

9주차 실습 시트

강 환수 교수

프로젝트	stash & clean: 임시 저장과 비우기 [교재 7장] <input type="checkbox"/> 브랜치에서 임시저장에 WD, SA(index)를 저장한 후 필요하면 다시 적용(저장)													
모듈1	<input type="checkbox"/> 브랜치 main에서 1개 커밋 후, 수정 후 임시 저장, 내용 보기													
모듈2	<input type="checkbox"/> 임시저장을 다시 브랜치의 WD에 적용(저장) <input type="checkbox"/> 브랜치 main에 add 후 파일 main.py 수정 후 stash <input type="checkbox"/> 임시저장한 이후 git diff, git show													
모듈3	<input type="checkbox"/> 최근 임시저장을 지정해 적용(저장) <input type="checkbox"/> 한 파일은 indexed, 다른 파일은 modified 상태로 만들어 임시저장을 저장 후 적용													
모듈4	<input type="checkbox"/> 현재 WD를 임시저장에 저장, -m: 메시지 추가, 다시 임시저장을 적용 <input type="checkbox"/> 다시 임시저장을 적용하기 위해 이전과 같은 상태를 만들고 임시저장 후 다시 적용 <input type="checkbox"/> 임시저장 저장 시 옵션: --keep-index													
모듈5	<input type="checkbox"/> 추적되지 않은 파일을 임시 저장하기 <input type="checkbox"/> 브랜치를 만들고 임시저장을 적용하고 적용된 임시저장은 삭제 <input type="checkbox"/> 가장 최근의 임시저장을 적용하고 임시저장 목록에서 제거													
모듈6	<input type="checkbox"/> WD에서 추적되지 않는 파일(untracked) 제거하기													
준비	<input type="checkbox"/> 폴더 git/09w 하부에 저장소 stsh 생성 후 시작 <input type="checkbox"/> 참고 사이트 ○ https://gmlwjd9405.github.io/2018/05/18/git-stash.html													
모듈1 실습 (15분)	<div> <input type="checkbox"/> 브랜치 main에서 1개 커밋 후, 수정 후 임시 저장, 내용 보기 </div> <table border="1"> <thead> <tr> <th>커밋이력</th><th>A</th><th>B</th></tr> </thead> <tbody> <tr> <td>파일</td><td>main.py</td><td>main.py</td></tr> <tr> <td>메시지</td><td>add comp</td><td>커밋 전</td></tr> <tr> <td>내용</td><td>a = [i for i in range(1, 6)]</td><td>print(a)</td></tr> </tbody> </table> <div> <pre> PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/09wa \$ git init stsh Initialized empty Git repository in C:/OSS/git/09wa/stsh/.git/ PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/09wa \$ cd stsh PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/09wa/stsh (main) \$ echo 'a = [i for i in range(1, 6)]' > main.py \$ git add . \$ git commit -m 'add comp' [main 50a7f04] add comp 1 file changed, 1 insertion(+) create mode 100644 "20211234 \355\231\215\352\270\270\353\217\231/main.py" \$ echo 'print(a)' >> main.py PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/09wa/stsh (main) \$ git diff diff --git a/main.py b/main.py index ace5c45..cbf6f5f 100644 --- a/main.py +++ b/main.py @@ -1,2 @@ a = [i for i in range(1, 6)] </pre> </div>		커밋이력	A	B	파일	main.py	main.py	메시지	add comp	커밋 전	내용	a = [i for i in range(1, 6)]	print(a)
커밋이력	A	B												
파일	main.py	main.py												
메시지	add comp	커밋 전												
내용	a = [i for i in range(1, 6)]	print(a)												
	<p>첫 커밋 이력 생성</p>	<p>두 번째 수정 후 add 전에 다음 stash 수행</p> <p>git diff: SA와 WD 비교</p> <p>SA:</p> <pre>a = [i for i in range(1, 6)]</pre> <p>WD:</p> <pre>a = [i for i in range(1, 6)] +print(a)</pre> <p>git ls-files 두 번째 수정 후 add 전에 다음 stash 수행</p>												

모듈2 실습 (20분)	<pre>+print(a) \$ git ls-files -s 100644 ace5c455c255bfb7ae623e96b146291b44492adf 0 main.py</pre>																	
	<pre>\$ git stash # 임시 저장 Saved working directory and index state WIP on main: 50a7f04 add comp \$ git stash list # 목록 보기 stash@{0}: WIP on main: 50a7f04 add comp \$ git stash show # 현재 WD와 바로 저장된 임시저장의 비교 main.py 1 + 1 file changed, 1 insertion(+)</pre> <pre>\$ git stash show -p diff --git a/main.py b/main.py index ace5c45..cbf6f5f 100644 --- a/main.py +++ b/main.py @@ -1,2 @@ a = [i for i in range(1, 6)] +print(a) \$ git st On branch main Your branch is ahead of 'origin/main' by 1 commit. (use "git push" to publish your local commits) nothing to commit, working tree clean</pre>	<p>임시 저장 내용</p> <p>SA:</p> <pre>a = [i for i in range(1, 6)]</pre> <p>WD:</p> <pre>a = [i for i in range(1, 6)] +print(a)</pre> <p>stash show: 현재 WD와 바로 저장된 임시저장의 비교</p> <p>stash show -p: 자세한 파일 내용까지 비교</p> <p>stash 이후에는 SA와 WD는 동일하며 working tree clean</p>																
	<pre>\$ git diff</pre>	stash 이후에는 SA와 WD는 동일																
	<div><input type="checkbox"/> 임시저장을 다시 브랜치의 WD에 적용(저장) ○ apply 옵션: --index: 스테이지 영역도 저장, 나중에 실습</div>																	
모듈2 실습 (20분)	<pre>\$ git stash apply On branch main Your branch is ahead of 'origin/main' by 1 commit. (use "git push" to publish your local commits) 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: main.py no changes added to commit (use "git add" and/or "git commit -a") \$ git stash list stash@{0}: WIP on main: 50a7f04 add comp \$ git diff diff --git a/main.py b/main.py index ace5c45..cbf6f5f 100644 --- a/main.py +++ b/main.py @@ -1,2 @@ a = [i for i in range(1, 6)] +print(a)</pre>	<p>WD가 이전에</p> <pre>a = [i for i in range(1, 6)]</pre> <p>였다가 임시저장이 복사되어 다음이 됨</p> <pre>a = [i for i in range(1, 6)] +print(a)</pre> <p>git stash list로 임시저장은 여전히 남아 있음</p> <p>전과 같이 SA와 WD는 1줄의 차이가 있음</p>																
	<div><input type="checkbox"/> 브랜치 main에 add 후 파일 main.py 수정 후 stash</div> <table><tr><td>커밋이력</td><td>A</td><td>B</td><td>C</td></tr><tr><td>파일</td><td>main.py</td><td>main.py</td><td>main.py</td></tr><tr><td>메시지</td><td>add comp</td><td>커밋 전</td><td>커밋 전</td></tr><tr><td>내용</td><td>a = [i for i in range(1, 6)]</td><td>print(a)</td><td>print([10, 20, 30])</td></tr></table>		커밋이력	A	B	C	파일	main.py	main.py	main.py	메시지	add comp	커밋 전	커밋 전	내용	a = [i for i in range(1, 6)]	print(a)	print([10, 20, 30])
	커밋이력	A	B	C														
파일	main.py	main.py	main.py															
메시지	add comp	커밋 전	커밋 전															
내용	a = [i for i in range(1, 6)]	print(a)	print([10, 20, 30])															
<pre>\$ git add . # 수정 반영 \$ git st # 상태: 수정된 내용이 indexed On branch main Changes to be committed: (use "git restore --staged <file>..." to unstage)</pre>																		

	<pre> modified: main.py \$ echo 'print([10, 20, 30])' >> main.py # 다시 수정 \$ git st # 동일 파일 main.py이 indexed, modified On branch main Changes to be committed: (use "git restore --staged <file>..." to unstage) modified: main.py 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: main.py \$ git sts MM main.py \$ git diff # SA와 WD 비교: 마지막 한 줄 차이 diff --git a/main.py b/main.py index cbf6f5f..b1b7164 100644 --- a/main.py +++ b/main.py @@ -1,2 +1,3 @@ a = [i for i in range(1, 6)] print(a) +print([10, 20, 30]) \$ git diff --staged # HEAD와 SA 비교: 마지막 한 줄 차이 diff --git a/main.py b/main.py index ace5c45..cbf6f5f 100644 --- a/main.py +++ b/main.py @@ -1 +1,2 @@ a = [i for i in range(1, 6)] +print(a) \$ git diff head # HEAD와 WD 비교: 마지막 두 줄 차이 diff --git a/main.py b/main.py index ace5c45..b1b7164 100644 --- a/main.py +++ b/main.py @@ -1 +1,3 @@ a = [i for i in range(1, 6)] +print(a) +print([10, 20, 30]) </pre>	<p>현재 저장 내용</p> <p>SA:</p> <pre>a = [i for i in range(1, 6)] print(a)</pre> <p>WD:</p> <pre>a = [i for i in range(1, 6)] print(a) print([10, 20, 30])</pre> <p>HEAD:</p> <pre>a = [i for i in range(1, 6)]</pre>
	<pre> \$ git stash save # 임시저장, save 생략 가능, -m 메시지 추가 가능 Saved working directory and index state WIP on main: 50a7f04 add comp \$ git stash list # 임시저장 목록 stash@{0}: WIP on main: 50a7f04 add comp stash@{1}: WIP on main: 50a7f04 add comp </pre>	<p>임시 저장 내용</p> <p>SA:</p> <pre>a = [i for i in range(1, 6)] print(a)</pre> <p>WD:</p> <pre>a = [i for i in range(1, 6)] print(a) print([10, 20, 30])</pre> <p>임시저장 목록이 2개</p>
<div> <input type="checkbox"/> 임시저장한 이후 git diff, git show </div>		
	<pre> \$ git st On branch main Your branch is ahead of 'origin/main' by 1 commit. (use "git push" to publish your local commits) nothing to commit, working tree clean \$ cat main.py a = [i for i in range(1, 6)] \$ git diff \$ git diff --staged \$ git diff head </pre>	<p>임시 저장 이후: WD가 clean, 마지막 커밋 상태가 됨</p> <p>파일 내용도 마지막 커밋 내용</p> <p>임시저장 이후는 WD clean이므로 모두 차이 결과가 없음</p>

	<pre> \$ git stash list stash@{0}: WIP on main: 50a7f04 add comp stash@{1}: WIP on main: 50a7f04 add comp \$ git stash show -p diff --git a/main.py b/main.py index ace5c45..b1b7164 100644 --- a/main.py +++ b/main.py @@ -1,3 @@ a = [i for i in range(1, 6)] +print(a) +print([10, 20, 30]) \$ git stash show stash@{1} -p diff --git a/main.py b/main.py index ace5c45..cbf6f5f 100644 --- a/main.py +++ b/main.py @@ -1,3 @@ a = [i for i in range(1, 6)] +print(a) \$ git stash show stash@{0} -p diff --git a/main.py b/main.py index ace5c45..b1b7164 100644 --- a/main.py +++ b/main.py @@ -1,3 @@ a = [i for i in range(1, 6)] +print(a) +print([10, 20, 30]) </pre>	<p>임시저장 목록 보기</p> <p>stash show: 현재 WD와 바로 저장된 임시저장의 비교</p> <p>stash show -p: 자세한 파일 내용까지 비교</p> <p>stash show stash@{i}: 현재 WD와 지정된 임시저장의 비교</p> <p>stash show stash@{i} -p: 자세한 파일 내용까지 비교</p> <p>stash@{0}: 가장 최근 임시저장</p> <p>stash@{1}: 가장 최근 2번째 임시저장</p>
<p>모듈3 실습 (15분)</p>	<p><input type="checkbox"/> 최근 임시저장을 지정해 적용</p> <pre> \$ git stash apply # 가장 최근 임시저장을 WD에 반영 On branch main Your branch is ahead of 'origin/main' by 1 commit. (use "git push" to publish your local commits) 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: main.py no changes added to commit (use "git add" and/or "git commit -a") \$ git diff # SA와 WD 비교 diff --git a/main.py b/main.py index ace5c45..b1b7164 100644 --- a/main.py +++ b/main.py @@ -1,3 @@ a = [i for i in range(1, 6)] +print(a) +print([10, 20, 30]) \$ git diff --staged # HEAD와 SA 비교 \$ git diff head # HEAD와 WD 비교 diff --git a/main.py b/main.py index ace5c45..b1b7164 100644 --- a/main.py +++ b/main.py @@ -1,3 @@ a = [i for i in range(1, 6)] +print(a) +print([10, 20, 30]) </pre>	<p>임시저장을 적용하면 WD만 수정됨</p>
	<p><input type="checkbox"/> 한 파일은 indexed, 다른 파일은 modified 상태로 만들어 임시저장을 저장 후 적용</p> <ul style="list-style-type: none"> ○ main.py 수정된 것이 stage에 올리고, hello.py WD에 modified 되도록 ○ main.py는 수정해서 add, hello.py는 수정만 	

커밋이력	A	B	C	
파일	main.py	main.py	hello.py	main.py hello.py
메시지	add comp	print list	create hello.py	
내용	a = [i for i in range(1, 6)]	print(a) print([10, 20, 30])		print('list') print('hello')


```

$ git add .

$ git commit -m 'print list'
[main c9360f1] print list
1 file changed, 2 insertions(+)

$ git log
* c9360f1 (HEAD -> main) print list
* 50a7f04 add comp

$ touch hello.py # 빈 파일 생성

$ git add . # 모든 파일 stage에 올리고

$ git commit -m 'create hello.py' # 커밋
[main 0a3e097] create hello.py
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 hello.py

$ echo 'print("list")' >> main.py # 다시 파일 수정

$ git add main.py # 파일 stage에 올리고

$ echo 'print("hello")' >> hello.py # 다시 파일 수정

$ git st
On branch main
Changes to be committed:
(use "git restore --staged <file>..." to unstage)
modified:   main.py

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:   hello.py

```

main.py 수정된 것이 stage에 올리고
hello.py WD에 modified 되도록


```

$ git diff head
diff --git a/hello.py b/hello.py
index e69de29..11b15b1 100644
--- a/hello.py
+++ b/hello.py
@@ -0,0 +1 @@
+print("hello")
diff --git a/main.py b/main.py
index b1b7164..ef92a2f 100644
--- a/main.py
+++ b/main.py
@@ -1,3 +1,4 @@
a = [i for i in range(1, 6)]
print(a)
print([10, 20, 30])
+print("list")

$ git diff --staged
diff --git a/main.py b/main.py
index b1b7164..ef92a2f 100644
--- a/main.py
+++ b/main.py
@@ -1,3 +1,4 @@
a = [i for i in range(1, 6)]
print(a)
print([10, 20, 30])
+print("list")

$ git diff
diff --git a/hello.py b/hello.py
index e69de29..11b15b1 100644

```

현재 상태

SA:

```

hello.py
"
main.py
a = [i for i in range(1, 6)]
print(a)
print([10, 20, 30])
+print("list")

```

WD:

```

hello.py
+print("hello")
,
main.py
a = [i for i in range(1, 6)]
print(a)
print([10, 20, 30])
+print("list")

```

HEAD:

```

hello.py
"
main.py
a = [i for i in range(1, 6)]
print(a)

```

	<pre> --- a/hello.py +++ b/hello.py @@ -0,0 +1 @@ +print("hello") </pre>	<pre> print([10, 20, 30]) </pre>
<div> <div>모듈4</div> <div>실습</div> <div>(15분)</div> </div>	<div> <input type="checkbox"/> 현재 WD를 임시저장에 저장, -m: 메시지 추가, 다시 임시저장을 적용 <ul style="list-style-type: none"> ○ 일반적으로 index에는 저장이 않됨 ○ 옵션 --index: 임시저장이 index로 되어있던 것(modified)은 다시 stage에도 반영 </div>	
	<pre> \$ git stash -m 'indexed and modified files' Saved working directory and index state On main: indexed and modified files \$ git stash list stash@{0}: On main: indexed and modified files stash@{1}: WIP on main: 50a7f04 add comp stash@{2}: WIP on main: 50a7f04 add comp \$ git st On branch main Your branch is ahead of 'origin/main' by 3 commits. (use "git push" to publish your local commits) nothing to commit, working tree clean \$ git diff \$ git diff head \$ git diff --staged </pre>	
	<pre> \$ git stash apply 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: hello.py modified: main.py no changes added to commit (use "git add" and/or "git commit -a") \$ git diff diff --git a/hello.py b/hello.py index e69de29..11b15b1 100644 --- a/hello.py +++ b/hello.py @@ -0,0 +1 @@ +print("hello") diff --git a/main.py b/main.py index b1b7164..ef92a2f 100644 --- a/main.py +++ b/main.py @@ -1,3 +1,4 @@ a = [i for i in range(1, 6)] print(a) print([10, 20, 30]) +print("list") </pre>	<p>apply에서 일반적으로 index에는 저장이 않됨 그러므로 이전의 상태와 다름</p> <p>Changes to be committed: (use "git restore --staged <file>..." to unstage) modified: main.py</p> <p>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: hello.py</p>
	<div> <input type="checkbox"/> 다시 임시저장을 적용하기 위해 이전과 같은 상태를 만들고 임시저장 후 다시 적용 <ul style="list-style-type: none"> ○ 옵션 --index: 임시저장이 index로 되어있던 것(modified)은 다시 stage에도 반영 </div>	
	<pre> \$ git add main.py \$ git st On branch main Your branch is ahead of 'origin/main' by 3 commits. (use "git push" to publish your local commits) Changes to be committed: (use "git restore --staged <file>..." to unstage) modified: main.py Changes not staged for commit: (use "git add <file>..." to update what will be committed) </pre>	

	<pre>(use "git restore <file>..." to discard changes in working directory) modified: hello.py \$ git stash Saved working directory and index state WIP on main: 0a3e097 create hello.py \$ git stash list stash@{0}: WIP on main: 0a3e097 create hello.py stash@{1}: On main: indexed and modified files stash@{2}: WIP on main: 50a7f04 add comp stash@{3}: WIP on main: 50a7f04 add comp \$ git stash apply --index On branch main Changes to be committed: (use "git restore --staged <file>..." to unstage) modified: main.py 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: hello.py</pre>	
<div> <input type="checkbox"/> 임시저장 저장 시 옵션: --keep-index <ul style="list-style-type: none"> ○ 현재 상태에서 staging area는 그대로 남기고 WD만 임시저장에 저장, 즉 staging area는 임시저장에 제외하고 저장 </div>		
	<pre>\$ git sts M hello.py M main.py \$ git stash --keep-index Saved working directory and index state WIP on main: 0a3e097 create hello.py \$ git sts M main.py \$ git stash list stash@{0}: WIP on main: 0a3e097 create hello.py stash@{1}: WIP on main: 0a3e097 create hello.py stash@{2}: On main: indexed and modified files stash@{3}: WIP on main: 50a7f04 add comp stash@{4}: WIP on main: 50a7f04 add comp</pre>	<p>index는 저장하지 않으므로 그대로 Modified인 것을 알 수 있음</p>
	<pre>\$ git stash apply On branch main Changes to be committed: (use "git restore --staged <file>..." to unstage) modified: main.py 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: hello.py \$ git sts M hello.py M main.py</pre>	<p>다시 임시저장 적용</p>
<div> <input type="checkbox"/> 추적되지 않은 파일을 임시 저장하기 </div>		
<p>모듈5 실습 (20분)</p>	<pre>\$ git stash list stash@{0}: WIP on main: 0a3e097 create hello.py stash@{1}: WIP on main: 0a3e097 create hello.py stash@{2}: On main: indexed and modified files stash@{3}: WIP on main: 50a7f04 add comp stash@{4}: WIP on main: 50a7f04 add comp \$ touch simple \$ ls hello.py main.py simple</pre>	<p>새로 생성된 파일(Untracked files)은 임시저장이 기본적으로 않됨</p> <p>옵션 --include-untracked, -u</p>

	<pre>\$ git sts M hello.py M main.py ?? simple \$ git stash --include-untracked Saved working directory and index state WIP on main: 0a3e097 create hello.py \$ git stash list stash@{0}: WIP on main: 0a3e097 create hello.py stash@{1}: WIP on main: 0a3e097 create hello.py stash@{2}: WIP on main: 0a3e097 create hello.py stash@{3}: On main: indexed and modified files stash@{4}: WIP on main: 50a7f04 add comp stash@{5}: WIP on main: 50a7f04 add comp</pre>	
<div> <input type="checkbox"/> 브랜치를 만들고 임시저장을 적용하고 적용된 임시저장은 삭제 </div>		
	<pre>\$ git stash branch sbrch1 Switched to a new branch 'sbrch1' On branch sbrch1 Changes to be committed: (use "git restore --staged <file>..." to unstage) modified: main.py 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: hello.py Untracked files: (use "git add <file>..." to include in what will be committed) simple Dropped refs/stash@{0} (315eb151b45a4d3776d8370786266799cf1eeff0) PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/09wc/gstash (sbrch1) \$ git branch main * sbrch1 \$ git lgg1 * 0ae0ce8 (HEAD -> sbrch1, main) create hello.py * de8ad83 print list * 0a6132b add comp \$ ls hello.py main.py simple \$ git stash list stash@{0}: WIP on main: 0a3e097 create hello.py stash@{1}: WIP on main: 0a3e097 create hello.py stash@{2}: On main: indexed and modified files stash@{3}: WIP on main: 50a7f04 add comp stash@{4}: WIP on main: 50a7f04 add comp</pre>	\$ git stash branch bname
	<pre>PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/09wc/gstash (sbrch1) \$ git switch main Switched to branch 'main' M hello.py M main.py PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/09wc/gstash (main) \$ git sts M hello.py M main.py ?? simple PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/09wc/gstash (main) \$ git lgg1 * 0ae0ce8 (HEAD -> main, sbrch1) create hello.py * de8ad83 print list * 0a6132b add comp PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/09wc/gstash (main) \$ ls hello.py main.py simple PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/09wc/gstash (main) \$ git stash list</pre>	<p>main으로 돌아와서</p> <p>다시 한번</p>

	<pre> stash@{0}: WIP on main: 0ae0ce8 create hello.py stash@{1}: WIP on main: 0ae0ce8 create hello.py stash@{2}: On main: indexed and modified files stash@{3}: WIP on main: 0a6132b add comp stash@{4}: WIP on main: 0a6132b add comp PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/09wc/gstash (main) \$ git stash --include-untracked Saved working directory and index state WIP on main: 0ae0ce8 create hello.py PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/09wc/gstash (main) \$ git stash list stash@{0}: WIP on main: 0ae0ce8 create hello.py stash@{1}: WIP on main: 0ae0ce8 create hello.py stash@{2}: WIP on main: 0ae0ce8 create hello.py stash@{3}: On main: indexed and modified files stash@{4}: WIP on main: 0a6132b add comp stash@{5}: WIP on main: 0a6132b add comp PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/09wc/gstash (main) \$ git stash branch sbrch2 Switched to a new branch 'sbrch2' On branch sbrch2 Changes to be committed: (use "git restore --staged <file>..." to unstage) modified: main.py 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: hello.py Untracked files: (use "git add <file>..." to include in what will be committed) simple Dropped refs/stash@{0} (92699b74f30fd9f9ecfc27160742e81fc2afd2a6) PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/09wc/gstash (sbrch2) \$ git stash list stash@{0}: WIP on main: 0ae0ce8 create hello.py stash@{1}: WIP on main: 0ae0ce8 create hello.py stash@{2}: On main: indexed and modified files stash@{3}: WIP on main: 0a6132b add comp stash@{4}: WIP on main: 0a6132b add comp PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/09wc/gstash (sbrch2) \$ git branch main sbrch1 * sbrch2 </pre>	
<div> <input type="checkbox"/> 가장 최근의 임시저장을 적용하고 임시저장 목록에서 제거 </div>		
	<pre> PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/09wc/gstash (sbrch2) \$ git st On branch sbrch2 Changes to be committed: (use "git restore --staged <file>..." to unstage) modified: main.py 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: hello.py Untracked files: (use "git add <file>..." to include in what will be committed) simple PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/09wc/gstash (sbrch2) \$ git stash Saved working directory and index state WIP on sbrch2: 0ae0ce8 create hello.py PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/09wc/gstash (sbrch2) \$ git stash list stash@{0}: WIP on sbrch2: 0ae0ce8 create hello.py stash@{1}: WIP on main: 0ae0ce8 create hello.py stash@{2}: WIP on main: 0ae0ce8 create hello.py stash@{3}: On main: indexed and modified files stash@{4}: WIP on main: 0a6132b add comp stash@{5}: WIP on main: 0a6132b add comp </pre>	<pre> \$ git stash pop </pre>

	<pre>PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/09wc/gstash (sbrch2) \$ git stash pop On branch sbrch2 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: hello.py modified: main.py Untracked files: (use "git add <file>..." to include in what will be committed) simple no changes added to commit (use "git add" and/or "git commit -a") Dropped refs/stash@{0} (51ed292dd4768fb3896ebaeab379e37a0cc9bf70) PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/09wc/gstash (sbrch2) \$ git stash list stash@{0}: WIP on main: 0ae0ce8 create hello.py stash@{1}: WIP on main: 0ae0ce8 create hello.py stash@{2}: On main: indexed and modified files stash@{3}: WIP on main: 0a6132b add comp stash@{4}: WIP on main: 0a6132b add comp</pre>	
	<pre>\$ git stash -h usage: git stash list [<options>] or: git stash show [<options>] [<stash>] or: git stash drop [-q --quiet] [<stash>] or: git stash (pop apply) [--index] [-q --quiet] [<stash>] or: git stash branch <branchname> [<stash>] or: git stash clear or: git stash [push [-p --patch] [-S --staged] [-k --[no-]keep-index] [-q --quiet] [-u --include-untracked] [-a --all] [-m --message <message>] [--pathspec--from-file=<file> [--pathspec-file-nul]] [--] [<pathspec>...] or: git stash save [-p --patch] [-S --staged] [-k --[no-]keep-index] [-q --quiet] [-u --include-untracked] [-a --all] [<message>]</pre> <pre>PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/09wc/gstash (sbrch2) \$ git stash list stash@{0}: WIP on main: 0ae0ce8 create hello.py stash@{1}: WIP on main: 0ae0ce8 create hello.py stash@{2}: On main: indexed and modified files stash@{3}: WIP on main: 0a6132b add comp stash@{4}: WIP on main: 0a6132b add comp PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/09wc/gstash (sbrch2) \$ git stash drop Dropped refs/stash@{0} (be3fb221c4f160229e6df9dad2c3370791403832) PC@DESKTOP-482NOAB MINGW64 /c/OSS/git/09wc/gstash (sbrch2) \$ git stash list stash@{0}: WIP on main: 0ae0ce8 create hello.py stash@{1}: On main: indexed and modified files stash@{2}: WIP on main: 0a6132b add comp stash@{3}: WIP on main: 0a6132b add comp</pre>	<p>가장 최근 임시저장 삭제 \$ git stash drop</p> <p>지정된 임시저장 삭제 \$ git stash drop stash@{2}</p>
<div>모듈6</div> <div>실습</div> <div>(10분)</div>	<div>❑ WD에서 추적되지 않는 파일(untracked) 제거하기</div>	
	<pre>\$ ls hello.py init.md main.py \$ touch a b \$ ls a b hello.py init.md main.py \$ git st On branch sbrch2 Untracked files: (use "git add <file>..." to include in what will be committed) a b nothing added to commit but untracked files present (use "git add" to track)</pre>	<p>\$ git clean -f</p>

	<pre>\$ git clean fatal: clean.requireForce defaults to true and neither -i, -n, nor -f given; refusing to clean \$ git clean -f Removing a Removing b \$ ls hello.py init.md main.py</pre>	
	<pre>\$ touch a b c \$ git clean -n Would remove a Would remove b Would remove c \$ ls a b c hello.py init.md main.py \$ git clean -i Would remove the following items: a b c *** Commands *** 1: clean 2: filter by pattern 3: select by numbers 4: ask each 5: quit 6: help What now> a Remove a [y/N]? y Remove b [y/N]? n Remove c [y/N]? y Removing a Removing c \$ ls b hello.py init.md main.py</pre>	<p>dry run: 먼저 알아보기 \$ git clean -n</p> <p>물어 보기 \$ git clean -i</p>
	<pre>\$ git stash list stash@{0}: WIP on sbrch2: 0a3e097 create hello.py stash@{1}: On main: indexed and modified files stash@{2}: WIP on main: 50a7f04 add comp stash@{3}: WIP on main: 50a7f04 add comp \$ git stash clear \$ git stash list</pre>	<p>stash 모두 제거</p>