Лабораторная работа № 8 GIT

«Работа с системой контроля версий GIT»

Цель работы:

Получить опыт практической работы с системой контроля версий на примере *GIT*.

Краткие теоретические сведения:

Система управления версиями (от англ. *Version Control System или Revision Control System*) — программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости, возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение и многое другое.

Общие сведения:

Ситуация, когда электронный документ за время своего существования претерпевает ряд изменений, достаточно типична. При этом часто бывает важно иметь не только последнюю версию, но и несколько предыдущих. В простейшем случае, можно просто хранить несколько вариантов документа, соответствующим образом их нумеруя. Но такой способ неэффективен (приходится хранить несколько практически идентичных копий), требует много внимания и дисциплины и часто ведёт к ошибкам. Поэтому были разработаны средства для автоматизации этой работы.

Большинство систем управления версиями используют централизованную модель, когда имеется единое хранилище документов, управляемое специальным сервером, который и выполняет большую часть функций по управлению версиями. Пользователь, работающий с документами, должен сначала получить нужную ему версию документа из хранилища; обычно создаётся локальная копия документа, т. н. «рабочая копия». Может быть получена последняя версия или любая из предыдущих, которая может быть выбрана по номеру версии или дате создания, иногда и по другим признакам. После того, как в документ внесены нужные изменения, новая версия помещается в хранилище. В отличие от простого сохранения файла, предыдущая версия не стирается, а тоже остаётся в хранилище и может быть оттуда получена в любое время. Сервер может использовать т. н. дельта-компрессию - такой способ хранения документов, при котором сохраняются только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.

Иногда создание новой версии выполняется незаметно для пользователя (прозрачно), либо прикладной программой, имеющей встроенную поддержку такой функции, либо за счёт использования специальной файловой системы. В этом случае пользователь просто работает с файлом, как обычно, и при сохранении файла автоматически создаётся новая версия.

Часто бывает, что над одним проектом одновременно работают несколько человек. Если два человека изменяют один и тот же файл, то один из них может случайно отменить изменения, сделанные другим. Системы управления версиями отслеживают такие конфликты и предлагают средства их решения. Большинство систем может автоматически объединить (слить) изменения, сделанные разными разработчиками. Однако такое автоматическое объединение изменений, обычно, возможно только для текстовых файлов и при условии, что изменялись разные (непересекающиеся) части этого файла. Такое ограничение связано с тем, что большинство систем управления версиями ориентированы на поддержку процесса разработки программного обеспечения, а исходные коды программ хранятся в текстовых файлах. Если автоматическое объединение выполнить не удалось, система может предложить решить проблему вручную.

Часто выполнить слияние невозможно ни в автоматическом, ни в ручном режиме, например, если формат файла слишком сложен или, вообще, неизвестен. Некоторые системы управления версиями дают возможность заблокировать файл в хранилище. Блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла (например, средствами

файловой системы) и обеспечивает, таким образом, исключительный доступ только тому пользователю, который работает с документом.

Многие системы управления версиями предоставляют ряд других возможностей:

- Позволяют создавать разные варианты одного документа, т. н. ветки, с общей историей изменений до точки ветвления и с разными — после неё.
- Дают возможность узнать, кто и когда добавил или изменил конкретный набор строк в файле.
- Ведут журнал изменений, в который пользователи могут записывать пояснения о том, что и почему они изменили в данной версии.
- Контролируют права доступа пользователей, разрешая или запрещая чтение или изменение данных, в зависимости от того, кто запрашивает это действие.

Порядок выполнения работы с использованием GIT индивидуально:

1. Скачать и установить систему контроля версий GIT.
2. Создать локальный репозиторий.
3. Выложить в репозиторий свой тестовый проект.
4. Добавить в тестовый проект новый класс. Изменить существующий код. Выложить в репозиторий.
5. Осуществить откат к старой версии выложенного в репозиторий проекта.
6. Удалить локальную копию проекта и скачать последнюю версию из репозитория.

Порядок выполнения работы с использованием GIT совместно:

1. Создается проект на общем компьютере или в сети Интернет (произвольно).
2. Каждая бригада делает чекаут
3. Предусмотреть использование branches и после выполнения задания – делать объединения в главную ветку.
4. Апдейт можно выполнять только для проектов, которые могут компилироваться
5. Код писать строго в соответствии с code convetion.
6. Требуемый функционал реализовывать в виде отдельного класса, размещенного в отдельном файле, содержащем в названии фамилии разработчиков.
7. Обязательно предусмотреть создание новых версий.
8. К разработанному функционалу предусмотреть создание юнит-тестов
9. Предусмотреть обработку возможных исключительных ситуаций и ошибок неправильного вода данных.

Общее задание: необходимо выполнять операции над объектами типа String, содержащимися в коллекции ArrayList.

1. Добавление и удаление объектов.
2. Поиск одинаковых элементов с подсчетом совпадений
3. Выгрузка в xml-файл.
4. Реверс всех строк, входящих в коллекцию
5. Статистика по всем символам, содержащимся в строках коллекции
6. Поиск подстроки в строках коллекции
7. Инициализация листа по текстовому файлу и вывод содержимого коллекции на экран.
8. Расширить функциональность класса ArrayList методом compareInnerObjects (int firstIndex, int secondIndex)
9. Посчитать длины срок входящих в коллекцию, и вывести результат в упорядоченном виде.
10. Реализовать возможность добавления в динамическую коллекцию, как если бы она была статической размерности, т.е. задаем статическую размерность коллекции, пока количество объектов меньше заданной размерности, происходит только добавление объектов, при достижении порогового значения, добавление нового элемента вызывает удаление первого элемента коллекции. Проверить, компилится ли данный проект, прежде чем записывать его в репозитарий.

Контрольные вопросы:

1. Назначение систем версионного контроля?
2. Основные команды, которые можно выполнять?
3. Зачем делать обновления перед тем, как «сливать» свои изменения в общий проект?