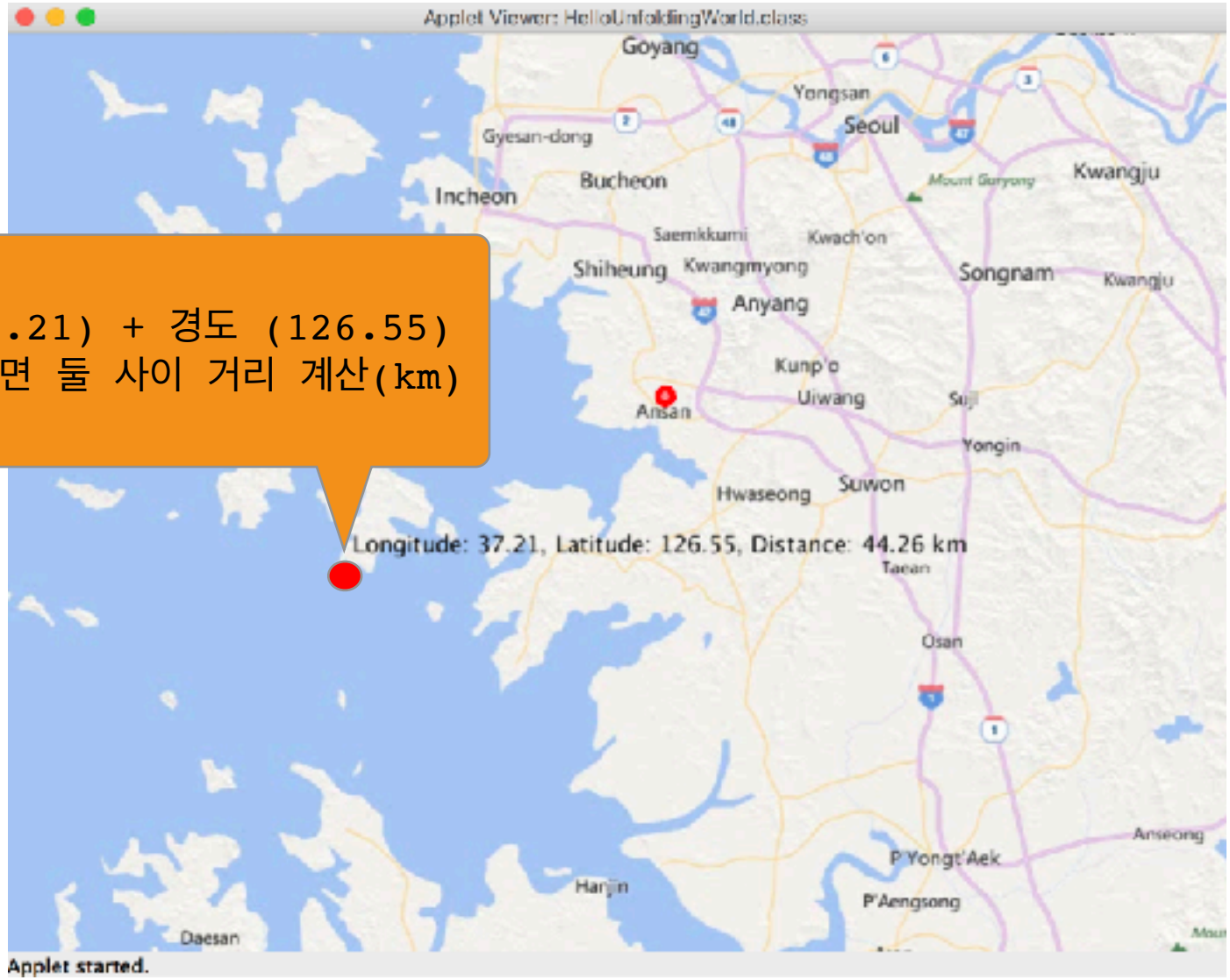
3
부품구조:
클래스와 메소드

간단한 지도 프로그램

위치: 위도 (37.21) + 경도 (126.55)
두 위치가 결정되면 둘 사이 거리 계산(km)



http://unfoldingmaps.org

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 속성 값을 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.

클래스의 정의

- 접근자 class 클래스 이름 { … }
  - 접근지정자 : 외부에서 이 클래스에 접근할 수 있는지 여부 (public: 다른 클래스에서 이 클래스를 이용할 수 있음)
  - 중괄호 안에 클래스의 **필드**와 **메소드**를 모두 작성
- 멤버 (필드)
  - 객체의 상태 값을 저장할 멤버변수.
  - "접근지정자 타입 이름;"
- 메소드
  - 실행가능한 객체의 행위를 구현 (함수)
  - "접근지정자 반환타입 함수이름(매개변수) { … return 값; }"

경우에 따라 생략가능!

클래스 정의

```
/** Location 클래스: 위도 경도 표현 및 다른 위치까지의 거리 계산 */
public class Location
{
  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
{
  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) {
    …
```

필드변수들 (Fields):
객체들이 저장해야 하는 정보들
(위도, 경도)

클래스 정의

```
/** Location 클래스: 위도 경도 표현 및 다른 위치까지의 거리 계산 */
public class Location
{
  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
{
  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(Constructors):
새로운 객체를 만들기 위한 메소드
(타 메소드들과 달리 반환타입 없음)

this: 현재 만들어지고 있는 객체
.(점): 소속을 의미 (예: a.b - a가 소유한 b)

객체 생성과 사용

```
public class Location { …
   public double distance(Location other) {
      // 현재 위치와 다른 위치 other 사이의 거리 계산 후 반환
      // this.long(lat)itude 와 other.long(lat)itude 를 가지고 계산
      return 계산된 거리값;
   }
}
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) {
      // 현재 위치와 다른 위치 other 사이의 거리 계산 후 반환
      // this.long(lat)itude 와 other.long(lat)itude 를 가지고 계산
      return 계산된 거리값;
   }
}
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 메소드를 수행하고 있는 객체

객체, 필드, 메소드

객체 **Object**	메모리에 존재하는 실체	메모리 주소로 식별
필드 **Field**	객체의 상태를 나타내는 정보	필드 변수에 저장하여 내부 공유 및 수정 (객체를 생성할 때 초기값을 정함*)
메소드 **Method**	객체가 수행 가능한 기능	메시지 호출을 받으면 메소드 실행

* 변수의 타입에 따라, 0, false, null로 초기값이 자동으로 매겨짐

객체의 생성과 사용 내부과정

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

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 선언되지 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;
  }
}
```

지역변수(Local variable)는 소속블록(변수를 감싸고 있는 중괄호 {}) 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 변수의 일종.
유효범위:메소드 내부

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

변수의 유효범위 (Variable Scope)

```
public class Scope {

    public static void main(String[] args) {

        {
            int n = 2;
            System.out.println(n);
        }
        double n = 3.14;
        System.out.println(n);
    }

}
```

```
2
3.14
```

같은 이름 중복 선언 오류

```
public class Scope {

    public static void main(String[] args) {

        int n = 2;
        System.out.println(n);
        {
            double n = 3.14;
            System.out.println(n);
        }

    }
}
```

```
Exception in thread "main"
java.lang.Error: Unresolved
compilation problem:
  Duplicate local variable n
```

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;
    }
    …
}
```

필드변수들은 이 클래스의 어떠한 메소드에서도 값이 정의되고 사용될 수 있음.

지역변수 vs 필드변수

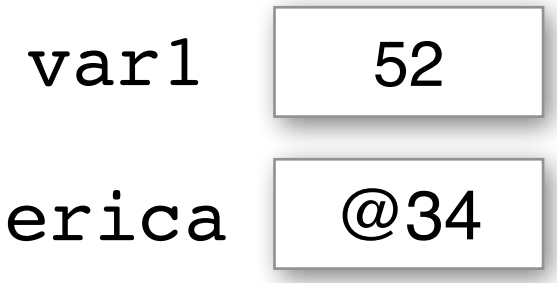
	지역 변수	필드 변수
탄생	선언시	객체 생성시
소멸	메소드 (블록) 실행 종료시	객체 소멸시
개수	메소드 호출 횟수 만큼	객체 생성 개수 만큼
초기화	수동	자동
유효 범위 scope	선언 이후부터 소속 블록 끝까지	객체 내부 전체 + 한정자(public 등)에 따라 객체 외부 접근 가능

메모리 모델

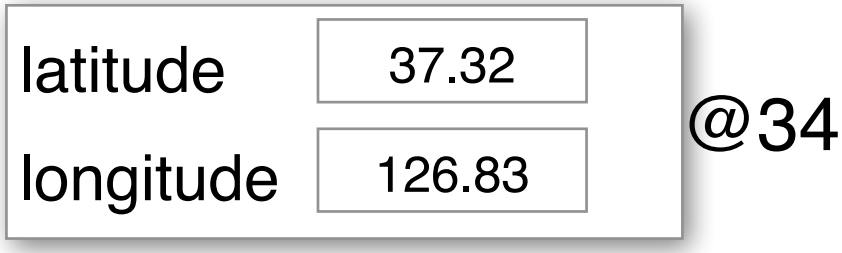
```
int var1 = 52;
Location erica = new Location(37.32, 126,83);
Location seoul = new Location(37.55, 127.04);
seoul.longitude = 127.05;
```

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;
```

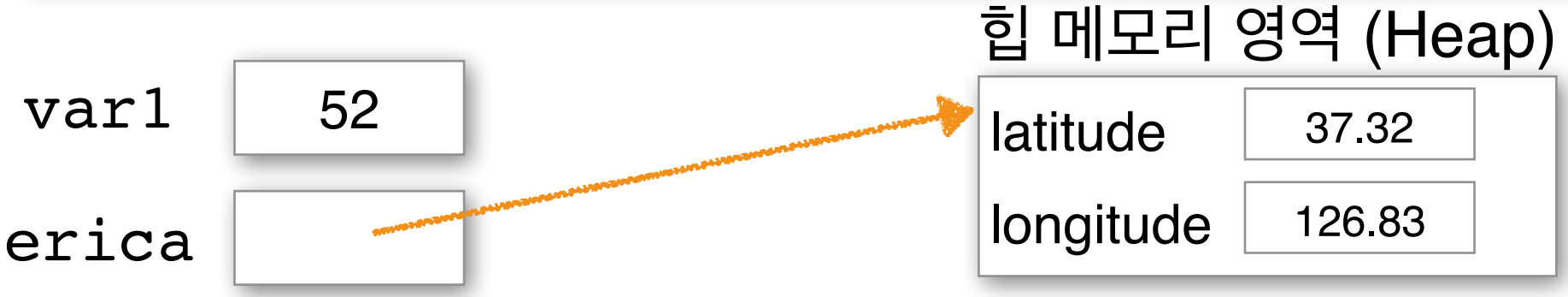


힙 메모리 영역 (Heap)



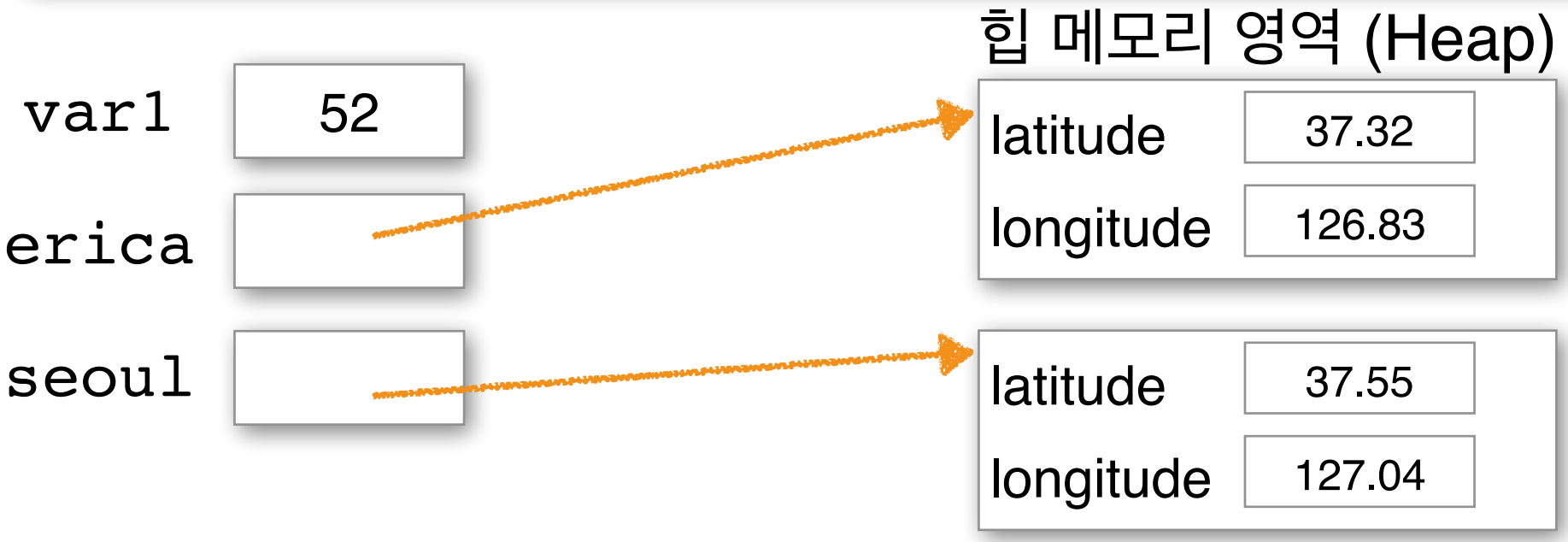
메모리 모델

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



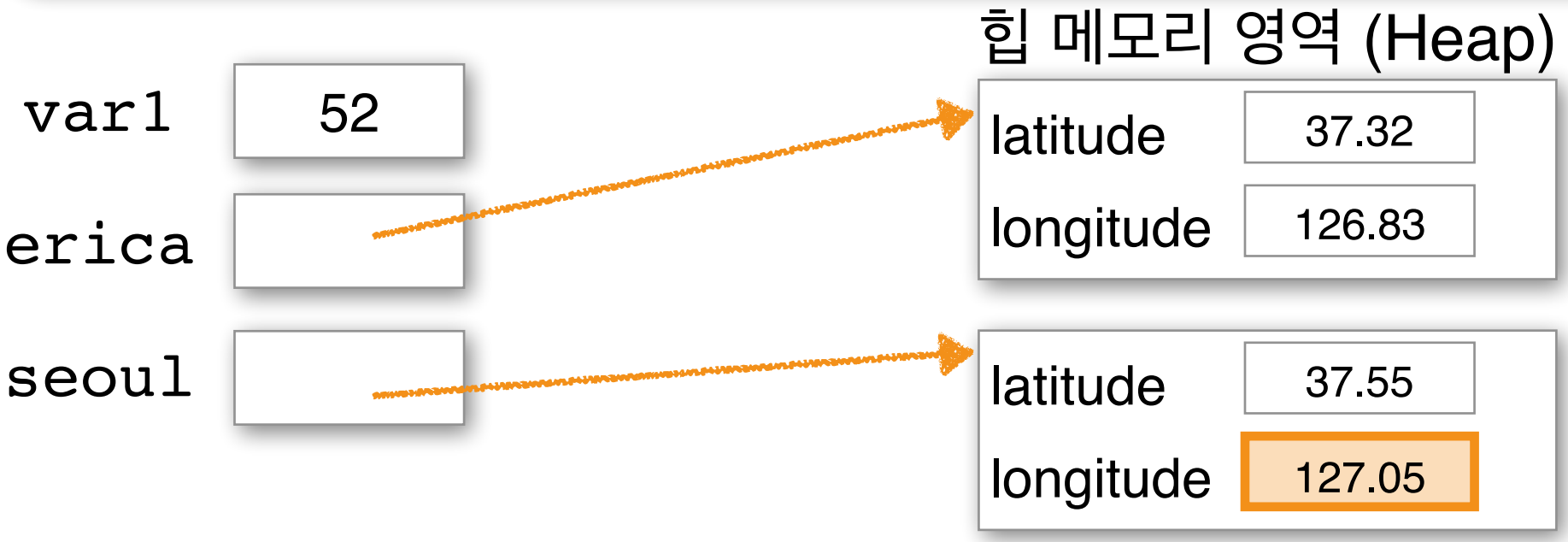
메모리 모델

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



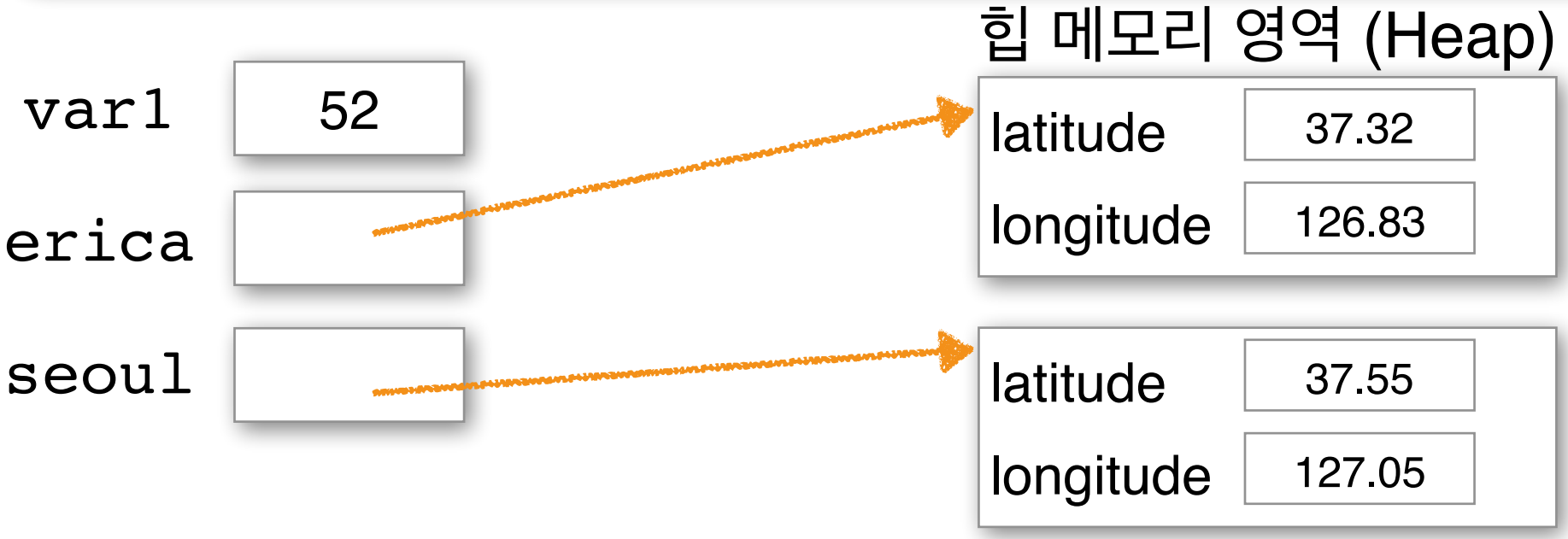
메모리 모델

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

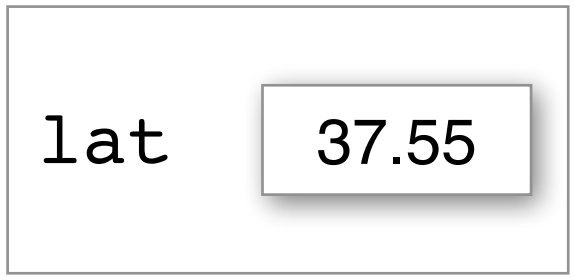


메모리 모델

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

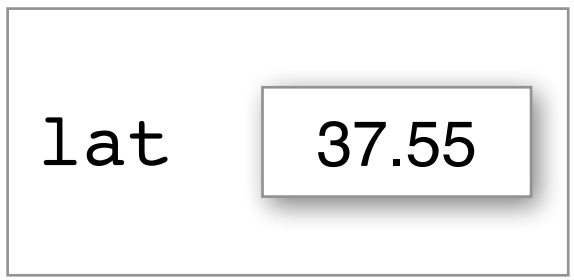


```
public class LocationTester {
  public static void main(String[] args) {
    double lat = 37.55;
    Location seoul = new Location(37.55, 127.04);
  }
}
```



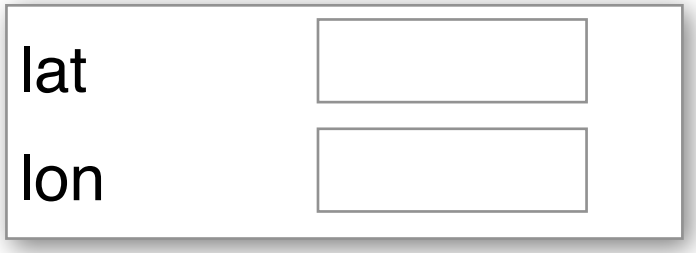
main의 환경

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

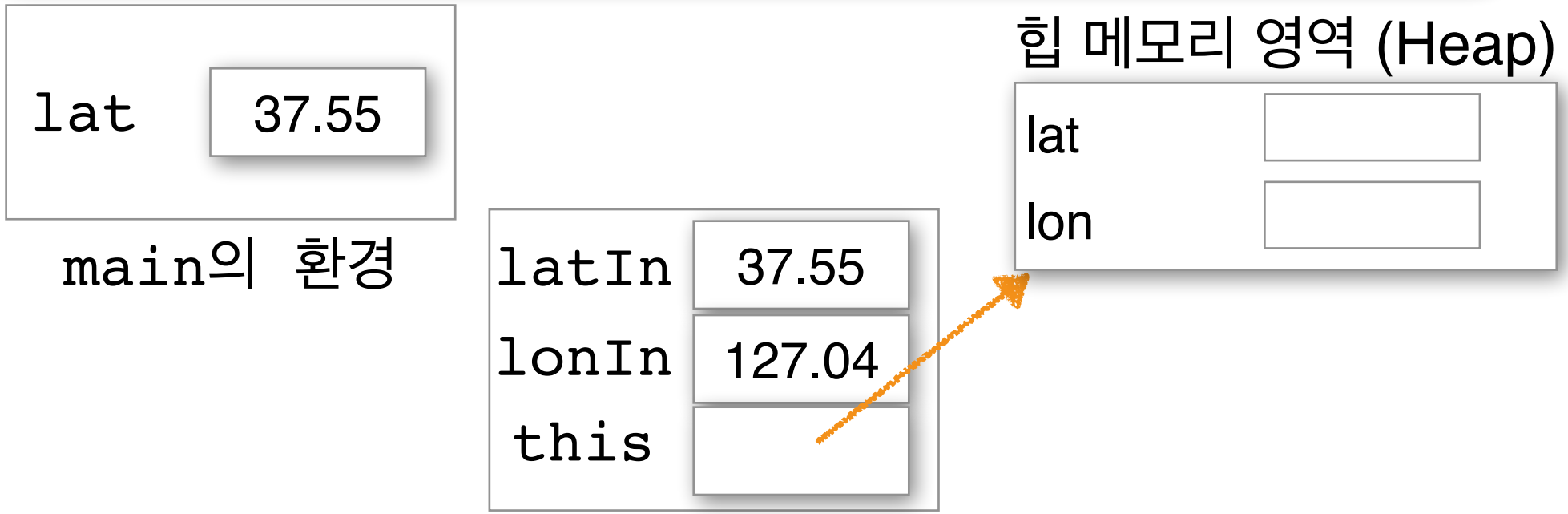


main의 환경

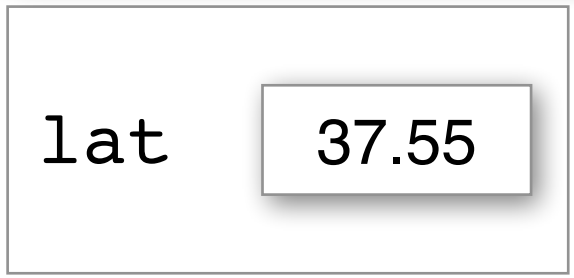
힙 메모리 영역 (Heap)



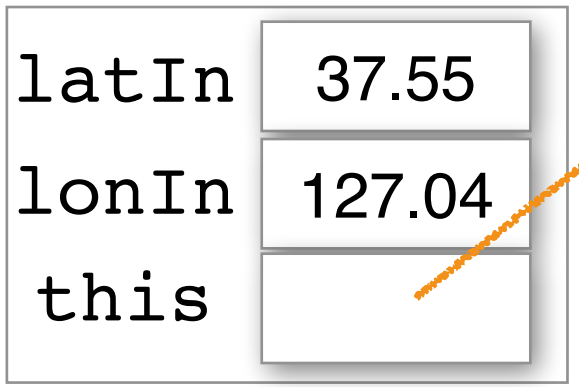
```
public class Location {
   public double lat;
   public double lon;
   public Location(double latIn, double lonIn) {
         this.lat = latIn;
         this.lon = lonIn;
   }
}
```



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

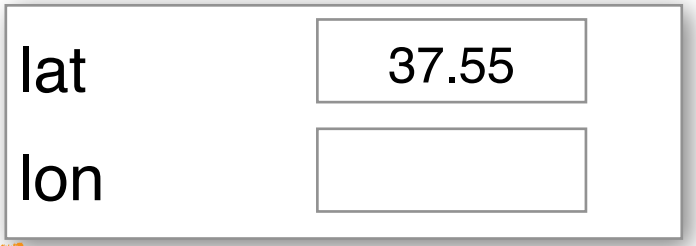


main의 환경

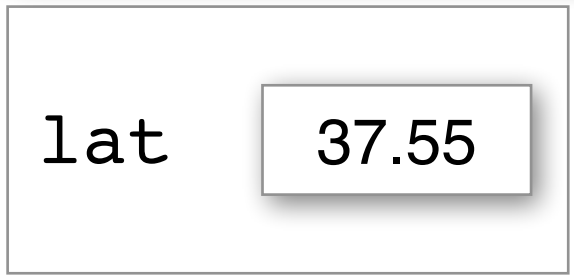


생성자의 환경

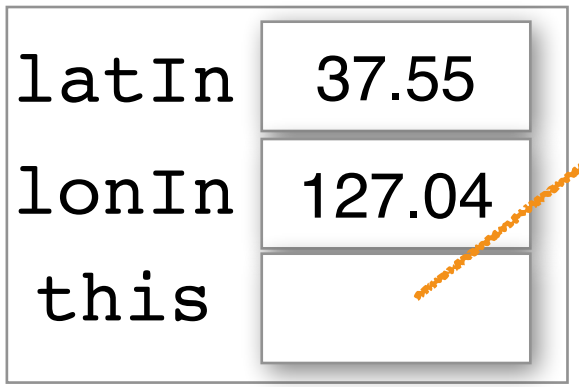
힙 메모리 영역 (Heap)



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

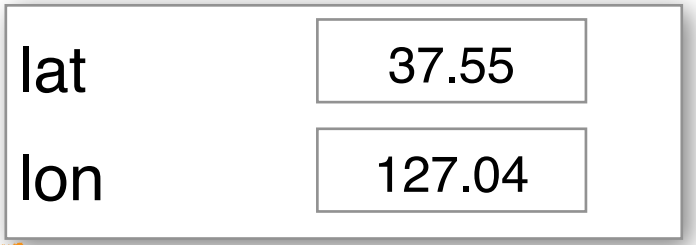


main의 환경



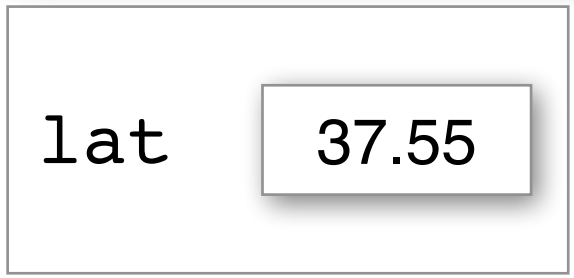
생성자의 환경

힙 메모리 영역 (Heap)

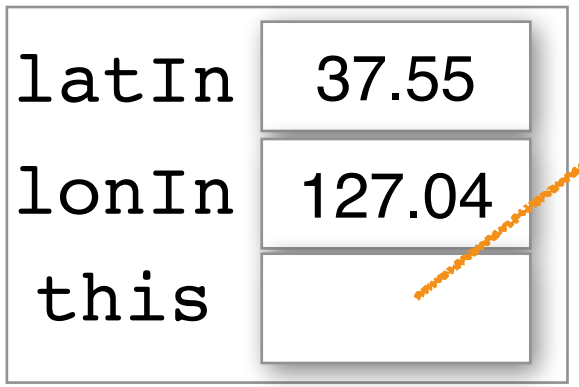


```
public class Location {
   public double lat;
   public double lon;
   public Location(double latIn, double lonIn) {
        this.lat = latIn;
        this.lon = lonIn;
        return this;
   }
```

생성자 메소드는 항상 **this** 를 반환

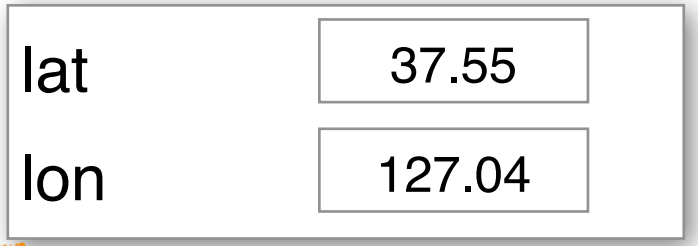


main의 환경

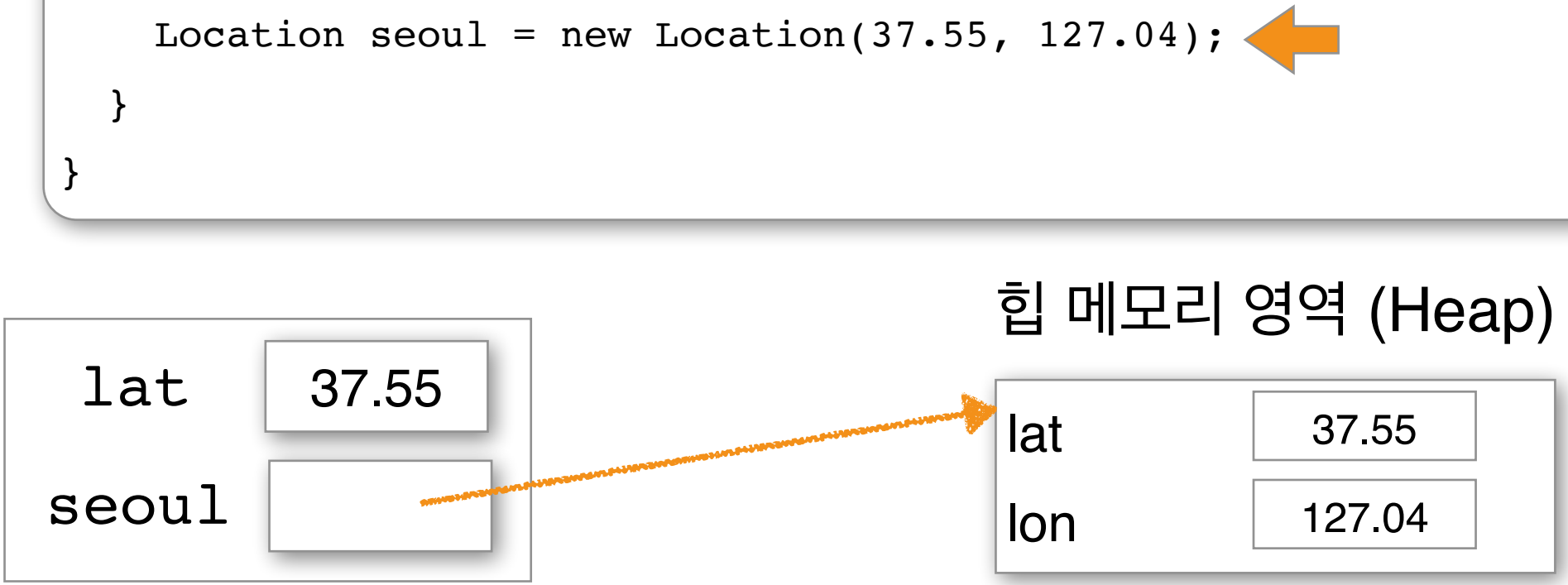


생성자의 환경

힙 메모리 영역 (Heap)

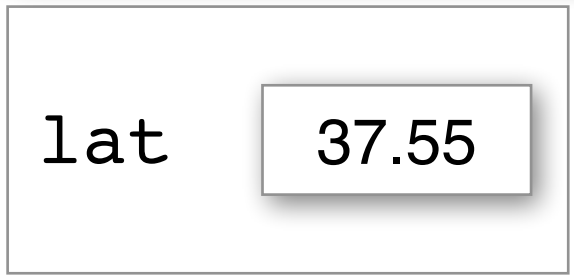


```
public class LocationTester {
  public static void main(String[] args) {
    double lat = 37.55;
    Location seoul = new Location(37.55, 127.04);
  }
}
```

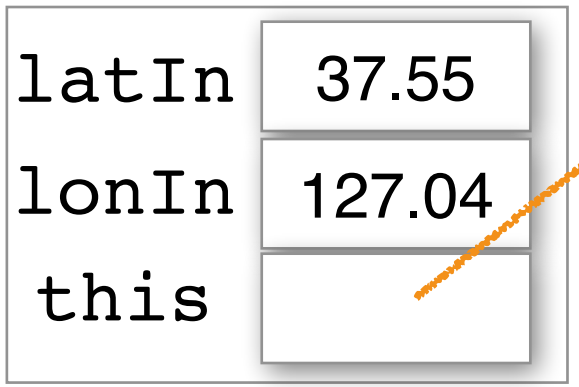


```
public class Location {
   public double lat;
   public double lon;
   public Location(double latIn, double lonIn) {
        this.lat = latIn;
        this.lon = lonIn;
   }
}
```

this 생략 가능

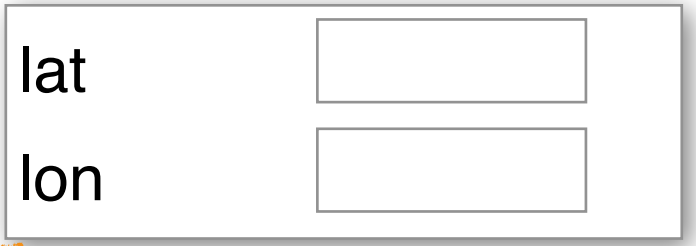


main의 환경



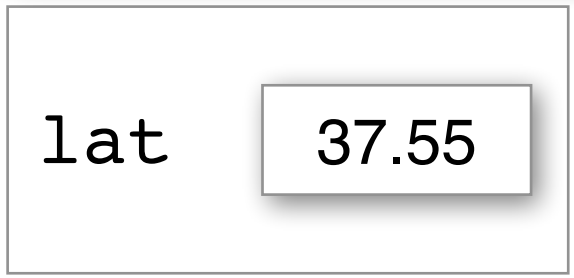
생성자의 환경

힙 메모리 영역 (Heap)

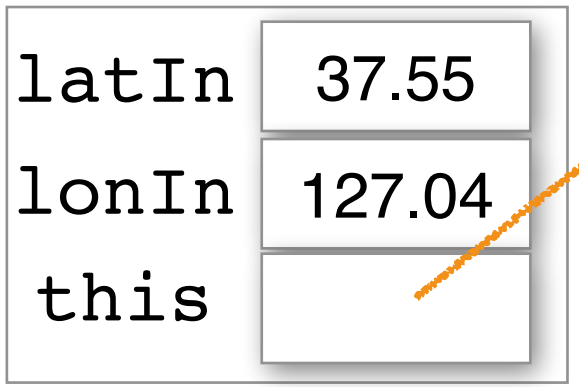


```
public class Location {
   public double lat;
   public double lon;
   public Location(double latIn, double lonIn) {
         lat = latIn;
         lon = lonIn;
   }
}
```

생성자 환경에서 변수 lat 찾음 → 없음!

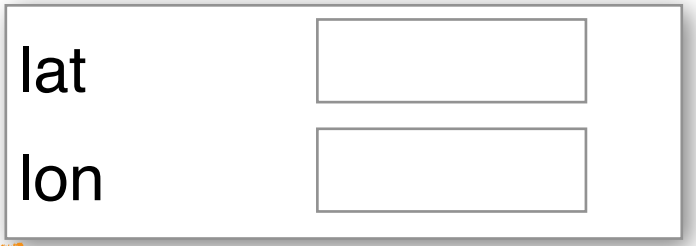


main의 환경



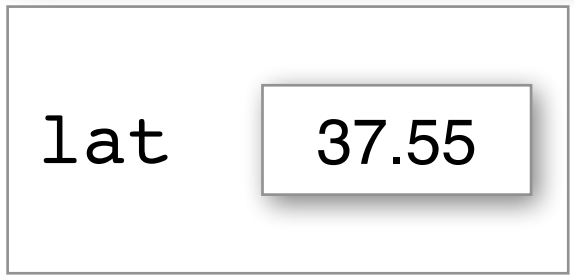
생성자의 환경

힙 메모리 영역 (Heap)

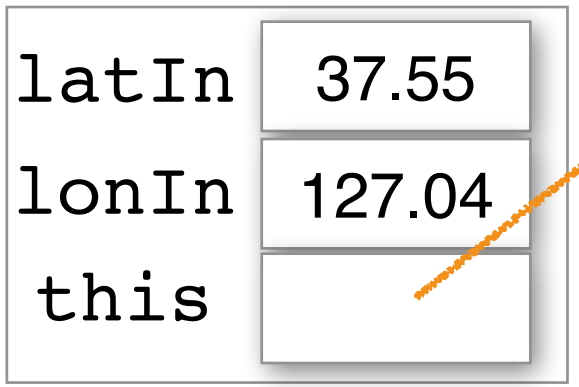


```
public class Location {
   public double lat;
   public double lon;
   public Location(double latIn, double lonIn) {
         lat = latIn;
         lon = lonIn;
   }
}
```

객체의 환경 (this가 가리키는) 에서 lat 찾음

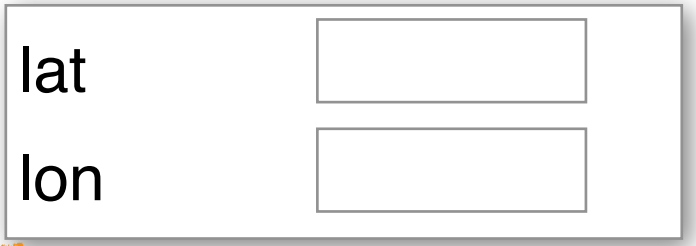


main의 환경



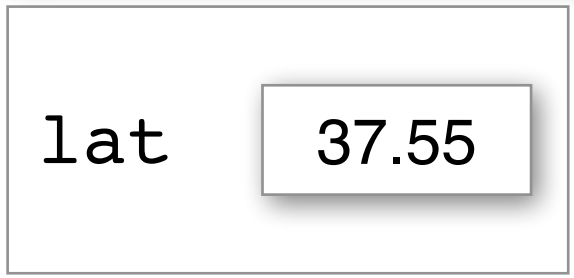
생성자의 환경

힙 메모리 영역 (Heap)

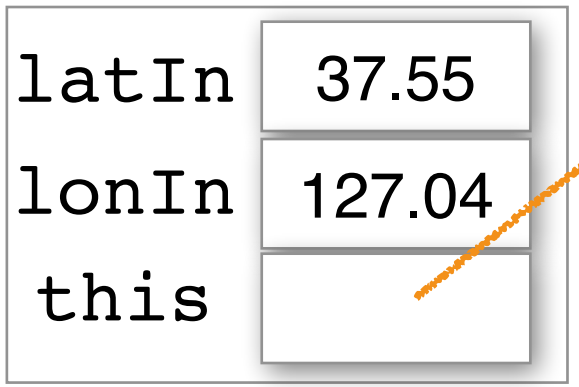


```
public class Location {
   public double lat;
   public double lon;
   public Location(double latIn, double lonIn) {
         lat = latIn;
         lon = lonIn;
   }
}
```

객체의 환경 (this가 가리키는) 에서 lat 찾음

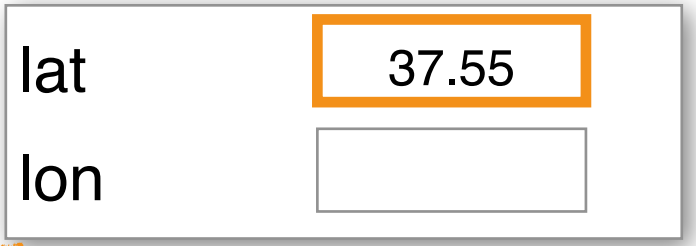


main의 환경



생성자의 환경

힙 메모리 영역 (Heap)



메모리 모델 심화

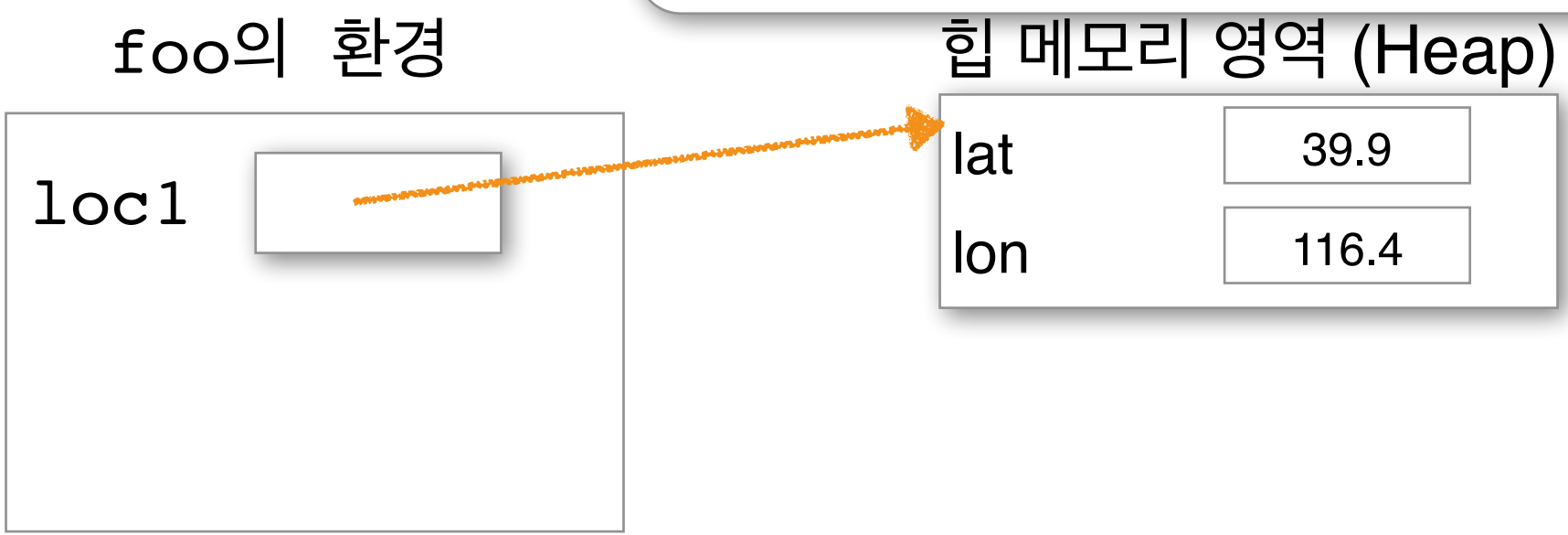
```
public class Loc {
    public double lat;
    public double lon;
    public Loc(double latIn,
               double lonIn)
        this.lat = latIn;
        this.lon = lonIn;
    }
    …
}
```

```
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?

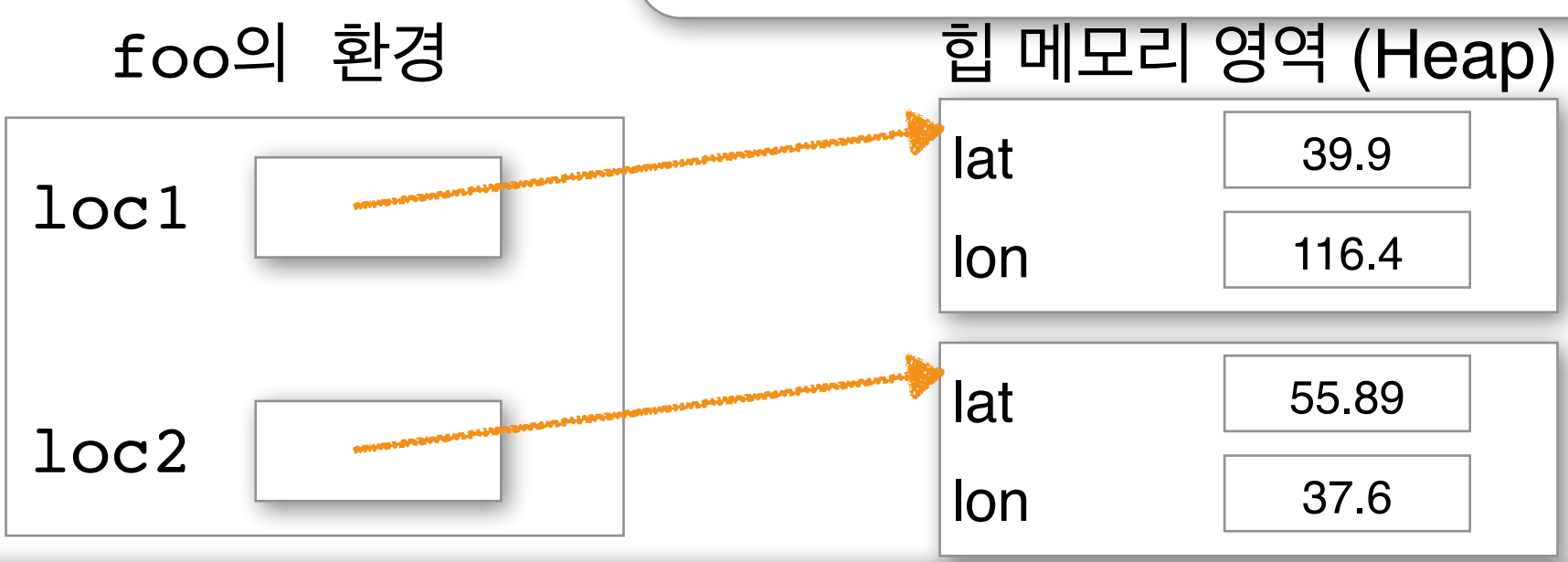
```
public class Loc {
    public double lat;
    public double lon;
    public Loc(double latIn,
               double lonIn)
        this.lat = latIn;
        this.lon = lonIn;
    }
    …
}
```

```
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());
 }
}
```



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

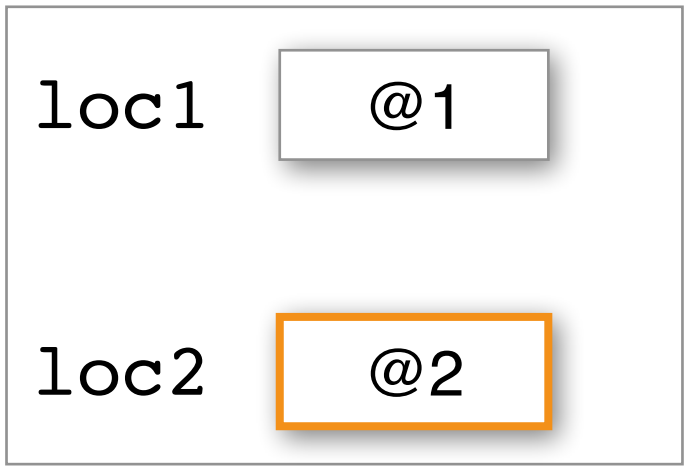
```
public class LocationTester {
 public static String foo () {
   Loc loc1 = new Loc(39.9,116.4);
   Loc loc2 = new Loc(55.8,37.6);
   loc1 = loc2;
   loc1.lat = -8.3;
   return (loc2.lat + ", " + loc2.lon);
 }
 public static void main(String[] args) {
   System.out.println(foo());
 }
}
```



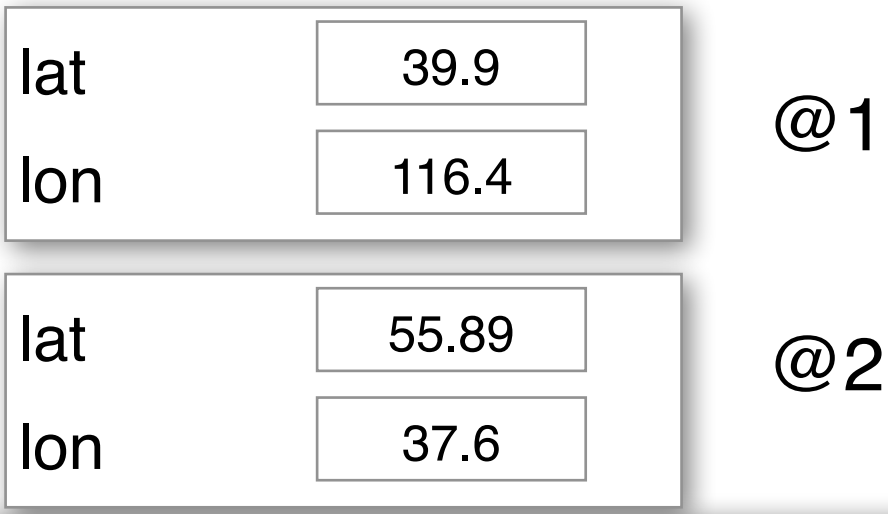
```
public class Loc {
    public double lat;
    public double lon;
    public Loc(double latIn,
               double lonIn)
        this.lat = latIn;
        this.lon = lonIn;
    }
    …
}
```

```
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

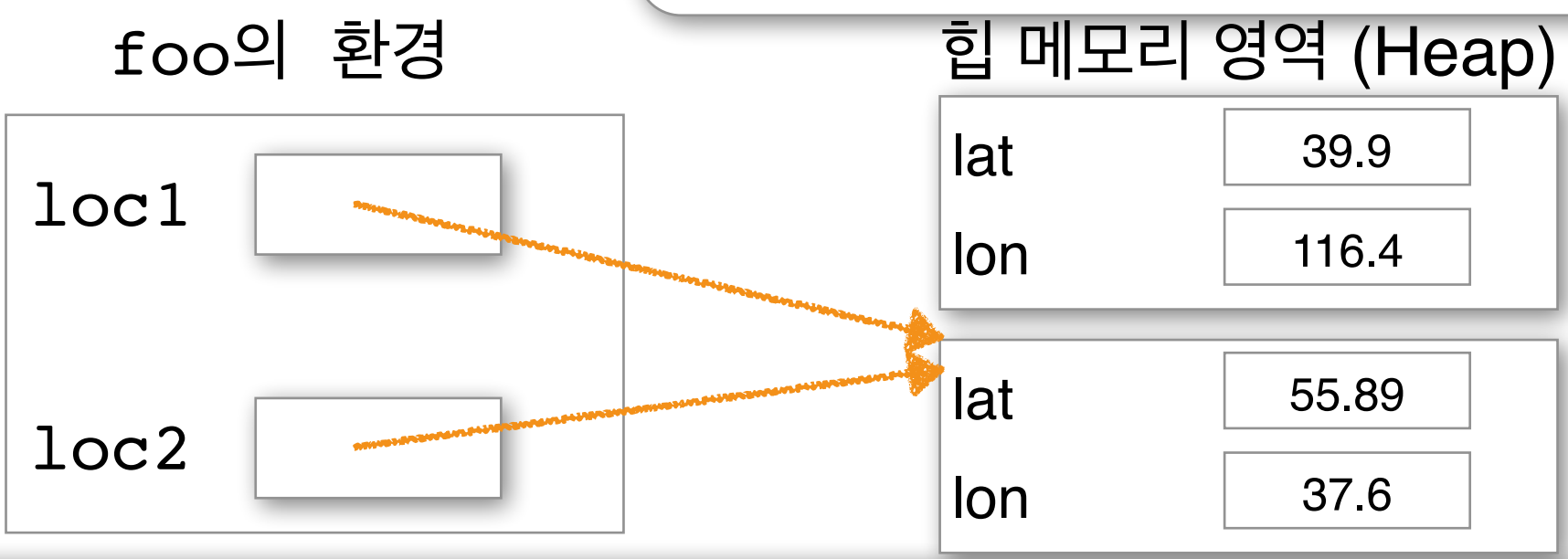


힙 메모리 영역 (Heap)



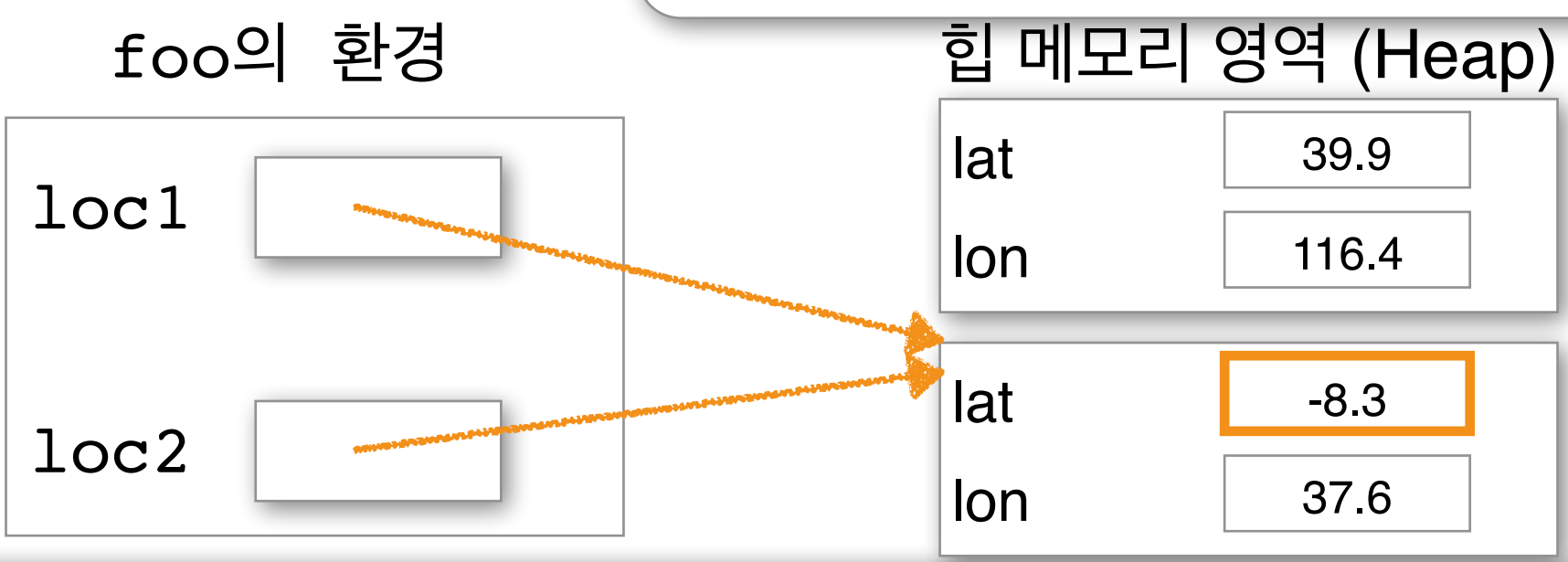
```
public class Loc {
    public double lat;
    public double lon;
    public Loc(double latIn,
               double lonIn)
        this.lat = latIn;
        this.lon = lonIn;
    }
    …
}
```

```
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());
 }
}
```



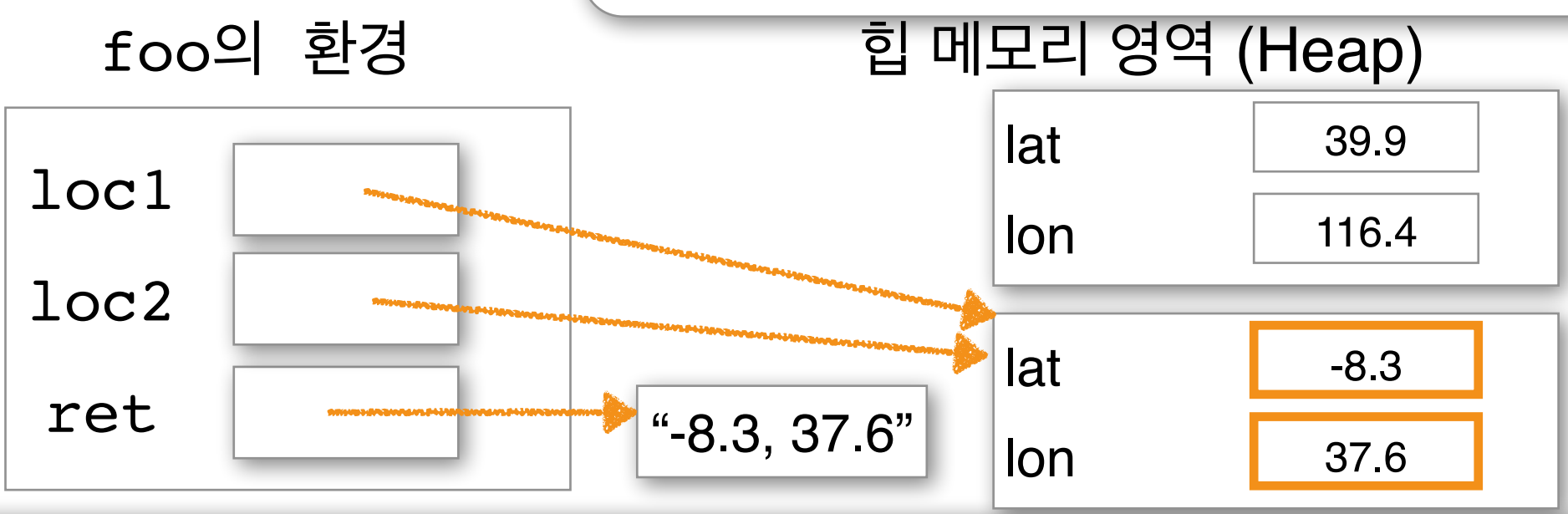
```
public class Loc {
    public double lat;
    public double lon;
    public Loc(double latIn,
               double lonIn)
        this.lat = latIn;
        this.lon = lonIn;
    }
    …
}
```

```
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());
 }
}
```



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

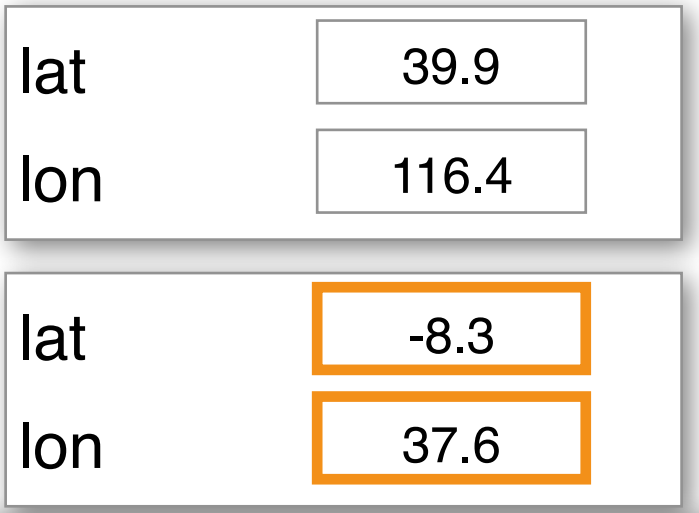
```
public class LocationTester {
 public static String foo () {
   Loc loc1 = new Loc(39.9,116.4);
   Loc loc2 = new Loc(55.8,37.6);
   loc1 = loc2;
   loc1.lat = -8.3;
   return (loc2.lat + ", " + loc2.lon);
 }
 public static void main(String[] args) {
   System.out.println(foo());
 }
}
```



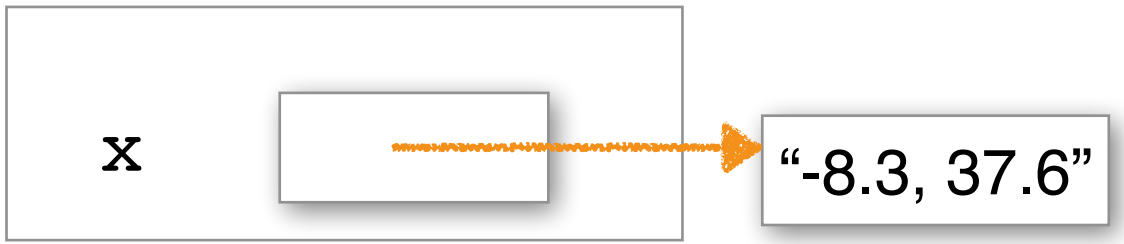
```
System.out
 println(String x) {
 …
 }
```

```
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

Public vs. Private

- 현재 문제점: 지도 프로그램에서 에리카 캠퍼스의 위도를 100.0 으로 설정가능 → 가능하지 않아야!
  (왜냐하면 |유효한 위도| <= 90)

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

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 …
   }
}
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));
  }
}
```

모두 허용됨 (public이므로)

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;
    System.out.println (erica.distance(erica));
  }
```

허용 안됨!

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 double getLatitude() {
    return latitude;
  }
```

- **Setter 때문에 필드값이 private이 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`
- 필드, 지역변수: `longitude, latitude`
  - 상수처럼 쓰일때는 `RADIUS_EARTH`

정적 (Static) 필드 변수 및 메소드

- 클래스에 정의된 필드 변수 및 메소드
- 객체 생성없이 바로 사용 가능하다.
  - 예, `Math.PI, Math.abs(f)`
- 필드 변수 및 메소드 정의 시 타입 앞에 `static` 을 붙이면 정의 가능
  - 예, `public static double PI = 3.14;`

요약

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