Prepared by: Eng. Sara Elghandour

**CpE 221L Introduction to Software Engineering Lab**

# Source Configuration Management

## Introduction

In software engineering, software configuration management (SCM or S/W CM)[1] is the task of tracking and controlling changes in the software, part of the larger crossdisciplinary field of configuration management.[2] SCM practices include revision control and the establishment of baselines. If something goes wrong, SCM can determine what was changed and who changed it. If a configuration is working well, SCM can determine how to replicate it across many hosts.

The acronym "SCM" is also expanded as source configuration management process and software change and configuration management. [3] However, "configuration" is generally understood to cover changes typically made by a system administrator.

### The Six Functional Areas of Software Configuration Management

The six core functional areas of CM are as follows:

1. Source code management
2. Build engineering
3. Environment configuration
4. Change control / Version Control
5. Release engineering
6. Deployment

Configuration management attempts to identify and track all relevant elements of the configuration of a system, so that all possible errors can be identified, and possible solutions to the problems found.

Version control is a special case of configuration management, where we are concerned with maintaining multiple versions of a software system.

### Importance of SCM

Software is written:

* As a combination of modules
* In one or more languages
* For one or more applications
* Using one or more libraries

Things that may (will) change over time:

* The modules required and used
* The languages employed
* The application requirements
* The libraries

Each change could cause the system to break.

### The Three Classic CM Problems

Any configuration management approach must address at least the following “classic” problems:

1. **The double maintenance problem**

Must prevent occurrence of multiple copies of the same file that must be independently updated.

1. **The shared data problem**

Must allow two or more developers to access the same file/data.

1. **The simultaneous update problem** Must prevent “clobbering” when two developers update the same file at the same time.

“clobbering”: only the second developer’s changes survive.

### Versions vs. Configurations

**Files exist in multiple versions**

* Sequences of updates over time
* Parallel variants

**Systems exist in multiple configurations**

* For different platforms
* For different customers
* For different functionality/pricing levels

**Foo.java**

**(1.1)**

**Foo.java**

**(1.2)**

**Foo.java**

**(1.3)**

**Foo.java**

**(1.4)**

**Foo.java**

**(1.5)**

**Foo.java**

**(1.2.1)**

**Foo.java**

**(1.2.2)**

**Bar.java**

**(1.1)**

**Bar.java**

**(1.2)**

**Bar.java**

**(1.3)**

**1.0.0**

**1.0.1**

**1.1.0**

### Version Control

**Version control systems support:**

* Multiple versions of a file
* Multiple paths of file revision
* Locking to prevent two people from modifying the same file at the same time
* Recovery of any prior version of a file
* Efficient storage via “deltas” (forward or backward)

|  |
| --- |
| **Foo.java**  **(1.1)** |

|  |
| --- |
| **Foo.java (1.2)** |

|  |
| --- |
| **Foo.java (1.5)** |

|  |
| --- |
| **Foo.java (1.3)** |

|  |
| --- |
| **Foo.java (1.4)** |

**Branch** **Merge**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | **Foo.java (1.2.1)** | |  | |  | | --- | | **Foo.java (1.2.2)** | |

What is Git?

By far, the most widely used modern version control system in the world today is Git. Git is a mature, actively maintained open source project originally developed in 2005 by Linus Torvalds, the famous creator of the Linux operating system kernel. A staggering number of software projects rely on Git for version control, including commercial projects as well as open source. Developers who have worked with Git are well represented in the pool of available software development talent and it works well on a wide range of operating systems and IDEs (Integrated Development Environments).

Note: If you do not have Egit installed on RSAD or Eclipse, you can install it from HelpMarketplace  search for “Egit”

### Git Terminology Repository

The master storage of all the files under version (or source) control, also known as the database.

### Branch

A branch represents an independent line of development. Branches serve as an abstraction for the edit/stage/commit process.

Merge

Join two or more development histories together

### Master

The default development branch. Whenever you create a git repository, a branch named "master" is created, and becomes the active branch.

Working copy

Your copy of the file that you actually make your changes in.

### Add

The git add command adds a change in the working directory to the staging area. It tells Git that you want to include updates to a particular file in the next commit. However, git add doesn't really affect the repository in any significant way—changes are not actually recorded until you run git commit.

### Commit

To save your working copy of the file into the repository. Committed snapshots can be thought of as “safe” versions of a project—Git will never change them unless you explicitly ask it to. Along with git add, this is one of the most important Git commands.

HEAD

The current –most recently committed– snapshot of a file.

### Fetch

The git fetch command imports commits from a remote repository into your local repository. The resulting commits are stored as remote branches instead of the normal local branches that we’ve been working with. This gives you a chance to review changes before integrating them into your copy of the project.

Push

Pushing is how you transfer commits from your local repository to a remote repository.

### Pull

Merging upstream changes into your local repository is a common task in Git-based collaboration workflows. We already know how to do this with git fetch followed by git merge, but git pull rolls this into a single command.

### Open-source web-based Git hosting services

There are many open-source web-based git repository hosting services like GitHub and BitBucket.

Example:

This class represents an integer array (whose size is greater or equal to one). It has a method, named OCCUR, that takes an integer key and returns the number of occurrences of the key in the array.

**public** **class** Array {

**private** **static** **int** [] *A*;

**public** Array() {

*A* = **new** **int** [10] ;

}

**public** **void** setArray(**int** [] v){

*A* = v;

}

**public** **int** OCCUR( **int** key){

**int** count=0;

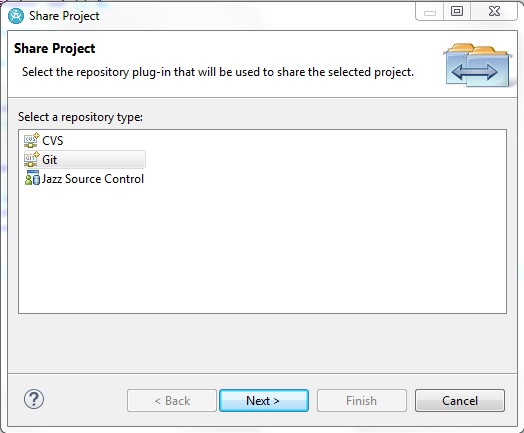
**for** (**int** i=0; i < *A*.length - 1 ; i++) **if** (*A*[i]==key) count++; **return** count;

}

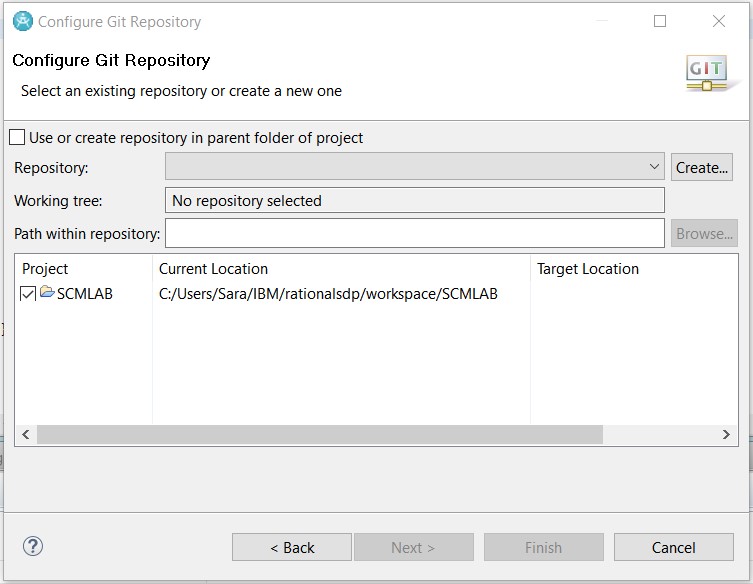
}

Steps:

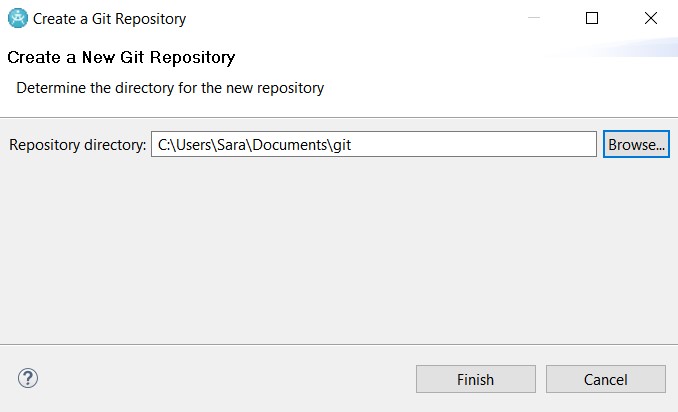
1. Run Eclipse
2. Create a new Java project named SCMLAB
3. Add a new class named Array, copy its code from above, and make sure that there no syntax errors in the code.
4. Right click on the project “SCMLAB” then choose Team  Share Project
5. Select “Git” as the Repository Type then click “Next”



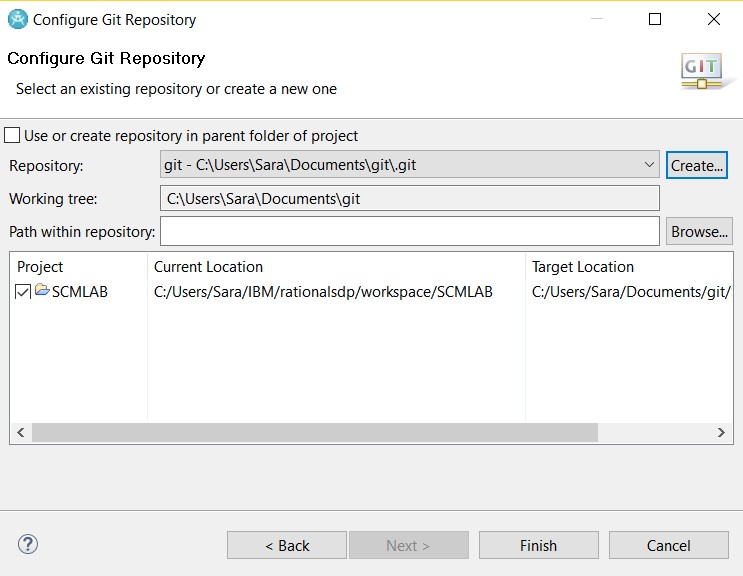
1. Click “Create”



1. Create a folder under “My Documents” and name it “git”
2. In the below window, browse for the created folder and select it, then click “Finish”

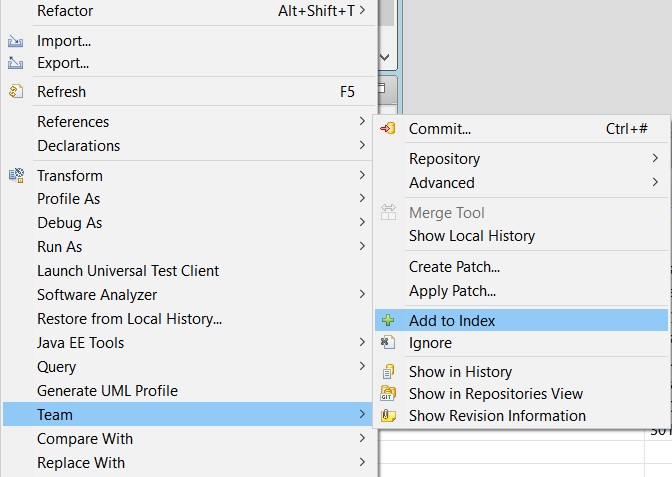


A local Repository has been created as shown, click “Finish”.

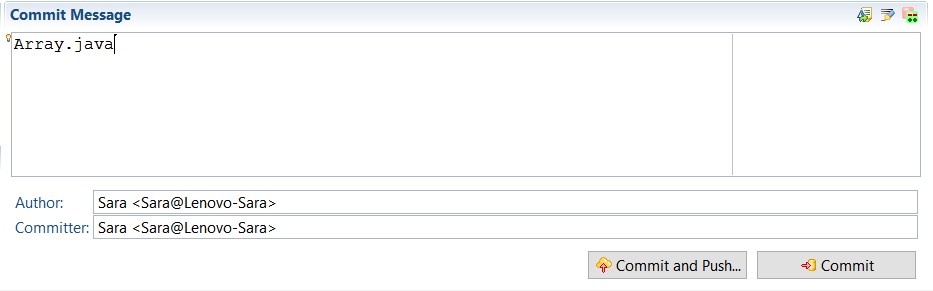


1. The newly created repository will be empty (Note the changed icons: the project node will have a repository icon, the child nodes will have an icon with a question mark, ignored files, e.g. the bin directory, won’t have any icons at all.)

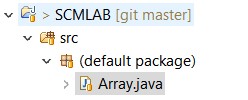
1. Before you can commit the files to your repository, you need to add them. Simply right click the shared project’s node and navigate to Team  Add to Index. After this operation, the question mark should change to a plus symbol. To set certain folders or files to be ignored by Git, e.g. the bin folder, right click them and select Team  Ignore. The ignored items will be stored in a file called gitignore, which you should add to the repository.



1. Now commit the project by right clicking the project node and selecting Team  Commit. In the Commit wizard, all files should be selected automatically. Enter a commit message (the first line should be headline-like, as it will appear in the history view) and hit the Commit button.



If the commit was successful, the plus symbols will have turned into repository icons.



1. Now you can start to modify files in your project. To save changes made in your workspace to your repository, you will have to commit them.

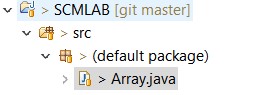
Change the following line

**for** (**int** i=0; i < *A*.length - 1 ;i++)

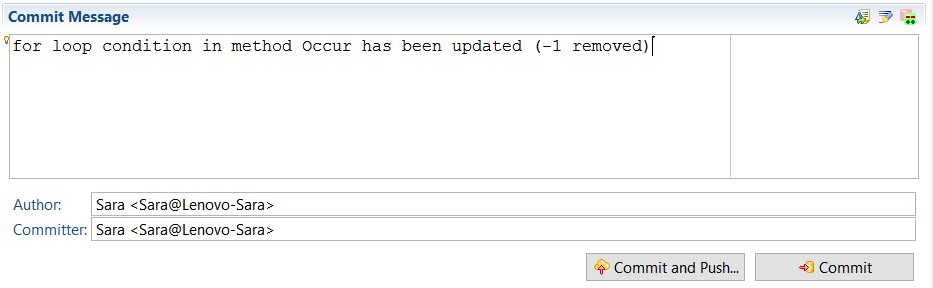
To

**for** (**int** i=0; i < *A*.length ; i++)

Then, click “save”. After changing files in your project, a “>” sign will appear right after the icon, telling you the status of these files is dirty. Any parent folder of this file will be marked as dirty as well.

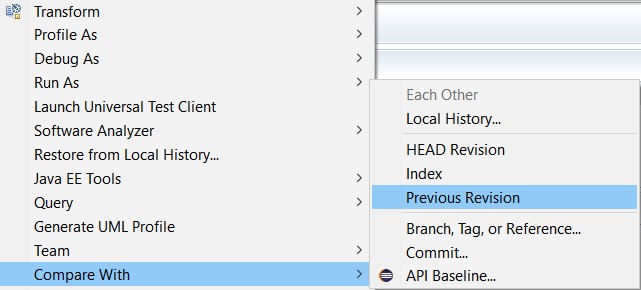


1. If you want to commit the changes to your repository, right click the project (or the files you want to commit) and select Team  Commit. Write a commit message briefly indicating the changes made then click “Commit”

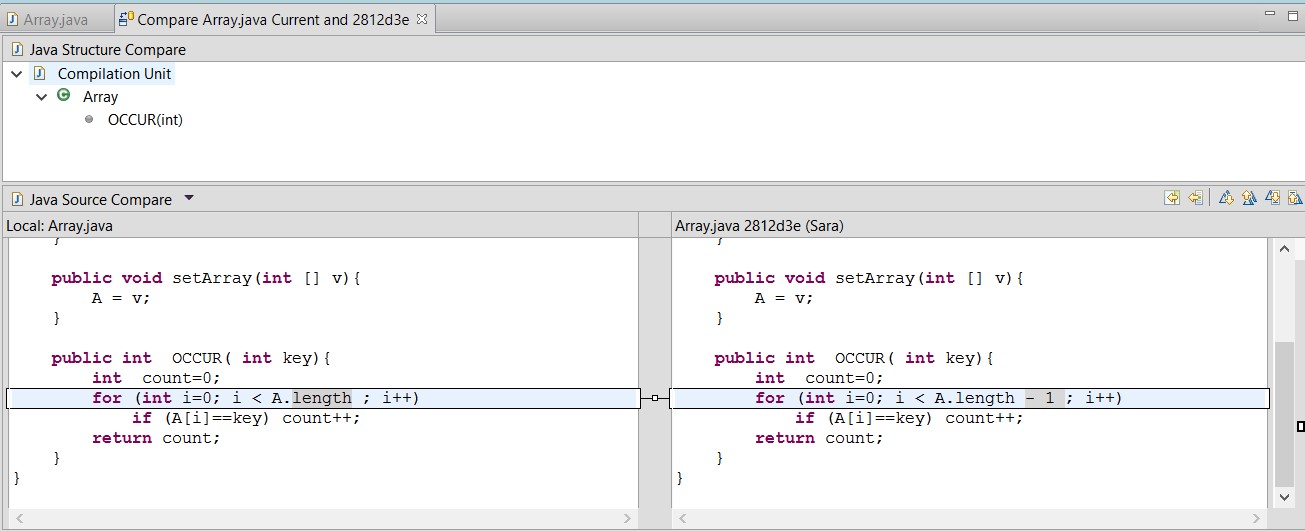


1. To compare the two versions of the Array.java file, right click Array.java file then select Compare with  Previous Revision

Or Compare with  Local History… , and select the required file.



A comparison window will be opened indicating the location of the changes between the previous version and the latest version

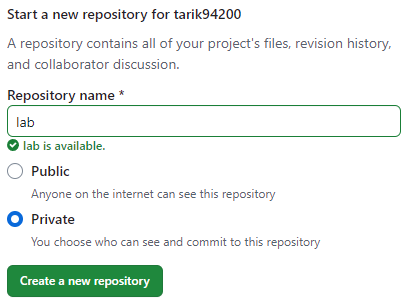


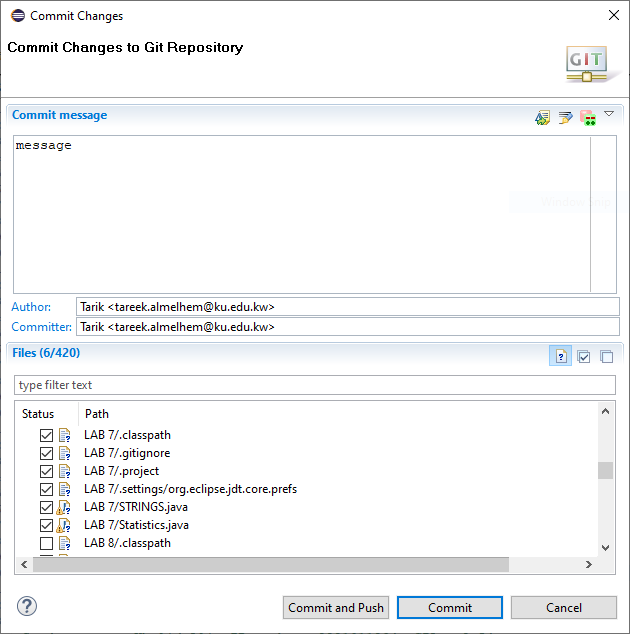
15. Assume you changed the for loop condition again to the shown below and saved the file (but did not commit it yet) then wanted to revert your changes to the last committed version (the HEAD Revision). **for** (**int** i=0; i < *A*.length + 5 ; i++)

Right click the .java file, then select Replace with  HEAD Revision.

Or select Replace with  Local History… , and select the required file.

### Using an online hosted repository

* 1. **GitHub** is a developer platform that allows developers to create, store, manage and share their code. It uses [Git](https://en.wikipedia.org/wiki/Git) software, which provides [distributed version control](https://en.wikipedia.org/wiki/Distributed_version_control) of [access control](https://en.wikipedia.org/wiki/Access_control), [bug tracking](https://en.wikipedia.org/wiki/Bug_tracking_system), [software feature](https://en.wikipedia.org/wiki/Software_feature) requests, [task management](https://en.wikipedia.org/wiki/Task_management), [continuous integration](https://en.wikipedia.org/wiki/Continuous_integration), and [wikis](https://en.wikipedia.org/wiki/Wiki) for every project. It is commonly used to host [open source](https://en.wikipedia.org/wiki/Open_source) software development projects.
  2. Create an account on <https://github.com>
  3. Create a new repository
  4. 

1. Right click Project Folder, Select Team , Commit. Then select all the files under the folder.
   1. 
2. Copy the github repository URI ( Right click the name  copy link).

A close up of a sign

Description automatically generated

1. Right click the ***project***  Team  Push branch…
2. A window will be opened, paste the URI and enter your github account username
3. Personal access tokens are an alternative to using passwords for authentication to GitHub
4. In the upper-right corner of any page on GitHub, click your profile photo, then click  **Settings**.
5. In the left sidebar, click **Developer settings**.

A screenshot of a computer

Description automatically generated

1. In the left sidebar, under **Personal access tokens**, click  **Tokens (classic)**,
2. Click **Generate new Token**  **Generate new Token(classic)**.

A screenshot of a chat

Description automatically generated

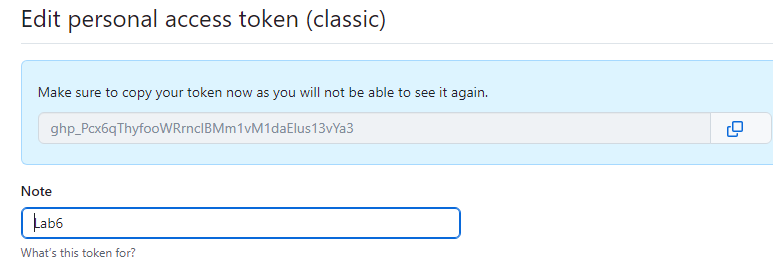
1. Under Select Scopes select all the options.

A screenshot of a computer

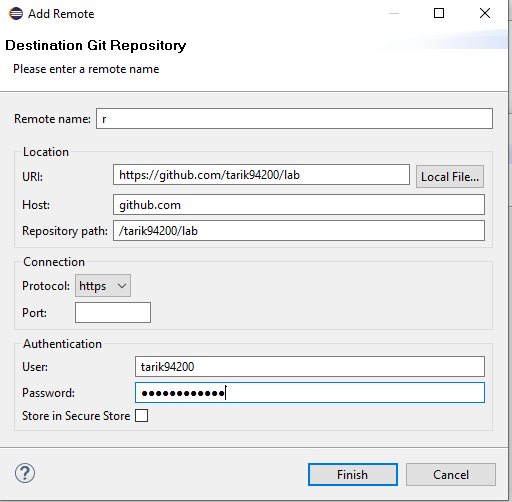
Description automatically generated

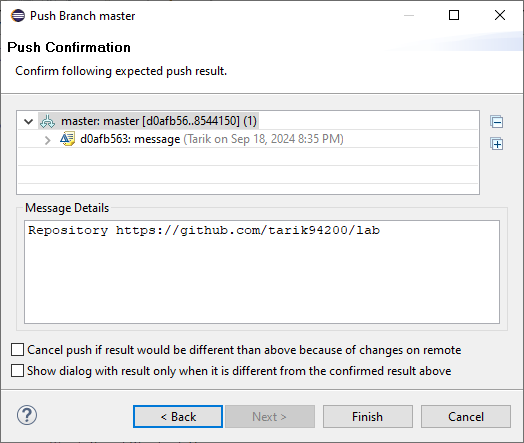
A screenshot of a computer

Description automatically generated



1. Copy token and paste it in password field, then click Finish.





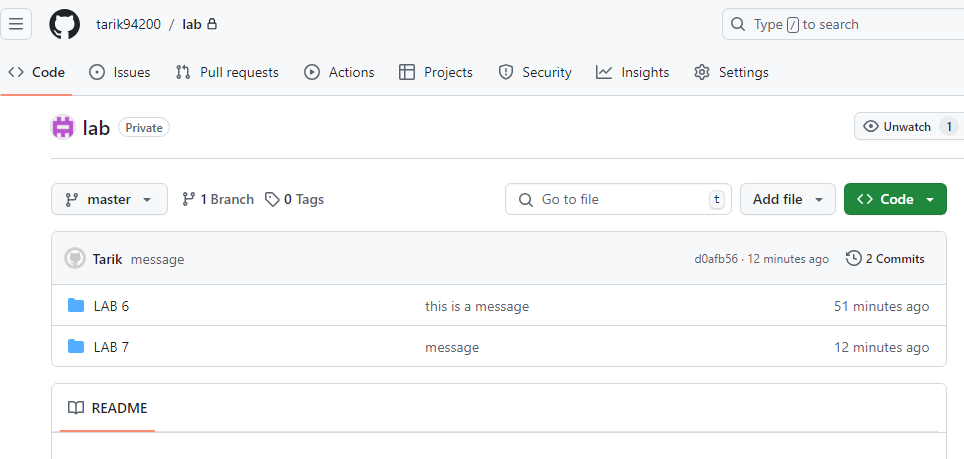
A screenshot of a computer

Description automatically generated

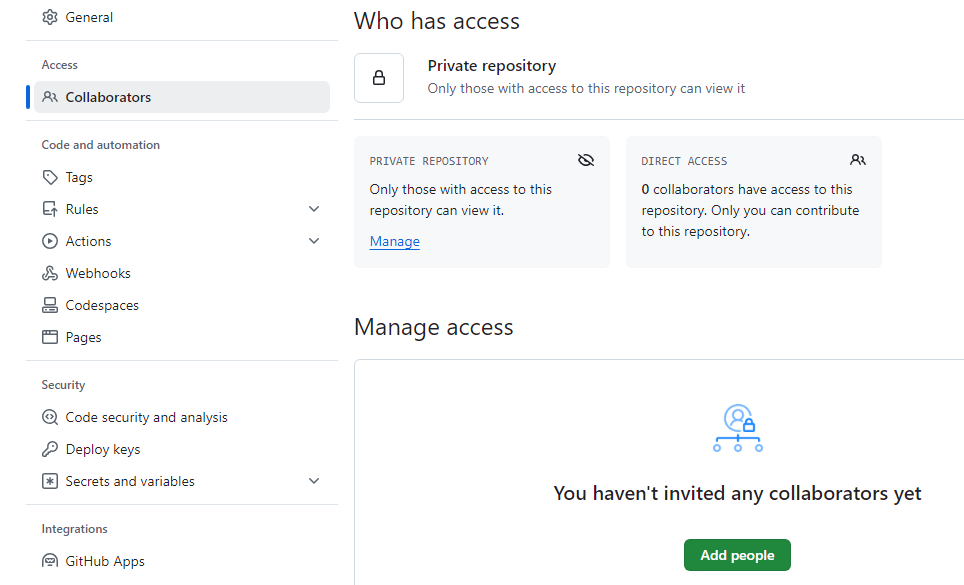
1. Open the home page of your github account, and see your folder on your github account.

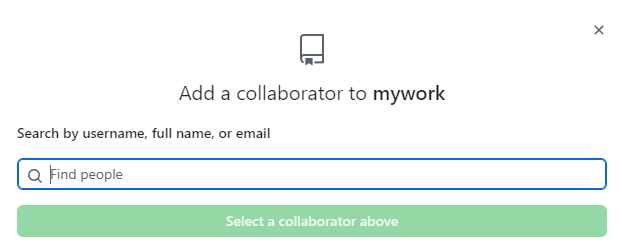
A screenshot of a computer program

Description automatically generated



1. To allow your teammates share this folder. Click Settings , Collaborators, Add people.





* 1. To add other files to your repository, Click Home.

A white rectangular object with a white background

Description automatically generated

* 1. Click your repository name.
  2. Click Add File 🡪 Upload Files.

A screenshot of a computer

Description automatically generated

* 1. Click Choose your files.

A screenshot of a computer

Description automatically generated

1. After uploading the file, click Commit Changes.

Lab work:

Given a class that represents a string of characters. It has a method named, *invert*, that checks every letter in the Text, and converts upper case letters into lower case, and vice versa. Characters in Text that represent punctuation marks are left unchanged. For example, if Text is “Good.”, then after calling *invert* it becomes “gOOD.”. A precondition on function *invert* is that Text must have at least one character.

**public** **class** mystring {

**private** String Text; **public** mystring() {

Text = **null**;

}

**public** **void** setString(String t){

Text = t;

}

**public** String invert( ){ **int** i , n = Text.length();

**char** [] Temp;

Temp = Text.toCharArray();

Text= "";

**for** ( i=0; i<n; i++) {

**if** (Temp[i] >= 'A' && Temp[i] <= 'Z') Temp[i]= (**char**)(Temp[i] - 'A' + 'a') ;

**if** (Temp[i] >= 'a' && Temp[i] <= 'z') Temp[i]= (**char**)(Temp[i] - 'a' + 'A') ;

Text += Temp[i];

}

**return** Text;

}

}

Individual Exercise:

1. Open Eclipse and Create a java project called SCMEX that includes the mystring class
2. Create a local repository, and add the mystring.java file to the index, then commit it
3. Change the following line

**if** (Temp[i] >= 'a' && Temp[i] <= 'z')

To

**else if** (Temp[i] >= 'a' && Temp[i] <= 'z')

then commit your changes.

1. Create an account on Bitbucket.org, then create a project SCMEX.
2. Use the Push command in Eclipse to share your local repository with the remote oneRight click on the Project  Team  Push branch master

1. Make another change but using Bitbucket source editor then commit it Change the following

**char** [] Temp; To **char** [] Temp= new char [n+1];

1. Use the pull command in Eclipse to view the changes in your local repository.
2. Right click on the Project  Team  Pull

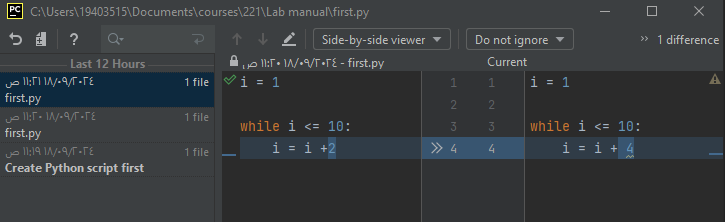
**SCM in Python**

Run Pycharm, write a program then save it.

To track changes in source code:

After each change click File, Save all

Click File , Local History , Show History. Pycharm displays a window with 3 lanes: left one for version names, middle one for old file version, and right one for latest version.



To restore a previous version: on left Click .py file , click Revert.

To share your file on GitHub:

- Create an account on GitHub : Click File , Settings..., Version Control , GitHub, Add account, Log in via GitHub.

- Browser will open with a button Authorize in GitHub.

- On PyCharm select the account , then click Apply, OK.

-File, New..., python File.

* VCS, Share Project on GitHub.
* Git , GitHub, Share Project on GitHub ,

## References

1. Atria (later [Rational Software,](https://en.wikipedia.org/wiki/Rational_Software) now a part of [IBM)](https://en.wikipedia.org/wiki/International_Business_Machines)
2. Roger S. Pressman (2009). Software Engineering: A Practitioner's Approach (7th International Ed.). New York: McGraw-Hill.
3. Gartner and Forrester Research
4. <https://en.wikipedia.org/wiki/Software_configuration_management>
5. Introduction to Version Control and Configuration Management by Philip Johnson ,

Collaborative Software Development Laboratory, Information and Computer Sciences, University of Hawaii

1. <https://www.atlassian.com/git/tutorials/what-is-git>
2. [http://techdocs.altium.com/display/ADOH/Version+Control+Terminology#](http://techdocs.altium.com/display/ADOH/Version+Control+Terminology)
3. <https://crunchify.com/how-to-configure-bitbucket-git-repository-in-you-eclipse/>
4. <http://eclipsesource.com/blogs/tutorials/egit-tutorial/>
5. <https://en.wikipedia.org/wiki/Bitbucket>
6. <https://git-scm.com/docs/>
7. <https://www.youtube.com/watch?v=8ZEssR8VTKo> (Python & gitHub)