The University of Melbourne

School of Computing and Information Systems

# COMP90041 Programming and Software Development

Lecturers: Prof. Udaya Parampalli, Dr Thuan Pham

Semester 1, 2021, 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:

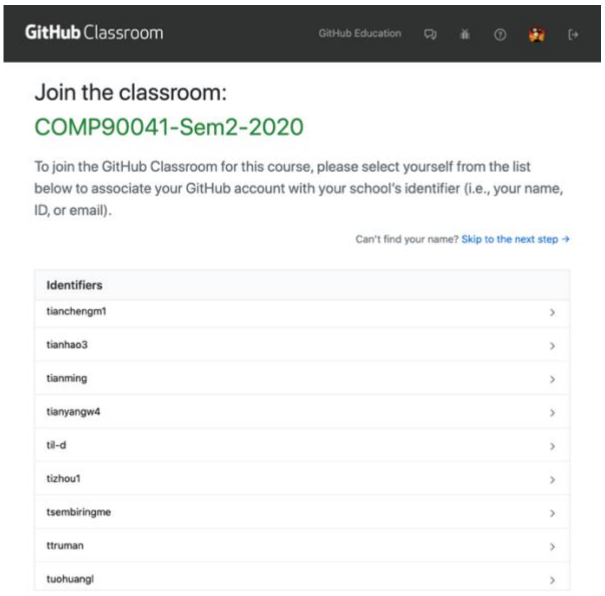
- First, you must go and create an account on https://github.com/ using your unimelb email. If you already have an account you can add your unimelb email by following the steps outlined here: http://go.unimelb.edu.au/o37j.

- 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/e-8-q50-

[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). *Note that the following figures were captured for COMP90041-Sem2-2020. In this semester, the classroom name is COMP90041-Sem1-2021.*



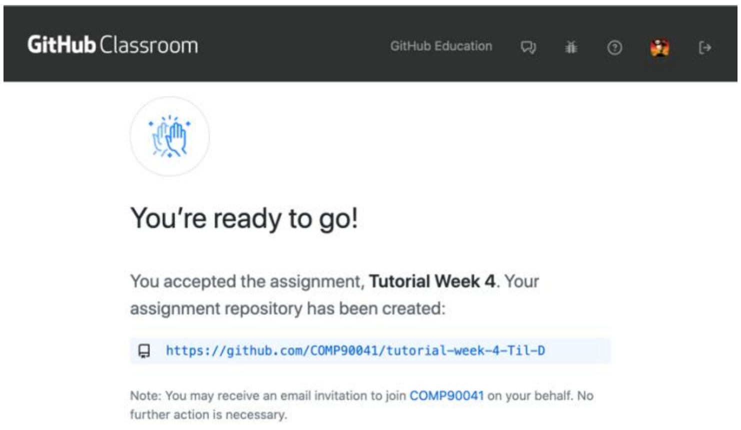
Join the classroom:
COMP90041-Sem2-2020

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!



You're ready to go!

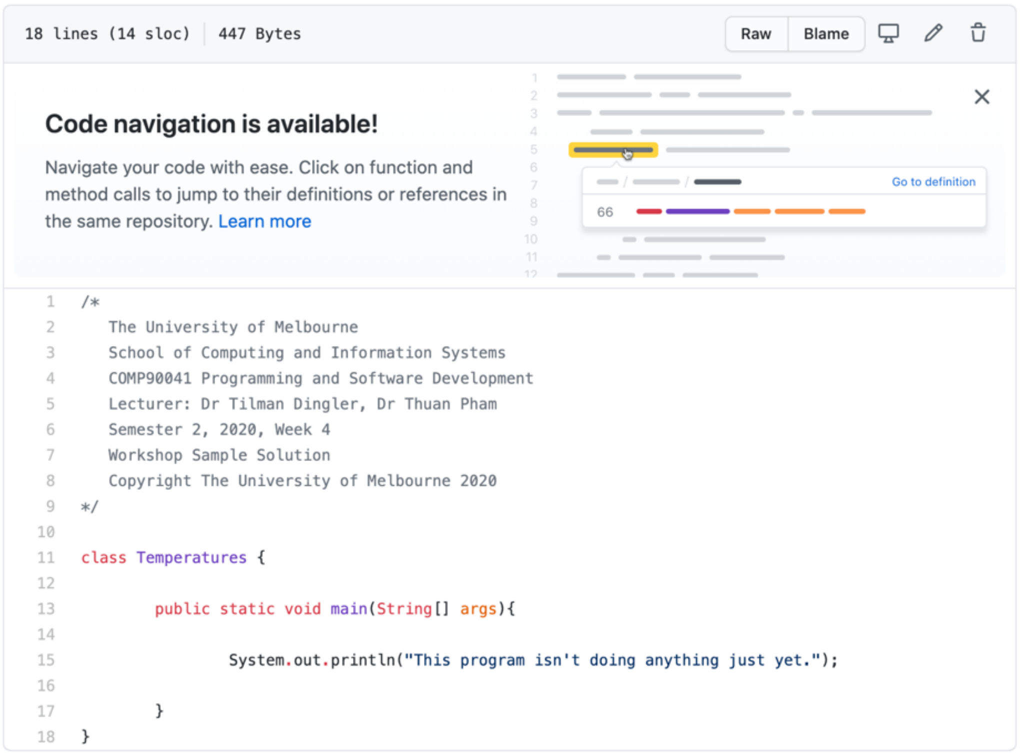
You accepted the assignment, **Tutorial Week 4**. Your assignment repository has been created:

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



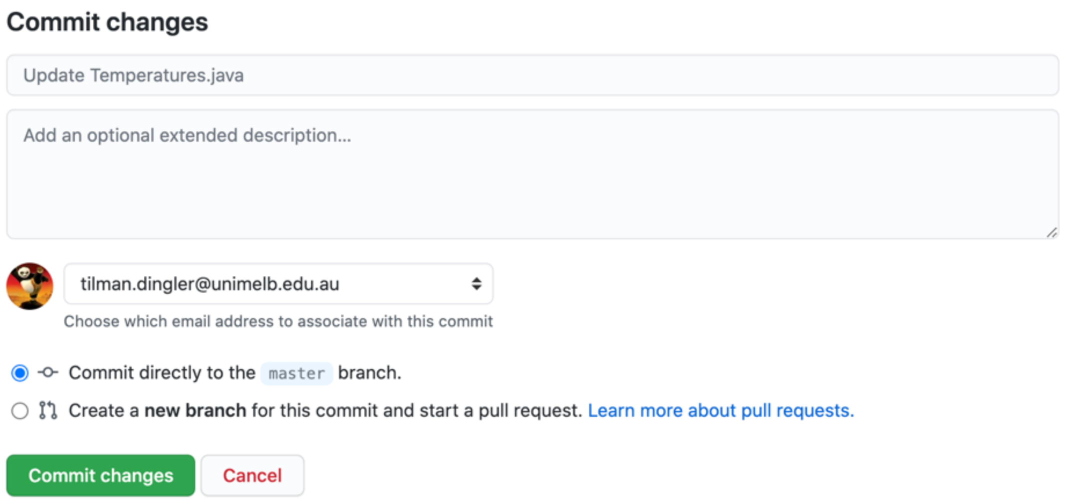
18 lines (14 sloc) | 447 Bytes

**Code navigation is available!**

Navigate your code with ease. Click on function and method calls to jump to their definitions or references in the same repository. Learn more

```java
/*
   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
   Workshop Sample Solution
   Copyright The University of Melbourne 2020
*/

class Temperatures {

        public static void main(String[] args){

                System.out.println("This program isn't doing anything just yet.");

        }
}
```

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



**Commit changes**

Update Temperatures.java

Add an optional extended description...

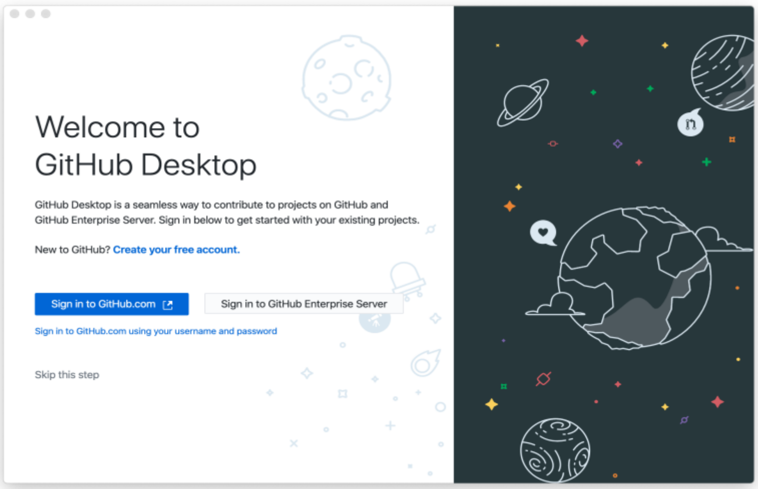
tilman.dingler@unimelb.edu.au

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 can check your history of commits.
- 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** (https://desktop.github.com/)
- Sign in with your GitHub credentials.

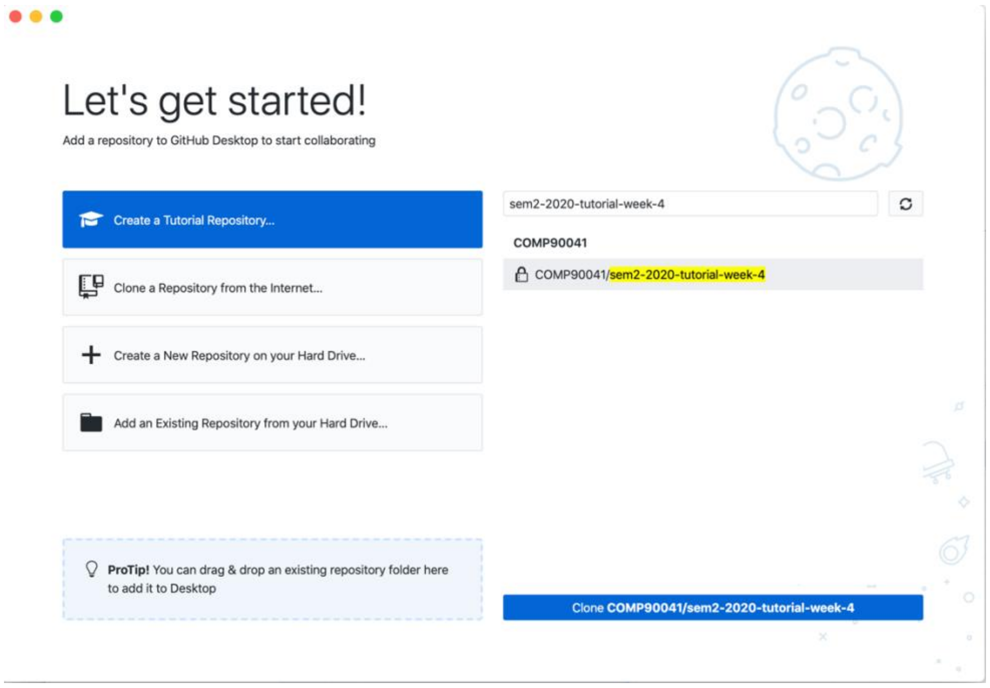

Welcome to GitHub Desktop

GitHub Desktop is a seamless way to contribute to projects on GitHub and GitHub Enterprise Server. Sign in below to get started with your existing projects.

New to GitHub? Create your free account.

Sign in to GitHub.com
Sign in to GitHub Enterprise Server
Sign in to GitHub.com using your username and password
Skip this step

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

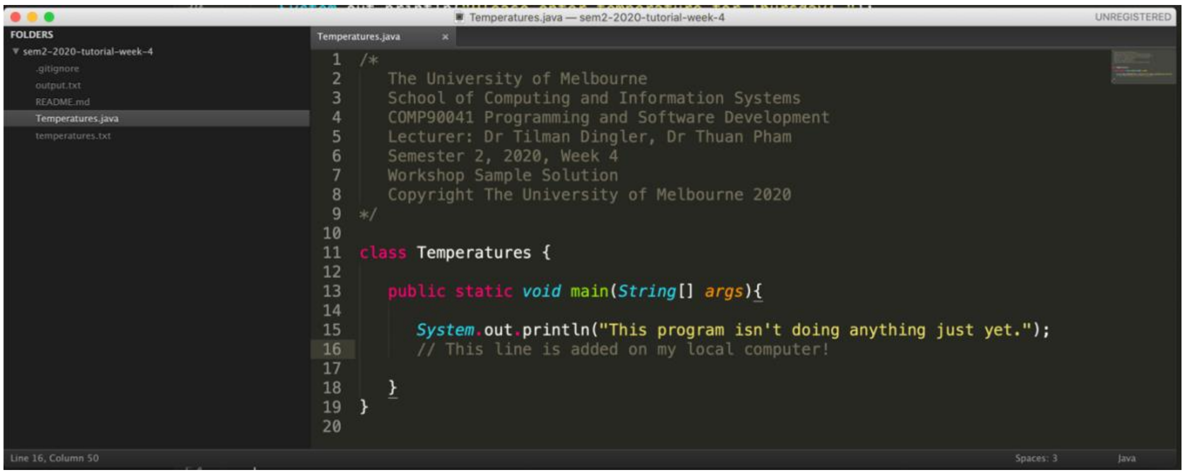

Let's get started!

Add a repository to GitHub Desktop to start collaborating

Create a Tutorial Repository...
Clone a Repository from the Internet...
Create a New Repository on your Hard Drive...
Add an Existing Repository from your Hard Drive...
sem2-2020-tutorial-week-4
COMP90041
COMP90041/sem2-2020-tutorial-week-4
Clone COMP90041/sem2-2020-tutorial-week-4

ProTip! You can drag & drop an existing repository folder here to add it to Desktop

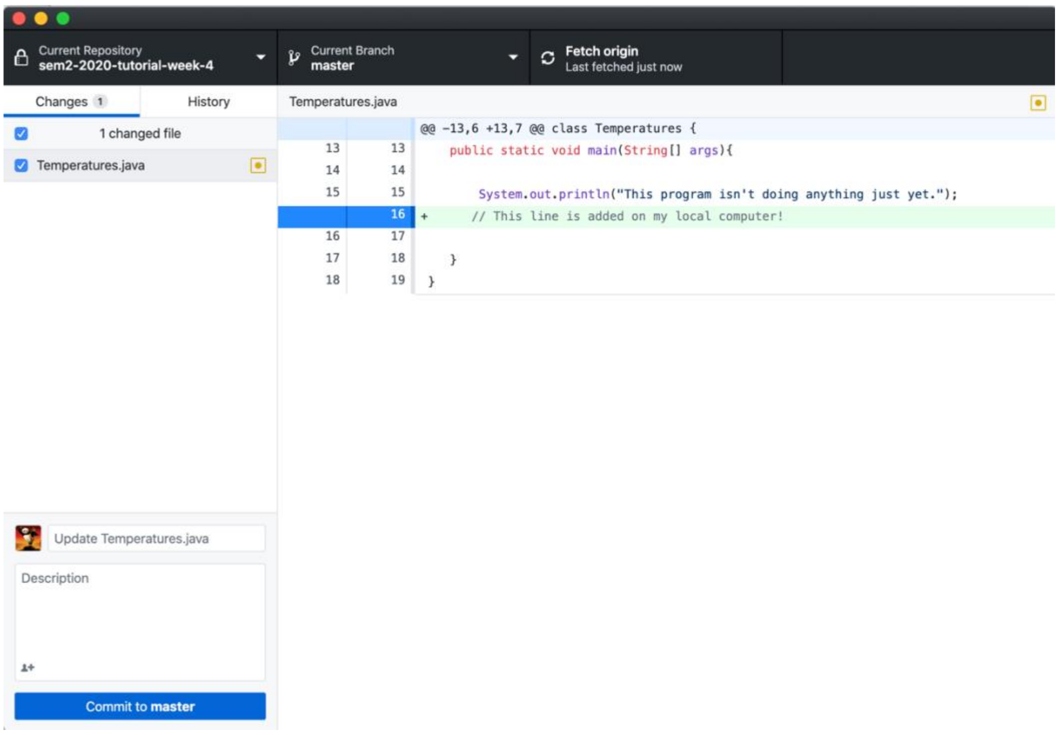
- After cloning you can go to the local path in your computer and find the files from your repo.
- Open this folder in your favourite text editor and add a new comment:
  // This line is added on my local computer!



```java
/*
   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
   Workshop Sample Solution
   Copyright The University of Melbourne 2020
*/

class Temperatures {

   public static void main(String[] args){

      System.out.println("This program isn't doing anything just yet.");
      // This line is added on my local computer!

   }
}
```

- You can also use your IDE, say Eclipse, to create a project from your local repository by choosing File→Open Projects from File System and following all the instructions. After that, you can modify the code like what you normally do for your previously created projects.

- 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.



```
@@ -13,6 +13,7 @@ class Temperatures {
13 13       public static void main(String[] args){
14 14
15 15          System.out.println("This program isn't doing anything just yet.");
   16 +        // This line is added on my local computer!
16 17
17 18       }
18 19    }
```

- 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.

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      | ********************
```

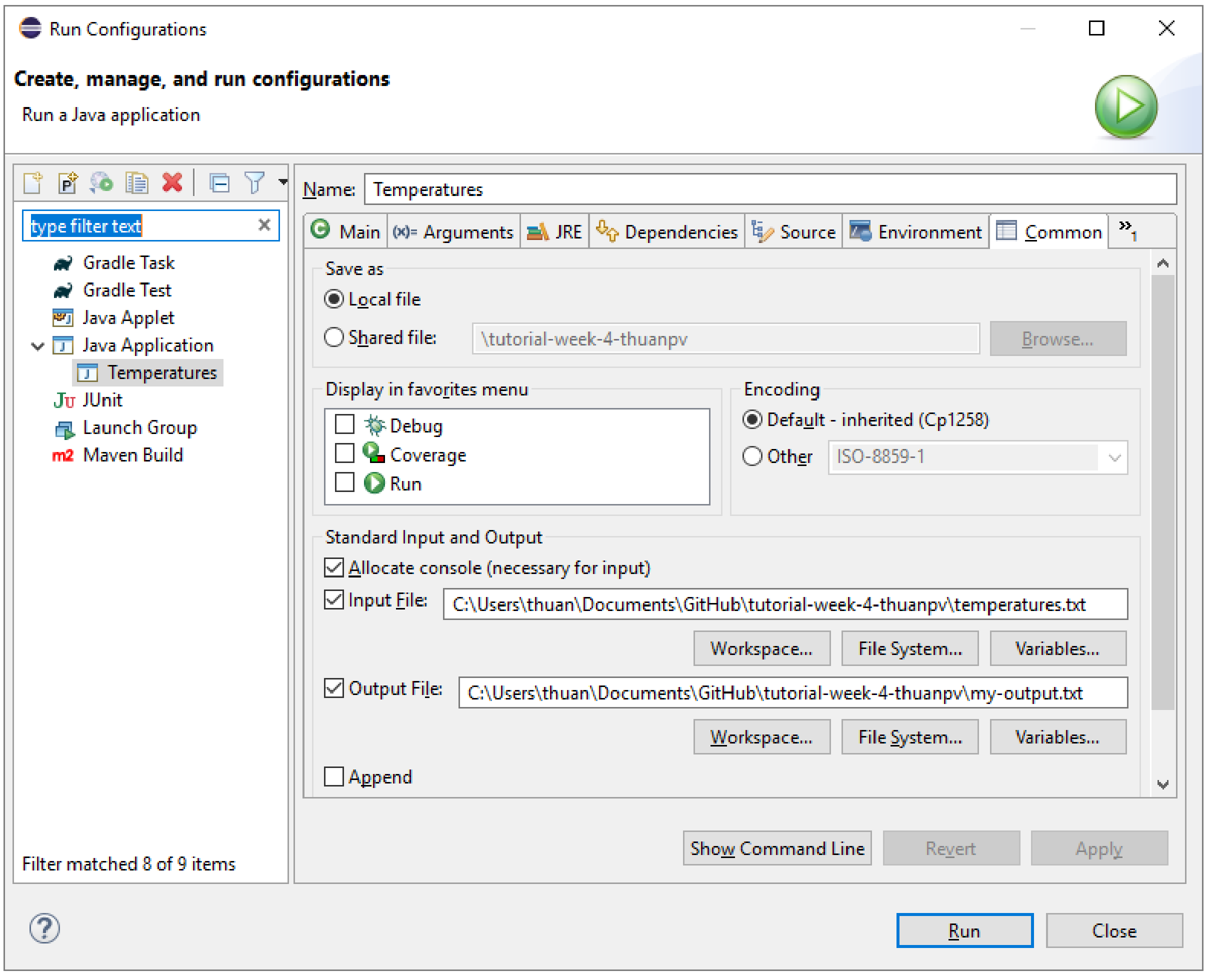
## Exercise 1b: Testing locally: Input and Output Redirection

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

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

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

To do input redirection and output redirection on your IDE, say Eclipse, you just need to update the configurations→Common. Specifically, you need to select the input file path (for input redirection) and output file path (for output redirection) as shown in the figure below.


Run Configurations

**Create, manage, and run configurations**

Run a Java application

Name: Temperatures

Save as: [✓ Local file] Shared file: \tutorial-week-4-thuanpv

Display in favorites menu: ☐ Debug ☐ Coverage ☐ Run

Encoding: [✓ Default - inherited (Cp1258)] Other ISO-8859-1

Standard Input and Output

☑ Allocate console (necessary for input)

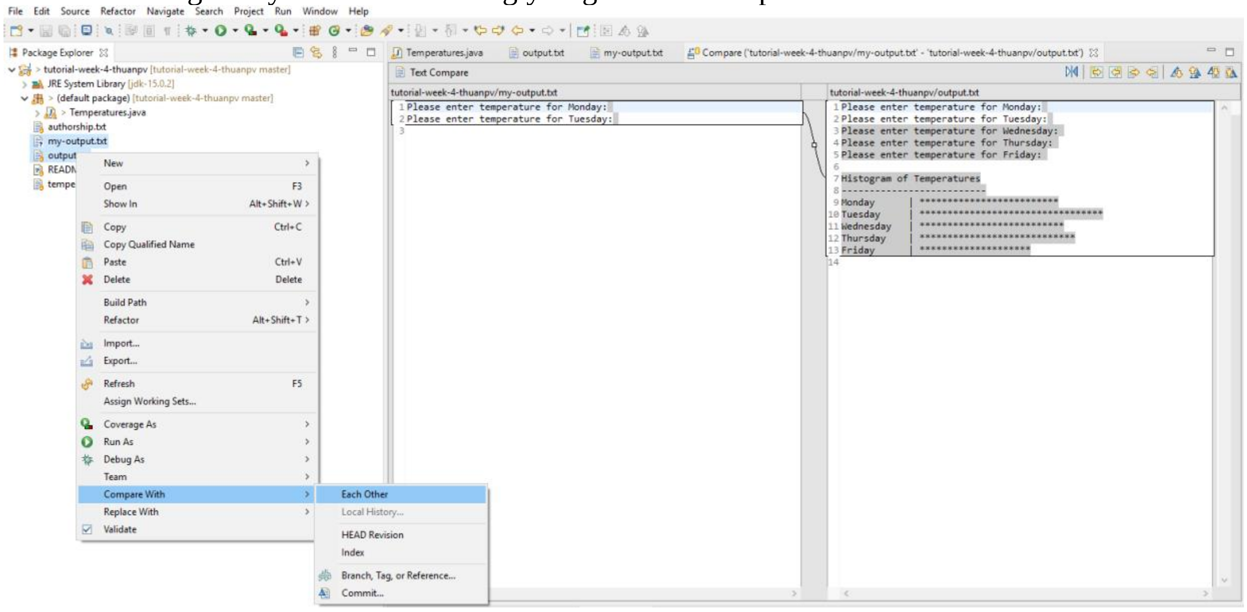
☑ Input File: C:\Users\thuan\Documents\GitHub\tutorial-week-4-thuanpv\temperatures.txt

☑ Output File: C:\Users\thuan\Documents\GitHub\tutorial-week-4-thuanpv\my-output.txt

☐ Append

Filter matched 8 of 9 items

To check your output (my-output.txt) and the provided one (output.txt) are exactly the same, you can use some diffing tools (e.g., "diff" command for Mac or Linux). Fortunately, Eclipse has a built-in feature for that. You just need to select two files you want to compare, and right click then select Compare With→Each Other. As shown in the figure below, Eclipse will open two panels side by side and highlight the differences between the two files. You can look at the differences and make changes to your code accordingly to get correct output.


Compare ('tutorial-week-4-thuanpv/my-output.txt' - 'tutorial-week-4-thuanpv/output.txt')

tutorial-week-4-thuanpv/my-output.txt
```
Please enter temperature for Monday:
Please enter temperature for Tuesday:
```

tutorial-week-4-thuanpv/output.txt
```
Please enter temperature for Monday:
Please enter temperature for Tuesday:
Please enter temperature for Wednesday:
Please enter temperature for Thursday:
Please enter temperature for Friday:

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

**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.

**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:

| *Condition* | *Message(s)* |
|---|---|
| > 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<br>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
****************************************************
```