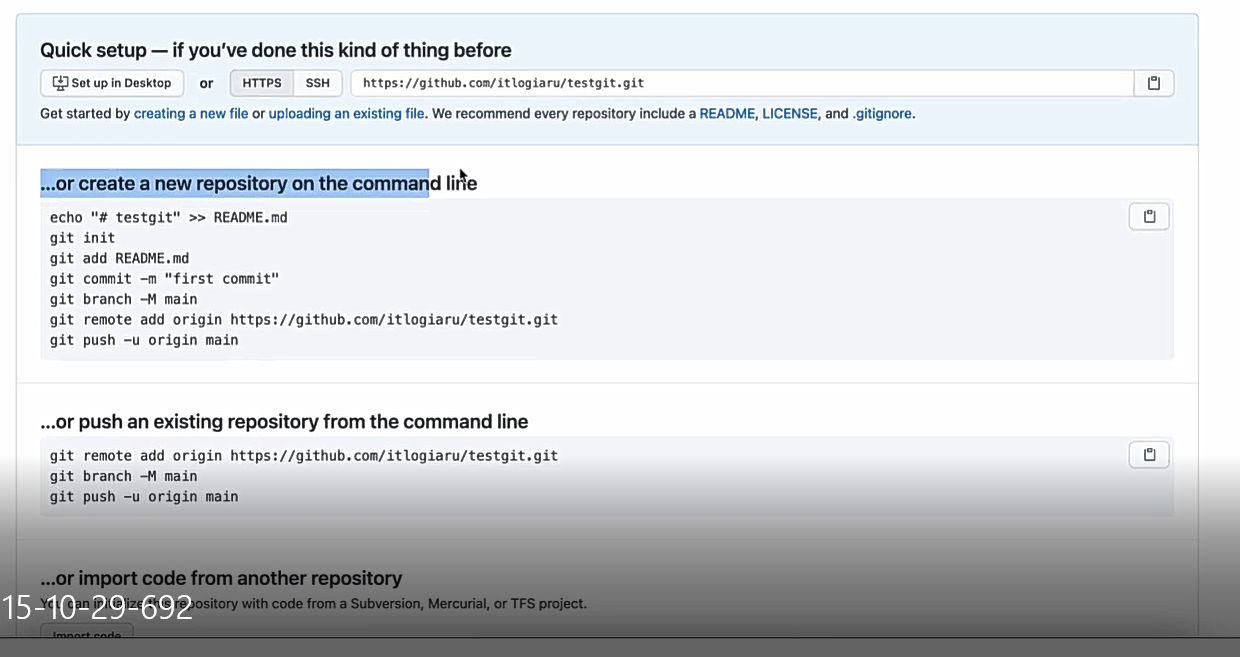
Git checkout +название ветки – позволяет переключиться на новую ветку для работы в ней

Git merge + название ветки (которое мы хотим влить **в мастер**) - все изменения, которых нет в мастере будут добавлены.

Для сервера в интернете

Для хранения используют сервер GitHub

1. Заходим в сервер GitHub
2. Создаём новый репозиторий и выбираем тип репозитория (Public или Private)
3. Необходимо связать свой код с созданным репозиторием на GitHub



* **1 пункт используется в тех случаях, когда нет проекта**
* **2 пункт используется в тех случаях, когда если у вас уже есть репозиторий на GitHub, пишет две команды:** git remote add origin <https://github.com/Artem-Galkin/market2.git> **(URL нашего репозитория), а затем команду:** git push -u origin main

1. Для добавления файлов в репозиторий необходимо обновить страницу на GitHub (также можно добавить вручную).
2. Git push – обновляет все изменения на GitHub
3. Git pull – позволяет обновить свой проект до последней версии, которая есть на GitHub, если вдруг она изменилась (другим разработчиком), также все изменения переносятся в код.
4. Также можно скачать код репозитория (в архиве). Если его скачать, то не будет никакой привязки к серверу GitHub. Если клонируете, тогда будет. То есть если Вас подключают к какому-либо проекту, то вам надо клонировать этот проект. Соответственно, если вы подключаете кого-либо к своему проекту, то ему нужно клонировать ваш проект. Указав команду: git clone +URL(репозитория) – смотри запись № 3
5. Если мы не хотим записывать какие-либо файлы и папки, находящиеся в папке, то название этих файлов и папок необходимо указать в файле «.gitignore»

Для создания новой ветки

1. Git branch+main\_market (название ветки)
2. Git branch – проверяет наличие новой созданной ветки.
3. Git checkout + название ветки – позволяет переключится на новую ветку
4. Git push -u origin+название ветки – позволяет связать ветку на локальном репозиторий на удаленном репозитории.

Залить файлы из branch в основную ветку.

Для этого необходимо вернуться в основную ветку

1. Git checkout – позволяет вернуться в основную ветку
2. Git merge + название той ветки, которую хотим влить в основную
3. Git push – обновляем наш комментарий в основной ветке
4. Git status

Конфликты при заливке

Если в удалённом репазитории добавили какой-нибудь файл, а локальном репазитории изменили какой-нибудь текущий файл. То есть появляется конфликт, т.к. два репазитория находятся не в синхронном состоянии.

1. Git pull
2. Git push

Занятие 13.1 (php) ООП

[Объектно-ориентированный PHP: работа с наследованием (ruseller.com)](https://ruseller.com/lessons.php?rub=37&id=1176)

1. Наследование
2. Полифармизм
3. Инкапсуляция
4. Абстракция

**Наследование** - Наследование базируется на понятиях класс-родитель и класс-наследник. Используя определенный синтаксис, вы можете создать класс, который будет наследовать другой класс (становится его наследником).

На заметку: классы-родители также называют базовыми классами или супер-классами. Классы-наследники, в свою очередь, можно назвать дочерними классами или подклассами.

Когда вы создаете дочерний класс, он наследует все поля и методы своего базового класса. В дочерний класс можно добавлять дополнительные поля и методы, тем самым расширяя функциональность класса-родителя.

К примеру, в веб-приложении форума есть класс Member, у которого есть методы createPost(), editProfile(), showProfile() и др. Так как администраторы форума также являются его членами, то вы можете создать класс Administrator - дочерний класса Member. Класс Administrator наследует все поля и методы класса Member, а значит, объект класса Administrator будет вести себя точно так же, как объект Member.

Вы можете расширить функциональность класса Administrator, добавив в него такие методы, как createForum(), deleteForm() и banMember(). А если хотите назначать роли еще и администраторам, то добавьте в данный дочерний класс поле, например, $adminLevel.

Действуя таким образом, вы не захламляете класс Member методами и полями, которые не подходят для обычных участников форума, а только для администраторов. Вам также не придется копировать и вставлять методы и поля из класса Member в дочерний класс. Так что наследование подойдет вам как нельзя лучше.

На заметку: вы можете создать столько дочерних классов, сколько вам понадобится, от одного единственного супер-класса, и в каждый из них можно добавлять всё новые и новые методы и поля.

1. Создаем класс «driver» - родительский класс для «bmw и mercedes».

Зачем создавать базовый класс и зачем наследоваться? – Оба класс «bmw и mercedes» имеют одинаковые методы(дублирование кода) – **не очень хорошая практика, поэтому их стоит вынести в отдельный файл «driver», а затем потом наследовать**. И использовать/вызывать их в родительском классе. Для этого удаляем метод «getname» из дочернего класса (bmw и mercedes) и перенесем этот метод в класс «driver».

<?php

/\*\*

 \* Базовый(родительский) класс автомобилей

 \*/

class Driver{

    /\*\*

     \* Доступен везде и всем.

     \* Метод для получения названия авто.

     \* @return string - название авто (что возвращать)

     \*/

    Public function getName()

    {

        return $this->name;

    }

}

То есть, при объявлении нашего класса и вызове метода «getname» мы будем вызывать в классе «driver». Чтобы его вызвать.

Итак, как же создать класс, который станет наследником другого класса, в PHP? Для этого существует ключевое слово extends:

С помощью ключевого слова «extend(s)» происходит наследование класса «driver» в классе «bmw, Mercedes – class Bmw extends driver, class Mercedes extends driver». **В данном случае будет показывать ошибку, т.к. файл Bmw и Mercedes не знают, где лежит класс «Driver».** Потому что надо «инклюдить» файл «driver» c помощью команды «include». Для этого заходим в файл Bmw и Mercedes и прописываем к каждому include «driver.php»;

**Так как система видит, что мы два раза подключаем один и тот же файл:**

**Include “driver.php”, что бы это исключить, надо сделать Include\_once “driver.php”.**

То есть сказали интепритатору PHP – не важно сколько раз я подключал, подключи мне один раз этот файл, то второй раз он его не будет подключать, а будет его только использовать.

1. Переносим класс privet $name из дочернего класса в родительский класс и переименовываем protected $name, чтобы она была доступна в дочерних классах, которые мы наследуем.

**Полифармизм –**

Теперь при получении имени, мы будем всегда к марки «Bmw» приписывать название «Concern».

В данном случае мы полностью переопределили метод getname

**Инкапсуляция**