**static?** 클래스 변수 & 클래스 메소드

- 어떠한 인스턴스에도 속하지 않는 상태로 메모리 공간에 단 1개만 존재한다.

- 인스턴스가 생성되기 전, 가상머신이 해당 클래스를 읽음과 동시에 메모리 공간에 할당된다.

static은 \*global state를 상징함. global state는 추론과 테스트가 매우 까다롭다.

**첫째, 객체 지향적이지 않다:** static 변수가 객체 지향의 패러다임과 상반된다.

static은 각 객체의 데이터들이 캡슐화 되어야 한다는 객체 지향 프로그래밍의 원칙에 위반 된다. static은 스코프를 고려할 필요가 없는 경우, 즉 전역 변수를 사용할 때 유용하다. 이는 절차지향적 프로그래밍 혹은 명령형 프로그래밍에서 중요한 개념이다.

Java에서 static을 사용하면 예측하기에 어려운 문제가 많아지게 된다. 예를 들어 하나의 가상머신에서 2개의 복사본이 실행된다고 가정해보자. 만약 이 두 파일이 동일한 static 변수를 공유하게 된다면, 서로의 상태에 영향을 주게 되지 않을까? \*오버 라이딩을 할 수 없는 static 멤버들 때문에 클래스를 확장하는 것이 어려워질 것이다. 뿐만 아니라 지나치게 많은 static 변수를 사용하게 되면 이 들로부터의 메모리 회수가 불가능 하기 때문에 가상머신이 메모리 부족을 겪게 될 것이다.

**둘째, 프로그램이 실행되고 있는 내내 살아있다:** 미 사용 메모리 회수가 불가능하다.

해당 클래스를 이용한 작업이 끝나더라도 static멤버가 점유하고 있는 메모리는 \*garbage collector에 의해서 회수되지 않아 메모리에 계속 남아있다. 반대로, 프로그래머가 그 변수를 static멤버로 만들지 않고, 인스턴스화 해서 main() 함수 내에 하나의 인스턴스로 생성 후 그 인스턴스에게 만 번의 함수 호출을 시키게 되면 만 번의 함수 호출이 끝난 후에 해당 인스턴스는 메모리 상에서 소멸된다. 따라서 인스턴스화 하는 것이 메모리를 훨씬 절약할 수 있게 된다.

**셋째, 재사용성이 떨어진다:** static 메소드는 인터페이스를 구현하는데 사용될 수 없다.

즉 static 메소드는 프로그래머가 (재사용성을 높여주는) Java의 유용한 객체지향적 기능들을 사용하는 것을 방해한다.

**static의 대안들:** 프로그래머에게 효율성은 가장 중요한 문제이다. 객체를 생성할 때 마다 생기는 사소한 불이익에도 민감한 상황일 수 있다. 이 경우에도 static 대신 다른 방법들을 사용하는 것이 가능하다. 먼저 “transient”나 “volatile”과 같은 제어자(modifier)를 쓸 수 있는지 먼저 고려해보자. 실행 속도를 빠르게 해주는 \*메소드 인라이닝을 위해 “final” 메소드를 사용하는 것도 생각해볼 수 있다. 또한 메소드의 parameter들과 변수들이 final로 선언이 되면 컴파일러 단에서의 최적화 작업이 가능해진다. 인스턴스를 사용할 때마다 새로 생성하는 대신에 여러 번 재사용할 수도 있다. 어쩌면 멀티 스레드를 이용해서 멀티코어 프로세스의 장점을 극대화하기 위해선 이런 디자인이 필수적일 수도 있다. \*이식성이 중요한 것이 아니라면, native 메소드를 사용 static을 사용하는 것보다 더 효율적으로 만들 수도 있을 것이다.

\* global state: 전역 상태. 프로그램 혹은 한 모듈 전체의 상태

\* garbage collector: 사용하지 않는 메모리를 회수하는 기능

\* 오버 라이딩: 1개의 클래스 내에 매개변수의 선언이 다른 동일한 이름의 메소드가 존재하는 것

\* 메소드 인라이닝: 실제 메소드를 호출하지 않고 바로 결과값을 돌려주는 방식

\* 이식성: 다른 플랫폼으로 쉽게 옮길 수 있는 특성

출처: <https://unabated.tistory.com/m/entry/%EC%99%9C-%EC%9E%90%EB%B0%94%EC%97%90%EC%84%9C-static%EC%9D%98-%EC%82%AC%EC%9A%A9%EC%9D%84-%EC%A7%80%EC%96%91%ED%95%B4%EC%95%BC-%ED%95%98%EB%8A%94%EA%B0%80>