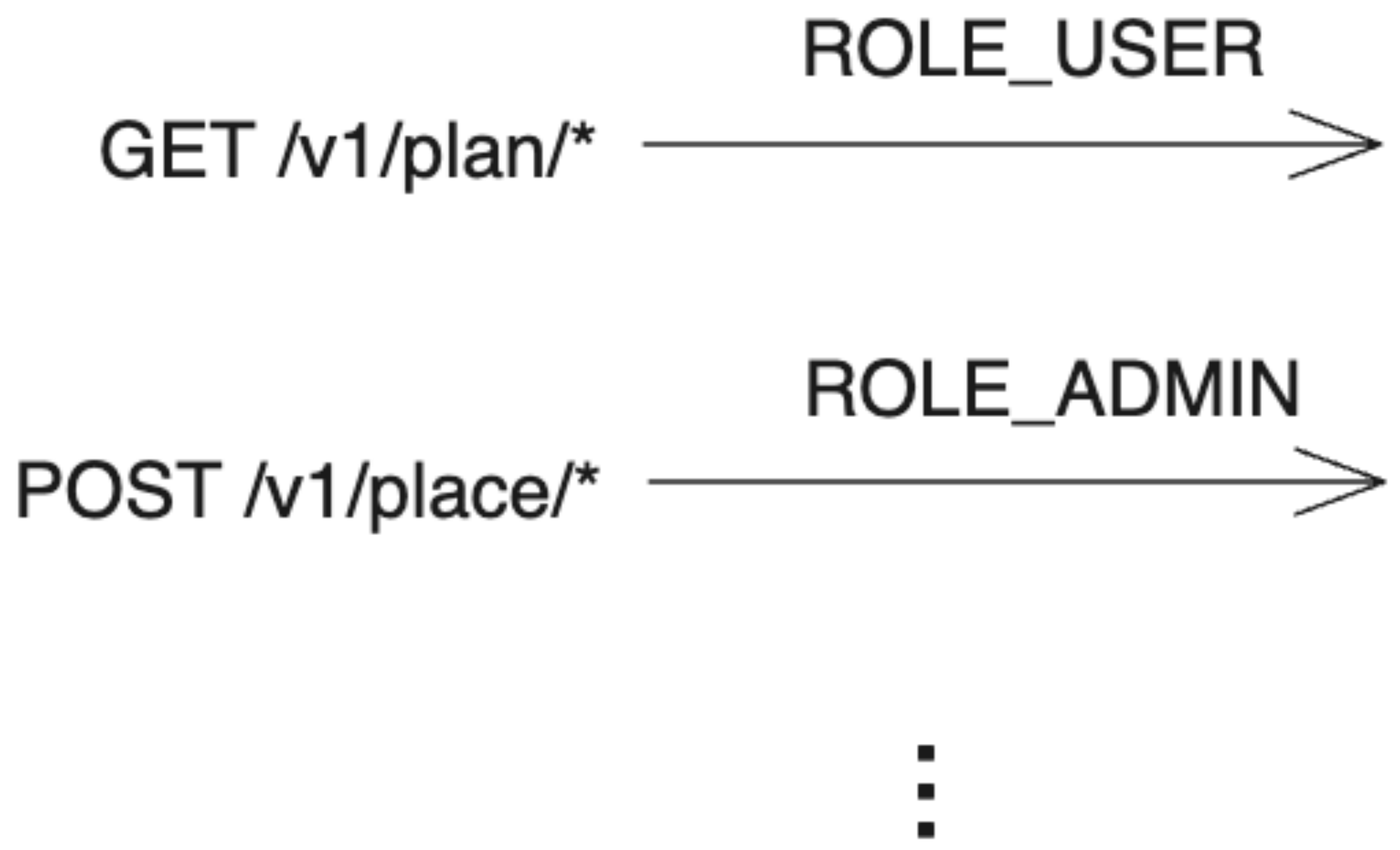
여행 프로젝트 진행기(2)

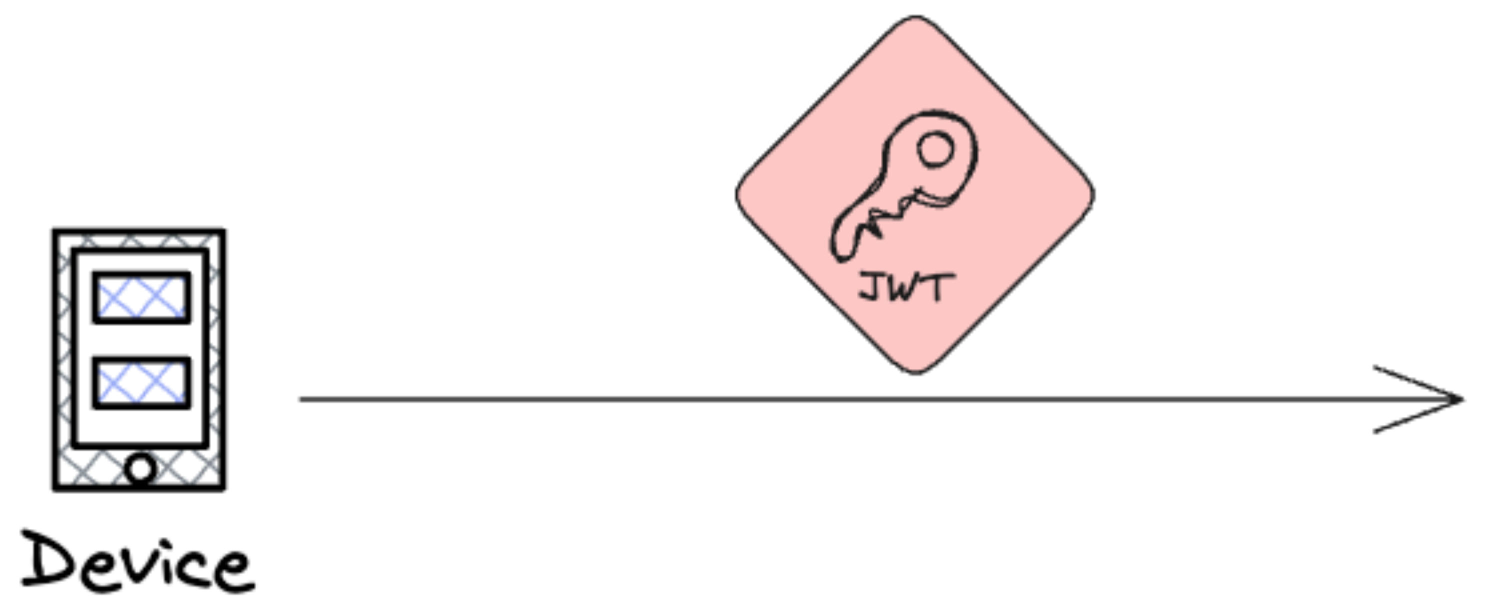
1/17~1/23

리소스 기반 접근 제어 구현

기존 방식



- ROLE이라는게 너무 큰 단위
- Account 서비스에서 모든 서비스의 접근 포인트를 알고 있어야함
- 리소스 소유권 기반(소유자, 동참자) 처리 불가능
- 너무 많은 역할 부여시 패킷 사이즈 증가



계층적 권한 부여 시스템 개발

ROLE

- ROLE_USER
- ROLE_ANNONYMOUS
- ROLE_STAFF
- ROLE_ADMIN

Account Service가 부여

AUTHORITY(PERMISSION)

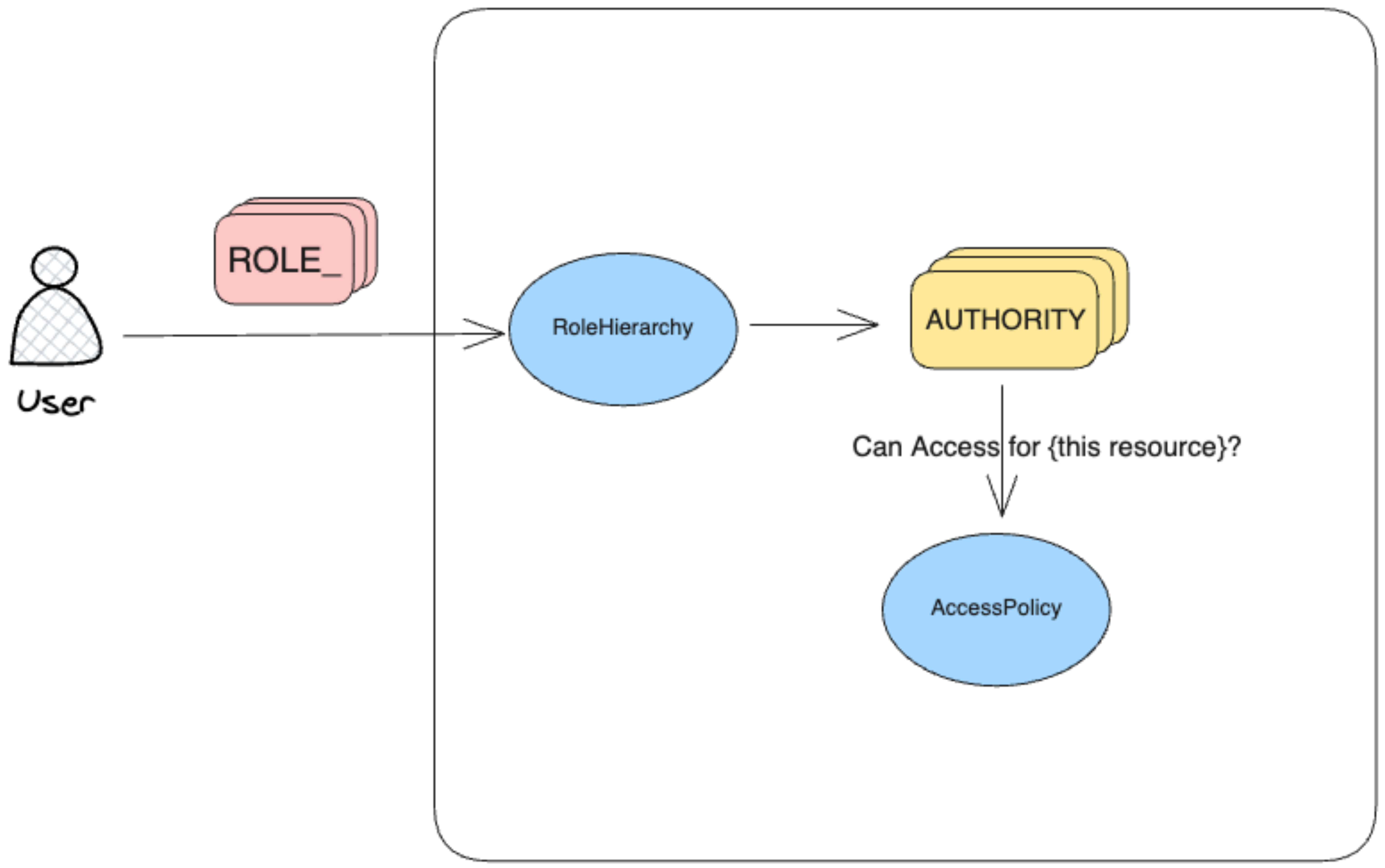
- plan:read:owned
- place:create:other
- plan:update:all
- plan:create

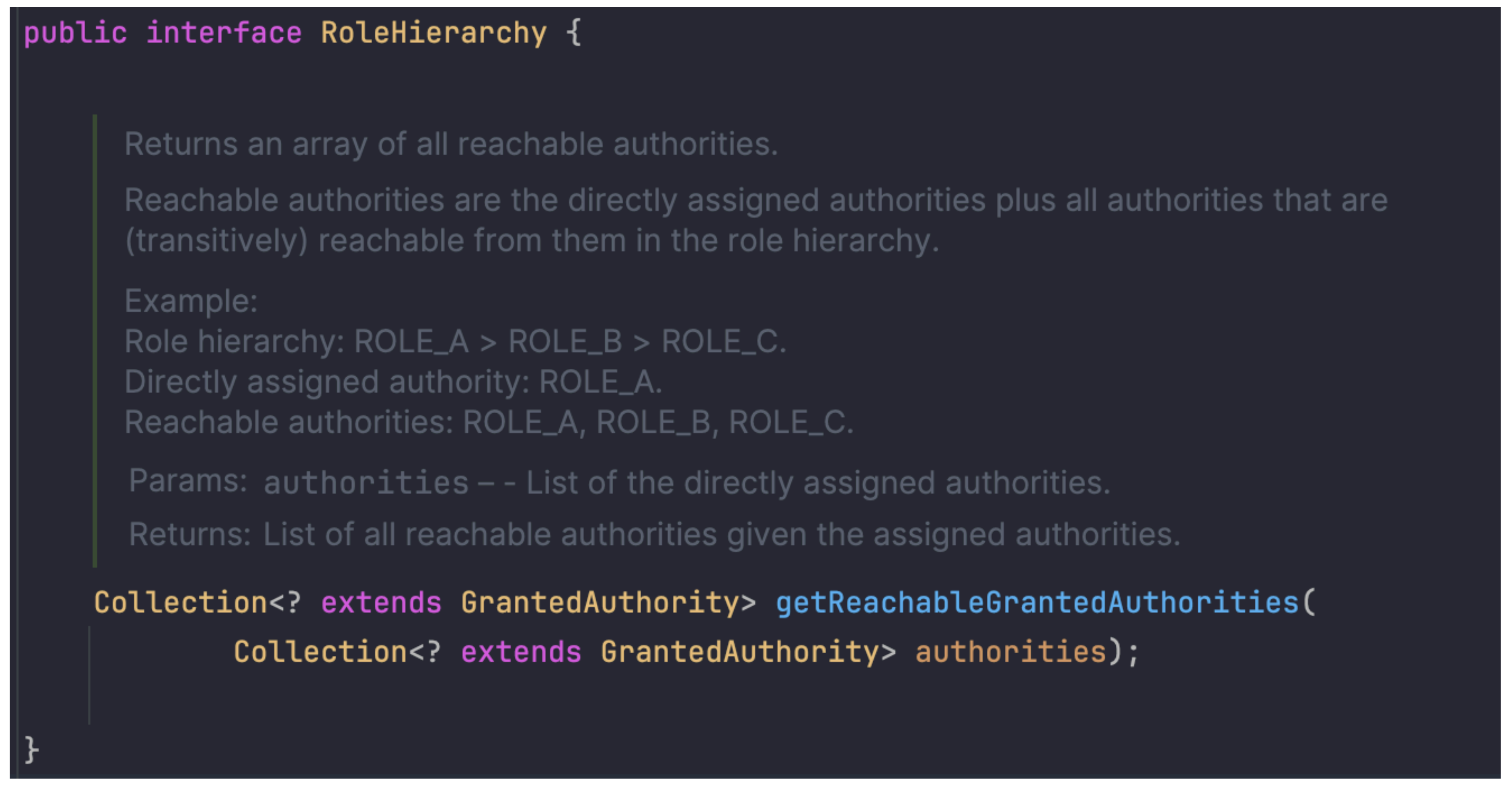
각 마이크로 서비스가 부여

{domain}:{action}:{scope}

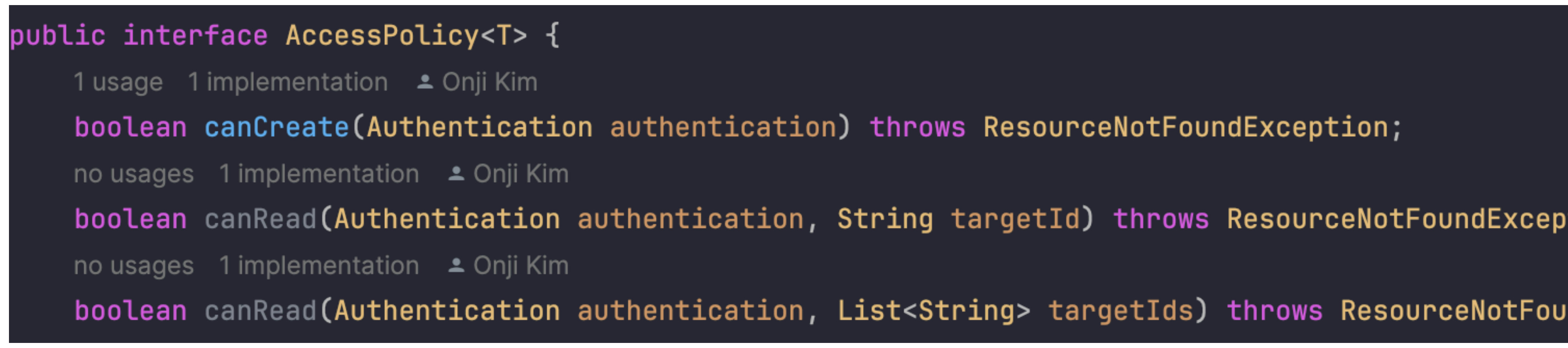
{domain}:{action}

계층적 권한 부여 시스템 개발



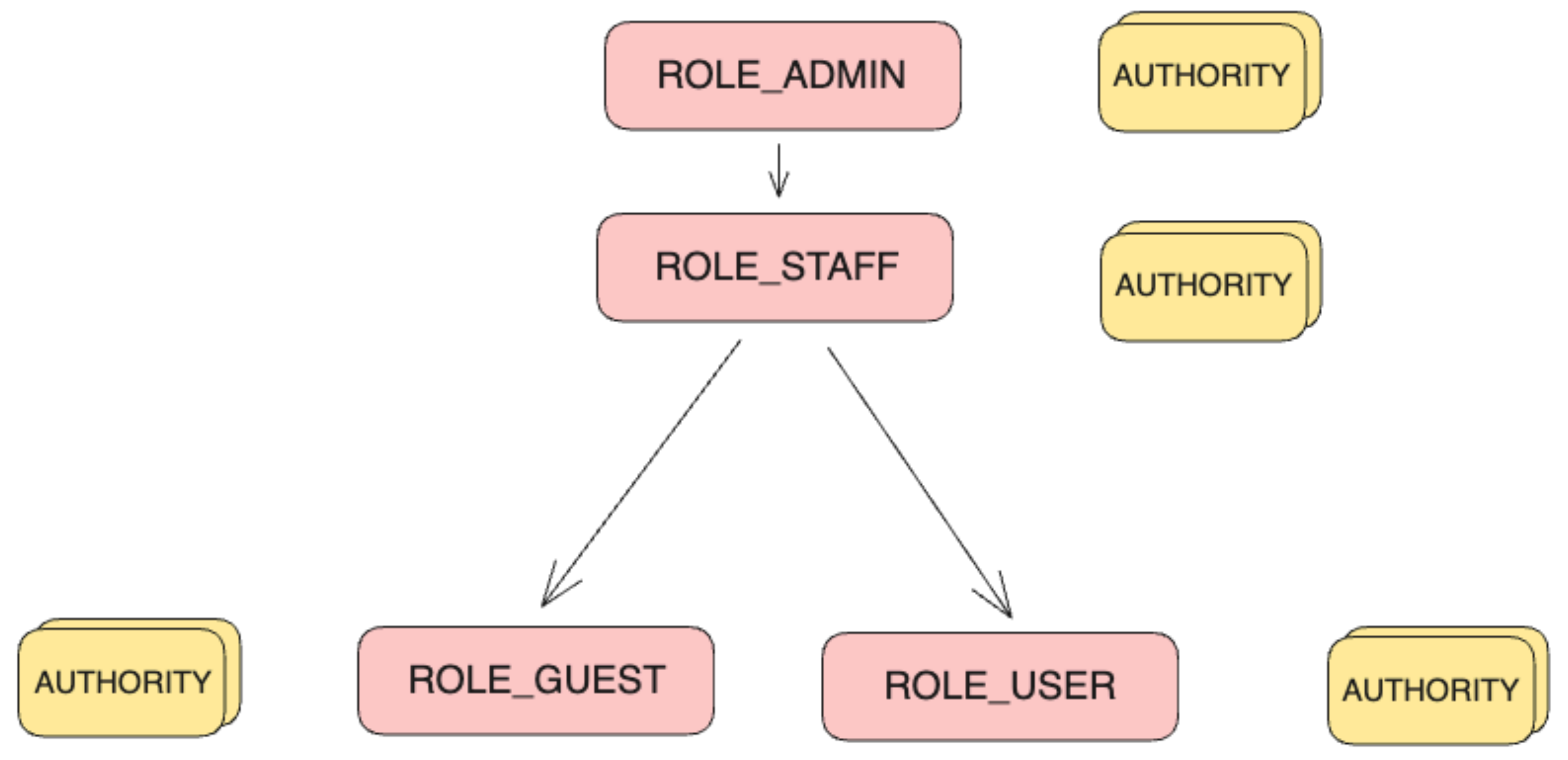


```
public interface RoleHierarchy {

    Returns an array of all reachable authorities.
    Reachable authorities are the directly assigned authorities plus all authorities that are
    (transitively) reachable from them in the role hierarchy.
    Example:
    Role hierarchy: ROLE_A > ROLE_B > ROLE_C.
    Directly assigned authority: ROLE_A.
    Reachable authorities: ROLE_A, ROLE_B, ROLE_C.
    Params: authorities – - List of the directly assigned authorities.
    Returns: List of all reachable authorities given the assigned authorities.
    Collection<? extends GrantedAuthority> getReachableGrantedAuthorities(
            Collection<? extends GrantedAuthority> authorities);

}
```

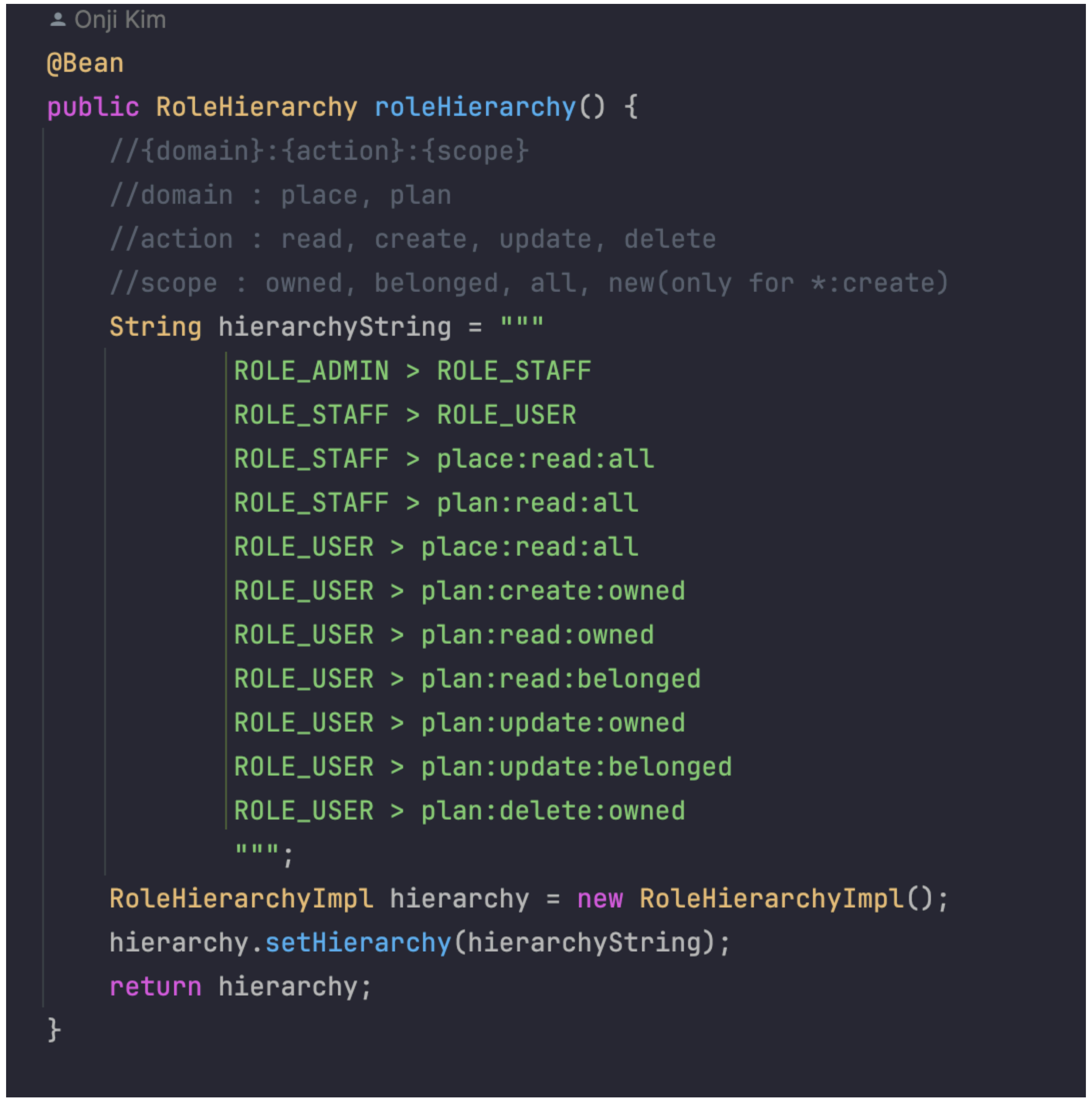


```
public interface AccessPolicy<T> {
    1 usage   1 implementation   Onji Kim
    boolean canCreate(Authentication authentication) throws ResourceNotFoundException;
    no usages   1 implementation   Onji Kim
    boolean canRead(Authentication authentication, String targetId) throws ResourceNotFoundExcep
    no usages   1 implementation   Onji Kim
    boolean canRead(Authentication authentication, List<String> targetIds) throws ResourceNotFou
```

계층적 권한 부여 시스템 개발

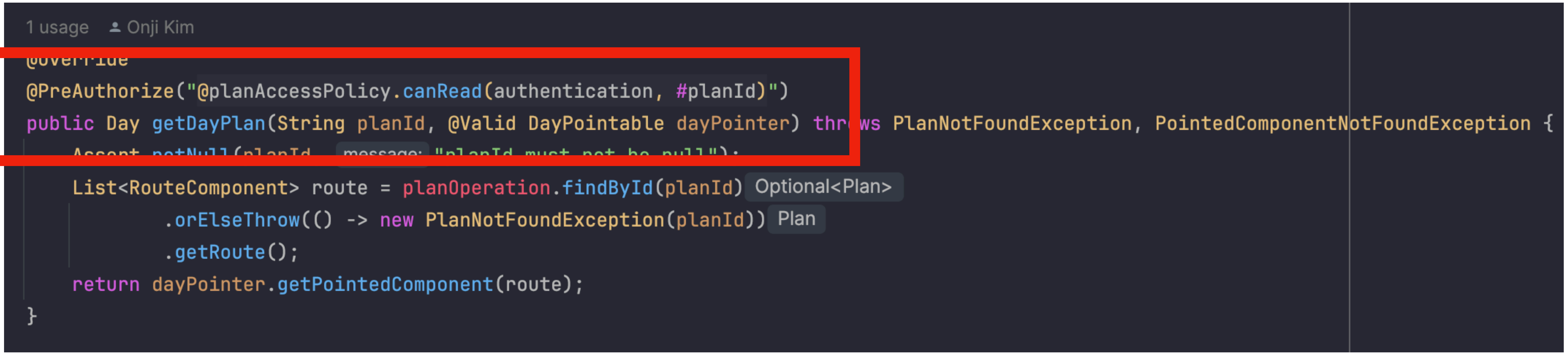


Config 실시간 업데이트



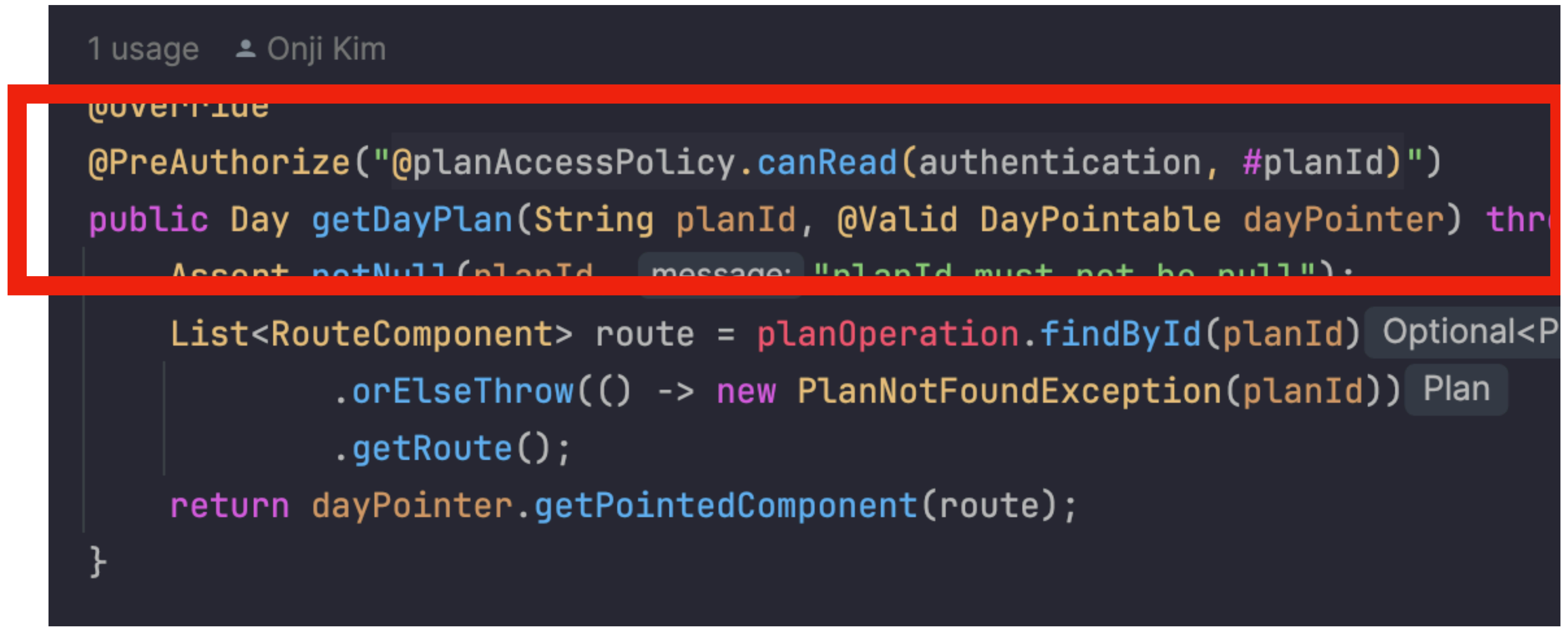
```
@Bean
public RoleHierarchy roleHierarchy() {
    //{domain}:{action}:{scope}
    //domain : place, plan
    //action : read, create, update, delete
    //scope : owned, belonged, all, new(only for *:create)
    String hierarchyString = """
            ROLE_ADMIN > ROLE_STAFF
            ROLE_STAFF > ROLE_USER
            ROLE_STAFF > place:read:all
            ROLE_STAFF > plan:read:all
            ROLE_USER > place:read:all
            ROLE_USER > plan:create:owned
            ROLE_USER > plan:read:owned
            ROLE_USER > plan:read:belonged
            ROLE_USER > plan:update:owned
            ROLE_USER > plan:update:belonged
            ROLE_USER > plan:delete:owned
            """;
    RoleHierarchyImpl hierarchy = new RoleHierarchyImpl();
    hierarchy.setHierarchy(hierarchyString);
    return hierarchy;
}
```

AOP 기반 선언형 접근제어 개발

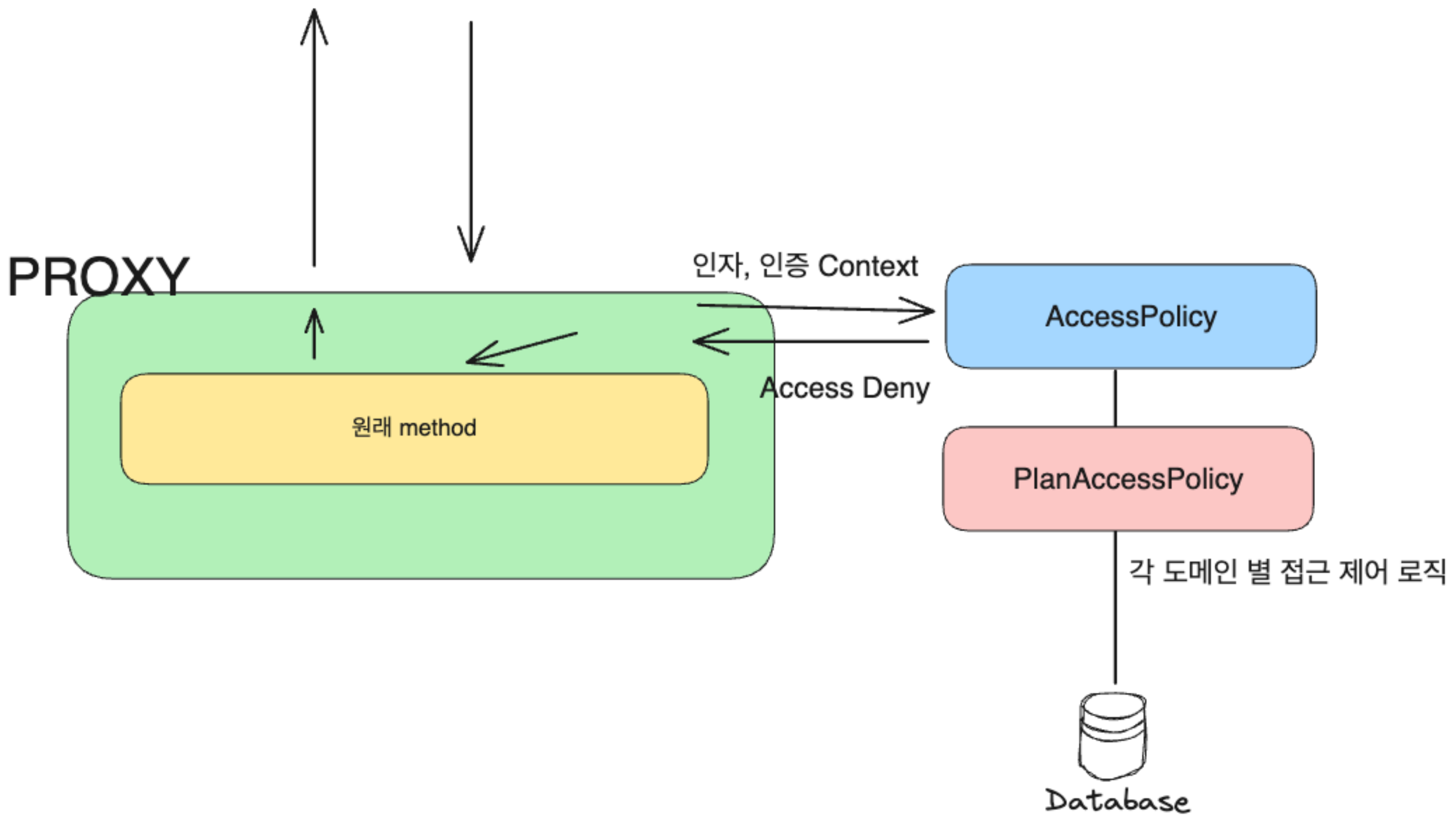


```
1 usage   Onji Kim
@Override
@PreAuthorize("@planAccessPolicy.canRead(authentication, #planId)")
public Day getDayPlan(String planId, @Valid DayPointable dayPointer) throws PlanNotFoundException, PointedComponentNotFoundException {
    Assert.notNull(planId,  message: "planId must not be null");
    List<RouteComponent> route = planOperation.findById(planId) Optional<Plan>
            .orElseThrow(() -> new PlanNotFoundException(planId)) Plan
            .getRoute();
    return dayPointer.getPointedComponent(route);
}
```

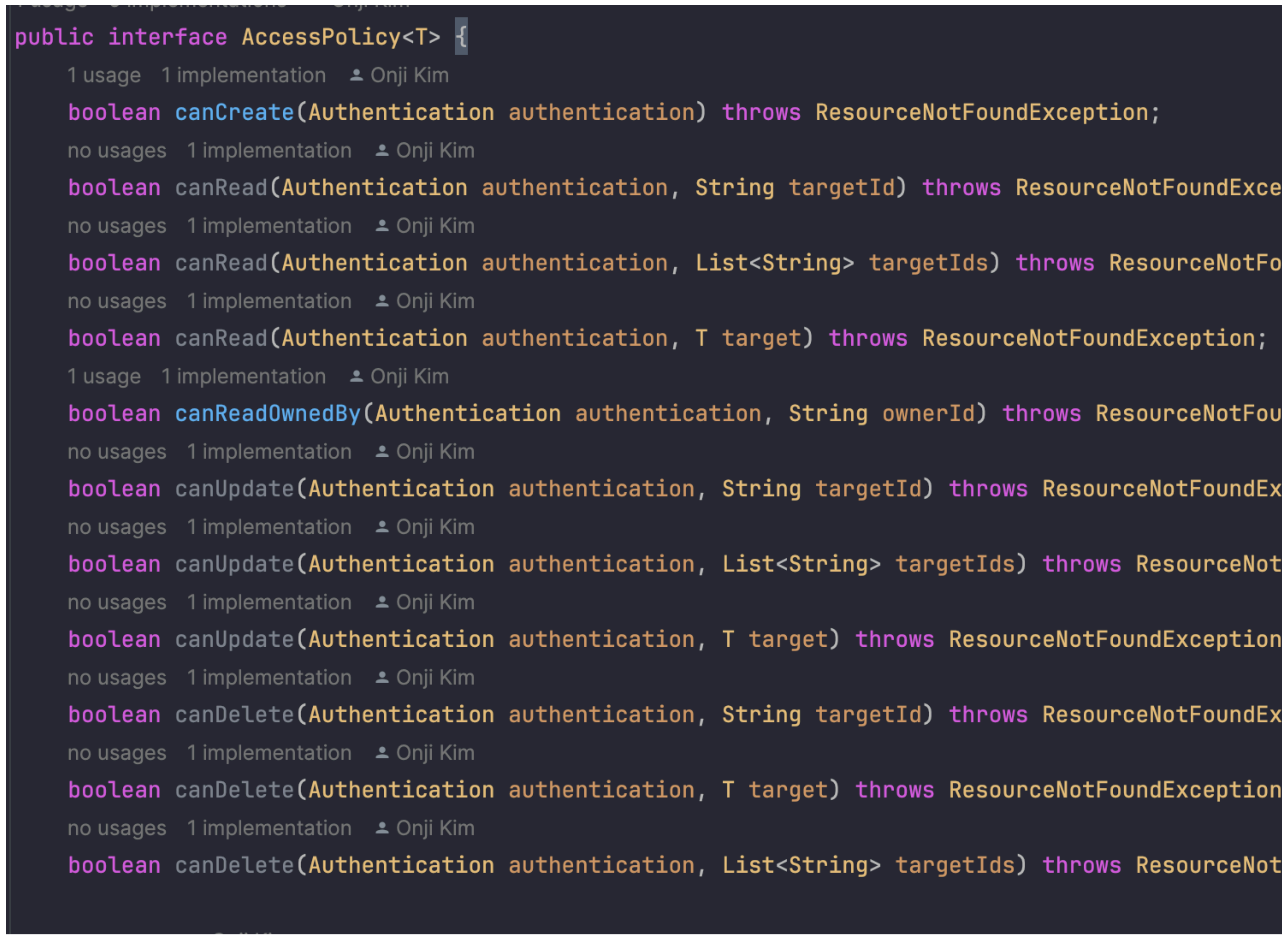
AOP 기반 선언형 접근제어 개발



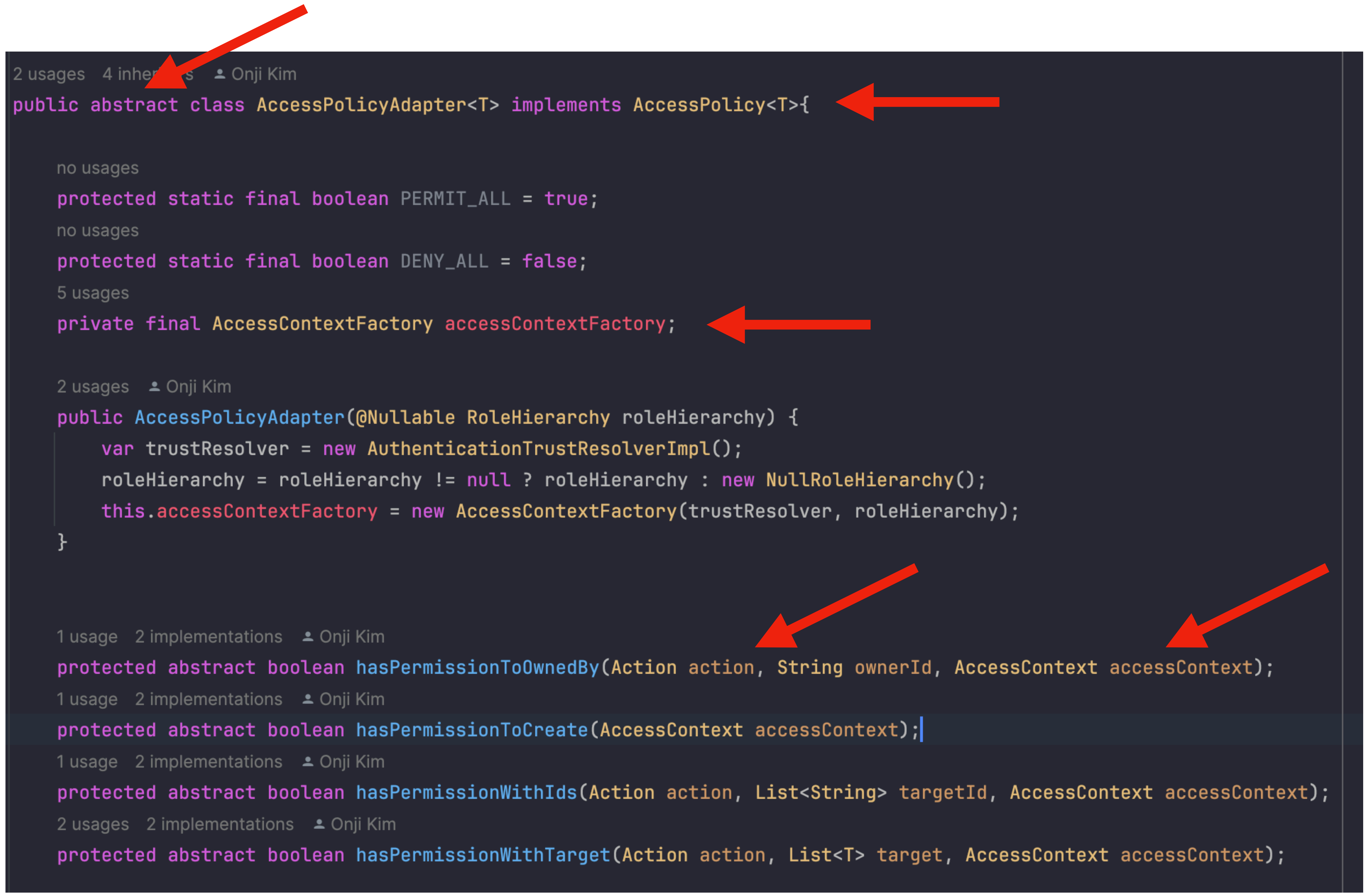
```
1 usage  Onji Kim
@Override
@PreAuthorize("@planAccessPolicy.canRead(authentication, #planId)")
public Day getDayPlan(String planId, @Valid DayPointable dayPointer) thr
    Assert.notNull(planId, message: "planId must not be null");
    List<RouteComponent> route = planOperation.findById(planId) Optional<P
            .orElseThrow(() -> new PlanNotFoundException(planId)) Plan
            .getRoute();
    return dayPointer.getPointedComponent(route);
}
```



세부 정책 구현

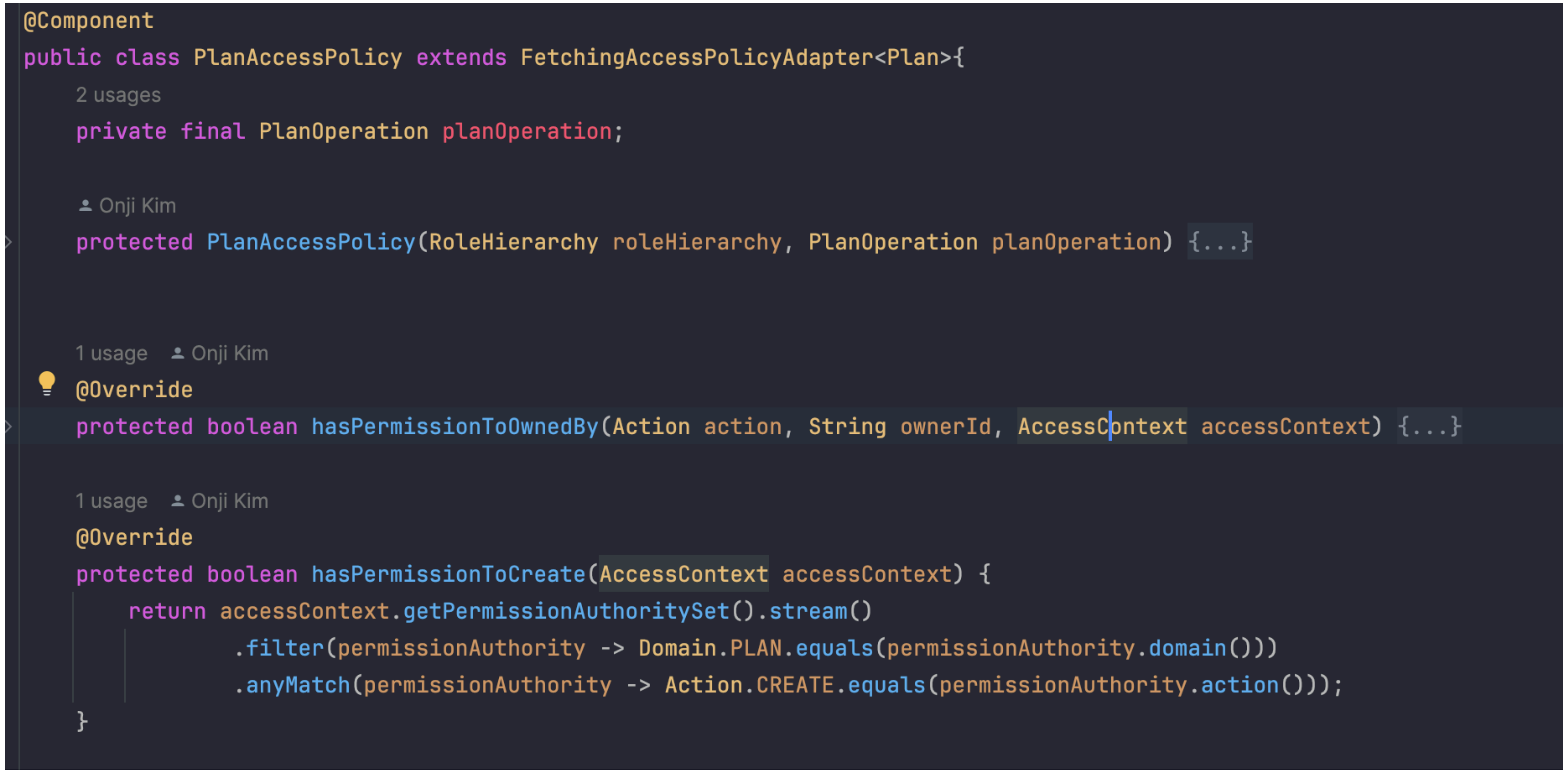


```
public interface AccessPolicy<T> {
    1 usage   1 implementation   Onji Kim
    boolean canCreate(Authentication authentication) throws ResourceNotFoundException;
    no usages   1 implementation   Onji Kim
    boolean canRead(Authentication authentication, String targetId) throws ResourceNotFoundExce
    no usages   1 implementation   Onji Kim
    boolean canRead(Authentication authentication, List<String> targetIds) throws ResourceNotFo
    no usages   1 implementation   Onji Kim
    boolean canRead(Authentication authentication, T target) throws ResourceNotFoundException;
    1 usage   1 implementation   Onji Kim
    boolean canReadOwnedBy(Authentication authentication, String ownerId) throws ResourceNotFou
    no usages   1 implementation   Onji Kim
    boolean canUpdate(Authentication authentication, String targetId) throws ResourceNotFoundEx
    no usages   1 implementation   Onji Kim
    boolean canUpdate(Authentication authentication, List<String> targetIds) throws ResourceNot
    no usages   1 implementation   Onji Kim
    boolean canUpdate(Authentication authentication, T target) throws ResourceNotFoundException
    no usages   1 implementation   Onji Kim
    boolean canDelete(Authentication authentication, String targetId) throws ResourceNotFoundEx
    no usages   1 implementation   Onji Kim
    boolean canDelete(Authentication authentication, T target) throws ResourceNotFoundException
    no usages   1 implementation   Onji Kim
    boolean canDelete(Authentication authentication, List<String> targetIds) throws ResourceNot
```



```
2 usages   4 inher[illegible]s   Onji Kim
public abstract class AccessPolicyAdapter<T> implements AccessPolicy<T>{

    no usages
    protected static final boolean PERMIT_ALL = true;
    no usages
    protected static final boolean DENY_ALL = false;
    5 usages
    private final AccessContextFactory accessContextFactory;

    2 usages   Onji Kim
    public AccessPolicyAdapter(@Nullable RoleHierarchy roleHierarchy) {
        var trustResolver = new AuthenticationTrustResolverImpl();
        roleHierarchy = roleHierarchy != null ? roleHierarchy : new NullRoleHierarchy();
        this.accessContextFactory = new AccessContextFactory(trustResolver, roleHierarchy);
    }


    1 usage   2 implementations   Onji Kim
    protected abstract boolean hasPermissionToOwnedBy(Action action, String ownerId, AccessContext accessContext);
    1 usage   2 implementations   Onji Kim
    protected abstract boolean hasPermissionToCreate(AccessContext accessContext);
    1 usage   2 implementations   Onji Kim
    protected abstract boolean hasPermissionWithIds(Action action, List<String> targetId, AccessContext accessContext);
    2 usages   2 implementations   Onji Kim
    protected abstract boolean hasPermissionWithTarget(Action action, List<T> target, AccessContext accessContext);
```

세부 정책 구현



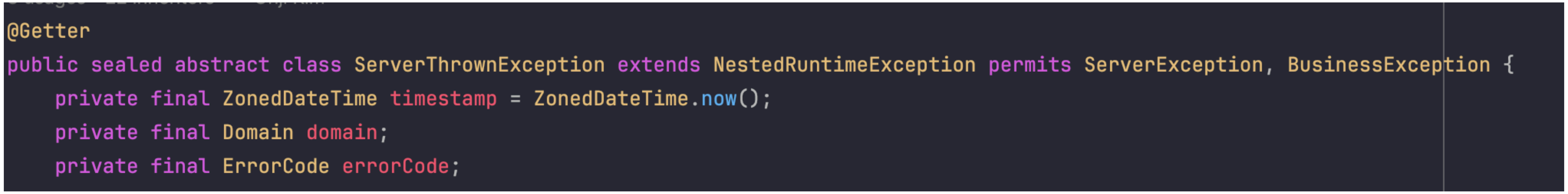
```
@Component
public class PlanAccessPolicy extends FetchingAccessPolicyAdapter<Plan>{
    2 usages
    private final PlanOperation planOperation;

    Onji Kim
    protected PlanAccessPolicy(RoleHierarchy roleHierarchy, PlanOperation planOperation) {...}


    1 usage   Onji Kim
    @Override
    protected boolean hasPermissionToOwnedBy(Action action, String ownerId, AccessContext accessContext) {...}

    1 usage   Onji Kim
    @Override
    protected boolean hasPermissionToCreate(AccessContext accessContext) {
        return accessContext.getPermissionAuthoritySet().stream()
                .filter(permissionAuthority -> Domain.PLAN.equals(permissionAuthority.domain()))
                .anyMatch(permissionAuthority -> Action.CREATE.equals(permissionAuthority.action()));
    }
```

예외 처리 리팩토링

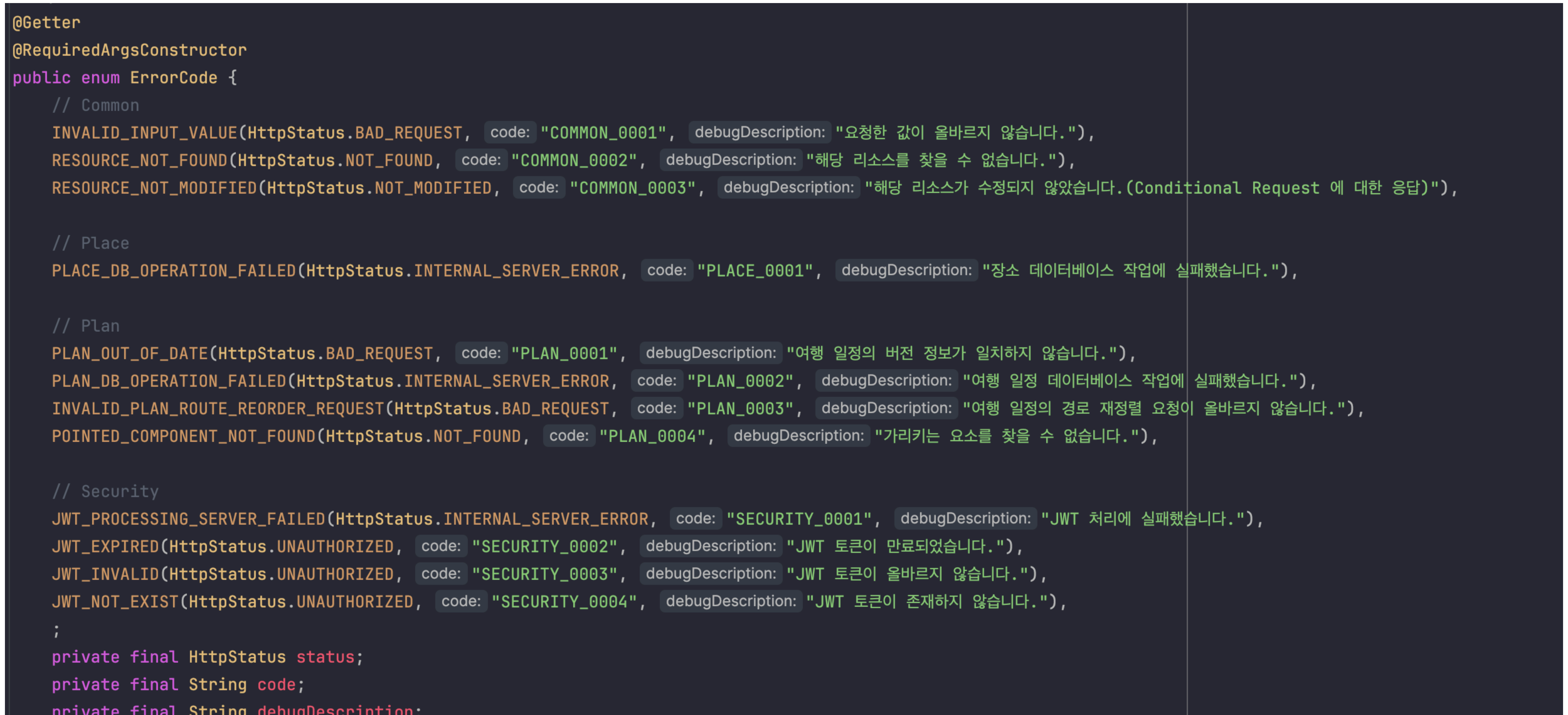
예외처리 체계화

NestedRuntimeException
Getter
ServerThrownException
ServerException
BusinessException
PlanServerException
PlaceBusinessException
PlanBusinessException
PlaceServerException
SecurityServerException
SecurityBusinessException
PlanOperationFailedException
PlaceNotFoundException
JwtTokenInvalidException
JwtTokenExpiredException
PointedComponentNotFoundException
PlanNotModifiedException
PlanNotFoundException
PlaceRetrieveFailedException
JwtProcessingServerException
JwtNotExistException
ResourceNotFoundException
InvalidRouteReorderRequestException
PlanVersionOutOfDateException
InvalidUpdateInfoException

예외처리 체계화

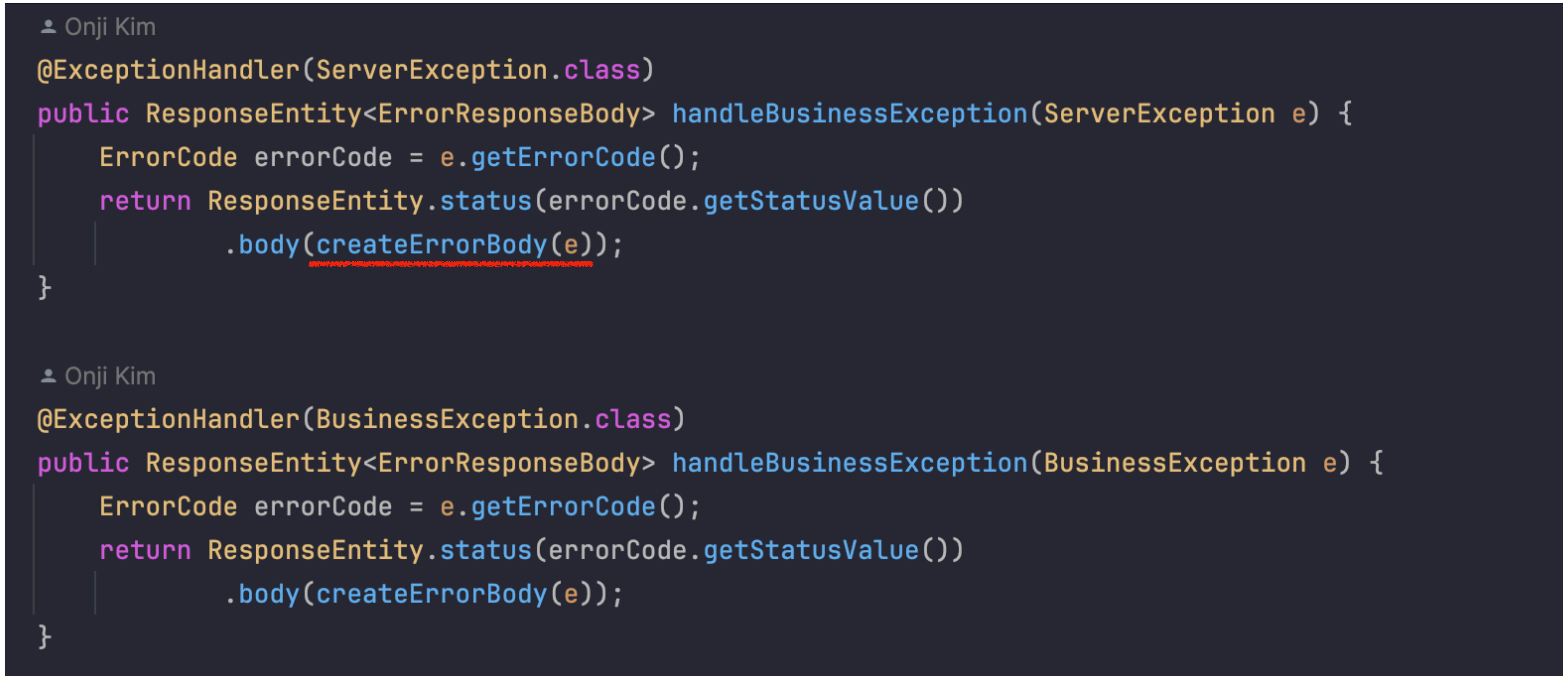


```java
@Getter
public sealed abstract class ServerThrownException extends NestedRuntimeException permits ServerException, BusinessException {
    private final ZonedDateTime timestamp = ZonedDateTime.now();
    private final Domain domain;
    private final ErrorCode errorCode;
```



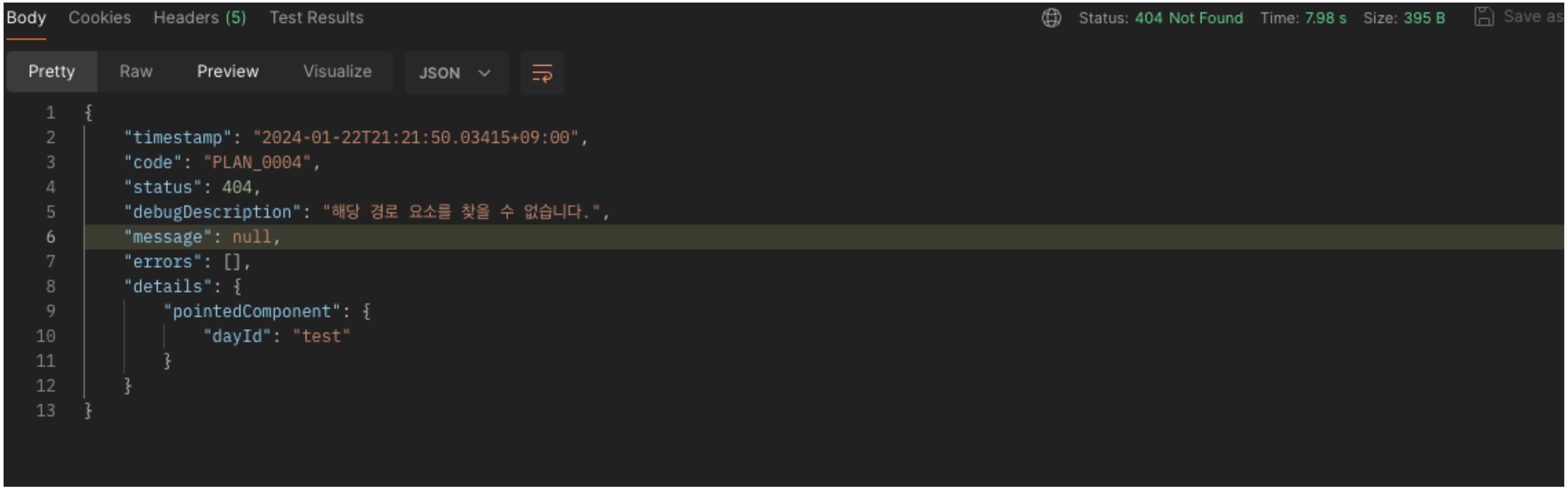
```java
@Getter
@RequiredArgsConstructor
public enum ErrorCode {
    // Common
    INVALID_INPUT_VALUE(HttpStatus.BAD_REQUEST, code: "COMMON_0001", debugDescription: "요청한 값이 올바르지 않습니다."),
    RESOURCE_NOT_FOUND(HttpStatus.NOT_FOUND, code: "COMMON_0002", debugDescription: "해당 리소스를 찾을 수 없습니다."),
    RESOURCE_NOT_MODIFIED(HttpStatus.NOT_MODIFIED, code: "COMMON_0003", debugDescription: "해당 리소스가 수정되지 않았습니다.(Conditional Request 에 대한 응답)"),

    // Place
    PLACE_DB_OPERATION_FAILED(HttpStatus.INTERNAL_SERVER_ERROR, code: "PLACE_0001", debugDescription: "장소 데이터베이스 작업에 실패했습니다."),

    // Plan
    PLAN_OUT_OF_DATE(HttpStatus.BAD_REQUEST, code: "PLAN_0001", debugDescription: "여행 일정의 버전 정보가 일치하지 않습니다."),
    PLAN_DB_OPERATION_FAILED(HttpStatus.INTERNAL_SERVER_ERROR, code: "PLAN_0002", debugDescription: "여행 일정 데이터베이스 작업에 실패했습니다."),
    INVALID_PLAN_ROUTE_REORDER_REQUEST(HttpStatus.BAD_REQUEST, code: "PLAN_0003", debugDescription: "여행 일정의 경로 재정렬 요청이 올바르지 않습니다."),
    POINTED_COMPONENT_NOT_FOUND(HttpStatus.NOT_FOUND, code: "PLAN_0004", debugDescription: "가리키는 요소를 찾을 수 없습니다."),

    // Security
    JWT_PROCESSING_SERVER_FAILED(HttpStatus.INTERNAL_SERVER_ERROR, code: "SECURITY_0001", debugDescription: "JWT 처리에 실패했습니다."),
    JWT_EXPIRED(HttpStatus.UNAUTHORIZED, code: "SECURITY_0002", debugDescription: "JWT 토큰이 만료되었습니다."),
    JWT_INVALID(HttpStatus.UNAUTHORIZED, code: "SECURITY_0003", debugDescription: "JWT 토큰이 올바르지 않습니다."),
    JWT_NOT_EXIST(HttpStatus.UNAUTHORIZED, code: "SECURITY_0004", debugDescription: "JWT 토큰이 존재하지 않습니다."),
    ;
    private final HttpStatus status;
    private final String code;
    private final String debugDescription;
```

예외처리 체계화



```
Onji Kim
@ExceptionHandler(ServerException.class)
public ResponseEntity<ErrorResponseBody> handleBusinessException(ServerException e) {
    ErrorCode errorCode = e.getErrorCode();
    return ResponseEntity.status(errorCode.getStatusValue())
            .body(createErrorBody(e));
}

Onji Kim
@ExceptionHandler(BusinessException.class)
public ResponseEntity<ErrorResponseBody> handleBusinessException(BusinessException e) {
    ErrorCode errorCode = e.getErrorCode();
    return ResponseEntity.status(errorCode.getStatusValue())
            .body(createErrorBody(e));
}
```

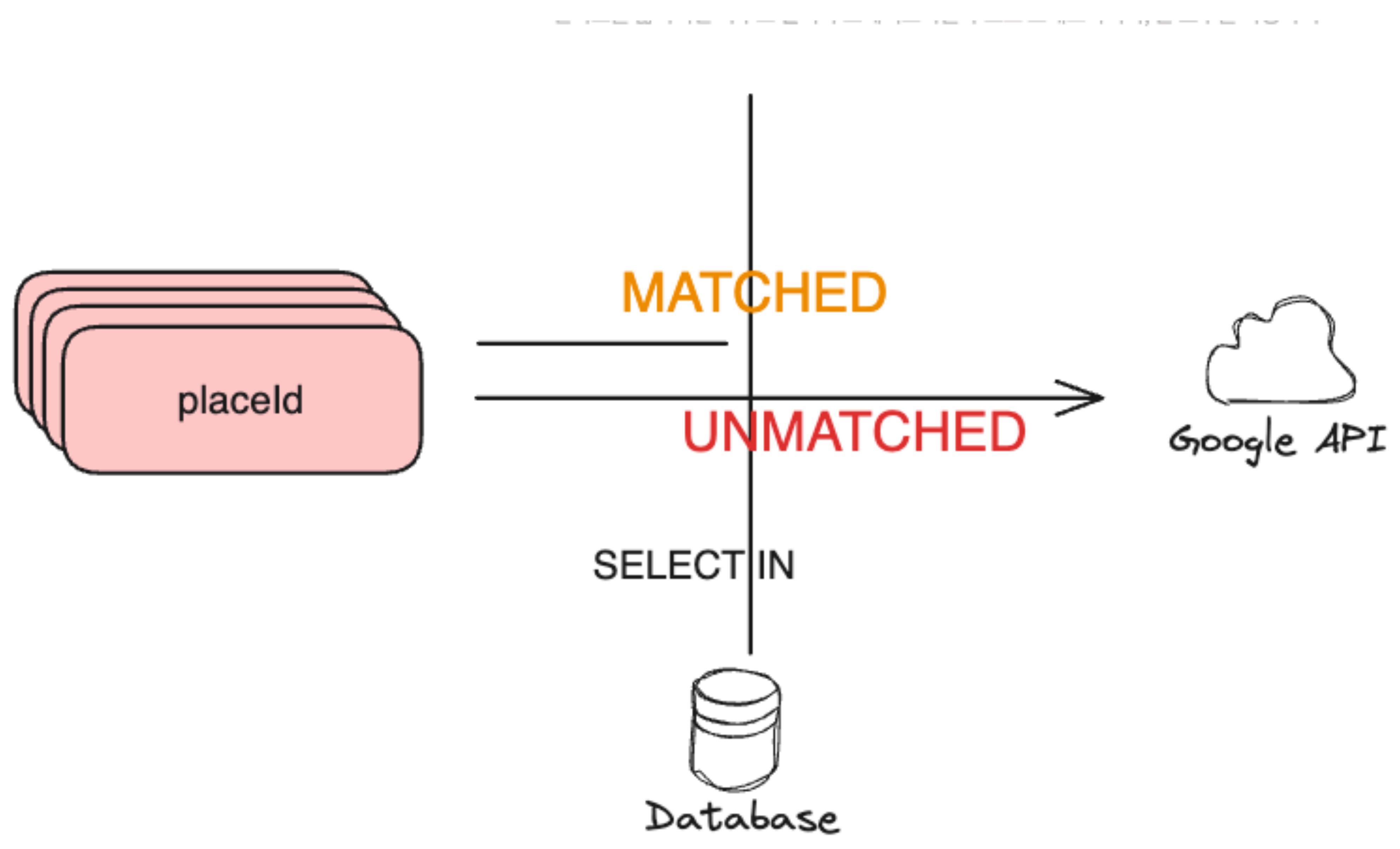
예외처리 체계화

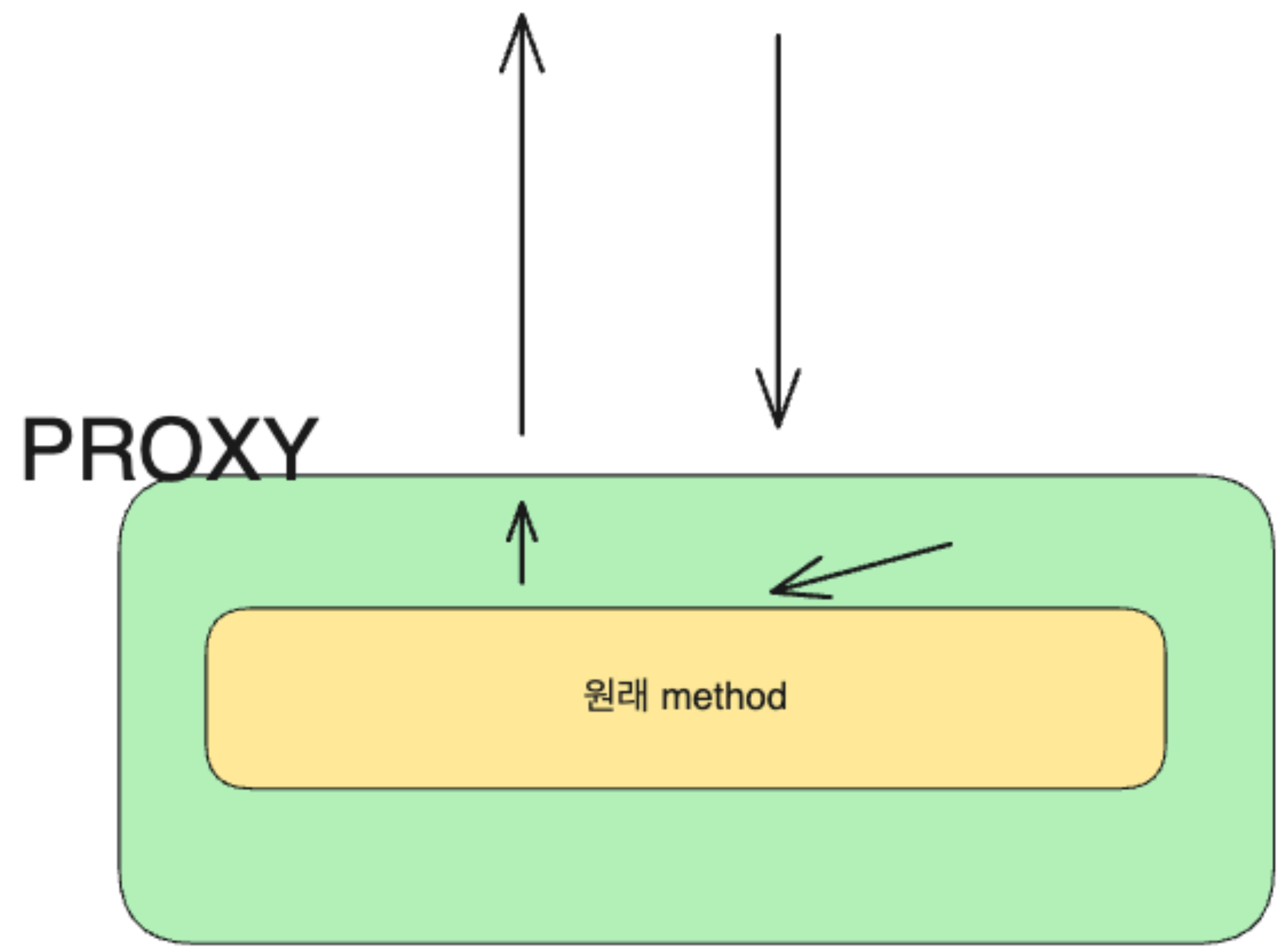


```
{
    "timestamp": "2024-01-22T21:21:50.03415+09:00",
    "code": "PLAN_0004",
    "status": 404,
    "debugDescription": "해당 경로 요소를 찾을 수 없습니다.",
    "message": null,
    "errors": [],
    "details": {
        "pointedComponent": {
            "dayId": "test"
        }
    }
}
```

부분 캐시 구현, Plan service 구현

부분 캐시 구현

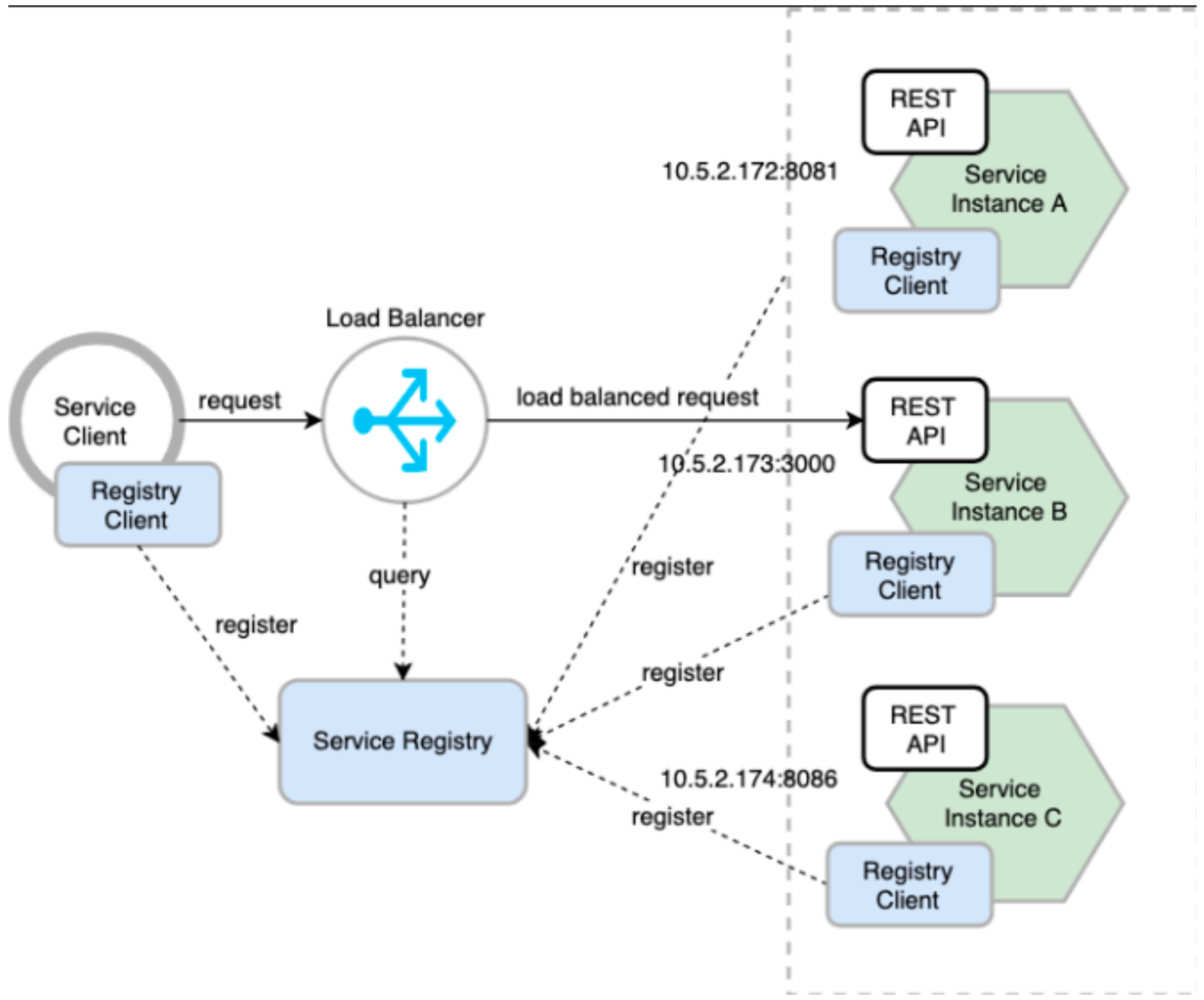


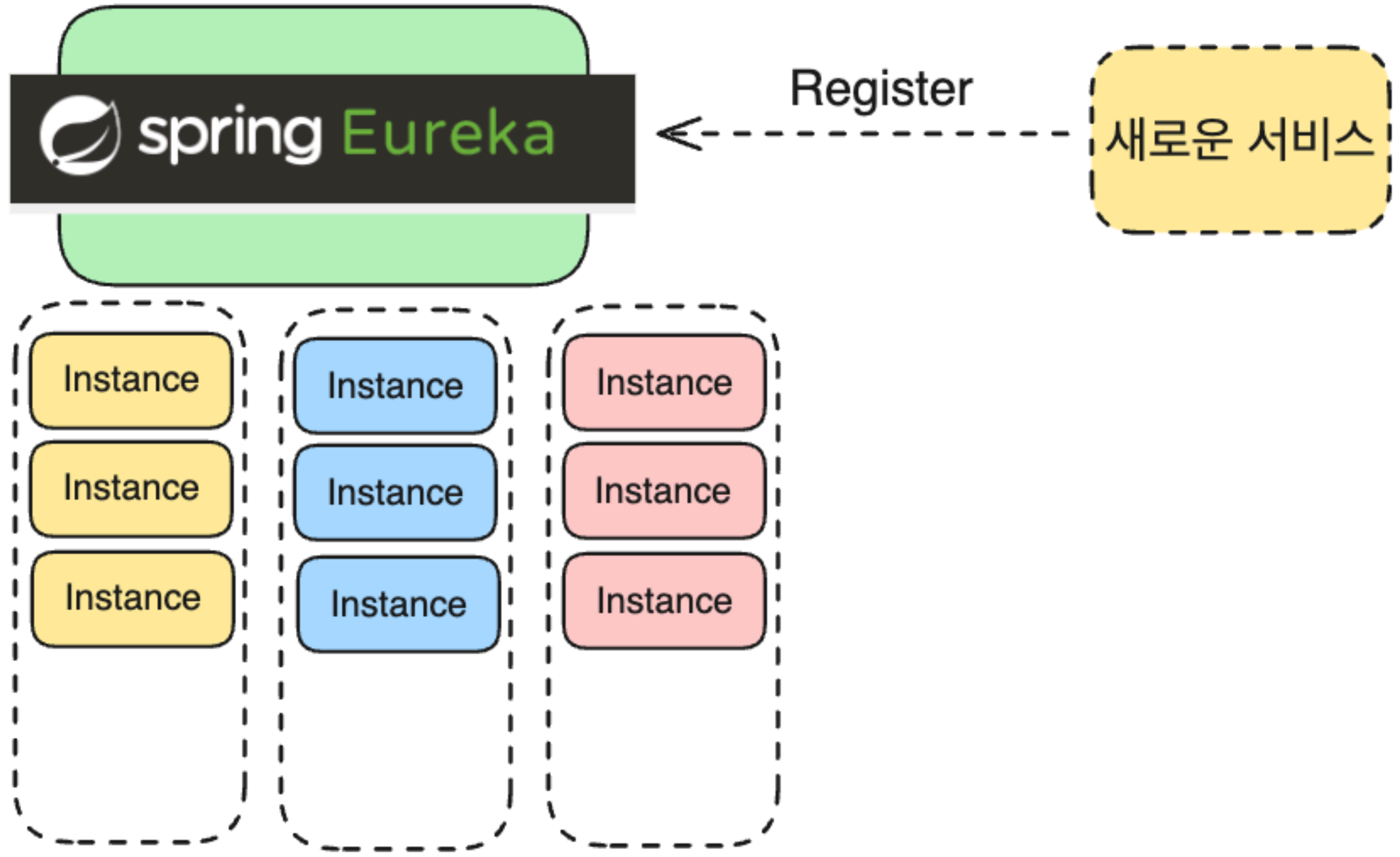


MATCHED + FETCHED ⟶ sort ⟶ return

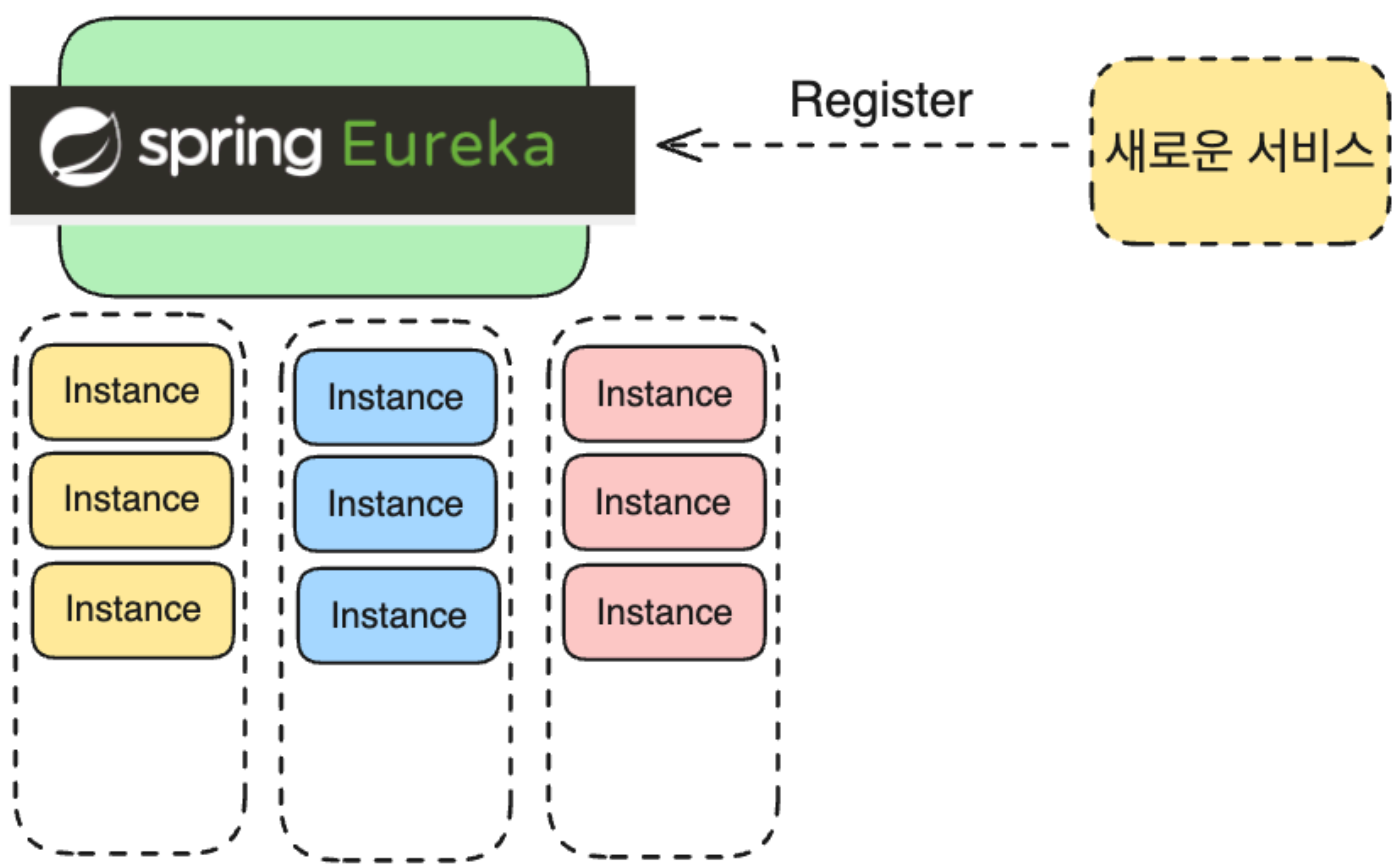
Service Discovery 도입

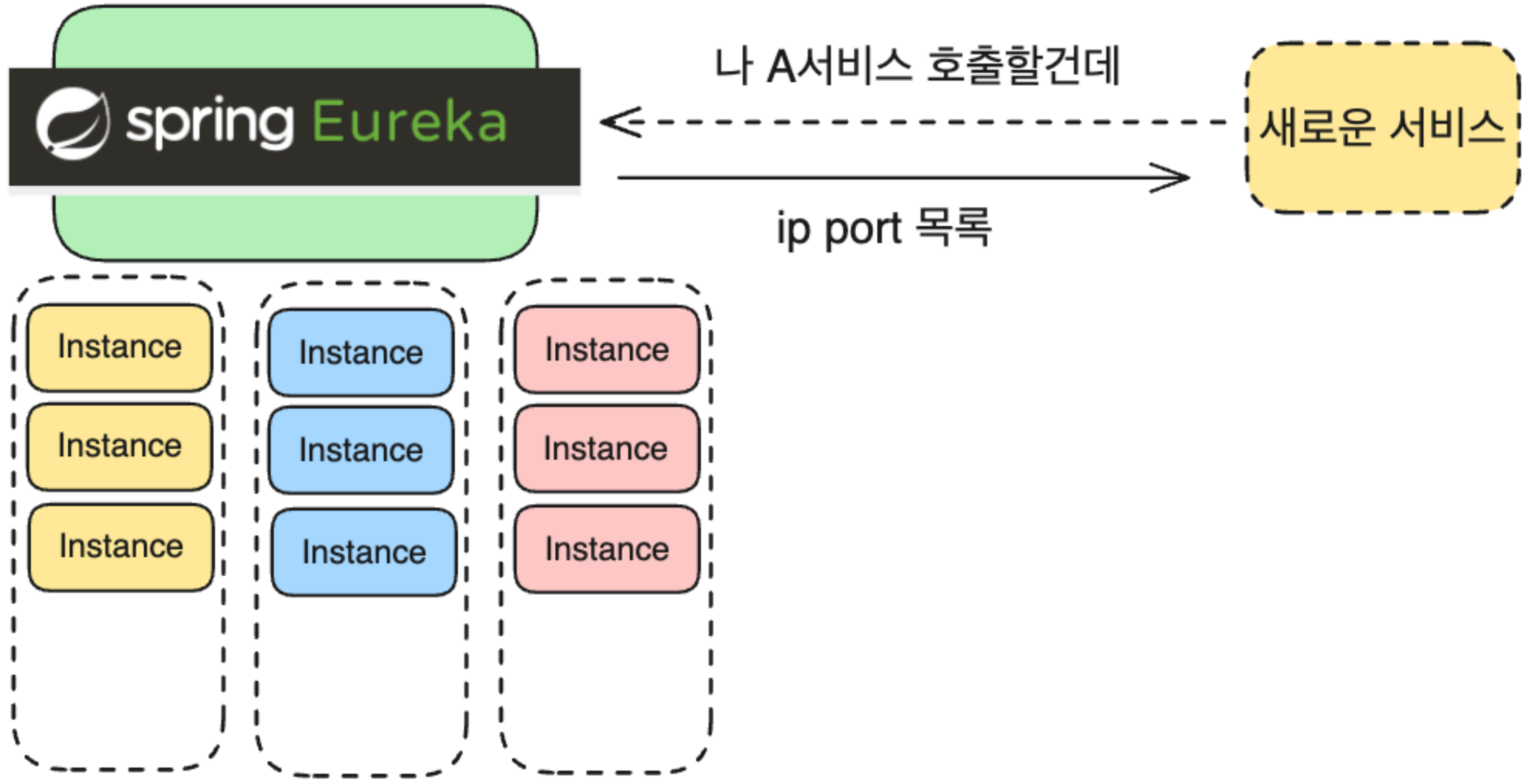
Service Discovery



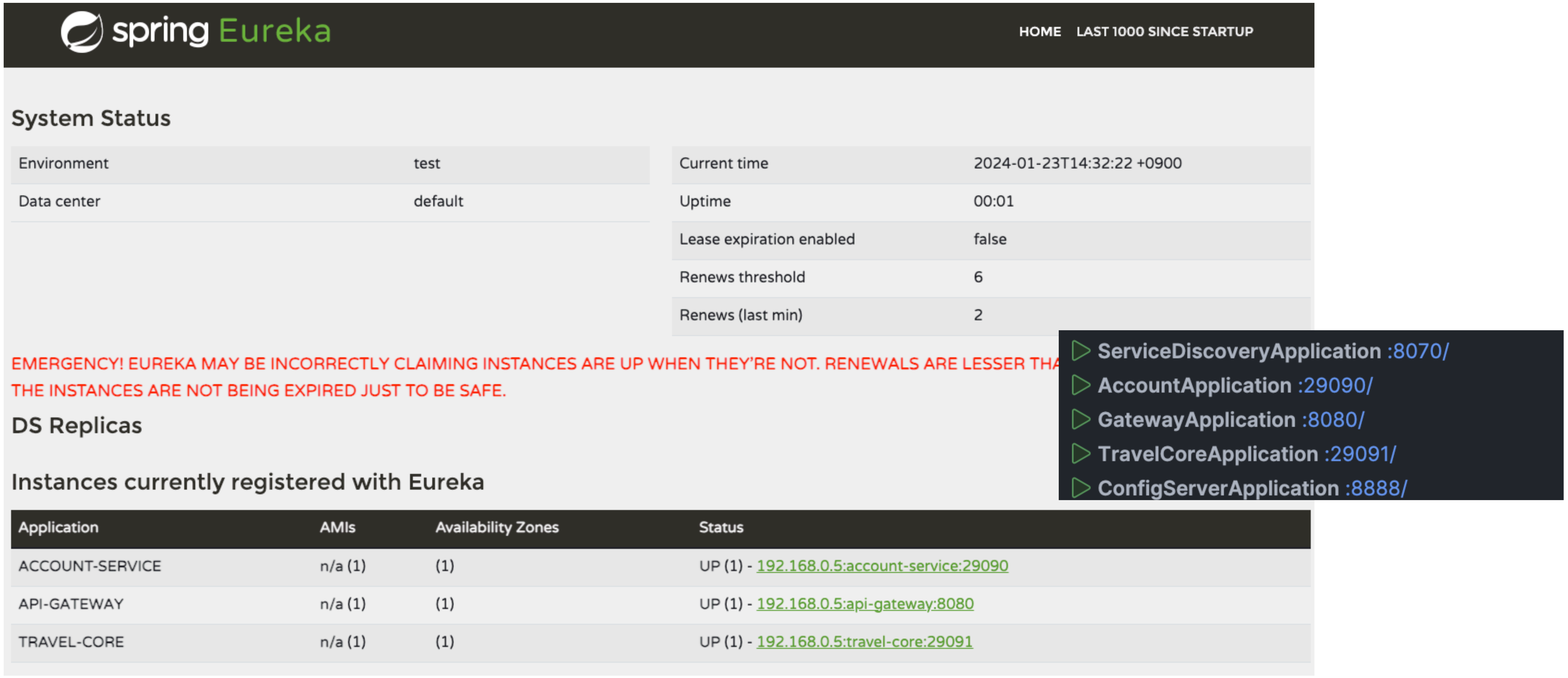


Service Discovery





Service Discovery



spring Eureka

HOME LAST 1000 SINCE STARTUP

System Status

Environment	test
Data center	default

Current time	2024-01-23T14:32:22 +0900
Uptime	00:01
Lease expiration enabled	false
Renews threshold	6
Renews (last min)	2

EMERGENCY! EUREKA MAY BE INCORRECTLY CLAIMING INSTANCES ARE UP WHEN THEY'RE NOT. RENEWALS ARE LESSER THA
THE INSTANCES ARE NOT BEING EXPIRED JUST TO BE SAFE.

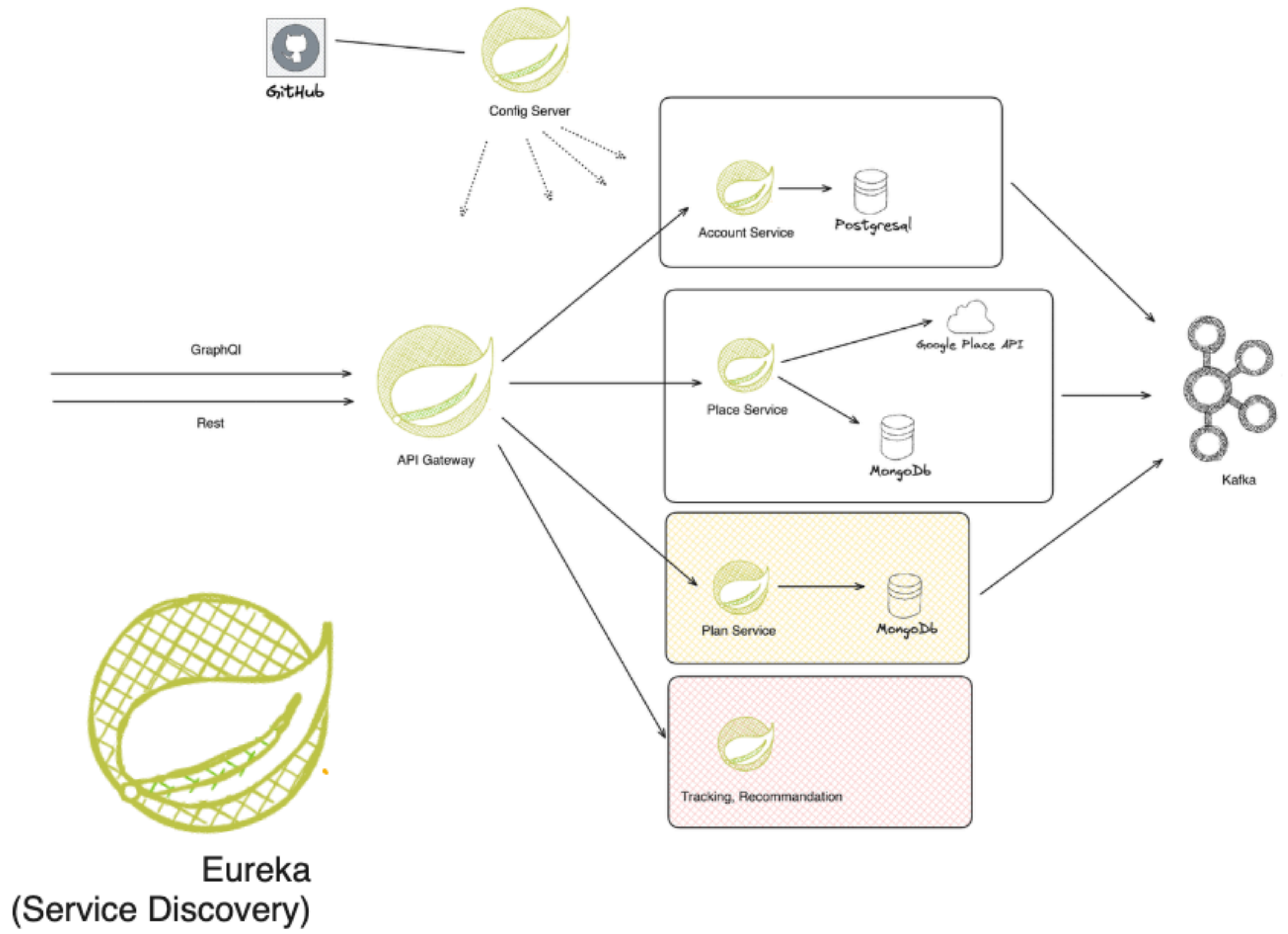
DS Replicas

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
ACCOUNT-SERVICE	n/a (1)	(1)	UP (1) - 192.168.0.5:account-service:29090
API-GATEWAY	n/a (1)	(1)	UP (1) - 192.168.0.5:api-gateway:8080
TRAVEL-CORE	n/a (1)	(1)	UP (1) - 192.168.0.5:travel-core:29091

```
ServiceDiscoveryApplication :8070/
AccountApplication :29090/
GatewayApplication :8080/
TravelCoreApplication :29091/
ConfigServerApplication :8888/
```

서버 동작 순서



부하가 걸린다.

1. 서버에 부하가 걸린다.
2. 자동으로 새로운 인스턴스를 추가한다.(수평적 확장)

서버가 추가된다.

1. Config 서버에 설정파일을 질의한다
2. Eureka에 자기자신을 등록한다

다른 서버가 이 서비스를 호출한다

1. Eureka에 Place 서비스 목록을 질의한다
2. Round Robin 방식으로 부하를 분산하며 호출한다

감사합니다.