Git + GitHub Lab

**Git** is free and open source software for distributed version control: tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development. **GitHub** is an Internet hosting service for software development and version control using Git.

Some of the advantages of using Git & GitHub for software development include:

* **Collaboration** - Can easily merge code implemented by team members.
* **Backup** - Cloud storage for the project folder.
* **Version Control** - Can easily revert to a previous version.
* **Tracking** - Compare code changes done by contributors.

Further reading on the advantages of using Git, you may check the following links:

* <https://guide.quickscrum.com/git-guide/>
* <https://www.atlassian.com/git/tutorials/what-is-version-control>
* <https://www.nobledesktop.com/blog/what-is-git-and-why-should-you-use-it>

To use **Git** on your computer, you need to download and install Git that is compatible with your operating system:

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

While we can use Git on our local computer or server, to take full advantage of its capabilities, we also need a Git repository hosting service. **GitHub** is one such service that we are going to use for CS102. You can sign up using the following link:

* <https://github.com/>

You need to have a GitHub account to complete this lab assignment. If you already have a GitHub account, you may use it; you do not need to create a new one for the course.

You may use **GitHub** through its website directly. The web interface allows creating new repositories, adding files to your repositories, editing the existing files, and commenting on the activities.

Using the command line interface, you can also use Git to work on your GitHub repository.

# GitKraken

The command line interface of Git or the web interface of GitHub does not provide tools to easily handle various cases like merge conflicts. By utilizing tools like **GitKraken**, version control becomes a better experience:

* <https://www.gitkraken.com/>

You may check the following video for a brief introduction to GitKraken:

* <https://youtu.be/RiAeNSFjjLc>

The following is the recording of the CS102 GitHub Tutorial from earlier years:

* <https://youtu.be/lm--GtoYNN0>
* <https://youtu.be/4mKYtIS11HM> (Turkish version of the same tutorial)

# Common Git Concepts

Some of the Git concepts that we are going to use a lot are as follows:

* **Repository:** This is a folder that is tracked by Git; there are two versions:
  + **Local Repository:** This is the local folder sitting on your computer.
  + **Remote Repository:** This is the cloud copy of that folder on GitHub.
* **Clone:** To make a copy of a remote repository on your computer. The remote repository can belong to some other user.
  + If you own the remote repository or are a contributor to it, you can affect the remote repository using its clone.
  + Otherwise, the changes you make do not directly influence the remote repository.
* **Fork:** Making a personal copy of a remote repository on GitHub. This time you have a personal copy that is sitting on your remote repository. You can edit it as you want and keep track of its different versions.
* **Pull Request:** To propose the changes you make to the owners of a repository. Usually, you first fork an existing project, make some changes to it, and then open a pull request to the original project when working on open-source projects.
* **Branch:** This is a parallel version of a repository. You may think of it as the same folder in alternate universes. This is usually beneficial when you want to introduce some new functionality without disturbing others. For example, while adding network capabilities, we may ruin the whole project, so we work on an alternate universe, solve all the problems, then finally try to combine everything together. When you first start a repository, there is the main branch.
* **Merge:** Combining two branches or combining two different versions of a repository. If there are no conflicts, the resulting branch or version contains a combination of the two. There can be merge conflicts, though.
* **Merge Conflict:** When trying to merge, some files may have conflict. The one that makes the merge should resolve these. For example, the same line may be edited differently by two different users; which update to keep? Maybe a combination of the two…
* **Staging:** The process of choosing which files are good to be included in the new version of the repository.
* **Commit:** The act of approving the staged changes to the repository. Usually, a commit message is added to summarize what has been done in that repository version.
* **Push:** The act of sending the committed changes to the remote repository.
* **Pull:** Getting the latest version from the remote repository.

For how to do these tasks on GitKraken, you may look at the following pages:

* [Interface Basics](https://help.gitkraken.com/gitkraken-client/interface/)
* [Introduction to Git with GitKraken](https://help.gitkraken.com/gitkraken-client/guide/)
* [Open, Clone, and Init](https://help.gitkraken.com/gitkraken-client/open-clone-init/)
* [Staging](https://help.gitkraken.com/gitkraken-client/staging/)
* [Branching and Merging](https://help.gitkraken.com/gitkraken-client/branching-and-merging/)
* [Pushing and Pulling](https://help.gitkraken.com/gitkraken-client/pushing-and-pulling/)
* [Pull Requests](https://help.gitkraken.com/gitkraken-client/pull-requests/)

*Note that if you are using the computers at the lab for using GitKraken the installation process is easier if you use Windows.*

# GitHub Assignment 1: The First Test

For this assignment, you are going to fork and work on a simple Java project.

1. Fork the following project: <https://github.com/sinansonlu/CS102-Git-Tutorial>
2. Fix the problems with the code; there are some hints in the README file on the repository’s page. Make sure the code is running and you get the correct output.
3. Stage your changes, commit with a meaningful commit message, and push your changes to your remote repository.
4. Open a pull request to the original repository, proposing your changes.
5. Show the working program and your repository’s commit history to the TA/Tutor in the lab.

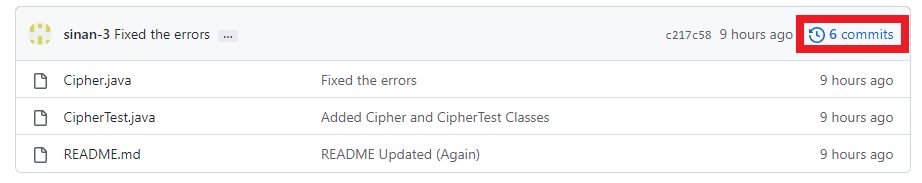
# GitHub Assignment 2: Collaboration

For this assignment, you are going to create a tiny Java project with a group of 3-5 students. This group is valid for only this assignment; you do not have to team up with your term project members, but if you already found your group, then doing this assignment with your project group is preferred.

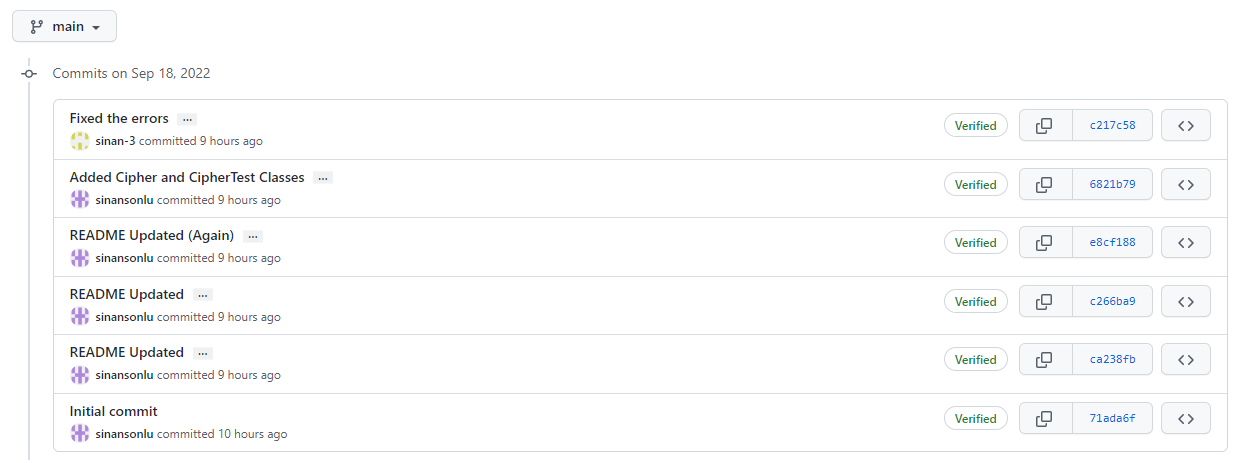
1. Choose one person to create the repository. This person should add other members as collaborators to this repository (On the GitHub page of the remote repository Settings > Collaborators > Add People).
2. Together, you are going to implement a menu-driven console application that supports the following operations:
   1. Given an array size, create an int array that contains random integers in the range [0,100]. All the remaining functionality will operate on this array.
   2. Display the menu options, get the user’s choice, and call the required methods to print out the result. Until the user chooses to exit, you should display the menu again.
   3. Find the minimum and the maximum of the array; these should be implemented as different methods.
   4. Find the average of the array. Display how each element of the array differs from the average. For example, if the array is {2, 6, 9, 3, 5}, and the average is 5, then the differences from the average would be {-3, 1, 4, -2, 0}.
   5. Find the sum of elements with odd- and even-numbered indexes. For example, if the array is {1, 5, 8, 9, 10, 2, 6}, the sum of the odd indexes is 5 + 9 + 2, and the sum of the even indexes is 1 + 8 + 10 + 6, assuming indexes start at 0.
3. Each person in the group should be responsible for one or more tasks above. You may use different class files if necessary, you need to include a main method to show your working program. You should work on different computers, push your commits to the remote repository and manage merge conflicts when necessary. Each person in the group should have at least one commit to the remote repository, we should be able to see that in the history of the remote repository.
4. As a group, show the working program and your repository’s commit history to the TA/Tutor in the lab.

# HW1 - GitHub Homework (Due on February 11, 23:55)

After completing the two GitHub assignments, go to your copy of the remote repository on GitHub and click on the commit history.



You will see a page that summarizes the commits of the repository.



Save that page as a PDF, both for GitHub Assignment 1 (GitHub-Assignment-1.pdf) and GitHub Assignment 2 (GitHub-Assignment-1.pdf) of this lab. **Also, include a text file (Report.txt) that includes your thoughts on using Git and GitHub for project version tracking and collaboration.** What are some of the obstacles that you encountered working on this assignment? What would be some advantages of using GitHub for your projects? Use your own words. This report should have at least 100 words.

Create a zip file “HW1\_Name\_Surname.zip” that contains *GitHub-Assignment-1.pdf*, *GitHub-Assignment-2.pdf*, and *Report.txt*. Upload *HW1\_Name\_Surname.zip* on Moodle until Sunday, 11 February 2024,

**Important Note:** Completing GitHub Assignment 1 and GitHub Assignment 2 during this week’s lab is a part of this Homework 1, and if you miss the lab, you will lose points from this homework. GitHub Assignment 1 and GitHub Assignment 2 are graded as part of this homework, so they are not counted as a part of the 15% Lab Assignments.