Инструкция по сдаче заданий через GitHub

# Начало работы с GitHub

**О чем речь?**

** GitHub** – это открытый ресурс на основе системы **Git** для хранения, обмена и управления разработкой программного обеспечения. Он позволяет контролировать изменения, вносимые в исходный код, и объединять результаты работы нескольких разработчиков в одну готовую программу. А еще это удаленное хранилище в сети, к которому можно получить доступ с любого компьютера.

GitHub бесплатен при условии, что ваши файлы будут публичными, т.е. доступными для просмотра для всех пользователей. Приватные хранилища доступны за отдельную плату.

Не следует путать GitHub и саму систему контроля версий Git.

**Что делать?**

1. Зарегистрироваться на [GitHub](https://github.com/)
2. Зайти в свой профиль: кнопка с аватаром в правом верхнем углу → Your Profile → кнопка Edit Profile.
3. В поле Name вести имя и фамилию (отчество – по желанию) на русском языке. В поле Company ввести «СГЭУ». Подтвердить изменения кнопкой Update profile.
4. Найти профиль преподавателя через поиск по GitHub или по ссылке: <https://github.com/KorobetskayaAA>.
5. На странице **профиля преподавателя** нажать кнопку Follow. После этого вы должны появиться в списке подписчиков (Followers).

# Создание собственного репозитория

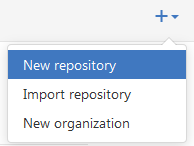
**О чем речь?**

**Репозиторий** (хранилище) – это то место, где хранятся все ваши файлы. Один пользователь может иметь несколько репозиториев. Обычно для каждой программы создается отдельный репозиторий.

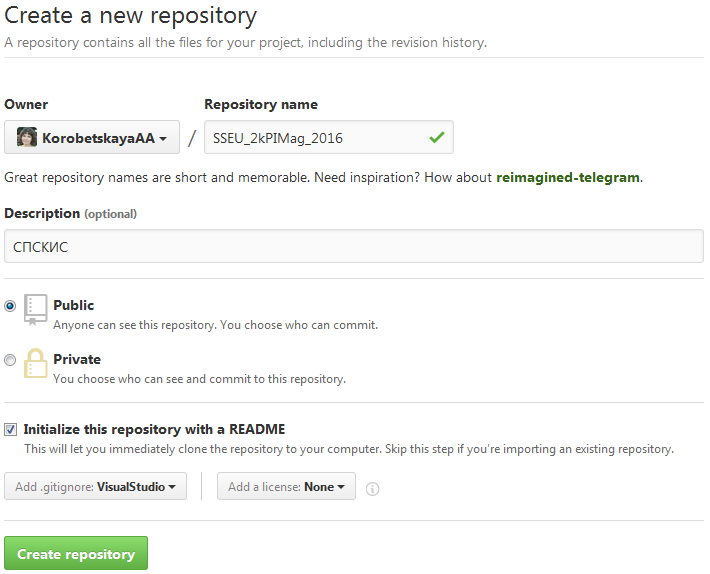
Для сдачи заданий вы будете сохранять все задания в один репозиторий, но в отдельные папки.

**Что делать?**

1. Зайти на GitHub под своей учетной записью.
2. Нажать кнопку с плюсом в правом верхнем углу → New Repository.



1. Задать настройки нового репозитория:
   1. Задать имя репозитория (Repository name). Имя должно начинаться с приставки «SSEU\_», далее указать курс, группу и год, например, SSEU\_2kPIMag\_2016 (2 курс, прикладная информатика, магистры, 2016 год).
   2. В описании (Decription) указать предмет на русском языке (можно сокращенно).
   3. Тип репозитория – открытый (Public, выбрано по умолчанию).
   4. Поставить флажок Initialize this repository with a README. В readme-файл будет добавлено описание репозитория.
   5. В списке Add .gitignore выбрать Visual Studio.



1. Нажать кнопку Create repository.
2. После этого вы попадете на главную страницу созданного репозитория. Он уже должен содержать два файла: .gitignore и README.md.
3. Изменить или удалить репозиторий можно через кнопку  Settings.

# Работа с репозиторием через клиент

**О чем речь?**

Вы можете просматривать и редактировать файлы в репозитории прямо через браузер. Но обычно программы пишутся в специальной среде (в нашем случае – Microsoft VisualStudio).

Программа-клиент подключается к Git-репозиторию и позволяет загрузить файлы с/на локальный компьютер.

Классический способ управления Git – через командную строку. Но сегодня существует много клиентов с графическим интерфейсом. Рекомендуется использовать **GitHub Desktop** (специально для GitHub, простой минималистичный интерфейс) или **SmartGit** (есть portable версия, более подробный интерфейс).

Клиентская программа «стыкует» между собой рабочую папку на локальном компьютере и удаленный репозиторий, а также контролирует изменение файлов в локальном хранилище и запоминает историю их изменений и веток.

**Основные действия, команды и термины:**

**Pull** – загрузить файлы из удаленного репозитория в локальный.



**Pull**

**Push** – загрузить файлы из локального репозитория в удаленный. Будут загружены только изменения, зафиксированные через Commit (см. ниже).



**Push**

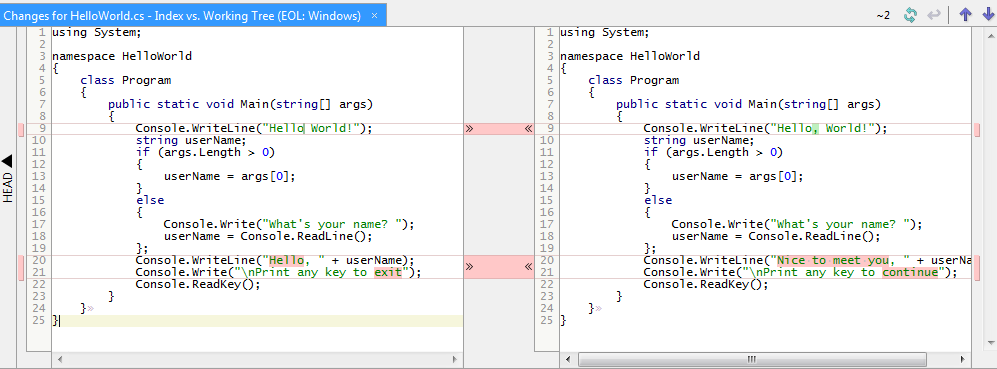
**Sync** – синхронизировать файлы в локальном и в удаленном репозитории. Файлы будут заменены на более новые.



**Sync**

**Commit** – подтвердить (зафиксировать) текущую версию программы в локальном хранилище. При этом выгрузка в удаленный репозиторий **не** происходит. Программа просто запомнит состояние всех файлов.

В дальнейшем можно будет отследить изменения или откатиться к предыдущей версии.



Каждый коммит **обязательно** сопровождается **комментарием**. В нем нужно описать суть выполненных изменений. Например, «добавлена функция авторизации через соцсети» или «исправлен баг с расчетом почасовой оплаты труда».

Коммитить код нужно достаточно часто – после каждой выполненной задачи или добавленной функции. Чем более мелкими порциями коммитится код (в пределах разумного, конечно), тем проще отслеживать его изменения и тем проще объединять разные версии файлов.

**Branch** – ветка разработки. В отдельные ветки выделяются разрабатываемые фрагменты программы, которые еще не вошли в финальную сборку, пробные идеи, новые фичи и т.п.



Также, если разработку ведет несколько человек, то у каждого может быть своя ветка. После того, как работа над веткой завершена, она переносится в основной код (или в другую, более крупную, ветку). Если от изменений, вносимых веткой, решено отказаться, то ее можно просто удалить.

**Master** – главная ветка в репозитории, созданная по умолчанию. Здесь должен находиться только окончательный, протестированный и готовый к релизу код.

**Внимание!** Проверяться будет только код из ветки master, если только в задании явно не указано создать другую ветку.

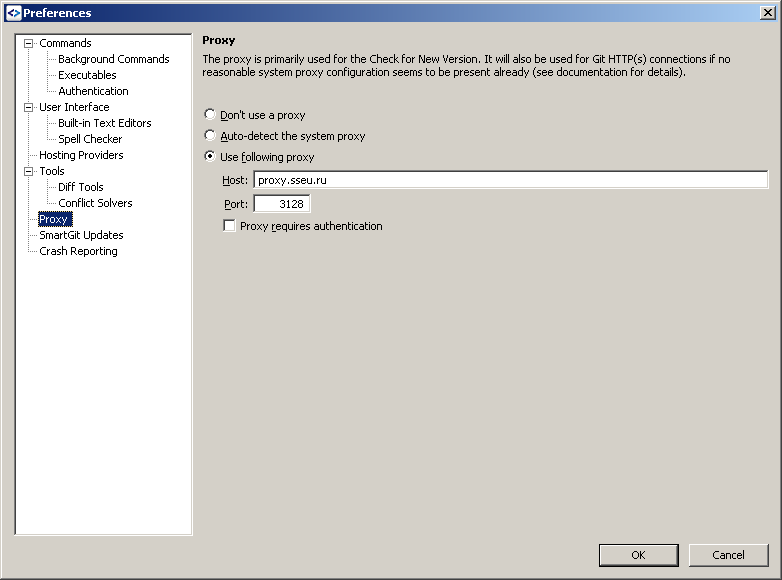
**Merge** – объединение (слияние) веток: файлы одной ветки присоединяются к другой. Ядро Git сравнит файлы и их содержимое в двух ветках. Отсутствующие файлы будут просто скопированы. Если какой-то файл присутствует в обеих ветках, то он покажет, в чем их отличия и разработчик сможет выбрать, какой вариант следует оставить.

## Что делать?

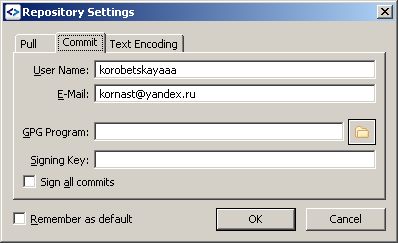
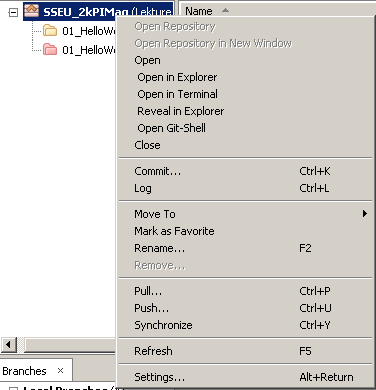
1. Скачать, установить и запустить клиент (далее на примере SmartGit).
2. При первом запуске выбрать в меню Repository → Clone... Указать ссылку на созданный вами репозиторий. Далее указать локальную папку, в которой будут храниться все ваши программы, например, «D:\\GitHub\». В ней будет автоматически создана подпапка с именем вашего репозитория.
3. При последующих запусках, если ваш репозиторий не открылся автоматически, выбрать в меню Repository → Add or create… и указать созданную вами папку.
4. Для каждого задания создавать **отдельную подпапку** внутри репозитория. Имя подпапки должно начинаться с номера задания, например, «01\_HelloWorld». В папке размещать все необходимые исходные файлы и ресурсы. Работайте с этой подпапкой напрямую из VisualStudio.
5. Создавать отдельные ветки для незаконченных версий каждого задания.
6. Не забывать коммитить промежуточные изменения.
7. EXE-файл коммитить не нужно, только исходники. Можно добавить EXE-файлы в игнорируемые через меню Local → Ignore…
8. Не забывать выгружать закоммиченные изменения на GitHub командой Push.
9. Зачитываются только программы, находящиеся в ветке Master на GitHub. Если вы хотите, чтобы преподаватель проверил другую ветку перед объединением с Master, попросите об этом явно.

## Настройки SmartGit в портативной версии для работы из СГЭУ

1. Доступ в Интернет из СГЭУ осуществляется через прокси-сервер. В настройках SmartGit (меню Edit → References → раздел Proxy) необходимо указать proxy.sseu.ru, порт 3128, либо выбрать «Auto-detect the system proxy».



1. Для успешного коммита SmartGit нуждается в идентификационных данных пользователя. Эти настройки могут сброситься при удалении и добавлении репозитория (что в портативной версии происходит очень часто). Кликните правой кнопкой по имени репозитория, выберите в меню Settings…, перейдите на вкладку Commit и укажите свой логин и пароль.



# Что еще почитать/посмотреть?

1. Начинаем работать с системой контроля версий GIT (для чайников). URL: <http://all-ht.ru/inf/prog/git_001.html>

*Коротко обо всем. Рекомендуется как справочник во время занятий.*

1. Образовательный IT-портал GeekBrains. Курс "Git. Быстрый старт". URL: <https://www.youtube.com/playlist?list=PLmRNNqEA7JoM77hOJkPrLOfJQGizCLR3P>

*Хорошие подробные уроки по работе через SmartGit. Рекомендуется к просмотру.*

1. GitHub Desktop – обзор графического Git клиента. URL: <https://www.youtube.com/watch?v=v7GbxUnwRY8>

*Обзор работы в GitHub Desktop и общие рекомендации по работе через Git.*

1. Как сделать репозиторий на Github Pages и как закоммитить в него изменения через Github for Desktop без командной строки. URL: <http://randomfederation.github.io/>

*Краткая инструкция по GitHub Desktop.*