Лабораторная работа 1

Тема работы:

Первоначальная подготовка среды Git. Создание репозитория Git и локальный репозиторий.

Теоретическая часть:

Git — это бесплатная распределенная система контроля версий с открытым исходным кодом, предназначенная для быстрой и эффективной работы с любыми проектами - от небольших до очень крупных. Git прост в освоении, занимает мало места и обладает молниеносной производительностью. Он превосходит такие SCM (Supply Chain Management – пер. Управление цепями поставок)-инструменты, как Subversion, CVS, Perforce и ClearCase, благодаря таким возможностям, как дешевое локальное ветвление, удобные области хранения и множество рабочих процессов.

Ветвление и слияние

Особенностью Git, которая действительно выделяет его среди практически всех других SCM, является модель ветвления. Git позволяет и поощряет создание нескольких локальных веток, которые могут быть совершенно независимы друг от друга. Создание, объединение и удаление этих ветвей занимает считанные секунды.

Это означает, что вы можете делать такие вещи, как:

- **Бесконтактное переключение контекста.** Создайте ветку для опробования какой-либо идеи, сделайте несколько фиксаций, вернитесь к тому месту, откуда была сделана ветка, примените исправление, вернитесь к тому месту, где проводились эксперименты, и объедините его.
- **Ролевые коделайны.** Иметь ветку, которая всегда содержит только то, что отправляется в продакшн, другую, в которую сливается работа для тестирования, и несколько небольших веток для повседневной работы.
- **Рабочий процесс, основанный на функциях.** Создавайте новые ветки для каждой новой функции, над которой вы работаете, чтобы можно было плавно переключаться между ними, а затем удаляйте каждую ветку, когда функция будет объединена с основной линией.
- **Одноразовые эксперименты.** Создайте ветку для экспериментов, поймите, что она не работает, и просто удалите ее, бросив работу, и никто больше не увидит ее (даже если за это время вы продвинули другие ветки).

Распределенный

Одной из самых приятных особенностей любой распределенной SCM, в том числе и Git, является ее распределенность. Это означает, что вместо того, чтобы выполнять "выдачу" текущей вершины исходного кода, вы выполняете "клонирование" всего репозитория.

Множественное резервное копирование

Это означает, что даже если вы используете централизованный рабочий процесс, каждый пользователь, по сути, имеет полную резервную копию главного сервера. Каждая из этих копий может быть поднята вверх, чтобы заменить основной сервер в случае сбоя или повреждения. По сути, в Git нет единой точки отказа, если только не существует единственной копии репозитория.

Любой рабочий процесс

Благодаря распределенной природе Git'a и превосходной системе ветвлений можно с относительной легкостью реализовать практически бесконечное число рабочих процессов.

Рабочий процесс в стиле Subversion

Централизованный рабочий процесс встречается очень часто, особенно у людей, переходящих с централизованной системы. Git не позволит вам выполнить push, если кто-то выполнил push с момента последнего извлечения, поэтому централизованная модель, при которой все разработчики выполняют push на одном сервере, работает просто замечательно.

Установка Git

Скачивайте установщик (https://git-scm.com/downloads), запускаете установщик, выбираете нужные для вас опции (либо оставляете все как есть, если не сильно разбираетесь), ожидаете завершения процесса установки и готово.

Использование Git

В рабочей папке где находятся исходные файлы вашего приложения/проекта, через терминал или командную строку используете команду **git init** (оф. документация - https://git-scm.com/docs/git-init) для инициализации локального репозитория, после чего Git будет отслеживать изменения исходных файлов и для фиксирования этих самых изменения - используйте команду **git commit** (оф. документация - https://git-scm.com/docs/git-commit). В дальнейшем если необходимо синхронизовать изменения в общем для всех репозитории – используйте команду **git push** (оф. документация - https://git-scm.com/docs/git-push)

Основное задание:

1. Установить среду Git на ваше устройство, подготовьте папку для вашего приложения/проекта и проинициализируйте локальный репозиторий в этой папке (команда git init).
2. Реализуйте простое приложение. В команде, распределите кто какую часть приложения будет реализовывать, что эта часть будет из себя представлять и какую(-ие) функцию(-и) будет(-ут) выполнять.
3. Сделайте commit-ы на изменения в коде приложения/проекта (команда git commit). Создайте аккаунт на GitHub (https://github.com/) (если нет такого) и отправьте преподавателю юзернейм или ссылку на профиль. После получения дальнейших указаний от преподавателя, push-ите изменения в вашу отдельную ветку в репозитории на GitHub (либо если вы согласовали реализацию собственного проекта, то на отдельный для проекта репозиторий) (команда git push).

Варианты для реализации задания:

1. Приложение, в котором выполняются простые арифметические операции – сложение, вычитание, умножение, деление, возведение в степень, извлечение корня. Каждый член команды реализует класс, выполняющий одно или два арифметический действия. В основном алгоритме (классе) реализовать взаимосвязь, как минимум, между двумя классами.
2. Приложение, в котором извлекаются конкретные части информации из введенного текста. Каждый член команды реализует класс, извлекающий конкретно необходимую информацию. В основном алгоритме (классе) реализовать взаимосвязь, как минимум, между двумя классами. (примечание: стараться прибегать к использованию регулярных выражений)
3. Приложение, в котором производятся манипуляции с текстом – удаление пробелов, переносов на новую строку, отзеркалить текста, отдельно слова, поменять слова, буквы в словах местами. Каждый член команды реализует класс, который производит конкретные манипуляции с текстом. В основном алгоритме (классе) реализовать взаимосвязь, как минимум, между двумя классами.

4. Приложение, в котором производятся манипуляции с файлами – примеры манипуляции аналогичны из предыдущего варианта. Каждый член команды реализует класс, который производит конкретные манипуляции с файлом. В основном алгоритме (классе) реализовать взаимосвязь, как минимум, между двумя классами.
5. Приложение, которое вычисляет геометрические свойства, такие как площадь и периметр, радиус для различных фигур (например, квадратов, прямоугольников, кругов, треугольников). Каждый член команды реализует класс, который вычисляет конкретное свойство для определенной фигуры. В основном алгоритме (классе) реализовать взаимосвязь, как минимум, между двумя классами.
6. Приложение, которое рекомендует рецепты на основе предпочтений пользователя и имеющихся ингредиентов. Каждый член команды реализует класс для рекомендации рецептов определенной кухни или категории. В основном алгоритме (классе) реализовать взаимосвязь, как минимум, между двумя классами.
7. Приложение для конвертации различных валют. Каждый член команды реализует класс, который обрабатывает конвертацию на определенную валюту. В основном алгоритме (классе) реализовать взаимосвязь, как минимум, между двумя классами.
8. Приложение, имитирующее бросание игральных костей и отслеживающее результаты. Каждый член команды может реализовать класс для моделирования определенного типа игральных костей (например: шестигранных, двенадцатигранных). В основном алгоритме (классе) реализовать взаимосвязь, как минимум, между двумя классами.
9. Приложение, которое имитирует выбор призрака (каждый призрак является классом). Базируясь на введённых параметрах пользователем (параметры - числа или текст) программа выдаст определённый вид призрака. Каждый член команды должен реализовать класс для призрака с тремя параметрами для определения. В основном алгоритме (классе) реализовать взаимосвязь, как минимум, между двумя классами.

По завершению работы, составьте отчет, в котором должно быть – Ваша фамилия, имя, группа, тема работы, Ваш вариант для реализации задания, краткое описание реализации задания, ссылку на исходный код на GitHub. Сохранить отчет в формате PDF и отправить на ELSE.