9주차 실습 시트

강 환수 교수

프로젝트	**stash & clean: 임시 저장과 비우기 [교재 7장]** ☐ 브랜치에서 임시저장에 WD, SA(index)를 저장한 후 필요하면 다시 적용(저장)
모듈1	☐ 브랜치 main에서 1개 커밋 후, 수정 후 임시 저장, 내용 보기
모듈2	☐ 임시저장을 다시 브랜치의 WD에 적용(저장) ☐ 브랜치 main에 add 후 파일 main.py 수정 후 stash ☐ 임시저장한 이후 git diff, git show
모듈3	☐ 최근 임시저장을 지정해 적용(저장) ☐ 한 파일은 indexed, 다른 파일은 modified 상태로 만들어 임시저장을 저장 후 적용
모듈4	☐ 현재 WD를 임시저장에 저장, -m: 메시지 추가, 다시 임시저장을 적용 ☐ 다시 임시저장을 적용하기 위해 이전과 같은 상태를 만들고 임시저장 후 다시 적용 ☐ 임시저장 저장 시 옵션: --keep-index
모듈5	☐ 추적되지 않은 파일을 임시 저장하기 ☐ 브랜치를 만들고 임시저장을 적용하고 적용된 임시저장은 삭제 ☐ 가장 최근의 임시저장을 적용하고 임시저장 목록에서 제거
모듈6	☐ WD에서 추적되지 않는 파일(untracked) 제거하기
준비	☐ 폴더 git/09w 하부에 저장소 stsh 생성 후 시작 ☐ 참고 사이트 ❍ https://gmlwjd9405.github.io/2018/05/18/git-stash.html

모듈1 실습 (15분)

☐ 브랜치 main에서 1개 커밋 후, 수정 후 임시 저장, 내용 보기

커밋이력	A	B
파일	main.py	main.py
메시지	add comp	커밋 전
내용	a = [i for i in range(1, 6)]	print(a)

```
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 +1,2 @@
 a = [i for i in range(1, 6)]
```

첫 커밋 이력 생성

두 번째 수정 후 add 전에 다음 stash 수행

git diff: SA와 WD 비교

SA:

```
 a = [i for i in range(1, 6)]
```

WD:

```
 a = [i for i in range(1, 6)]
+print(a)
```

git ls-files두 번째 수정 후 add 전에 다음 stash 수행

```
+print(a)

$ git ls-files -s
100644        ace5c455c255bfb7ae623e96b146291b44492adf        0
main.py
```

```
$ 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(+)

$ git stash show -p
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 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
```

임시 저장 내용

SA:

```
 a = [i for i in range(1, 6)]
```

WD:

```
 a = [i for i in range(1, 6)]
+print(a)
```

stash show: 현재 WD와 바로 저장된 임시저장의 비교

stash show –p: 자세한 파일 내용까지 비교

stash 이후에는 SA와 WD는 동일하며

working tree clean

```
$ git diff
```

stash 이후에는 SA와 WD는 동일

모듈2 실습 (20분)

□ 임시저장을 다시 브랜치의 WD에 적용(저장)

○ apply 옵션: --index: 스테이지 영역도 저장

```
$ 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 +1,2 @@
 a = [i for i in range(1, 6)]
+print(a)
```

WD가 이전에

```
 a = [i for i in range(1, 6)]
```

였다가 임시저장이 복사되어 다음이 됨

```
 a = [i for i in range(1, 6)]
+print(a)
```

git stash list로 임시저장은 여전히 남아 있음

전과 같이 SA와 WD는 1줄의 차이가 있음

□ 브랜치 main에 add 후 파일 main.py 수정 후 stash

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])

```
$ git add .                # 수정 반영

$ git st                   # 상태: 수정된 내용이 indexed
On branch main
Changes to be committed:
(use "git restore --staged <file>..." to unstage)
```

```
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])
```

현재 저장 내용

SA:

```
 a = [i for i in range(1, 6)]
print(a)
```

WD:

```
 a = [i for i in range(1, 6)]
print(a)
print([10, 20, 30])
```

HEAD:

```
 a = [i for i in range(1, 6)]
```

```
$ 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
```

임시 저장 내용

SA:

```
 a = [i for i in range(1, 6)]
print(a)
```

WD:

```
 a = [i for i in range(1, 6)]
print(a)
print([10, 20, 30])
```

임기저장 목록이 2개

□ 임시저장한 이후 git diff, git show

```
$ 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
```

임시 저장 이후: WD가 clean, 마지막 커밋 상태가 됨

파일 내용도 마지막 커밋 내용

임시저장 이후는 WD clean이므로 모두 차이 결과가 없음

	$ 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 +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 +1,2 @@
	 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 +1,3 @@
	 a = [i for i in range(1, 6)]
	+print(a)
	+print([10, 20, 30])
		임시저장 목록 보기
		
		stash show: 현재 WD와 바로 저장된 임시저장의 비교
		stash show –p: 자세한 파일 내용까지 비교
		
		stash show stash@{i}: 현재 WD와 지정된 임시저장의 비교
		stash show stash@{i} –p: 자세한 파일 내용까지 비교
		
		stash@{0}: 가장 최근 임시저장
		stash@{1}: 가장 최근 2번째 임시저장

모듈3 실습 (15분)	□ 최근 임시저장을 지정해 적용	

	$ 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 +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 +1,3 @@
	 a = [i for i in range(1, 6)]
	+print(a)
	+print([10, 20, 30])
		임시저장을 적용하면 WD만 수정됨

	□ 한 파일은 indexed, 다른 파일은 modified 상태로 만들어 임시저장을 저장 후 적용 ○ 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 logg1
* 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 amain.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)
```

	--- a/hello.py +++ b/hello.py @@ -0,0 +1 @@ +print("hello")	print([10, 20, 30])
모듈4 실습 (15분)	□ 현재 WD를 임시저장에 저장, -m: 메시지 추가, 다시 임시저장을 적용 ❍ 일반적으로 index에는 저장이 않됨 ❍ 옵션 --index: 임시저장이 index로 되어있던 것(modified)은 다시 stage에도 반영	

	$ 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	

	$ 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")
		apply에서 일반적으로 index에는 저장이 않됨
		그러므로 이전의 상태와 다름
		
		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

	□ 다시 임시저장을 적용하기 위해 이전과 같은 상태를 만들고 임시저장 후 다시 적용 ❍ 옵션 --index: 임시저장이 index로 되어있던 것(modified)은 다시 stage에도 반영	

	$ 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)	

	(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
	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)
	(use "git restore <file>..." to discard changes in working directory)
	modified: hello.py	

	□ 임시저장 저장 시 옵션: --keep-index ❍ 현재 상태에서 staging area는 그대로 남기고 WD만 임시저장에 저장, 즉 staging area는 임시저장에 제외하고 저장	

	$ 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	

	$ git stash apply
	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)
	(use "git restore <file>..." to discard changes in working directory)
	modified: hello.py
	
	$ git sts
	M hello.py
	M main.py
		다시 임시저장 적용

모듈5 실습 (20분)	□ 추적되지 않은 파일을 임시 저장하기	
	$ 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	새로 생성된 파일(Untracked files)은 임시저장이 기본적으로 않됨 옵션 --include-untracked, -u

```
$ touch simple

$ git stash
No local changes to save

$ git st
On branch main
Your branch is ahead of 'origin/main' by 3 commits.
(use "git push" to publish your local commits)

Untracked files:
(use "git add <file>..." to include in what will be committed)
        simple

nothing added to commit but untracked files present (use "git add" to track)

$ 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
```

□ 브랜치를 만들고 임시저장을 적용하고 적용된 임시저장은 삭제

```
$ git stash branch sbrch1
Switched to a new branch 'sbrch1'
Already up to date.
On branch sbrch1
Untracked files:
(use "git add <file>..." to include in what will be committed)
        simple

nothing added to commit but untracked files present (use "git add" to track)
Dropped                                         refs/stash@{0} (982b9642dc70c73357ebdfbb6ca866fa8a4925a1)

$ git branch
main
* sbrch1

$ git logg1
* 0a3e097 (HEAD -> sbrch1, main) create hello.py
* c9360f1 print list
* 50a7f04 add comp
* 94272b5 (origin/main, origin/HEAD) Create init.md
* dae2f78 Initial commit

$ ls
hello.py  init.md  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
```

$ git stash branch bname

```
$ 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
```

다시 한번

```
Untracked files:
(use "git add <file>..." to include in what will be committed)
simple

Dropped                                                refs/stash@{0}
(77a5e67b0f1263141887eee8bfc8f740f99e8c7a)

$ 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 branch
main
sbrch1
* sbrch2
```

□ 가장 최근의 임시저장을 적용하고 임시저장 목록에서 제거

```
$ 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


$ git stash
Saved working directory and index state WIP on sbrch2: 0a3e097
create hello.py

$ git stash list
stash@{0}: WIP on sbrch2: 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

$ 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}
(3197eff12bfdf6650a9ed759a7490bbe3feb0fca)

$ 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 pop

```
$ 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]
```

가장 최근 임시저장 삭제
$ git stash drop

지정된 임시저장 삭제
$ git stash drop stash@{2}

```
[-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>]


$ 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 drop
Dropped                                          refs/stash@{0}
(6cd4a858eaad9a9dbfca689ffac0d15cab3ad150)

$ 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
```

```
$ ls
hello.py  init.md  main.py  simple

$ git st
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")

$ git add main.py

$ git sts
M hello.py
M  main.py
?? simple

$ git stash
Saved working directory and index state WIP on sbrch2: 0a3e097
create hello.py

$ ls
hello.py  init.md  main.py  simple

$ git sts
?? simple

$ 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}
(adcef3e864dc3824d6b7053347d9b5c236b06e5d)

PC@DESKTOP-482NOAB                               MINGW64
/c/OSS/OSS-training/20211234 홍길동 (sbrch2)
$ 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

PC@DESKTOP-482NOAB                                    MINGW64
/c/OSS/OSS-training/20211234 홍길동 (sbrch2)
$ git stash -u
Saved working directory and index state WIP on sbrch2: 0a3e097
create hello.py

$ git sts

$ ls
hello.py   init.md   main.py

$ 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
```

모듈6
실습
(10분)

□ WD에서 추적되지 않는 파일(untracked) 제거하기

```
$ 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)

$ 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
```

$ git clean -f

```
$ 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
```

dry run: 먼저 알아보기
$ git clean –n

물어 보기
$ git clean -i

	$ 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
		stash 모두 제거