Java Programming II

Lab3

514770-1
Fall 2025
9/24/2025

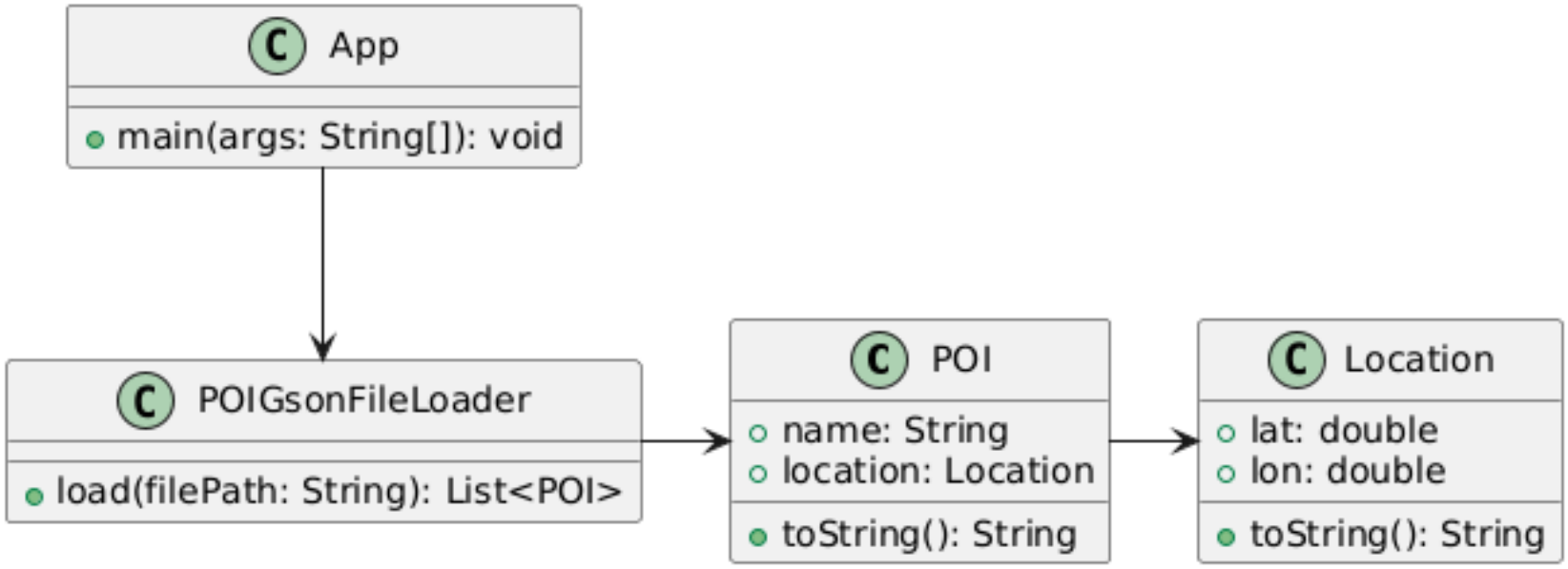
Kyoung Shin Park
Computer Engineering
Dankook University

Lab3

- 사용자 Location (위치 변화 통지)에 따라 사용자가 POI에 Proximity (근접 이벤트 통지) 했는지를 보여주는 이중 Observer 패턴을 구현하라.
  - **LocationManager**는 **LocationSubject**로서 위치 변화를 **updateLocation** 즉 의미 있는 변화(스로틀 + 이동 임계)인 경우에만 **LocationObserver**들에게 통지
  - **ProximityManager**는 **LocationObserver**로서 위치 변화를 수신하면, 거리 정렬을 통해 Top-K 근접 목록 변화 **onNearestChanged**로, 근접 반경 진입을 **onEnter**로, 근접 반경 이탈을 **onExit**로 **ProximityObserver**들에게 통지
  - **POIDetailView**와 **ProximityLogger**는 **ProximityObserver**로서 근접 이벤트를 통해

Lab3

- **App**은 POIGsonFileLoader를 이용하여 POI.json데이터를 로딩하여 List<POI>로 저장한다.



Lab3

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

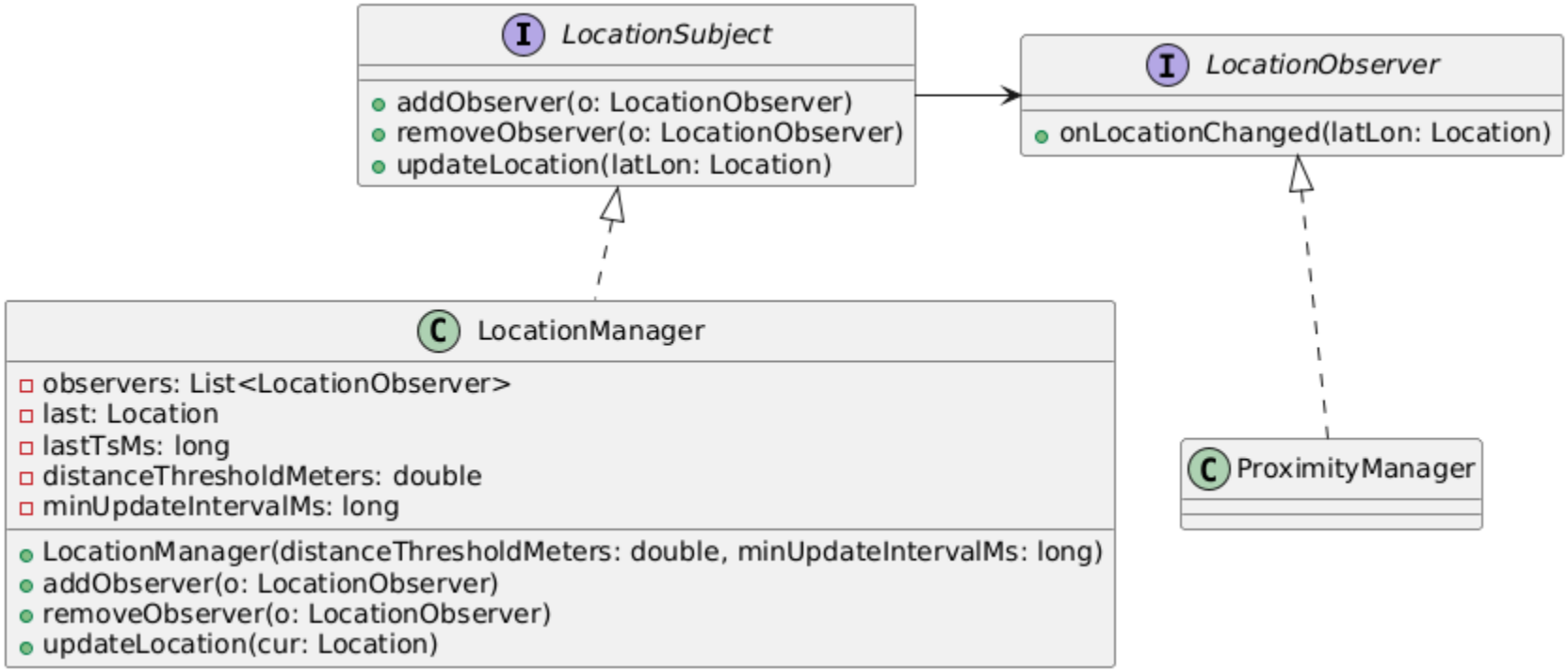
```
public class  POI {
    public final String name;
    public final Location location;
    public POI(String name, Location location) {
        this.name = name;
        this.location = location;
    }…
}
```

Lab3

```
public final class GeoUtil {
  public static double distanceMeters(Location a,
Location b) {// Haversine distance (m)
    final double R = 6371000.0;
    double dLat = Math.toRadians(b.lat - a.lat);
    double dLon = Math.toRadians(b.lon - a.lon);
    double la1 = Math.toRadians(a.lat), la2 =
Math.toRadians(b.lat);
    double h = Math.sin(dLat/2)*Math.sin(dLat/2) +
Math.cos(la1)*Math.cos(la2)*Math.sin(dLon/2)*Math.si
n(dLon/2);
    double c = 2 * Math.atan2(Math.sqrt(h),
Math.sqrt(1-h));
    return R * c;
  }
}
```

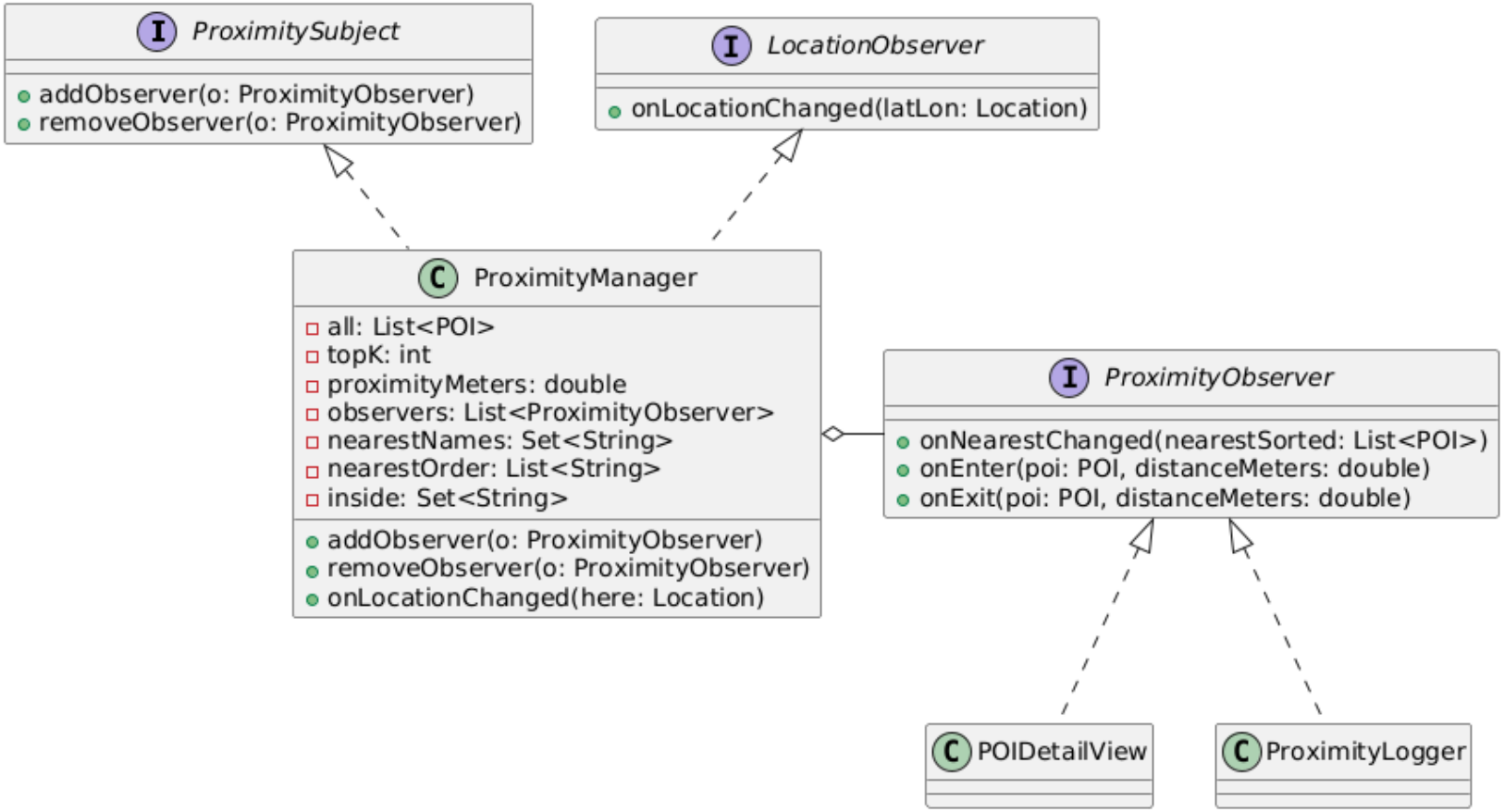
Lab3

- **LocationManager**는 ProximityManager 옵저버를 추가하고, 위치 변경에 따른 updateLocation 를 호출한다.
  - **updateLocation()** 이전과 현재 위치 이동 거리와 업데이트 시간이 의미있으면 (GeoUtil사용) 위치 업데이트 알림 전달



Lab3

- **ProximityManager**는 POIDetailView와 ProximityLogger 옵저버를 추가하고, onLocationChanged에서 근접 여부에 따라 onNearestChanged, onEnter, onExit을 호출한다.



Lab3

```
public class ProximityManager implements
LocationObserver, ProximitySubject {
    public ProximityManager(List<POI> all, int topK,
double proximityMeters) {
        this.all = Objects.requireNonNull(all);
        this.topK = Math.max(1, topK);
        this.proximityMeters = Math.max(0.0,
proximityMeters);
    } …
    @Override
    public void onLocationChanged(Location here) {
      // 거리 계산(GeoUtil사용) 및 정렬
      // 가장 근접한 Top-K 목록 작성
      // 변화가 있으면 onNearestChanged 통지
      // 근접 반경 enter/exit 통지
    }
}
```

Lab3

```
public class ProximityLogger implements
ProximityObserver {
    @Override
    public void onNearestChanged(List<POI>
nearestSorted) {
        System.out.println("[LOG] nearestChanged ->
" + nearestSorted);
    }
    @Override
    public void onEnter(POI poi, double d) {
        System.out.printf("[LOG] enter %s @ %.2fm%n",
poi.name, d);
    }
    @Override
    public void onExit(POI poi, double d) {
        System.out.printf("[LOG] exit %s @ %.2fm%n",
poi.name, d);
    }
}
```

Lab3

```
public static void main(String[] args) throws Exception {
  List<POI> pois = ..
  // Subject
  LocationManager lm = new LocationManager(5.0, 500);
  // Subject&Observer
  ProximityManager pm = new ProximityManager(pois, 2, 100.0);
  // Observer
  POIDetailView ui = new POIDetailView();
  ProximityLogger log = new ProximityLogger();
  pm.addObserver(ui);
  pm.addObserver(log);
  // when location update..
  Location[] path = …
  for (Location g : path) {
      lm.updateLocation(g);
      Thread.sleep(600); // minUpdateIntervalsMs보다 길게
  }
```

```
[UI] Top-K nearest updated:
  #1 경복궁 근정문 및 행각 @ (37.57814, 126.9767)
  #2 경복궁 @ (37.5772775, 126.97691)
[LOG] nearestChanged -> [경복궁 근정문 및 행각 @ (37.57814, 126.9767), 경복궁 @ (37.5772775,
126.97691)]
[UI] ENTER range: 경복궁 근정문 및 행각 (67.2m)
[LOG] enter 경복궁 근정문 및 행각 @ 67.21m
[UI] Top-K nearest updated:
  #1 경복궁 근정문 및 행각 @ (37.57814, 126.9767)
  #2 경복궁 사정전 @ (37.57918, 126.977)
[LOG] nearestChanged -> [경복궁 근정문 및 행각 @ (37.57814, 126.9767), 경복궁 사정전 @
(37.57918, 126.977)]
[UI] ENTER range: 경복궁 사정전 (93.6m)
[LOG] enter 경복궁 사정전 @ 93.62m
[UI] ENTER range: 경복궁 수정전 (98.7m)
[LOG] enter 경복궁 수정전 @ 98.75m
[UI] Top-K nearest updated:
  #1 경복궁 사정전 @ (37.57918, 126.977)
  #2 경복궁 근정문 및 행각 @ (37.57814, 126.9767)
[LOG] nearestChanged -> [경복궁 사정전 @ (37.57918, 126.977), 경복궁 근정문 및 행각 @
(37.57814, 126.9767)]
[UI] EXIT range: 경복궁 수정전 (109.3m)
[LOG] exit 경복궁 수정전 @ 109.34m
[UI] Top-K nearest updated:
  #1 경복궁 사정전 @ (37.57918, 126.977)
  #2 경복궁 향원정 @ (37.5806903, 126.9769632)
[LOG] nearestChanged -> [경복궁 사정전 @ (37.57918, 126.977), 경복궁 향원정 @ (37.5806903,
126.9769632)]
[UI] EXIT range: 경복궁 근정문 및 행각 (232.2m)
[LOG] exit 경복궁 근정문 및 행각 @ 232.23m
[UI] EXIT range: 경복궁 사정전 (154.0m)
[LOG] exit 경복궁 사정전 @ 153.98m
[UI] Top-K nearest updated:
  #1 경복궁 향원정 @ (37.5806903, 126.9769632)
  #2 경복궁 근정전 @ (37.5806903, 126.9769632)
```

Submit to e-learning

- 메인에 Observer 추가 삭제 루틴 포함할 것
- Lab3 과제에 yourcode (e.g.: 다른 LocationObserver 추가 등)를 추가 (yourcode 없을시 10점에서 -1점 감점)
- **Java25-2-HW3-YourID-YourName.zip** 과제(보고서에 반드시 yourcode 설명 포함)를 e러닝에 제출 **(due by 9/30)**.