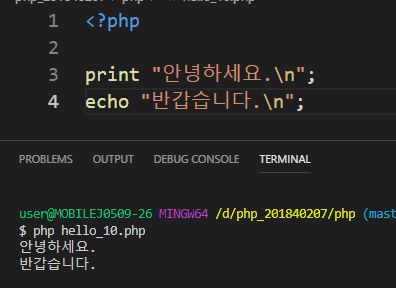
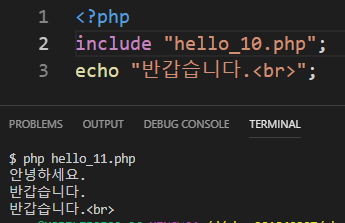
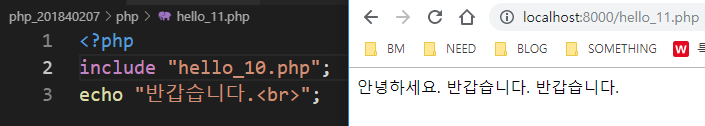


둘 다 똑같음

 \n 하면 줄바꿈

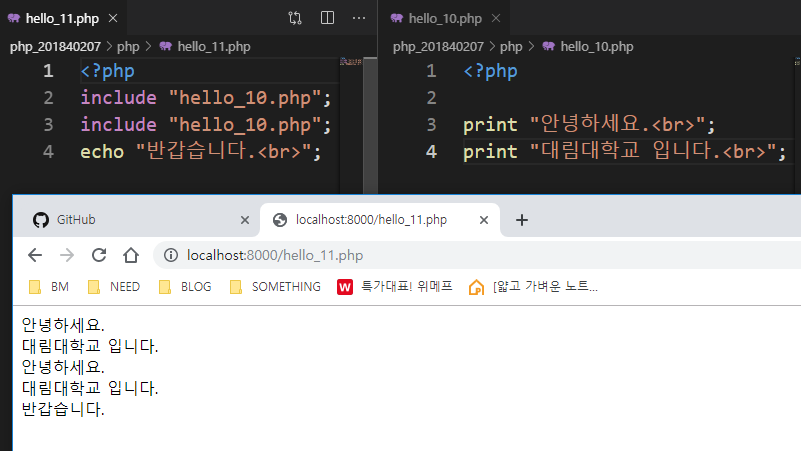
 

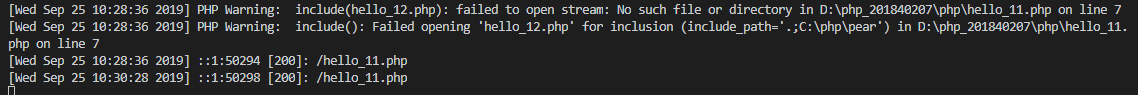
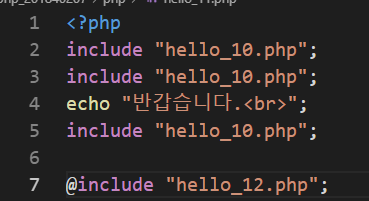
<br>은 브라우저에서 줄바꿈

include하면 합쳐져서 같이 나옴



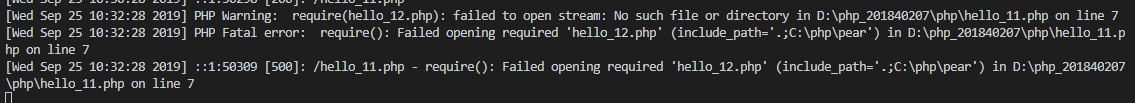
 둘이 똑같음 (시험에 나옴)

include는 그 파일에 있는 모든 소스를 가져다쓰는 것

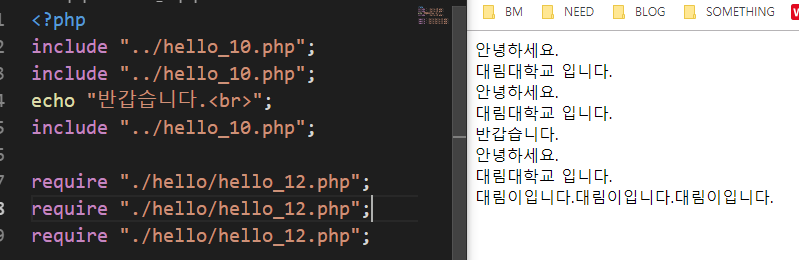


오류제어연산자 ‘@’를 쓰면 hello\_12.php파일이 없어도 콘솔창에 오류가 뜨지 않는다.



Include는 파일이 없어도 실행되지만 require은 파일이 없으면 실행이 중단됨

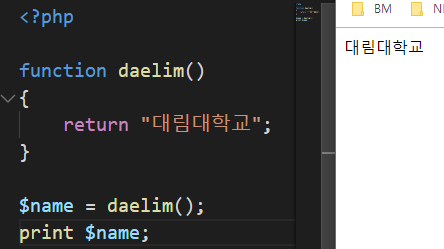
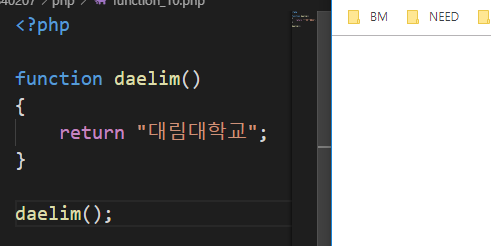
둘다 중복해서 사용가능



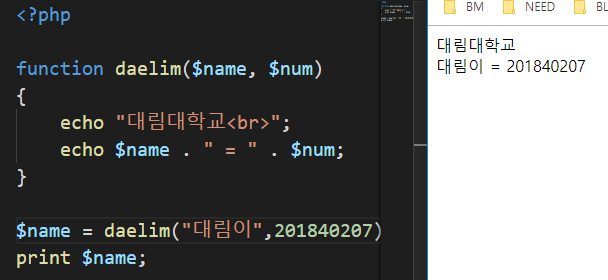
(php\_201840207/hello\_10.php php\_201840207/php/hello\_11.php php\_201840207/php/hello/hello\_12.php)

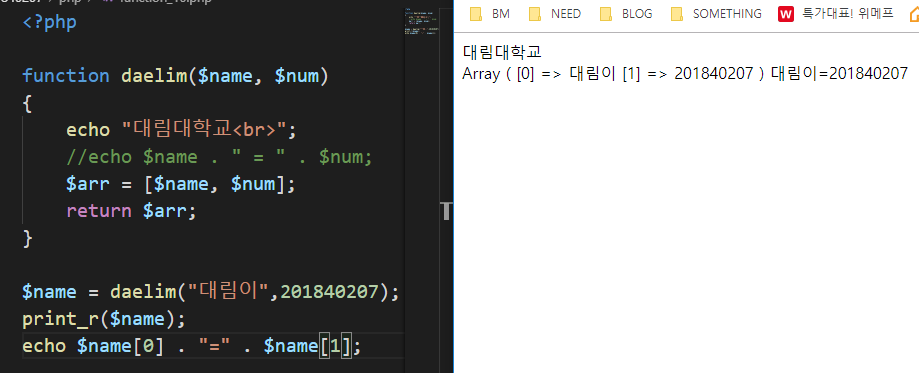
(require)동일한 폴더/동일한 경로 안에 파일이 없으면 실행중단

저런식으로 경로를 지정해주면 실행됨 (시험 중요)

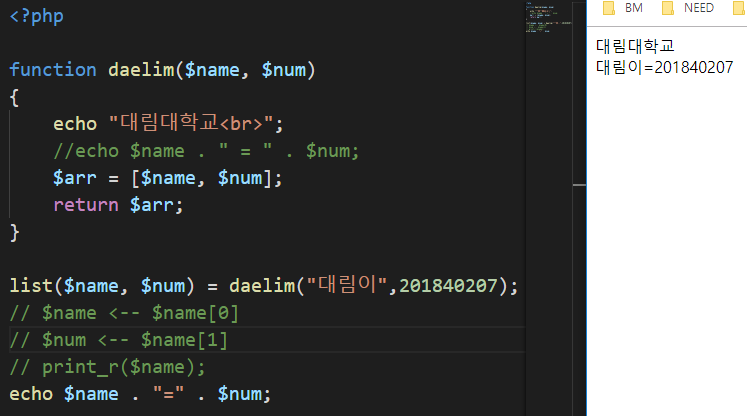


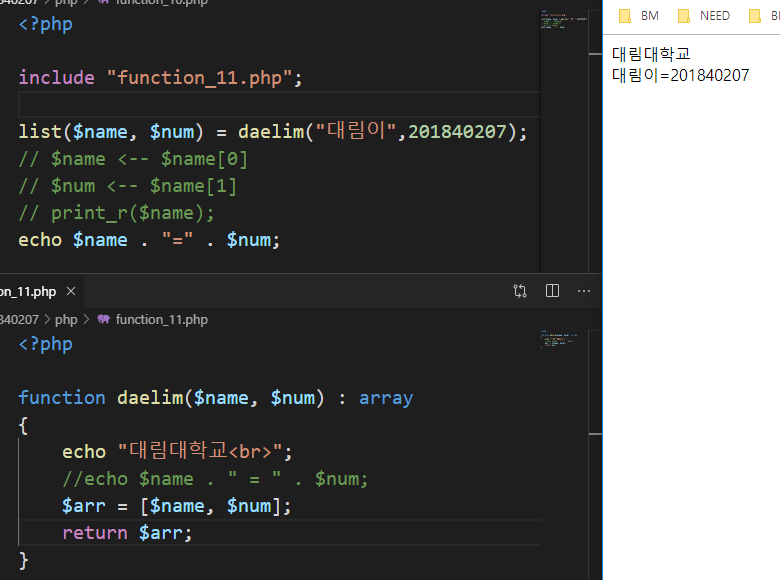
Return 값은 하나만, 반환값을 받아줄 변수를 쓰고 출력

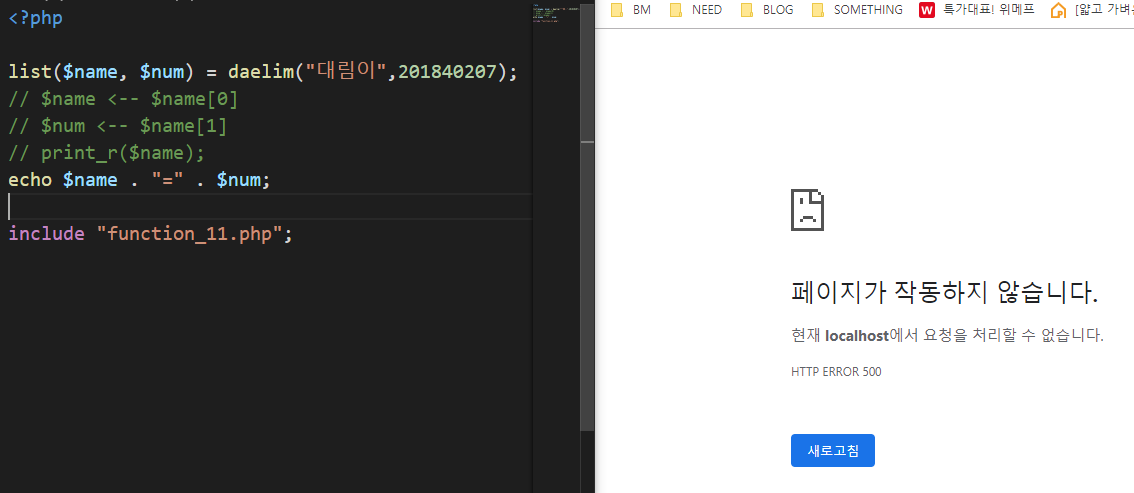


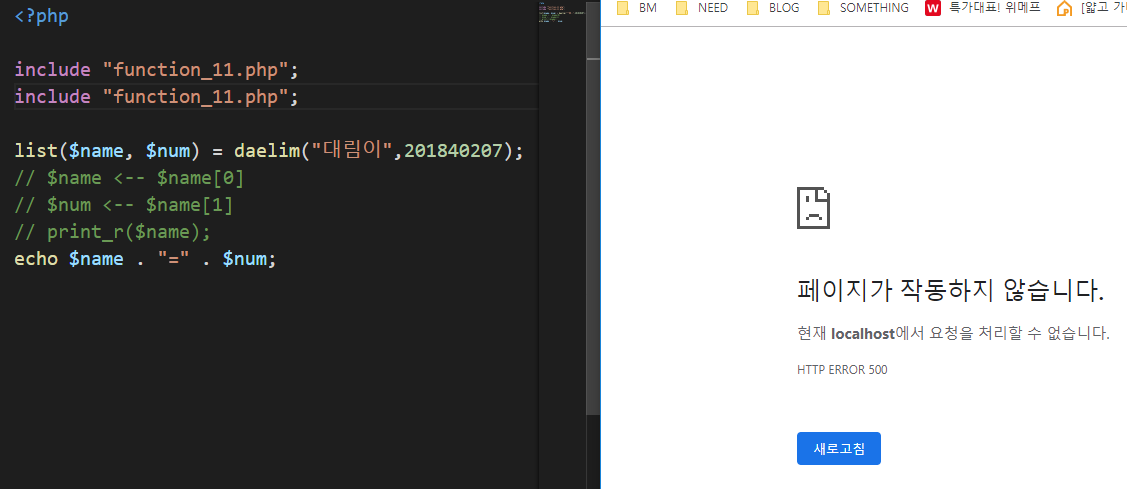


배열을 이용하여 두개의 값을 하나의 값처럼 만들어서 반환

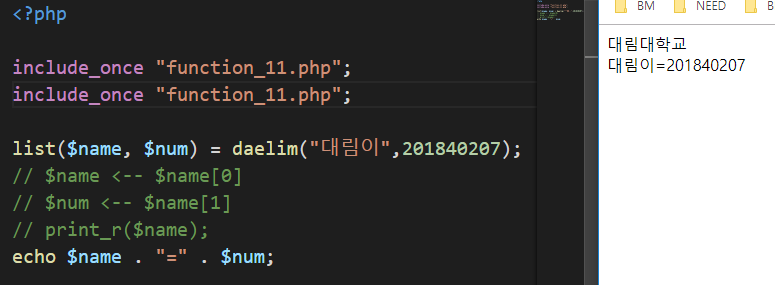
 list 사용

 include를 사용하여 실행

아래에 넣으면 실행안됨

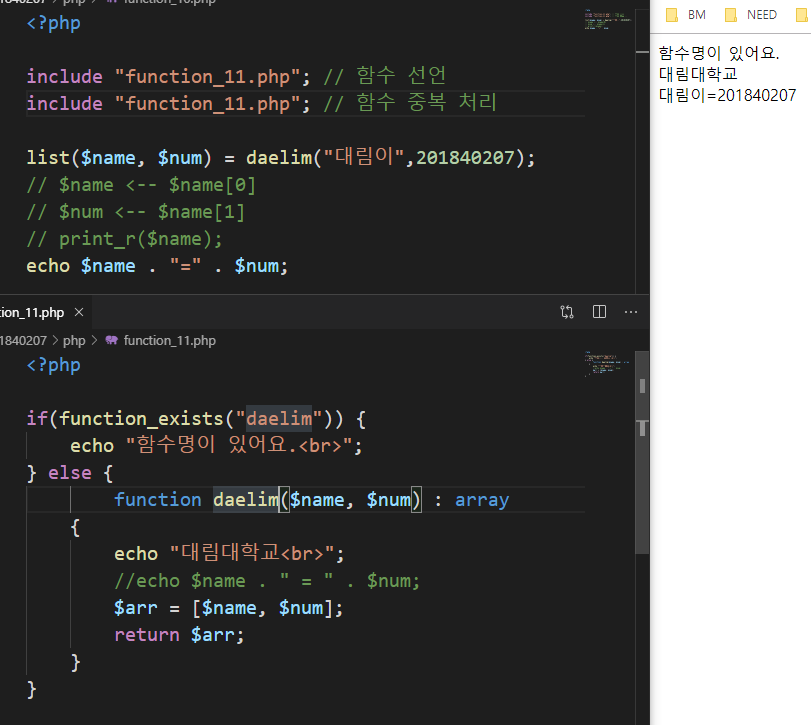


두번 include하면 함수명이 중복이 되어서 실행오류

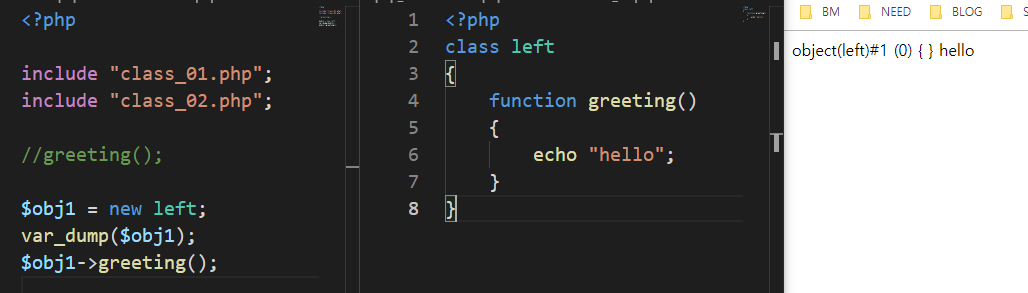


Include\_once하면 한번 include된 파일은 다시 include되지않아서 실행됨

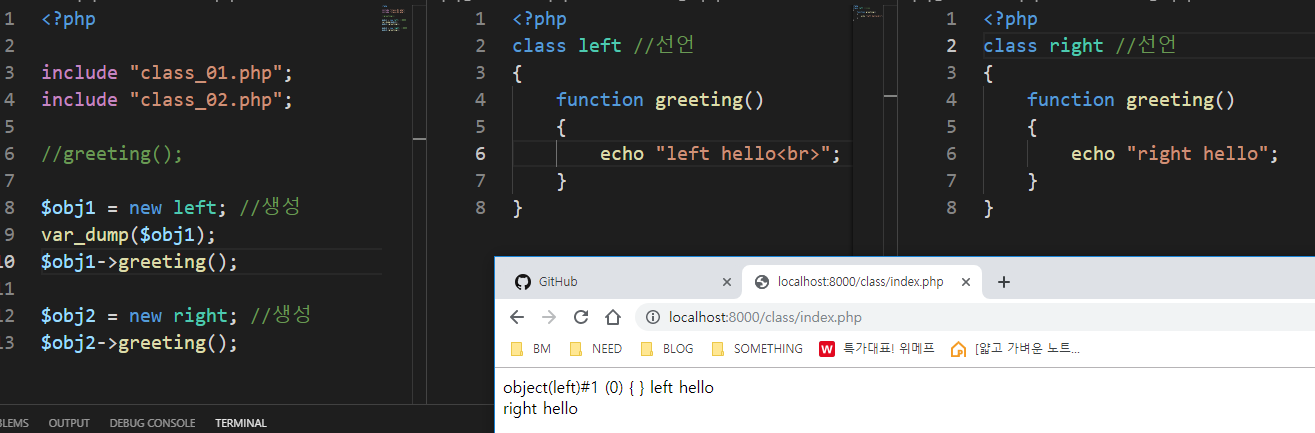
Require\_once도 있음

 함수가 존재하면 함수명이 있어요 출력 존재하지 않으면 함수 선언

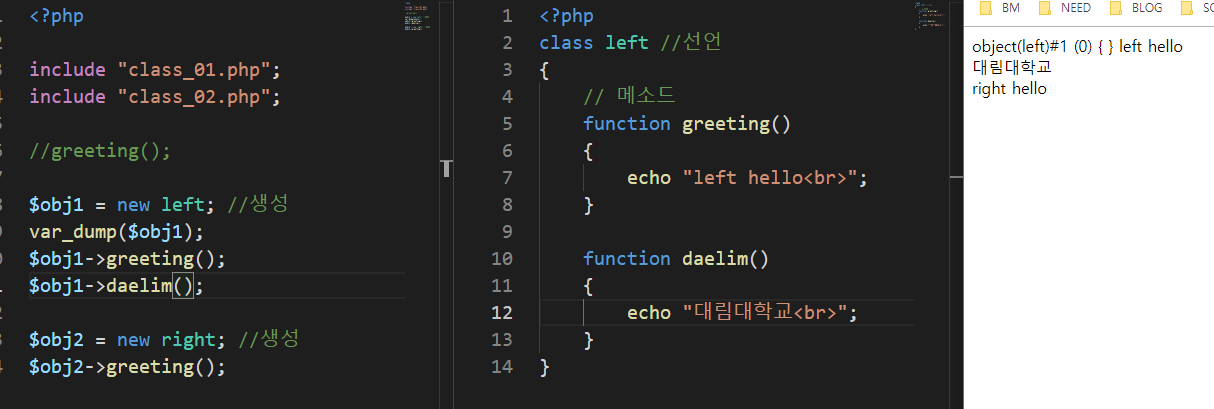
함수는 선언, 호출 / 클래스는 선언, 생성(인스턴스화 = 객체화), 호출



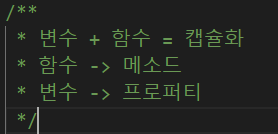
Class를 선언하고 생성한 후 함수 호출

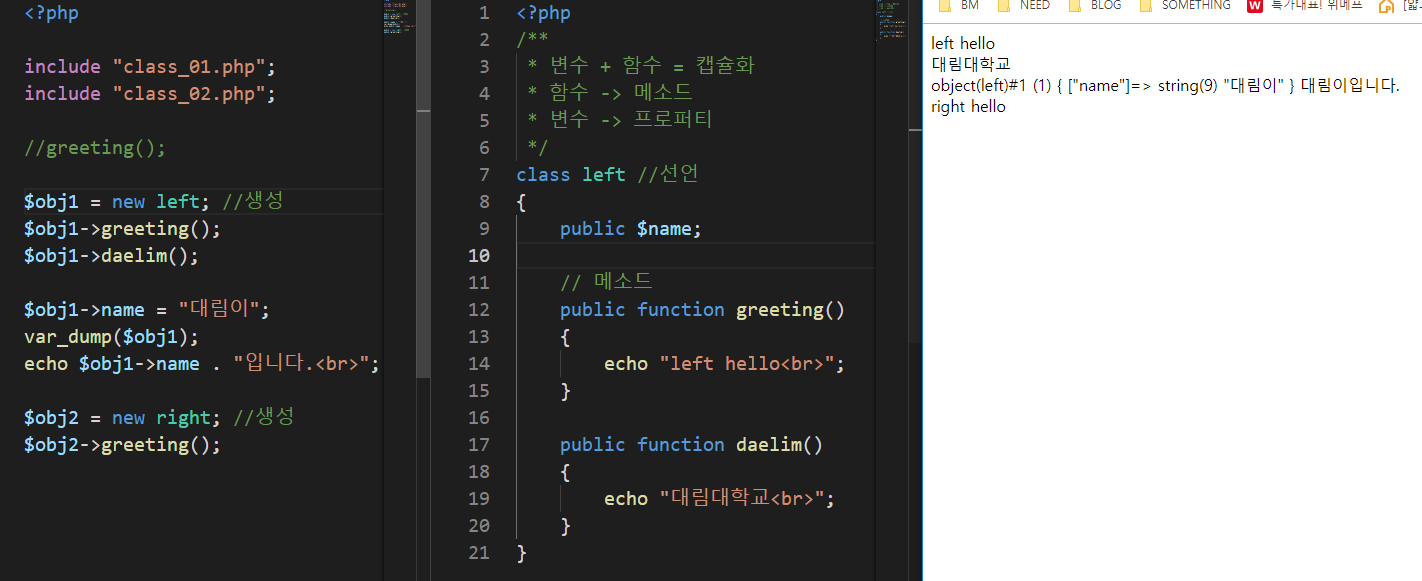


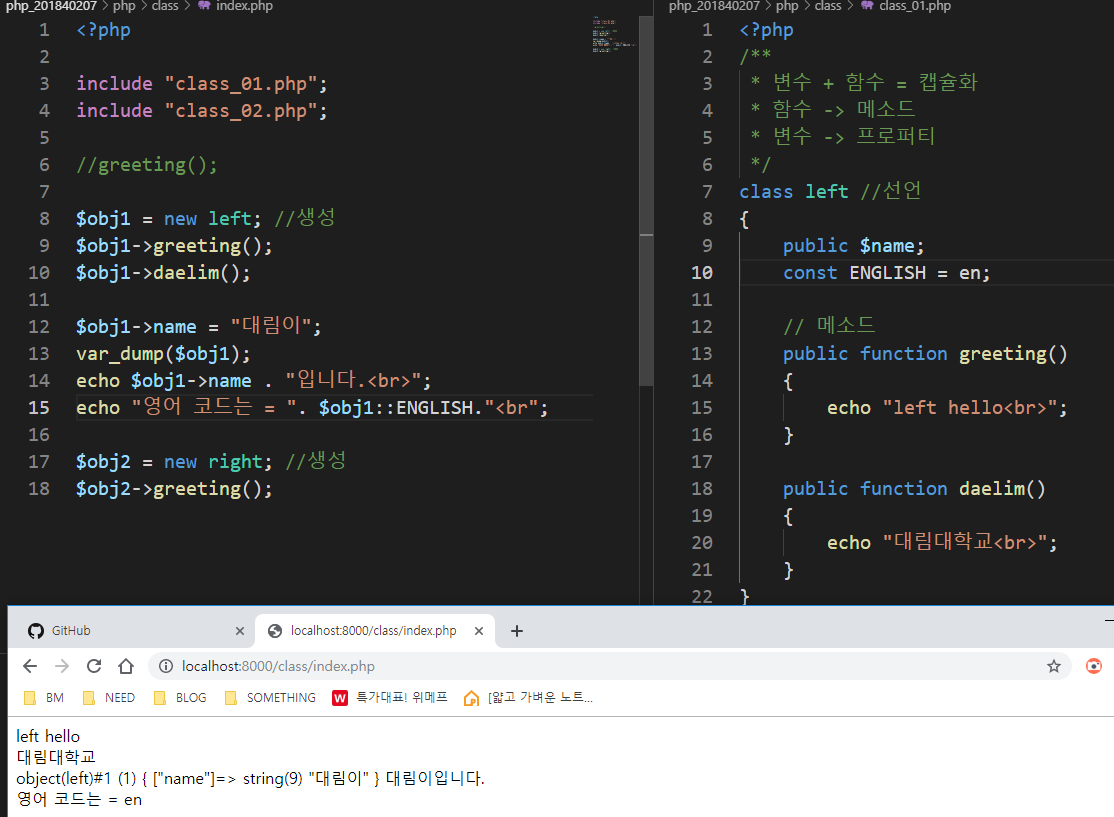
클래스명이 다른 두 개를 선언하여 같은 이름의 함수를 호출

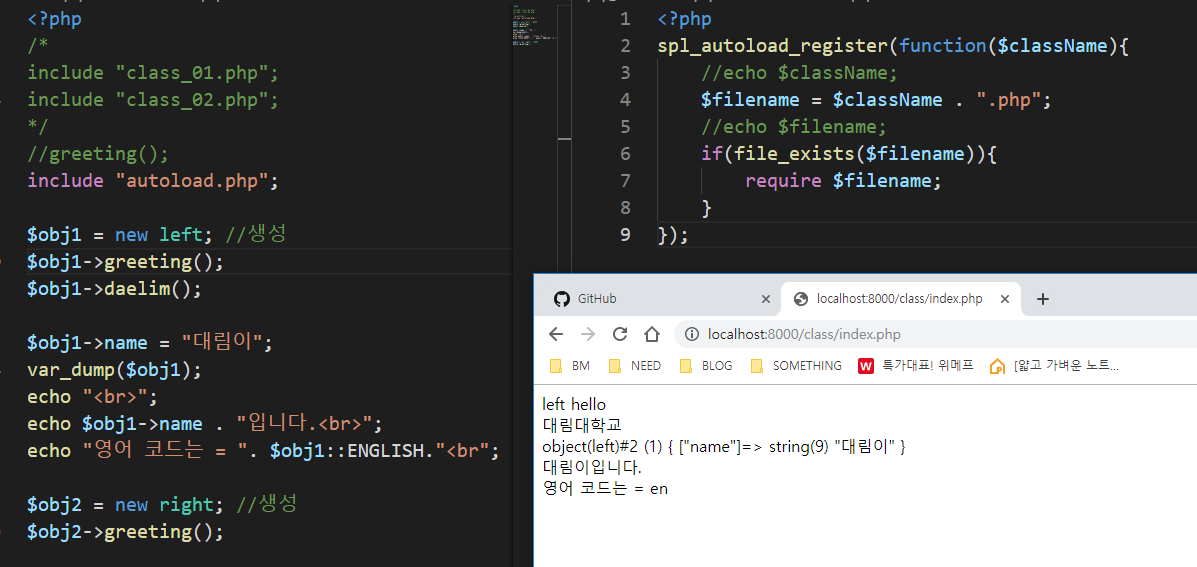


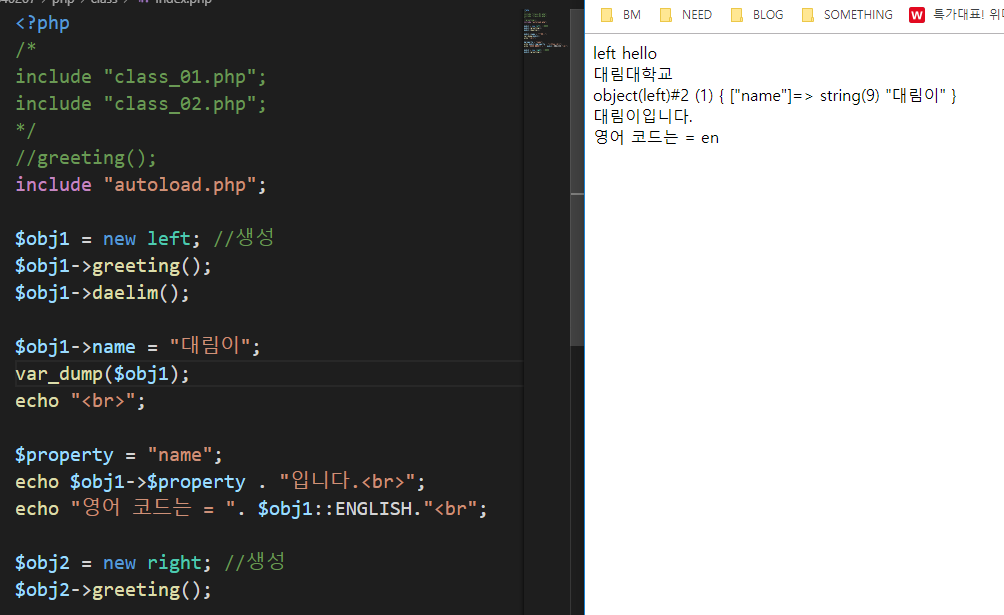
클래스 하나에 함수 여러 개 만들 수 있음 (캡슐화)

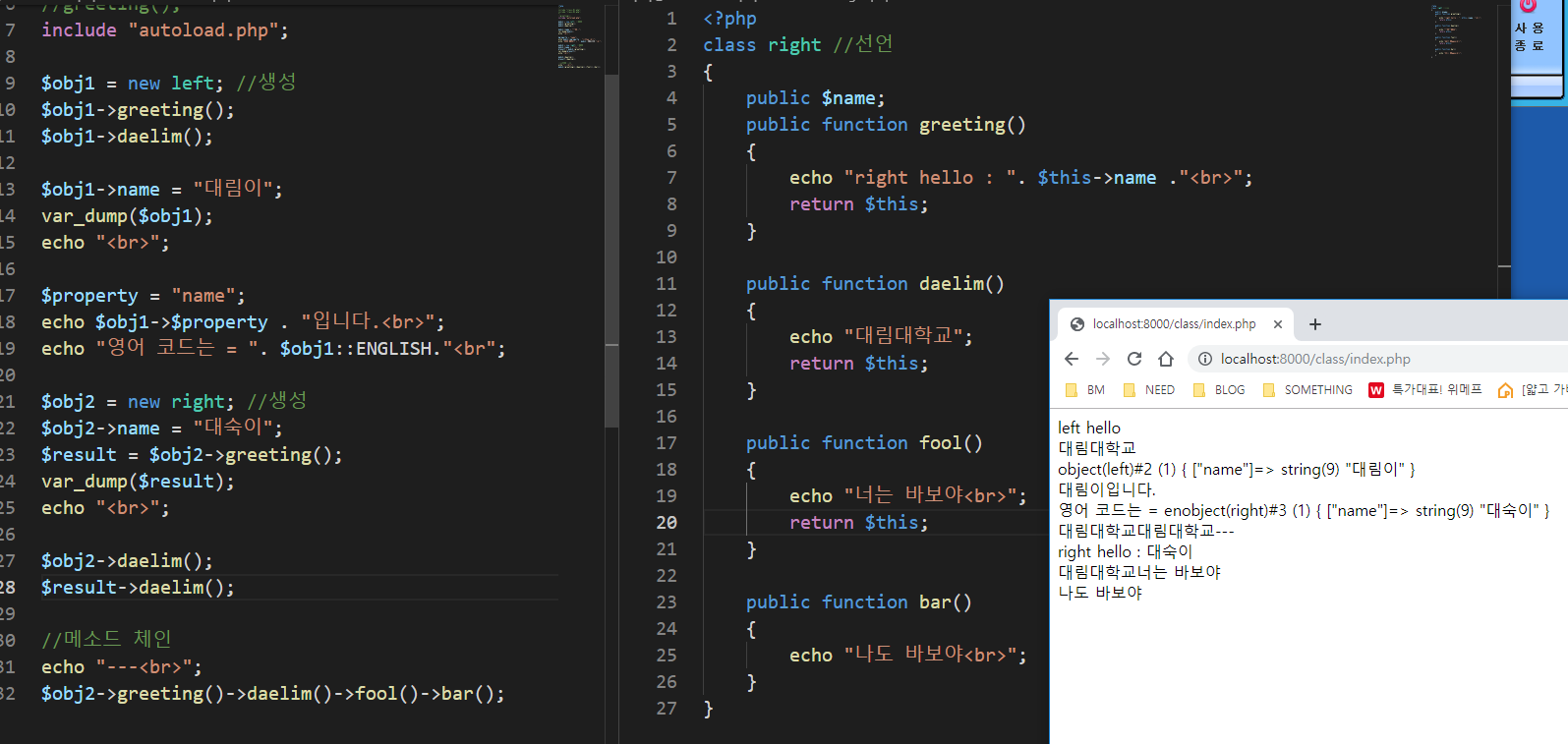


변수

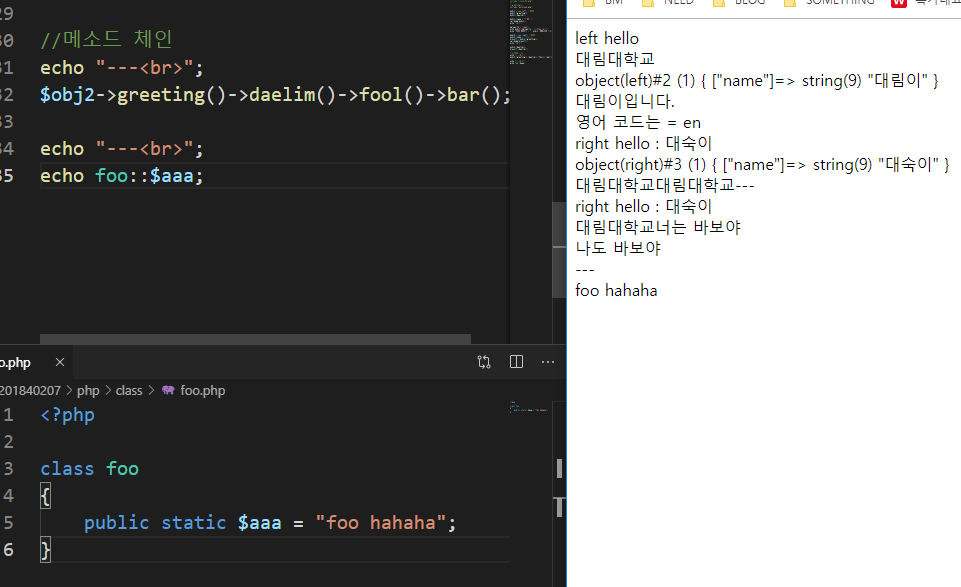
 상수

autoload



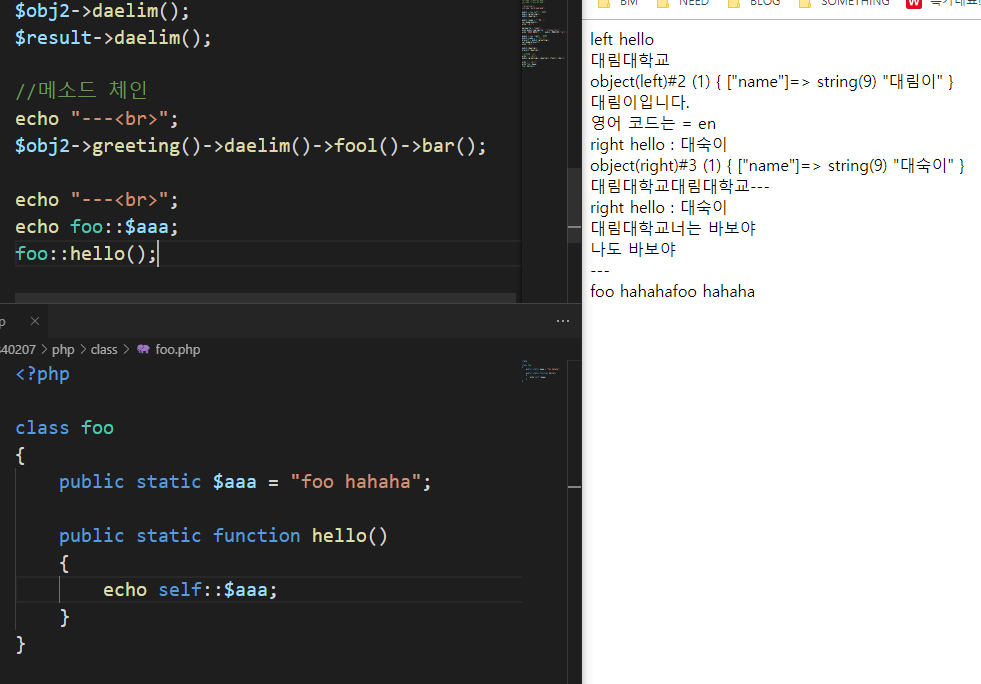


메소드 체인



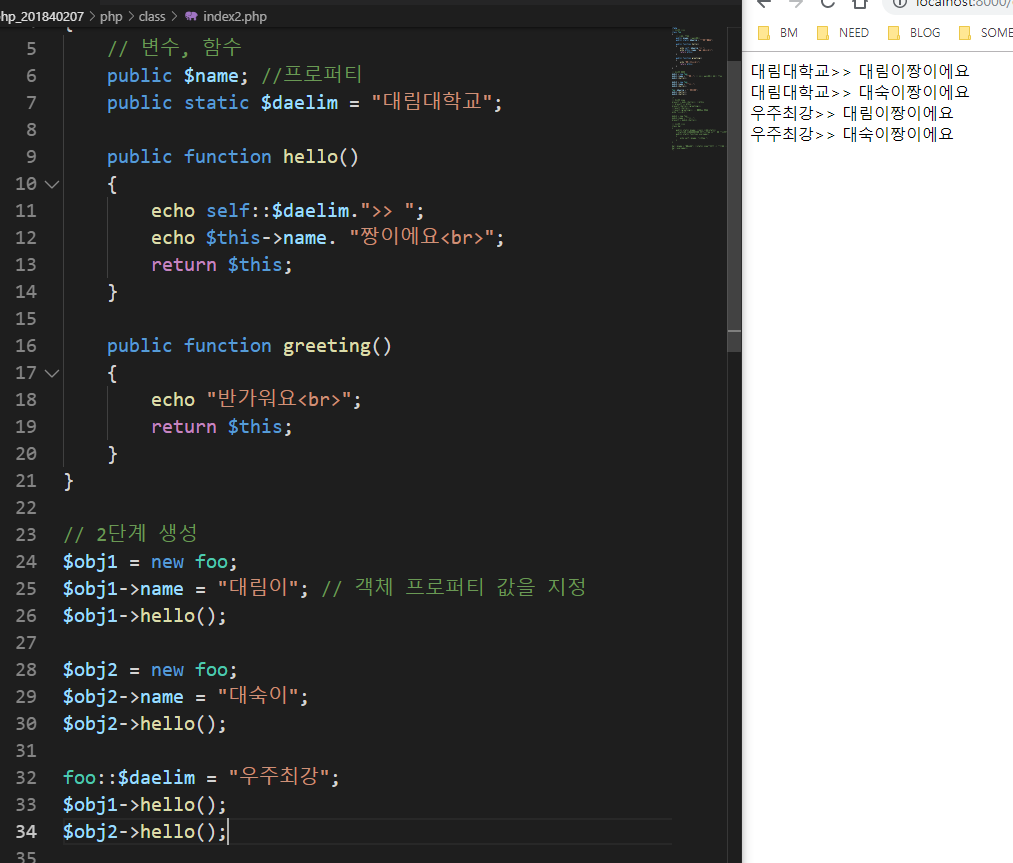
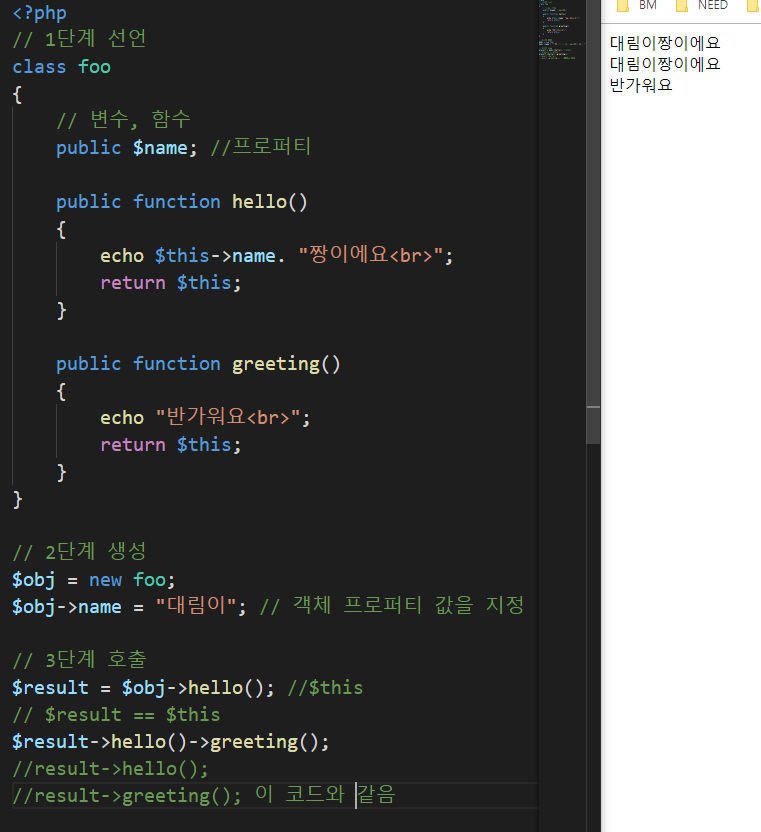
인스턴스를 만들지 않아도 접근 허용 (static)

정적으로 사용하면 $를 쓰고 인스턴스를 만들고 사용하려면 $제외하고 쓴다.

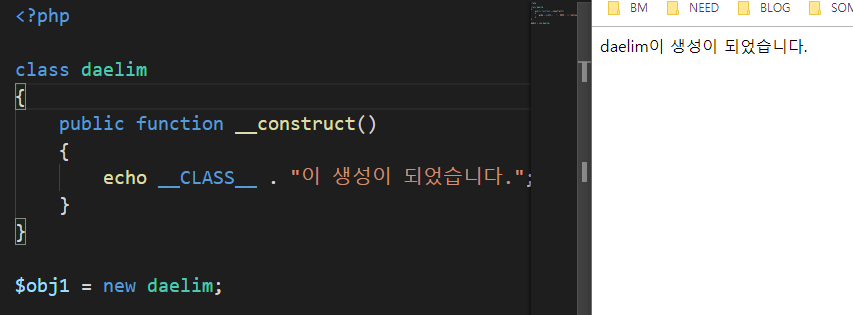


인스턴스화를 하지않으면 this를 사용할 수 없다. (static - self사용)

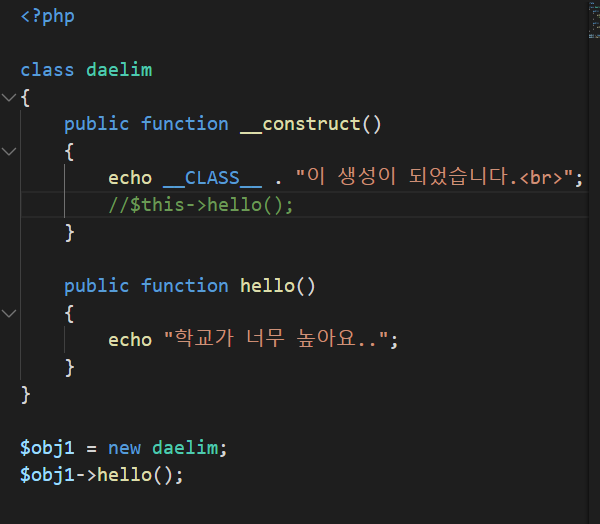
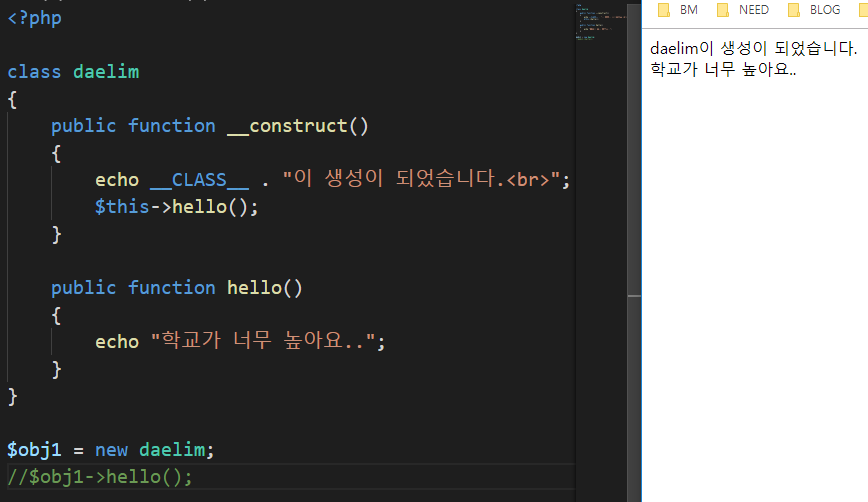
인스턴스화를 하면 this를 사용할 수 있다.



ㄴ정리



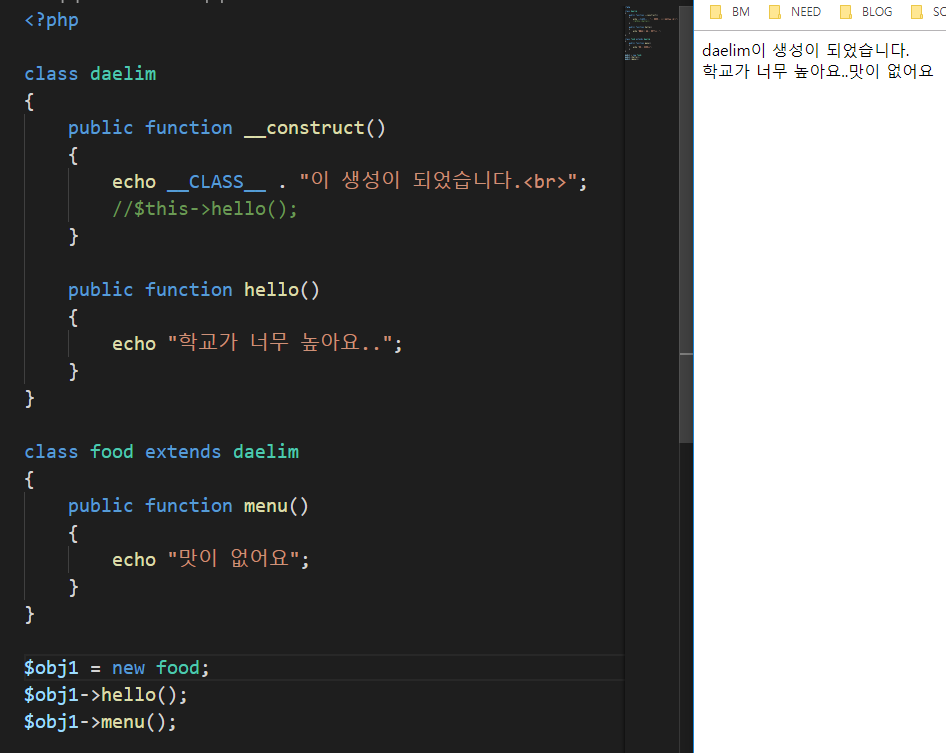
\_\_construct -> 생성하면 자동 실행



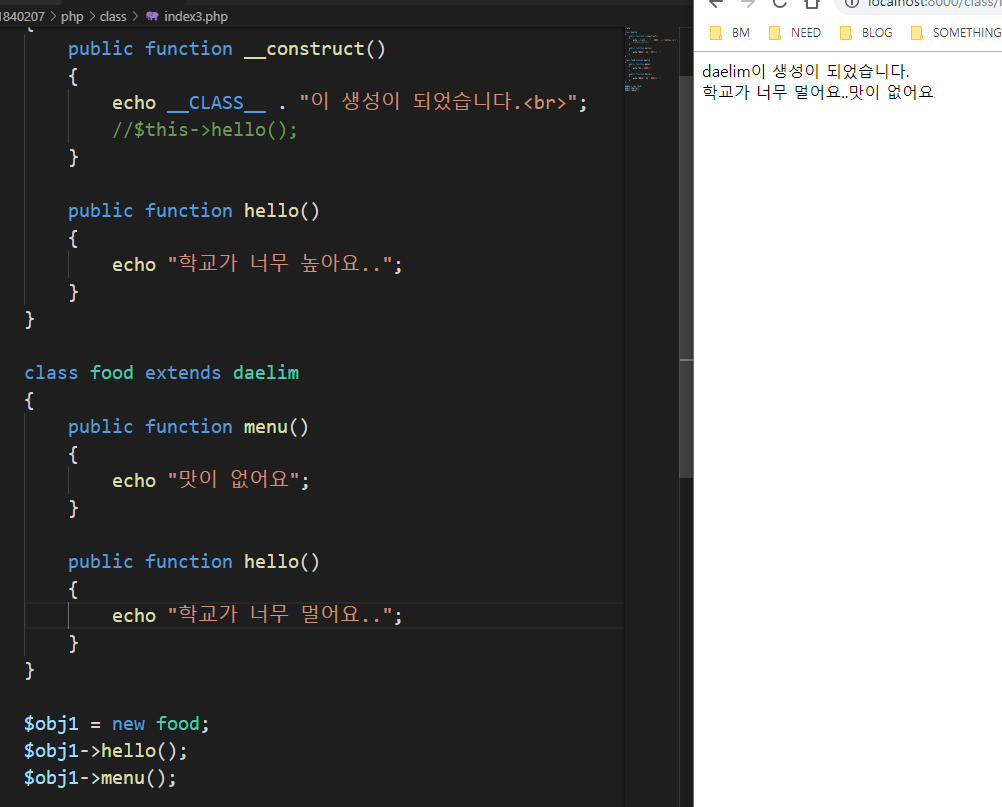
둘 다 결과 같음

Public은 어디에서나 가져다가 쓸 수 있음 private은 해당 클래스 안에서만 접근 가능(은닉화)

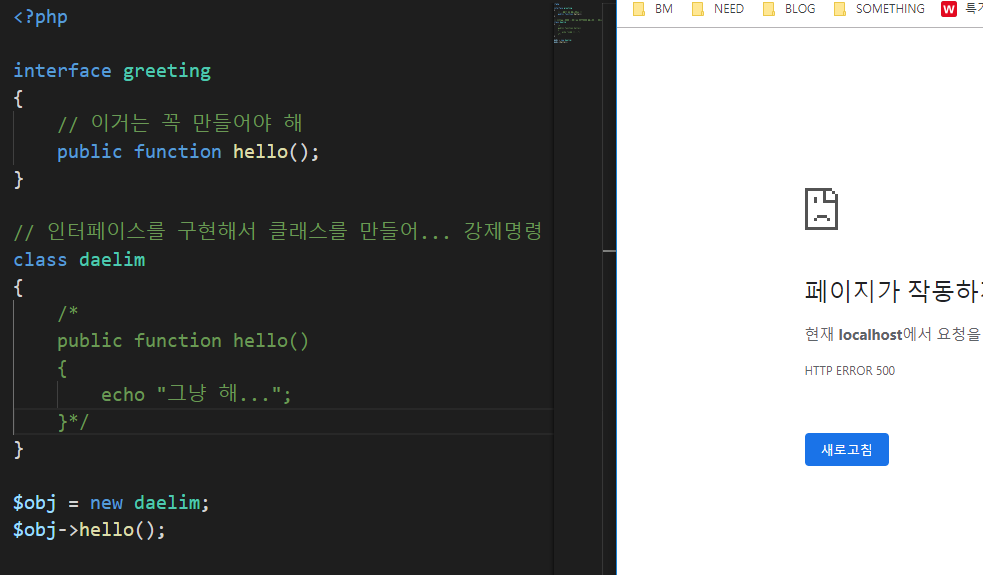
 18번째 줄은 실행되지않음



Daelim을 상속받으면 daelim 안에 있는 메소드 호출 가능

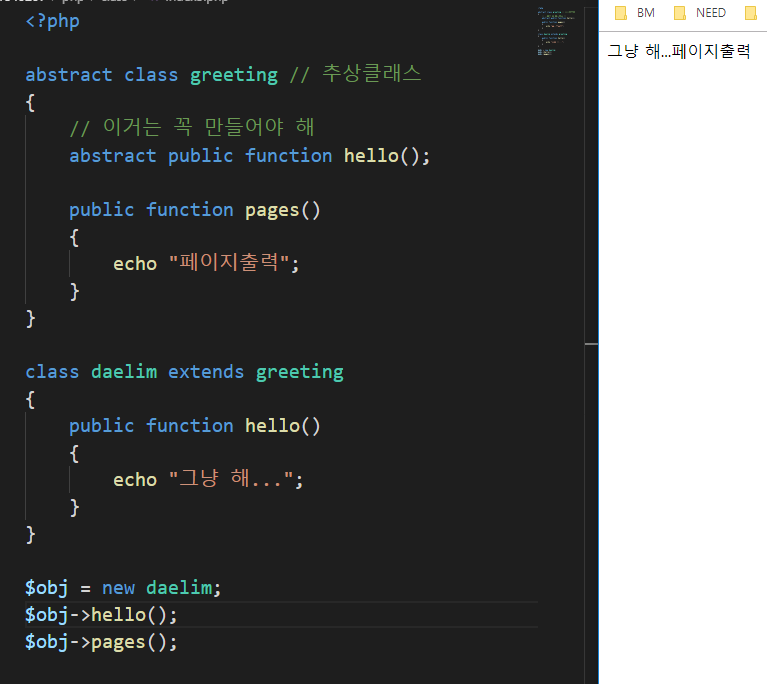


ㄴ오버라이딩 (앞에 final을 쓰면 오버라이딩 불가)



interface안에 있는 메소드는 무조건 만들어야 함/ 안 만들면 오류

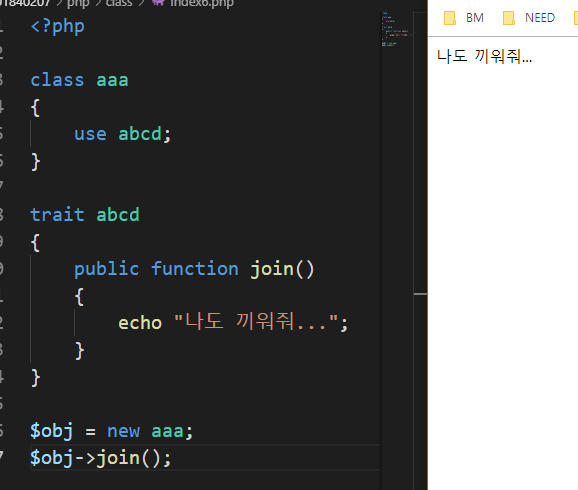
interface는 implements로 상속받음



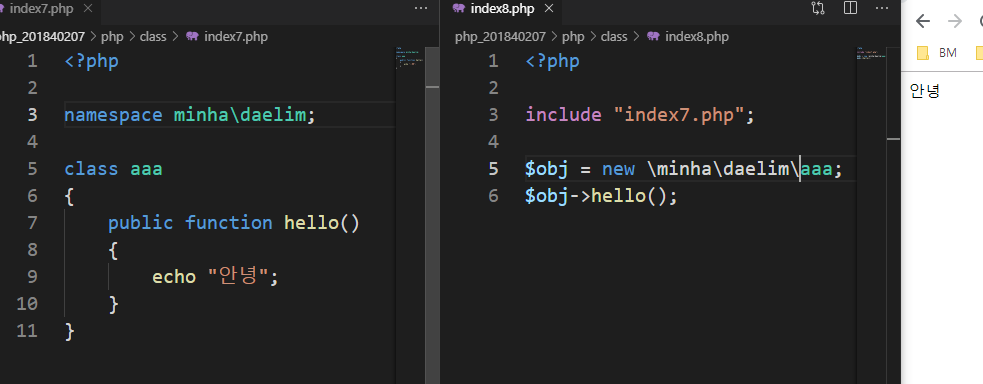
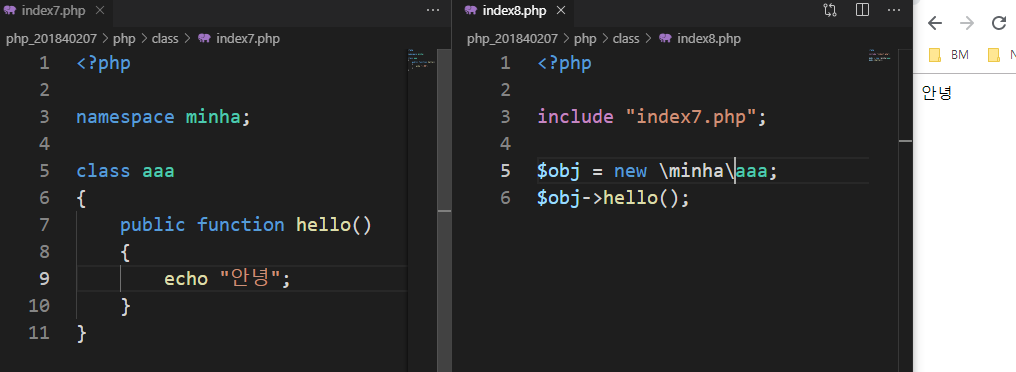
추상클래스는 extends로 상속받음

앞에 abstract 붙임

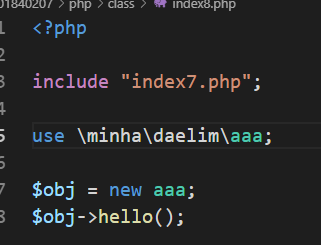
인터페이스랑 추상클래스는 인스턴스화 불가

 trait 혼자서 객체를 만들 수 없음(상속가능)

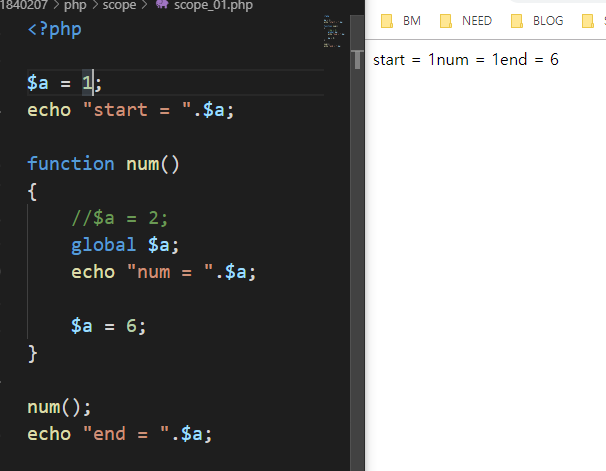
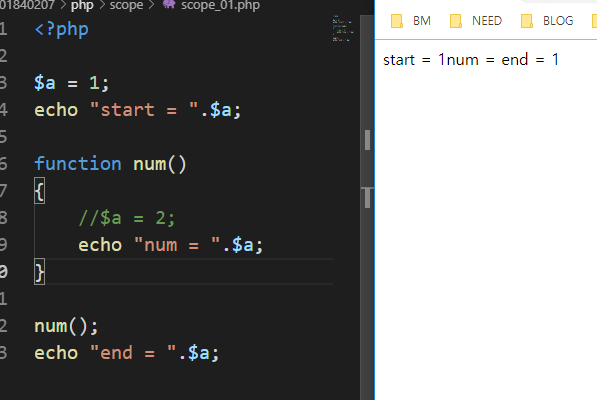
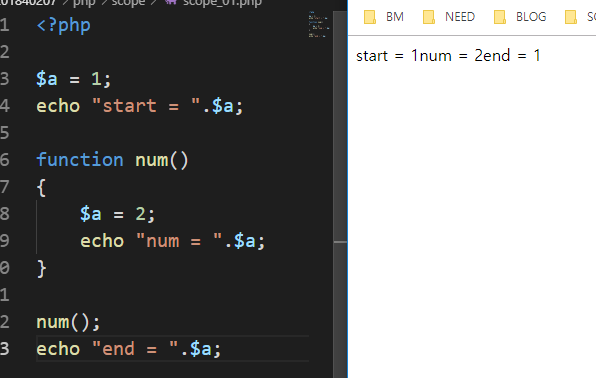
new trait / new 추상클래스 / new 인터페이스 불가 / new 클래스 만 가능

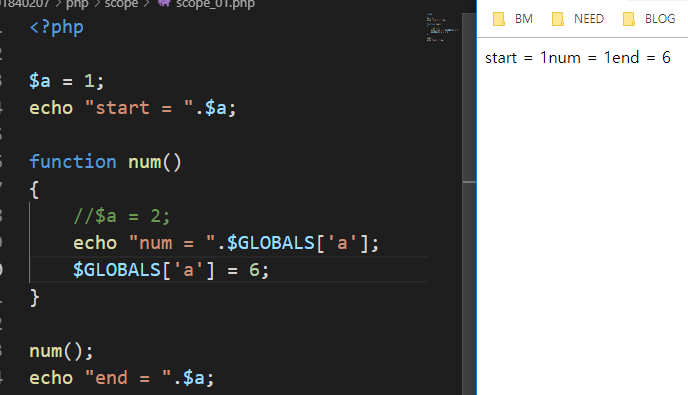


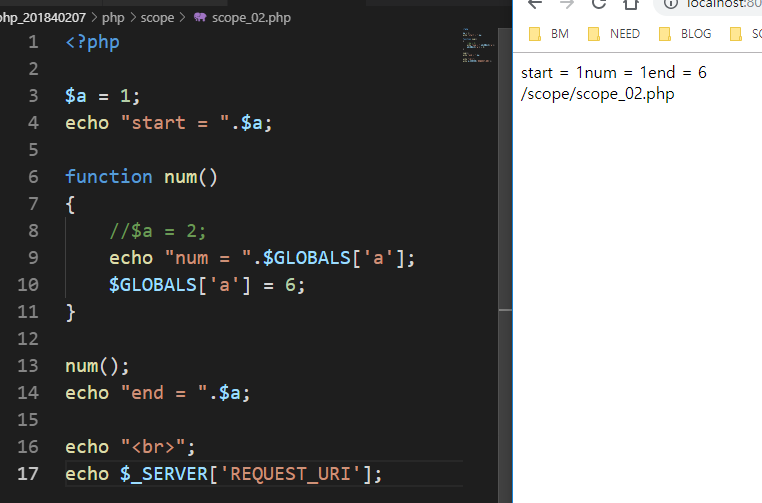
닉네임 설정 (namespace)

 별칭이 너무 길면 use를 써서 간단히 쓸 수 있음



 global을 쓰면 바깥에 있는 변수를 쓸 수 있음

 이렇게 쓸 수도 있음



REQUEST\_URI를 쓰면 경로 보여줌