

The University of Melbourne
School of Computing and Information Systems
COMP90041 Programming and Software Development

Lecturer: Dr Tilman Dingler, Dr Thuan Pham

Semester 2, 2020, Week 4

In this tutorial, you will be introduced to the basics of Git and on how to host your projects online on GitHub. This knowledge will be crucial for your assignments and final project so please follow this guide closely

What is Git?

Git is a **version control software**, which means it lets you create snapshots of your work. Think about a Word document, for example. Every time you save, it overwrites the previously saved, complete file. With version control, you only save the changes between two versions, thus making it easier for you to go back to a previous version. This is extremely useful on any software development project. When working as a team, Git “keeps track of every modification to the code in a special kind of database. If a mistake is made, developers can turn back the clock and compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members”¹. While all of this is great, teams now need to collaborate by sharing the code, and this is where GitHub shines.

GitHub is a **service** that lets developers **use version control online**, enabling us to have an online space (**repository**) that holds all our code. Every time you save your code you must create a new **commit** (adding the changes to the files). After committing you can **push** changes to your online repository and share it with your team. Even if you work alone it can be used as an online backup of your project.

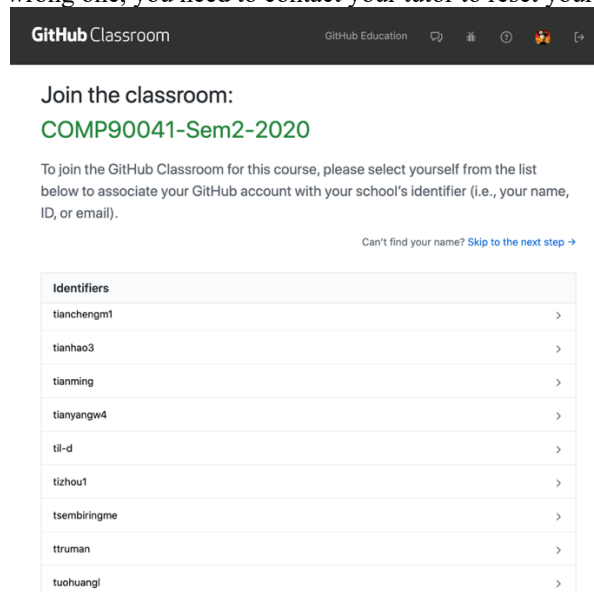
For this workshop, you will create a new repository, commit, and push code from GitHub. Excited? Then follow these steps:

- First, you must go and create an account on <https://github.com/> using your unimelb email. If you already have an account you can add your unimelb email by following the steps outlined here: <http://go.unimelb.edu.au/o37j>.
- Congratulations! You are the proud owner of a Git account. Once you are logged in, you can create your own repositories, but because this is a classroom where Tutors will check your code, you must create the repository using the following link (this enable the permission for tutors to access to the repository):

<https://classroom.github.com/a/F73ku8LY>

¹ <https://www.atlassian.com/git/tutorials/what-is-version-control>

- Find your unimelb login id and select it (make sure to be **careful to select the correct** one. If you accidentally select a wrong one, you need to contact your tutor to reset your repository link):



GitHub Classroom

Join the classroom:

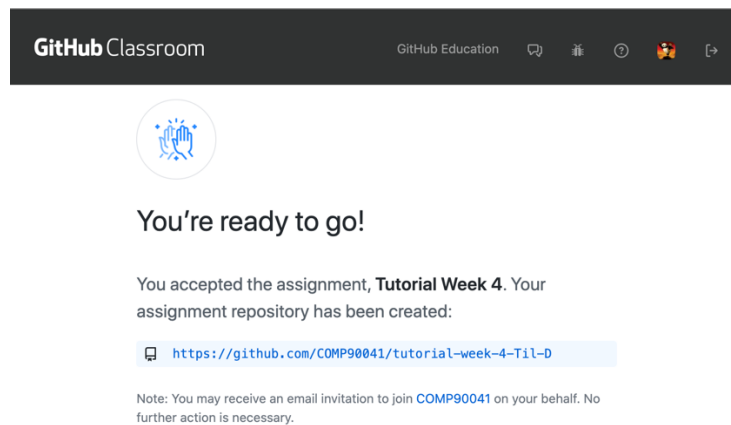
COMP90041-Sem2-2020

To join the GitHub Classroom for this course, please select yourself from the list below to associate your GitHub account with your school's identifier (i.e., your name, ID, or email).

Can't find your name? [Skip to the next step](#) →

Identifiers	
tianchengm1	>
tianhao3	>
tianming	>
tianyangw4	>
til-d	>
tizhou1	>
tsembiringme	>
ttruman	>
tuohuangl	>

- Accept the assignment and your repository is ready!



GitHub Classroom

You're ready to go!

You accepted the assignment, **Tutorial Week 4**. Your assignment repository has been created:

<https://github.com/COMP90041/tutorial-week-4-Til-D>

Note: You may receive an email invitation to join **COMP90041** on your behalf. No further action is necessary.

- Click on your repository URL and you will be in your own online space to store your code.
- Note that your repository is pre-filled with the following files:
 - .gitignore > lists files and file types that should not be committed
 - README.md > task description
 - Temperatures.java > skeleton code. Here is where you write your program.
 - oputput.txt > test output for you to locally test
 - temperatures.txt > test input for you to locally test

- Click on the `Temperatures.java` file

```

1  /*
2   The University of Melbourne
3   School of Computing and Information Systems
4   COMP90041 Programming and Software Development
5   Lecturer: Dr Tilman Dingler, Dr Thuan Pham
6   Semester 2, 2020, Week 4
7   Workshop Sample Solution
8   Copyright The University of Melbourne 2020
9  */
10
11  class Temperatures {
12
13      public static void main(String[] args){
14
15          System.out.println("This program isn't doing anything just yet.");
16
17      }
18  }

```

- Try to edit the output string and commit your changes:

Commit changes

Update Temperatures.java

Add an optional extended description...

tilman.dingler@unimelb.edu.au

Choose which email address to associate with this commit

☒ Commit directly to the `master` branch.
 ☐ Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

- Now your `Temperatures.java` is updated and has been committed.
- You will notice a red cross indicating that the server has run some tests on your code and found some error

Til-D Update Temperatures.java

All checks have failed

1 failing check

GitHub Classroom Workflow / Autograding (push)
 [Details](#)

1 contributor

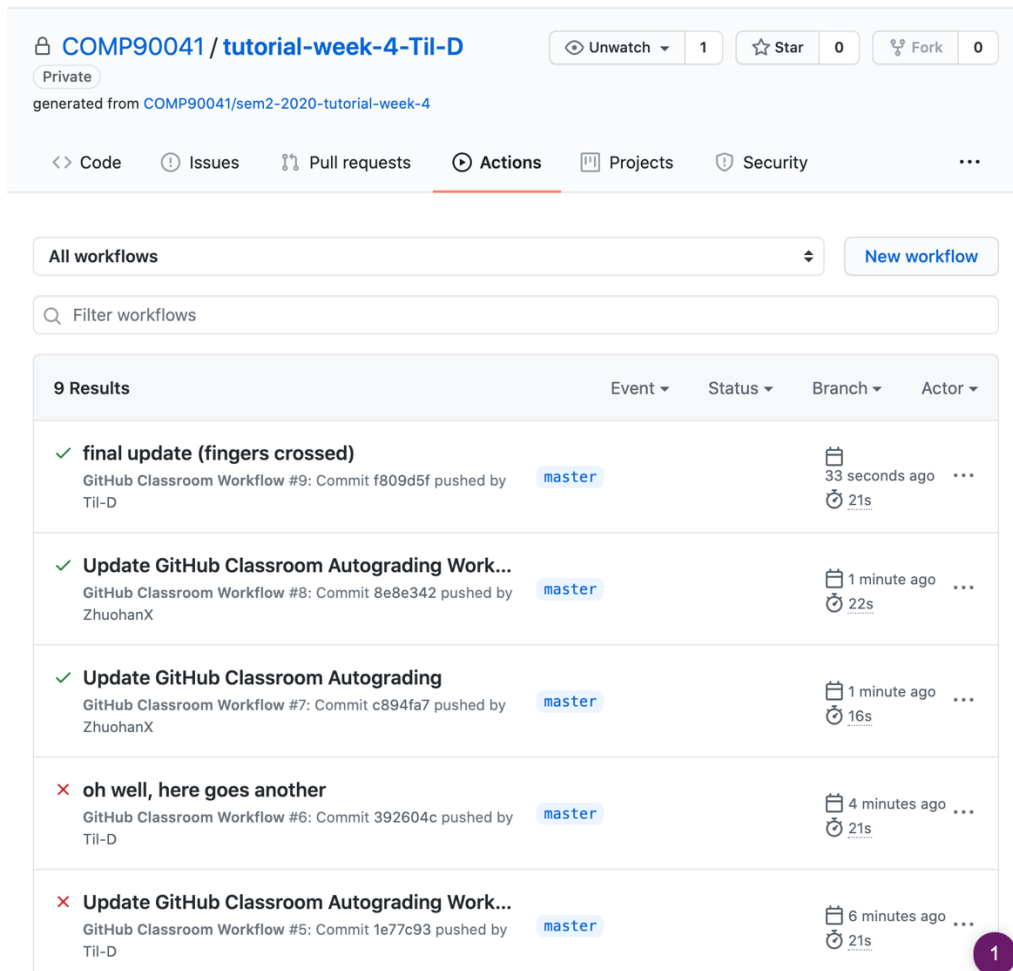
- Click on *Details* to explore the tests that were run and the resulting error messages.
- Keep editing (and committing) your `Temperatures.java` file until the tests come back positively:

Til-D Update Temperatures.java

Latest commit 88f4e64 2 minutes ago [History](#)

1 contributor

- You can check your history of commits under *Actions*:



COMP90041 / tutorial-week-4-Til-D

Private

generated from COMP90041/sem2-2020-tutorial-week-4

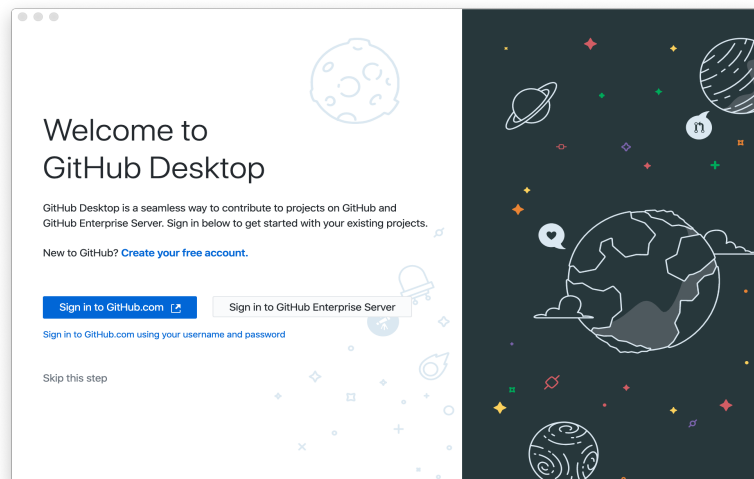
<> Code ⓘ Issues 🔗 Pull requests ▶ Actions 📁 Projects ⓘ Security ⋮

All workflows Filter workflows New workflow

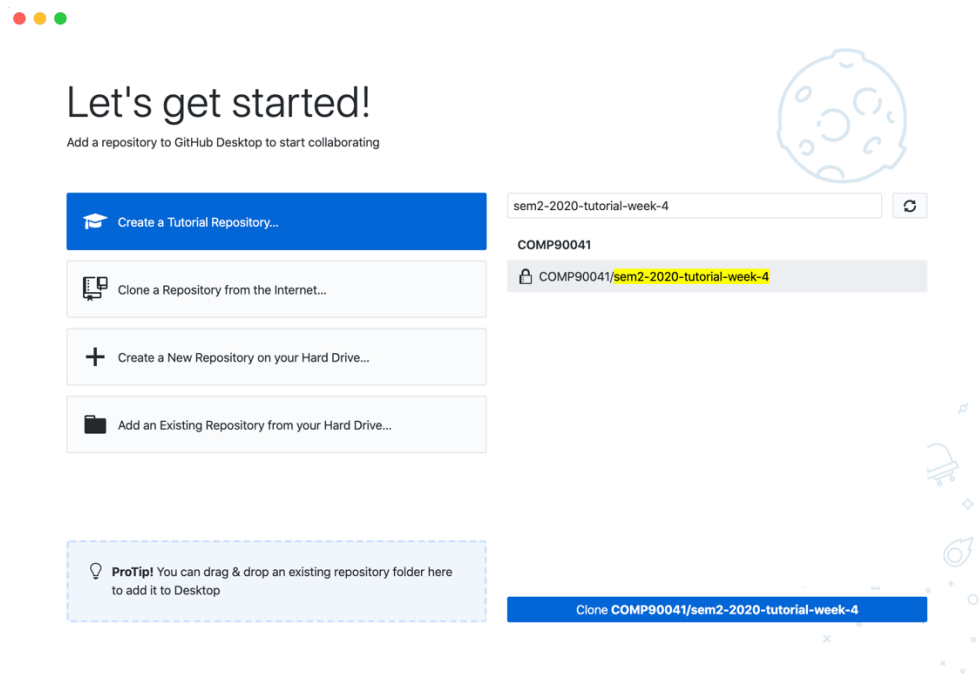
9 Results

Event	Status	Branch	Actor
✓ final update (fingers crossed)	GitHub Classroom Workflow #9: Commit f809d5f pushed by Til-D	master	33 seconds ago 21s
✓ Update GitHub Classroom Autograding Work...	GitHub Classroom Workflow #8: Commit 8e8e342 pushed by ZhuohanX	master	1 minute ago 22s
✓ Update GitHub Classroom Autograding	GitHub Classroom Workflow #7: Commit c894fa7 pushed by ZhuohanX	master	1 minute ago 16s
✗ oh well, here goes another	GitHub Classroom Workflow #6: Commit 392604c pushed by Til-D	master	4 minutes ago 21s
✗ Update GitHub Classroom Autograding Work...	GitHub Classroom Workflow #5: Commit 1e77c93 pushed by Til-D	master	6 minutes ago 21s

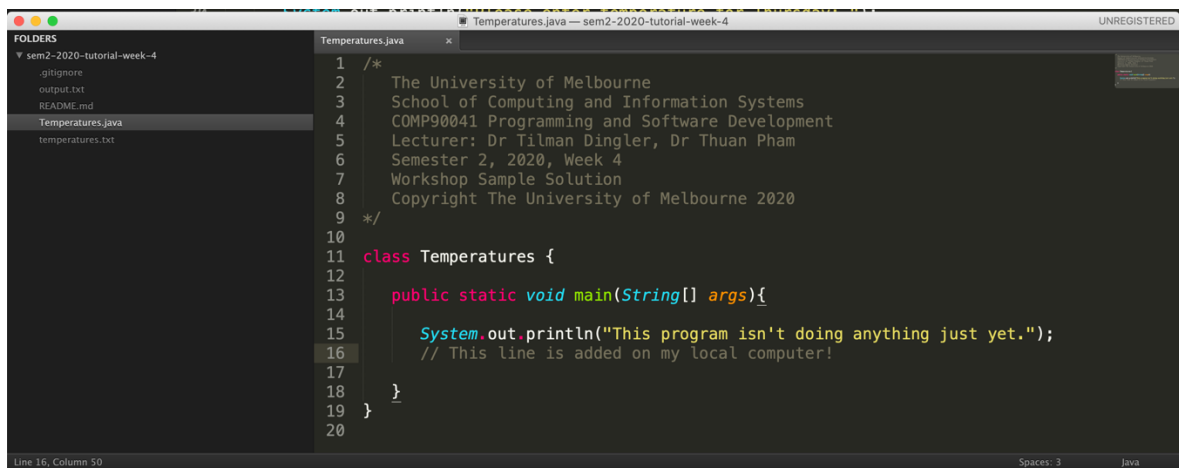
- Next, we are going to connect this repository to a local folder on your computer (This process is called *Clone a Repository*). For this, we are using *GitHub Desktop* (But you can any tool you prefer, including command-line tools).
- You will need to download and install the following software on your computer:
 - GitHub Desktop ([here](#))
- Sign in with your GitHub credentials.



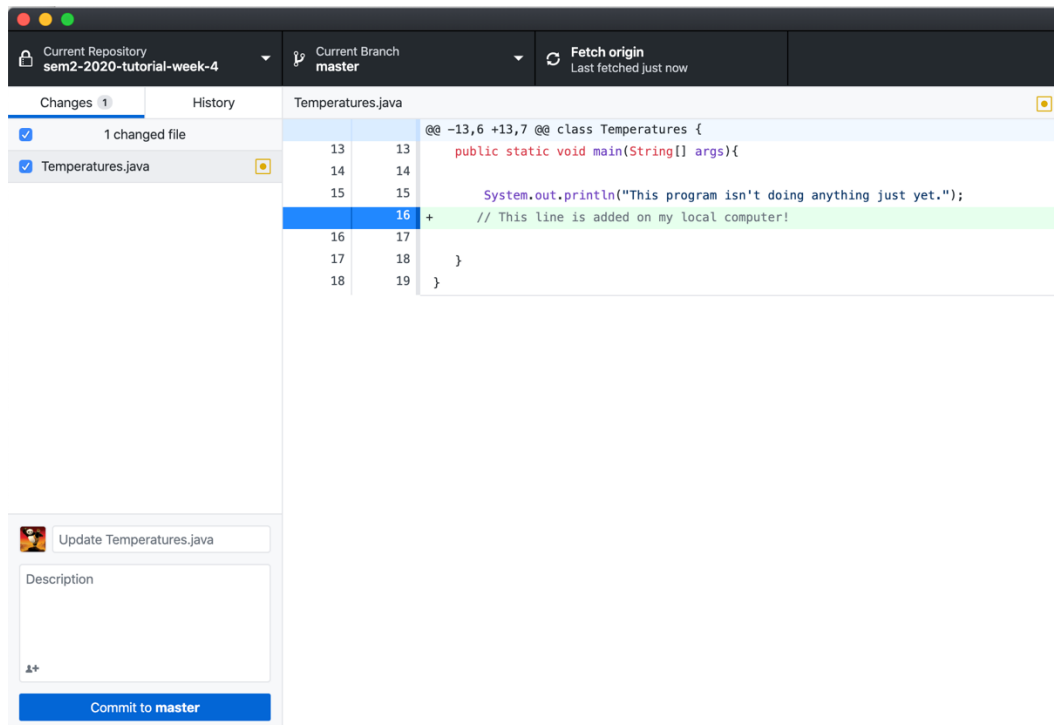
- Find your repository (should be something like: *sem2-2020-tutorial-week-4*). Select a **folder to sync your online files** and click on Clone.








- After cloning you can go to the local path in your computer and find the files from your repo.
- Open this folder in your favourite text editor and add a new comment:
// This line is added on my local computer!



- Save the file and go back to **GitHub Desktop**, you should be able to see the new changes in the Readme file highlighted in green.



- Do not forget to add a message and click on **Commit to master**. This records the changes in version control on your local machine. To make this change available online in your repository click on **Push origin**.
- You can now go to your repository in GitHub.com and check that your *Temperatures.java* file is updated with the extra line.
- Next to the commit ID (highlighted) you will find a red cross or green checkmark indicating the results of the automated tests.

	Til-D Update Temperatures.java	✓ 88f4e64 23 minutes ago	🕒 15 commits
	.github	Update GitHub Classroom Autograding	23 minutes ago
	.gitignore	Initial commit	1 hour ago
	README.md	Initial commit	1 hour ago
	Temperatures.java	Update Temperatures.java	23 minutes ago
	output.txt	Initial commit	1 hour ago
	temperatures.txt	Initial commit	1 hour ago

Congratulations! Welcome to the wonderful world of version control! Now let's code...

Exercise 1a: Histogram of temperatures

Write a program that reads in temperatures (in Celsius) for five days, that is, from Monday to Friday and plots a histogram showing the temperatures. The name of your class should be `Temperatures`. Given below is a sample run of the program.

```
Please enter temperature for Monday: 25
Please enter temperature for Tuesday: 33
Please enter temperature for Wednesday: 26
Please enter temperature for Thursday: 28
Please enter temperature for Friday: 20

Histogram of Temperatures
-----
Monday      | *****
Tuesday     | *****
Wednesday   | *****
Thursday    | *****
Friday      | *****
```

Exercise 1b: Testing locally: Input and Output Redirection

You will be given a sample test input file `test0.txt` and the corresponding sample output file `test0-output.txt`. When you run your program by the following command in a terminal (or Windows command line):

```
java Temperatures < test0.txt > my-output.txt
```

your program should produce a file name `my-output.txt`, which should be exactly the same as `test0-output.txt`. In this command, “`< test0.txt`” and “`> my-output.txt`” are called “input redirection” and “output redirection.” They use the content in `test0.txt` as the command line input, and print the program output into `my-output.txt`.

Exercise 1c: Submission

Once your program produces the correct output on your local machine, go ahead and submit your projects via GitHub.

You should always test your code locally on your machine before pushing it into your repository. Once you are confident that your program compiles correctly you can commit your changes and push your code to your repository. As soon as you submit your code, the GitHub server will start running automated on your code and compare your code’s output to what it is expecting. This process can take a few minutes. You can view the results online. Note that you can submit as many times as you like to test your code.

How you edit, compile and run your Java program is up to you. You are free to use any editor or development environment. However, you need to ensure that your program compiles and runs correctly with java’s version 1.8.0.

Note this exercise is for practice purpose only. It will NOT be marked.

Exercise 2: Traffic Infringements

The traffic section of a Police Department wishes to automate the writing of warnings, fines etc. to motorists who exceed the 60km/hr speed limit and whether doing it under influence of liquor or not. Your task is to implement the following warning and fines in the program based on the corresponding conditions:

<u>Condition</u>	<u>Message(s)</u>
> 60 and <65	Warning
>60 and <65 and drunk	Warning + Take a shower
65 to <= 70	\$5 fine for each km/hr over 60 km/hr
65 to <= 70 and drunk	\$7 fine for each km/hr over 60 km/hr + Take a shower
> 70	\$10 fine for each km/hr over 60 km/hr
> 70 and drunk	\$15 fine for each km/hr over 60 km/hr Spend the day/night in cell until you sober up

The program should ask the traffic officer to type in the km/hr speed of the offending driver. It should then ask whether driver is drunk or not. (The officer answers with a 'y' or 'n' and the appropriate message is then given.) The program should then display the appropriate message and where any fine is applicable, the program should compute and display the fine.

NOTE: You are not required to submit Exercise 2 via GitHub.

Sample Run 1

Please enter speed: 64
Is the driver drunk? ('Y' for drunk, 'N' otherwise): N

Warning

You have a fine of \$0.0

Sample Run 2

Please enter speed: 64
Is the driver drunk? ('Y' for drunk, 'N' otherwise): Y

Warning + Take a shower

You have a fine of \$0.0

Sample Run 3

Please enter speed: 85
Is the driver drunk? ('Y' for drunk, 'N' otherwise): Y

\$15.0 fine for each km/hr over 60 km/hr

Spend the day/night in cell until become sober.

You have a fine of \$375.0
