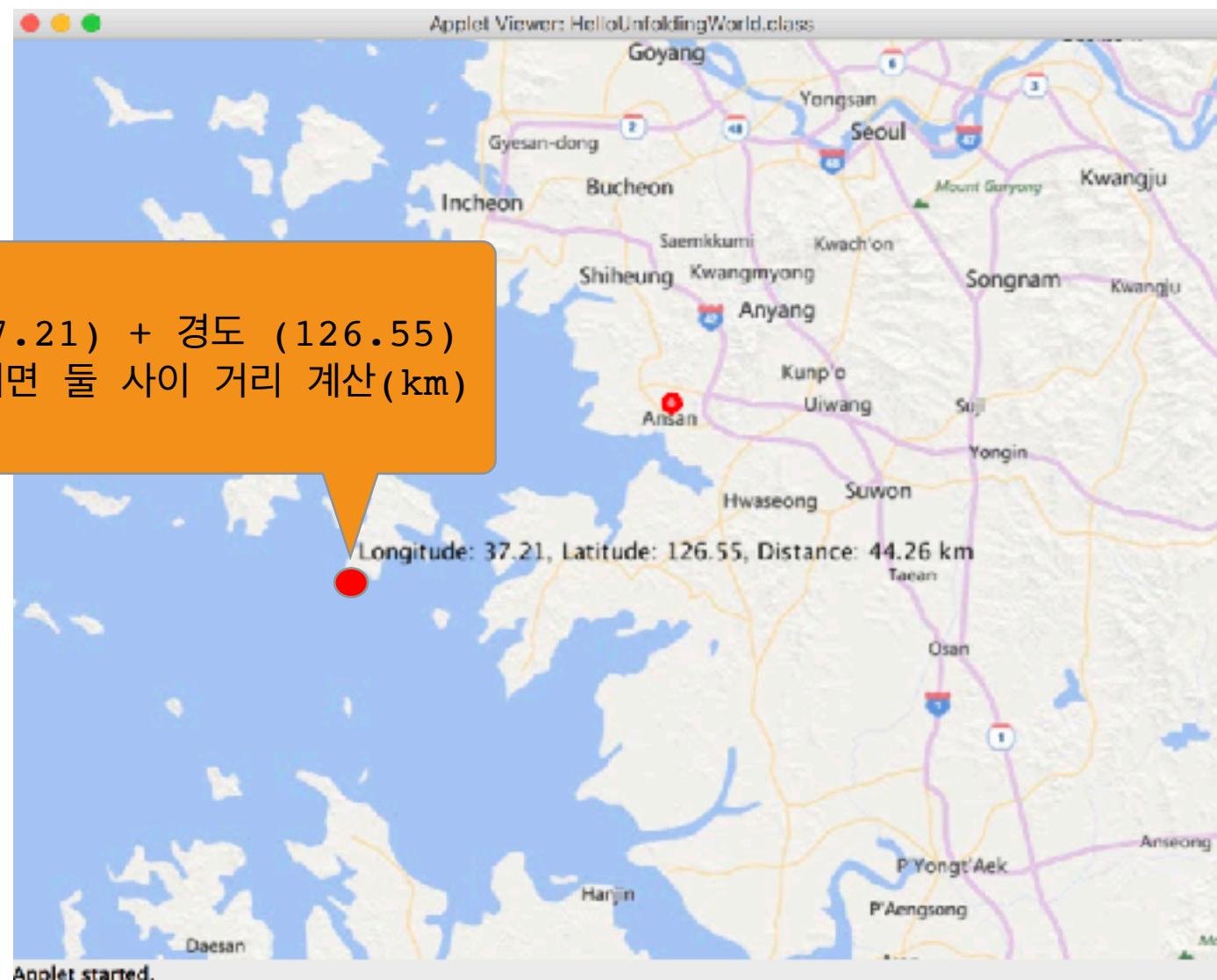


3

부품구조:  
클래스와 메소드

# 간단한 지도 프로그램



# 관찰

- 수 많은 위치가 존재 가능
  - 한양대 에리카의 위치, 여의도의 위치, ...
- 모든 위치들은 위도와 경도로 표현됨
- 두 위치가 정해지면 항상 동일한 방법으로 거리를 계산할 수 있음

# 관찰

개개의 위치는 하나의 물건 (object)

- 수 많은 위치가 존재 가능
  - 한양대 에리카의 위치, 여의도의 위치, ...
- 모든 위치들은 위도와 경도로 표현됨
  - 각 위치는 위도와 경도라는 각기 고유한 속성값을 가짐 (필드)
- 두 위치가 정해지면 항상 동일한 방법으로 거리를 계산할 수 있음

두 위치의 위도, 경도를 변수로 표현하고, 변수들간의 연산으로 위치 값을 계산하는 방법 표현 가능 (메소드)

# 클래스의 필요

- 위치들의 공통점
  - 위도와 경도로 표현
  - 다른 위치가 주어졌을 때 거리를 계산할 수 있어야.
- 임의의 위치가 여러개 생성 될 수 있어야 함
- 위치들의 “틀”을 정의하고 그 틀로부터 무수히 많은 위치 뽑아내기.

# 클래스 정의

```
/** Location 클래스: 위도 경도 표현 및 다른 위치까지의 거리 계산 */

public class Location
{ /** 생성자: 초기화한다 */
    public double latitude;
    public double longitude;
    public Location(double lat, double lon) // 생성 메소드
    {
        this.latitude = lat;
        this.longitude = lon;
    }
    public double distance(Location other) {
        ...
    }
}
```

# 클래스 정의

```
/** Location 클래스: 위도 경도 표현 및 다른 위치까지의 거리 계산 */
```

```
public class Location
{ /** 생성자: 초기화한다 */
    public double latitude;
    public double longitude;

    public Location(double lat, double lon) // 생성 메소드
    {
        this.latitude = lat;
        this.longitude = lon;
    }

    public double distance(Location other) {
        ...
    }
}
```

**속성값들 (Fields):**  
객체들이 저장해야 하는 정보들  
(위도, 경도)

# 클래스 정의

```
/** Location 클래스: 위도 경도 표현 및 다른 위치까지의 거리 계산 */
public class Location
{ /** 생성자: 초기화한다 */
    public double latitude;
    public double longitude;

    public Location(double lat, double lon) // 생성 메소드
    {
        this.latitude = lat;
        this.longitude = lon;
    }
    public double distance(Location other) {
        ...
    }
}
```

**메소드 (Methods):**  
이 클래스가 할 수 있는 일들

# 클래스 정의

```
/** Location 클래스: 위도 경도 표현 및 다른 위치까지의 거리 계산 */
public class Location
{ /** 생성자: 초기화한다 */
    public double latitude;
    public double longitude;

    public Location(double lat, double lon) // 생성 메소드
    {
        this.latitude = lat;
        this.longitude = lon;
    }

    public double distance(Location other) {
        ...
    }
}
```

**생성자(Constructor)**:  
새로운 객체를 만들기 위한 메소드

# 객체 생성과 사용

```
public class Location { ...  
    public double distance(Location other) {  
        return getDist(this.latitude, this.longitude,  
                      other.latitude, other.longitude);  
    }  
}  
  
public class LocationTester {  
    public static void main(String[] args) {  
        Location erica = new Location(37.32, 126, 83);  
        Location seoul = new Location(37.55, 127.04);  
        System.out.println (erica.distance(seoul));  
    }  
}
```

# 객체 생성과 사용

```
public class Location { ...  
    public double distance(Location other) {  
        return getDist(this.latitude, this.longitude,  
                      other.latitude, other.longitude);  
    }  
}  
  
public class LocationTester {  
    public static void main(String[] args) {  
        Location erica = new Location(37.32, 126, 83);  
        Location seoul = new Location(37.55, 127.04);  
        System.out.println (erica.distance(seoul));  
    }  
}
```

**this:** 현재 이 메소드를 호출하고 있는 객체

# 객체의 생성과 사용 내부과정

- 함수의 인자 전달 (parameter passing)
- 메모리 모델 (memory model)
- 변수의 유효 범위 (variable scope)

# 메모리 모델

```
int var1 = 52; ←  
Location erica = new Location(37.32, 126, 83);  
Location seoul = new Location(37.55, 127.04);  
seoul.longitude = 127.05;
```

var1

52

# 메모리 모델

```

int var1 = 52;

Location erica = new Location(37.32, 126,83);
Location seoul = new Location(37.55, 127.04);
seoul.longitude = 127.05;
    
```



var1

52

erica

@34

힙 메모리 영역 (Heap)

latitude

37.32

longitude

126.83

@34

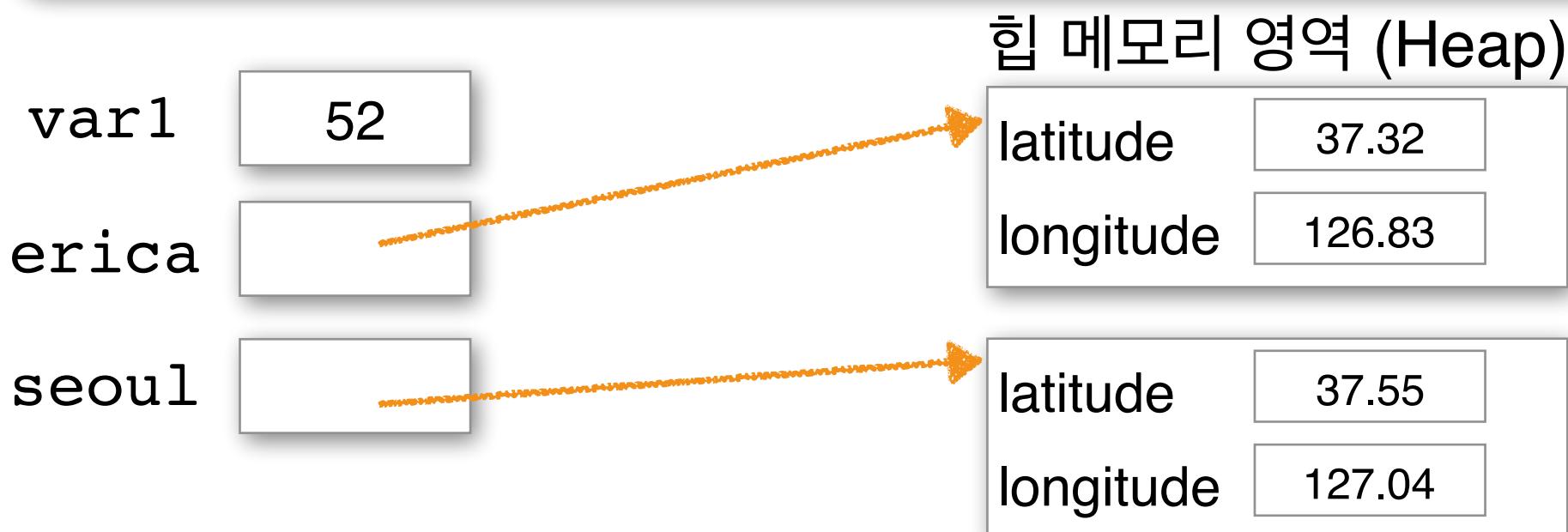
# 메모리 모델

```
int var1 = 52;  
  
Location erica = new Location(37.32, 126,83);  
  
Location seoul = new Location(37.55, 127.04);  
  
seoul.longitude = 127.05;
```



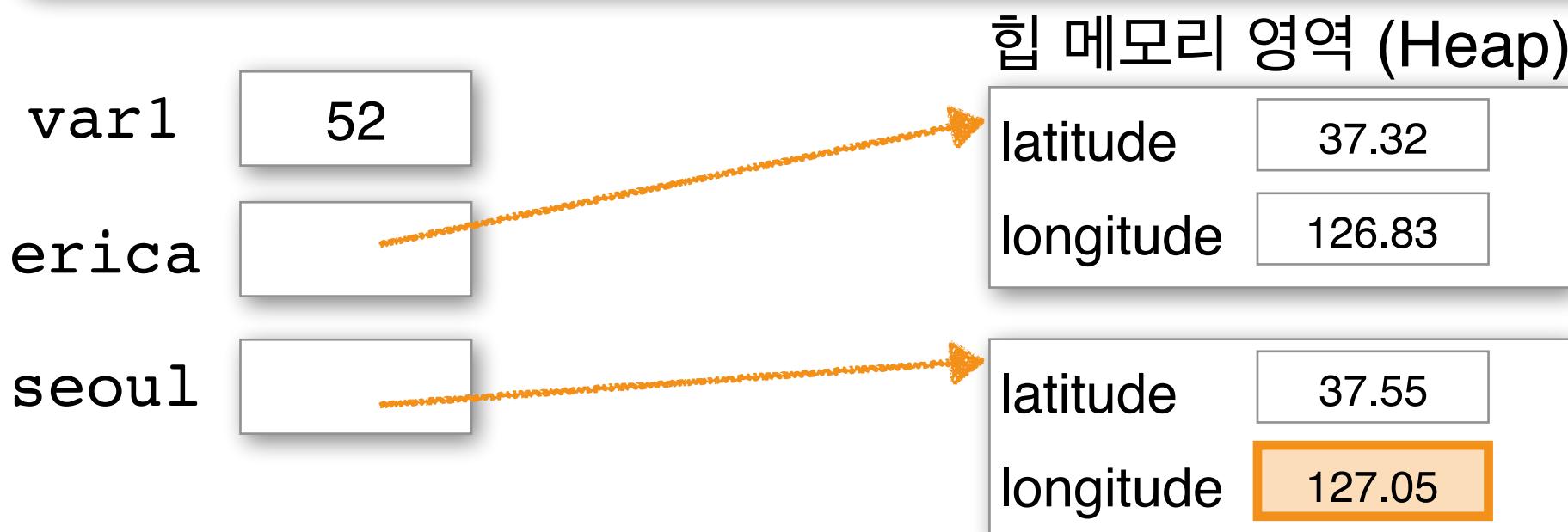
# 메모리 모델

```
int var1 = 52;  
  
Location erica = new Location(37.32, 126,83);  
Location seoul = new Location(37.55, 127.04);  
seoul.longitude = 127.05;
```



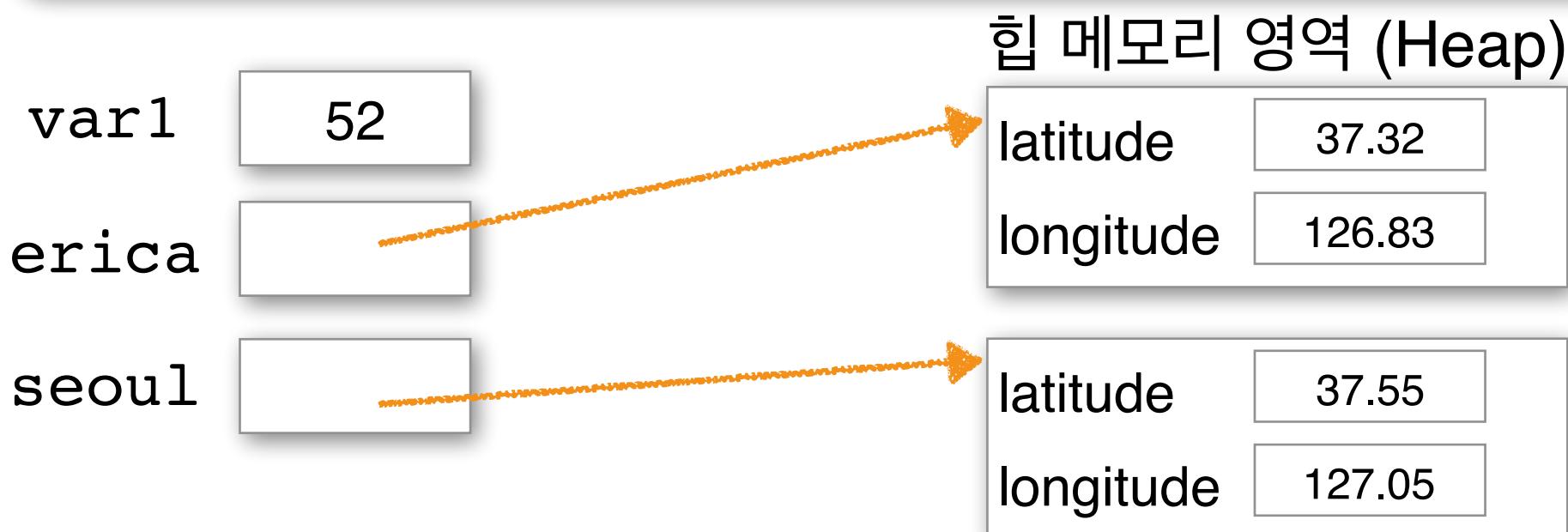
# 메모리 모델

```
int var1 = 52;  
  
Location erica = new Location(37.32, 126,83);  
  
Location seoul = new Location(37.55, 127.04);  
  
seoul.longitude = 127.05;
```



# 메모리 모델

```
int var1 = 52;  
  
Location erica = new Location(37.32, 126,83);  
Location seoul = new Location(37.55, 127.04);  
seoul.longitude = 127.05;
```



# 변수의 유효범위 (Variable Scope)

- 총 사용된 변수의 갯수는?

```
public class Location {  
    public double lat;  
    public double lon;  
    public Location(double latIn,  
                   double lonIn){  
        this.lat = latIn;  
        this.lon = lonIn;  
    }  
    ...  
}
```

```
public class LocationTester {  
    public static void main(String[] args) {  
        Loc loc1 = new Loc(39.9,116.4);  
    }  
}
```

# 변수의 유효범위 (Variable Scope)

- 총 사용된 변수의 갯수는?

```
public class Location {  
    public double lat;  
    public double lon;  
    public Location(double latIn,  
                   double lonIn) {  
        this.lat = latIn;  
        this.lon = lonIn;  
    }  
    ...  
}
```

```
public class LocationTester {  
    public static void main(String[] args) {  
        Loc loc1 = new Loc(39.9, 116.4);  
    }  
}
```

# 변수의 유효범위 (Variable Scope)

```
public class LocationTester {  
    public static void main(String[] args) {  
        Loc loc1 = new Loc(39.9,116.4);  
        latitude = 12.04;  
    }  
}
```

ERROR. 변수 `latitude` 가 정의되지 않았음.

변수의 유효범위는 변수의 값을 할당하고 사용할 수 있는 범위

# 변수의 유효범위 (Variable Scope)

```
public class LocationTester {  
    public static void main(String[] args) {  
        Loc loc1 = new Loc(39.9,116.4);  
        latitude = 12.04;  
    }  
}
```

지역 변수 (Local variables)는 메소드 안에서만 값이 정의되고 사용될 수 있음.

# 변수의 유효범위 (Variable Scope)

```
public class Location {  
    public double lat;  
    public double lon;  
    public Location(double latIn,  
                   double lonIn){  
        this.lat = latIn;  
        this.lon = lonIn;  
    }  
    ...  
}
```

매개변수(Parameters)  
는 지역 변수

# 변수의 유효범위 (Variable Scope)

```
public class Location {  
    public double lat;  
    public double lon;  
    public Location(double latIn,  
                   double lonIn){  
        this.lat = latIn;  
        this.lon = lonIn;  
    }  
    ...  
}
```

멤버 변수들은 이 클래스의 어떠한 메소드에서도 값이 정의되고 사용될 수 있음.

# 변수의 유효범위 (Variable Scope)

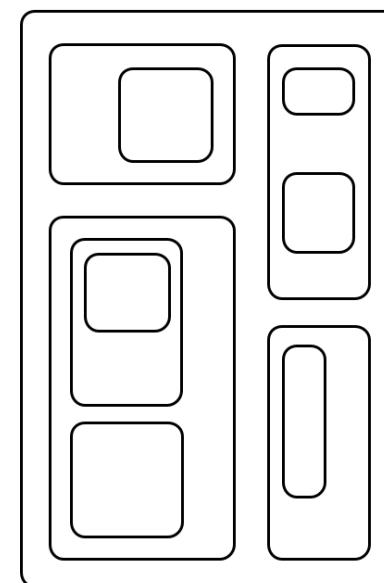
- 이런 간단한 유효범위는 수리논술의 2000년 전통

**Theorem** The intersection of all addition-closed sets is addition-closed.

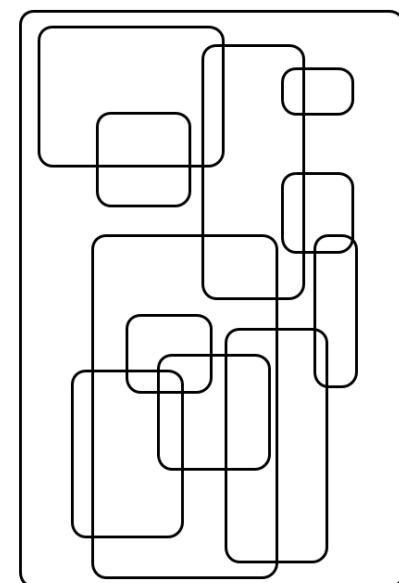
**Proof.** Let  $S$  be the intersection set. Let  $x$  and  $y$  be elements of  $S$ . Because  $x$  and  $y$  are elements of ... hence in  $S$ .

# 변수의 유효범위 (Variable Scope)

- 이름의 유효범위가 한정됨에 따라
  - 이름 재사용 가능
  - 전체 프로그램의 모든 이름을 외울 필요 없음
  - 이름이 필요한 곳에만 알려짐
- 이름의 유효범위는 쉽게 결정됨  
(프로그램 텍스트에서 결정:  
lexical scoping)



YES



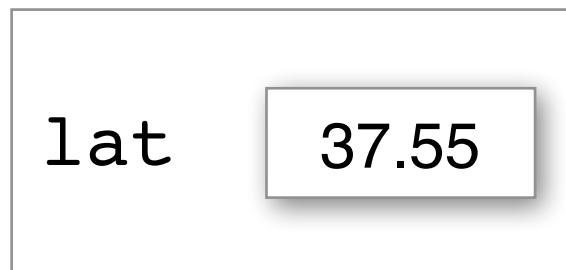
No

```
public class LocationTester {  
    public static void main(String[] args) {  
        double lat = 37.55; ←  
        Location seoul = new Location(37.55, 127.04);  
    }  
}
```

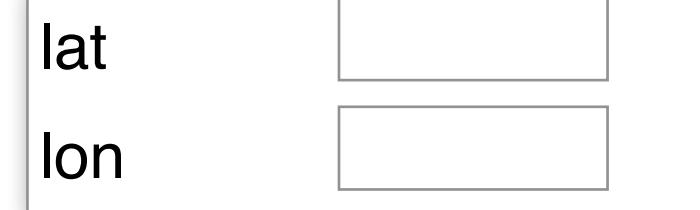
lat      37.55

main의 환경

```
public class LocationTester {  
    public static void main(String[] args) {  
        double lat = 37.55;  
        Location seoul = new Location(37.55, 127.04);  
    }  
}
```



힙 메모리 영역 (Heap)



```

public class Location {
    public double lat;
    public double lon;
    public Location(double latIn, double lonIn) { ←
        this.lat = latIn;
        this.lon = lonIn;
    }
}

```

lat 37.55

main의 환경

|       |        |
|-------|--------|
| latIn | 37.55  |
| lonIn | 127.04 |
| this  |        |

생성자의 환경

힙 메모리 영역 (Heap)

|     |  |
|-----|--|
| lat |  |
| lon |  |

```
public class Location {
    public double lat;
    public double lon;
    public Location(double latIn, double lonIn) {
        this.lat = latIn; ←
        this.lon = lonIn;
    }
}
```

lat 37.55

main의 환경

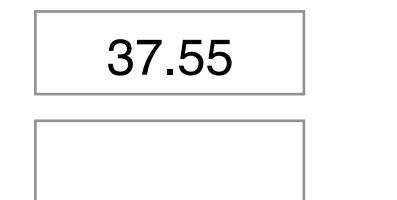
|       |        |
|-------|--------|
| latIn | 37.55  |
| lonIn | 127.04 |
| this  |        |

생성자의 환경

힙 메모리 영역 (Heap)

lat 37.55

lon



```

public class Location {
    public double lat;
    public double lon;
    public Location(double latIn, double lonIn) {
        this.lat = latIn;
        this.lon = lonIn; ←
    }
}

```

lat 37.55

main의 환경

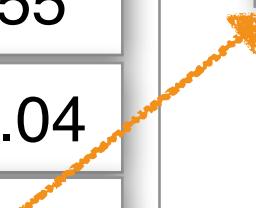
|       |        |
|-------|--------|
| latIn | 37.55  |
| lonIn | 127.04 |
| this  |        |

생성자의 환경

힙 메모리 영역 (Heap)

lat 37.55

lon 127.04



```
public class Location {
    public double lat;
    public double lon;
    public Location(double latIn, double lonIn) {
        this.lat = latIn;
        this.lon = lonIn;
        return this;
    }
}
```

생성자 메소드는 항상  
**this** 를 반환

lat 37.55

main의 환경

|       |        |
|-------|--------|
| latIn | 37.55  |
| lonIn | 127.04 |
| this  |        |

생성자의 환경

힙 메모리 영역 (Heap)

|     |        |
|-----|--------|
| lat | 37.55  |
| lon | 127.04 |

```
public class LocationTester {  
    public static void main(String[] args) {  
        double lat = 37.55;  
        Location seoul = new Location(37.55, 127.04); ←  
    }  
}
```



```

public class Location {
    public double lat;
    public double lon;
    public Location(double latIn, double lonIn) {
        this.lat = latIn;
        this.lon = lonIn;
    }
}

```

**this 생략 가능**

lat 37.55

main의 환경

|       |        |
|-------|--------|
| latIn | 37.55  |
| lonIn | 127.04 |
| this  |        |

생성자의 환경

힙 메모리 영역 (Heap)

|     |  |
|-----|--|
| lat |  |
| lon |  |

```

public class Location {
    public double lat;
    public double lon;
    public Location(double latIn, double lonIn) {
        lat = latIn; ←
        lon = lonIn;
    }
}

```

생성자 환경에서 변수  
lat 찾음 → 없음!

lat 37.55

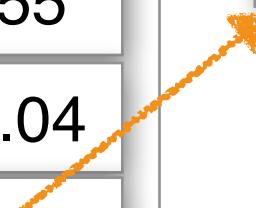
main의 환경

|       |        |
|-------|--------|
| latIn | 37.55  |
| lonIn | 127.04 |
| this  |        |

생성자의 환경

힙 메모리 영역 (Heap)

|     |  |
|-----|--|
| lat |  |
| lon |  |



```
public class Location {
    public double lat;
    public double lon;
    public Location(double latIn, double lonIn) {
        lat = latIn; ←
        lon = lonIn;
    }
}
```

작체의 환경  
(`this`가 가리키는)  
에서 `lat` 찾음

`lat` 37.55

main의 환경

|                    |        |
|--------------------|--------|
| <code>latIn</code> | 37.55  |
| <code>lonIn</code> | 127.04 |
| <code>this</code>  |        |

생성자의 환경

힙 메모리 영역 (Heap)

|                  |  |
|------------------|--|
| <code>lat</code> |  |
| <code>lon</code> |  |

```
public class Location {
    public double lat;
    public double lon;
    public Location(double latIn, double lonIn) {
        lat = latIn; ←
        lon = lonIn;
    }
}
```

작체의 환경  
(`this`가 가리키는)  
에서 `lat` 찾음

lat

37.55

main의 환경

latIn

37.55

lonIn

127.04

this

생성자의 환경

힙 메모리 영역 (Heap)

lat

37.55

lon



# 메모리 모델 심화

```
public class Loc {
    public double lat;
    public double lon;
    public Loc(double latIn,
              double lonIn)
        this.lat = latIn;
        this.lon = lonIn;
    }
    ...
}
```

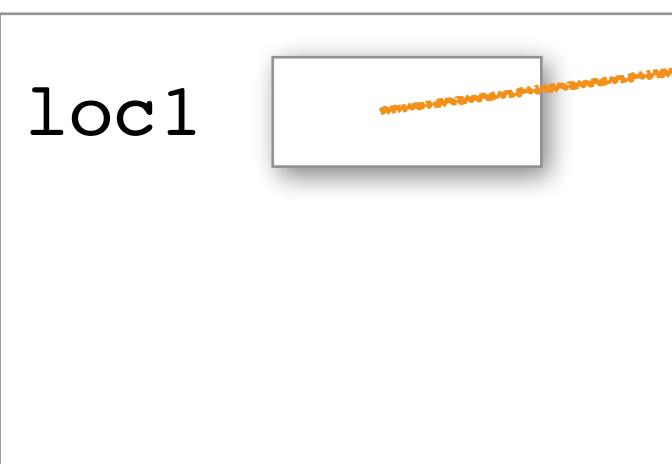
```
public class LocationTester {
    public static String foo () {
        Loc loc1 = new Loc(39.9,116.4);
        Loc loc2 = new Loc(55.8,37.6);
        loc1 = loc2;
        loc1.lat = -8.3;
        return (loc2.lat + ", " + loc2.lon);
    }
    public static void main(String[] args) {
        System.out.println(foo());
    }
}
```

LocationTester 클래스의 메인 함수 실행 후 결과는?

```
public class Loc {
    public double lat;
    public double lon;
    public Loc(double latIn,
              double lonIn)
        this.lat = latIn;
        this.lon = lonIn;
    }
    ...
}
```

```
public class LocationTester {
    public static String foo () {
        Loc loc1 = new Loc(39.9,116.4); ←
        Loc loc2 = new Loc(55.8,37.6);
        loc1 = loc2;
        loc1.lat = -8.3;
        return (loc2.lat + ", " + loc2.lon);
    }
    public static void main(String[] args) {
        System.out.println(foo());
    }
}
```

foo의 환경



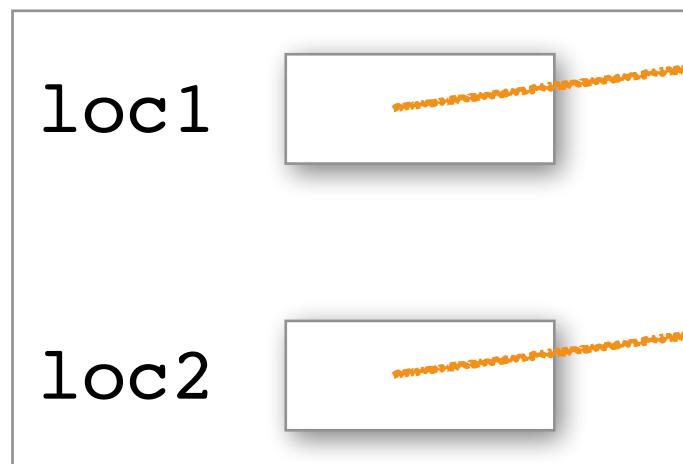
힙 메모리 영역 (Heap)

|     |       |
|-----|-------|
| lat | 39.9  |
| lon | 116.4 |

```
public class Loc {
    public double lat;
    public double lon;
    public Loc(double latIn,
              double lonIn)
        this.lat = latIn;
        this.lon = lonIn;
    }
    ...
}
```

```
public class LocationTester {
    public static String foo () {
        Loc loc1 = new Loc(39.9,116.4);
        Loc loc2 = new Loc(55.8,37.6); ←
        loc1 = loc2;
        loc1.lat = -8.3;
        return (loc2.lat + ", " + loc2.lon);
    }
    public static void main(String[] args) {
        System.out.println(foo());
    }
}
```

foo의 환경



힙 메모리 영역 (Heap)

|     |       |
|-----|-------|
| lat | 39.9  |
| lon | 116.4 |
| lat | 55.89 |
| lon | 37.6  |

```
public class Loc {
    public double lat;
    public double lon;
    public Loc(double latIn,
              double lonIn)
        this.lat = latIn;
        this.lon = lonIn;
    }
    ...
}
```

```
public class LocationTester {
    public static String foo () {
        Loc loc1 = new Loc(39.9,116.4);
        Loc loc2 = new Loc(55.8,37.6);
        loc1 = loc2; ←
        loc1.lat = -8.3;
        return (loc2.lat + ", " + loc2.lon);
    }
    public static void main(String[] args) {
        System.out.println(foo());
    }
}
```

### foo의 환경

loc1 @1

loc2 @2

### 힙 메모리 영역 (Heap)

|     |       |
|-----|-------|
| lat | 39.9  |
| lon | 116.4 |

|     |       |
|-----|-------|
| lat | 55.89 |
| lon | 37.6  |

@1

@2

```
public class Loc {
    public double lat;
    public double lon;
    public Loc(double latIn,
              double lonIn)
        this.lat = latIn;
        this.lon = lonIn;
    }
    ...
}
```

```
public class LocationTester {
    public static String foo () {
        Loc loc1 = new Loc(39.9,116.4);
        Loc loc2 = new Loc(55.8,37.6);
        loc1 = loc2; ←
        loc1.lat = -8.3;
        return (loc2.lat + ", " + loc2.lon);
    }
    public static void main(String[] args) {
        System.out.println(foo());
    }
}
```

### foo의 환경



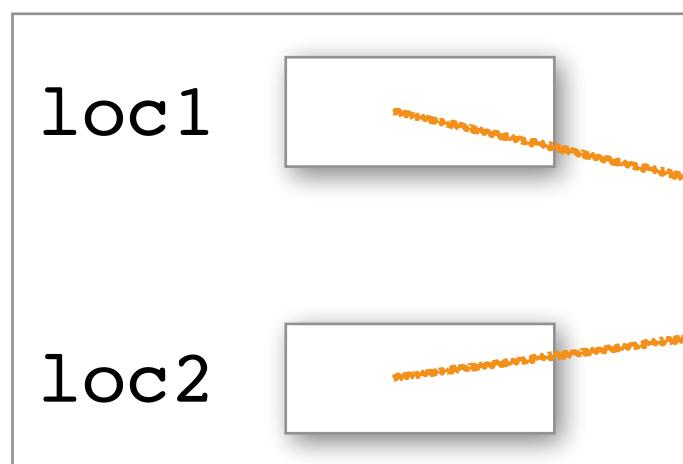
### 힙 메모리 영역 (Heap)

|     |       |
|-----|-------|
| lat | 39.9  |
| lon | 116.4 |
| lat | 55.89 |
| lon | 37.6  |

```
public class Loc {
    public double lat;
    public double lon;
    public Loc(double latIn,
              double lonIn)
        this.lat = latIn;
        this.lon = lonIn;
    }
    ...
}
```

```
public class LocationTester {
    public static String foo () {
        Loc loc1 = new Loc(39.9,116.4);
        Loc loc2 = new Loc(55.8,37.6);
        loc1 = loc2;
        loc1.lat = -8.3; ←
        return (loc2.lat + ", " + loc2.lon);
    }
    public static void main(String[] args) {
        System.out.println(foo());
    }
}
```

### foo의 환경



### 힙 메모리 영역 (Heap)

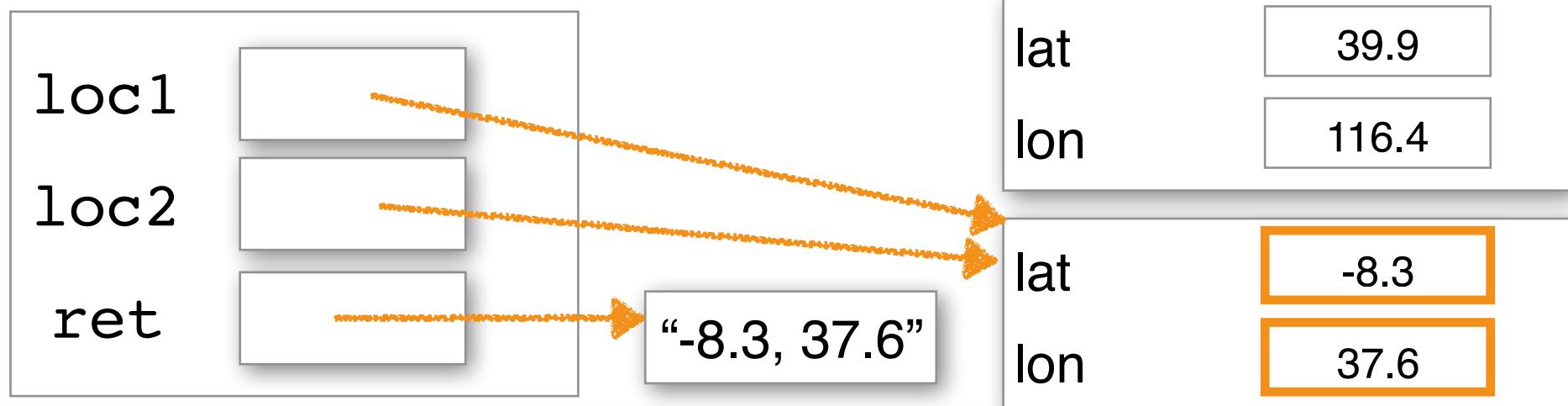
|     |       |
|-----|-------|
| lat | 39.9  |
| lon | 116.4 |
| lat | -8.3  |
| lon | 37.6  |

```
public class Loc {
    public double lat;
    public double lon;
    public Loc(double latIn,
              double lonIn)
        this.lat = latIn;
        this.lon = lonIn;
    }
    ...
}
```

```
public class LocationTester {
    public static String foo () {
        Loc loc1 = new Loc(39.9,116.4);
        Loc loc2 = new Loc(55.8,37.6);
        loc1 = loc2;
        loc1.lat = -8.3;
        return (loc2.lat + ", " + loc2.lon);
    }
    public static void main(String[] args) {
        System.out.println(foo());
    }
}
```

foo의 환경

힙 메모리 영역 (Heap)



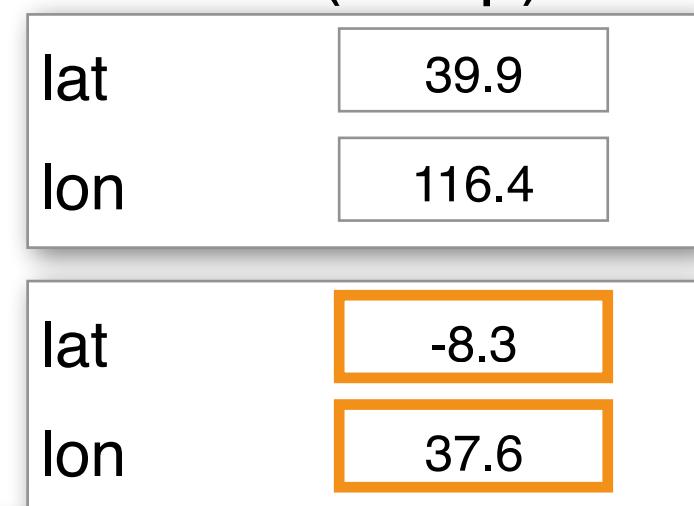
```
System.out
```

```
println(String x) {  
...  
}
```

```
public class LocationTester {  
    public static String foo () {  
        Loc loc1 = new Loc(39.9,116.4);  
        Loc loc2 = new Loc(55.8,37.6);  
        loc1 = loc2;  
        loc1.lat = -8.3;  
        return (loc2.lat + ", " + loc2.lon);  
    }  
    public static void main(String[] args) {  
        System.out.println(foo()); ←  
    }  
}
```

## 힙 메모리 영역 (Heap)

### println의 환경



# 반환값에 대한 소소한 주의

- return 이후의 코드는 실행되지 않는다.
  - return 문은 메소드 수행 종료를 의미
- 반환값을 받지 않아도 무방하다.
  - 값을 받지 않아도 메소드는 수행됨
  - 받지 않은 값은 무시
    - foo();

# 질문

- 앞으로 만들 지도에서 에리카 캠퍼스의 위치 자주 사용
- 하지만 간간히 그와 다른 위치도 생성
- 어떻게 하면 효율적으로 프로그램을 짤 수 있을까?

# 같은 이름의 메소드 여러개 정의 가능

```
public class Location {  
    public double latitude;  
    public double longitude;  
    public Location() // 기본 생성 메소드  
    {  
        this.latitude = 37.32;  
        this.longitude = 126.83;  
    }  
  
    public Location(double lat, double lon) // 생성 메소드  
    {  
        this.latitude = lat;  
        this.longitude = lon;  
    }  
    public double distance(Location other) {  
        ...  
    }  
}
```

디폴트 생성자  
(에리카 캠퍼스 위치)

# 같은 이름의 메소드 여러개 정의 가능

```
public class Location {  
    ...  
    public double distance(Location other) {  
        ...  
    }  
  
    public double distance(double lat, double lon) {  
        ...  
    }  
}
```



```
public class Location {  
    ...  
    public double distance(Location other) {  
        ...  
    }  
  
    public int distance(Location other) {  
        ...  
    }  
}
```



# 같은 이름의 메소드 여러개 정의 가능

- **메소드 오버로딩 (overloading)**: 같은 이름의 메소드 여러 개 정의
- (문제) 리턴 타입이 다르고 이름이 동일한 두 함수를 언제 정의 할 수 있을까?
  1. 항상 불가능
  2. 항상 가능
  3. 두 함수의 매개변수가 다를 경우

# 질문

- 지도 프로그램에서 에리카 캠퍼스의 위도를 100.0 으로 설정하고 실행하면? → 에러발생! ( $| \text{유효한 위도} | \leq 90$ )

```
Location erica = new Location();
erica.latitude = 100.0;
```

```
2018-09-17 18:03:53.632 java[49978:32503121] *** Assertion failure in -[NSEvent _cgsEventRecord], /BuildRoot/Library/Caches/com.apple.xbs/Sources/AppKit/AppKit-1561.20.106/AppKit.subproj/NSEvent.m:1972
Exception in thread "AWT-EventQueue-0" java.lang.RuntimeException: Non-Java exception raised, not handled! (Original problem: Deprecated in 10_12... DO NOT EVER USE CGSEventRecord directly. Bad things, man.... bad things.)
    at apple.awt.ComponentModel._handleEvent(Native Method)
    at apple.awt.ComponentModel.handleEvent(ComponentModel.java:273)
    at java.awt.DefaultKeyboardFocusManager.dispatchKeyEvent(DefaultKeyboardFocusManager.java:753)
    at java.awt.DefaultKeyboardFocusManager.preDispatchKeyEvent(DefaultKeyboardFocusManager.java:1000)
    at java.awt.DefaultKeyboardFocusManager.typeAheadAssertions(DefaultKeyboardFocusManager.java:865)
    at java.awt.DefaultKeyboardFocusManager.dispatchEvent(DefaultKeyboardFocusManager.java:686)
    ...
    at java.awt.EventQueue.pumpEvents(EventDispatchThread.java:188)
    at java.awt.EventQueue.run(EventDispatchThread.java:122)
```

# Public

```
/** Location 클래스: 위도 경도 표현 및 다른 위치까지의 거리 계산 */
```

```
public class Location
{ /** 생성자: 초기화한다 */
    public double latitude;
    public double longitude;

    public Location(double lat, double lon) // 생성 메소드
    {
        this.latitude = lat;
        this.longitude = lon;
    }
    public double distance(Location other) {
        ...
    }
}
```

public 은 어디에서나 접근(읽기/쓰기)이 가능하다는 의미

# 객체 생성과 사용

```
public class Location {  
    public double latitude;  
    public double longitude;  
    public double distance(Location other) {  
        return ...  
    }  
}
```

모두 허용됨 (public이므로)

```
public class LocationTester {  
    public static void main(String[ ] args) {  
        Location erica = new Location(37.32, 126,83);  
        erica.latitude = 100.0;  
        System.out.println (erica.distance(erica));  
    }  
}
```

# Private

```
/** Location 클래스: 위도 경도 표현 및 다른 위치까지의 거리 계산 */

public class Location
{ /** 생성자: 초기화한다 */
    private double latitude;
    private double longitude;

    public Location(double lat, double lon) // 생성 메소드
    {
        this.latitude = lat;
        this.longitude = lon;
    }

    public double distance(Location other) {
        ...
    }
}
```

허용!

# Private

```
public class Location {  
    private double latitude;  
    private double longitude;  
    public double distance(Location other) {  
        return ...  
    }  
}  
  
public class LocationTester {  
    public static void main(String[ ] args) {  
        Location erica = new Location(37.32, 126.83);  
        erica.latitude = 100.0; 허용 안됨!  
        System.out.println (erica.distance(erica));  
    }  
}
```

# Setter

```
/** Location 클래스: 위도 경도 표현 및 다른 위치까지의 거리 계산 */

public class Location
{ /** 생성자: 초기화한다 */
    private double latitude;
    private double longitude;

    ...
    public void setLatitude(double lat) {
        this.latitude = lat;
    }
}
```

# Setter

/\*\* Location 클래스: 위도 경도 표현 및 다른 위치까지의 거리 계산 \*/

```
public class Location
```

```
{ /** 생성자: 초기화한다 */
```

```
private double latitude;
```

```
private double longitude;
```

```
...
```

```
public void setLatitude(double lat) {
```

```
    if (lat < -90 || lat > 90) {
```

```
        System.out.println("Illegal value of latitude!");
```

```
}
```

```
    else { this.latitude = lat; }
```

```
}
```

- 외부에서 옳지 못한 값으로 경도를 수정하는 것을 방지
- 옳은 값이 들어올 경우만 값 변경 허용

# Getter

```
/** Location 클래스: 위도 경도 표현 및 다른 위치까지의 거리 계산 */
```

```
public class Location  
{ /** 생성자: 초기화한다 */  
    private double latitude;  
    private double longitude;  
    ...  
    public void getLatitude() {  
        return latitude;  
    }
```

- **Setter** 때문에 필드값이 **private**이 되어서 외부에서 읽기 접근도 못하게 됨
- **Getter**를 통해 읽기 허용

# Public vs. Private

- 필드들(멤버 변수들)은 Private 으로
- 메소드들은 성질에 따라 Public 혹은 Private 으로

# 이름, 이름, 이름

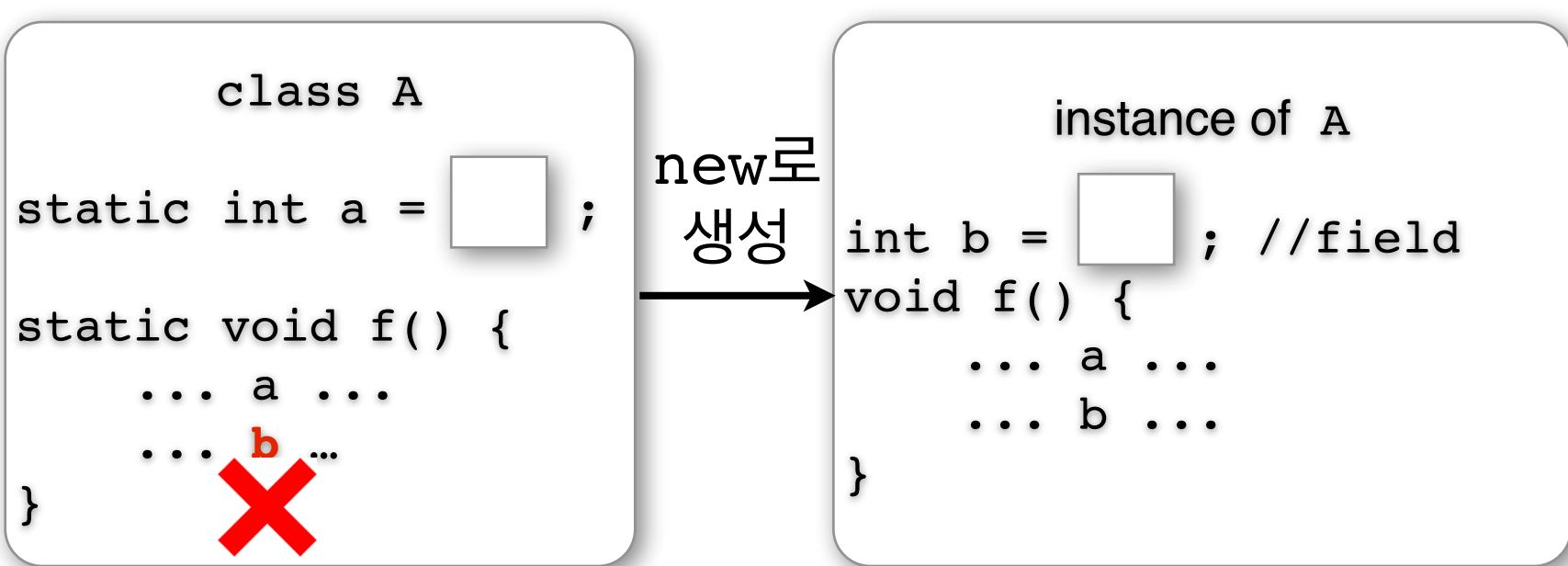
- 이름만 봐도 클래스인지, 메소드인지, 변수인지 알 수 있도록 관례 존재
- 클래스: Location, GregorianCalendar
- 메소드: getLatitude, distance
- 형식인수, 필드, 지역변수: longitude, latitude
  - 필드가 상수처럼 쓰일때는 RADIUS\_EARTH

# 정적 (Static) 메소드

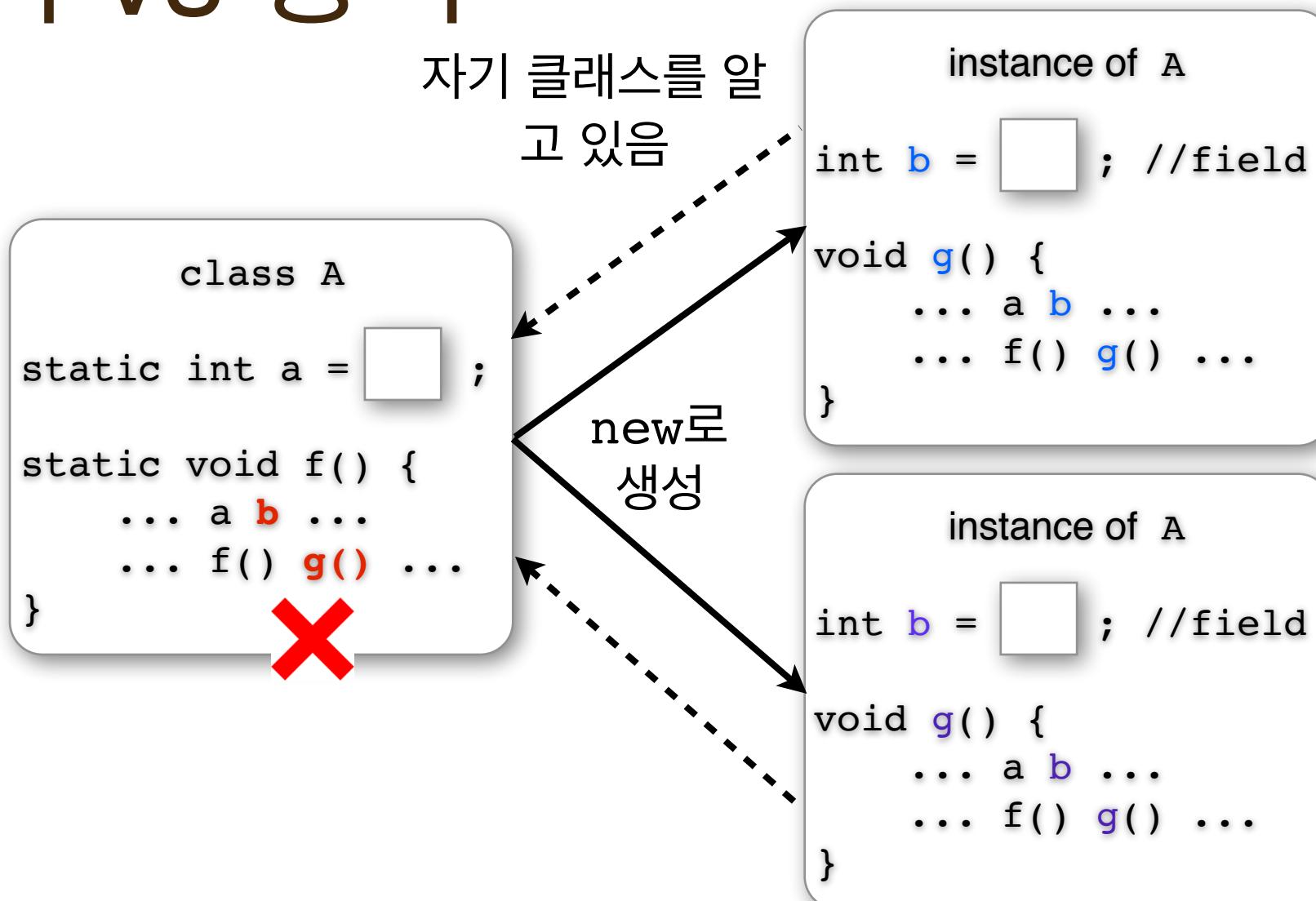
- 클래스에 정의된 메소드
- 객체 생성없이 바로 호출 가능하다.
- 예, Math.abs( f )

# 정적 필드

- 정적 (static): 클래스에 위치, 처음부터 존재
- 동적 (기본): 객체에 위치, new로 생성
- 정적 메소드에서 동적 필드에 접근할 수 없다.



# 정적 vs 동적



# 요약

- 클래스의 정의와 객체의 생성 및 사용
- 메모리 모델, 환경, 변수 유효범위
- Public vs. private
- Getter and Setter