

GIT HUB MANUAL



메가스터디 IT 아카데미 Ver 2.0

megastudy 

서론

1. 매뉴얼을 보고 순차적으로 진행 하는 것이 아닙니다.
필요할 때 필요한 부분만 찾아서 보고 작업합니다.

[최초 한번]

Git Hub 회원가입 및 토큰생성

Git Hub 저장소 생성

[개발환경 셋업 할 때]

Git Hub 저장소에 처음으로 소스 업로드하기

Git Hub 저장소로부터 처음으로 소스 다운로드하기

[자주 사용하는 부분]

수업시간에 연습한 내용 깃허브에 업로드(push)하기

깃허브의 소스로 내 이클립스에 다운로드(pull)하기

2. 처음 개발 환경 셋업하는 부분이 생소하고 낯설지만 소스를 업로드 및 다운로드하는 부분은 굉장히 간단하고 쉽습니다.

3. 메뉴얼에 없는 내용은 스스로 구글링하여 문제를 해결합니다. (이렇게 저렇게 해보아도 안될 때 질문합니다.)

INDEX

깃허브(Git hub) 회원가입 및 토큰생성

깃허브 저장소(Git hub Repository) 생성

이클립스와 깃허브 연동 (처음으로 이클립스의 소스를 깃허브 저장소에 업로드 하기)

최신화된 소스를 깃허브에 업로드 하기

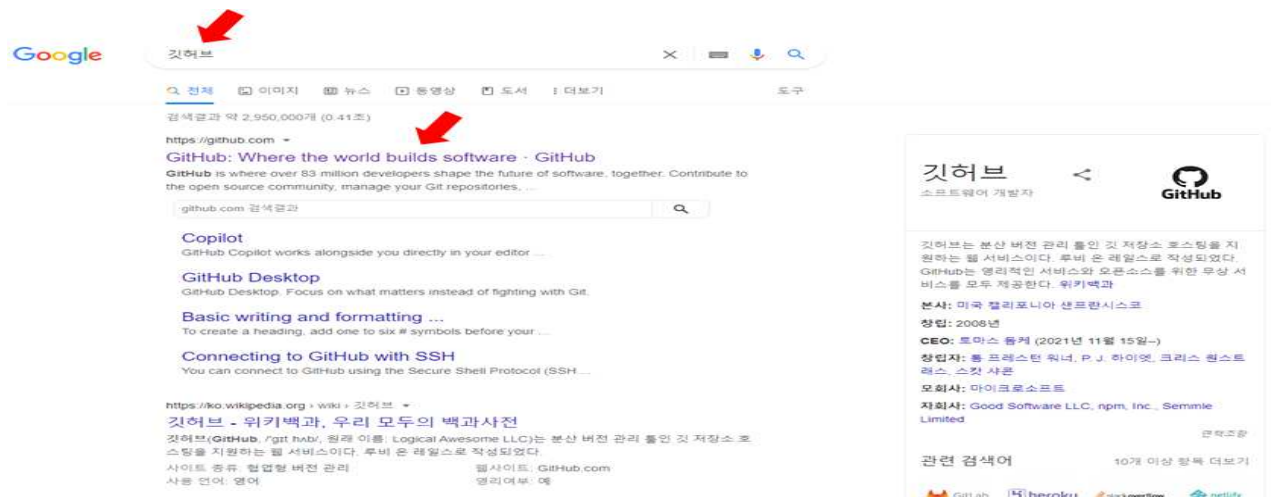
이클립스와 깃허브 연동(처음으로 이클립스의 깃허브에 있는 소스를 다운로드 하기)

최신화된 소스를 깃허브에서 다운로드 받기

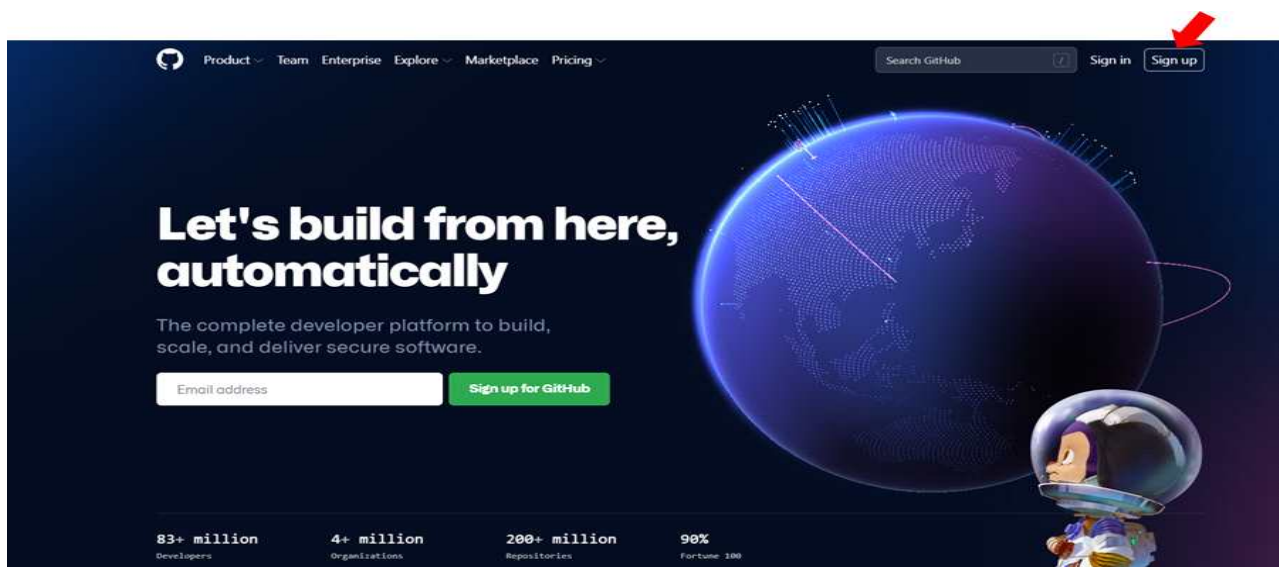
깃허브 히스토리 보기

충돌발생시 해결방법

* 회원가입 , 로그인 , Access Token 생성



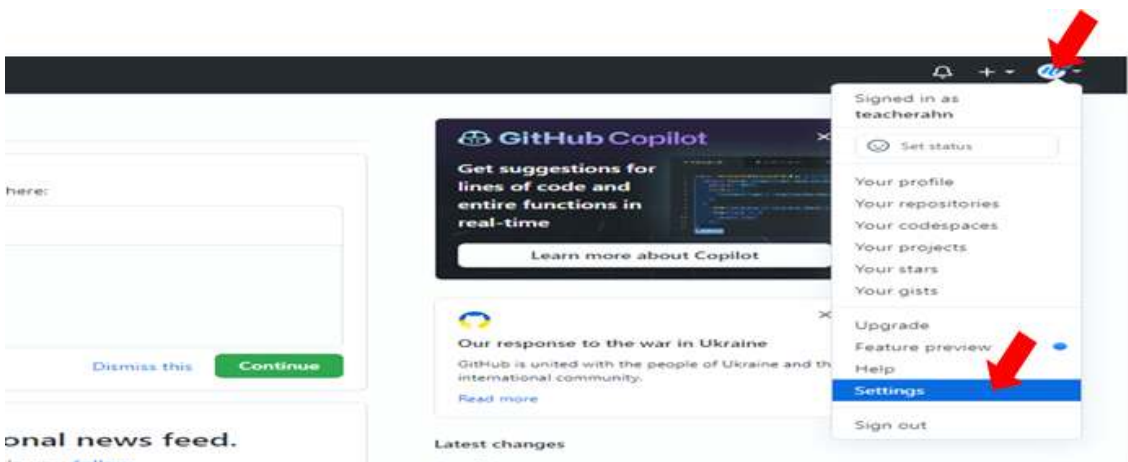
- 구글에 깃허브를 검색하여 깃허브 공식 사이트로 이동합니다.



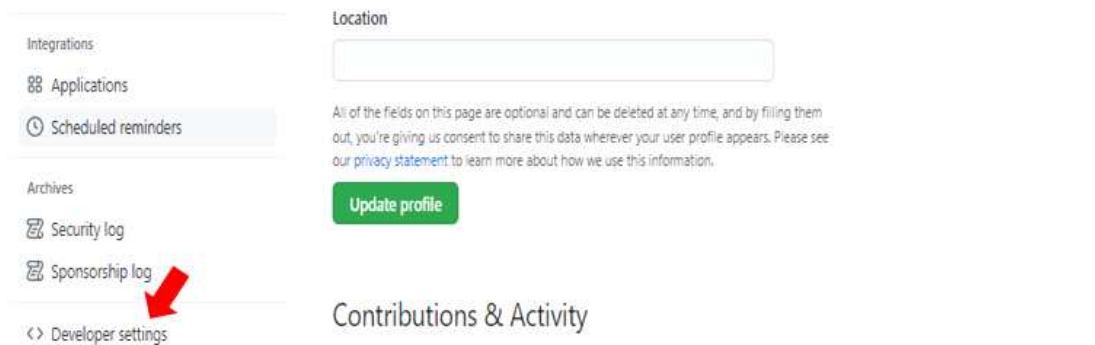
- 회원가입을 진행합니다.
- 화면이 자주 바뀌기 때문에 회원가입의 상세 내역은 캡처된 화면이 없습니다. (개발자버전 무료로 사용하시면 됩니다.)



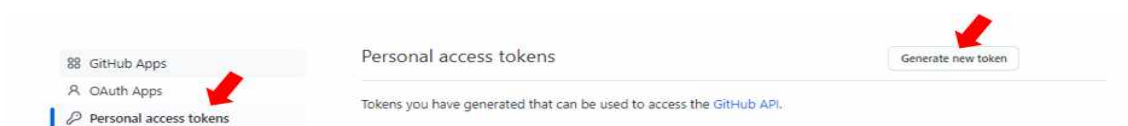
- 로그인을 진행합니다.



- 우측 상단에 프로필 이미지를 클릭하여 Settings를 클릭합니다.



- 좌측 하단 태그에 Developer settings를 클릭합니다.



- Personal access tokens을 클릭한 뒤 Generate new token을 클릭합니다.

New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Note:

web_development

What is this token for?

Expiration *

No expiration

The token will never expire

GitHub strongly recommends that you set an expiration date for your token to help keep your information secure.
[Learn more](#)

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

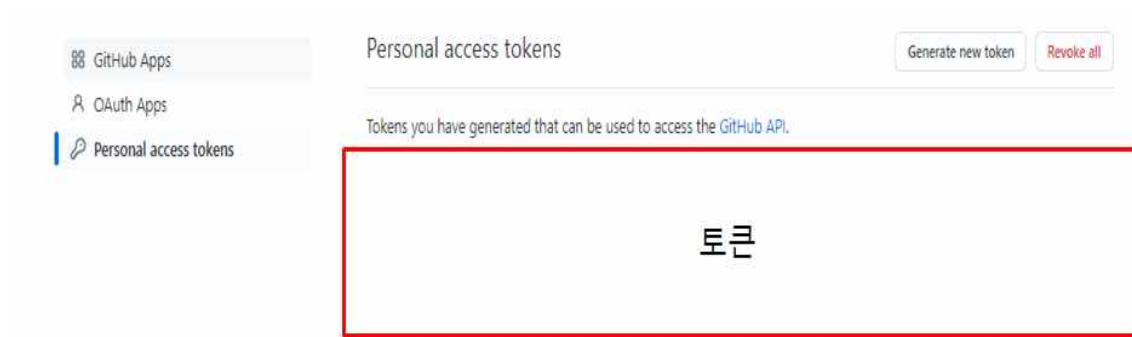
<input checked="" type="checkbox"/> repo	Full control of private repositories
<input type="checkbox"/> repo:status	Access commit status
<input type="checkbox"/> repo:deployment	Access deployment status
<input type="checkbox"/> public_repo	Access public repositories
<input type="checkbox"/> repo:write	Access repository invitations
<input type="checkbox"/> security_events	Read and write security events
<input checked="" type="checkbox"/> workflow	Update GitHub Action workflows
<input checked="" type="checkbox"/> write:packages	Upload packages to GitHub Package Registry
<input type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<input checked="" type="checkbox"/> delete:packages	Delete packages from GitHub Package Registry
<input checked="" type="checkbox"/> admin:org	Full control of orgs and teams, read and write org projects
<input type="checkbox"/> write:org	Read and write org and team membership, read and write org projects
<input type="checkbox"/> read:org	Read org and team membership, read org projects

<input checked="" type="checkbox"/> admin:public_key	Full control of user public keys
<input type="checkbox"/> write:public_key	Write user public keys
<input type="checkbox"/> read:public_key	Read user public keys
<input checked="" type="checkbox"/> admin:repo_hook	Full control of repository hooks
<input type="checkbox"/> write:repo_hook	Write repository hooks
<input type="checkbox"/> read:repo_hook	Read repository hooks
<input checked="" type="checkbox"/> admin:org_hook	Full control of organization hooks
<input checked="" type="checkbox"/> gist	Create gists
<input checked="" type="checkbox"/> notifications	Access notifications
<input checked="" type="checkbox"/> user	Update ALL user data
<input type="checkbox"/> read:user	Read ALL user profile data
<input type="checkbox"/> user:email	Access user email addresses (read-only)
<input type="checkbox"/> user:follow	Follow and unfollow users
<input checked="" type="checkbox"/> delete_repo	Delete repositories
<input checked="" type="checkbox"/> write:discussion	Read and write team discussions
<input type="checkbox"/> read:discussion	Read team discussions
<input type="checkbox"/> admin:enterprise	Full control of enterprises
<input type="checkbox"/> manage_runners:enterprise	Manage enterprise runners and runner-groups
<input type="checkbox"/> manage_billing:enterprise	Read and write enterprise billing data
<input type="checkbox"/> read:enterprise	Read enterprise profile data
<input checked="" type="checkbox"/> project	Full control of projects
<input type="checkbox"/> read:project	Read access of projects
<input type="checkbox"/> admin:gpg_key	Full control of public user GPG keys
<input type="checkbox"/> write:gpg_key	Write public user GPG keys
<input type="checkbox"/> read:gpg_key	Read public user GPG keys

Generate token

Cancel

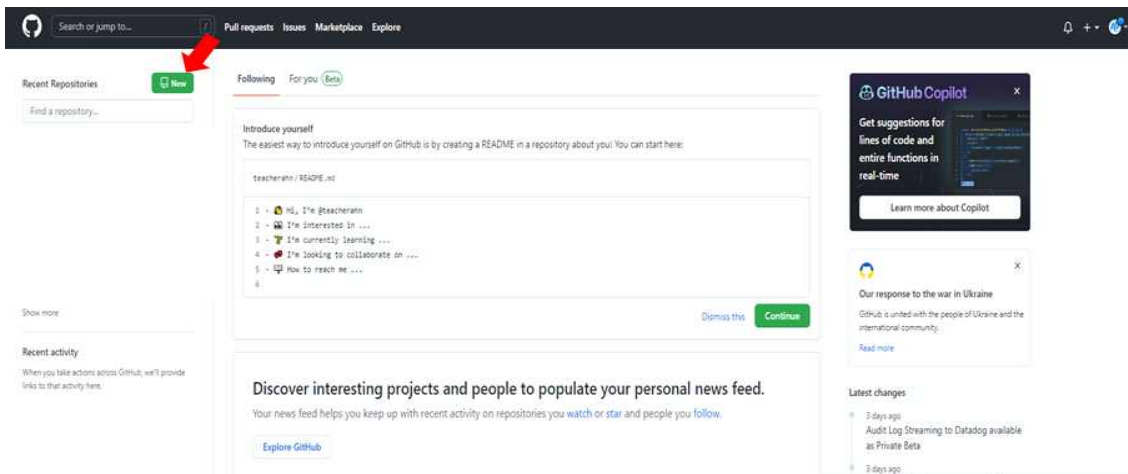
- Note에는 자유롭게 토큰 이름을 입력합니다. (예시 web_development)
- Expiration에는 No expiration으로 설정하여 토큰의 갱신일을 무기한으로 설정합니다.
- admin:enterprise와 admin:gpg_key를 제외한 모든 체크박스를 체크한 후 Generate token버튼을 클릭합니다.



- 해당 토큰을 백업해두어서 깃허브 사용(push & pull)시 인증 수단으로 사용합니다.
- 토큰을 잃어버리면 다시 찾는 것을 불가능하며 토큰을 삭제한 후 다시 생성해야 합니다.

- 끝 -

* 이클립스에서 깃허브에 소스코드 업로드하기



- 메인화면에서 New버튼을 클릭하여 저장소를 생성합니다.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)

Owner * / Repository name *

Great repository names are [test](#) is available. [orable](#). Need inspiration? How about [shiny-bassoon](#)?

Description (optional)

☐ 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.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ Add a README file
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: None

Choose a license

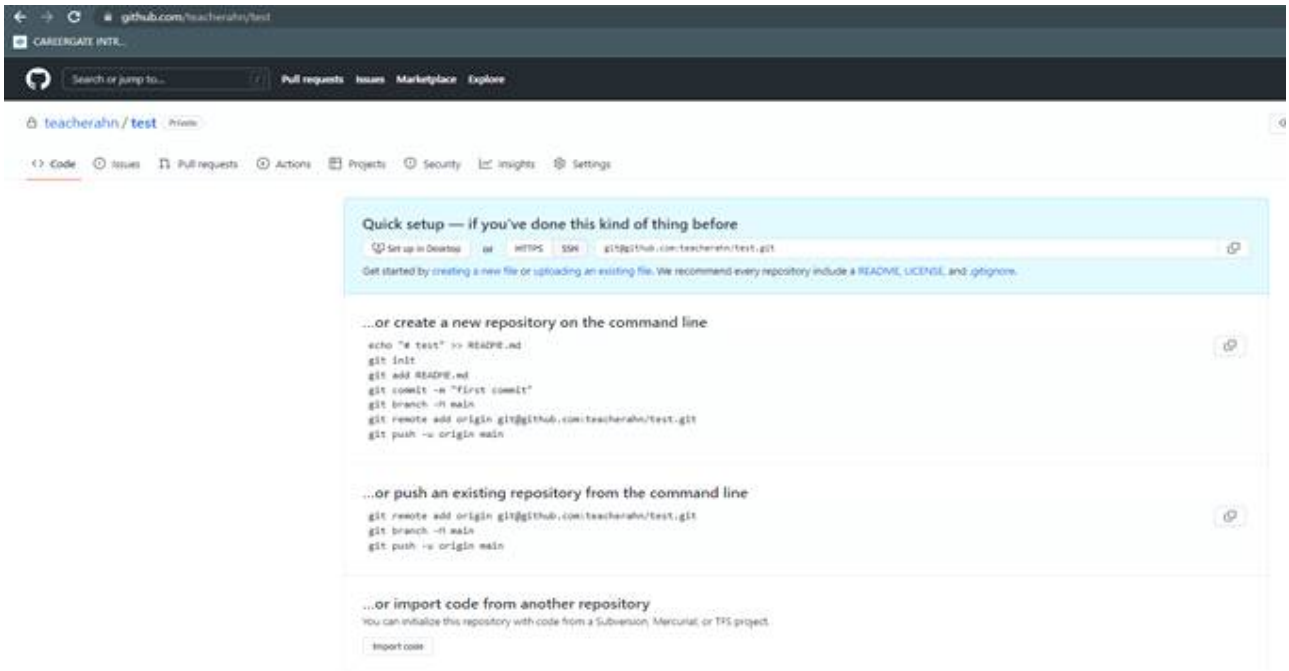
A license tells others what they can and can't do with your code. [Learn more.](#)

License: None

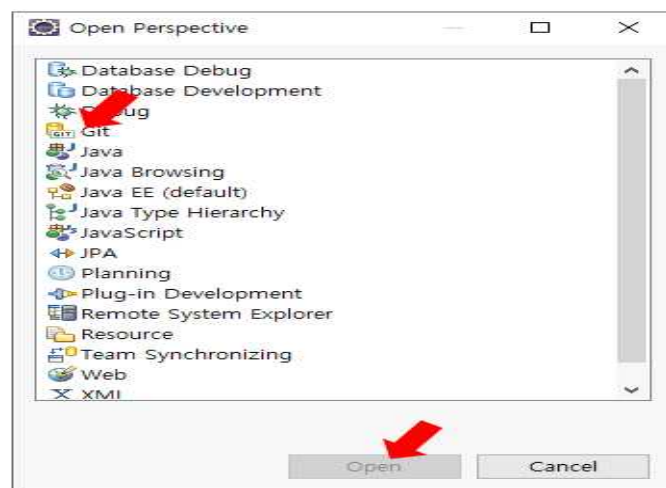
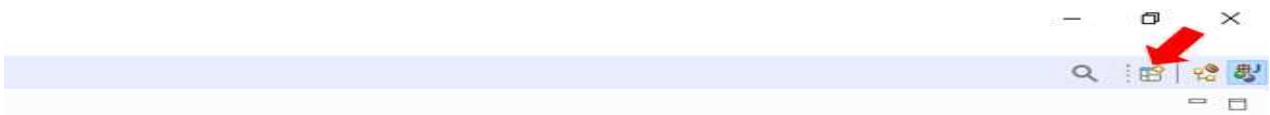
ⓘ You are creating a private repository in your personal account.

Create repository

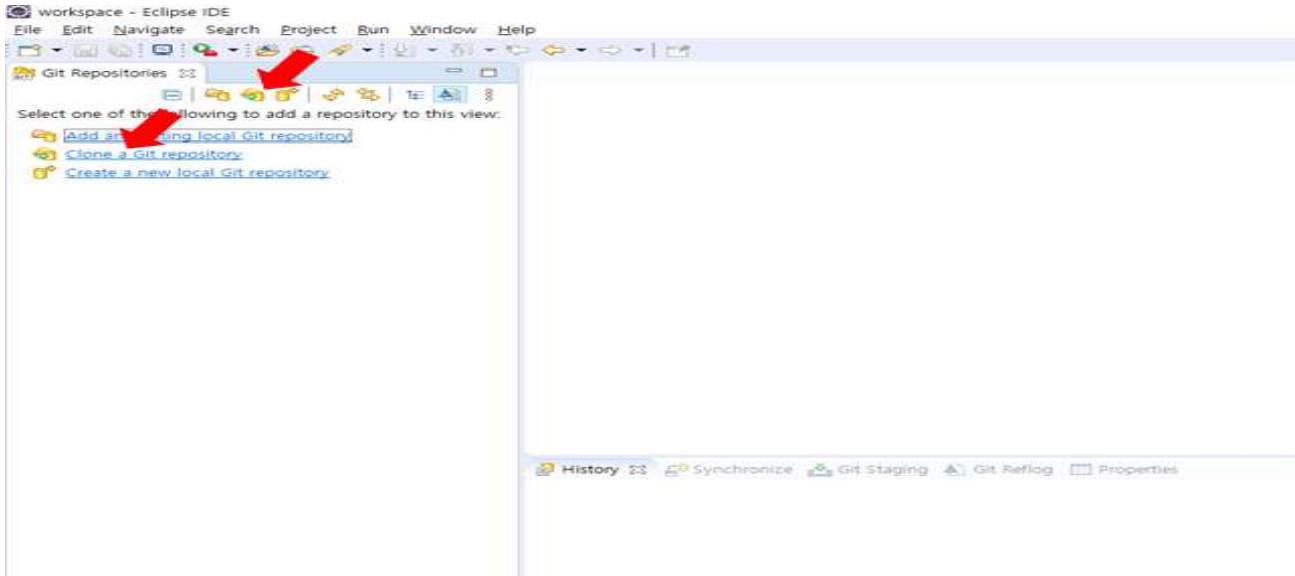
- 저장소 이름을 작성합니다.
- 공개 & 비공개의 여부를 결정합니다. (비공개로 작업)
- Create repository 버튼을 클릭합니다.



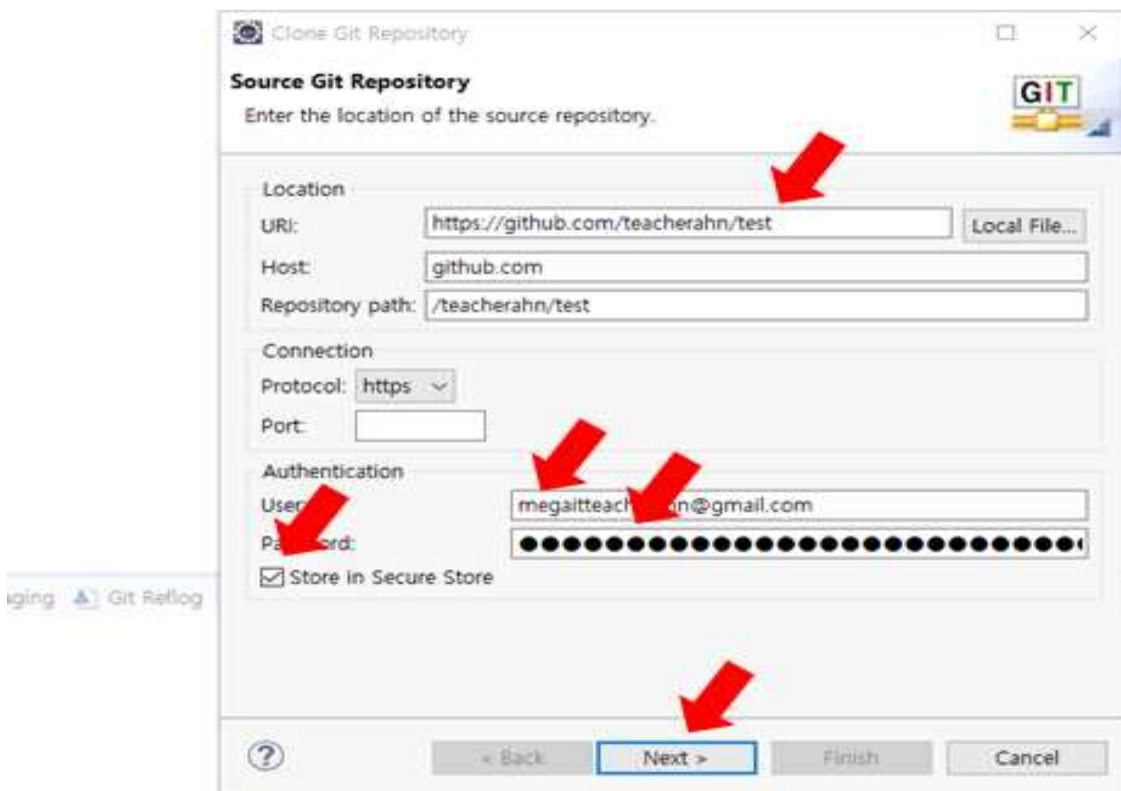
- 저장소가 성공적으로 생성된 화면입니다.



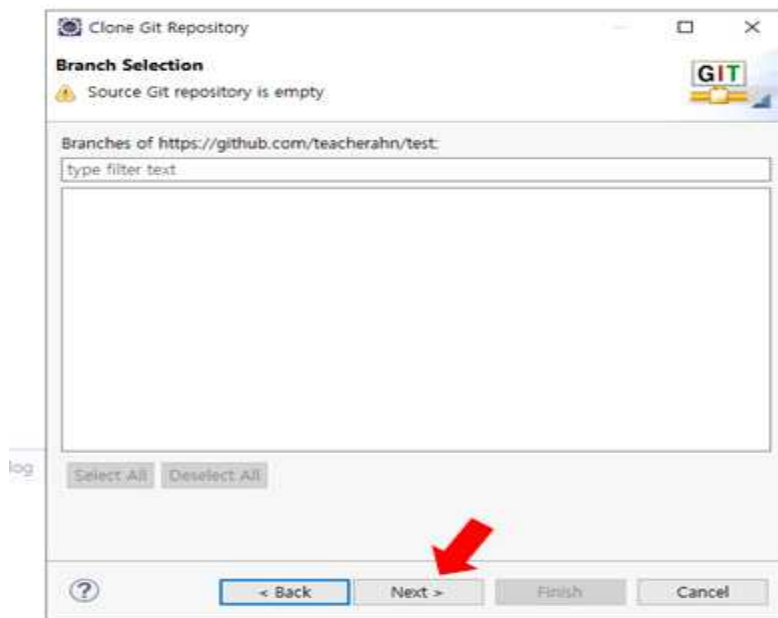
- 이클립스에서 우측상단 Perspective의 +를 클릭합니다.
- Git을 선택한뒤 Open버튼을 클릭합니다.



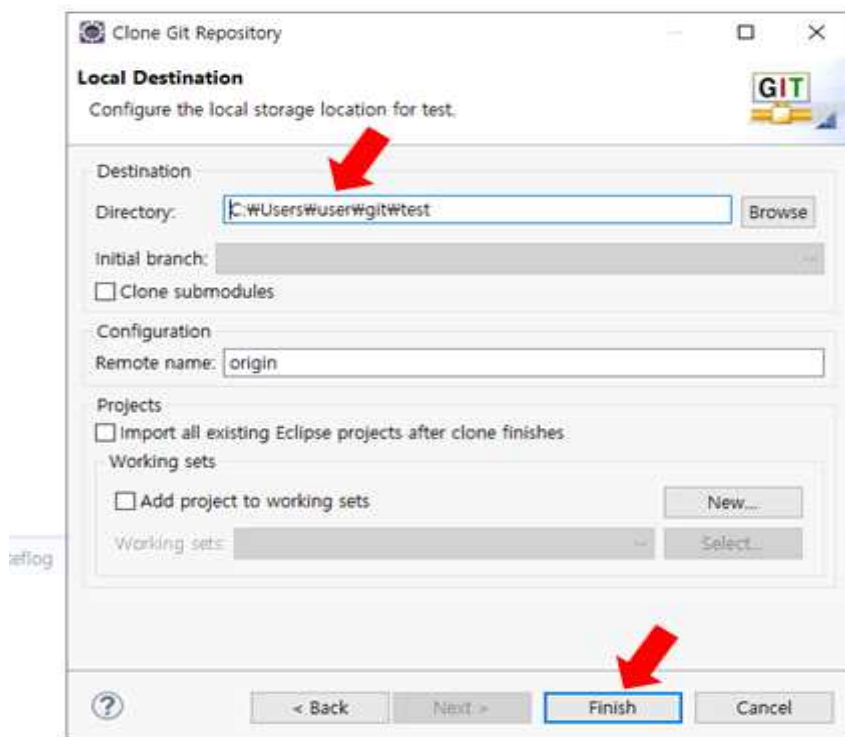
- 좌측화면에 Clone a Git repository를 클릭합니다.
- 최초화면에는 단어가 보이지만 최소 1개이상의 프로젝트가 있을 경우 Clone a Git repository 단어가 보이지 않으므로 위의 아이콘을 클릭합니다.



- URI은 생성된 깃 화면의 URL에서 복사하여 붙여 넣기 합니다.
- User는 로그인시에 사용하는 계정을 기록합니다.
- Password는 로그인 할 때 사용하는 비밀번호가 아니라 Personal Access Token을 기록합니다.
- 자주 패스워드를 입력하는 것이 귀찮을 수 있으므로 Store in Secure Store 체크박스를 선택하여 이후 부터는 유저명과 토큰을 입력하지 않도록 해도 좋습니다.



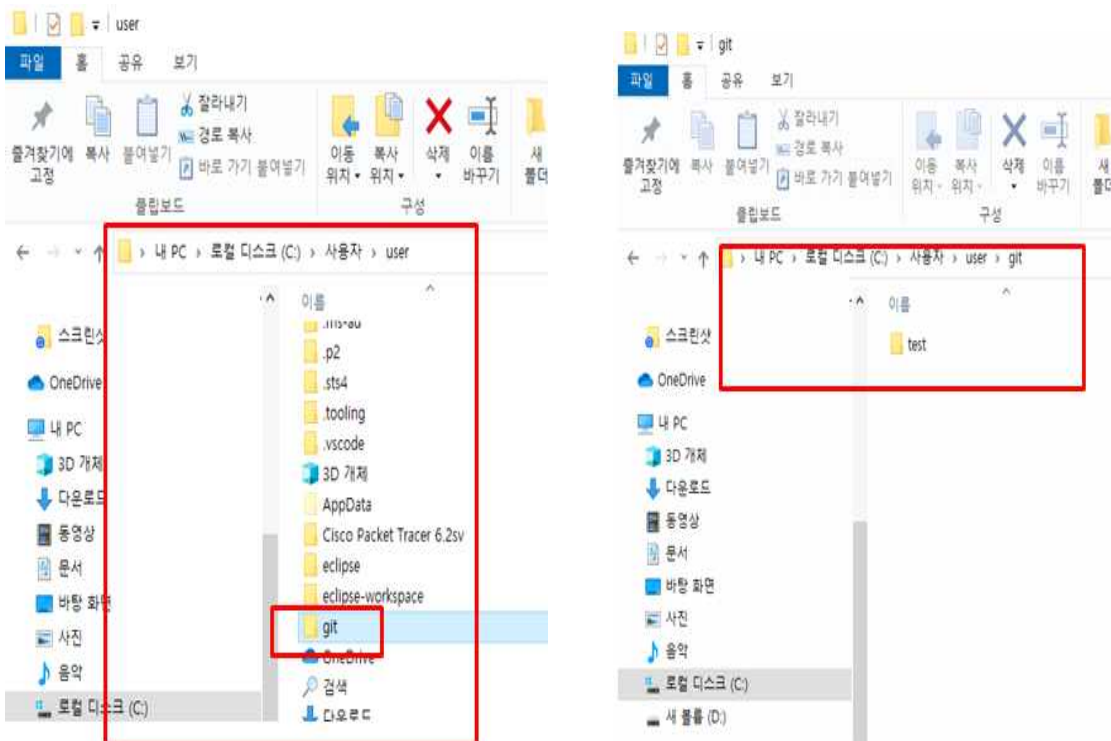
- Next를 클릭합니다.



- Directory에서 저장된 위치를 확인 혹은 지정합니다.
- Finish를 클릭합니다.



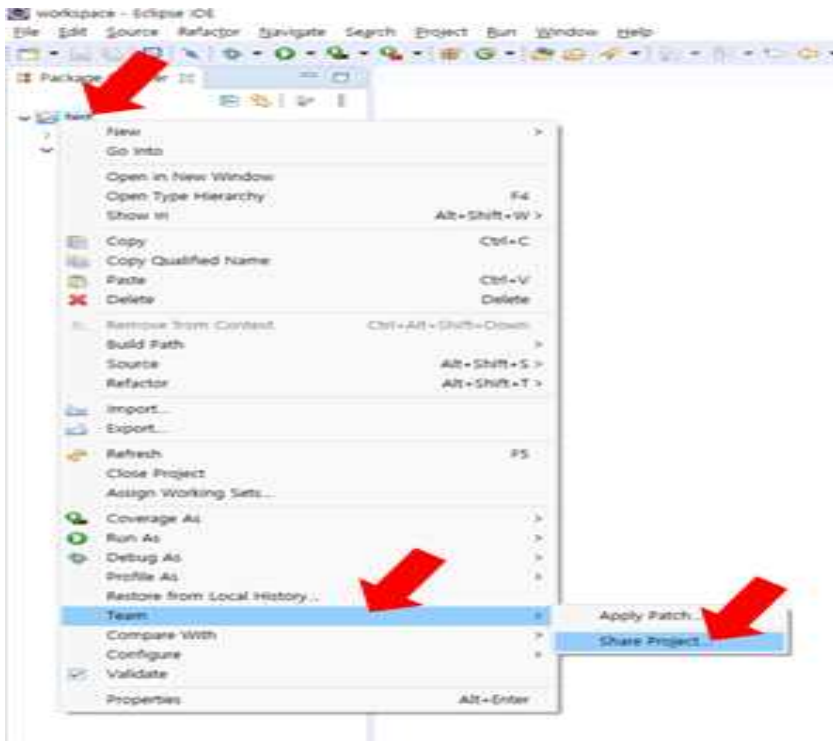
- 저장소가 성공적으로 생성된 모습입니다.



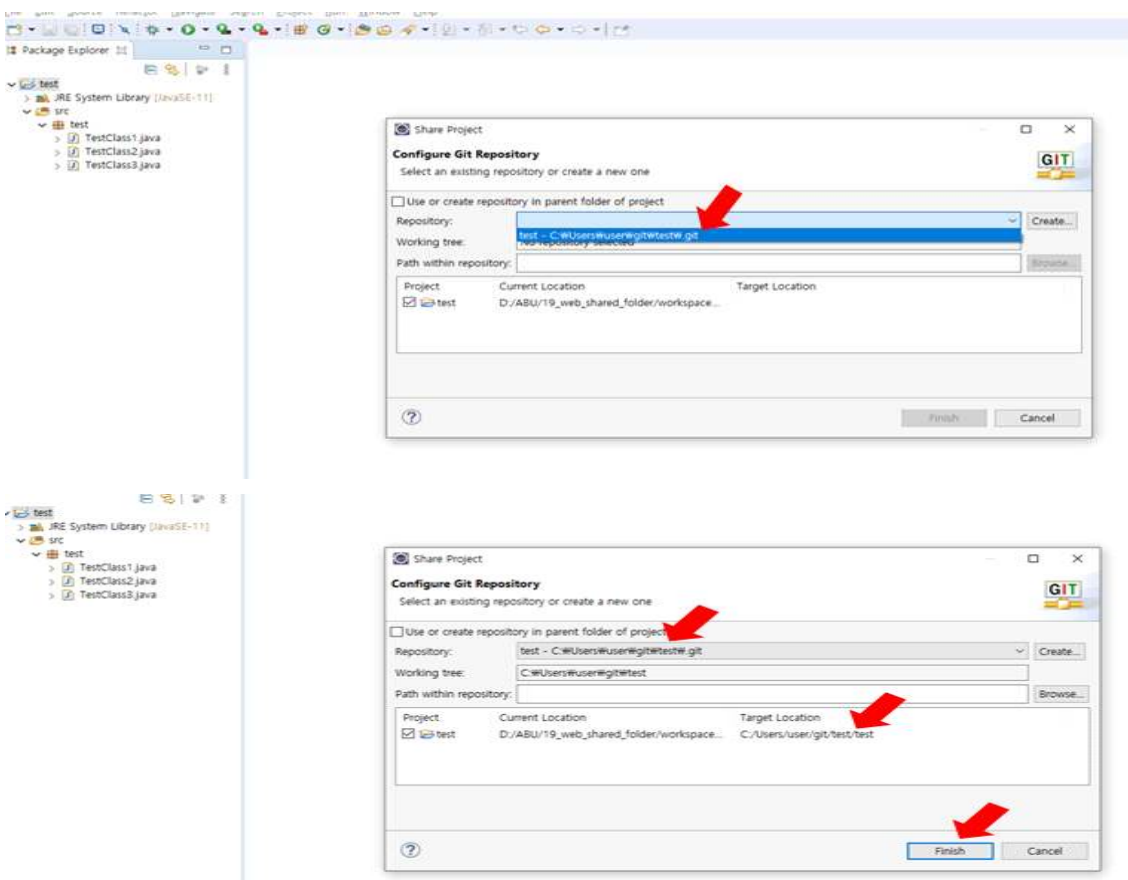
- 실제 저장소가 생성된 위치를 확인합니다.



- 이클립스의 Perspective를 다시 개발환경으로 전환합니다.

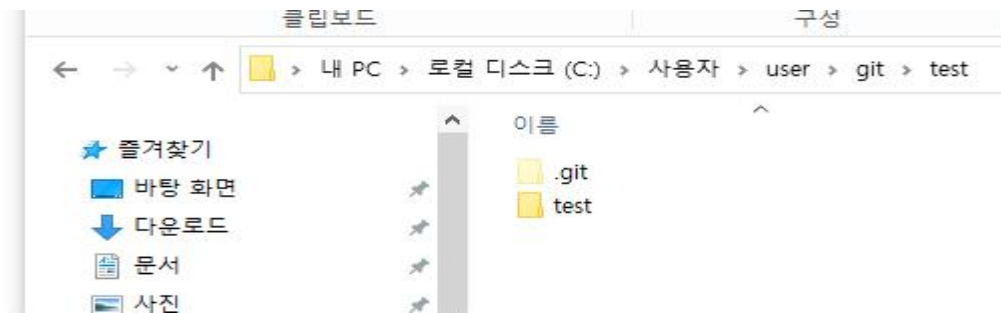


- 프로젝트를 우클릭한뒤 Team의 Share Project를 클릭합니다.

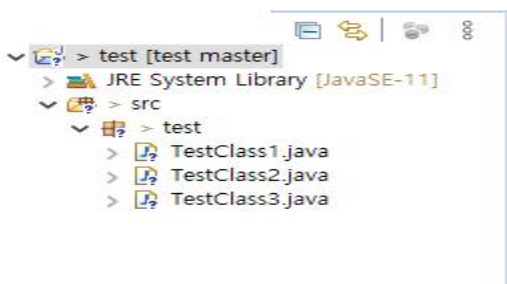


- Repository를 git에서 clone한 저장소를 정확히 선택합니다.

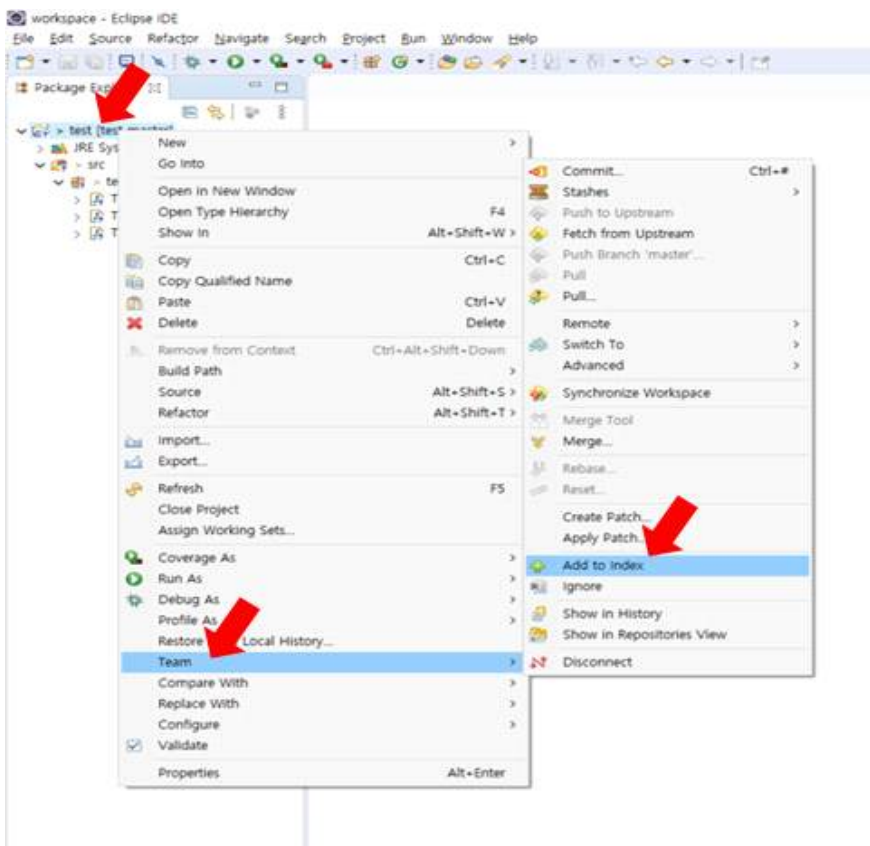
(다른 저장소를 선택할 경우 복구 작업이 상당히 번거로우므로 한 번에 정확히 선택을 합니다.)



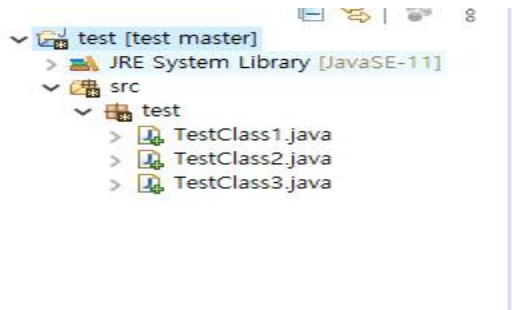
- 실제 프로젝트소스가 git폴더로 이동하였습니다.



- Share Project를 선택한 결과 보여지는 화면입니다.



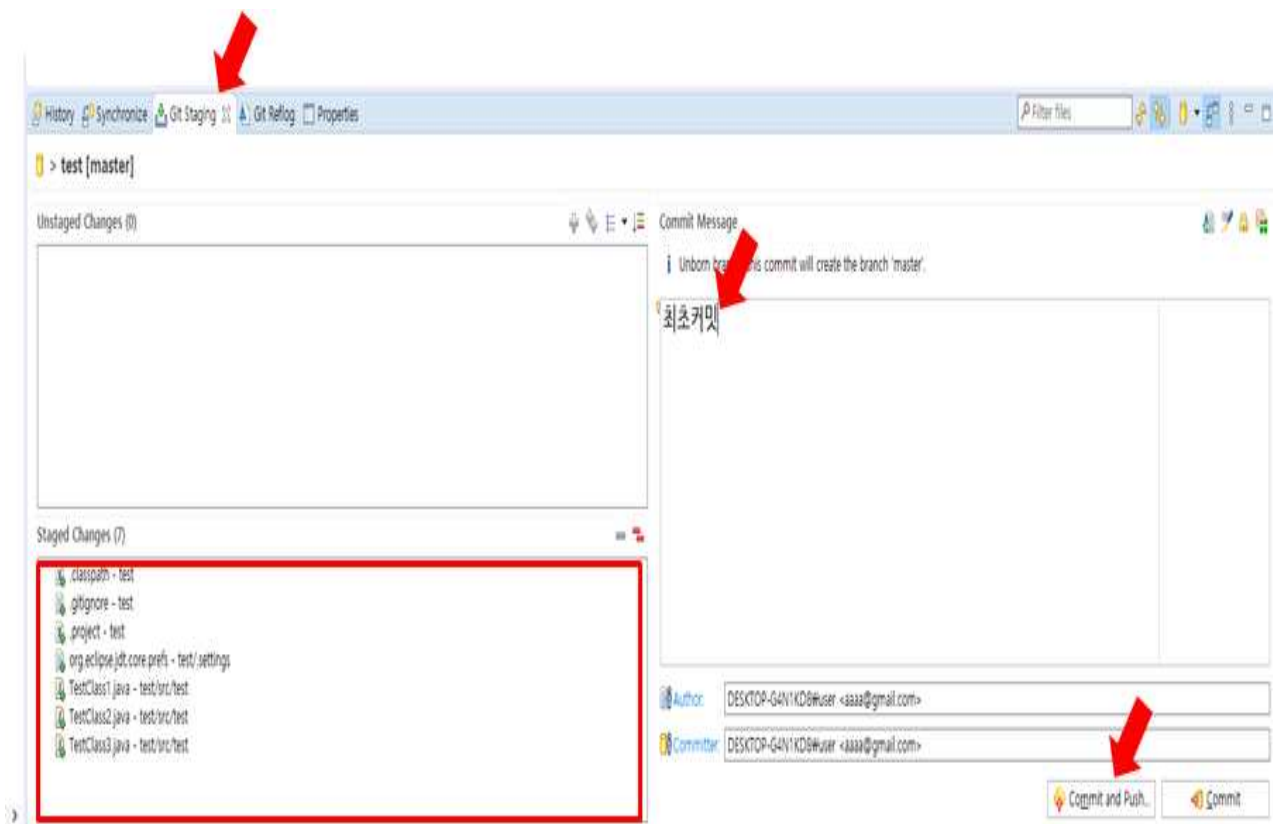
- 다시 프로젝트를 우클릭한뒤 Team > Add to Index를 클릭합니다.



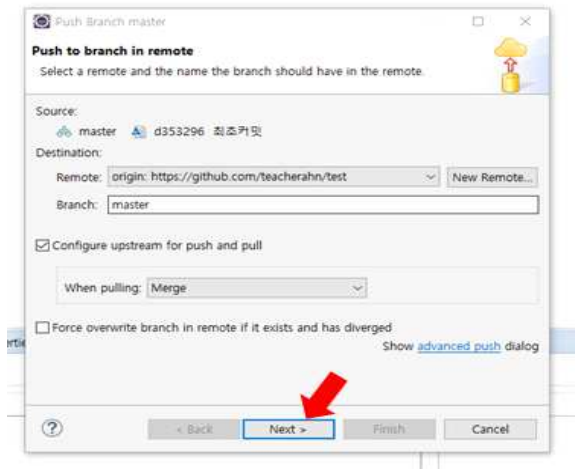
- Add to Index를 클릭하였을 경우 보여지는 화면입니다.



- 다시 Perspective를 Git으로 변경합니다.



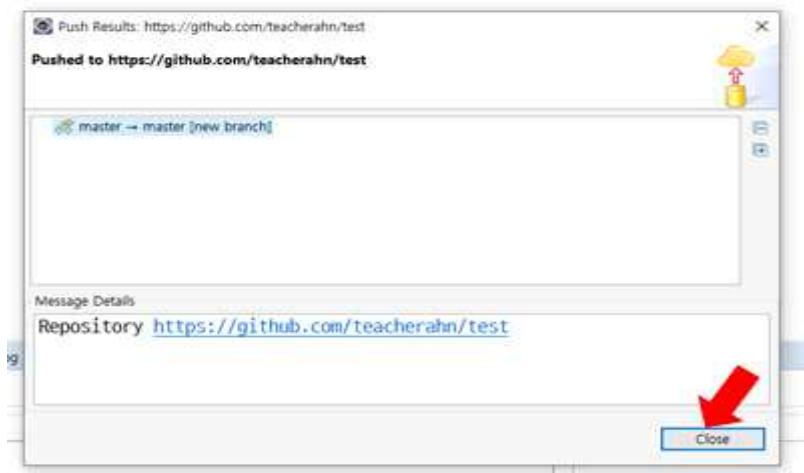
- 화면 중앙에 Git Staging태그를 클릭합니다.
- Unstaged Changes의 리스트는 깃허브와 다르지만 깃허브로 Push되지 않을 리스트입니다.
- Staged Changes의 리스트는 깃허브로 Push할 리스트입니다.
- Unstaged Changes의 + 및 ++버튼 , Staged Changes의 - 및 --버튼을 활용하여 최종적으로 Push할 목록들을 지정합니다.
- Commit Message를 작성합니다. 필수로 작성해야 합니다.



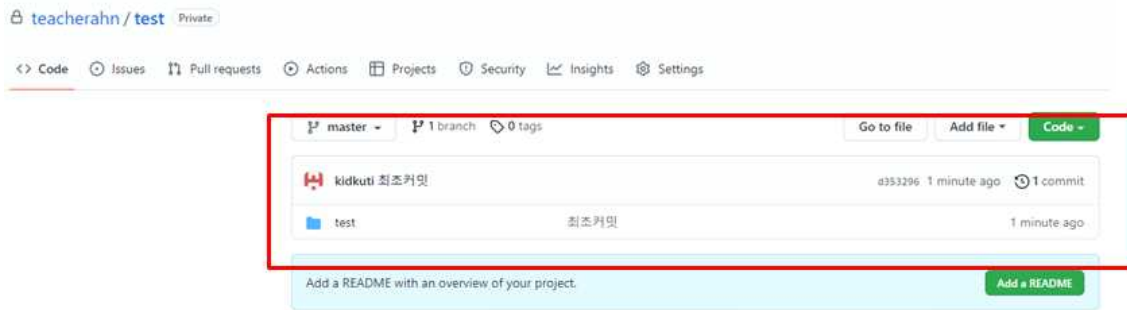
- Next를 클릭합니다.



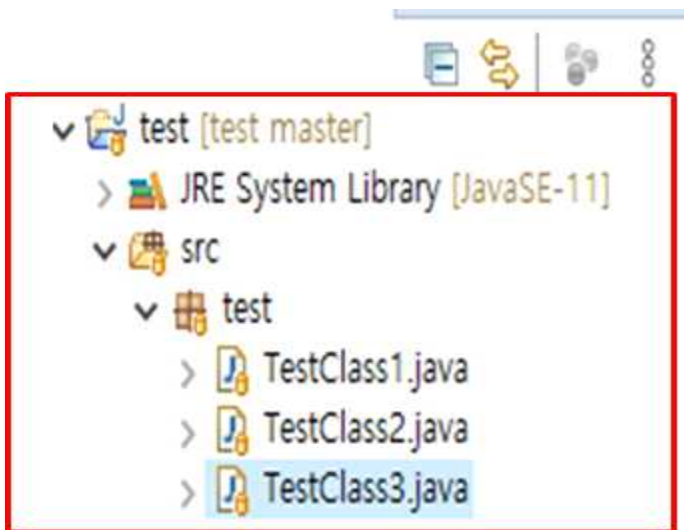
- Finish를 클릭합니다.



- Close를 클릭합니다.



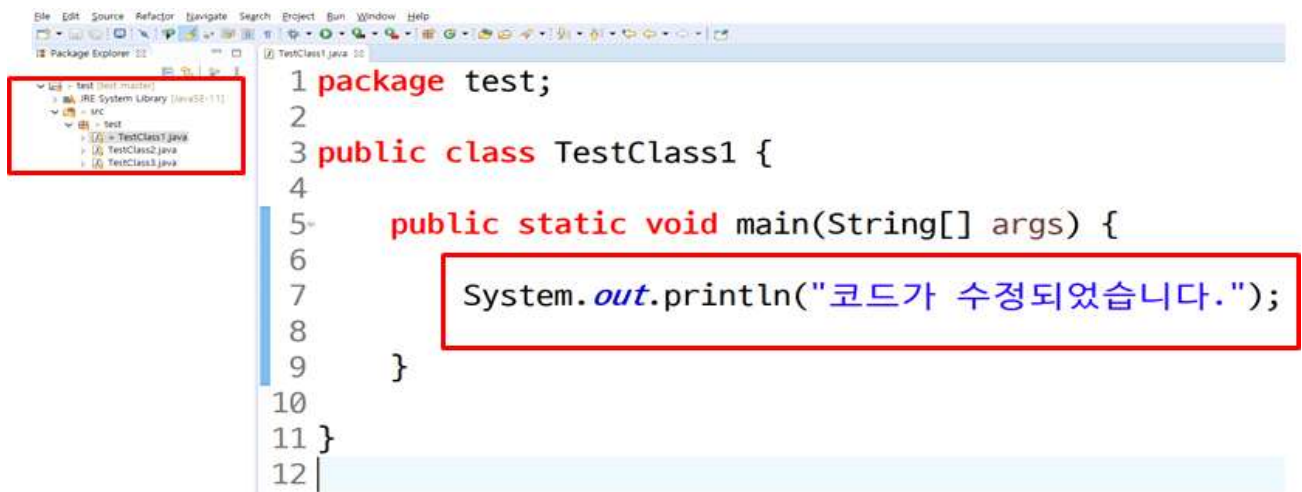
- 깃허브 사이트에서 정상적으로 Push가 된 모습을 확인합니다.



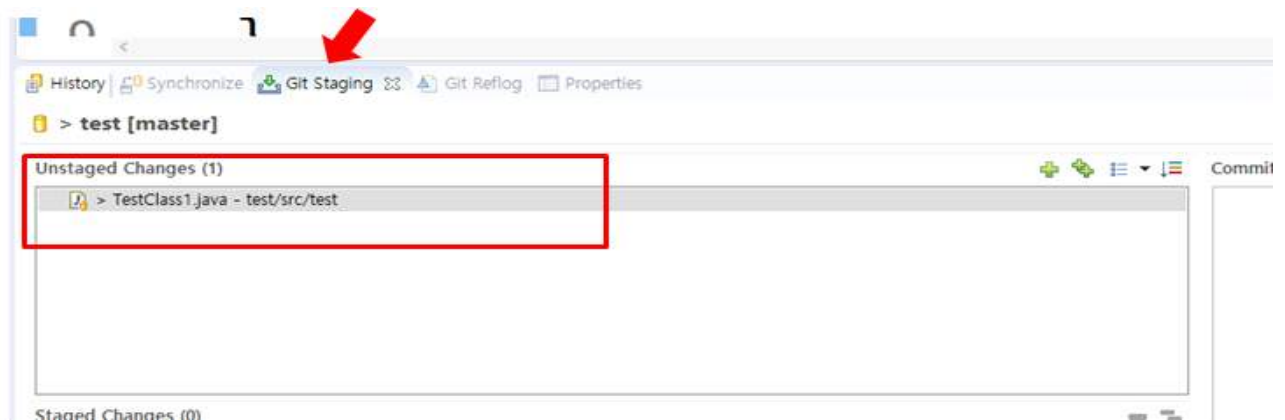
- 이클립스에서 정상적으로 Push가 된 모습을 확인합니다.

- 끝 -

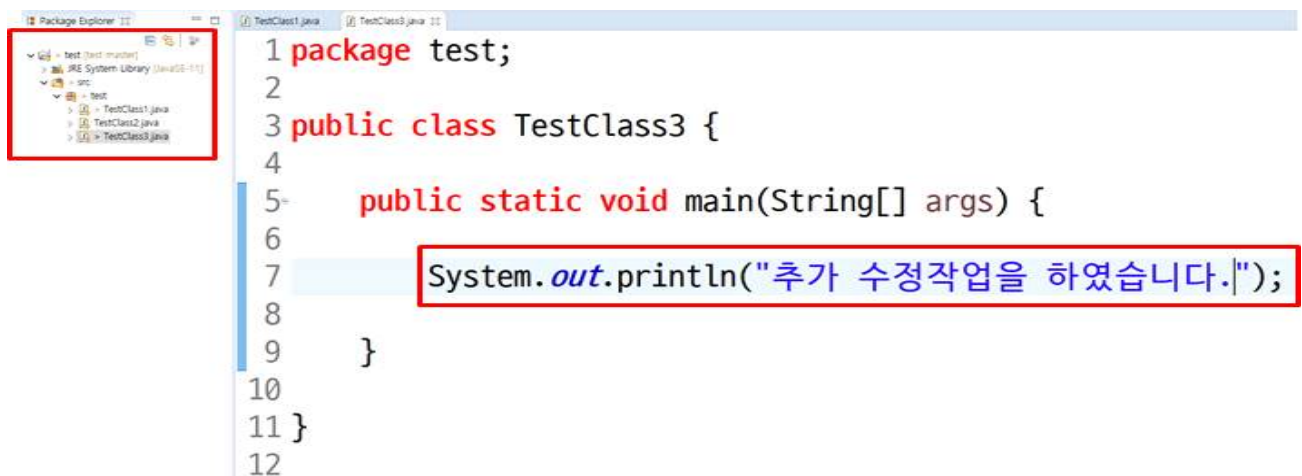
- * 수정된 소스를 깃허브에 업데이트(최신화) 하기

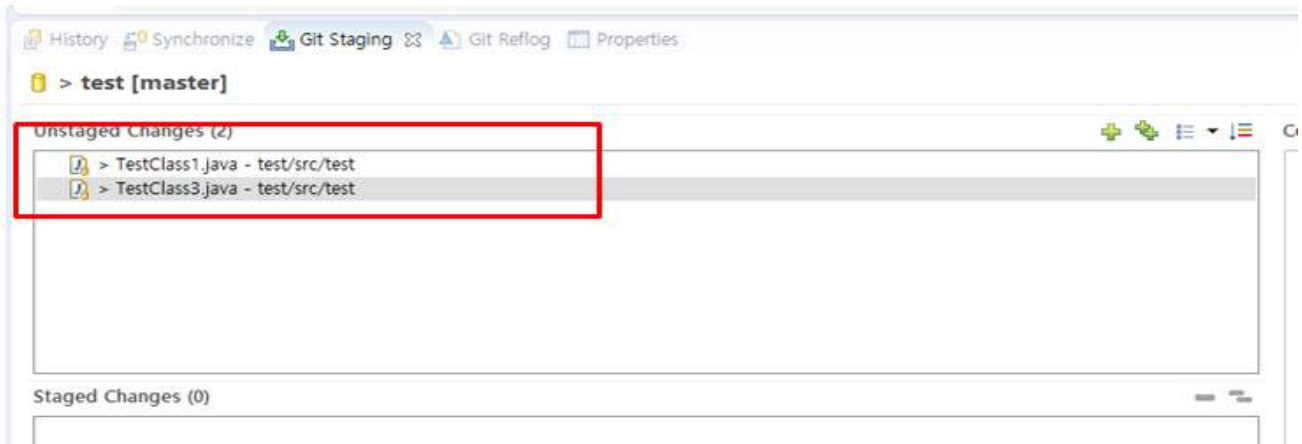


- 코드가 수정될 경우 Package Explorer에 > 표시를 확인할 수 있습니다.

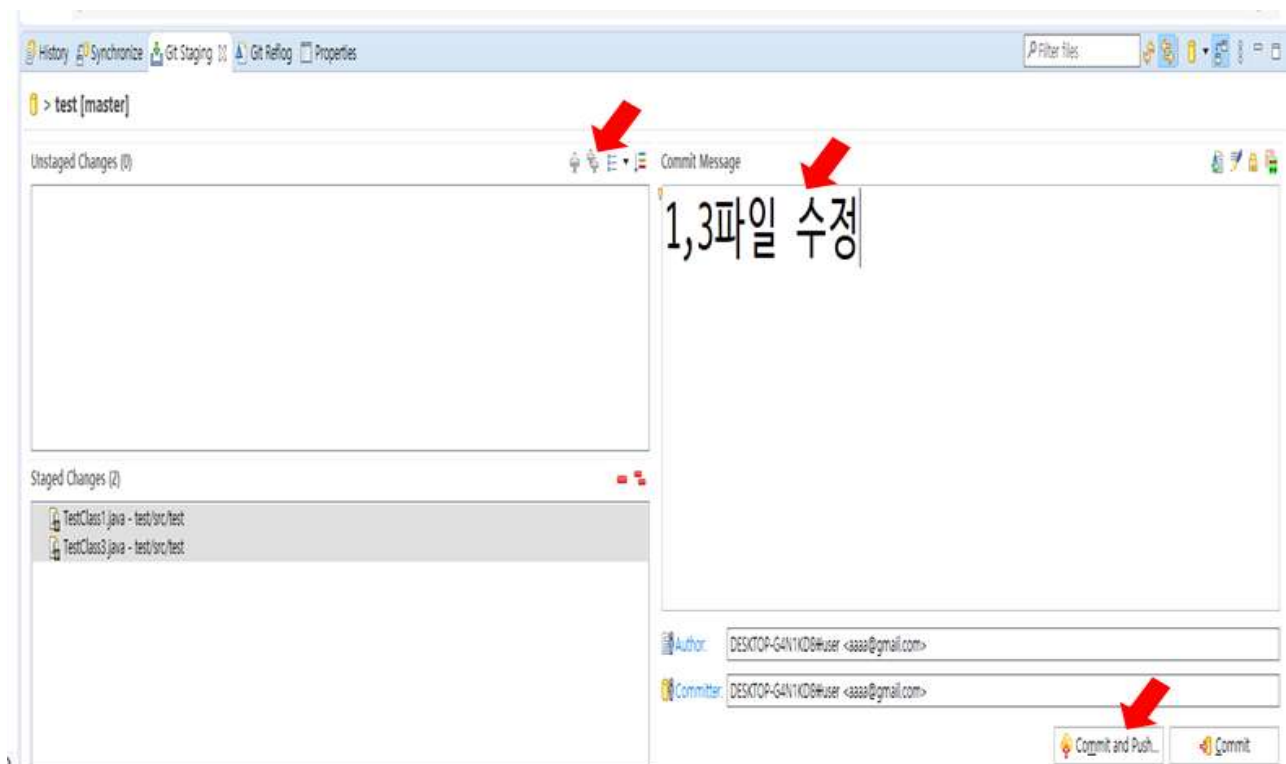


- Git Perspective의 Git Staging에서도 수정된 코드가 리스팅된 것을 확인할 수 있습니다.





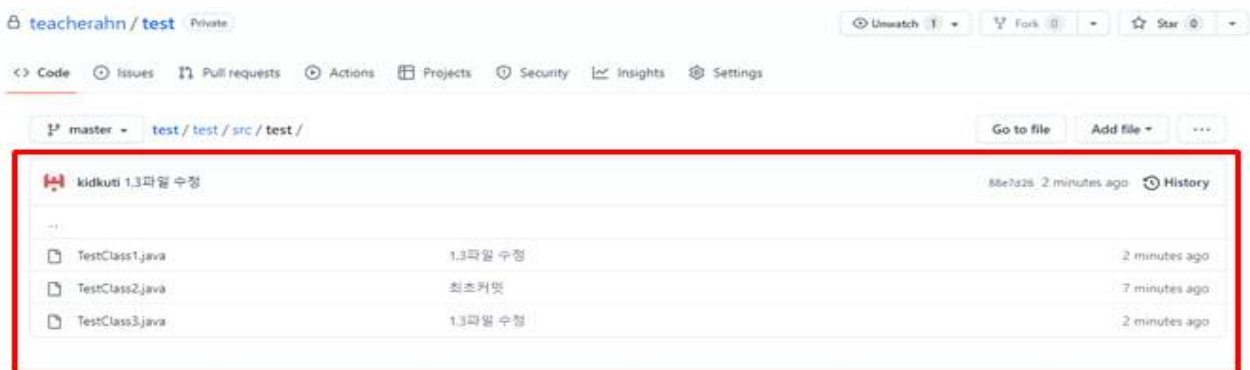
- Git Perspective의 Git Staging에서도 수정된 코드가 리스팅된 것을 확인할 수 있습니다.



- Unstaged의 리스트를 + 및 ++아이콘을 클릭하여 Stage리스트로 이동합니다.
- Commit Message를 작성합니다. (Commit Message는 옵션이아니라 필수값이어서 반드시 작성합니다.)
- Commit and Push버튼을 클릭합니다.



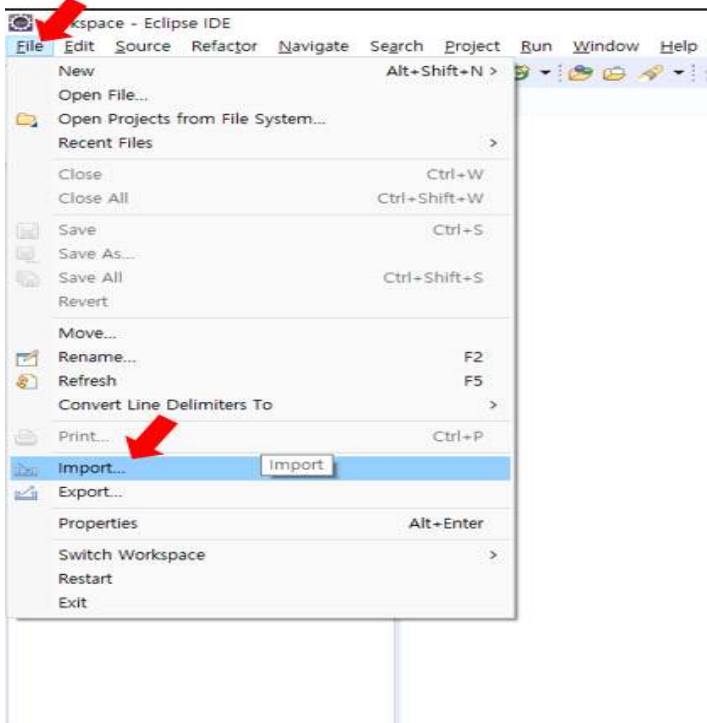
- Close버튼을 클릭합니다.



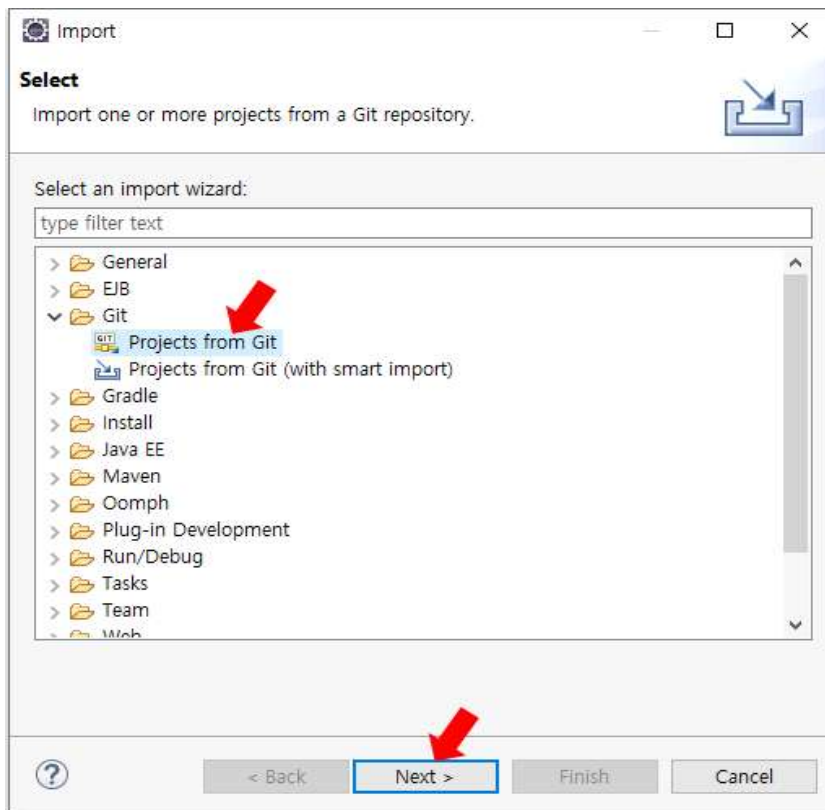
- 수정된 내역을 깃허브에서 확인합니다.

- 끝 -

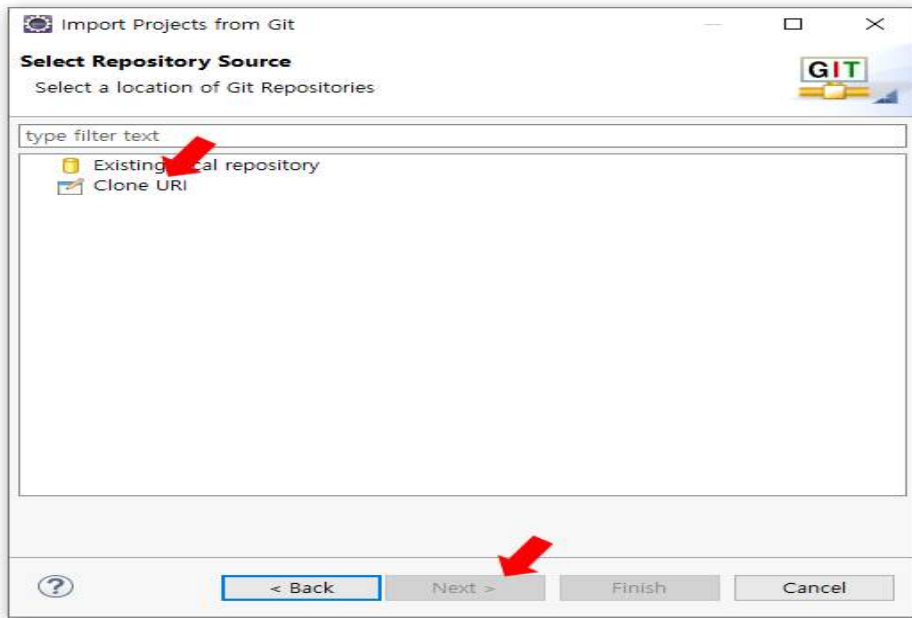
* 깃허브의 소스를 이클립스로 import하기



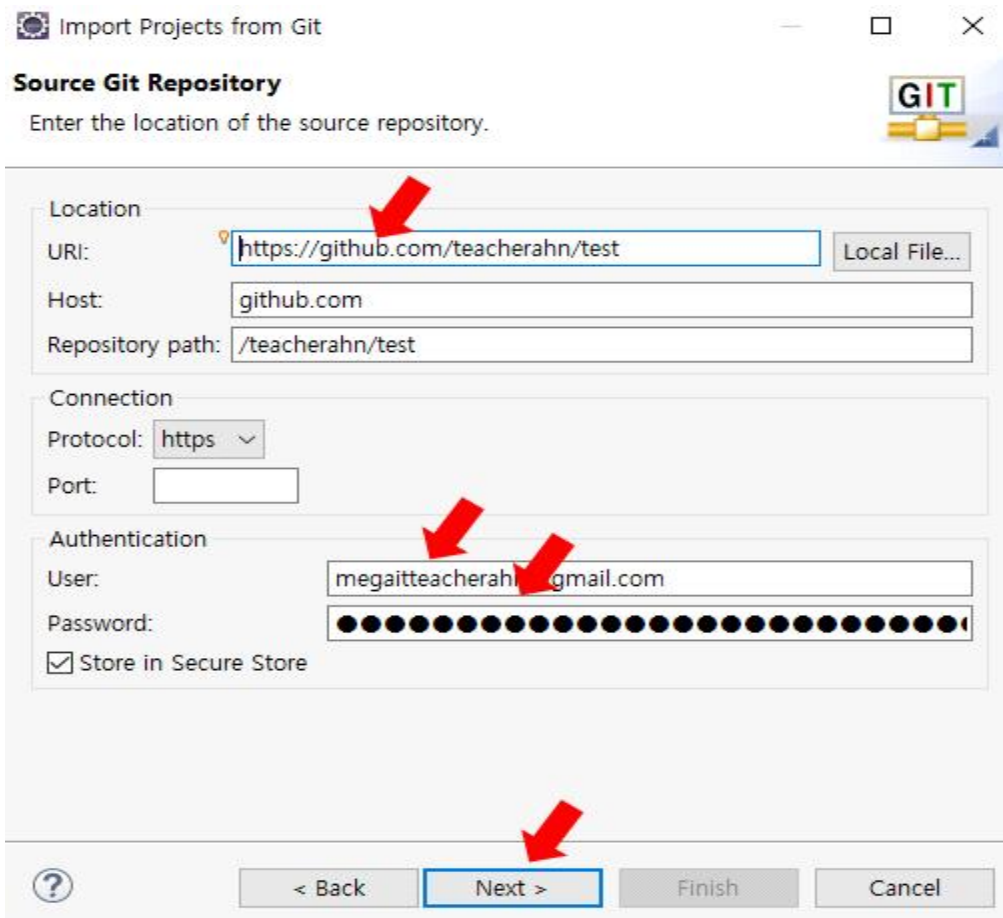
- 이클립스 좌측 상단에 File태그에서 Import태그를 클릭합니다.



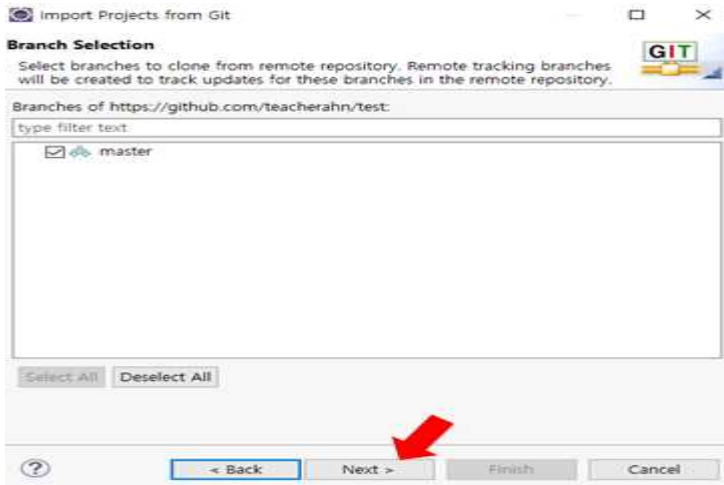
- Git 폴더에서 Projects from Git을 클릭한 뒤에 Next버튼을 클릭합니다.



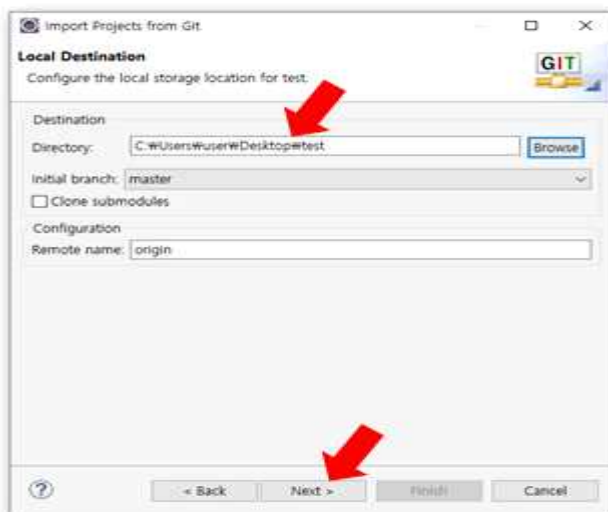
- Clone URI를 클릭 한 뒤에 Next버튼을 클릭합니다.



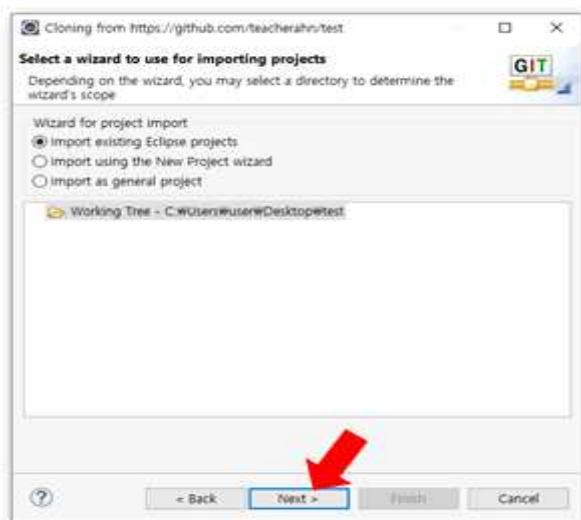
- 깃허브로부터 import하고 싶은 URL을 입력합니다. (깃허브 사이트에서 url을 복사해 옵니다.)
- User명과 Password를 입력합니다. (Store를 하였다면 User명과 Password를 입력하지 않아도 됩니다.)



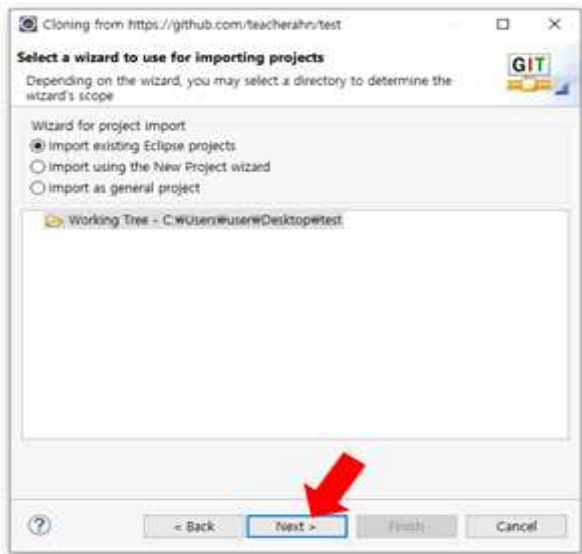
- Next버튼을 클릭합니다.



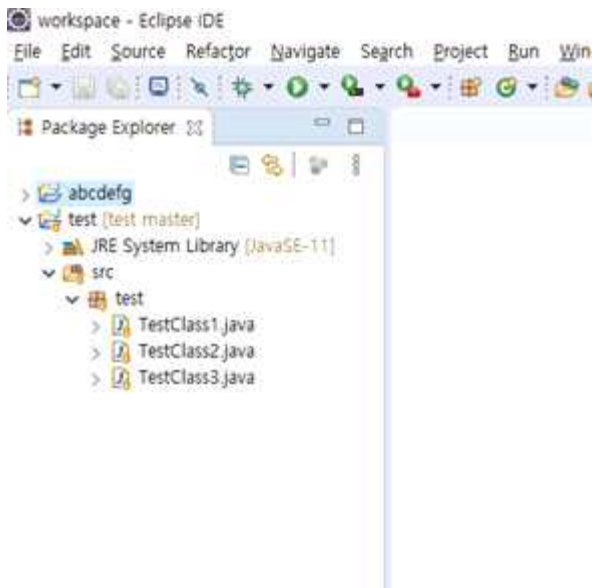
- 저장할 위치를 지정한 후에 Next를 클릭합니다.



- Next를 클릭합니다.



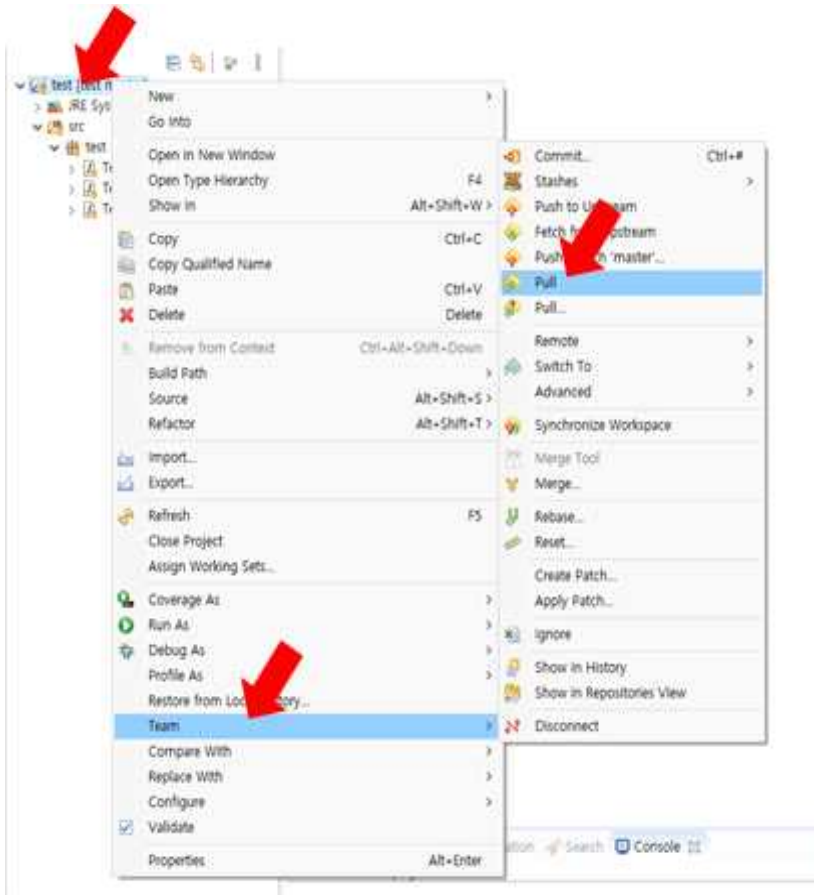
- Next를 클릭합니다.



- 성공적으로 Import된 화면을 확인합니다.

- 끝 -

* 깃허브의 최신소스를 이클립스에 업데이트(최신화) 하기

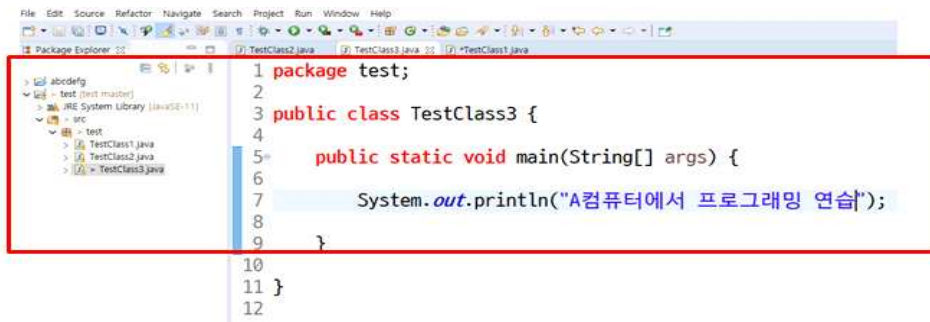


- 프로젝트를 우클릭 한뒤 Team의 Pull태그를 클릭합니다.



- Close버튼을 클릭합니다.

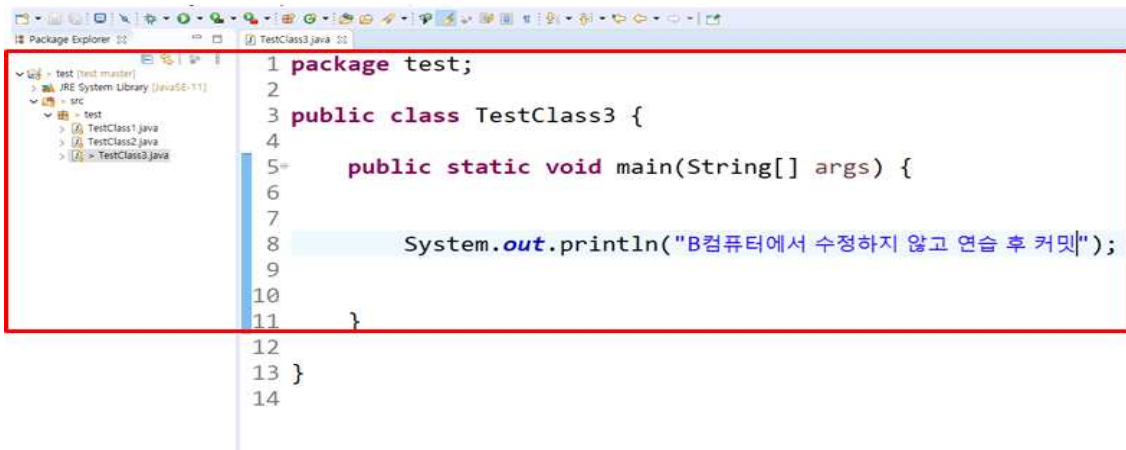
* 충돌(Conflict)이 발생했을 경우 해결 방법



```

1 package test;
2
3 public class TestClass3 {
4
5     public static void main(String[] args) {
6
7         System.out.println("A컴퓨터에서 프로그래밍 연습");
8
9     }
10
11 }
12

```

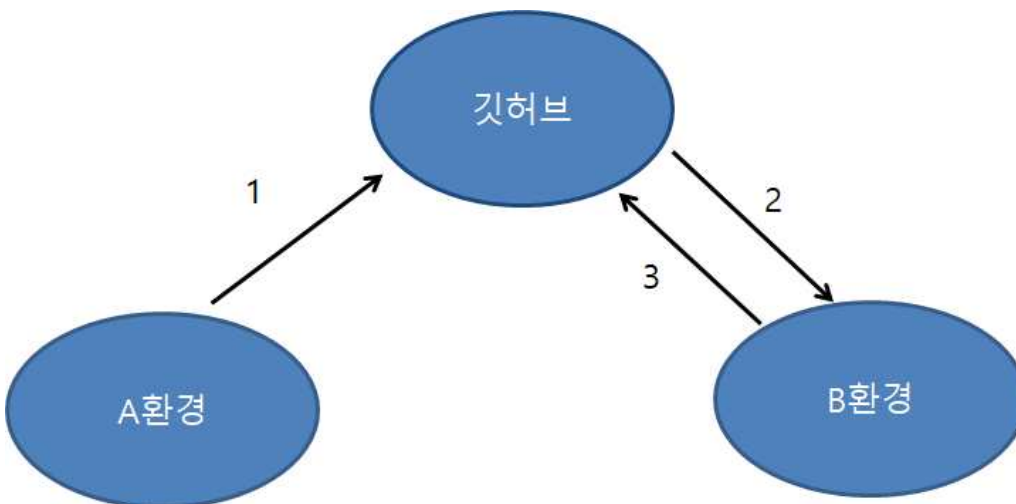


```

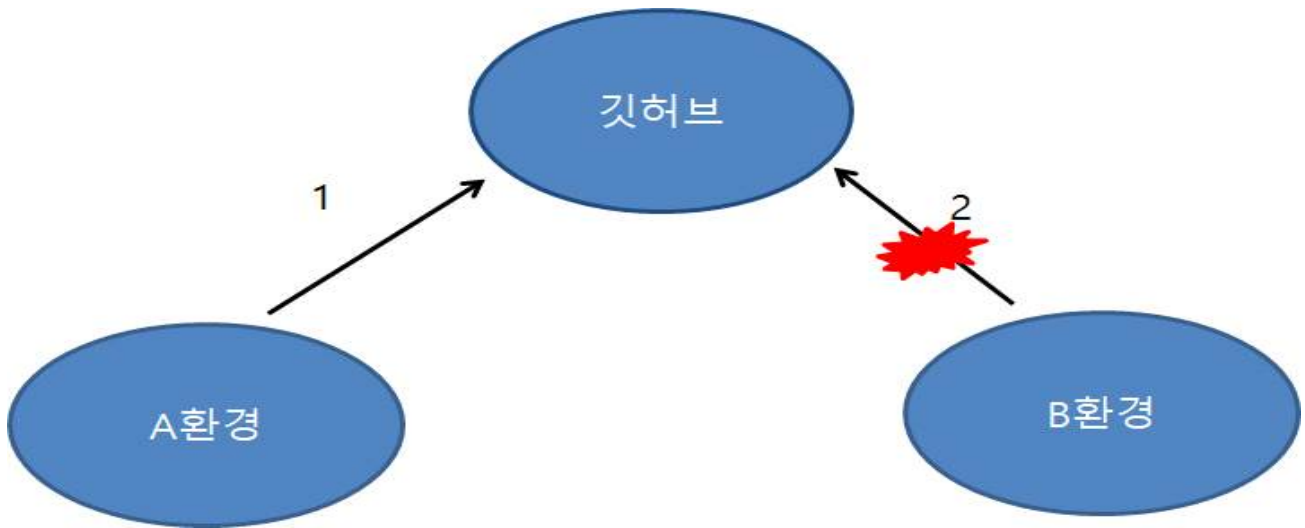
1 package test;
2
3 public class TestClass3 {
4
5     public static void main(String[] args) {
6
7         System.out.println("B컴퓨터에서 수정하지 않고 연습 후 커밋");
8
9     }
10
11 }
12
13 }
14

```

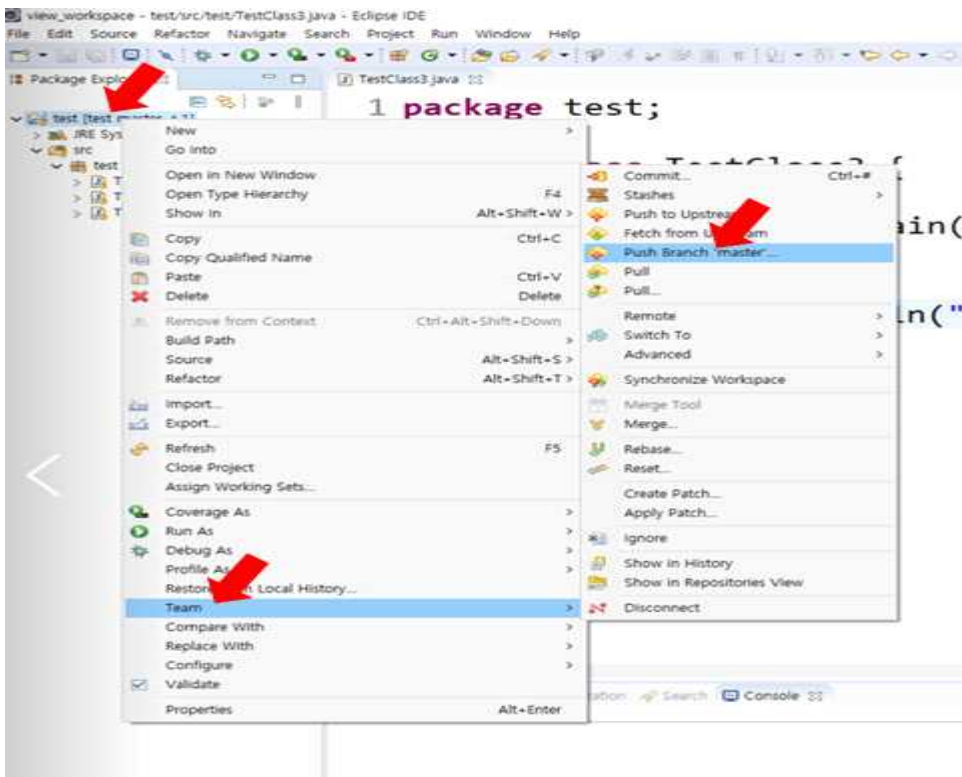
- 충돌(Conflict)이란 깃허브의 최신상태 소스를 pull하지 않은 상태로 새로운 버전으로 깃허브에 push를 시도할 때 발생하는 오류입니다.



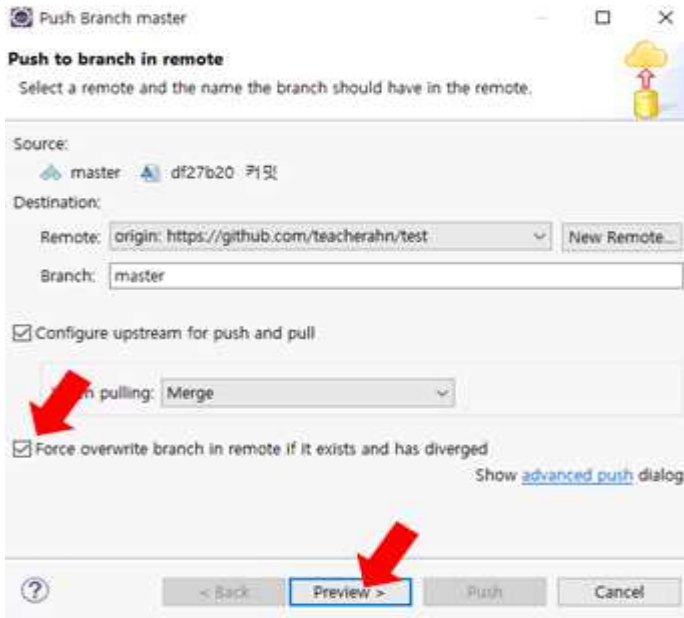
- 위의 그림과 같이 A 환경에서 Push한 최신소스를 다른 환경에서 먼저 Pull을 한 다음 Push를 작업을 해야 최신상태가 유지되는데



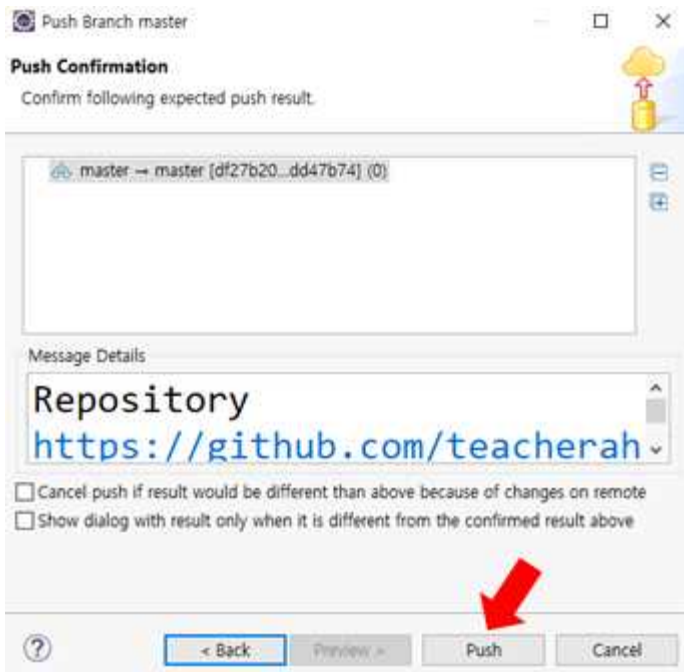
- 위 그림과 같이 A환경에서 Push한 코드로 인하여 깃허브는 A환경의 코드로 최신화가 되어있는데 B환경에서 새로운 코드를 Push할 경우 충돌이 발생합니다.



- 충돌을 해결할 경우 최신으로 업로드하고 싶은 프로젝트에서 우클릭 Team의 Push Branch master를 클릭합니다.



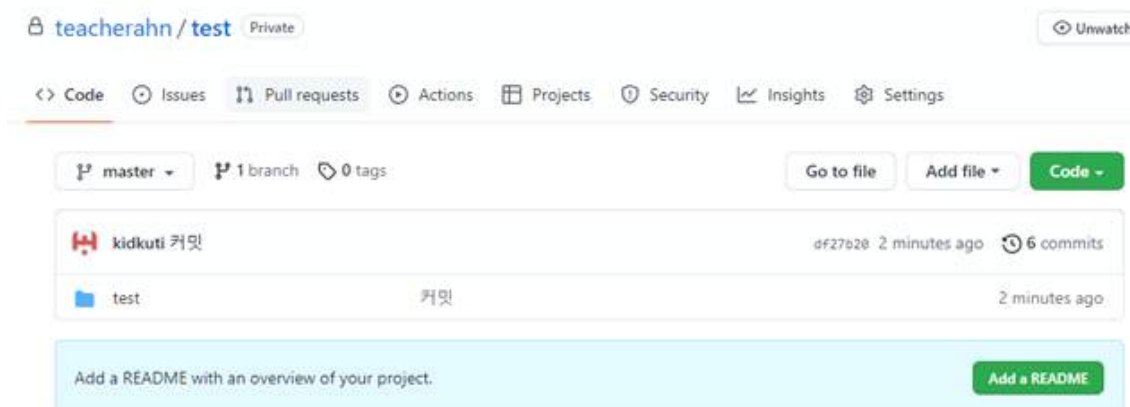
- Force overwrite branch in remote if it exists and has diverge의 체크박스를 클릭하고 Preview 버튼을 클릭합니다.



- Push 버튼을 클릭합니다.



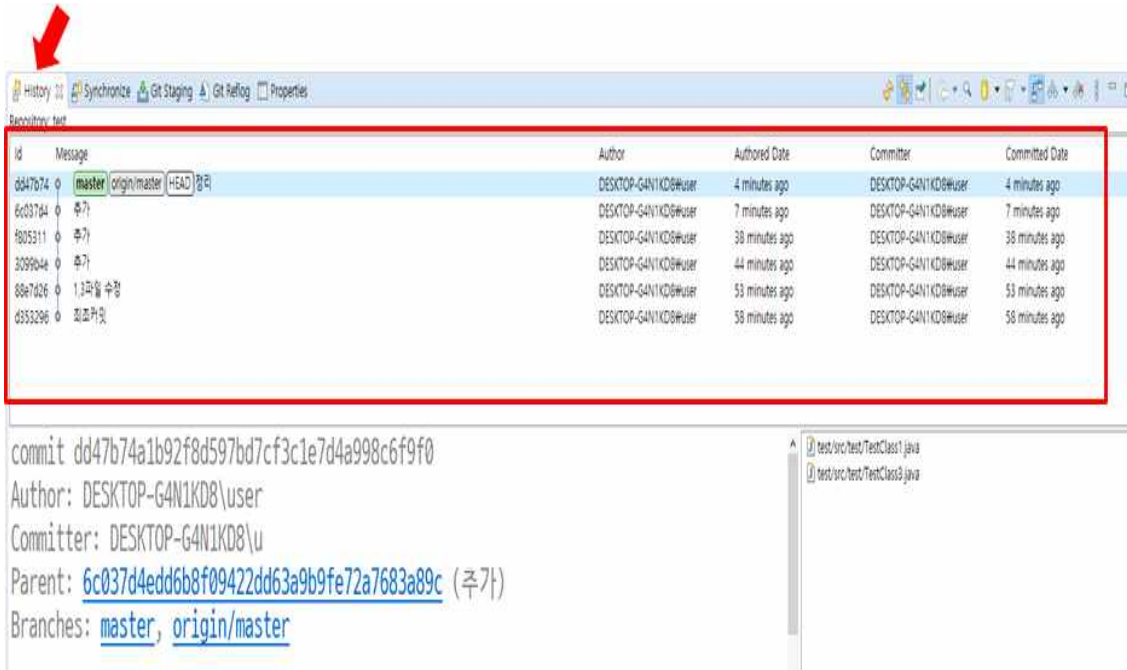
- Close 버튼을 클릭합니다.



- 충돌 에러가 더 이상 발생하지 않고 깃허브에서 최신화된 소스를 확인합니다.

- 끝 -

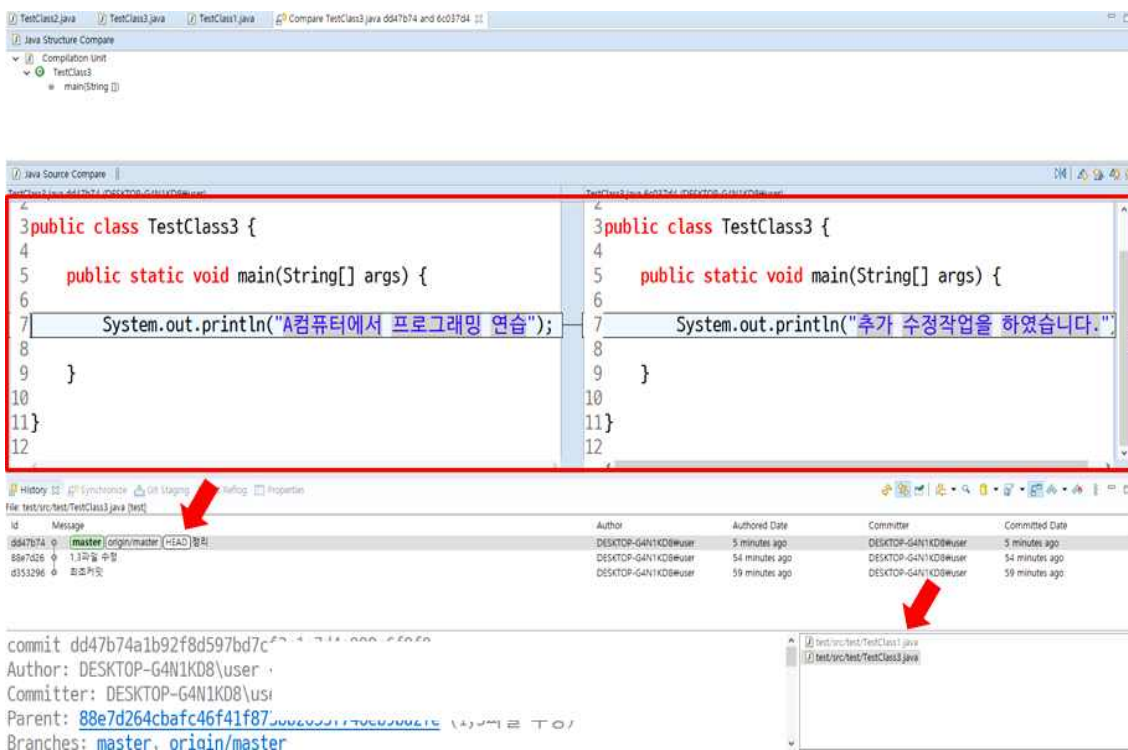
* 깃허브 히스토리 보기



Id	Message	Author	Authored Date	Committer	Committed Date
dd47b74	master origin/master HEAD 정리	DESKTOP-G4N1KD8\user	4 minutes ago	DESKTOP-G4N1KD8\user	4 minutes ago
6c037d4	추가	DESKTOP-G4N1KD8\user	7 minutes ago	DESKTOP-G4N1KD8\user	7 minutes ago
1805311	추가	DESKTOP-G4N1KD8\user	38 minutes ago	DESKTOP-G4N1KD8\user	38 minutes ago
3099b4e	추가	DESKTOP-G4N1KD8\user	44 minutes ago	DESKTOP-G4N1KD8\user	44 minutes ago
88e7d26	1.3파일을 수정	DESKTOP-G4N1KD8\user	53 minutes ago	DESKTOP-G4N1KD8\user	53 minutes ago
d353296	최초 커밋	DESKTOP-G4N1KD8\user	58 minutes ago	DESKTOP-G4N1KD8\user	58 minutes ago

commit dd47b74a1b92f8d597bd7cf3c1e7d4a998c6f9f0
 Author: DESKTOP-G4N1KD8\user
 Committer: DESKTOP-G4N1KD8\user
 Parent: 6c037d4edd6b8f09422dd63a9b9fe72a7683a89c (추가)
 Branches: master, origin/master

- Perspective Git에서 history태그를 클릭하면 Push이력을 볼수 있습니다.



```

3 public class TestClass3 {
4
5     public static void main(String[] args) {
6
7         System.out.println("A컴퓨터에서 프로그래밍 연습");
8     }
9 }
10
11 }
12
  
```

```

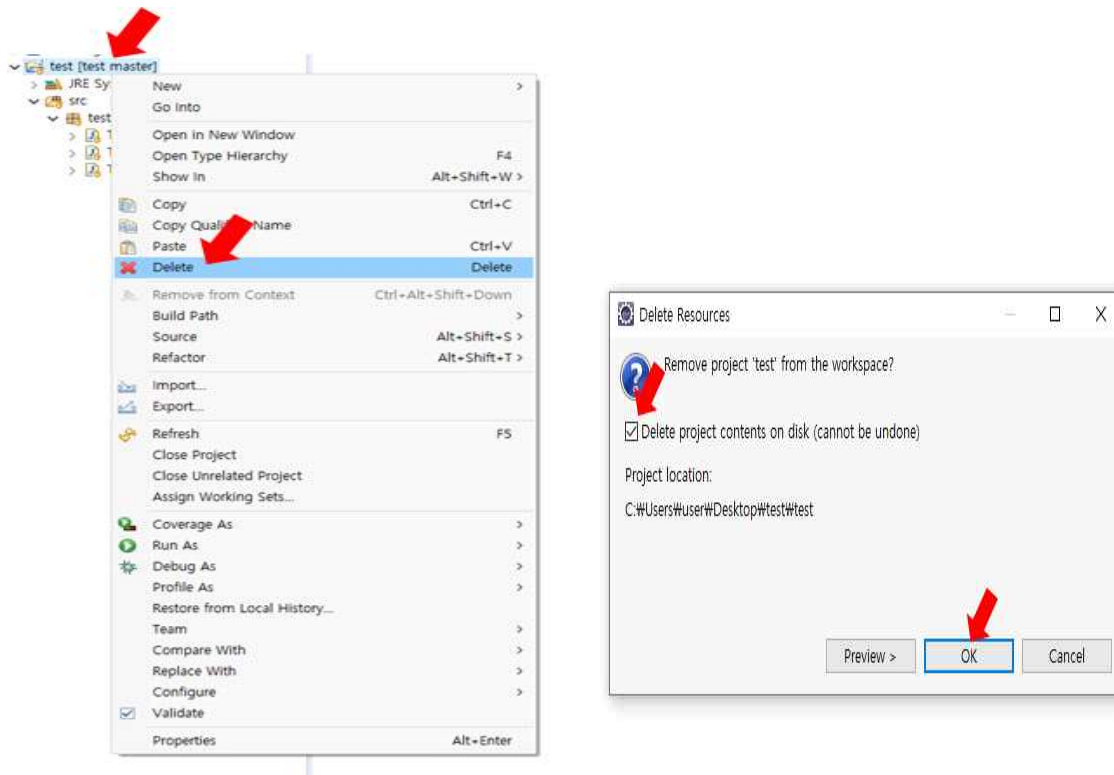
3 public class TestClass3 {
4
5     public static void main(String[] args) {
6
7         System.out.println("추가 수정작업을 하였습니다.");
8     }
9 }
10
11 }
12
  
```

commit dd47b74a1b92f8d597bd7c...
 Author: DESKTOP-G4N1KD8\user
 Committer: DESKTOP-G4N1KD8\user
 Parent: 88e7d264cbafc46f41f87...
 Branches: master, origin/master

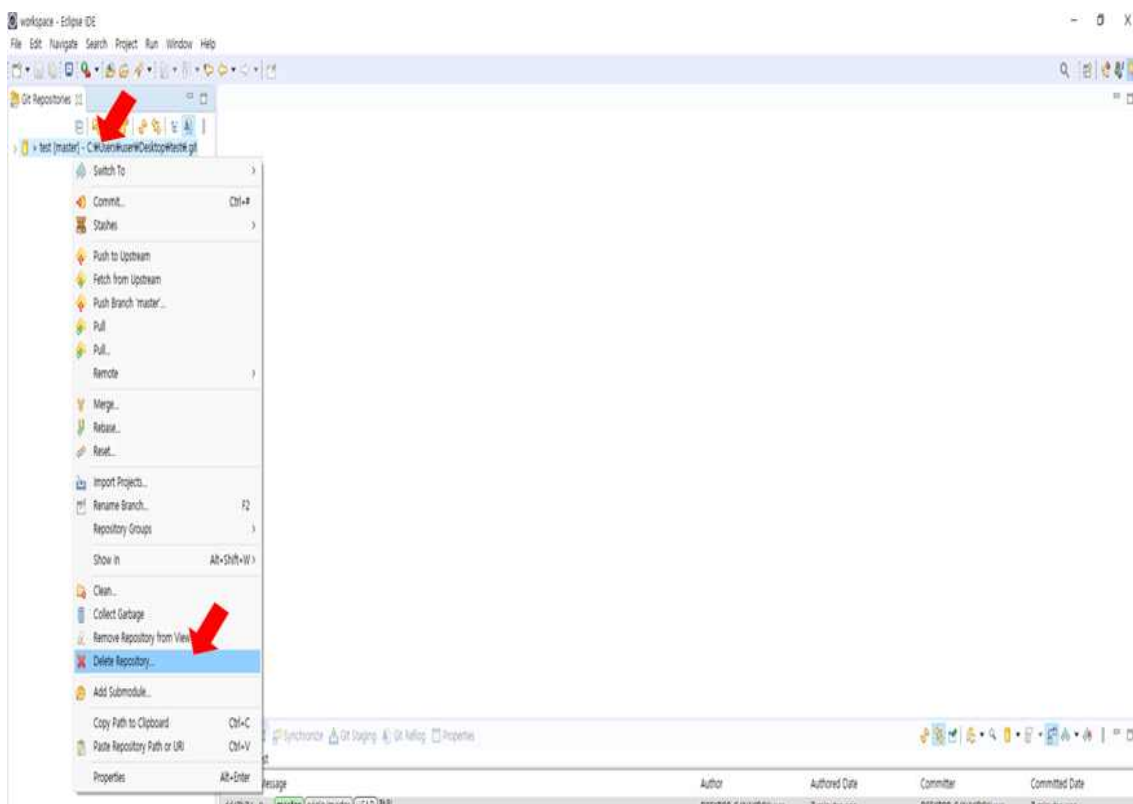
- 아래와 같이 특정 파일을 클릭하면 파일이 수정되기 전후의 모습을 확인할 수 있습니다.

- 끝 -

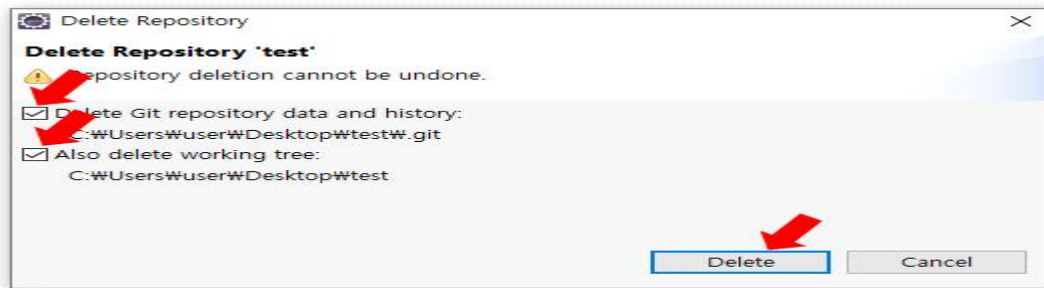
* 저장소 삭제하기



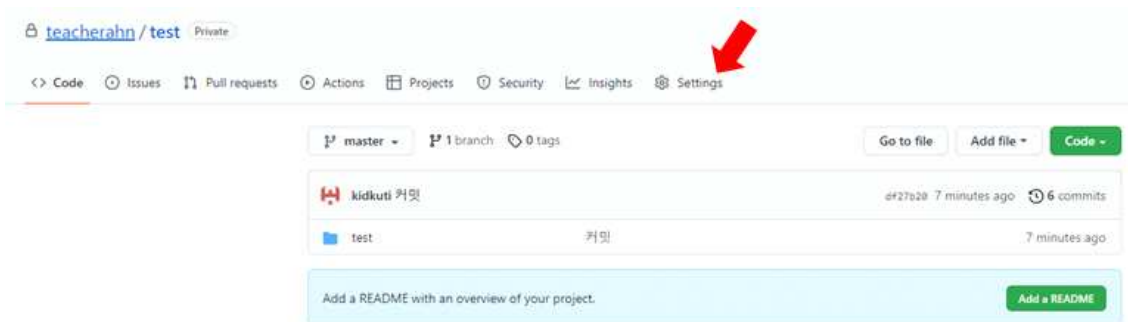
- 첫째로 프로젝트 소스를 우클릭하여 삭제합니다.



- 둘째로 Git Perspective에서 삭제하고 싶은 Repository를 클릭한 뒤에 Delete Repository를 클릭합니다.



- 모든 체크박스를 클릭해주시고 Delete버튼을 누릅니다.



- 마지막으로 깃허브 저장소를 삭제하기 위해서 깃허브 프로젝트에서 Settings태그를 클릭합니다.



- Delete this Repository를 클릭한뒤에 확인 메시지를 입력한후 삭제를 진행합니다.