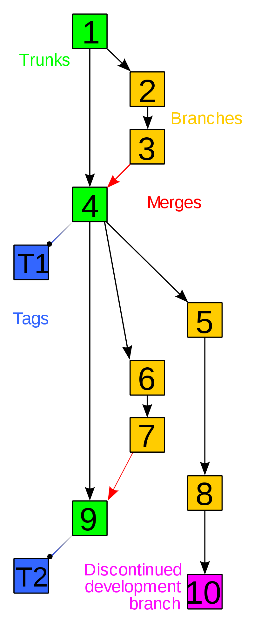
Software Engineering, University of Oulu

**Exercise 1: Version Control**

****

**1. Version Control**

Have you ever found yourself copying and pasting files as a backup strategy or spamming Ctrl+Z after your changes did not work as expected? In software development, projects can often become complex, involving numerous files, updates, features, and contributors. That's where version control steps in as a professional solution to manage and track these changes.

Version control is a powerful tool that changed the way we handle project files. Teams can seamlessly collaborate, allowing concurrent work on the same projects without conflicts. Version control also offers a way to save previous versions of a project, acting as a reliable backup mechanism.

https://en.wikipedia.org/wiki/Version\_control#/media/File:Revision\_controlled\_project\_visualization-2010-24-02.svg

With version control, you no longer need to worry about losing crucial work or struggling to find out who made which changes. It is a true game-changer for any project of any size.

So, whether you're a seasoned developer or starting out, utilizing version control will improve your workflow and provide a safety net for all your valuable work.

**2. Git**

[Git](https://en.wikipedia.org/wiki/Git) is an extremely popular version control system. It is both free and open-source under the GPL-2.0-only license. Git was originally created by a Finnish software engineer Linus Torvalds with a goal of providing a powerful framework for tracking changes in source code and coordinating the work of multiple contributors.

https://en.wikipedia.org/wiki/Git#/media/File:Git-logo.svg

Today, a great number of software projects rely on Git, and understanding it is essential for every software engineer. You will encounter Git in nearly all software projects, both in studies and working life, so make sure you understand the fundamentals thoroughly.

**2.1 Installing Git**

If you do not happen to have Git installed already, you can download it here:

<https://git-scm.com/>

**2.2 Using Git**

If you have not used Git before, you should learn and understand the [basics](https://www.youtube.com/watch?v=hwP7WQkmECE) before proceeding.

For practice, you can complete the first few exercises on <https://learngitbranching.js.org/>.

Some code editors offer a GUI for working with Git, and GitHub has their own GitHub Desktop for that purpose. However, it is useful to learn the use of the command line since you will be using it at some point in your career.

Some of the most useful Git commands include:

|  |  |
| --- | --- |
| git init | Initialize a new empty Git repository |
| git clone | Create a local copy of a remote repository |
| git branch | Add a new branch, view all existing branches or delete a branch |
| git checkout | Switch to a branch |
| git add | Add changes to the staging area |
| git commit | Save the changes in the staging area in your local repository |
| git push | Push the committed changes to the remote repository |
| git pull | Fetch the last pushed changes from the remote server |
| git merge | Merge a branch with the parent branch |
| git status | View an overview of the current status of your repository |

**3. Exercise**

1. Create a GitHub classroom environment: <link here>. This works as the remote repository for this exercise.
2. Clone the remote repository to your computer using Git commands. The created directory is your local repository.
3. Find a way to examine all branches and their names in the repository. Then, merge the two development branches into the master branch.
4. After merging, you should encounter a merge conflict. Fix it accordingly so that you keep both functions.
5. You can find a TODO.txt file in the repository. Exclude it from the repository and edit the .gitignore file so that all .txt files are ignored by Git if any are added in the future.
6. Create a README.md file for your repository and write some freeform content into it.
7. Make a commit where you include all your changes above.
8. Push the changes to the created remote repository.
9. Return a link to your repository in Moodle.

Note #1: If you cannot find more than one branch, the “-a” flag might come in handy.