8253jang@daum.net
장희정

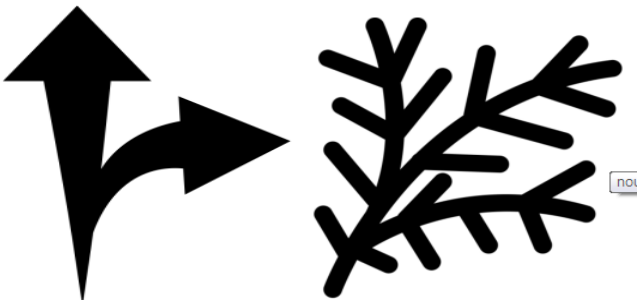
브랜치(Branch)

Git에서 브랜치란?

프로젝트시 개발자들은 동일한 소스 코드를 서로 공유하고 다루게 된다.

하지만 개발을 하다 보면, 동일한 버전의 프로그램이라도 어떤 사람은 버그 수정을 맡고, 다른 사람은 새로운 기능을 담당하는 등 서로 다른 버전의 코드를 작업하게 된다.

즉 동일한 뿌리에서 출발했지만 서로 다른 방향을 지향해서 뻗어나가게 되는데 이 모습이 바로 하나의 줄기에서 뻗어나가는 나뭇가지를 연상할 수 있다.



즉, 여러 개발자가 동시에 서로 다른 작업을 하는 것을 효율적으로 만들어 준다.
작업이란 결과적으로 독립적인 영역에서 하는 것이니만큼 GitHub에서 기둥이(Master or main) 되는 소스를 내려받아서 자신 마음대로 작업을 해도 그것을 각각의 버전으로 만들어 낼 수 있다.
이것이 branch 기능이다.
또한 branch로 작업을 하다가 실패하거나 원래로 돌아가려고 하면 가지를 잘라버리듯이 원래의 Master 기능으로 돌아갈 수도 있다.

Ex) 배포용, 테스트용 새로운시도, 신기능, 코드개선, 긴급수정

실무에서 아래와 같은 브랜치명을많이 사용하지만 이는 팀마다 다를 수 있다.

feature/xxx_register
- 새로운 기능을 추가할 때 사용
- 가장 많이 사용

bugfix/xxx_register
- 버그를 수정할 때 사용.

8253jang@daum.net
장희정

hotfix/xxx_register
- 긴급한 버그 수정을 위한 브랜치

브랜치 생성 및 통합 작업 순서

1) local 에서 main 브랜치 최신화 한다 (**git pull origin main**)

2) 현재 main 기준으로 새로운 브랜치생성한다.(**git switch -c 새로운브랜치이름**)

3) 작업 후 git push origin 새로운브랜치명
Local 의 main 하고 합치지 않고 원격(remote)에 push 한다
(git push origin 새로운브랜치이름)

4) Pull Request 를 요청한다.
github 에서 pull requests > new pull request

5) 조원들이 PR 검토하고 main 에 합치는 권한을 가지고 있는 조원이 PR 을 처리한다.

6)작업 완료된 브랜치삭제(git branch –D 브랜치이름)

-Branch 생성

☞ git branch 브랜치이름

- Branch검색

☞ git branch

: 결과 앞에 * 있는것이 현재 브랜치 상태라는것.

- Branch이동하기

☞ git switch 브랜치이름

: 예전에는 switch 아니라 checkout 이었으나 2.23 version부터 변경

8253jang@daum.net
장희정

-Branch생성과 동시에 이동하기

git switch -c 브랜치이름

-Branch삭제하기

git branch -d 삭제할이름

tip: 지울 브랜치에 다른 브랜치로 적용되지 않은 내용의 커밋이 있을 시에는
-D(대문자) 옵션으로 강제 삭제
ex) git branch -D 브랜치명

-Branch이름 변경하기

git branch -m 기존브랜치이름 새로운브랜치이름

시나리오 01– 충돌 없는 경우

- 현재 mina 브랜치를 최신화 시킨다.
- 새로운 브랜치를 생성해서 이동한다.
- 이동한 새로운 브랜치에서 코딩을 하고 커밋 이력을 만든다.
- 새로운 브랜치 작업이 완료 되면 원격에 새로운 브랜치을 push한다.
- 원격에 접속해서 올라간 새로운 브랜치를 확인하고 pull Request를 요청한다.
- 조원들이 PR 검토하고 main에 합치는 권한을 가지고 있는 조원이 PR을 처리한다.
- 로컬의 main을 최신화 시켜 원격에 merge된 상황을 로컬로 가져온다.
- 작업이 완료된 local 의브랜치를 삭제한다.

시나리오 02– 충돌 있는 경우

- 현재 mina 브랜치를 최신화 시킨다.

8253jang@daum.net
장희정

- 새로운 브랜치를 생성해서 이동한다.

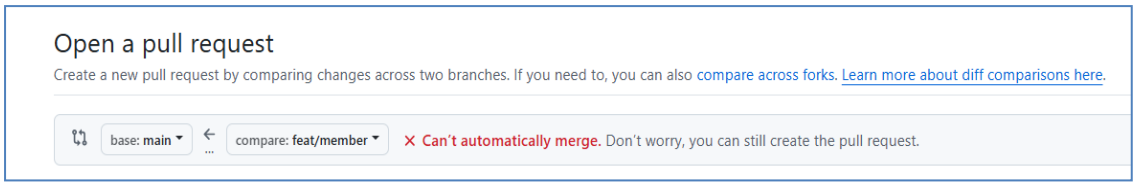
- 이동한 새로운 브랜치에서 코딩을 하고 커밋 이력을 만든다.

- **원격에서 main브랜치의 내용을 수정하고 커밋이력을 만든다. 이때 local의 새로운 브랜치에서 작업하고 있는 같은 파일의 같은 라인을 원격 main에서 수정하여 충돌이 발생할 수 있도록 한다.**

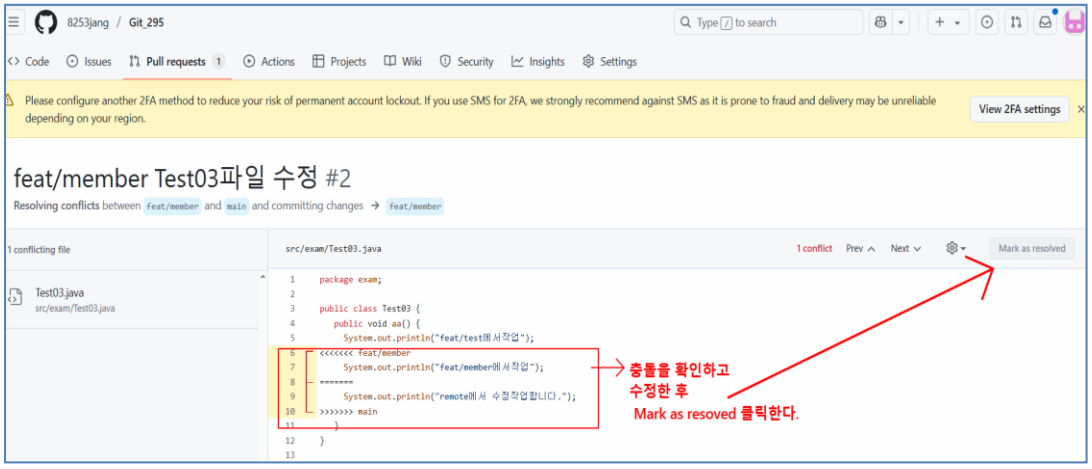
- 새로운 브랜치 작업이 완료 되면 원격에 새로운 브랜치을 push한다.

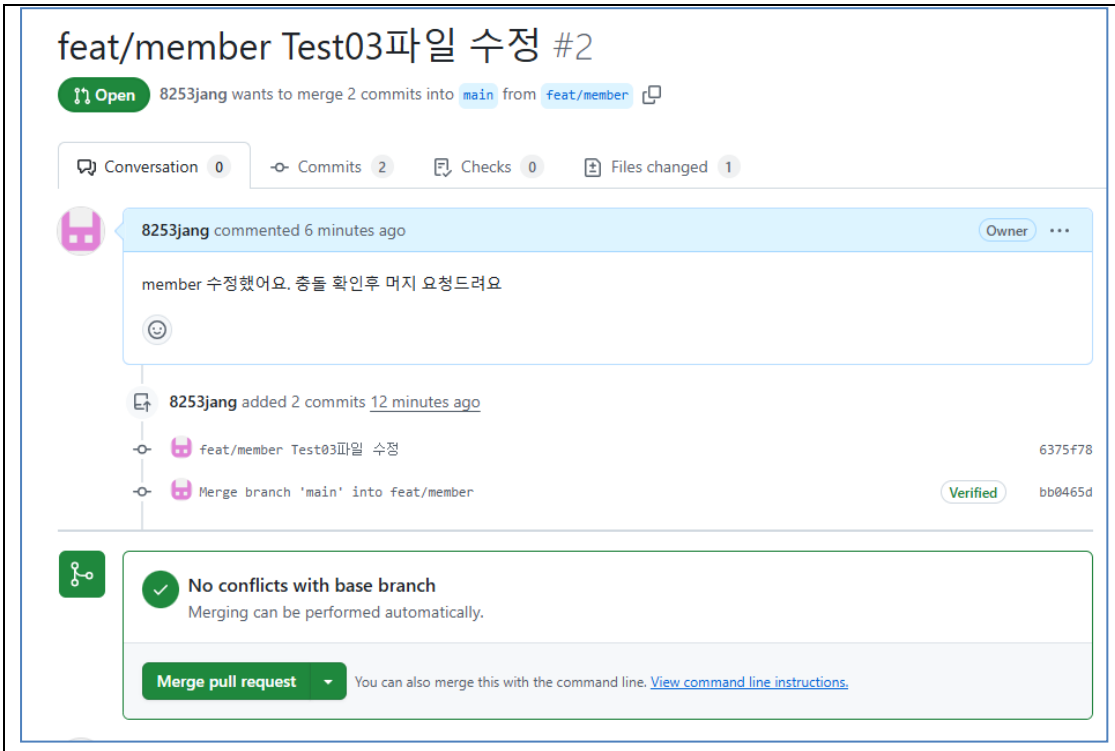
- 원격에 접속해서 올라간 새로운 브랜치를 확인하고 pull Request를 요청한다.

- pull Request를 확인하면 충돌이 발생하여 "자동 merge가 안된다"는 메시지를 확인 한다.



- 조원들이 PR 검토하고 main에 합치는 권한을 가지고 있는 조원이 **충돌을 원격에서 해결하고**PR을 처리한다.





- commit을 완료 한 후 Merge pull Request를 클릭 > Confirm merge 클릭 한다.
- 로컬의 main을 최신화 시켜 원격에 merge된 상황을 로컬로 가져온다.
- 작업이 완료된 local 의브랜치를 삭제한다.

Branch 사용 알아두기

Branch 생성시 주의 사항

항상 최신화된 main 에서 새로운브랜치 생성

- ✓ 새로 생성된브랜치를 작은 기능단위와 짧은 기간으로 유지한다.
- ✓ 수정 예정 프로그램에 대한 중복이 발생하지 않도록 동료간 충분한 의사소통한다.

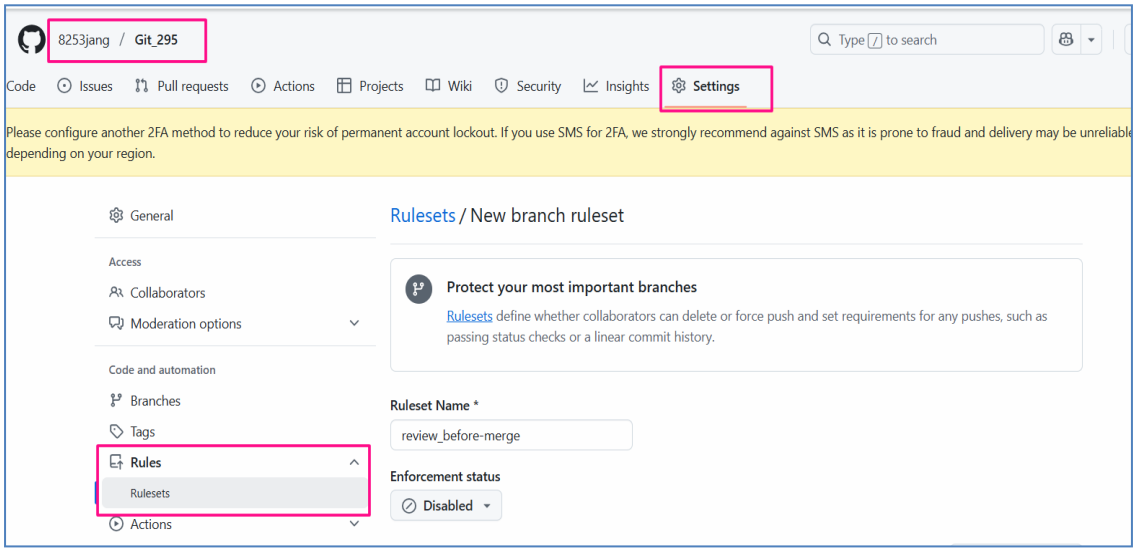
권한

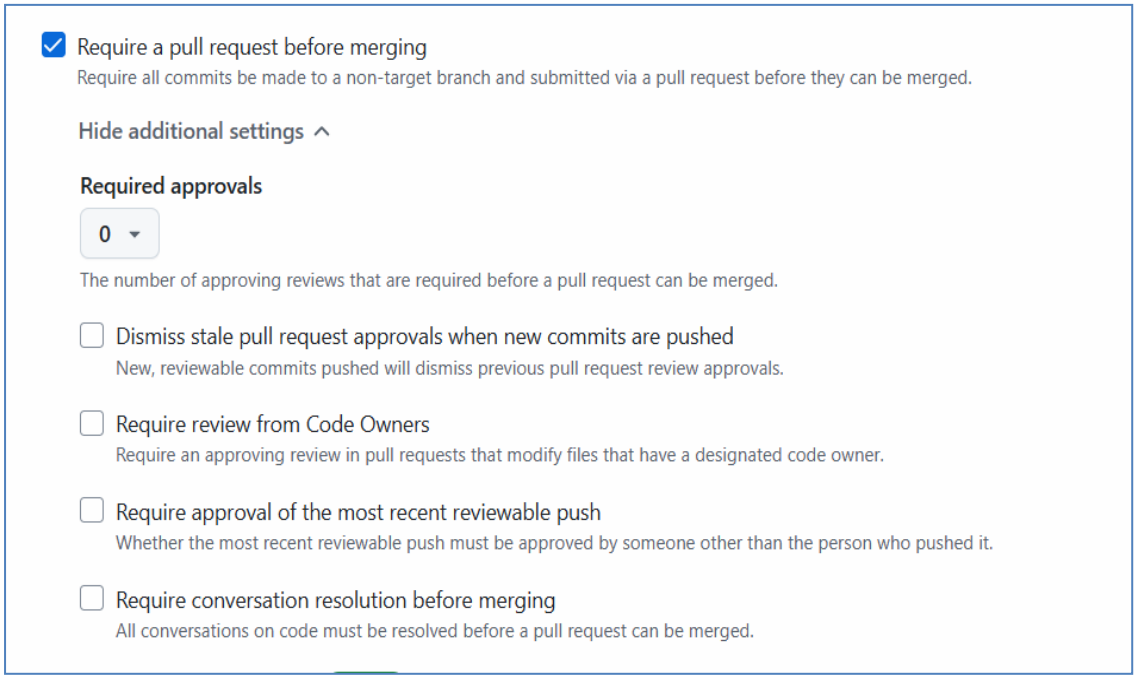
8253jang@daum.net
장희정

branch 관리 rule

✓ Require pull request reviews before merging 등의 rule 지정

Github> Repository선택 > Settings > Rules > Rulesets





8253jang@daum.net
장희정

Merge전략(merge , rebase , squash)

1) merge -merge는 두 브랜치의 변경 사항을 통합하는 기본적인 방법

2개의 가지를 이어 붙이는것으로 branch1에서 넘어온 commitID와
신규 merge commitID가 main브랜치에 남게되어커밋ID가 더 생긴다.
많은 브랜치를 사용하는 곳에서는 복잡할 수 있다.
Merge는 잔가지가 많이 남는다.

2) rebase

한 브랜치의커밋을 다른 브랜치의 최신 커밋에 "재적용"(re-apply)하는 방식이다.
이때에는 브랜치에서 넘어온 commitID 가 아닌, 새로운 commitID 가 발급되어
main 브랜치에 생성된다.
히스토리가 깔끔하게 한줄기로 표현된다.

- ✓ **장점**
  merge commitID 는 남지 않게 되어, 불필요한 커밋 없이 깔끔한 커밋관리
- ✓ **단점**
  기존의 commit history 자체는 유지되지만, 모든 commitID 가 변경됨으로서 이후에
  동일 브랜치에서 재 pull Request 시 충돌이 발생하므로, 사용하던 branch 는
  더이상 사용이 어렵다.

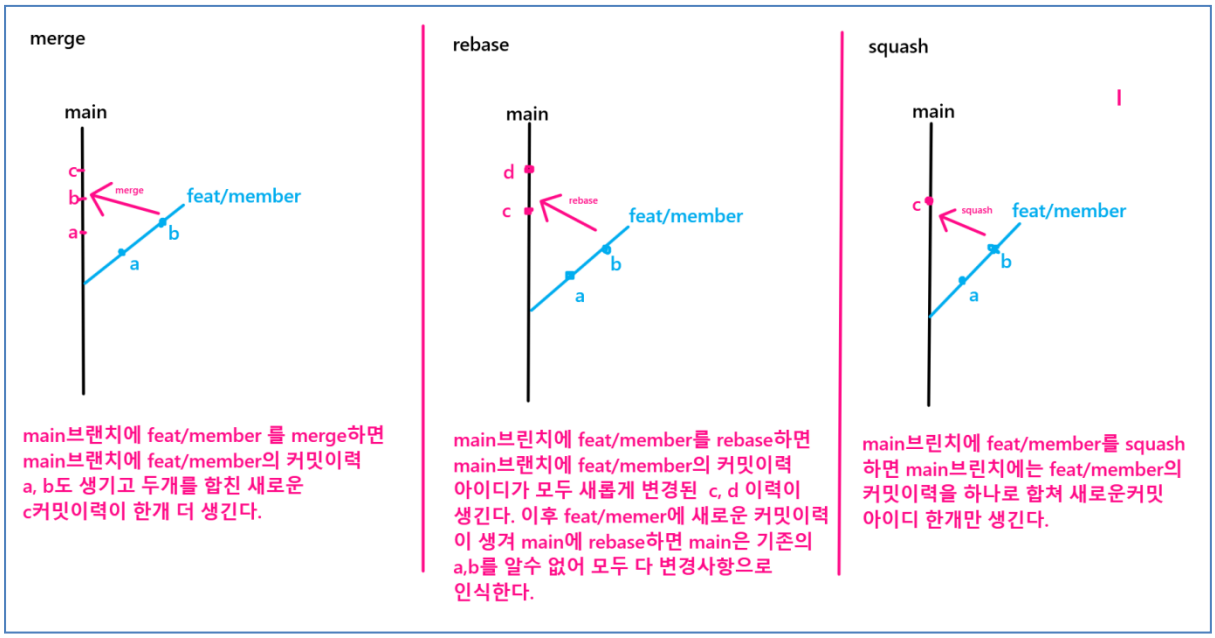
3) squash

squash 는 여러 커밋을 하나의 커밋으로 합치는 과정이다.
local repository 에서 여러 커밋을 발생시켰을때 해당 커밋 ID 를 통합하여 하나의
commitID 로 만들어 main 에는 하나의 commitID 로만 이력 생성된다.

- ✓ **장점**
  불필요한 커밋 없이 깔끔한 커밋관리
- ✓ **단점**
  기존에 사용하던 branch 는 더이상 사용이 어렵다.

*** 진행하는 프로젝트 성격에 따라 선택 merge 전략을 선택한다.**

8253jang@daum.net
장희정



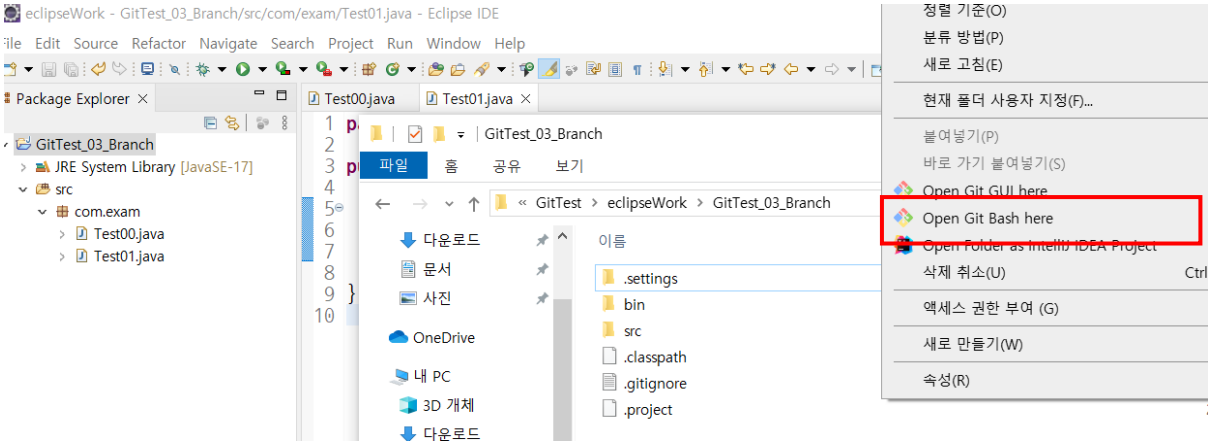
실습하기

: 이번 실습은 eclipse tool을 이용하여 프로젝트를 생성하고 생성된 프로젝트에서 2개의 branch(main + order)를 가지고 실습해보자.

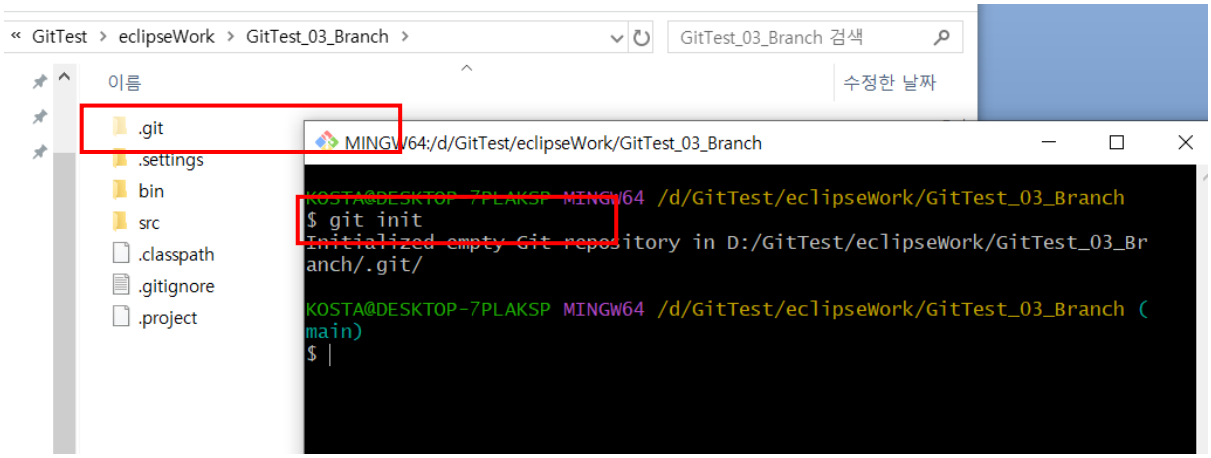
1) Git을 test할 workspace를 정한다.
2) Eclipse을 열고 새로운 javaProject를 생성한다.
3) src/com/exam/Test00.java , Test01.java 파일을 만든다.
4) Git을 초기화 한다.
   : git을 초기화 하는 방법은 gitbash를 이용해도 되고 이클립스 tool을 이용해도 된다.

gitBash이용하기

8253jang@daum.net
장희정



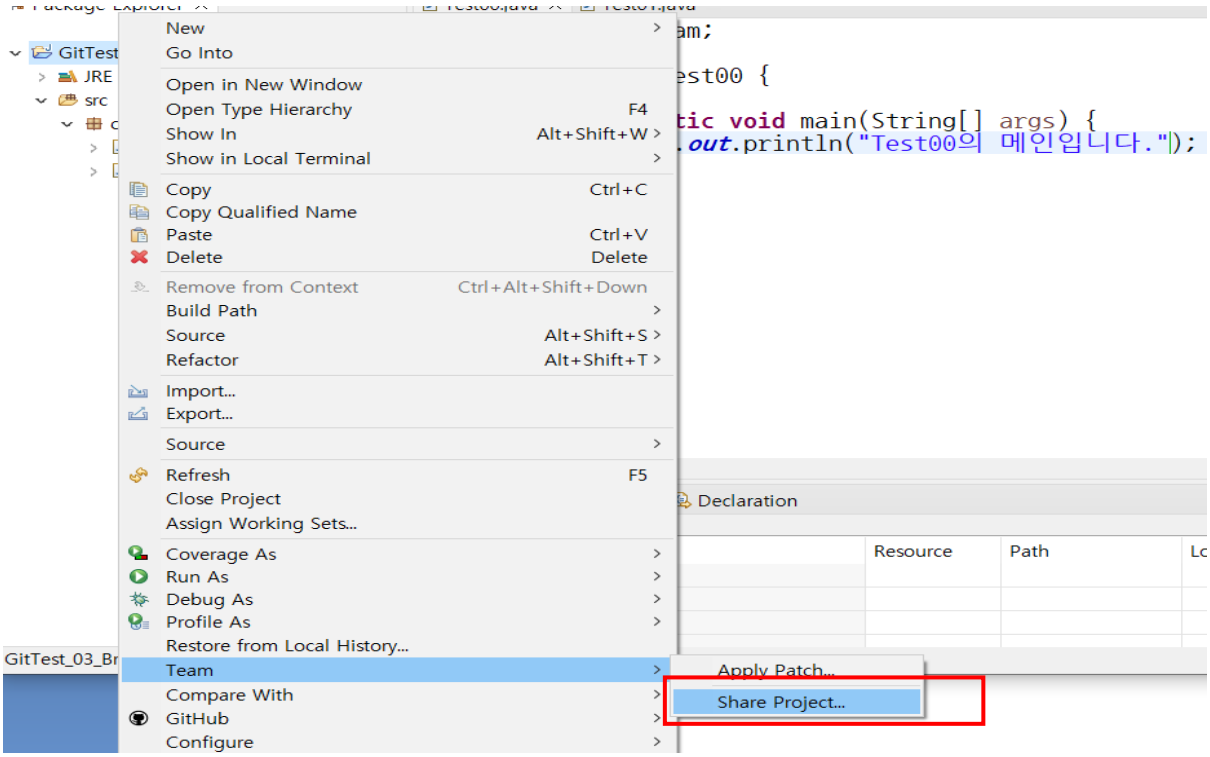
git init 명령어를 실행하면 .git이 만들어진다.



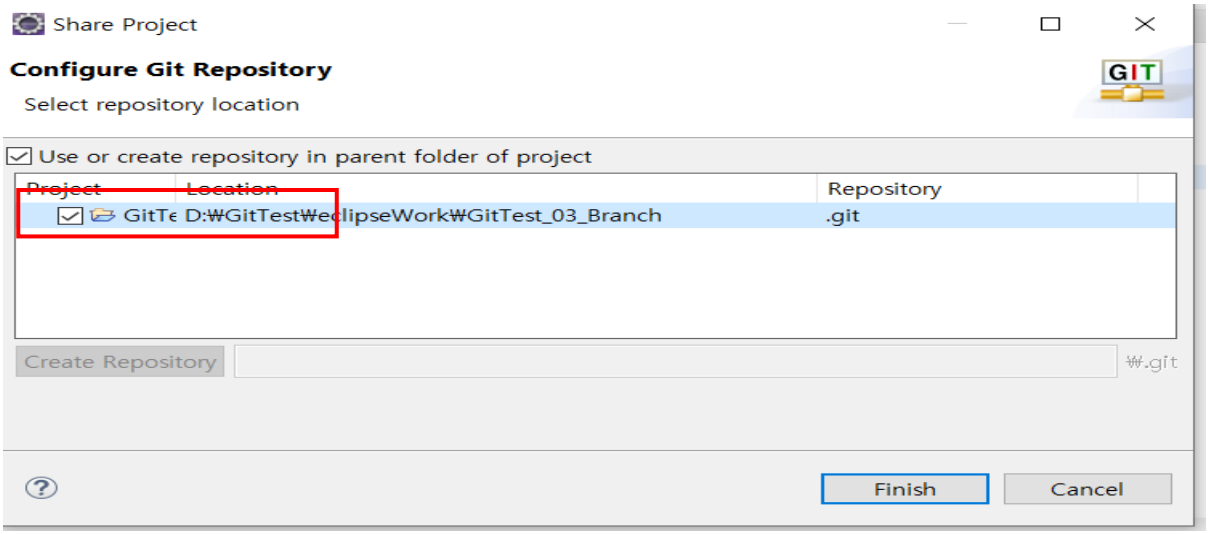
Eclipse tool 이용하기

프로젝트를 선택하고 오른쪽 버튼 클릭 -> Team 선택 -> Share Project선택

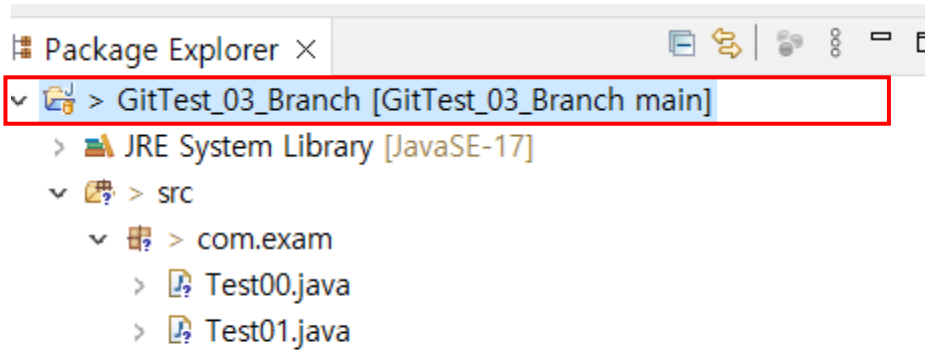
8253jang@daum.net
장희정



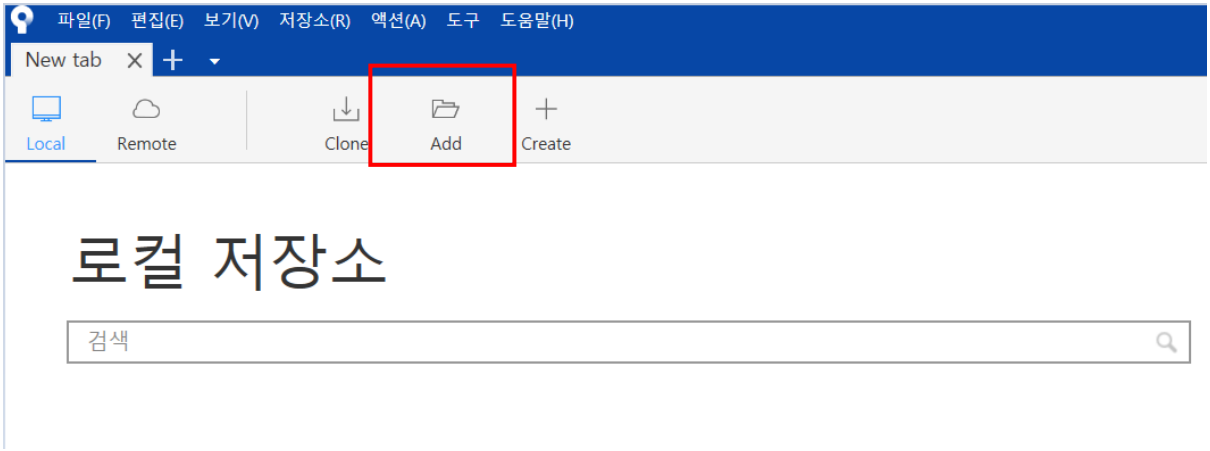
체크박스 클릭 ->Finsh 클릭

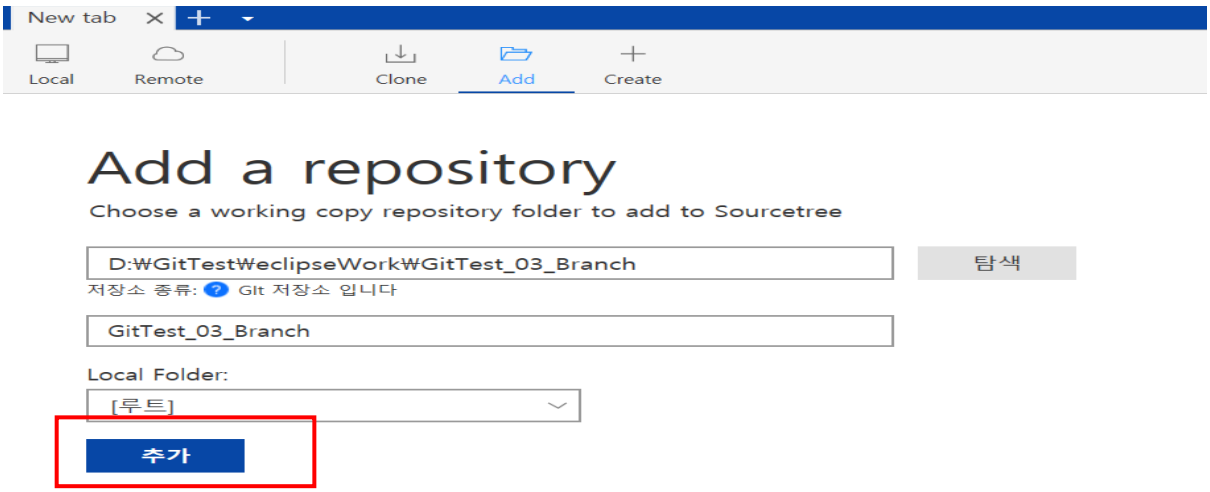


8253jang@daum.net
장희정

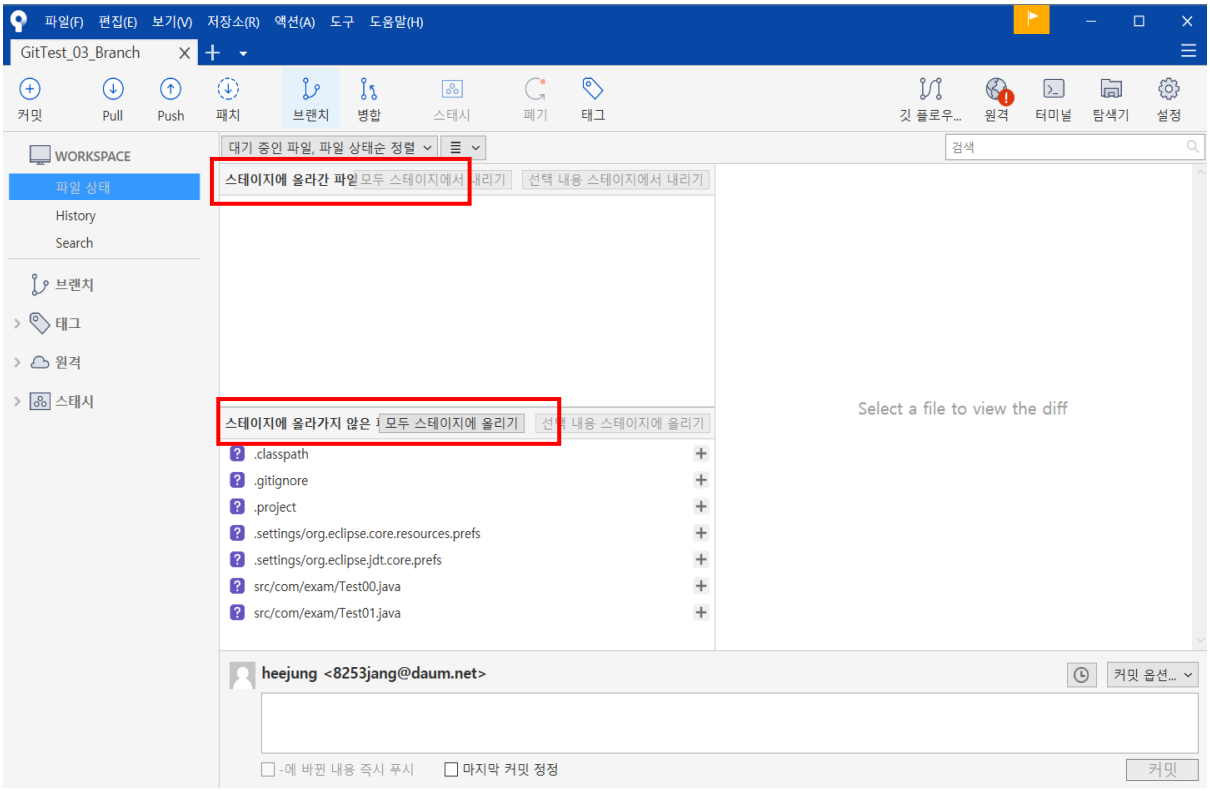


SourceTree 연결하기





8253jang@daum.net
장희정



.gitignore 파일 수정하기

프로젝트에서 원하지 않는 백업 파일이나 로그파일 혹은 컴파일 된 파일들을 Git 에서 제외할 수 있는 설정 파일을 말한다.

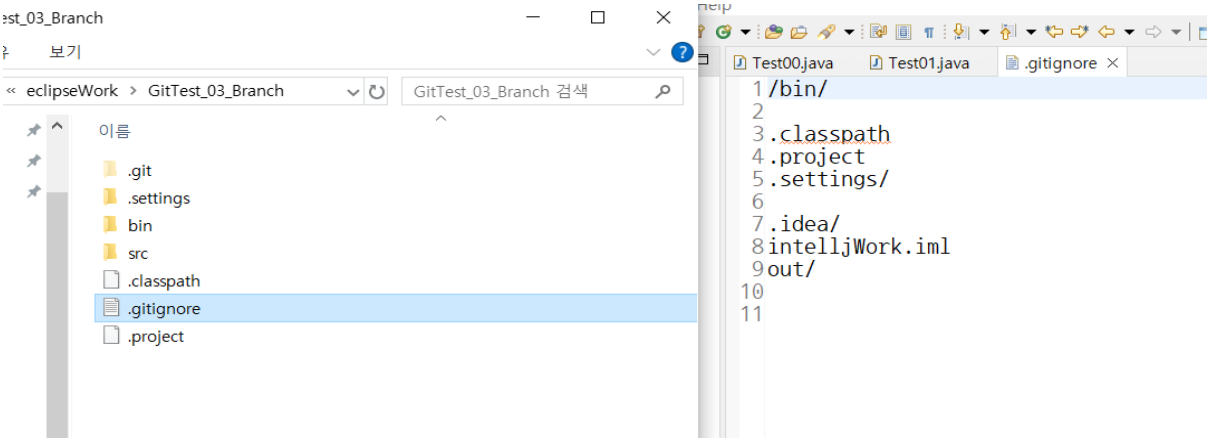
- 보안상으로 위험성이 있는 파일
- 프로젝트와 관계없는 파일
- 용량이 너무 커서 제외해야되는 파일
- 편리성 제공

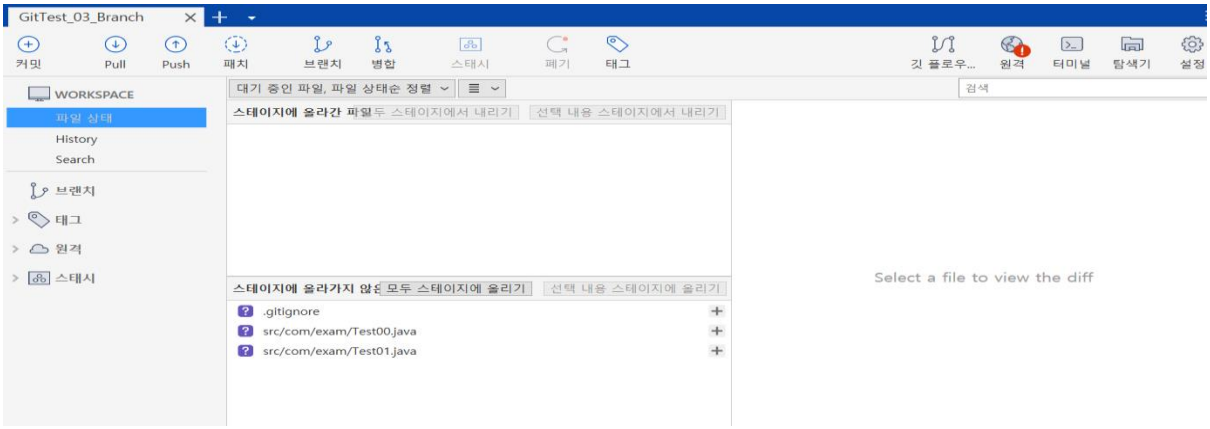
8253jang@daum.net
장희정

.gitignore 파일 규칙

표현	의미
#, 빈라인	#은 주석을 의미하며, 빈라인은 아무런 영향을 주지 않습니다.
*.a	확장자가 .a 인 모든 파일을 무시합니다.
folder_name/	해당 폴더의 모든 파일을 무시합니다.
folder_name/*.a	해당 폴더의 확장자가 .a 인 모든 파일을 무시합니다.
folder_name/*/.a	해당 폴더 포함한 하위 모든 폴더에서 확장자가 .a 인 모든 파일을 무시합니다.
/*.a	현재 폴더의 확장자가 .a 인 모든파일을 무시합니다.

.gitignore파일을 열어서 제외하고 싶은 파일을 선언한다.





8253jang@daum.net
장희정

git 캐시 삭제

이미 add 또는 commit 해버린경우 git 에서 추적이 되기시작하여, gitignore 에서 제외되지 않으므로 캐시삭제 필요하다

git rm -r --cached .

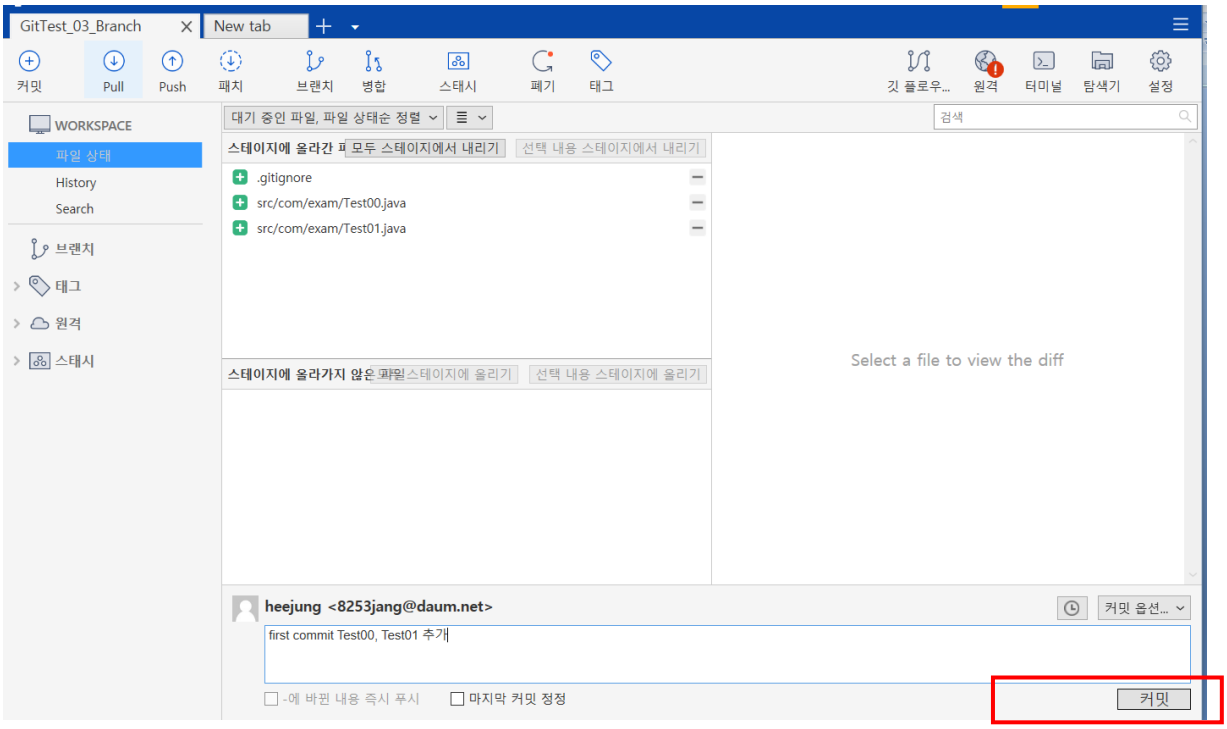
3개의 commit 이력을 만들어 보자.

1) Test00.java, Test01.java , .gitignore 추가

2) Test00.java 파일 수정 ,Test02.java 추가

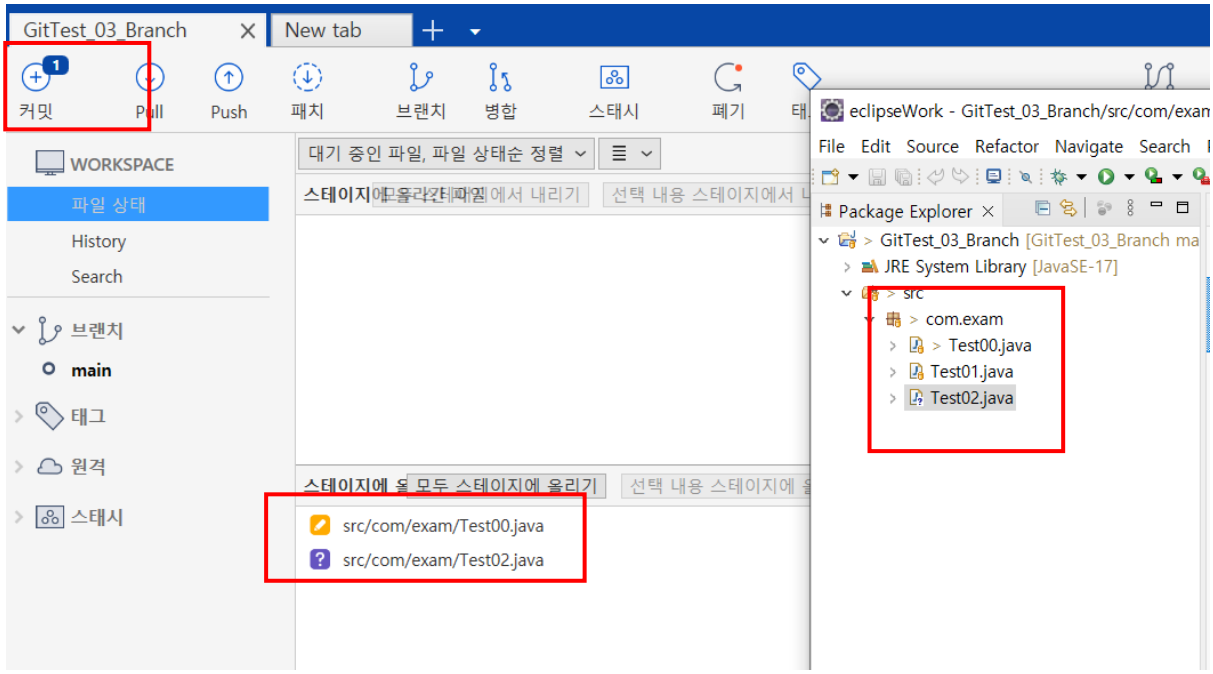
3) Test01.java 삭제, Test02.java 수정

1)

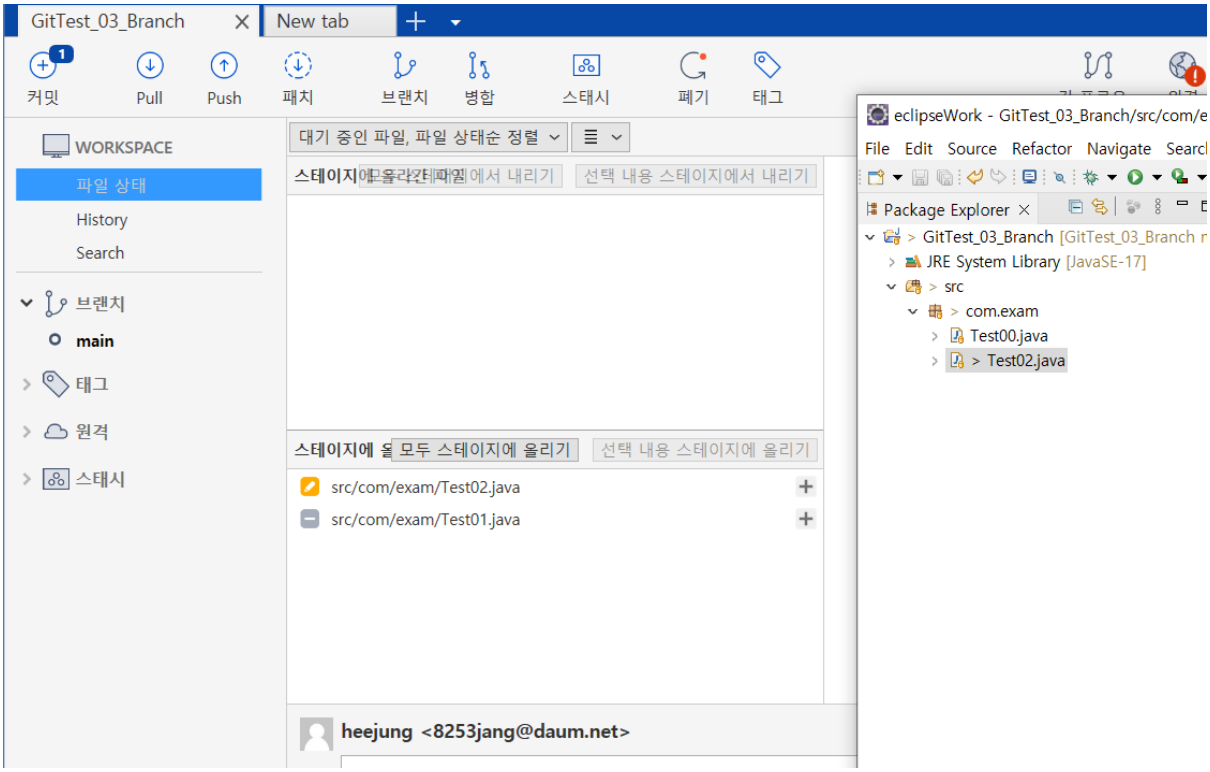


8253jang@daum.net
장희정

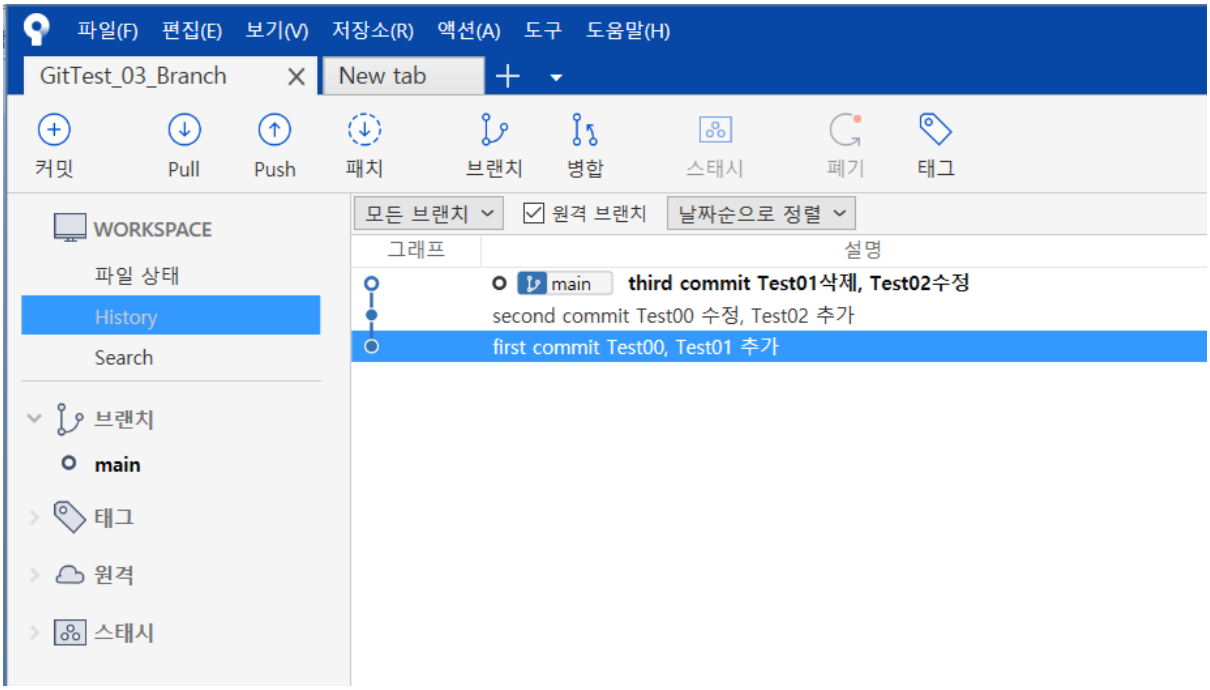
2)



3)



8253jang@daum.net
장희정

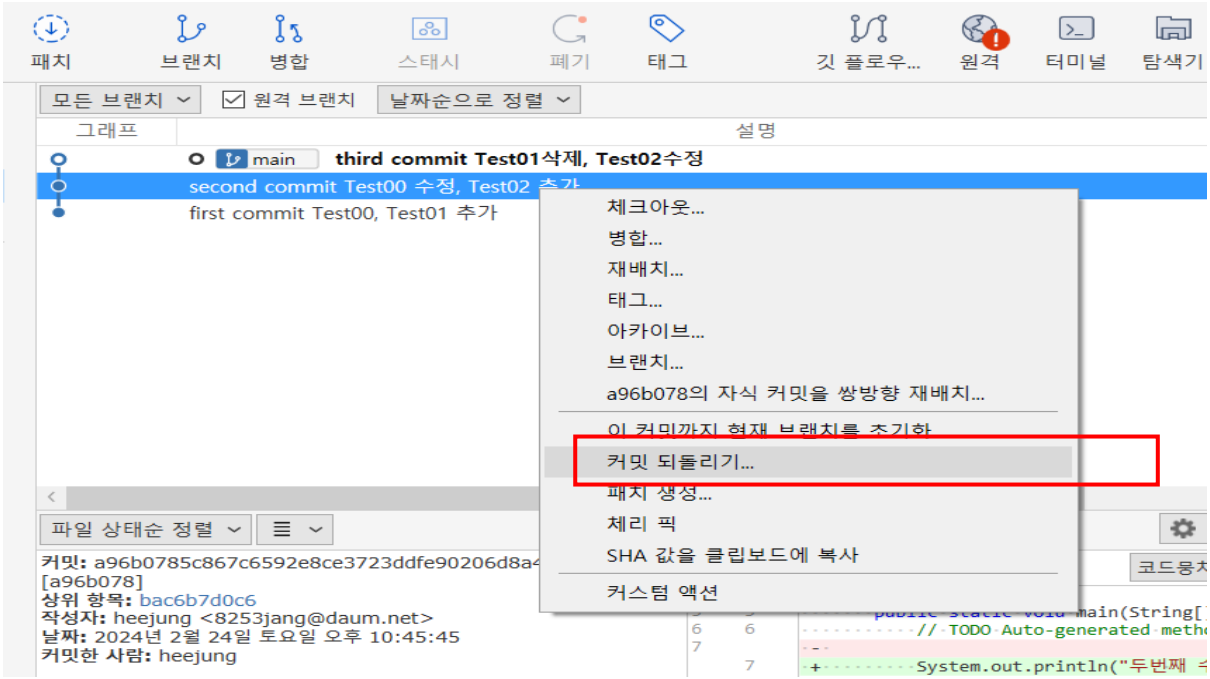


SourceTree에서 과거로 돌아가기 실습해보자.

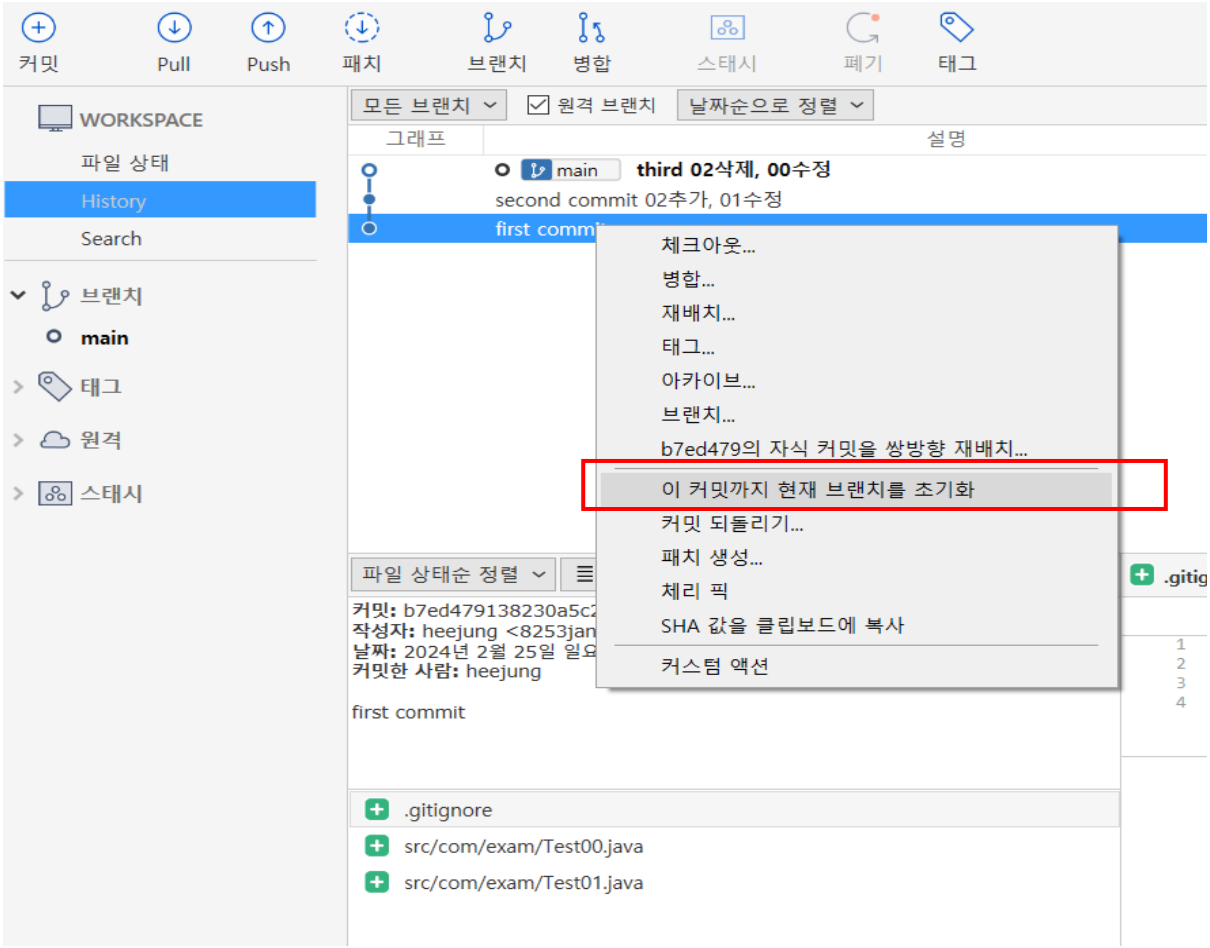
: 실습전에 .git을 다른 폴더에 복사해 둔다.

1) Revert

8253jang@daum.net
장희정



2) Reset

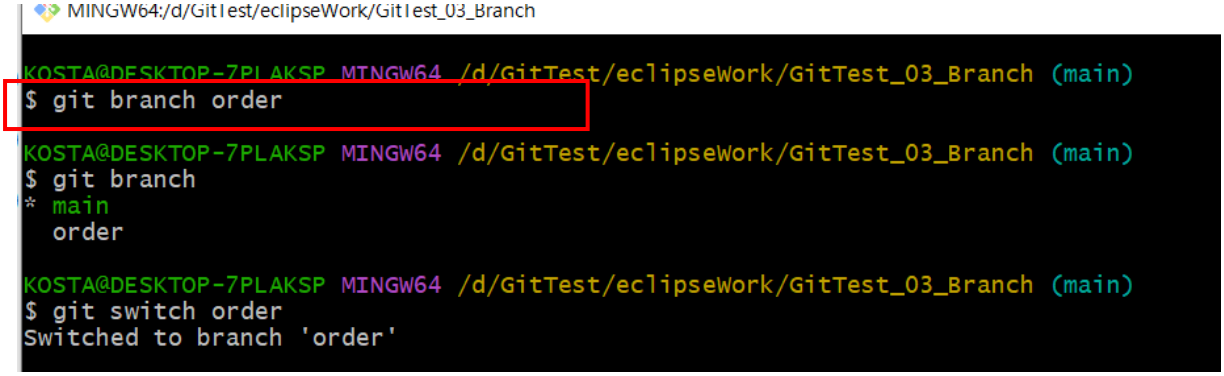


8253jang@daum.net
장희정

이제 SourceTree에서 git 사용 연습이 되었다면
Gitbash와 sourcetree를 함께 사용하여 Branch를 Test해보자.

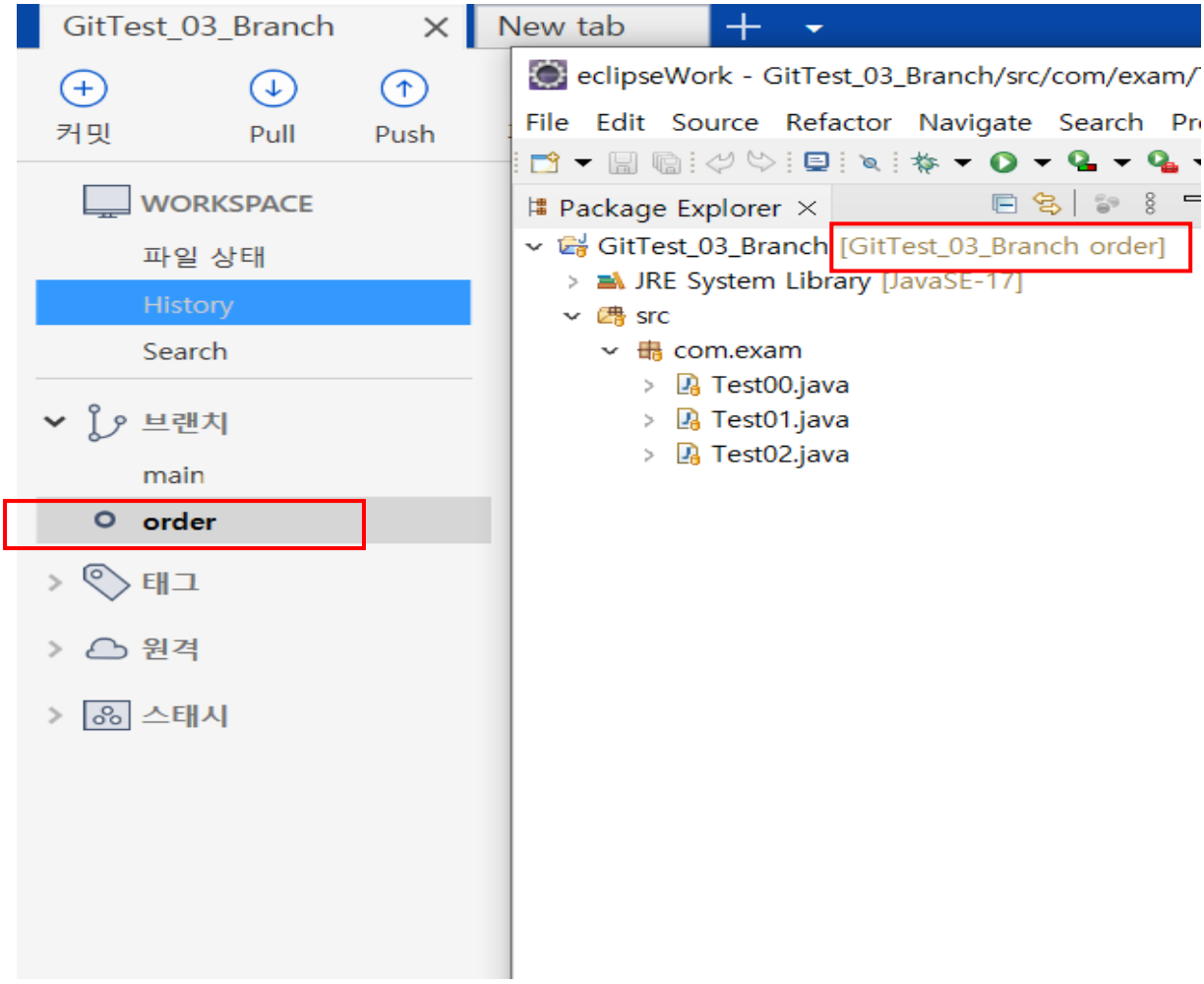
1) gitbash에서 브랜치를 생성한다.

2) 생성 생성된 브랜치로 이동한다.



3) sourcetree와 eclipse의 화면을 확인한다.

8253jang@daum.net
장희정



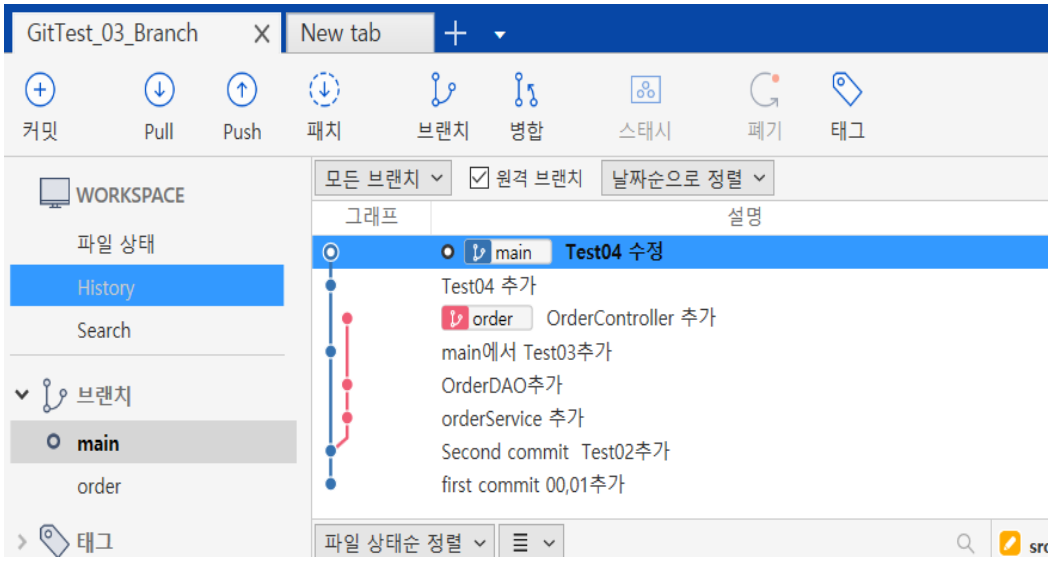
4) order branch에서 3개의 commit 이력을 만든다.

☞ git switch order

5) main branch에서 3개의 commit 이력을 만든다.

☞ git switch main

6) sourcetree에서 확인한다.



Main branch로 order branch 합치기

1) Merge : 2개의 가지를 이어 붙이는것으로 커밋하나가 더 생긴다.
많은 브랜치를 사용하는 곳에서는 복잡할 수 있다.
Merge는 잔가지가 많이 남는다.

2) Rebase :브랜치의 마디 커밋들을 대상(메인)으로 옮겨 붙이는것
Rebase는 히스토리가 깔끔하게 한줄기로 표현.

8253jang@daum.net
장희정

* 진행하는 성격에 따라 선택

브랜치의사용내역을 남겨야 하는경우는 Merge를,

history 깔끔하게 해야 하는 경우는 Rebase

(팀원들과 공유된 파일에 대해서는 rebase는 사용하지 않는것이 좋다.)

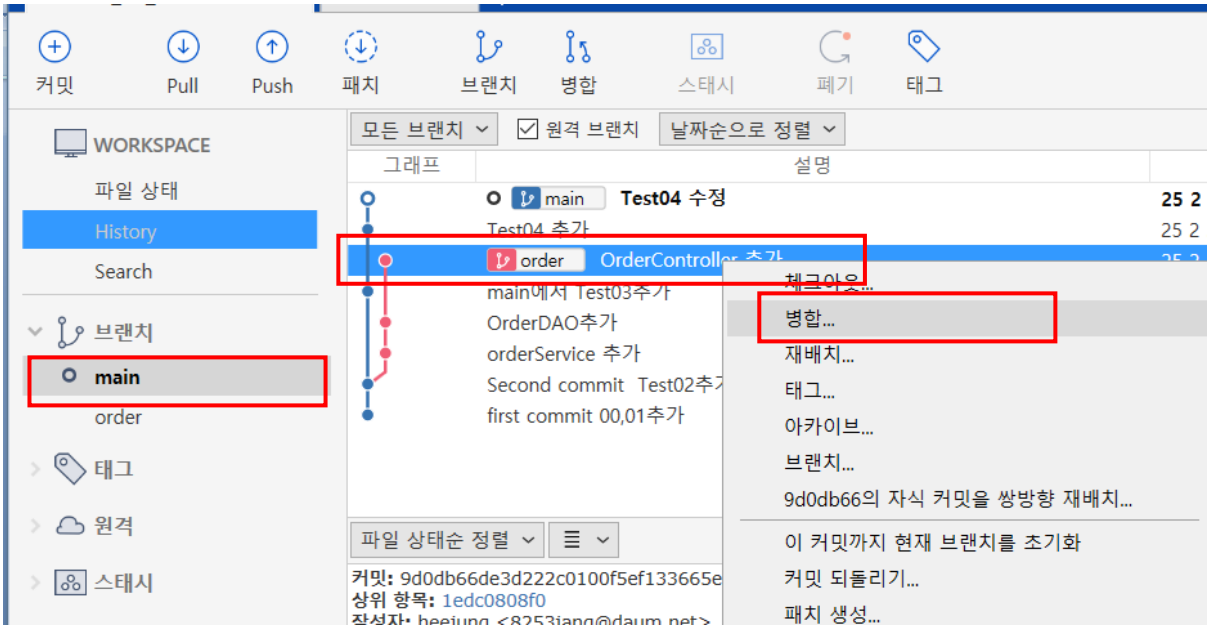
***실습하기 전에 변경된 모든 내역을 최종 commit 완료한다.**

*** 실습전에 .git을 다른 곳에 복사해 둔다.**

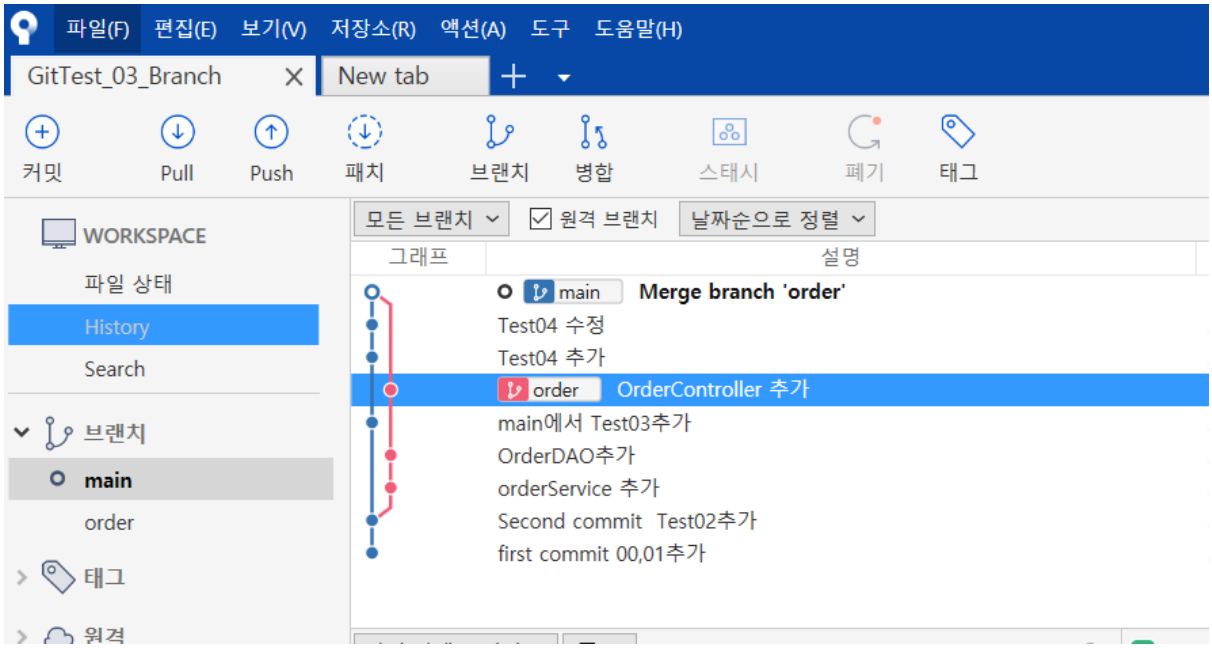
- Sourcetree에서 main에 order 브랜치merge 해보기

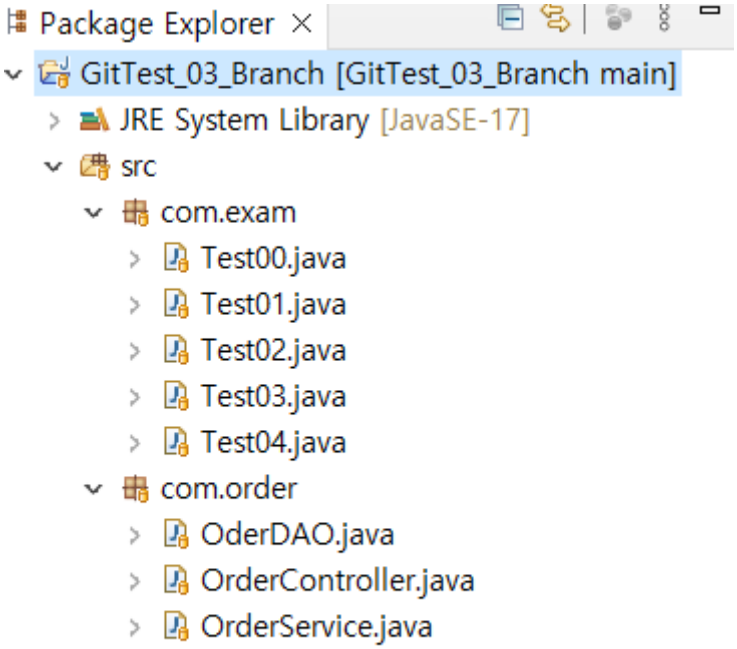
: main branch로 이동한다.

: order를 클릭 -> 오른쪽마우스 클릭 -> 병합 클릭



8253jang@daum.net
장희정



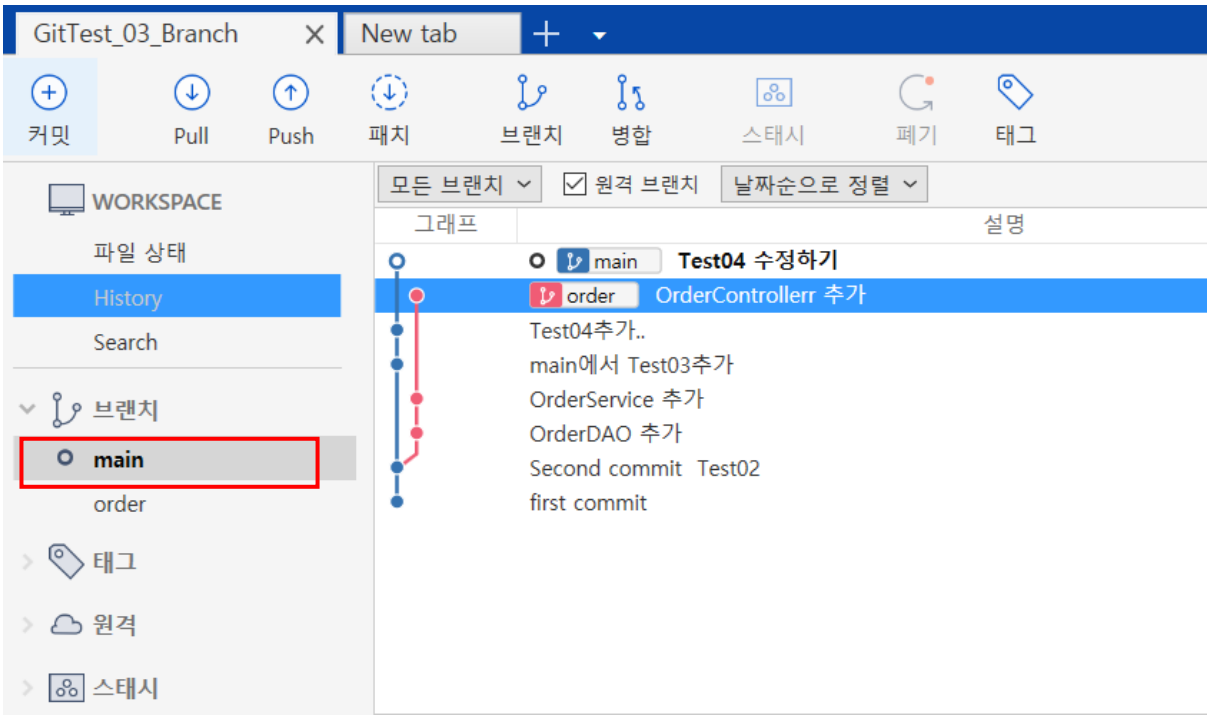


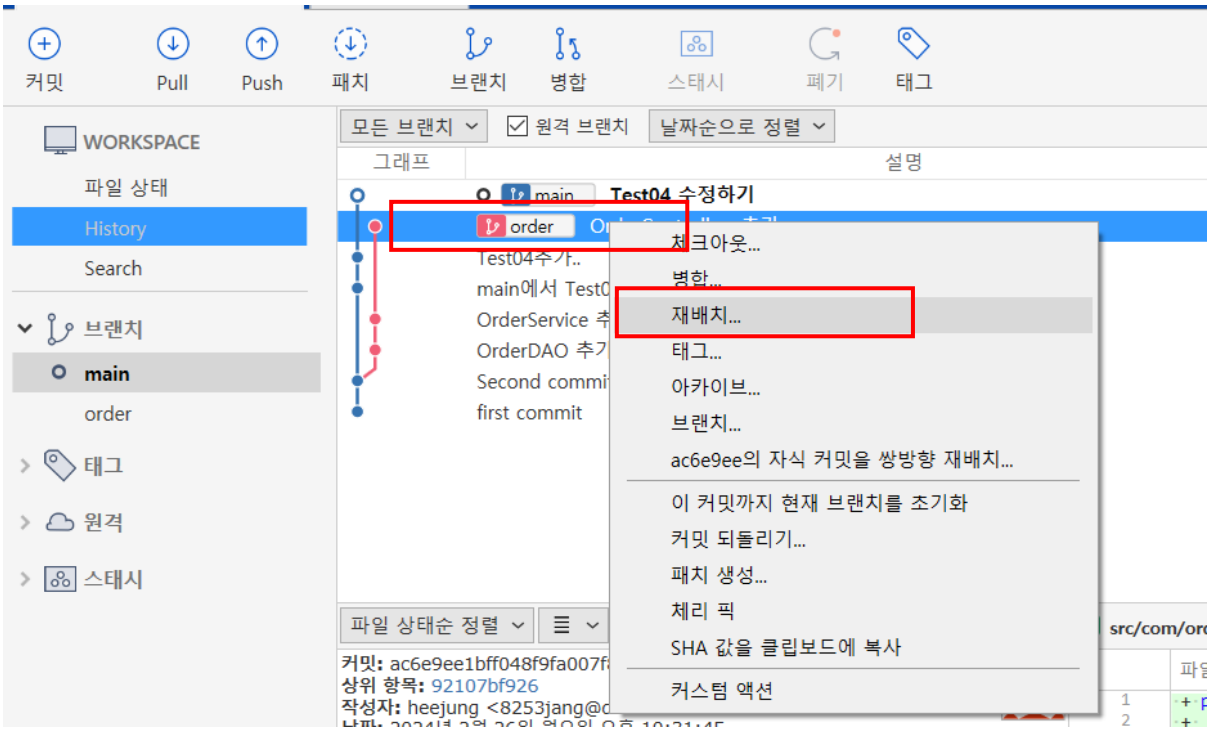
8253jang@daum.net
장희정

- Sourcetree에서 main에 order 브랜치rebase 해보기

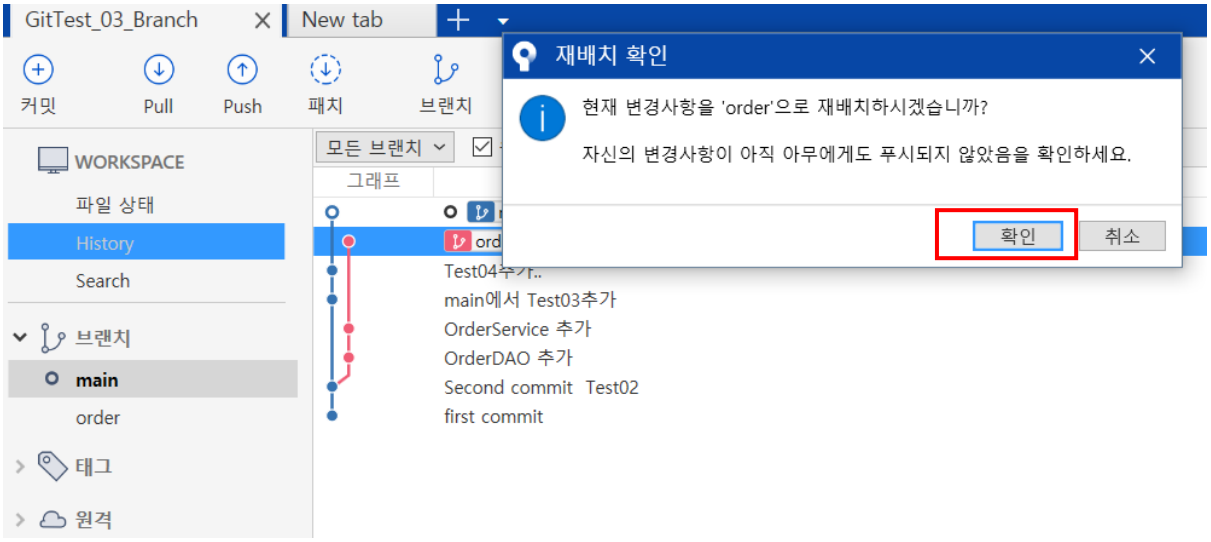
: 먼저 main브랜치로 이동한다.

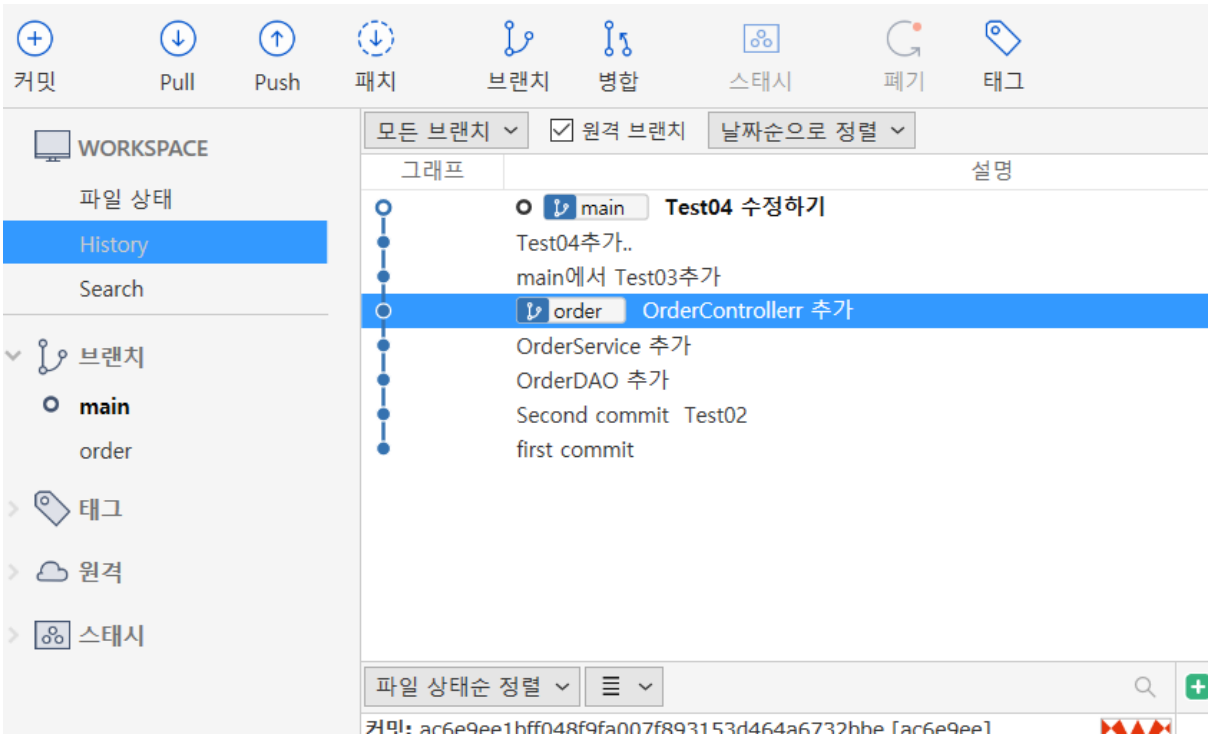
order를 클릭 -> 오른쪽마우스 클릭 -> 재배치 클릭



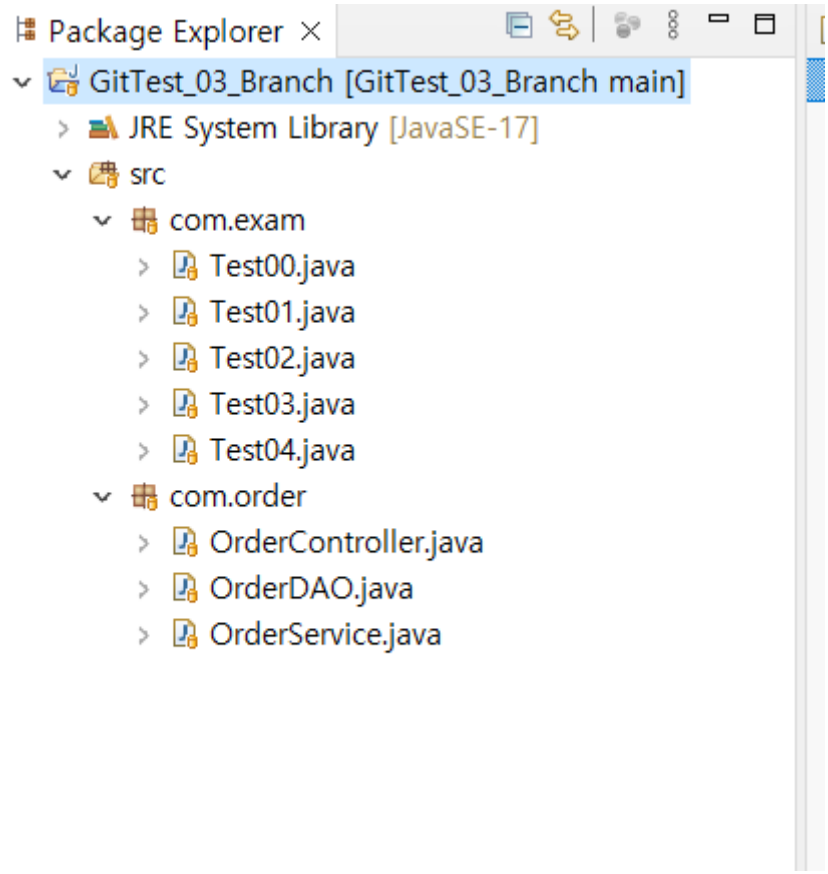


8253jang@daum.net
장희정





8253jang@daum.net
장희정



merge할 때 충돌해결하기

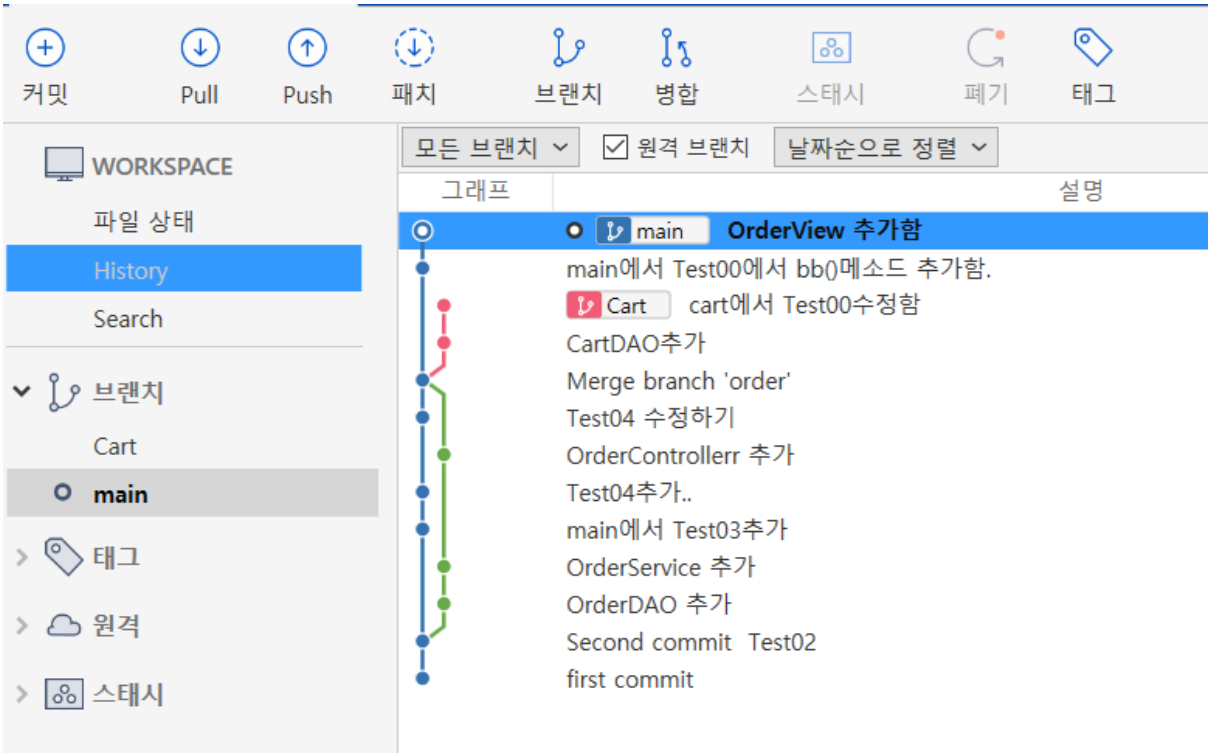
1) Cart 브랜치를 만든다.

2) Cart 브랜치에서 2개의 커밋을 만든다.

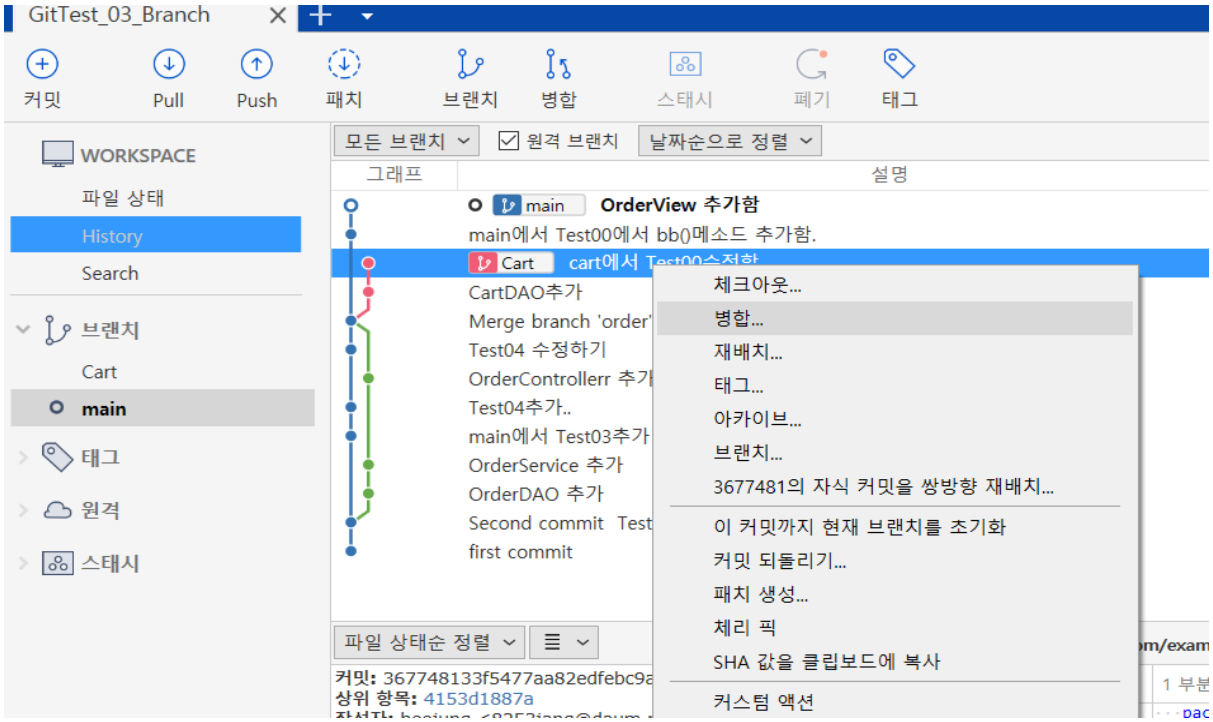
3) Main 브랜치에서 2개의 커밋을 만든다.

*** cart와 mian브랜치 각각 Test00파일의 같은 라인을 서로 다른 내용으로 수정한다.**

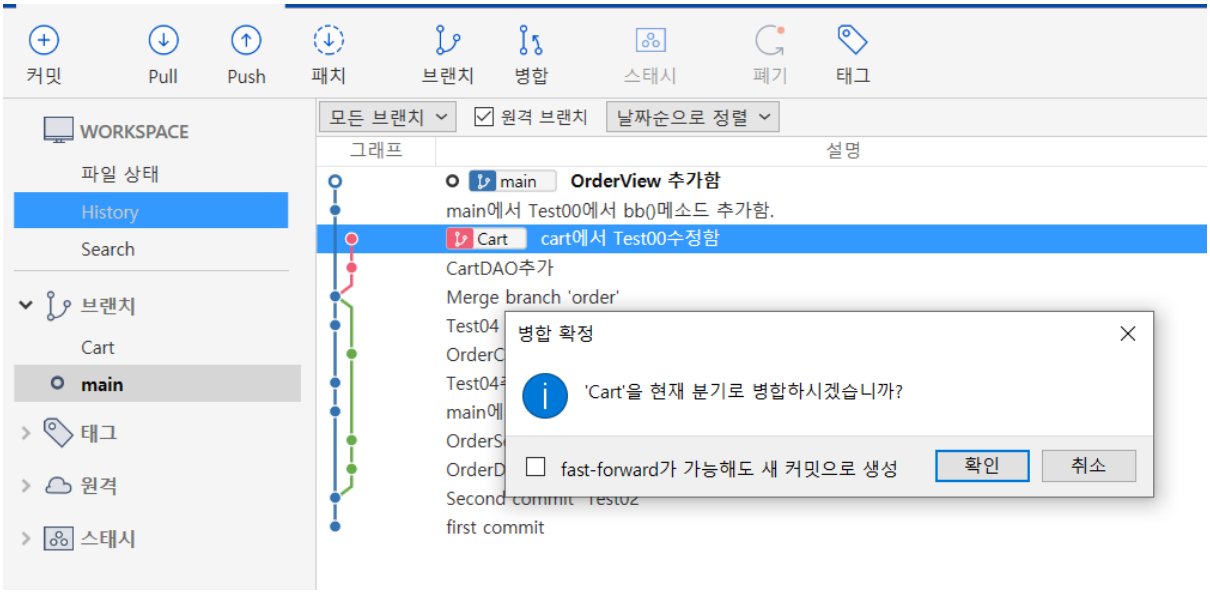
8253jang@daum.net
장희정

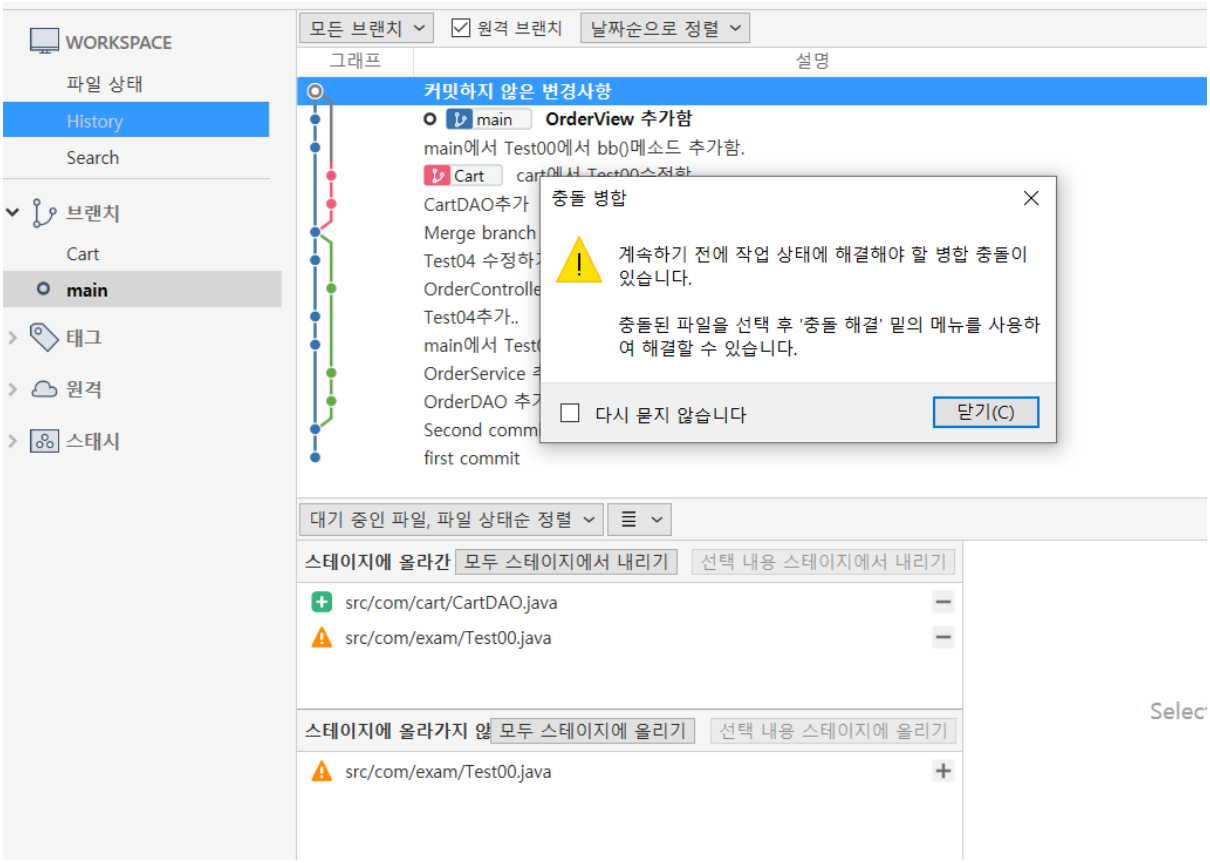


4) main에 cart를 merge한다.

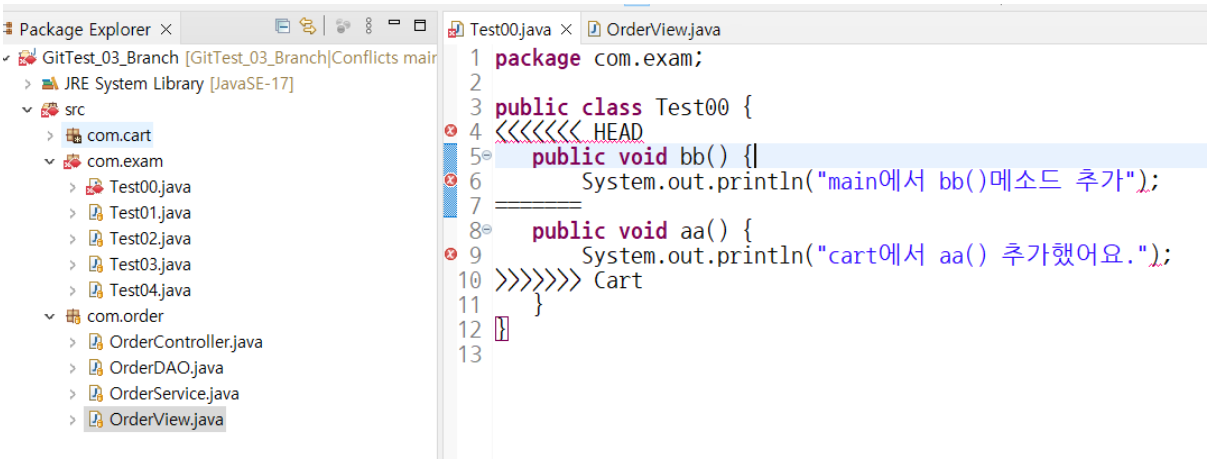


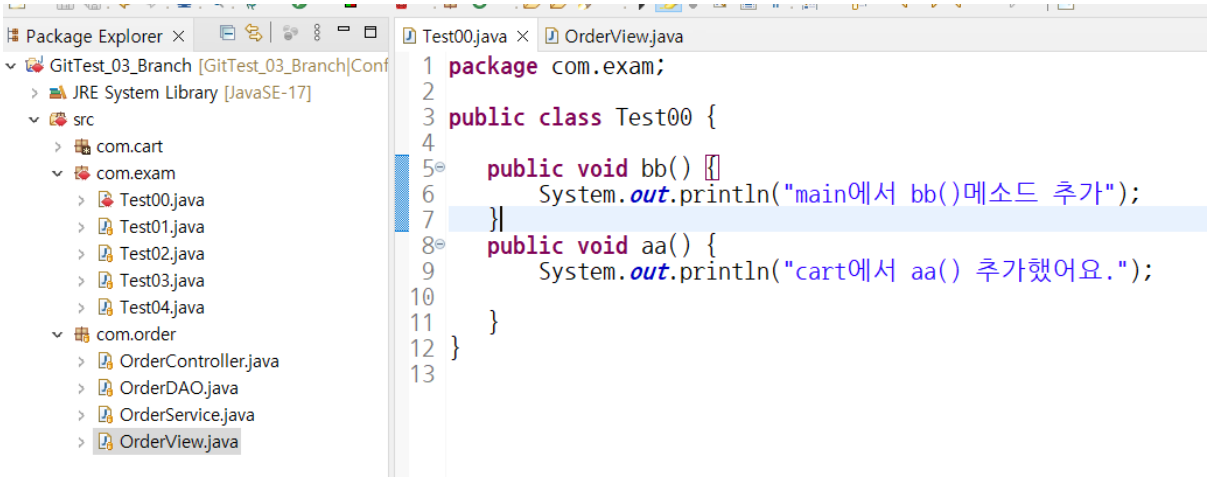
8253jang@daum.net
장희정

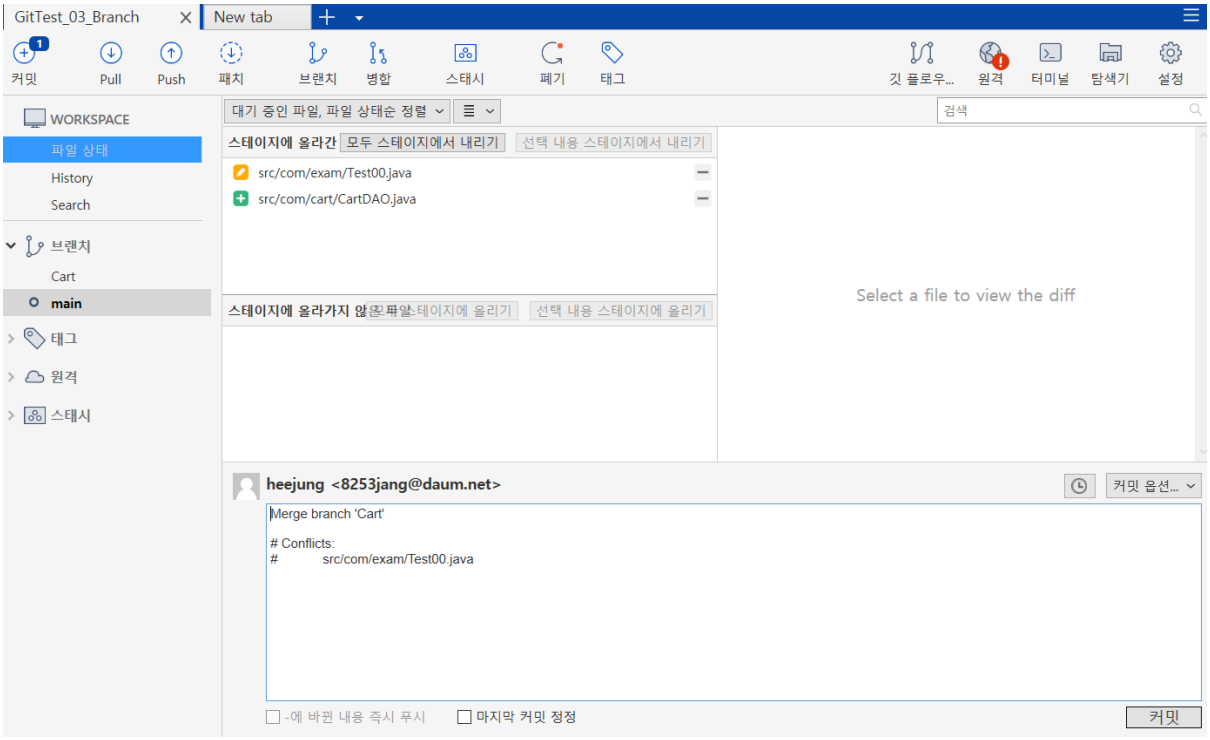




8253jang@daum.net
장희정







8253jang@daum.net
장희정