COMP3111: Software Engineering

Extending the base code and teamwork at GitHub

Learning Outcomes

- Be able to extend the base code
- Be able to describe branching and pull requests of GitHub

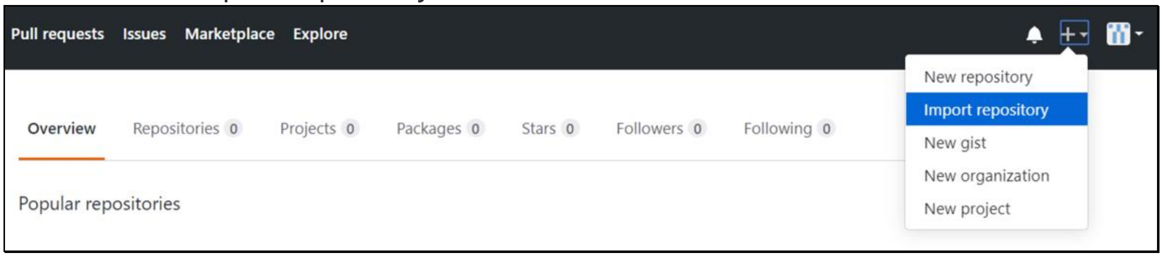
There are four guided exercises in this lab:
- Exercise 1: Setup a team repository at GitHub
- Exercise 2: Clone your team repository from GitHub (remote) to Eclipse IDE (local)
- Exercise 3: Git Branching
- Exercise 4: Pull Request at GitHub

Exercise 1: Setup a team repository at GitHub

In this exercise, you are going to setup a team repository that contains the project skeleton code. This repository is to be shared by the team. Therefore, only one member of your team is required to create the team repository. To facilitate the lab work, please grant all the teammates the ability to approve a Pull-Request.

1.1: Login with your GitHub user account (https://github.com/)

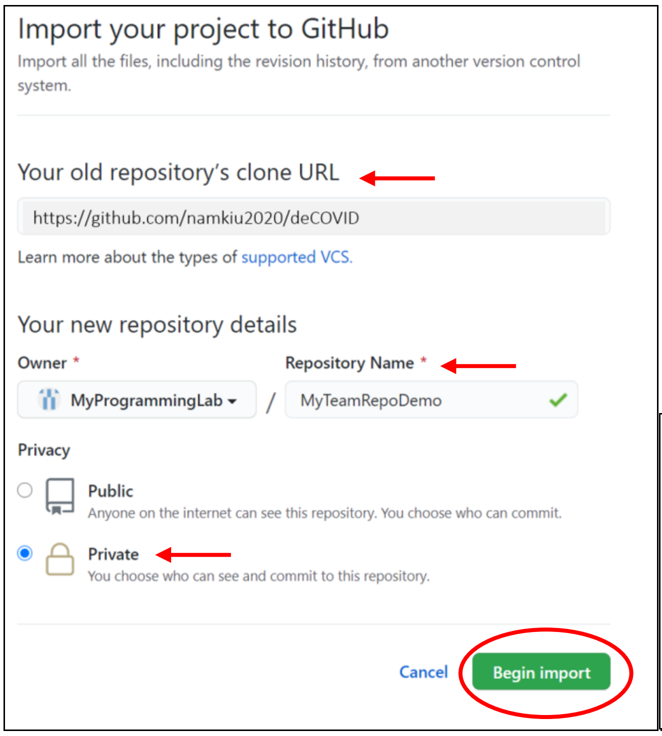
1.2: Choose "Import repository"



1.3: Follow the instructions and fill in the information:

* Your old repository's clone URL: https://github.com/namkiu2020/deCOVID
* Owner / Name: <your_github_id> / <chosen_name_for_your_team_repo>
* Privacy: Private

Click the "Begin import" button


Import your project to GitHub

Import all the files, including the revision history, from another version control system.

Your old repository's clone URL

https://github.com/namkiu2020/deCOVID

Learn more about the types of supported VCS.

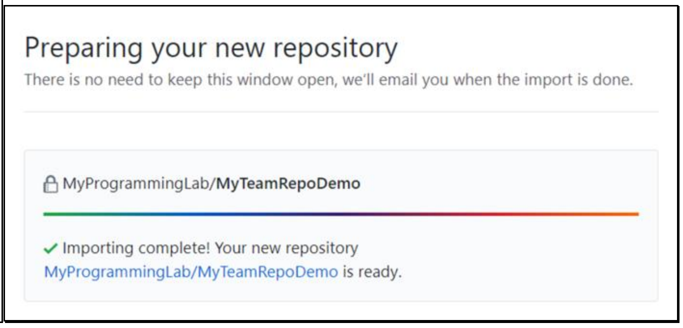
Your new repository details

Owner * MyProgrammingLab / Repository Name * MyTeamRepoDemo

Privacy

Public — Anyone on the internet can see this repository. You choose who can commit.

[✓ Private] — You choose who can see and commit to this repository.

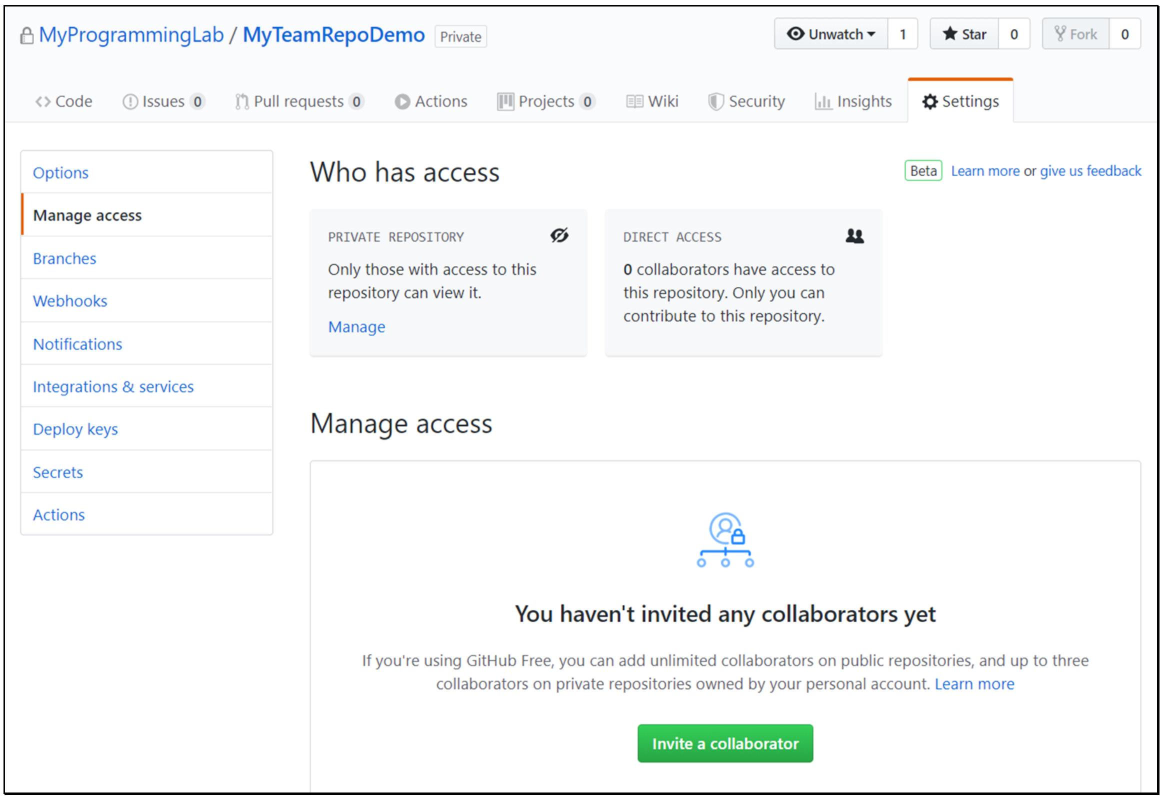

Preparing your new repository

There is no need to keep this window open, we'll email you when the import is done.

MyProgrammingLab/MyTeamRepoDemo

✓ Importing complete! Your new repository MyProgrammingLab/MyTeamRepoDemo is ready.

1.4: Add collaborators: On your team repo page, select "Setting > Collaborator" and invite all your teammates and TAs (GitHub IDs for all TAs can be found on the Teaching Team page at Canvas) as collaborators


MyProgrammingLab / MyTeamRepoDemo Private

Who has access

Beta Learn more or give us feedback

PRIVATE REPOSITORY — Only those with access to this repository can view it. Manage

DIRECT ACCESS — 0 collaborators have access to this repository. Only you can contribute to this repository.

Manage access

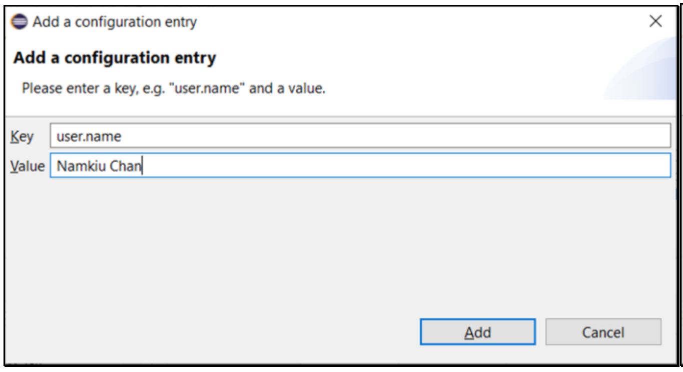
You haven't invited any collaborators yet

If you're using GitHub Free, you can add unlimited collaborators on public repositories, and up to three collaborators on private repositories owned by your personal account. Learn more

Exercise 2: Clone your Team Repo from GitHub (remote) to Eclipse IDE (local)

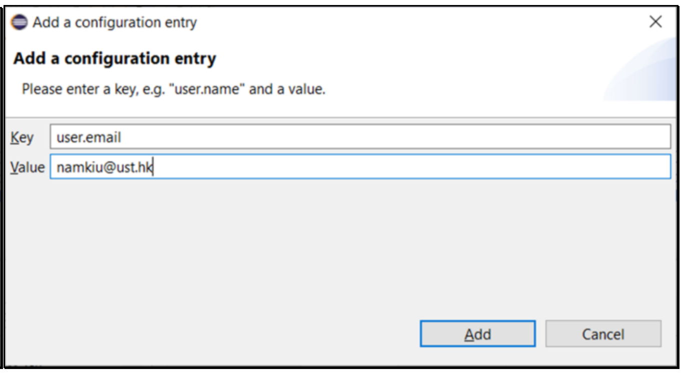
2.1: Configuration of Git in Eclipse IDE:

* From the top menu, Choose: Window > Preferences
* Select: Version Control (Team) > Git > Configuration > User Settings
* Click the "New Entry.." button; Type **user.name** as Key and your name as Value
* Click the "New Entry.." button; Type **user.email** as Key and your email address as Value
* Click the "Apply and Close" button to complete the process


Add a configuration entry

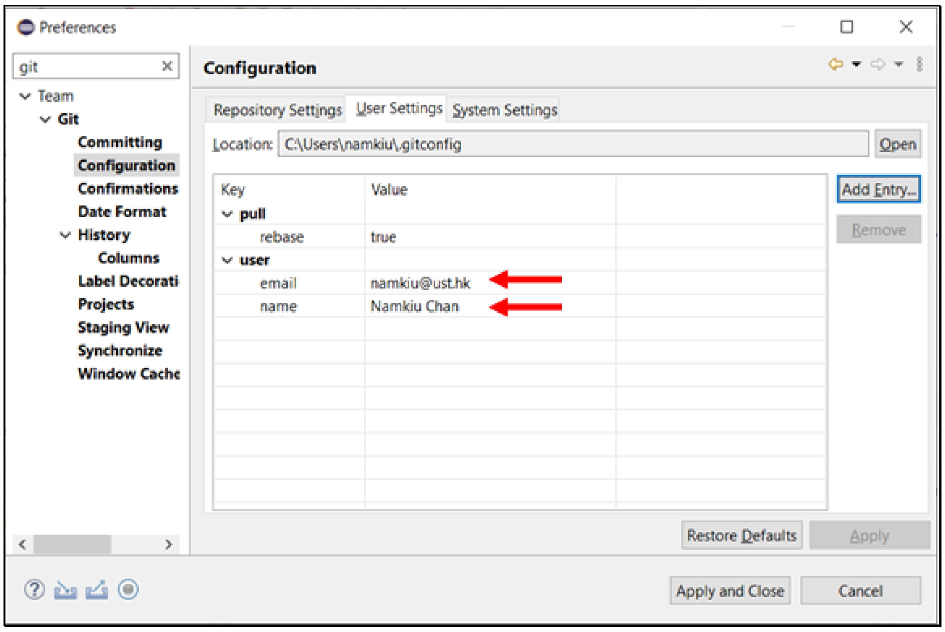
Please enter a key, e.g. "user.name" and a value.

Key: user.name
Value: Namkiu Chan


Add a configuration entry

Please enter a key, e.g. "user.name" and a value.

Key: user.email
Value: namkiu@ust.hk

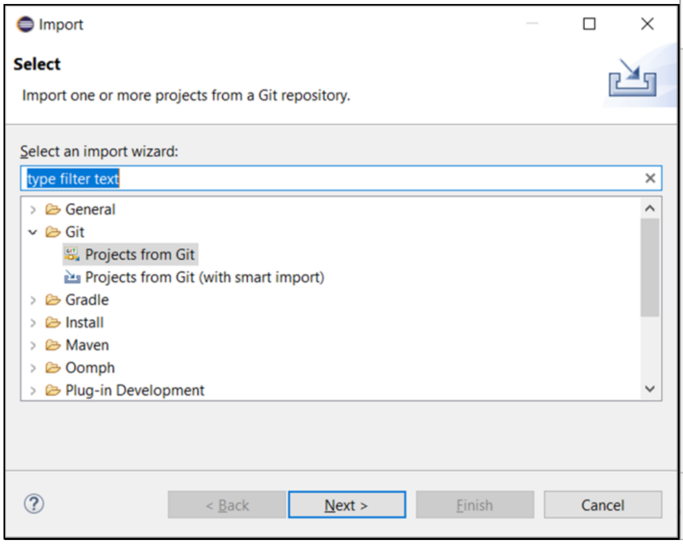


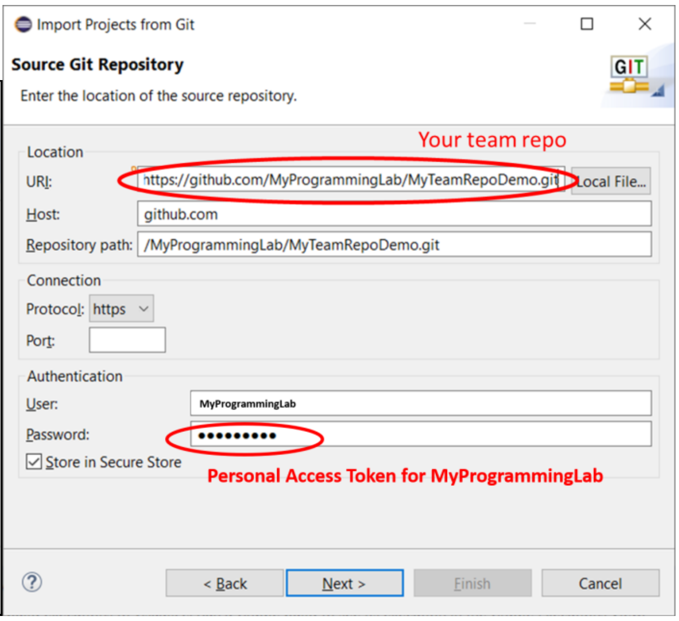
Location: C:\Users\namkiu\.gitconfig

Key	Value
pull	
rebase	true
user	
email	namkiu@ust.hk
name	Namkiu Chan

2.2: Cloning Repo from GitHub (remote) to Eclipse (local)

* From the top menu, Choose: File > Import..
* Select: Git > Projects from Git; Click the Next button
* Select: Clone URI; Click the Next button
* For URI, type in the URL of your team repo
* For User, type in the GitHub ID hosting the team repo
* For Password, type in the corresponding Personal Access Token (ref: Lab 1 > Step 4.4)
* Click the Next buttons a few times (accepting all default settings)
* Click the Finish button to complete the process



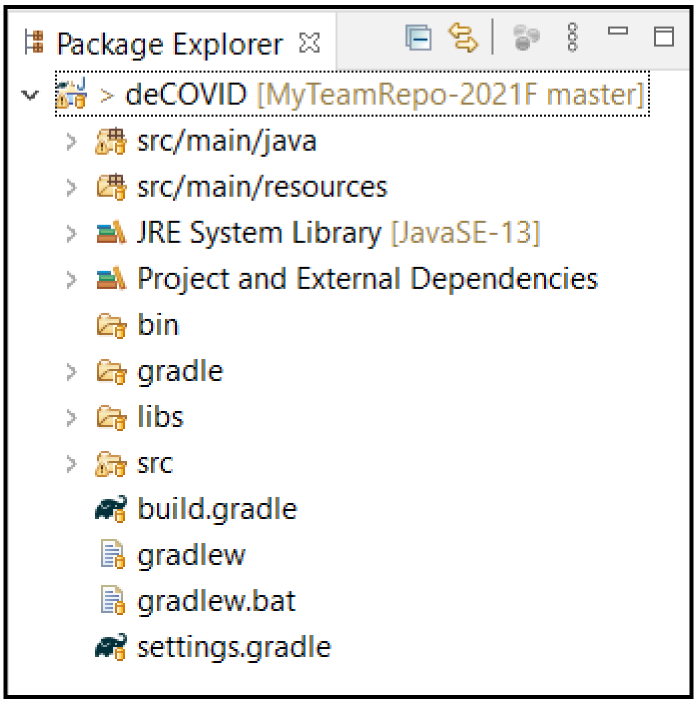


Location
URI: https://github.com/MyProgrammingLab/MyTeamRepoDemo.git
Host: github.com
Repository path: /MyProgrammingLab/MyTeamRepoDemo.git

Connection
Protocol: https
Port:

Authentication
User: MyProgrammingLab
Password: ●●●●●●●●●
☑ Store in Secure Store

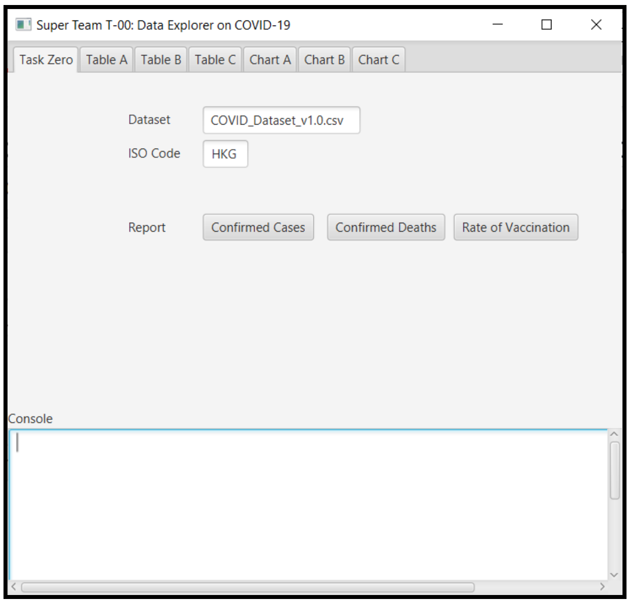
2.3: After importing your team repo, the Eclipse project explorer should look like the following (doesn’t matter whether it shows JavaSE-13 or JavaSE-15 in JRE System Library):

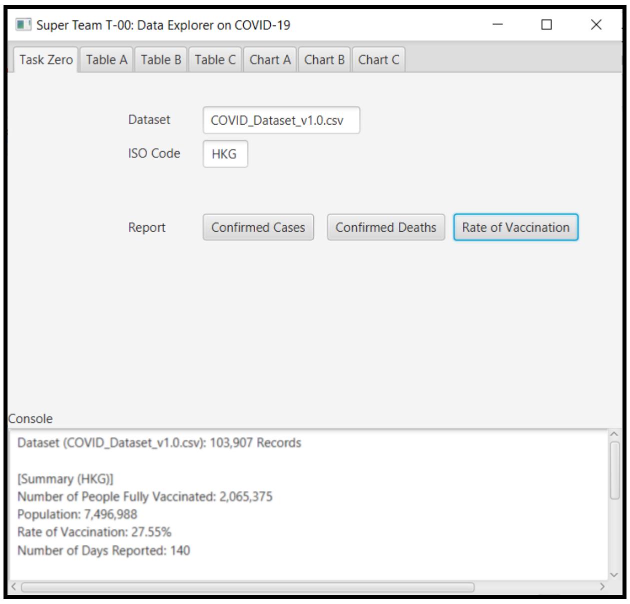


```
> deCOVID [MyTeamRepo-2021F master]
    src/main/java
    src/main/resources
    JRE System Library [JavaSE-13]
    Project and External Dependencies
    bin
    gradle
    libs
    src
    build.gradle
    gradlew
    gradlew.bat
    settings.gradle
```

2.4: Right click the file **build.grade**; Select “Gradle > Refresh Gradle Project”

2.5: Use Gradle Run to compile and execute. You shall see the sample application running.





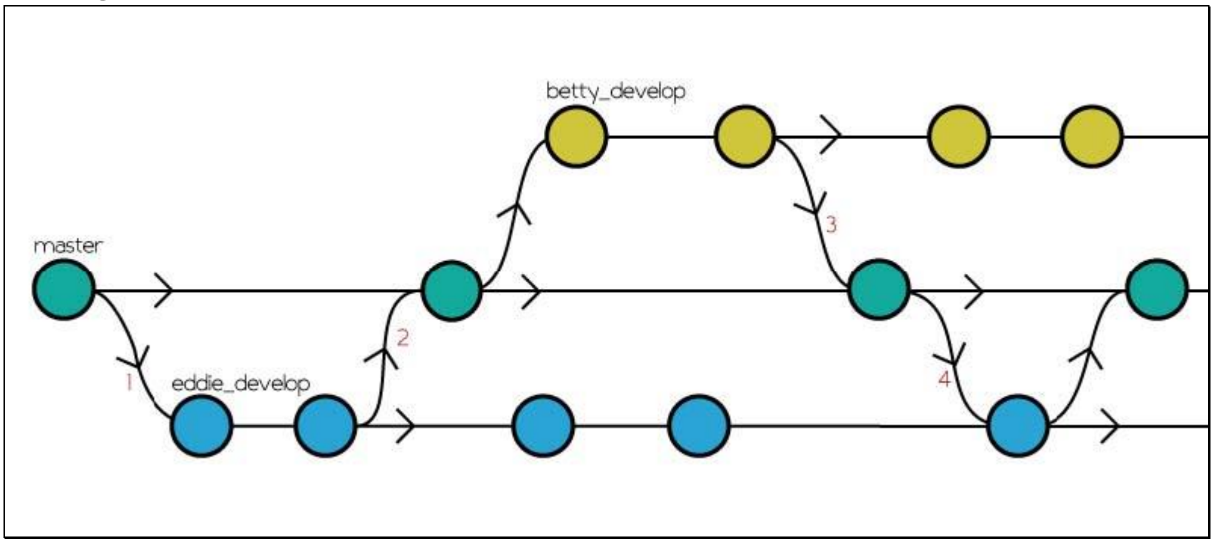
```
Dataset (COVID_Dataset_v1.0.csv): 103,907 Records

[Summary (HKG)]
Number of People Fully Vaccinated: 2,065,375
Population: 7,496,988
Rate of Vaccination: 27.55%
Number of Days Reported: 140
```

<u>**Exercise 3**</u>: Git Branching

In this exercise you are required to create a local branch and update **MyApplication.java**. Review your changes and push it to GitHub as a new branch of development.

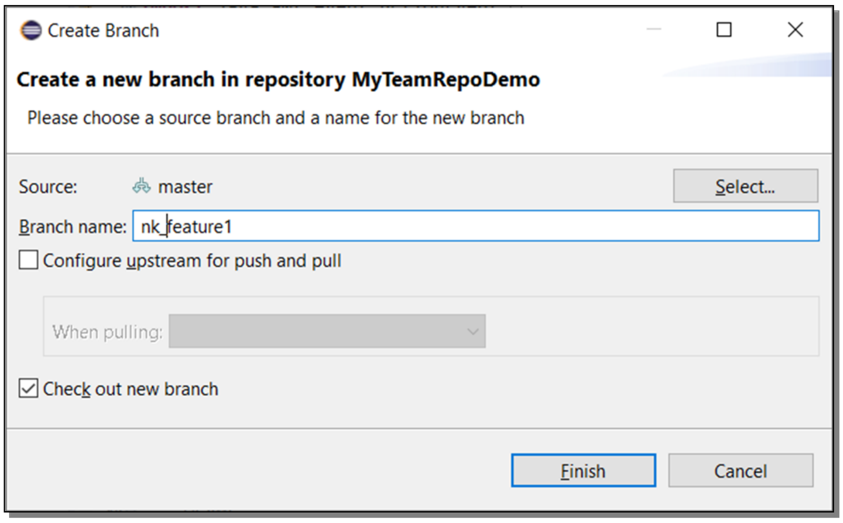
There is always a default branch (called **master** branch). In practice, the master branch should be a stable release of the project. In most projects (your project included), it is not recommended to pushing new code to the master branch directly. Changes to the master branch should only be made through explicit endorsement of pull requests.

Typically, a software developer should first create a unique branch, split off from the master branch (or other appropriate development branches). After that, the developer should focus on developing the specific features, and must test the implemented features thoroughly before sending off the new code to be merged in the master branch.



3.1: Creating a new branch (e.g. **nk_feature1**)

* Right-click the project folder
* Select Team > Switch To > New Branch..
* Make sure “Check out new branch” is checked
* Click the “Finish” button
* Note: Branch names must be unique; Do not use the same branch name as your teammates


Create Branch

Create a new branch in repository MyTeamRepoDemo

Please choose a source branch and a name for the new branch

Source: master

Branch name: nk_feature1

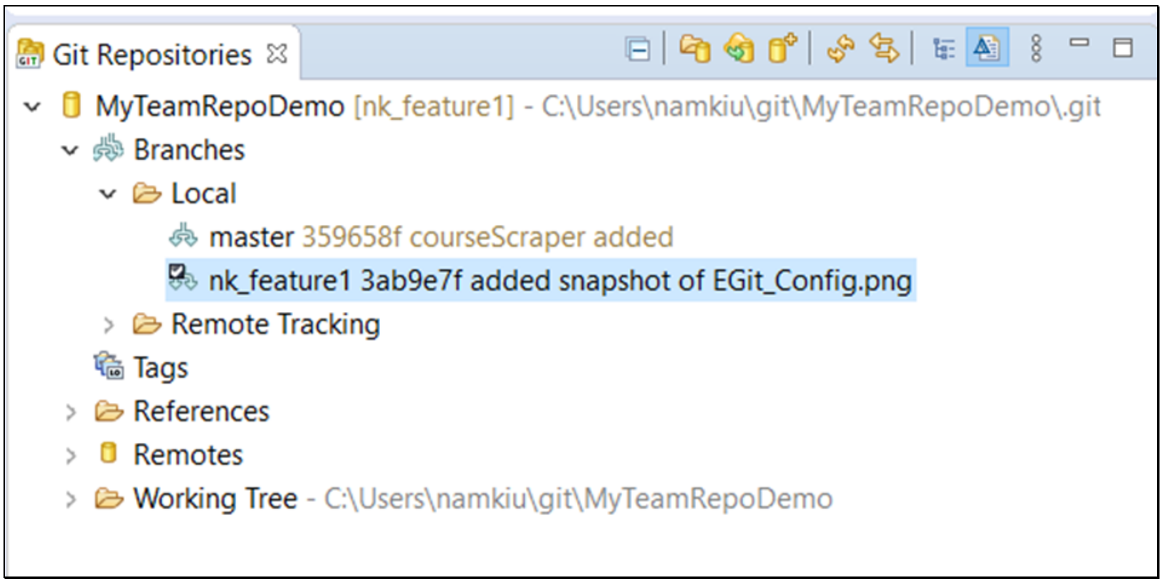
☐ Configure upstream for push and pull

When pulling:

☑ Check out new branch

3.2: Switching to different branches

In Eclipse, “Git Repositories” Window is the best tool to review and switch different branches. You can double-click any local branch to checkout and switch. If it is not visible, choose “Window > Show View > Other” to re-open this window.

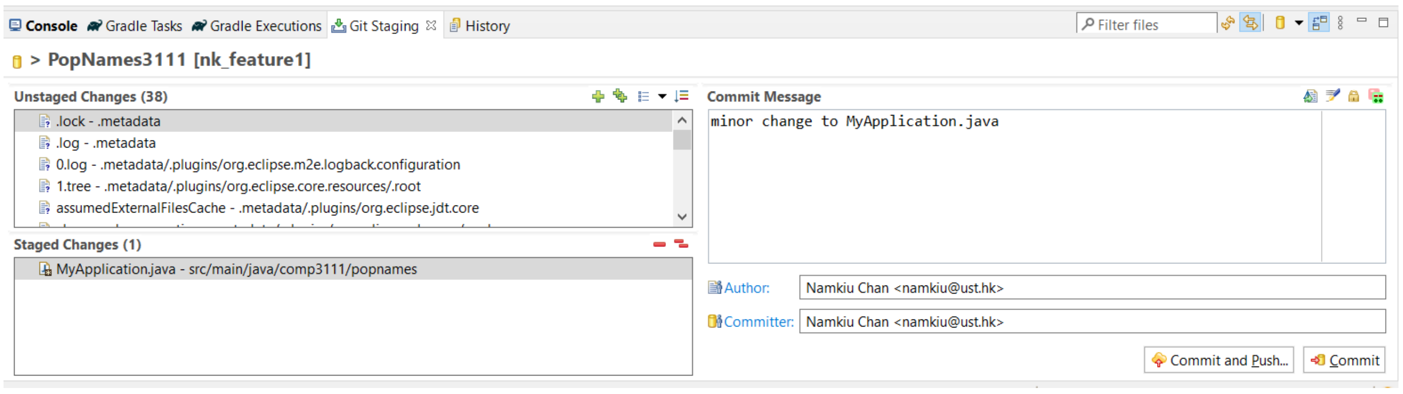

```
Git Repositories
MyTeamRepoDemo [nk_feature1] - C:\Users\namkiu\git\MyTeamRepoDemo\.git
  Branches
    Local
      master 359658f courseScraper added
      nk_feature1 3ab9e7f added snapshot of EGit_Config.png
    Remote Tracking
  Tags
  References
  Remotes
  Working Tree - C:\Users\namkiu\git\MyTeamRepoDemo
```

3.3: Make some changes

- Check to confirm that you are currently working in a development branch (e.g. **nk_feature1**)
- Double-click at MyApplication.java and make the following changes to **line 50**:
  - [BEFORE] stage.setTitle("**Super Team T-00: Data Explorer on COVID-19**");
  - [AFTER] stage.setTitle("**Any Title You Like**");
- Try running the program and observe the differences before and after the changes

3.4: Commit the changes to the development branch (local repo)

- Check to confirm that you are currently working in a development branch (e.g. **nk_feature1**)
- At “Git Staging” tab, stage the change of **MyApplication.java** (by clicking the ‘+’ button); Type in an appropriate commit message and click at the “Commit” button (not the “Commit and Push..”)



> PopNames3111 [nk_feature1]

Unstaged Changes (38)

.lock - .metadata
.log - .metadata
0.log - .metadata/.plugins/org.eclipse.m2e.logback.configuration
1.tree - .metadata/.plugins/org.eclipse.core.resources/.root
assumedExternalFilesCache - .metadata/.plugins/org.eclipse.jdt.core

Staged Changes (1)

MyApplication.java - src/main/java/comp3111/popnames

Commit Message

```
minor change to MyApplication.java
```

Author: Namkiu Chan <namkiu@ust.hk>

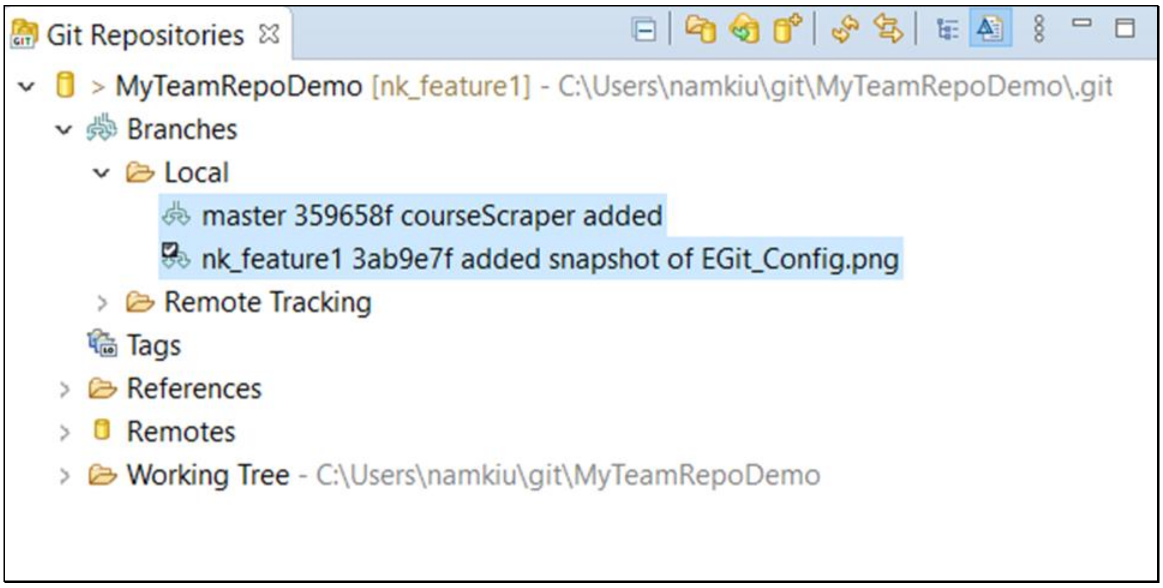
Committer: Namkiu Chan <namkiu@ust.hk>

3.5: Keep working on the development branch
You should keep working on your development branch. The changes won’t affect other branches. There are many advanced features (e.g. restore the branch from a previous commit, rebasing, renaming branch, merging, etc.). Eclipse IDE should support most of these advanced features in “Git Repositories” or “Git Staging” window. In this lab, you might edit **README.md** to show the updated information about the team, the members, and the task assignment, in collaboration with other team members.

3.6: Pull changes from GitHub
Before pushing the changes to GitHub, we should pull, check, and merge changes from GitHub. Ideally, there won’t be any conflict. If conflicts happen, the project team needs to resolve them.

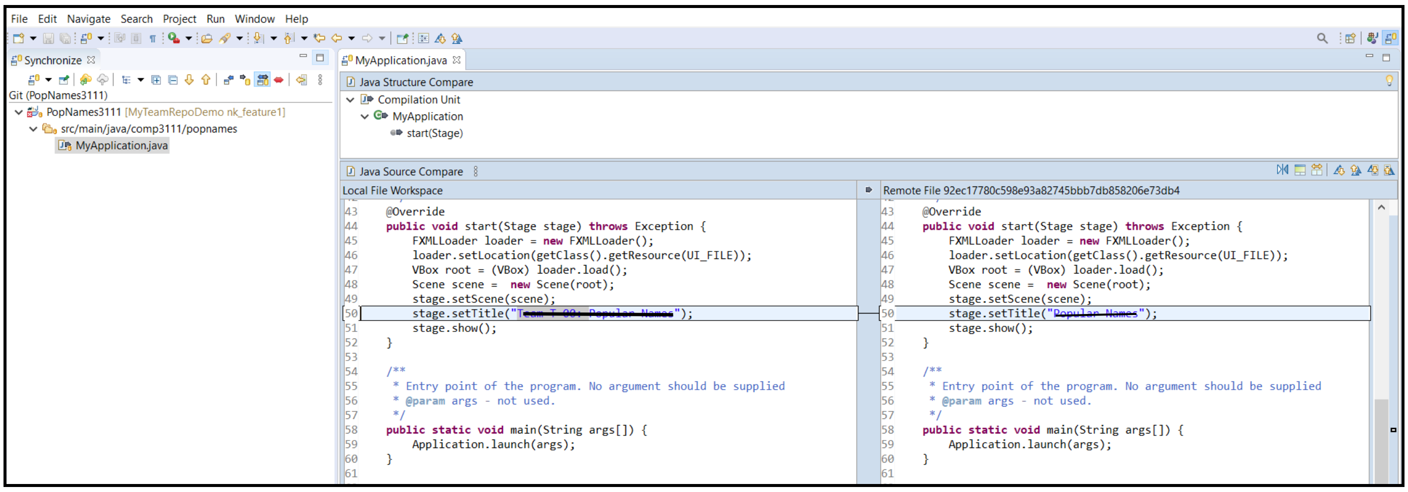
- Right-click the repository icon in “Git Repositories” window
- Select “Pull”: There may be an error message (which can be ignored). The reason is that in the current step, we don’t have the development branch (i.e. **nk_feature1**) in GitHub.

3.7: Review the differences between the master branch and development branch
- In “Git Repositories” tab, press “SHIFT” key to select 2 branches
- 2 items selected should be shown in the status below



> MyTeamRepoDemo [nk_feature1] - C:\Users\namkiu\git\MyTeamRepoDemo\.git
Branches
Local
master 359658f courseScraper added
nk_feature1 3ab9e7f added snapshot of EGit_Config.png
Remote Tracking
Tags
References
Remotes
Working Tree - C:\Users\namkiu\git\MyTeamRepoDemo

- Right click, and select “Synchronize with each other” to enable a “Synchronize” perspective
- Double-click at **MyApplication.java**, and you can review the differences between 2 branches
- You can switch back to Java perspective by: Window > Perspective > Close perspective



Local File Workspace

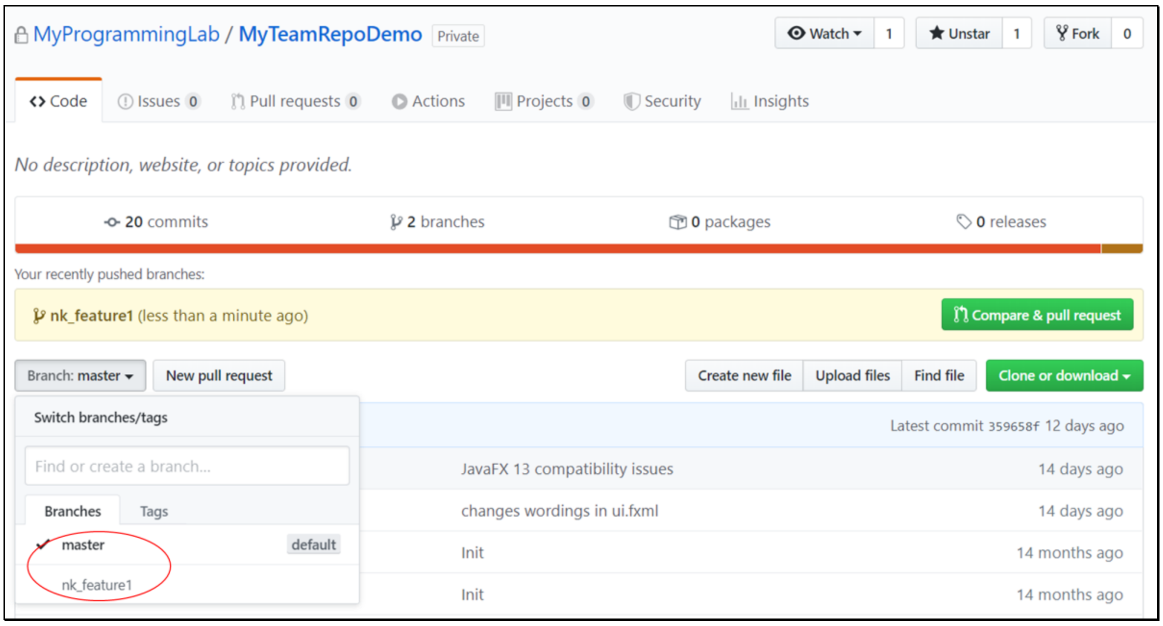
```
    @Override
    public void start(Stage stage) throws Exception {
        FXMLLoader loader = new FXMLLoader();
        loader.setLocation(getClass().getResource(UI_FILE));
        VBox root = (VBox) loader.load();
        Scene scene =  new Scene(root);
        stage.setScene(scene);
        stage.setTitle("~~Team T-00: Popular Names~~");
        stage.show();
    }

    /**
     * Entry point of the program. No argument should be supplied
     * @param args - not used.
     */
    public static void main(String args[]) {
        Application.launch(args);
    }
```

Remote File 92ec17780c598e93a82745bbb7db858206e73db4

```
    @Override
    public void start(Stage stage) throws Exception {
        FXMLLoader loader = new FXMLLoader();
        loader.setLocation(getClass().getResource(UI_FILE));
        VBox root = (VBox) loader.load();
        Scene scene =  new Scene(root);
        stage.setScene(scene);
        stage.setTitle("~~Popular Names~~");
        stage.show();
    }

    /**
     * Entry point of the program. No argument should be supplied
     * @param args - not used.
     */
    public static void main(String args[]) {
        Application.launch(args);
    }
```

3.8: Push the development branch to GitHub

- Select the development branch (i.e. **nk_feature1**) in “Git Repositories”
- Right-click, choose “Push Branch…”
- Accept all default settings
- Go to the GitHub repository via web browser, and confirm that both branches (master and nk_feature1) have been made available



No description, website, or topics provided.

20 commits | 2 branches | 0 packages | 0 releases

Your recently pushed branches:

nk_feature1 (less than a minute ago)

✓ master default

nk_feature1

Latest commit 359658f 12 days ago

JavaFX 13 compatibility issues	14 days ago
changes wordings in ui.fxml	14 days ago
Init	14 months ago
Init	14 months ago

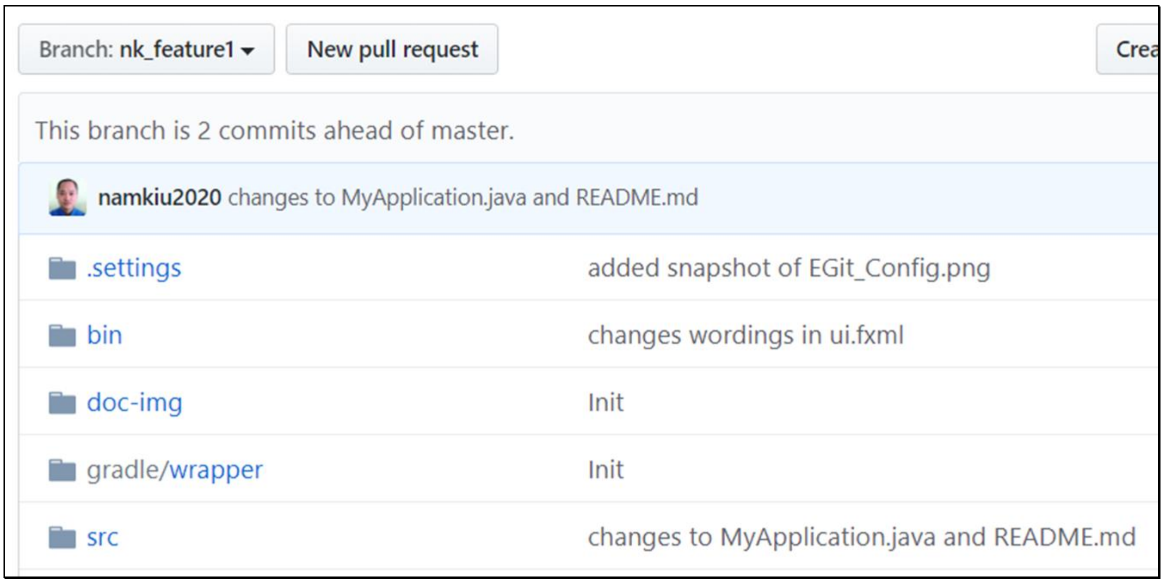
Exercise 4: Pull Request at GitHub

Pull request is a systematic way to merge changes from one branch (e.g. **nk_feature1**) to another branch (e.g. master). Before merging the changes, a GitHub pull request provides a web interface to let developers discuss and review the changes.

With some advanced settings in GitHub, several operations can be applied (e.g. continuous integration) as well to automatically rebuild and run unit tests. Students who are interested can explore relevant tutorials to add these features.

4.1: Switch to the development branch at GitHub

- From the "Branch" combo box, select the development branch (i.e. **nk_feature1**)
- You could check to see that MyApplication.java has been changed in this branch

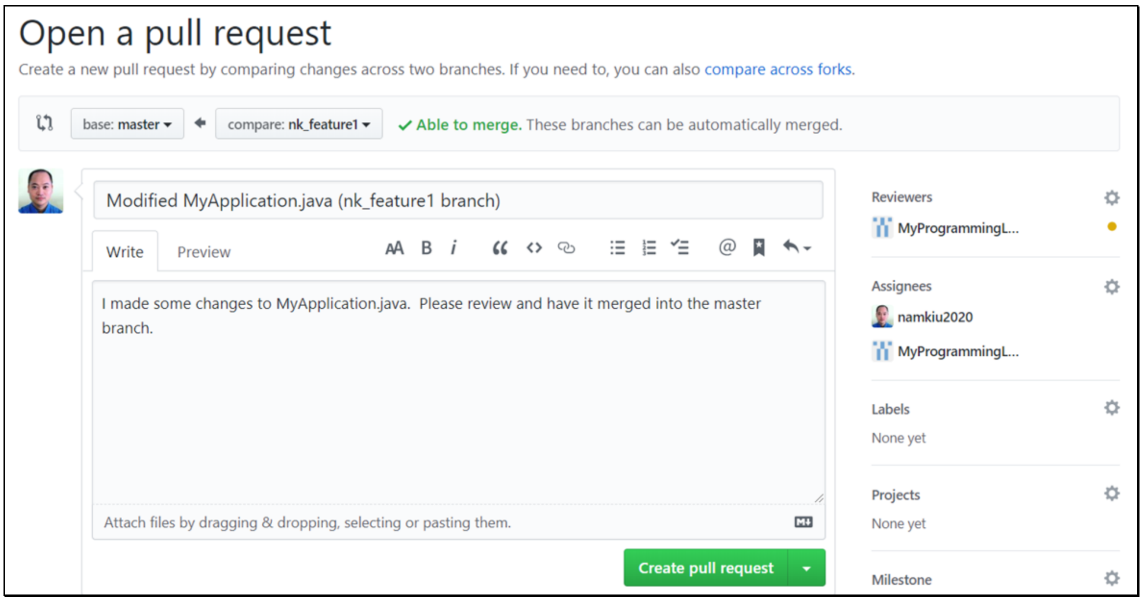


This branch is 2 commits ahead of master.

namkiu2020 changes to MyApplication.java and README.md

.settings	added snapshot of EGit_Config.png
bin	changes wordings in ui.fxml
doc-img	Init
gradle/wrapper	Init
src	changes to MyApplication.java and README.md

4.2: Creating a new pull request
- Click "New pull request"; Type in a message and click "Create pull request"
- You can assign reviewers (if your project has multiple contributors) on the right-hand side



Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

✓ Able to merge. These branches can be automatically merged.

Modified MyApplication.java (nk_feature1 branch)

I made some changes to MyApplication.java. Please review and have it merged into the master branch.

Attach files by dragging & dropping, selecting or pasting them.

Reviewers: MyProgrammingL...

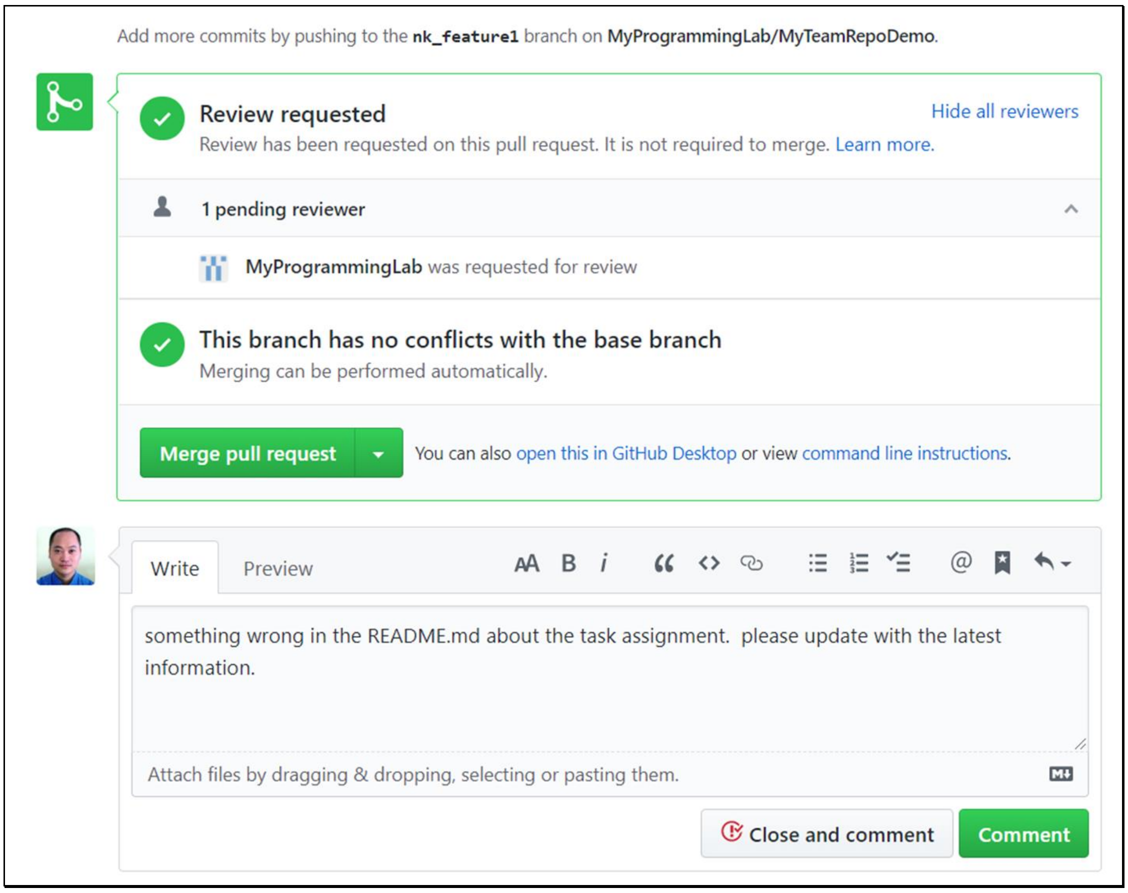
Assignees: namkiu2020, MyProgrammingL...

Labels: None yet

Projects: None yet

4.3: Have an online discussion before merging

- The project team leader **SHOULD** review the codes before pressing the "Merge pull request" button
- If the code quality is not good, the project team leader **SHOULD** provide feedback with suggestions
- No need to close this request. Just ask the developer to modify and re-commit subsequent changes.


Add more commits by pushing to the `nk_feature1` branch on MyProgrammingLab/MyTeamRepoDemo.

Review requested

Review has been requested on this pull request. It is not required to merge. Learn more.

1 pending reviewer

MyProgrammingLab was requested for review

This branch has no conflicts with the base branch

Merging can be performed automatically.

You can also open this in GitHub Desktop or view command line instructions.

something wrong in the README.md about the task assignment. please update with the latest information.

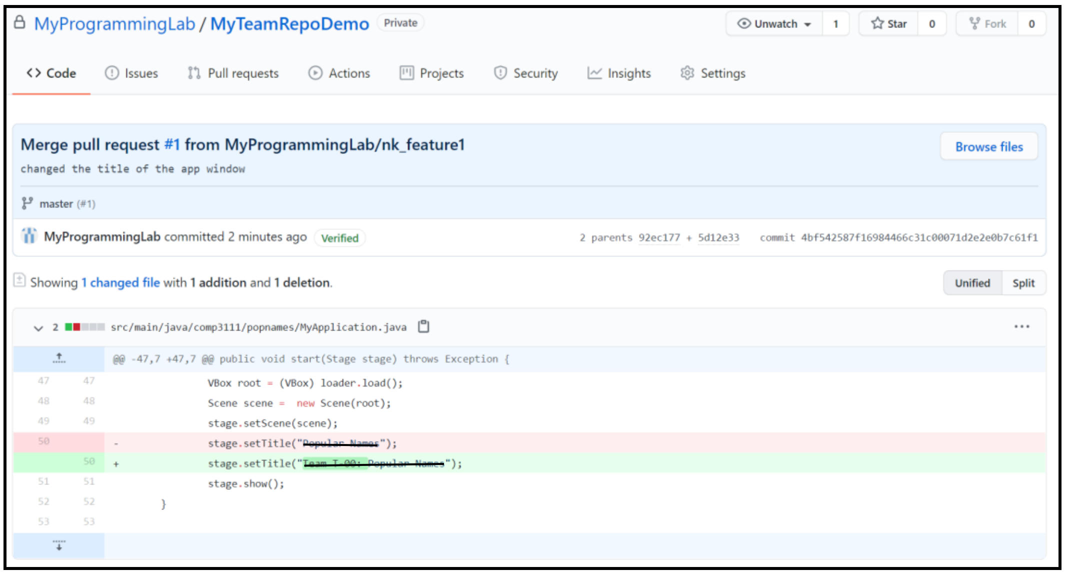
Attach files by dragging & dropping, selecting or pasting them.

4.4: Modifying the development branch and commit

- Developer can keep modifying and committing the development branch
- You can choose "Commit and Push…" to the remote branch directly to streamline the process

4.5: Further review and comment

In the project, we recommend having the team leader as the **ONLY** person to merge changes to the master branch. However, for the lab work, please allow every team member to have a temporary power to approve a pull request.


MyProgrammingLab / MyTeamRepoDemo Private
Unwatch 1 Star 0 Fork 0
Code Issues Pull requests Actions Projects Security Insights Settings
Browse files
Unified Split

Merge pull request #1 from MyProgrammingLab/nk_feature1

```
changed the title of the app window
```

master (#1)

MyProgrammingLab committed 2 minutes ago Verified

```
2 parents 92ec177 + 5d12e33    commit 4bf542587f16984466c31c00071d2e2e0b7c61f1
```

Showing 1 changed file with 1 addition and 1 deletion.

```
2 src/main/java/comp3111/popnames/MyApplication.java

@@ -47,7 +47,7 @@ public void start(Stage stage) throws Exception {
                VBox root = (VBox) loader.load();
                Scene scene =  new Scene(root);
                stage.setScene(scene);
-               stage.setTitle("~~Popular Names~~");
+               stage.setTitle("~~Team T-00: Popular Names~~");
                stage.show();
        }

```

Lab Activity

- Setup a team repo and work through Exercise 1, 2, 3 and 4 together with your teammates
- Edit **README.md** and provide your personal information and individual task assignment
- Create a development branch and commit the change; Push the change to GitHub
- Make a pull request and comment it as “Trivial PR”
- Approve the pull request

Assessment

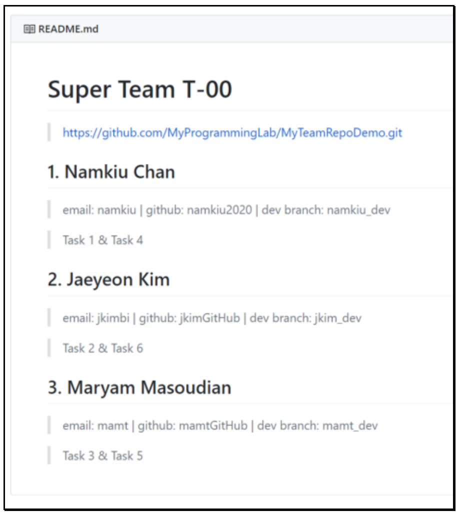
- Submit the URL of your team repo for assessment through Canvas by the due date
- Students from the same team should each submit an identical URL of their team repo
- Criteria of assessment for the team repo

(i) Must be a private repository;

(ii) Must have invited all members and TAs as collaborators;

(iii) Must contain an audit record showing that every team member had already made at least one pull request (the “Trivial PR”) and approved it;

(iv) Must display the content of **README.md** showing the Team ID, members’ personal information, and individual task assignment (presented in similar format as shown below):


README.md

Super Team T-00

https://github.com/MyProgrammingLab/MyTeamRepoDemo.git

1. Namkiu Chan

email: namkiu | github: namkiu2020 | dev branch: namkiu_dev

Task 1 & Task 4

2. Jaeyeon Kim

email: jkimbi | github: jkimGitHub | dev branch: jkim_dev

Task 2 & Task 6

3. Maryam Masoudian

email: mamt | github: mamtGitHub | dev branch: mamt_dev

Task 3 & Task 5

Online Learning Resources on Git/GitHub

Topics	Resources
Concept	What is Git - Atlassian
	Video: Git Basics Episode 2 – What is Git? (8:15)
	Video: Git Basics Episode 3 – Get Going with Git (4:26)
Basic	Git Handbook - GitHub
	EGit Tutorial - EclipseSource
	Pro Git (Chapter 2): Git Basics
	Pro Git (Chapter 6): GitHub
	Git for Beginners
Branching	Video: GIT – Working with Branches (6:36)
	Video: GIT – Merging and Workflow (7:10)
	Interactive Tutorial: Learn Git Branching
	Git Branch - Atlassian
Cheat Sheet	Git Cheat Sheet - Atlassian
	Git Cheat Sheet - GitHub

Glossary: Key terms and operations about Git/GitHub

Key Terms	**Description**
branch	Branch contains a series of commits derived from the main (master) trunk. Very useful when developing and testing a new feature.
commit *(noun)*	Commits are snapshots of a repo (project milestones) where rollback in time can be made possible.
master	The name of the default branch of a repo.
origin	The name of the default remote repo.
repo	A repository where all project files are stored. There are local repos (e.g. Git) and remote repos (e.g. GitHub, Bitbucket).
Key Operations	**Descriptions**
checkout	To switch between branches.
clone	To download the entire repo from a remote site (remote repo) and create an identical repo in your local machine (local repo).
commit *(verb)*	To take a snapshot of your repo.
fork	To duplicate a remote repo on GitHub and acquire full access rights on it. We are NOT using fork in our project.
merge	To combine two branches.
pull	To download from your remote repo to your local repo.
pull request (PR)	To request the repo owner to adopt a change made by the requester.
push	To upload your commit (a snapshot of repo) to your remote repo.