

## 1. GitHub에 프로젝트 올리기

1) github에 로그인해서 저장소 생성한다.

2) 내 pc 폴더에 github 저장소 주소 알려주기.

```
git remote add origin https://github.com/아이디/이름.git
```

3) 만든 커밋 push 하기

```
git push origin master
```

- 위 명령어에서 "master" 는 메인브랜치의 이름으로 기본브랜명을 main 으로 변경한 경우는 **git push origin main** 이다.

### github 사용을 위한 인증과 환경설정

#### 인증방식

##### 1) OAuth(제 3 자인증)방식

- oauth 방식은 간편한데 반해, 권한제한설정 하는데에는 한계 존재
- intelliJ 등 개발 tool 을 연결
  - IntelliJ IDEA 의 Get From VCS 를 통해 Authorize in GitHub
  - settings 의 applications 에서 확인
- git credential 을 통한 웹 로그인

##### 2) Personal Access Token(PAT)을 생성방식

- developer settings 에서 personal access token 중 class tokens 생성
- 윈도우에 Github 자격증명 토큰 설정
  - 제어판 - 사용자 계정 - windows 자격 증명 관리자 탭으로 이동
  - 일반 자격 증명 섹션으로 스크롤하고 새로운 자격 증명 추가를 클릭
  - 주소 필드에 git:<https://github.com> 입력
  - github 이름과 token 값 저장

- mac github 자격증명 토큰 설정
  - 키체인 접근
  - 상단에서 github 를 검색
  - github 이름과 token 값 저장

## Github 사이트에서 올라간 커밋 확인하기

### 실습해보기

github에 로그인해서 저장소 생성한다

**Create a new repository**

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

Required fields are marked with an asterisk (\*).

Owner \*      Repository name \*

/   
 GitTest\_03 is available.

Great repository names are short and memorable. Need inspiration? How about [probable-fortnight](#) ?

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:

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

Add .gitignore

.gitignore template:

The screenshot shows a GitHub repository page for 'GitTest\_03'. At the top right, there are buttons for Pin, Unwatch (1), Fork (0), and Star (0). Below the header, there are sections for 'Set up GitHub Copilot' and 'Add collaborators to this repository'. A 'Quick setup' section provides instructions for creating a new repository or pushing an existing one from the command line, along with corresponding Git commands.

**Set up GitHub Copilot**  
Use GitHub's AI pair programmer to autocomplete suggestions as you code.  
[Get started with GitHub Copilot](#)

**Add collaborators to this repository**  
Search for people using their GitHub username or email address.  
[Invite collaborators](#)

**Quick setup — if you've done this kind of thing before**

[ Set up in Desktop] or [HTTPS] [SSH] [https://github.com/8253jang/GitTest\\_03.git](https://github.com/8253jang/GitTest_03.git)

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

**...or create a new repository on the command line**

```
echo "# GitTest_03" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/8253jang/GitTest_03.git
git push -u origin main
```

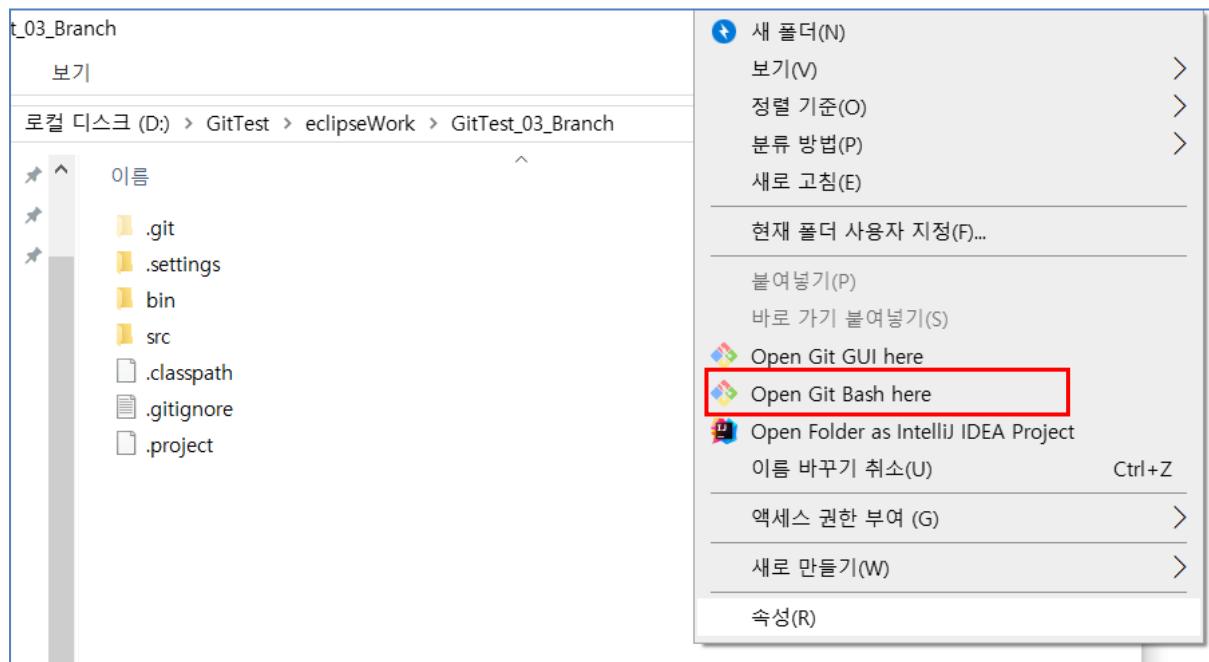
**...or push an existing repository from the command line**

```
git remote add origin https://github.com/8253jang/GitTest_03.git
git branch -M main
git push -u origin main
```

**...or import code from another repository**  
You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

[Import code](#)

## gitbash를 연다.



```
KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipseWork/GitTest_03_Branch (main)
$ git remote add origin https://github.com/8253jang/GitTest_03.git

KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipseWork/GitTest_03_Branch (main)
$ git push origin main
Enumerating objects: 84, done.
Counting objects: 100% (84/84), done.
Delta compression using up to 4 threads
Compressing objects: 100% (54/54), done.
Writing objects: 100% (84/84), 6.29 KiB | 536.00 KiB/s, done.
Total 84 (delta 13), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (13/13), done.
To https://github.com/8253jang/GitTest_03.git
 * [new branch]      main -> main

KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipseWork/GitTest_03_Branch (main)
$ |
```

8253jang / GitTest\_03

Issues Pull requests Actions Projects Wiki Security Insights Settings

GitTest\_03 (Public)

main 1 Branch 0 Tags Go to file Add file Code

8253jang Merge branch 'Cart' c0a9855 · yesterday 14 Commits

src/com Merge branch 'Cart' yesterday

.gitignore first commit yesterday

README

GitTest\_03\_Branch New tab +

커밋 Pull Push 패치 브랜치 병합 스테시 폐기 태그

WORKSPACE 파일 상태 History Search

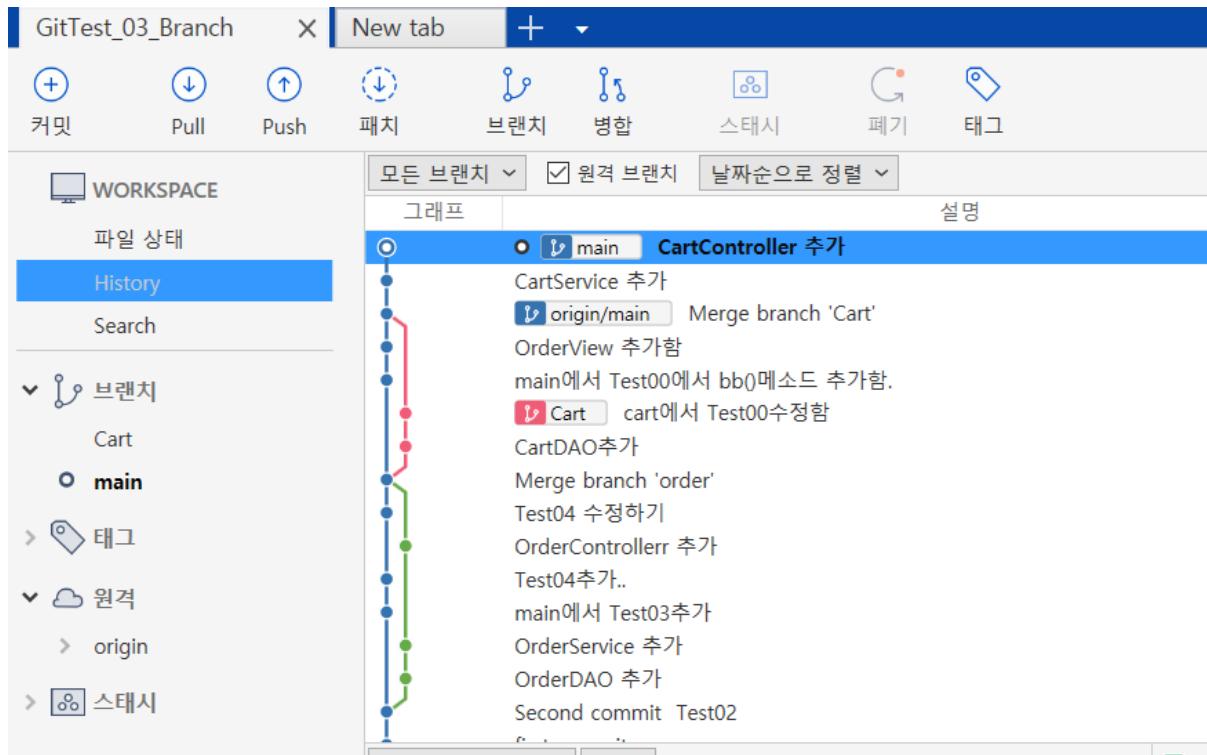
모든 브랜치 원격 브랜치 날짜순으로 정렬

그래프 설명

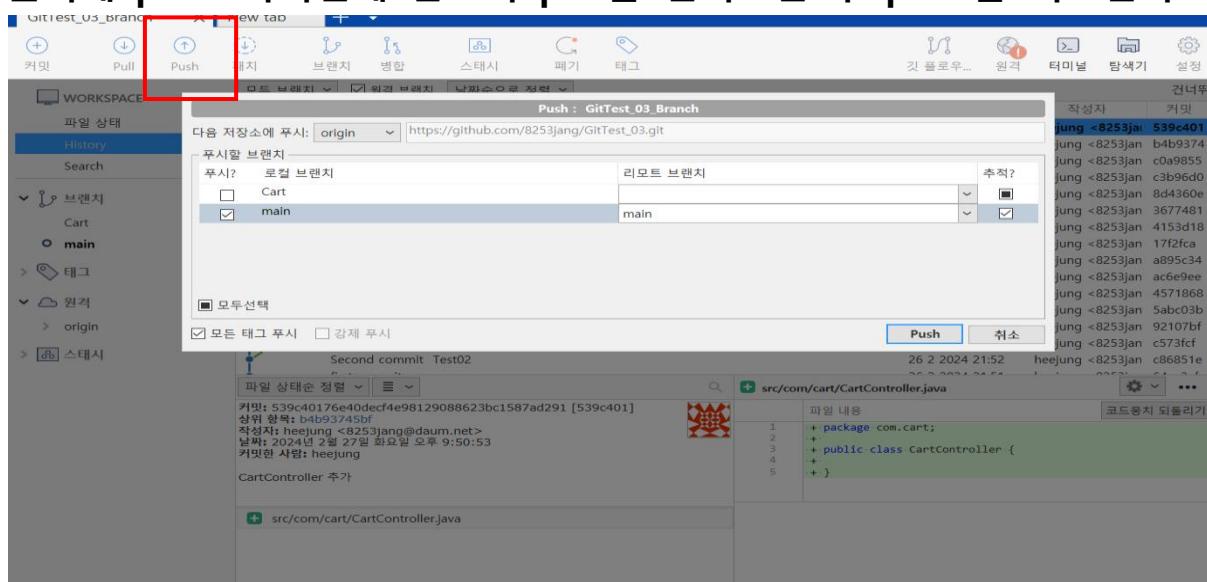
● main origin/main Merge branch 'Cart'

- OrderView 추가함
- main에서 Test00에서 bb0메소드 추가함.
- Cart cart에서 Test00수정함
- CartDAO추가
- Merge branch 'order'
- Test04 수정하기
- OrderController 추가
- Test04추가..
- main에서 Test03추가
- OrderService 추가
- OrderDAO 추가
- Second commit Test02
- first commit

Local 저장소에 2번의 commit을 만들고  
remote저장소에 push를 해본다.



원격에 push하기 전에 반드시 pull을 먼저 한 후 push를 시도한다.



## git pull vs git fetch 차이

명령어	설명	동작방식
git fetch	원격 저장소에서 최신 변경 사항을 가져옴	로컬 브랜치를 변경하지 않고 원격의 최신 상태만 업데이트
git pull	원격 저장소에서 변경 사항을 가져오고, 자동으로 병합	git fetch + git merge 실행 (로컬 브랜치가 원격 변경 사항과 병합됨)

**git fetch 후에 병합할 때**

**git merge origin/main**

**git pull origin main**

## 언제 구분해서 사용하면 좋을까?

**git fetch** - 원격 저장소에 어떤 변경 사항이 있는지 미리 확인하고 싶을 때.

**git pull**- 원격 변경 사항을 즉시 로컬 브랜치에 반영할 때.

## git diff

파일의 변경 내용을 비교하는 명령어

명령어	설명
git diff	수정했지만 아직 git add하지 않은 변경 사항을 보여줌
git diff --staged	git add한 후 커밋 전 변경 사항을 보여줌
git diff HEAD	현재 작업 중인 변경 사항과 마지막 커밋의 차이를 보여줌
git diff 브랜치명	현재 브랜치와 지정된 브랜치의 차이를 비교
git diff 커밋ID1 커밋ID2	두 커밋 간의 변경 사항을 비교

## local저장소 push 와 remote 저장소 pull 해보기

### 시나리오 01

local 저장소에 수정 사항은 있으나 commit을 하기전 인 상태에서  
remote 저장소와 local 저장소의 commit 이력이 같은 상황인 경우

**git pull origin main** 명령어 실행하면

```

MINGW64:/d/GitTest/eclipseWork/Git_295 (main)
$ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   src/exam/Test01.java

no changes added to commit (use "git add" and/or "git commit -a")

KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipseWork/Git_295 (main)
$ git pull origin main
From https://github.com/8253jang/Git_295
 * branch            main      -> FETCH_HEAD
Already up to date.

KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipseWork/Git_295 (main)
$ 

```

Remote 저장소에 와 현재 local 저장소 커밋이력이 같기 때문에 받아 올 변경 사항이 없다.

## 시나리오 02

Local 저장소에 변경 이력이 없고 remote 저장소에 커밋 이력이 더 생긴 경우

**git pull origin main** 명령어 실행하면

```

MINGW64:/d/GitTest/eclipseWork/Git_295 (main)
$ git pull origin main
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (5/5), 1.11 KiB | 6.00 KiB/s, done.
From https://github.com/8253jang/Git_295
 * branch            main      -> FETCH_HEAD
   54dalad..03faf0a main      -> origin/main
Updating 54dalad..03faf0a
Fast-forward
 src/exam/test02.java | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)

KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipseWork/Git_295 (main)
$ 

```

충돌없이 local에 커밋 이력이 내려오고 소스가 합쳐진다.

이런 상황을 **fast-forward** 이라고 한다.

## 시나리오 03

Local 저장소에 변경 이력이 있고 remote 저장소에도 커밋 이력이 있지만 서로 다른 파일의 수정인 경우

먼저, 현재 git log를 확인해보자

```
KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipsework/Git_295 (main)
$ git log --oneline
94cd6c4 (HEAD -> main) 로컬에서 다른 파일 수정함
4f65b22 (origin/main) 원격에서 수정함
03faf0a remote Ttest02 bb메소드 수정
54dalad Test02 추가
9e7923d first commit
```

**git pull origin main** 명령어 실행해보자.

```
KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipsework/Git_295 (main)
$ git pull origin main
\remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 5 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (5/5), 1.04 KiB | 2.00 KiB/s, done.
From https://github.com/8253jang/Git_295
 * branch           main      -> FETCH_HEAD
   1f65b22..d80f66e main      -> origin/main
Merge made by the 'ort' strategy.
src/exam/Test02.java | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

그러면, 원격의 커밋이력과 로컬의 커밋 이력을 하나로 **합치는(merge) 커밋이력이** 하나 더 생긴다.

```
KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipsework/Git_295 (main)
$ git log --oneline
f4f44f5 (HEAD -> main) Merge branch 'main' of https://github.com/8253jang/Git_295
94cd6c4 로컬에서 다른 파일 수정함
d80f66e (origin/main) 원격에서 수정함
4f65b22 원격에서 수정함
03faf0a remote Ttest02 bb메소드 수정
54dalad Test02 추가
9e7923d first commit
```

로컬의 커밋이력은 pull하기전 5개였고 pull을 하면 원격에서 발생했던 커밋이력 1개와 Merge를 진행한 커밋이력 1개를 합쳐 총 7개의 커밋이 생긴 것을 확인한다.

현재 로컬의 커밋이력이 원격의 커밋 이력보다 2개 커밋이력이 앞서 있다.

로컬 커밋 이력을 원격에 push 해본다.

```
KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipsework/Git_295 (main)
$ git push origin main
Enumerating objects: 17, done.
Counting objects: 100% (16/16), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (9/9), 884 bytes | 884.00 KiB/s, done.
Total 9 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/8253jang/Git_295.git
  d80f66e..f4f44f5  main -> main
```

## 시나리오 04

Local 저장소에 변경 이력이 있고 remote 저장소에도 커밋 이력이 있으면서

서로 같은 파일의 수정인 경우 **-충돌발생한다!!!**

이 경우는 자동으로 merge를 할수 없고 개발자가 직접 소스를 합쳐서 merge커밋 이력을 만들어야 한다.

```
KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipsework/Git_295 (main)
$ git pull origin main
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 5 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (5/5), 1.10 KiB | 6.00 KiB/s, done.
From https://github.com/8253jang/Git_295
 * branch            main      -> FETCH_HEAD
   f4f44f5..596dee6  main      -> origin/main
AUTO-Merging src/exam/Test01.java
CONFLICT (content): Merge conflict in src/exam/Test01.java
Automatic merge failed; fix conflicts and then commit the result.

KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipsework/Git_295 (main|MERGING)
```

```

3 public class Test01 {
4     public void aa() {
5         System.out.println("로컬에서 수정함");
6     }
7 }
8 public void bb() {
9     <<<<< HEAD
10    System.out.println("로컬에서 Test01 bb메소드 수정함");
11    =====
12    System.out.println("원격에서 수정함");
13    >>>>> 596dee660b4b1b779fe0de2a2de6a6cf7e2ee81b
14 }
15 }
16

```

위 화면처럼 conflict 발생하고 소스를 확인해보면 로컬과 원격을 표시 해주고 있다.

개발자가 직접 소스를 수정하고 다시 커밋을 한다.

```

KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipseWork/Git_295 (main|MERGING)
$ git add .

KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipseWork/Git_295 (main|MERGING)
$ git commit -m "충돌 해결 후 커밋"
[main 75b16a2] 충돌 해결 후 커밋

KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipseWork/Git_295 (main)
$ git log --oneline
75b16a2 (HEAD -> main) 충돌 해결 후 커밋
8e06b6e Test01 bb 메소드 로컬에서 수정함
596dee6 (origin/main) 원격에서 Test02 bb메소드 수정함
f4f44f5 Merge branch 'main' of https://github.com/8253jang/Git_295
94cd6c4 로컬에서 다른 파일 수정함
d80f66e 원격에서 수정함
4f65b22 원격에서 수정함
03faf0a remote Ttest02 bb메소드 수정
54da1ad Test02 추가
9e7923d first commit

```

그리고, 원격에 push한다.

```

KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipseWork/Git_295 (main)
$ git push origin main
Enumerating objects: 18, done.
Counting objects: 100% (18/18), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (10/10), 863 bytes | 863.00 KiB/s, done.
Total 10 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To https://github.com/8253jang/Git_295.git
  596dee6..75b16a2 main -> main

```

```
KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipseWork/Git_295 (main)
$ git log --oneline
75b16a2 (HEAD -> main, origin/main) 충돌 해결 후 커밋
8e06b6e Test01 bb 메소드 토거에서 수정함
596dee6 원격에서 Test02 bb메소드 수정함
f4f44f5 Merge branch 'main' of https://github.com/8253jang/Git_295
94cd6c4 로컬에서 다른 파일 수정함
d80f66e 원격에서 수정함
4f65b22 원격에서 수정함
03faf0a remote Ttest02 bb메소드 수정
54dalad Test02 추가
9e7923d first commit
```

## 시나리오 05

Local 저장소에 수정 사항은 있으나 아직 커밋하기 전, remote 저장소에 커밋 이력이 있어 pull 하려는 상황에서 원격 커밋 이력이 로컬에서 수정하고 있는 파일 인 경우 – fetch는 되지만 merge도 되지 않고 어떤 부분이 다른지 정보를 알려주지 못 한다.

```
KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipseWork/Git_295 (main)
$ git pull origin main
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 5 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (5/5), 1.07 KiB | 5.00 KiB/s, done.
From https://github.com/8253jang/Git_295
 * branch            main      -> FETCH_HEAD
   75b16a2..7e9fbcd main      -> origin/main
error: Your local changes to the following files would be overwritten by merge:
      src/exam/Test02.java
Please commit your changes or stash them before you merge.
Aborting
Updating 75b16a2..7e9fbcd
```

## 해결 방법 2가지

- 먼저 로컬 수정사항을 커밋 한다.

다시 `git pull origin main` 한다

충돌 상황에서 개발자가 수정하고 다시 로컬 커밋한다.

원격에 `git push origin main` 한다.

```
KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipsework/Git_295 (main)
$ git add .

KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipsework/Git_295 (main)
$ git commit -m "로컬 수정사항 커밋"
[main 3b45edc] 로컬 수정사항 커밋
 1 file changed, 1 insertion(+)
```

```
KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipsework/Git_295 (main)
$ git pull origin main
From https://github.com/8253jang/Git_295
 * branch            main      -> FETCH_HEAD
Auto-merging src/exam/Test02.java
CONFLICT (content): Merge conflict in src/exam/Test02.java
Automatic merge failed; fix conflicts and then commit the result.
```

```
3 public class Test02 {
4     public void bb() {
5         System.out.println("원격에서 수정함");
6         System.out.println("원격에서 두번째 수정했음");
7         <<<<< HEAD
8         System.out.println("로컬에서 같은 부분 수정중....");
9         =====
10        System.out.println("원격에서 세번째 수정함");
11        >>>>> 7e9fbcd9ef2118941564bef962a7c36856bfe30e
12    }
13
14 }
15
```

```
KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipsework/Git_295 (main|MERGING)
$ git add .

KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipsework/Git_295 (main|MERGING)
$ git status
On branch main
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:
  modified:   src/exam/Test02.java

KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipsework/Git_295 (main|MERGING)
$ git commit -m "충돌 해결 후 커밋"
[main 6c257c6] 충돌 해결 후 커밋
```

```
KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipsework/Git_295 (main)
$ git log --oneline
6c257c6 (HEAD -> main) 충돌 해결 후 커밋
3b45edc 로컬 수정사항 커밋
7e9fbcd (origin/main) 원격에서 Test02.java bb 수정함
75b16a2 충돌 해결 후 커밋
8e06b6e Test01 bb 메소드로 커서에서 수정함
596dee6 원격에서 Test02 bb메소드 수정함
f4f44f5 Merge branch 'main' of https://github.com/8253jang/Git_295
94cd6c4 로컬에서 다른 파일 수정함
d80f66e 원격에서 수정함
4f65b22 원격에서 수정함
03faf0a remote Ttest02 bb메소드 수정
54da1ad Test02 추가
9e7923d first commit
```

```
KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipsework/Git_295 (main)
$ git push origin main
Enumerating objects: 18, done.
Counting objects: 100% (18/18), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (10/10), 857 bytes | 857.00 KiB/s, done.
Total 10 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To https://github.com/8253jang/Git_295.git
    7e9fbcd..6c257c6 main -> main
```

```
KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipsework/Git_295 (main)
$ git log --oneline
6c257c6 (HEAD -> main, origin/main) 충돌 해결 후 커밋
3b45edc 로컬 수정사항 커밋
7e9fbcd 원격에서 Test02.java bb 수정함
75b16a2 충돌 해결 후 커밋
8e06b6e Test01 bb 메소드로 커서에서 수정함
596dee6 원격에서 Test02 bb메소드 수정함
f4f44f5 Merge branch 'main' of https://github.com/8253jang/Git_295
94cd6c4 로컬에서 다른 파일 수정함
d80f66e 원격에서 수정함
4f65b22 원격에서 수정함
03faf0a remote Ttest02 bb메소드 수정
54da1ad Test02 추가
9e7923d first commit
```

2) 먼저, 로컬의 수정사항을 **git stash**(현재 작업 중인 변경 사항을 임시로 저장하는 기능)로 임시 저장한다.

다시 **git pull origin main** 한다.

**git stash pop**으로 저장된 정보를 불러온다.

충돌 상황에서 개발자가 수정하고 다시 로컬 커밋한다.

원격에 git push origin main 한다.

```
KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipsework/Git_295 (main)
$ git stash
Saved working directory and index state WIP on main: 6c257c6 총 둘 해 결 후 커밋
```

```
KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipsework/Git_295 (main)
$ git pull origin main
From https://github.com/8253jang/Git_295
 * branch           main      -> FETCH_HEAD
Updating 6c257c6..715f641
Fast-forward
  src/exam/Test02.java | 2 ++
  1 file changed, 1 insertion(+), 1 deletion(-)
```

```
KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipsework/Git_295 (main)
$ git stash pop
Auto-merging src/exam/Test02.java
CONFLICT (content): Merge conflict in src/exam/Test02.java
On branch main
Unmerged paths:
  (use "git restore --staged <file>..." to unstage)
  (use "git add <file>..." to mark resolution)
    both modified:  src/exam/Test02.java

no changes added to commit (use "git add" and/or "git commit -a")
The stash entry is kept in case you need it again.
```

```
3 public class Test02 {
4     public void bb() {
5         System.out.println("원격에서 수정함");
6         System.out.println("원격에서 두번째 수정했음");
7         System.out.println("로컬에서 같은 부분 수정중....");
8         System.out.println("원격에서 세번째 수정함");
9         <<<<< Updated upstream
10        System.out.println("원격에서 네번째 수정함");
11        =====
12        System.out.println("로컬에서 네번째 수정함");
13        >>>>> Stashed changes
14    }
15
16 }
17
```

```
(KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipsework/Git_295 (main)
$ git add .

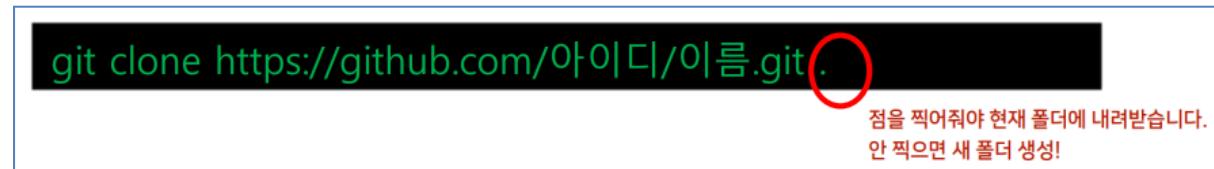
KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipsework/Git_295 (main)
$ git commit -m "stash pop 후 충돌 해결 커밋"
[main 26e231c] stash pop 후 충돌 해결 커밋
  1 file changed, 5 insertions(+), 1 deletion(-)

KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipsework/Git_295 (main)
$ git log --oneline
26e231c (HEAD -> main) stash pop 후 충돌 해결 커밋
715f641 (origin/main) 원격 Test02.java 내용 수정
6c257c6 충돌 해결 후 커밋
3b45edc 로컬 수정사항 커밋
7e9fbcd 원격에서 Test02.java bb 수정함
75b16a2 충돌 해결 후 커밋
8e06b6e Test01 bb 메소드로 커에서 수정함
596dee6 원격에서 Test02 bb메소드 수정함
f4f44f5 Merge branch 'main' of https://github.com/8253jang/Git_295
94cd6c4 로컬에서 다른 파일 수정함
d80f66e 원격에서 수정함
4f65b22 원격에서 수정함
03faf0a remote Ttest02 bb메소드 수정
54da1ad Test02 추가
9e7923d first commit
```

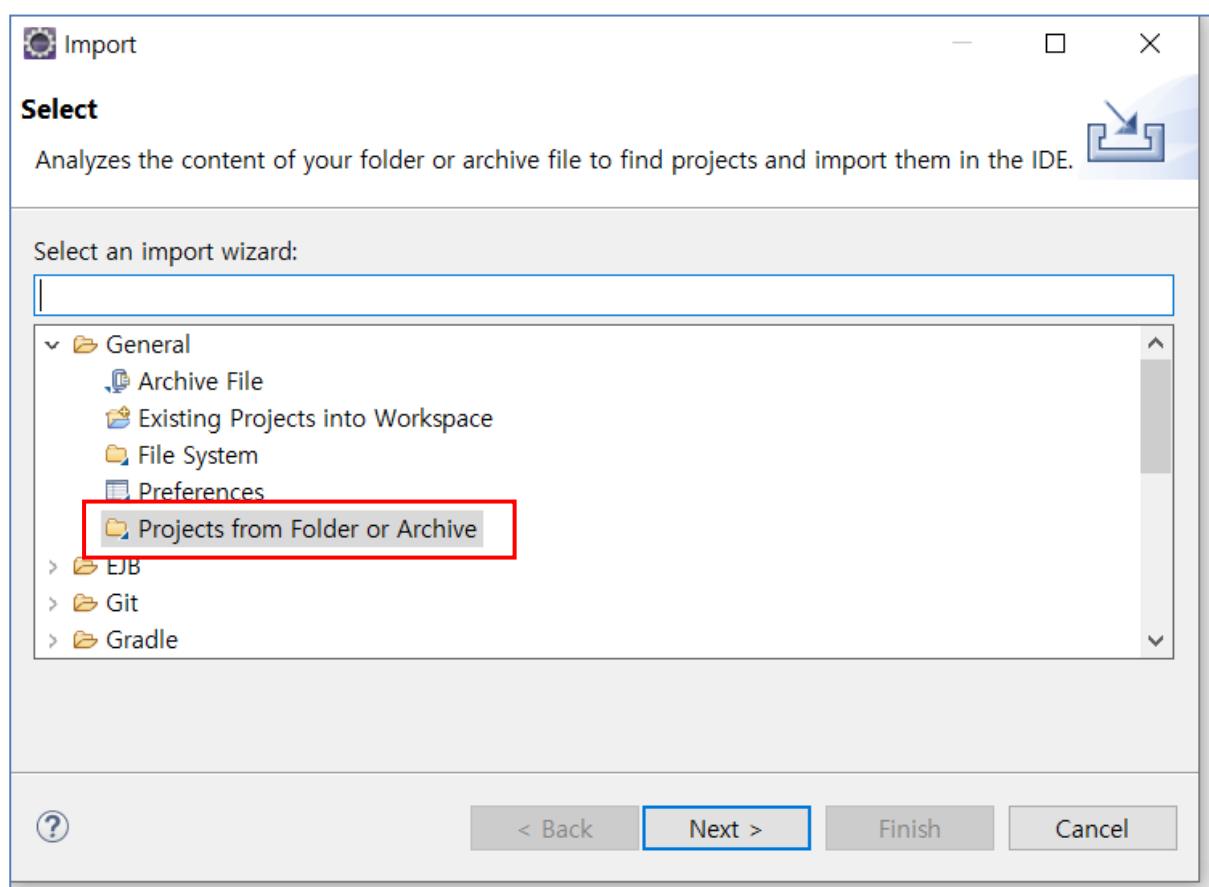
```
KOSTA@DESKTOP-7PLAKSP MINGW64 /d/GitTest/eclipsework/Git_295 (main)
$ git push origin main
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 442 bytes | 442.00 KiB/s, done.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/8253jang/Git_295.git
  715f641..26e231c  main -> main
```

## GitHub저장소를 내 컴퓨터에 받아오기(Clone)

- 1) 내 컴퓨터에 GitTestClone 폴더를 만든다.
- 2) GitHub의 GitTest03 저장소 받아오기



- 3) Clone 이후 이클립스를 열고 import를 한다.



## IntelliJ에서 Clone 해보기

