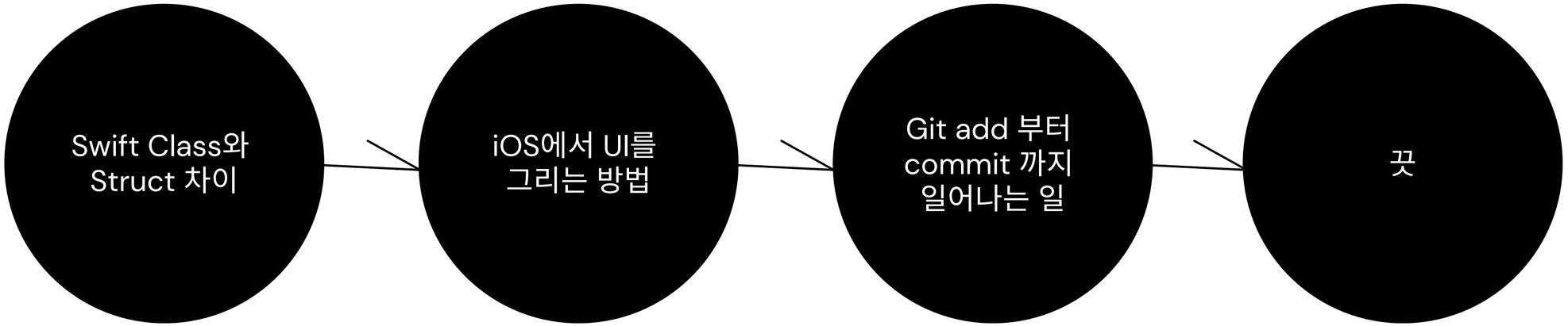
iOS 재미있는 사실과 Git의 동작.

김민석(M)

발표는 이 순서대로 진행할게요.



Referene Type?
Value Type?

명령형과 선언형이 뭘까요?

.git 파일이 뭐죠?

□들어주셔서 감사합니다!

Swift Class와 Struct 차이

Swift Class와 Struct 차이

Swift 타입 선언 방법

Class

참조 타입

Struct

값 타입

Enum

값 타입

Swift 이전 언어인
Object-C 등에서 사용

현재 Swift에서 70~90% 정도

Swift Class와 Struct 차이

Value Types

```
let origin = CGPoint(x: 0, y: 0)
var other = origin
other.x += 10
```

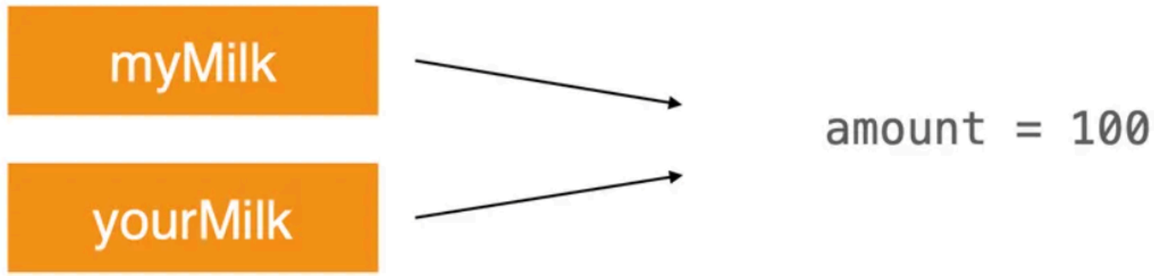
Call by value

Swift Class와 Struct 차이

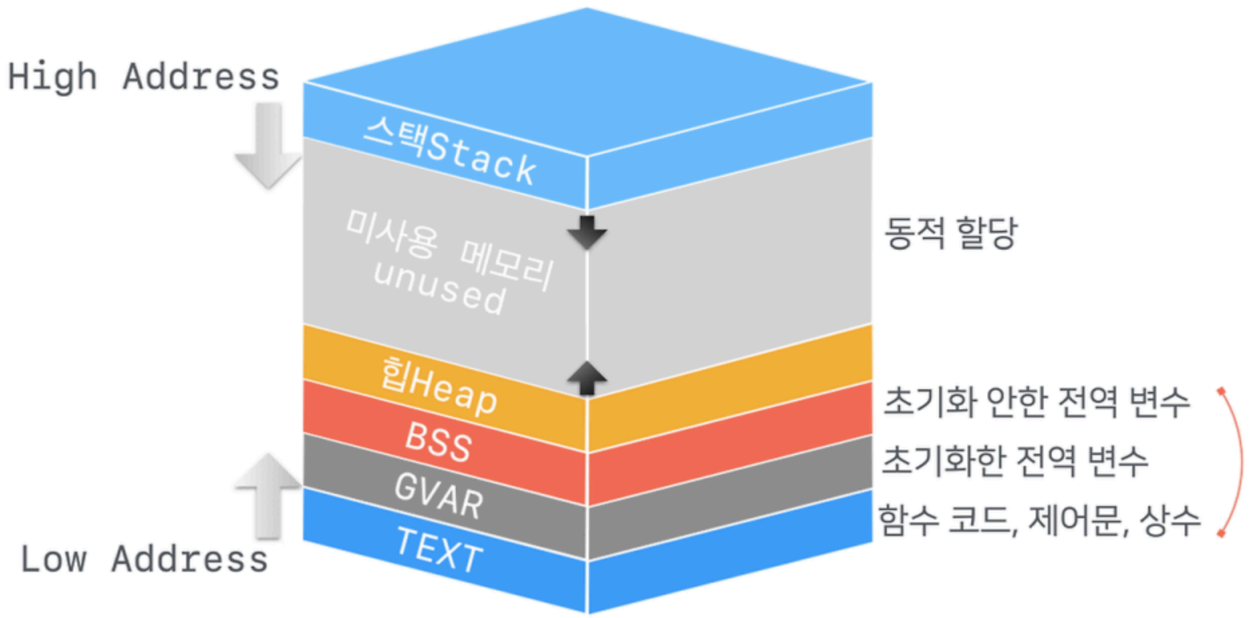
Reference Types

```
var myMilk = ChocoMilk()
myMilk.amount = 300
var yourMilk = myMilk
yourMilk.amount = 100
print(myMilk.amount)
```

Call by reference



Swift Class와 Struct 차이



Swift 메모리 구조

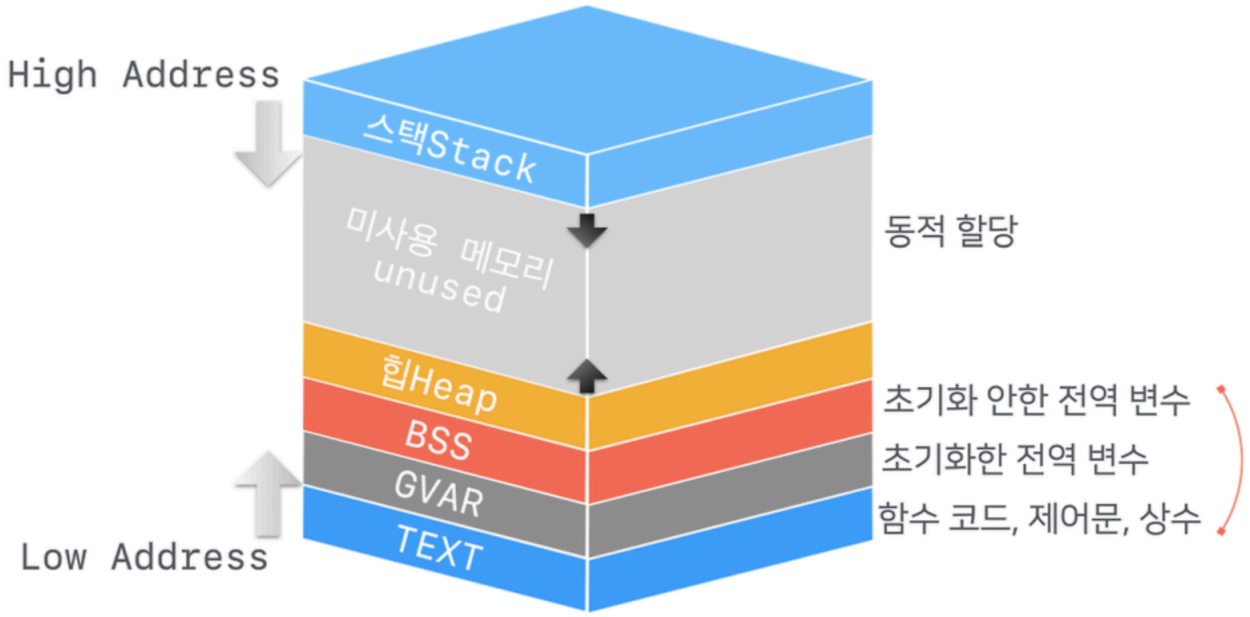
Text 영역

- 기계어로 변경된 코드(실행 가능한 코드)가 저장되는 곳
- 프로그램이 실행될 때 메모리에 로드되며 종료 시 메모리에서 제거

Data 영역

- 전역 변수와 정적 변수(static variables) 상수가 저장되는 곳
- Code 영역과 같이 프로그램이 실행될 때 메모리에 로드되며 종료 시 메모리에서 제거

Swift Class와 Struct 차이



Swift 메모리 구조

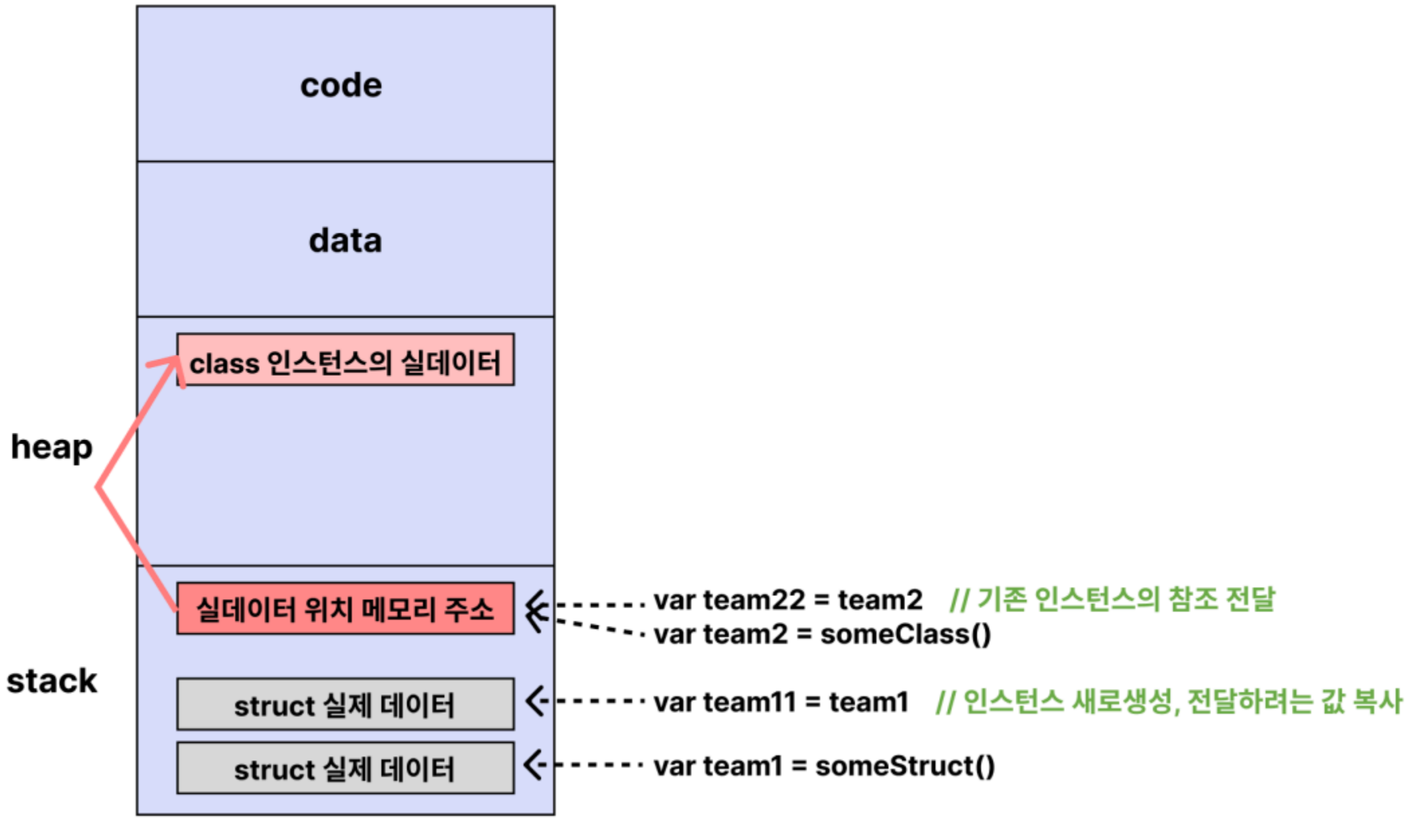
Stack 영역

- 지역 변수와 함수의 매개변수가 저장되는 곳
- 메소드가 종료되면 메모리 해제
- 몇몇 예외상황을 빼고는 Swift의 Value Type이 저장됨

Heap 영역

- 동적으로 할당되는 데이터
- (e.g. 클래스 인스턴스, 배열과 같은 데이터 구조)가 저장되는 곳
- 이 곳에 저장되는 객체는 lifetime을 가짐
- ARC로 메모리 관리가 됨

Swift Class와 Struct 차이



Swift Class와 Struct 차이

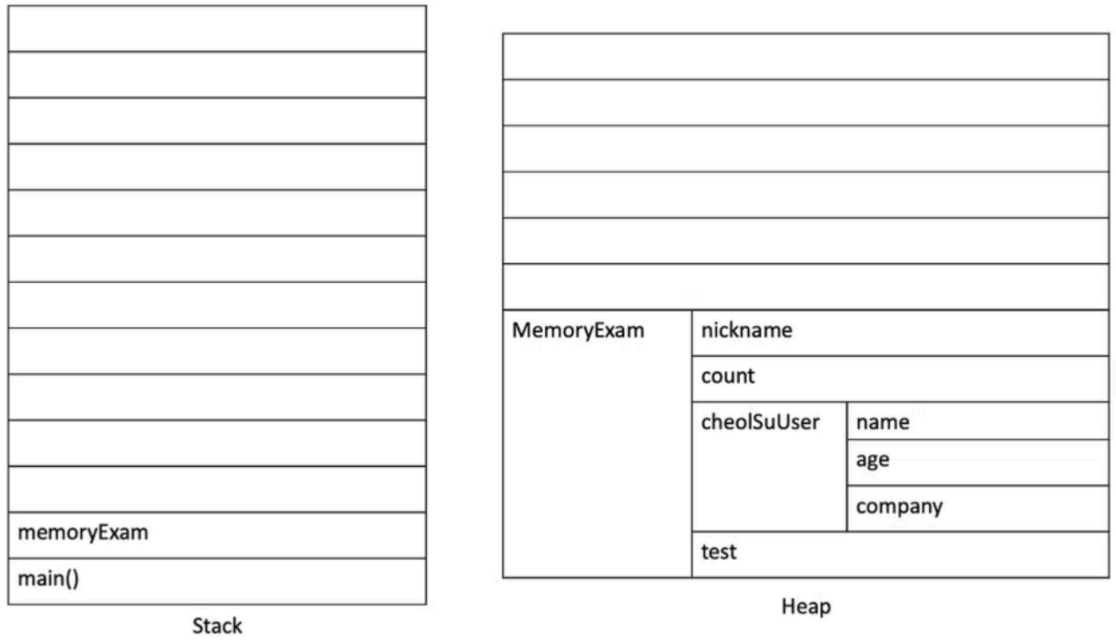
```swift
func main() {
    let memoryExam = MemoryExam()  // 스택에 추가
}

main() // 스택에 추가

class MemoryExam { // 힙에 추가
    let nickname = "shark"
    let count = 10
    let cheolSuUser = User(name: "철수", age: 15, company: "구글")
    let test = Test()

    init() {
        run()
    }

    func run() {
        let youngHeeUser = User(name: "영희", age: 30, company: "애
플")     let copyCheolSuUser = cheolSuUser
        let copyTest = test
    }
}
```




```swift
// 전역 변수로 데이터(GVAR) 영역에 할당 됨
var age: Int = 9 // GVAR
var name: String // BSS

func addNum(_ num1: Int, _ num2: Int) ->
Int{var result = num1 + age

    return result
}
```

Swift Class와 Struct 차이

https://www.slideshare.net/slideshow/ss-63881606/63881606#7

본인이 사용하는 프레임워크 본인이 사용하는 언어 이해를 하고 시작하자 본인이 사용하는 프레임워크 본인이 사용하는 언어 이해를 하고 시작하자

그래서 제가 전하고 싶은 내용은

본인이 사용하는 프레임워크 본인이 사용하는 언어 이해를 하고 시작하자 본인이 사용하는 프레임워크 본인이 사용하는 언어 이해를 하고 시작하자

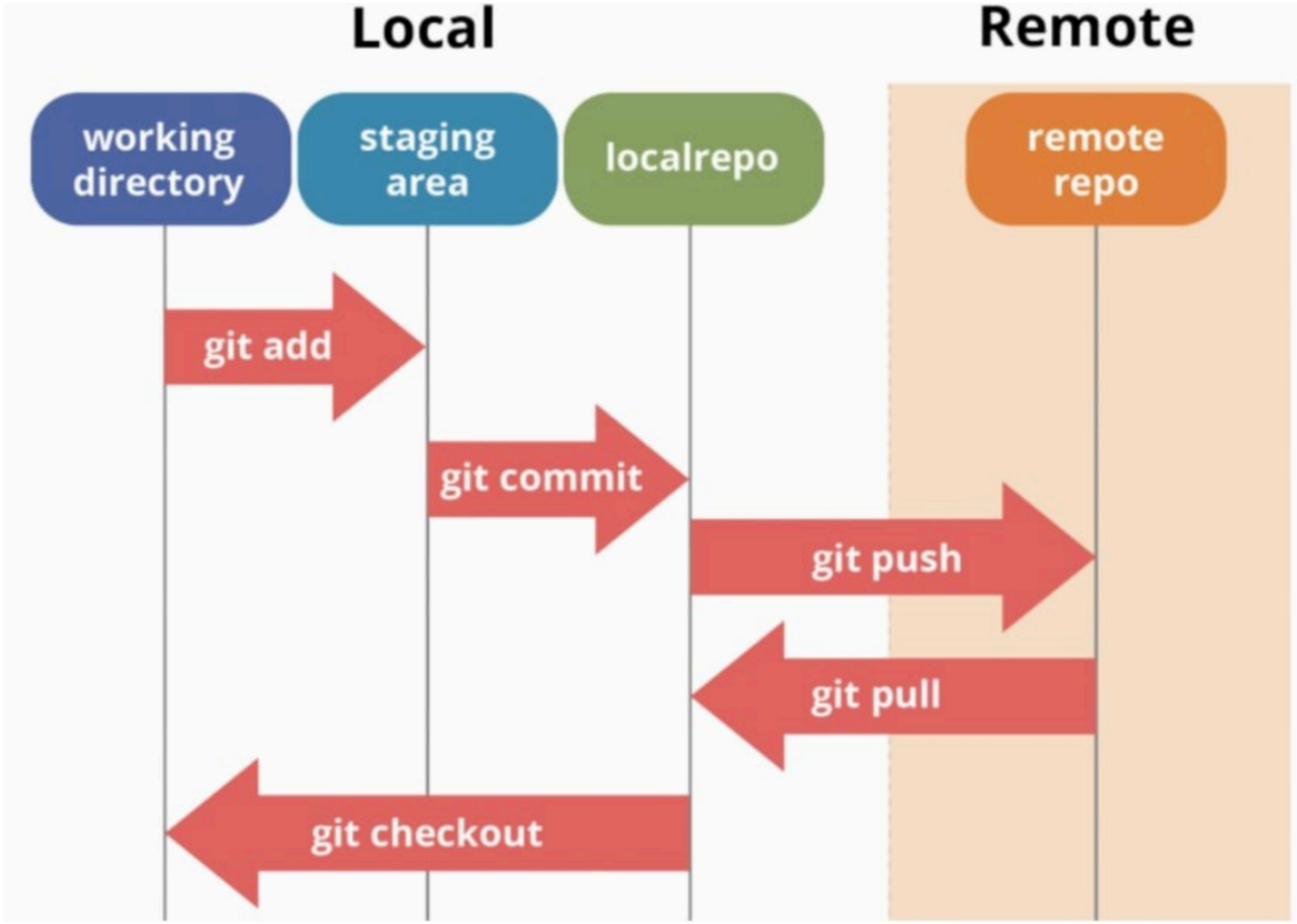
iOS에서 UI를 그리는 방법

iOS에서 UI를 그리는 방법

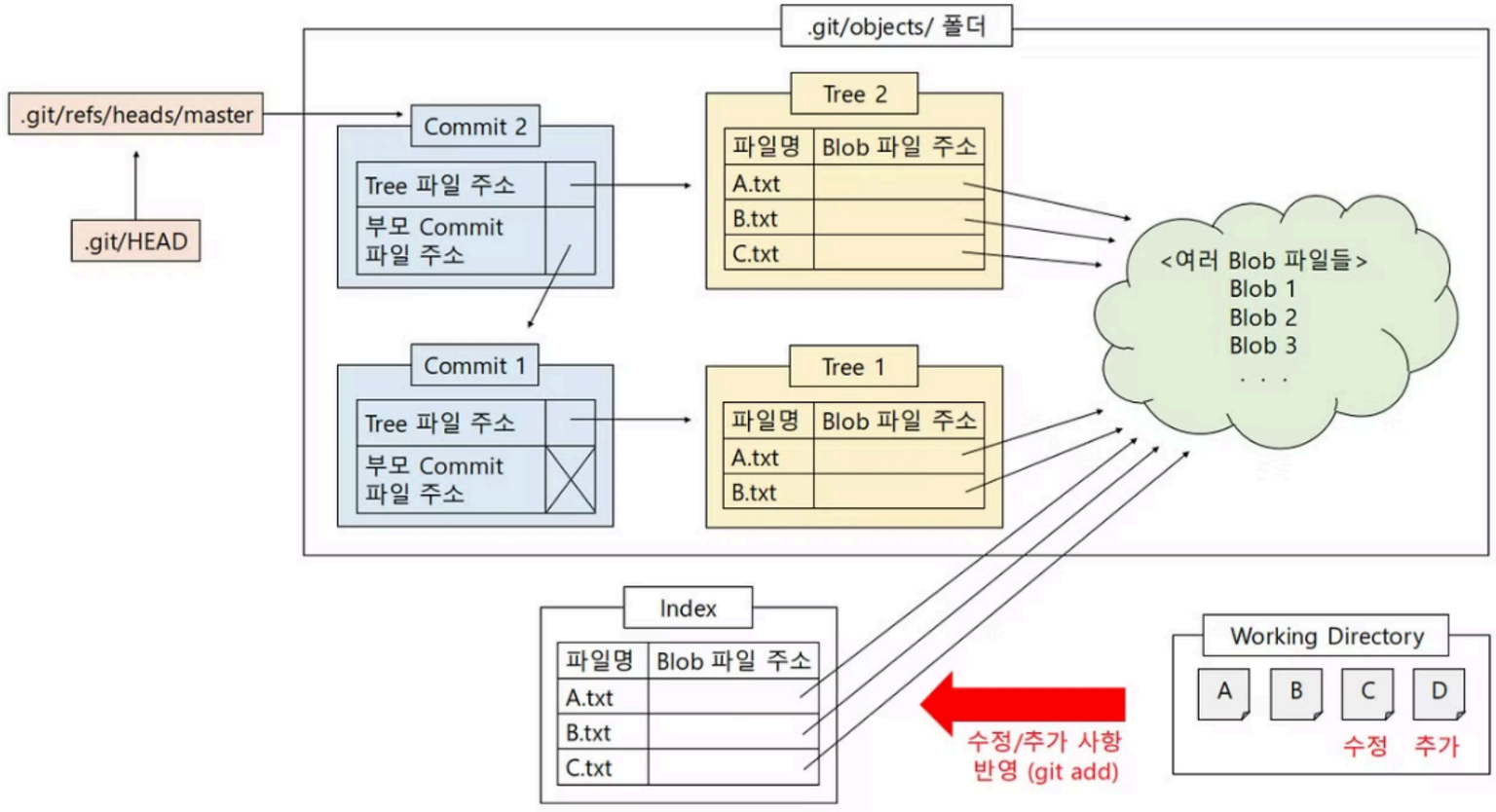
https://www.youtube.com/watch?v=jpxmOR6BJI8&t=952s

Git 내부 동작 톺아보기

Git add부터 commit 까지 일어나는 일



Git add부터 commit 까지 일어나는 일



Local: 실제 위치 프로젝트 폴더

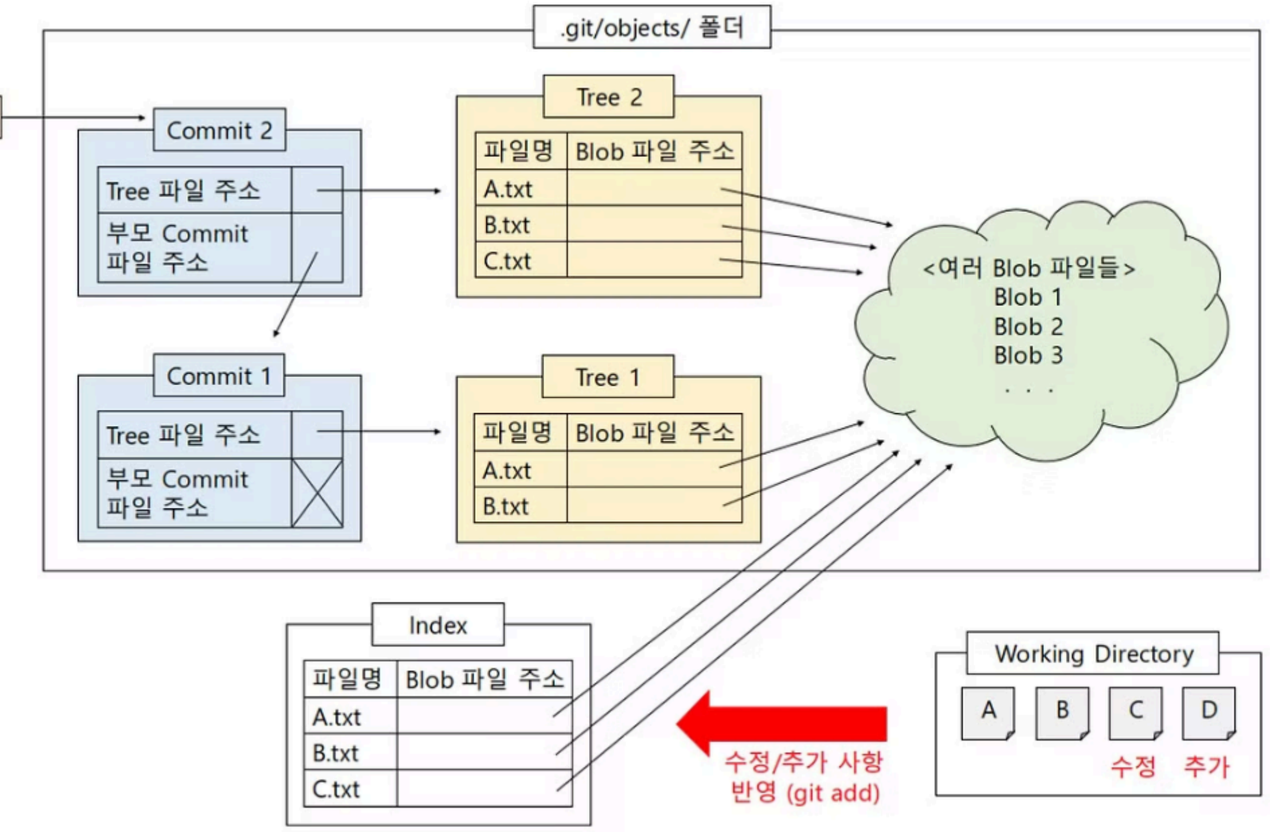
Index: StagingArea, Cache

.git/index 파일
커밋이 이뤄질 준비가 된 파일의 내용들이 위치
add시 변경

Repository: 저장소

.git/objects 폴더안에 저장
버전 관리를 하기 위해서
필요로 하는 데이터 저장하는 곳
commit시 변경

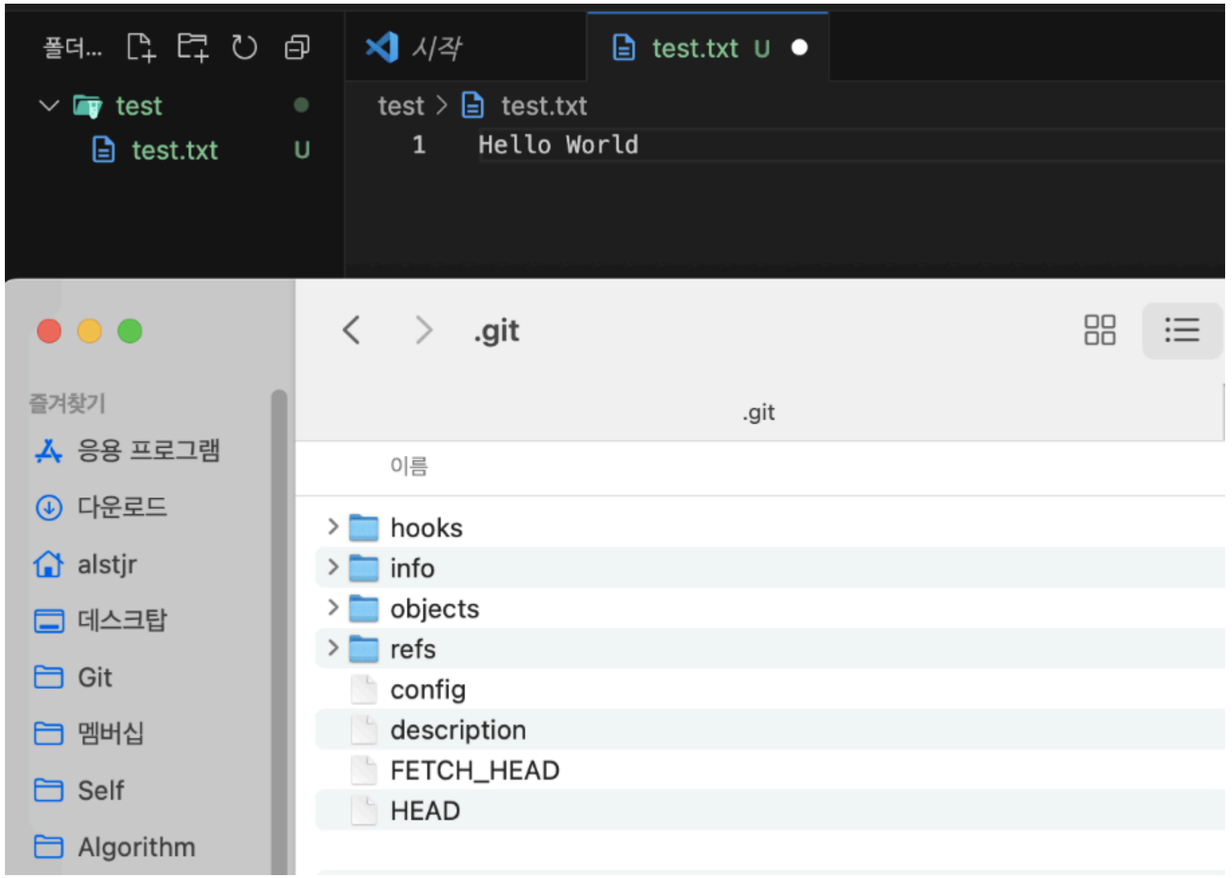
Git add부터 commit 까지 일어나는 일



종류	설명
Blob 파일	버전 관리하는 파일들 각각의 내용은 깃의 저장소에서 `Blob 파일의 형태로 저장` 된다. 파일의 **내용**에 SHA1이라는 **해싱 기법을 적용**하여 Blob 파일의 이름을 얻어내기 때문에, `내용이 같은 파일들은 모두 하나의 Blob 파일로서 저장` 된다. 이러한 원리로 깃은 여러 버전에 걸쳐 존재하는 파일들의 내용을 중복 없이 관리할 수 있게 된다.
Commit 파일	하나의 버전을 생성한다는 것은 하나의 Commit 파일을 만드는 것을 의미한다. Commit 파일은 하나의 Tree 파일을 가리키게 되어 있다. 이 파일에는 `가리키고 있는 Tree 파일의 주소(이름)와 직전 버전에 해당하는 Commit 파일의 주소(이름)가 기록` 된다.
Tree 파일	`커밋 시점의 파일들 각각에 대해 그 파일명과 해당 파일의 내용을 담고 있는 Blob 파일의 주소(이름)가 기록` 된다. 위에서 설명했던 인덱스 파일(.git/index)과 성격이 유사하다.
Tag파일	Commit object 파일을 가리키며 `태그명` 과 `작성자, 주석` 을 담는다.

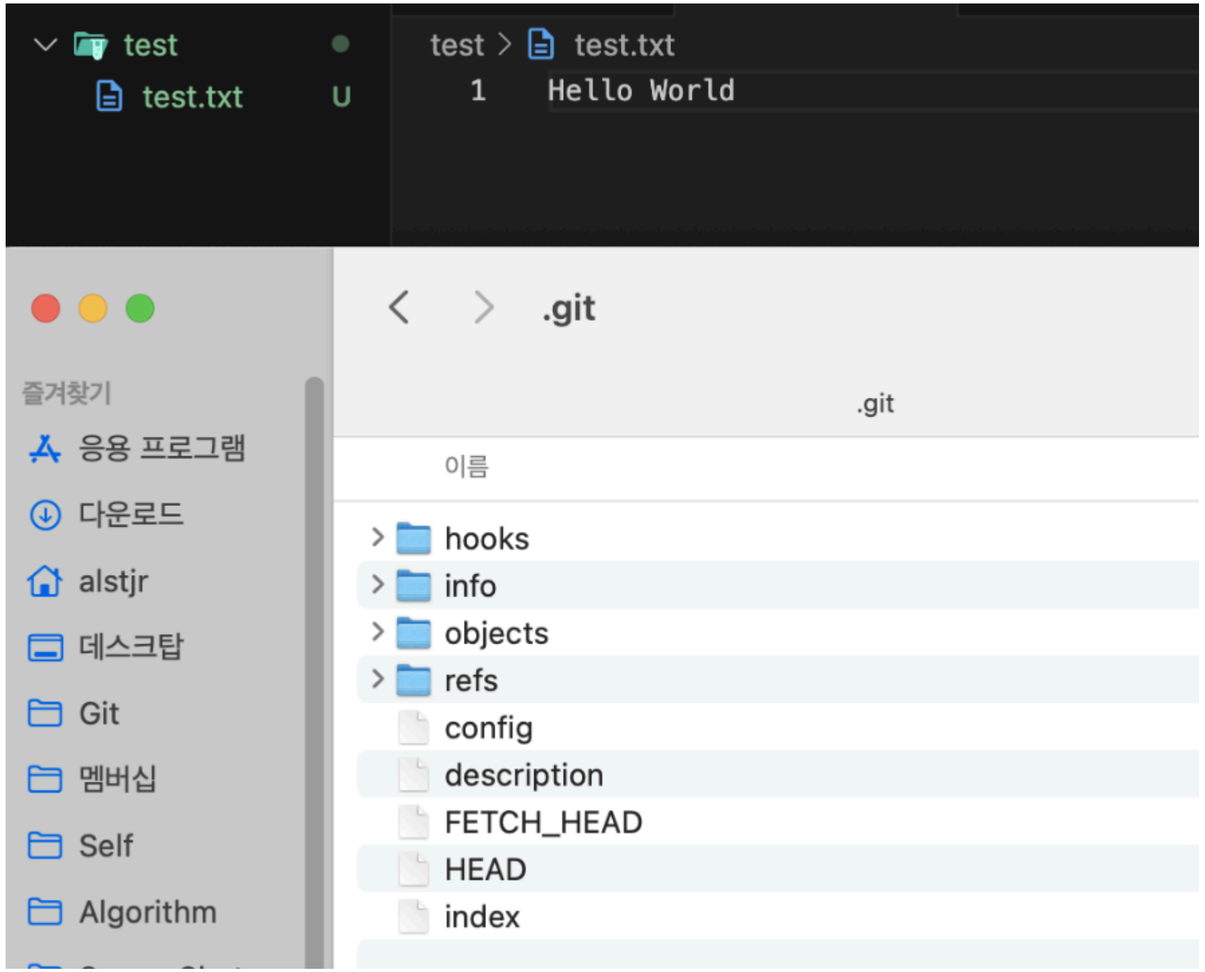
Git add부터 commit 까지 일어나는 일

git init



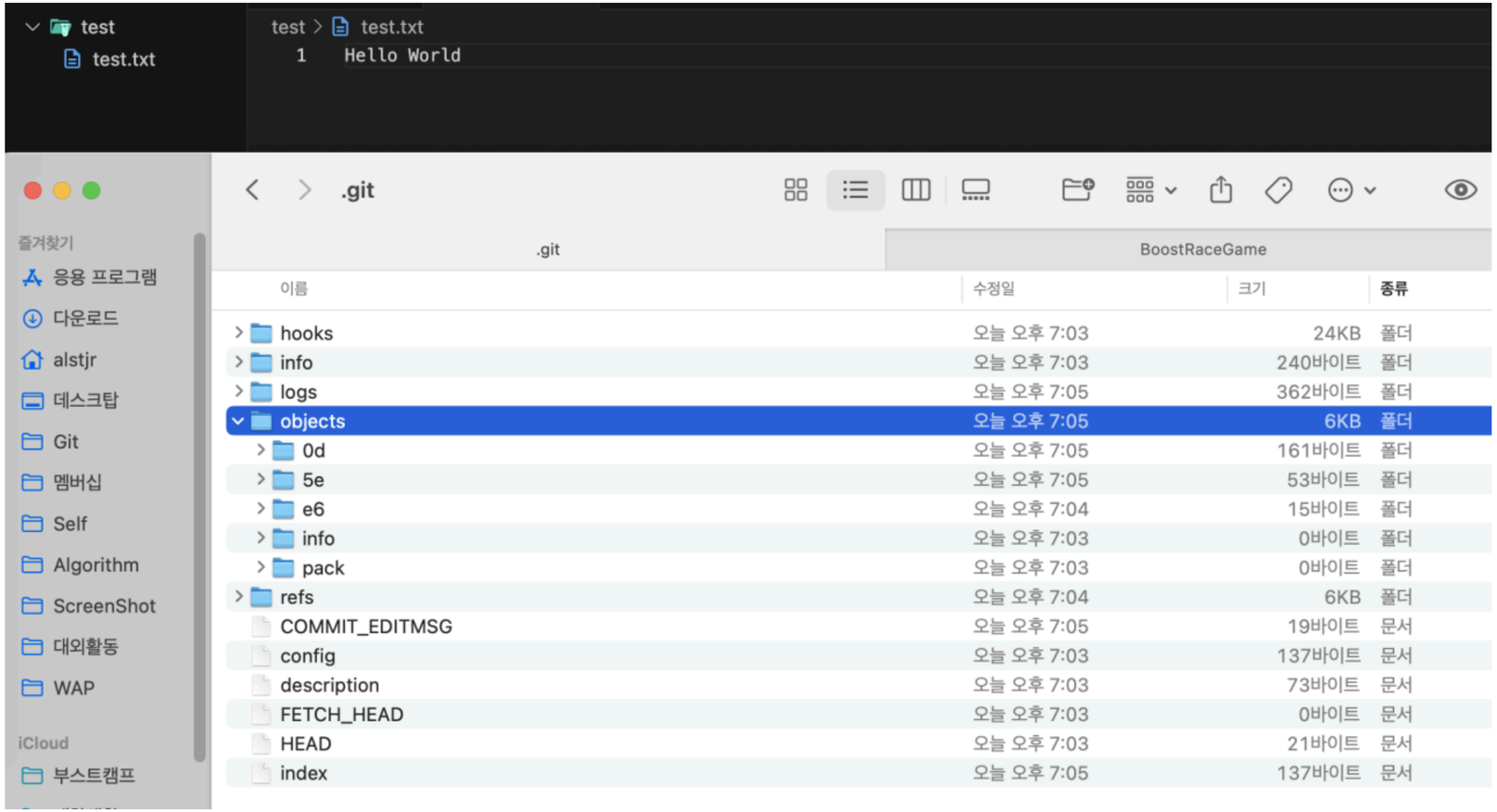
Git add부터 commit 까지 일어나는 일

git init -> git add



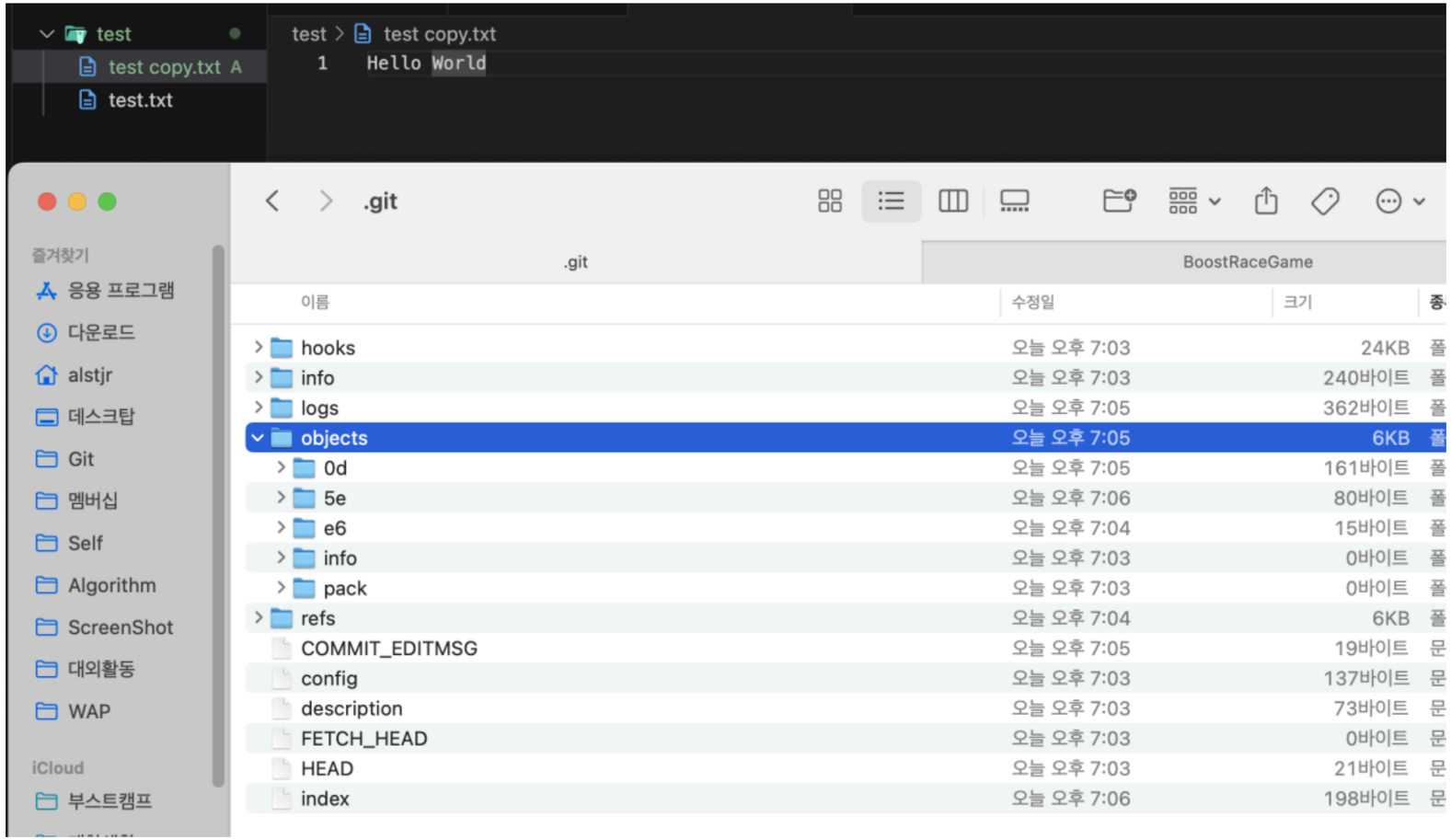
Git add부터 commit 까지 일어나는 일

git init -> git add -> commit



Git add부터 commit 까지 일어나는 일

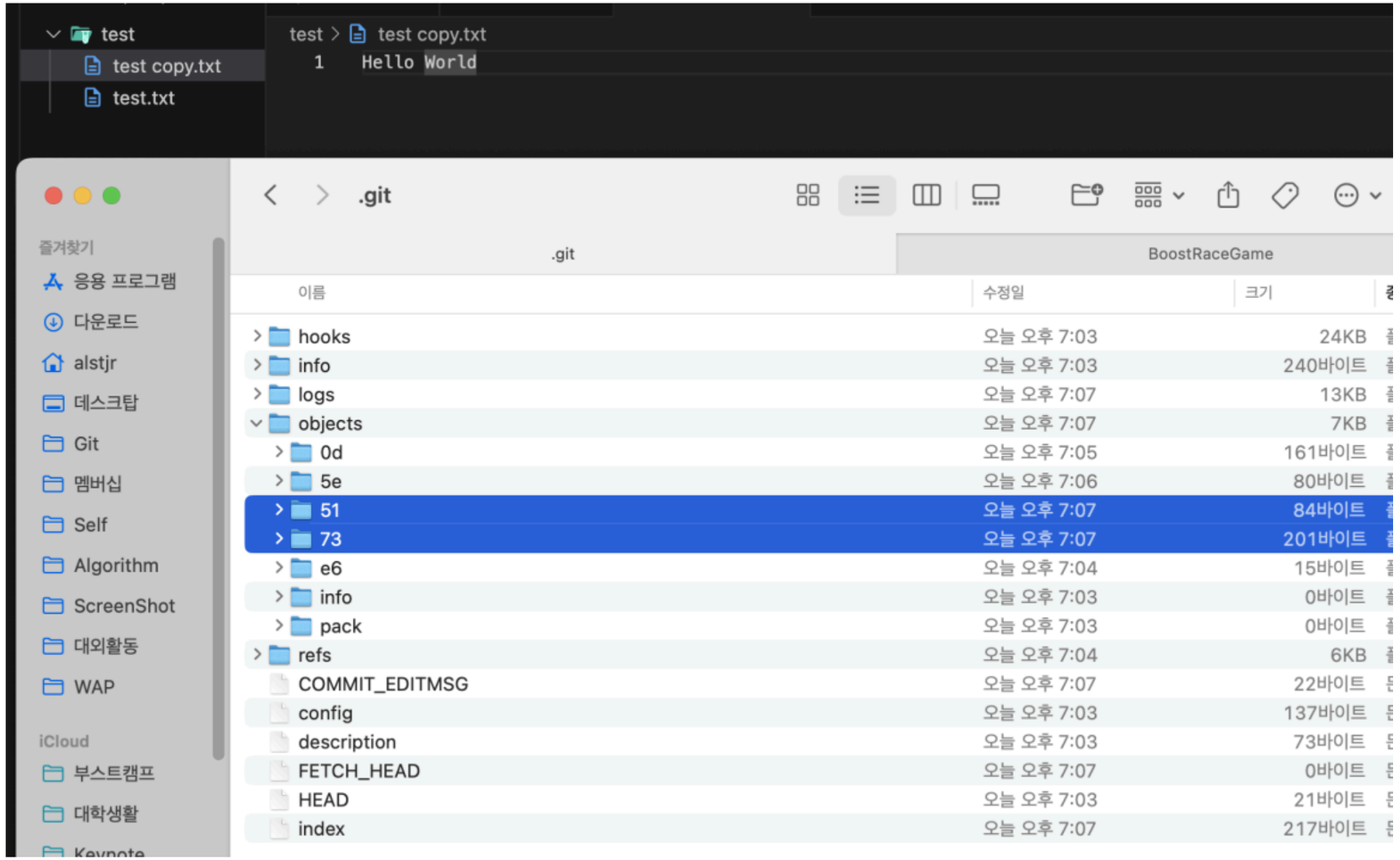
파일 복사 -> git add



Git add부터 commit 까지 일어나는 일

파일 복사 -> git add -> commit

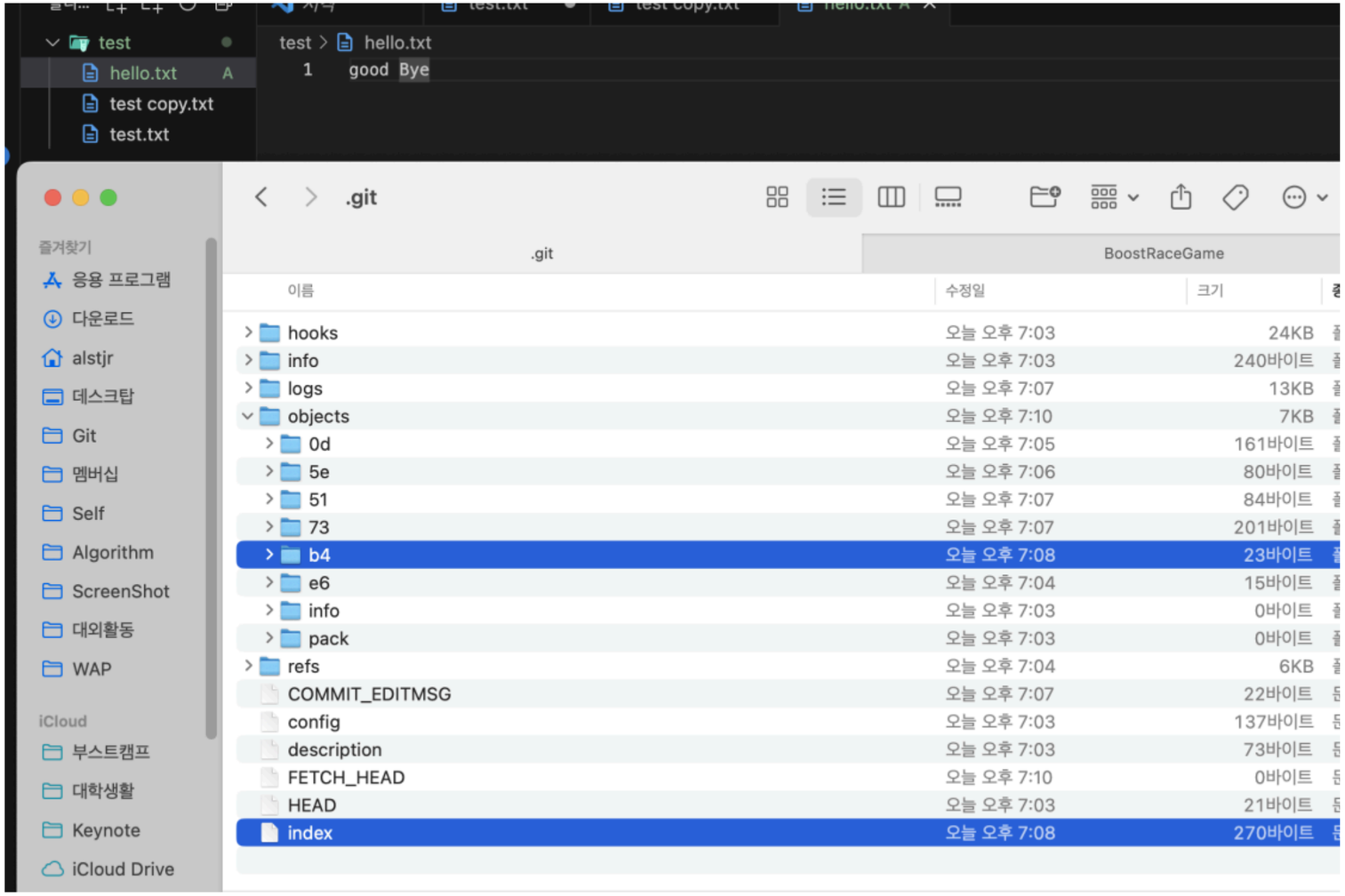
tree와 commit 파일 생성



Git add부터 commit 까지 일어나는 일

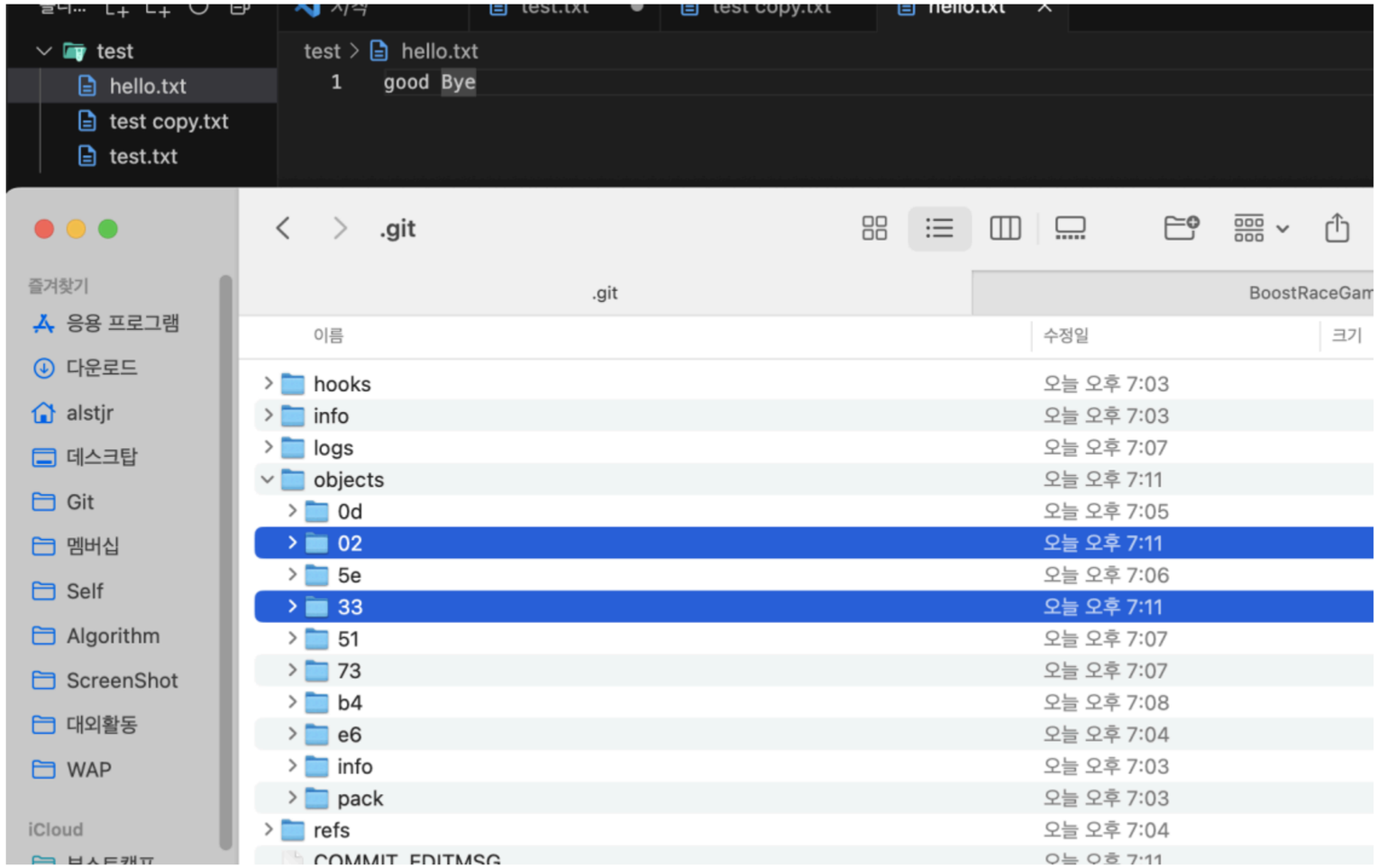
새로운 파일 생성 후 add

blob 파일 생성



Git add부터 commit 까지 일어나는 일

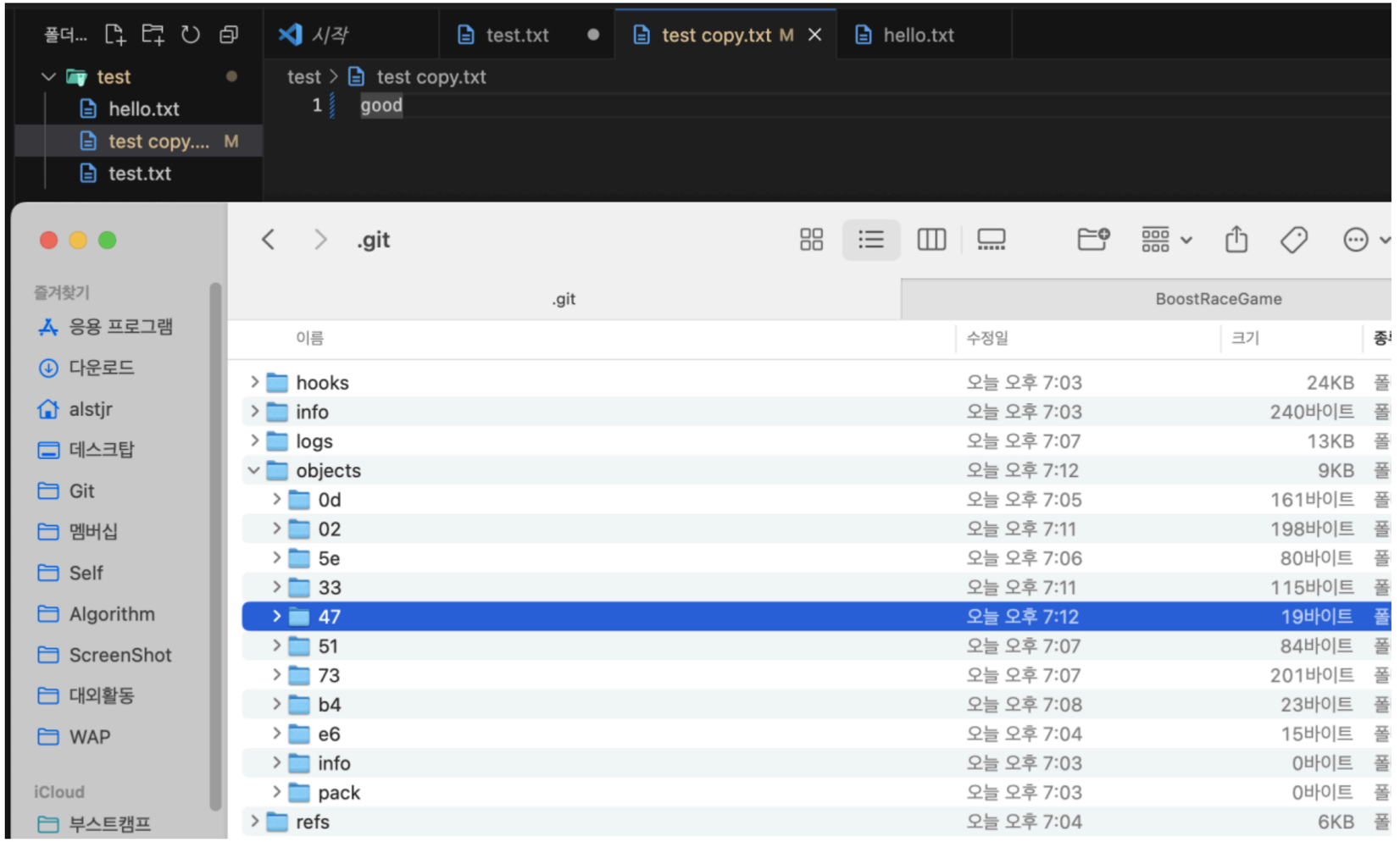
새로운 파일 생성 후 add -> commit tree와 commit 파일 생성



Git add부터 commit 까지 일어나는 일

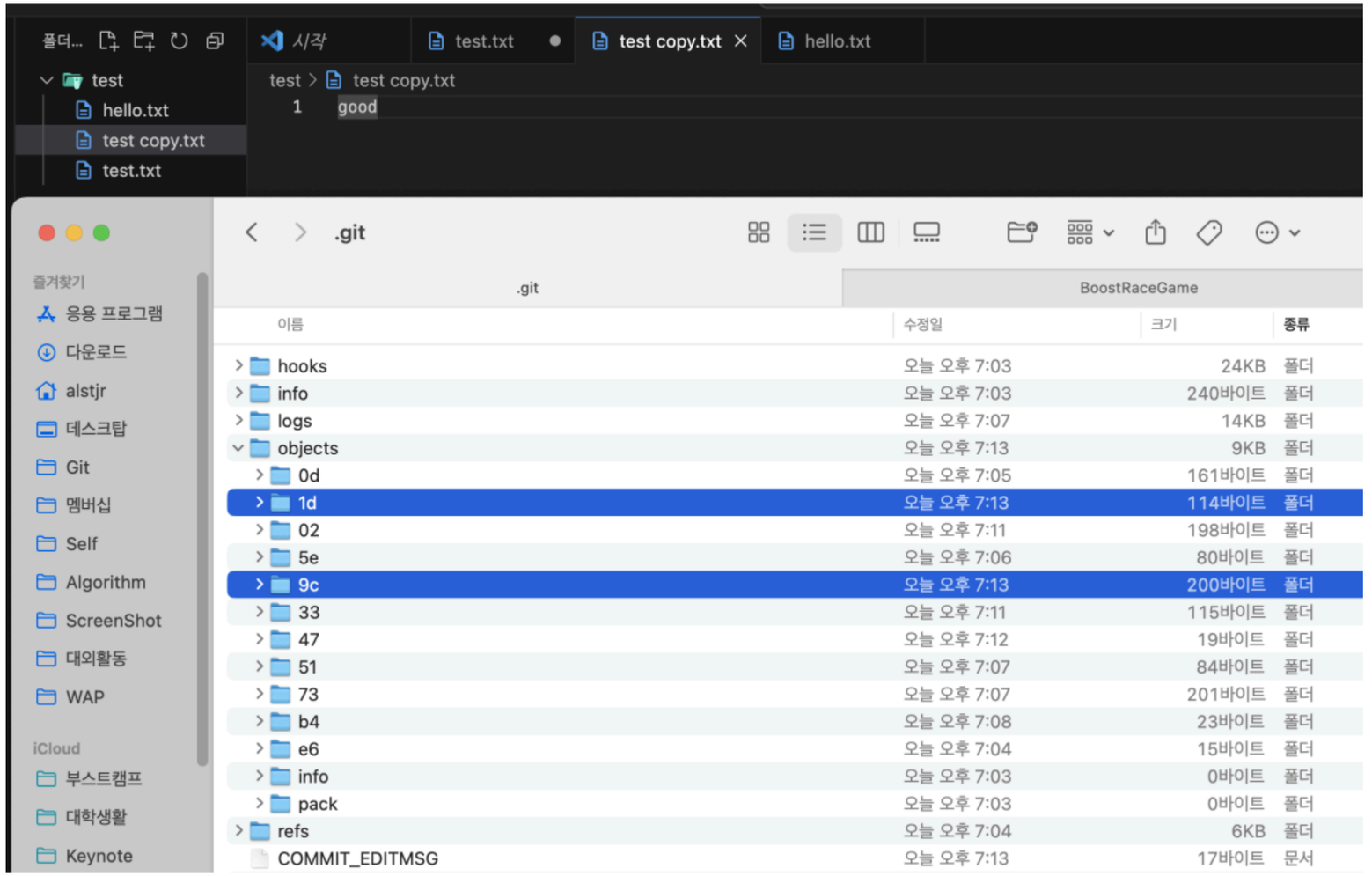
복사본 파일 수정 후 add

blob 파일 생성



Git add부터 commit 까지 일어나는 일

복사본 파일 수정 후 add -> commit tree와 commit 파일 생성



감사합니다!