# Day 2

## Executive summary

Delivered a clean, layered domain model and API surface for the Uber-like backend: core JPA entities, DTOs for API transport, repository interfaces, service contracts, and a ModelMapper bean to centralize mapping. The design separates persistence concerns from API contracts and prepares the project for spatial queries, ride lifecycle orchestration, and wallet/payment flows.

## Deliverables (added)

- **Configuration**
  - MapperConfig — Spring @Configuration that exposes a ModelMapper bean for DTO ⇔ entity mapping.
- **DTO layer (API contract)**
  - SignupDto, UserDto, DriverDto, RiderDto, RideRequestDto, RideDto — shapes the JSON/api surface for auth, profiles, ride requests, rides and payments. DTOs mirror domain concerns and include spatial points for pickup/dropoff.
- **Domain model (JPA entities)**
  - User — core user with roles as a collection.
  - Driver / Rider — role-specific profile entities linked @OneToOne to User; driver contains availability, rating, and current location (Point).
  - RideRequest — posted by rider; pickup/dropoff points, requested time, and request status.
  - Ride — lifecycle record with created/started/ended timestamps, fare, status, and references to rider & driver.
  - Wallet & WalletTransaction — balance and transaction audit trail tied to user and rides.
- **Repository layer**
  - UserRepository, DriverRepository, RiderRepository, RideRepository, RideRequestRepository — JpaRepository interfaces for persistence and query extension.
- **Service contracts**
  - AuthService — login, signup, and driver onboarding.
  - RiderService / DriverService — ride request lifecycle, rating, profile retrieval and ride operations.
  - RideService — matching, ride creation, status updates, and paginated ride retrieval.
  - DistanceService — abstraction for geospatial distance calculation.

## Data flow & responsibilities

1. **Signup / Auth**: SignupDto → AuthService.signup(...) → persist User.

2. **Rider posts a ride request**: client sends RideRequestDto → RideService.matchWithDrivers(...) uses DistanceService to locate drivers → Ride is created and persisted.
3. **Driver lifecycle**: DriverService methods accept/start/end/cancel rides, update Ride status, and produce RideDto for API responses.
4. **Payments**: Wallet and WalletTransaction capture monetary flows; transactions reference rides for traceability.

## Design intent & architectural notes

- **Separation of concerns**: DTOs abstract the transport layer; entities model persistence. ModelMapper centralizes mapping to avoid leaking JPA internals to controllers.
- **Spatial readiness**: Usage of org.locationtech.jts.geom.Point and columnDefinition = "Geometry(Point,4326)" signals intent to use PostGIS/hibernate-spatial for geo-queries.
- **Auditability**: WalletTransaction and timestamps on Ride/RideRequest provide an audit trail for payments and lifecycle events.
- **Extensibility**: Service interfaces are defined to allow multiple implementations (e.g., local distance calc vs. external geo-service).

## Quick risks / considerations (to align expectations)

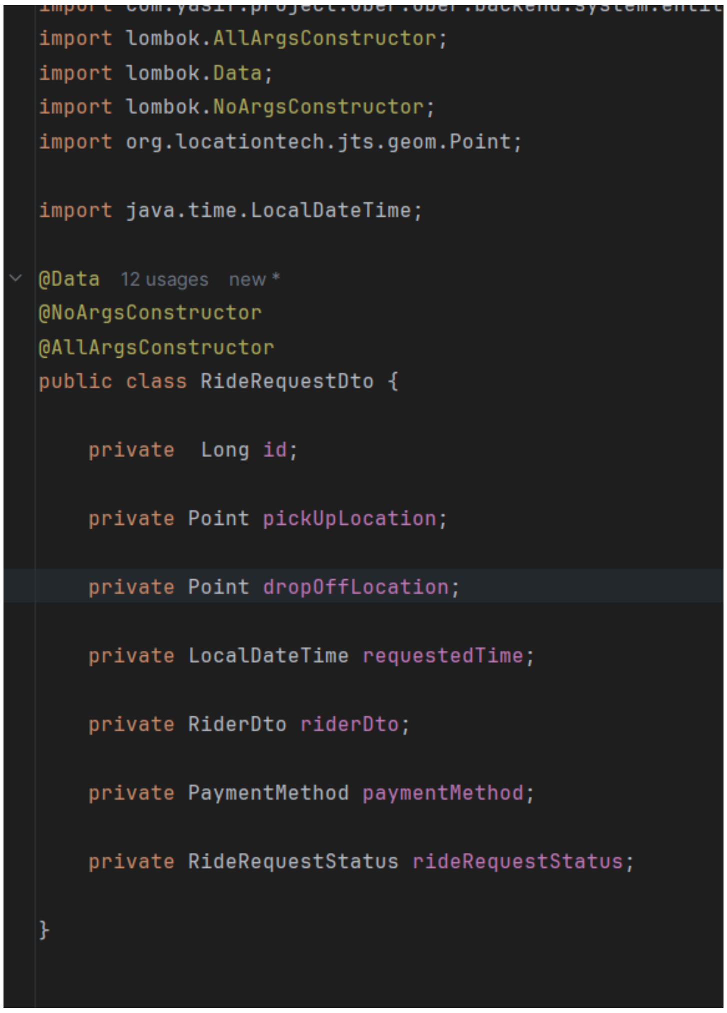
- Point serialization and DB support require additional configuration (Jackson serializers + PostGIS + hibernate-spatial).
- Concurrency in driver matching needs atomic DB operations or locks to prevent double-assignments.
- Enum fields and timestamp annotations may need consistency checks (@Enumerated, @CreationTimestamp etc.).
- Passwords must be hashed and never exposed via DTOs.

## Closing

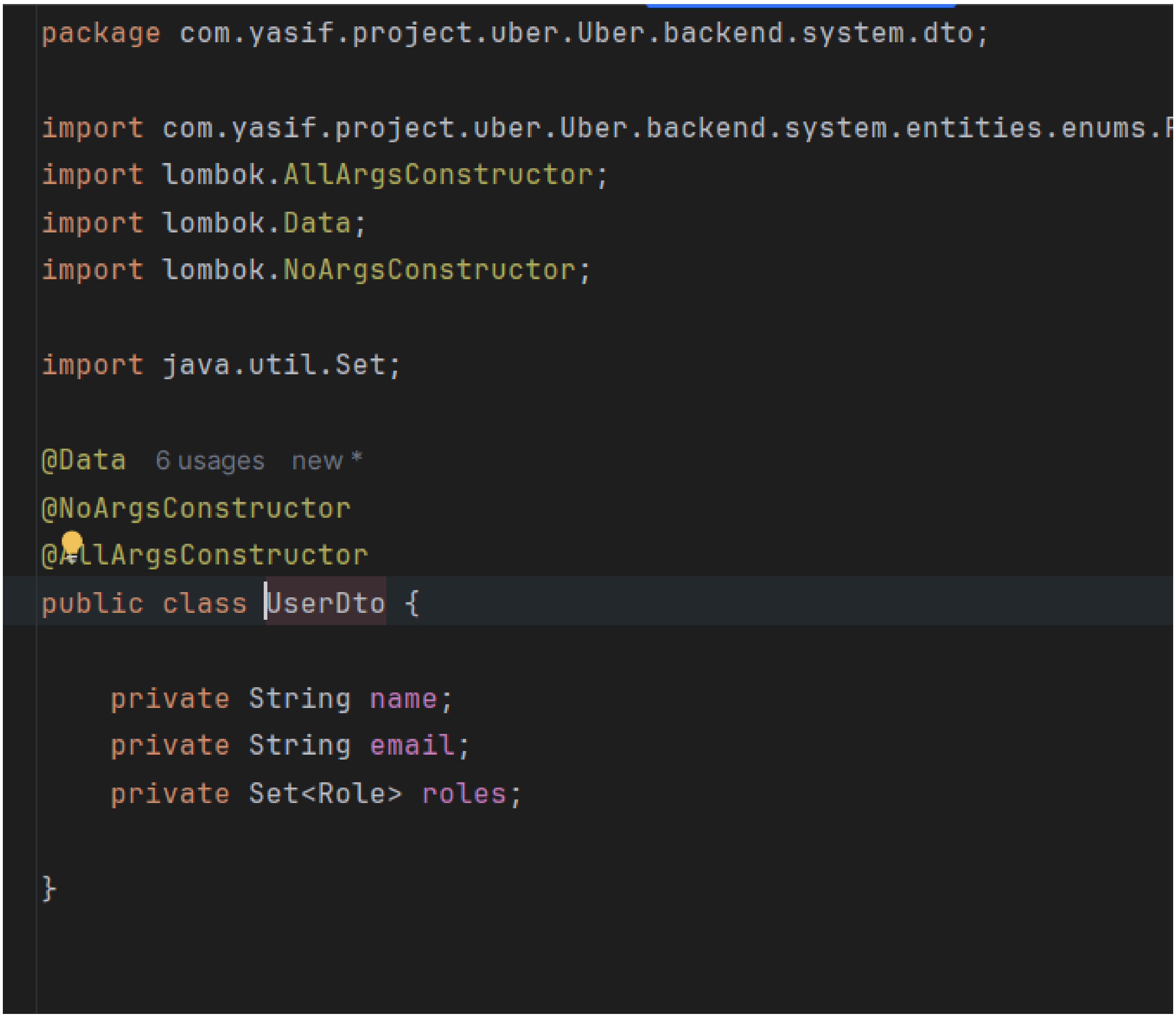
This is a production-ready scaffold that outlines entities, contracts, and persistence boundaries — a strong foundation for implementing matching logic, secure auth, spatial queries, and payment integration.

## ScreenShots

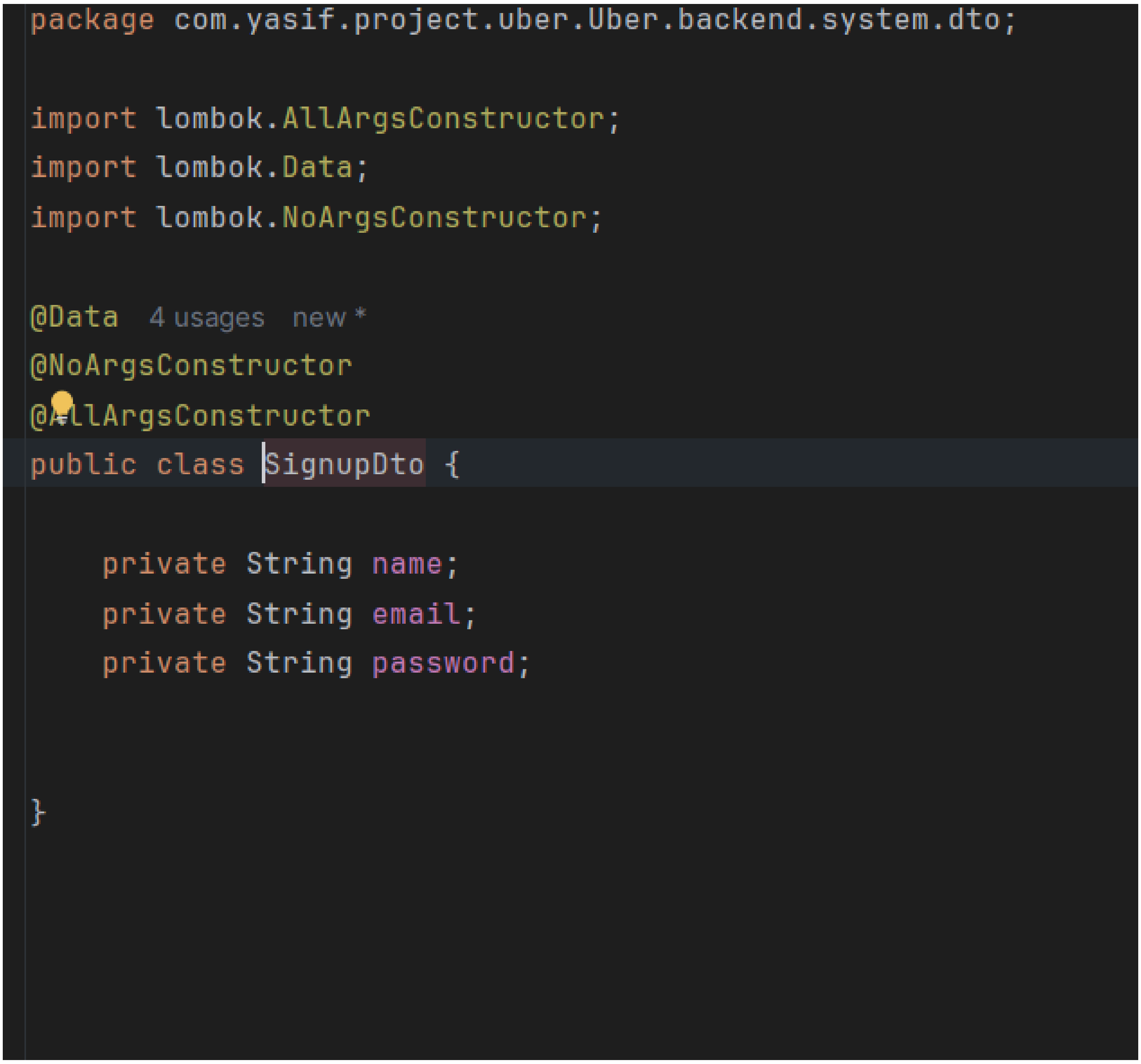
### Dto



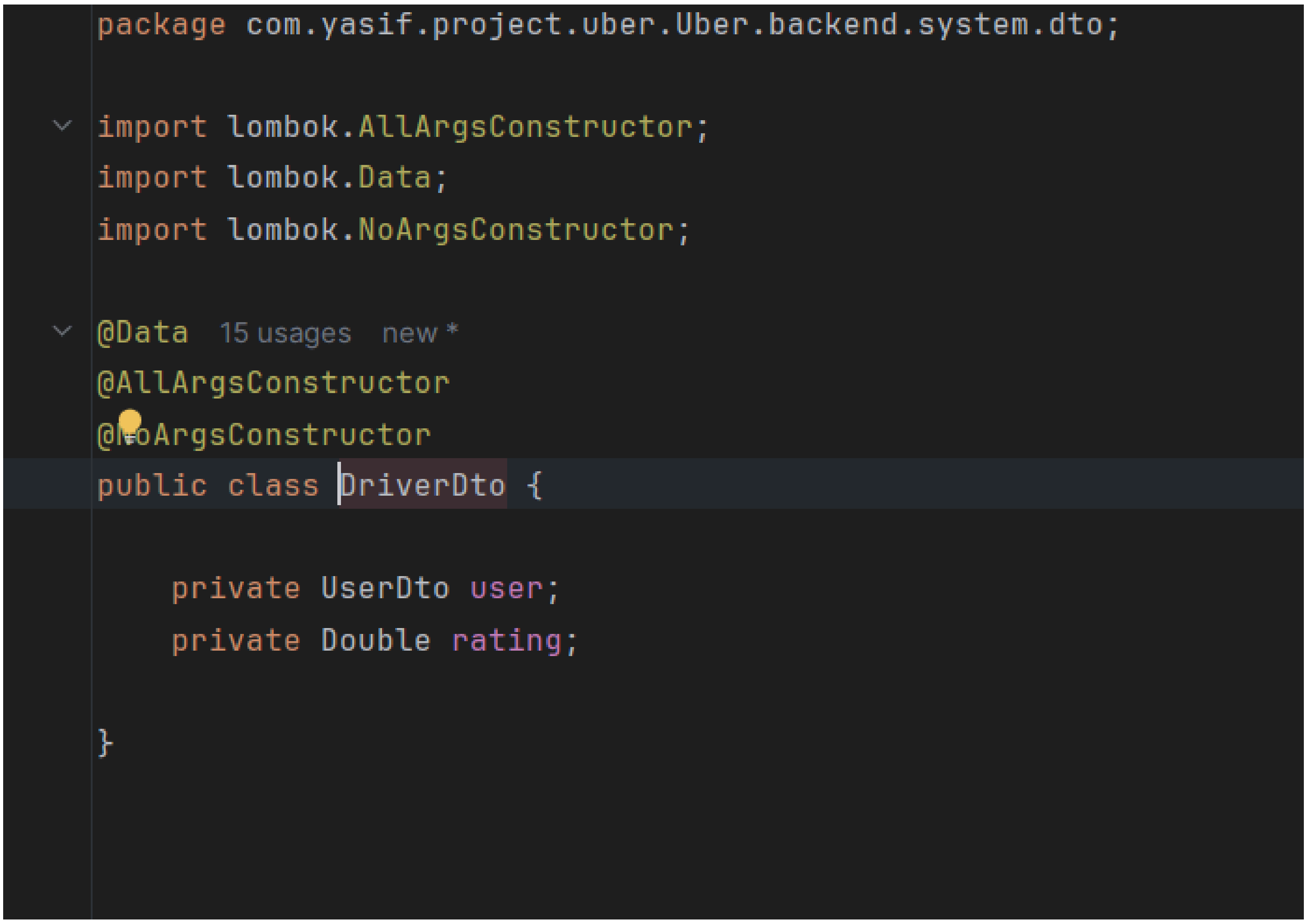
```java
import com.yasir.project.uber.uber.backend.system.entit
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import org.locationtech.jts.geom.Point;

import java.time.LocalDateTime;

@Data  12 usages  new *
@NoArgsConstructor
@AllArgsConstructor
public class RideRequestDto {

    private  Long id;

    private Point pickUpLocation;

    private Point dropOffLocation;

    private LocalDateTime requestedTime;

    private RiderDto riderDto;

    private PaymentMethod paymentMethod;

    private RideRequestStatus rideRequestStatus;

}
```



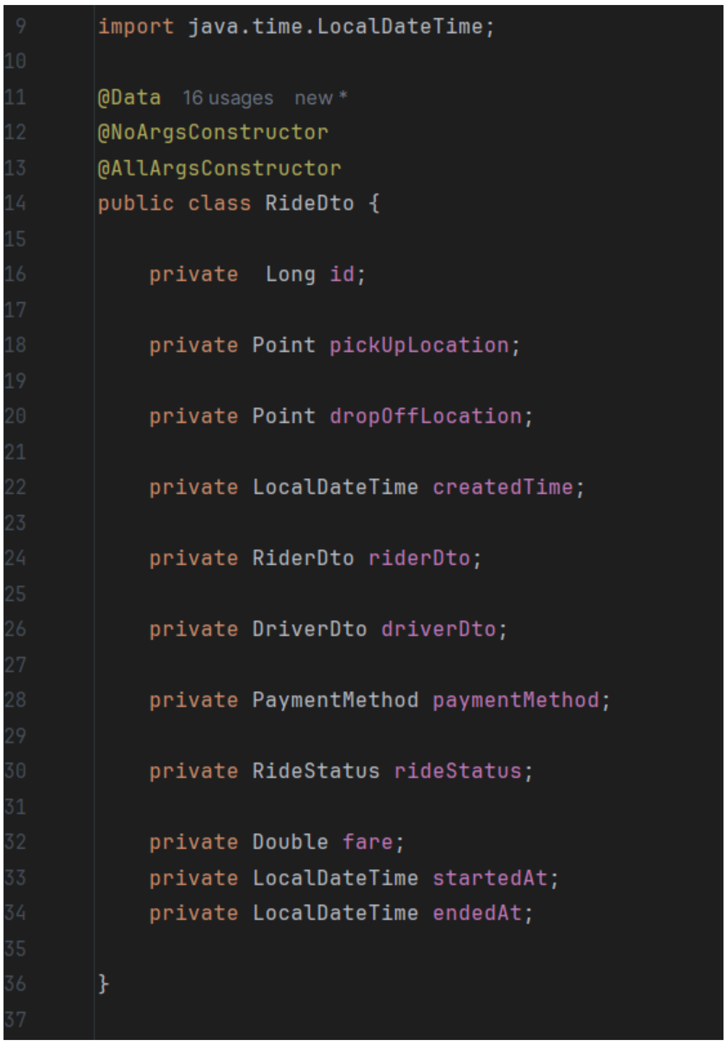
```java
package com.yasif.project.uber.Uber.backend.system.dto;

import com.yasif.project.uber.Uber.backend.system.entities.enums.R
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.Set;

@Data  6 usages  new *
@NoArgsConstructor
@AllArgsConstructor
public class UserDto {

    private String name;
    private String email;
    private Set<Role> roles;

}
```



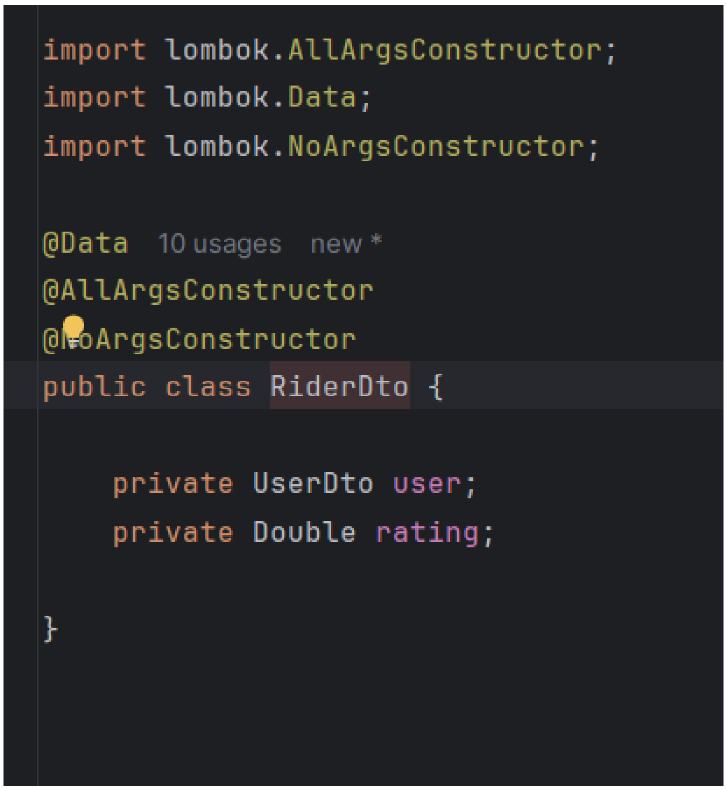
```java
package com.yasif.project.uber.Uber.backend.system.dto;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data  4 usages  new *
@NoArgsConstructor
@AllArgsConstructor
public class SignupDto {

    private String name;
    private String email;
    private String password;


}
```



```java
package com.yasif.project.uber.Uber.backend.system.dto;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data  15 usages  new *
@AllArgsConstructor
@NoArgsConstructor
public class DriverDto {

    private UserDto user;
    private Double rating;

}
```

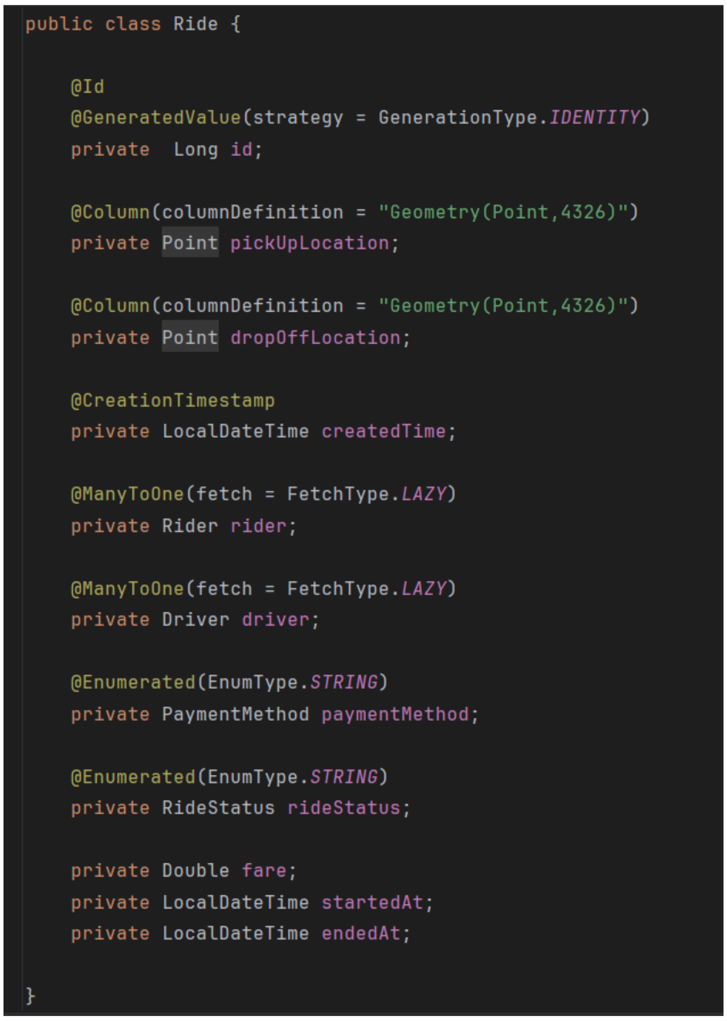


```java
import java.time.LocalDateTime;

@Data  16 usages  new *
@NoArgsConstructor
@AllArgsConstructor
public class RideDto {

    private  Long id;

    private Point pickUpLocation;

    private Point dropOffLocation;

    private LocalDateTime createdTime;

    private RiderDto riderDto;

    private DriverDto driverDto;

    private PaymentMethod paymentMethod;

    private RideStatus rideStatus;

    private Double fare;
    private LocalDateTime startedAt;
    private LocalDateTime endedAt;

}
```

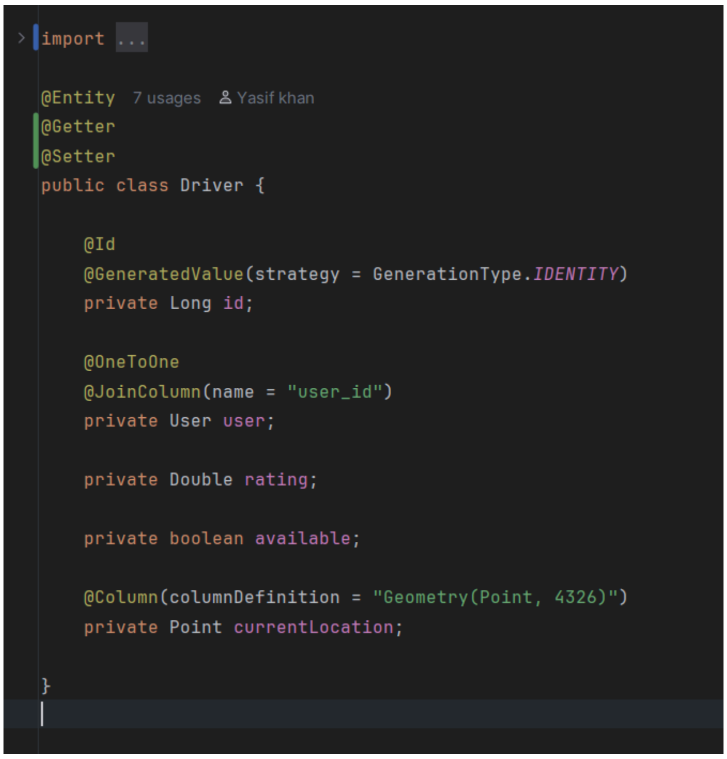


```java
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data   10 usages   new *
@AllArgsConstructor
@NoArgsConstructor
public class RiderDto {

    private UserDto user;
    private Double rating;

}
```

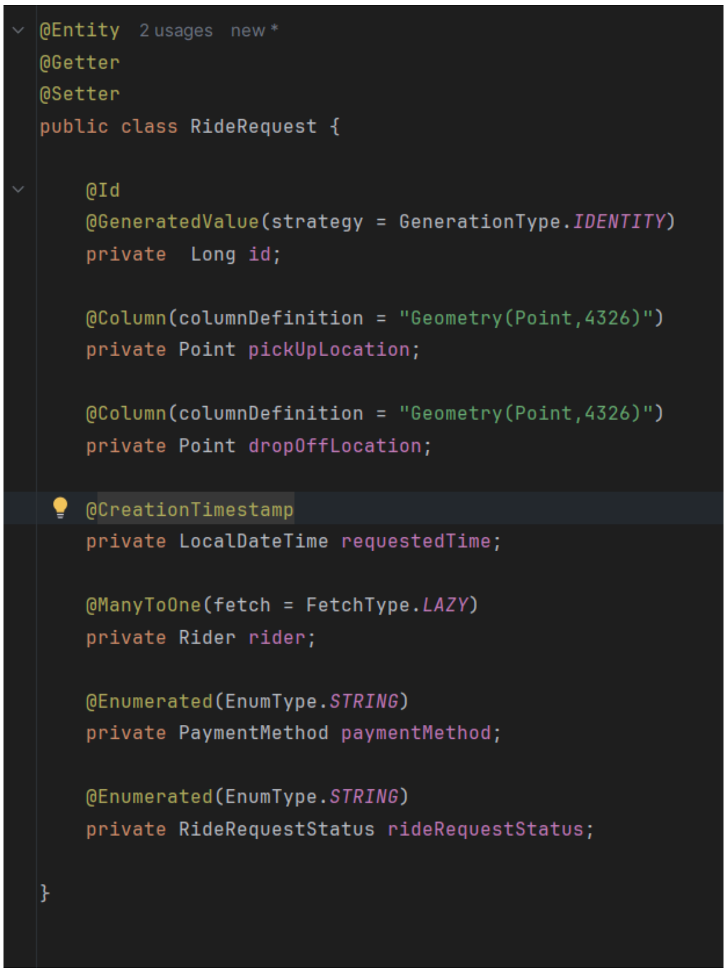
## Entities



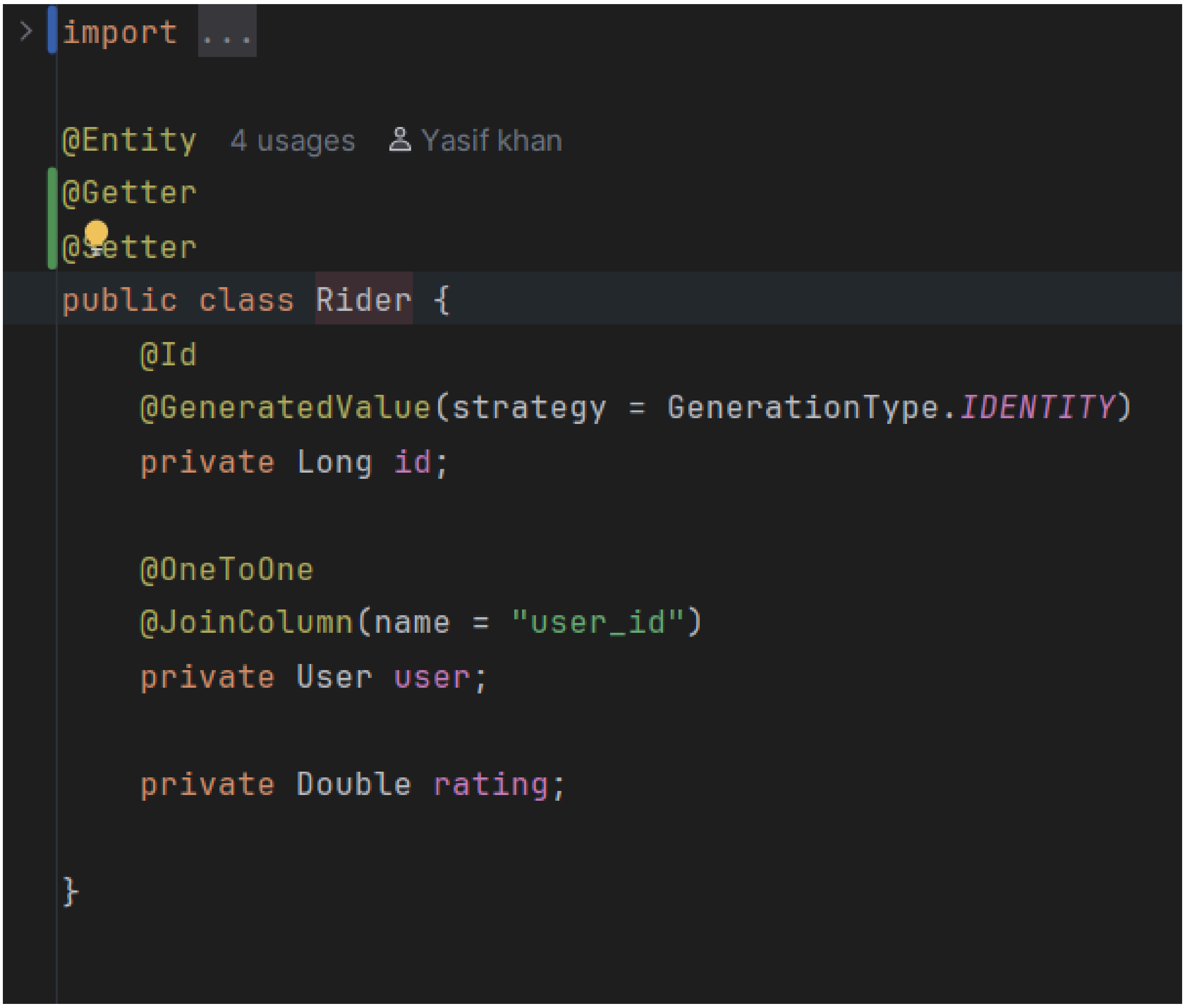
```java
public class Ride {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private  Long id;

    @Column(columnDefinition = "Geometry(Point,4326)")
    private Point pickUpLocation;

    @Column(columnDefinition = "Geometry(Point,4326)")
    private Point dropOffLocation;

    @CreationTimestamp
    private LocalDateTime createdTime;

    @ManyToOne(fetch = FetchType.LAZY)
    private Rider rider;

    @ManyToOne(fetch = FetchType.LAZY)
    private Driver driver;

    @Enumerated(EnumType.STRING)
    private PaymentMethod paymentMethod;

    @Enumerated(EnumType.STRING)
    private RideStatus rideStatus;

    private Double fare;
    private LocalDateTime startedAt;
    private LocalDateTime endedAt;

}
```



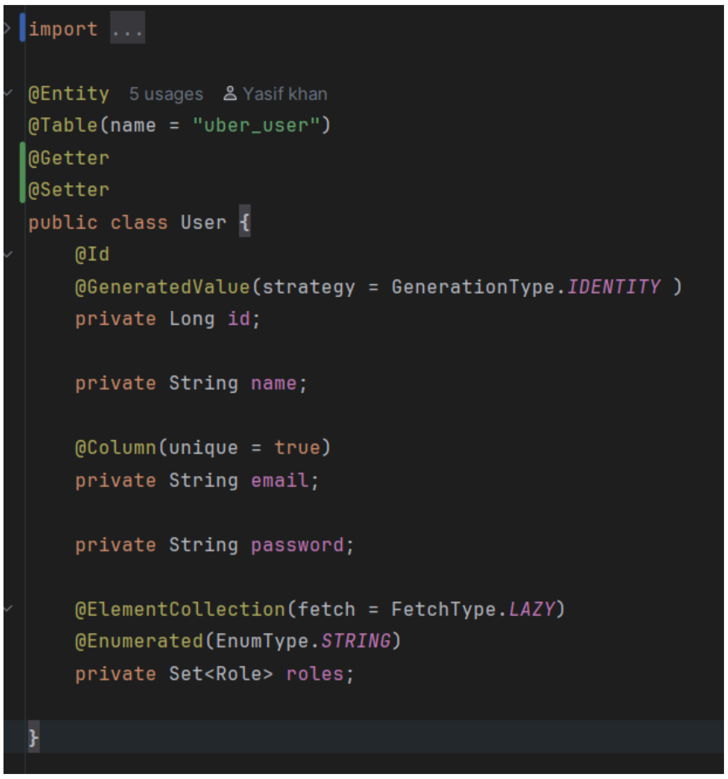
```
import ...

@Entity  7 usages  Yasif khan
@Getter
@Setter
public class Driver {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @OneToOne
    @JoinColumn(name = "user_id")
    private User user;

    private Double rating;

    private boolean available;

    @Column(columnDefinition = "Geometry(Point, 4326)")
    private Point currentLocation;

}
```



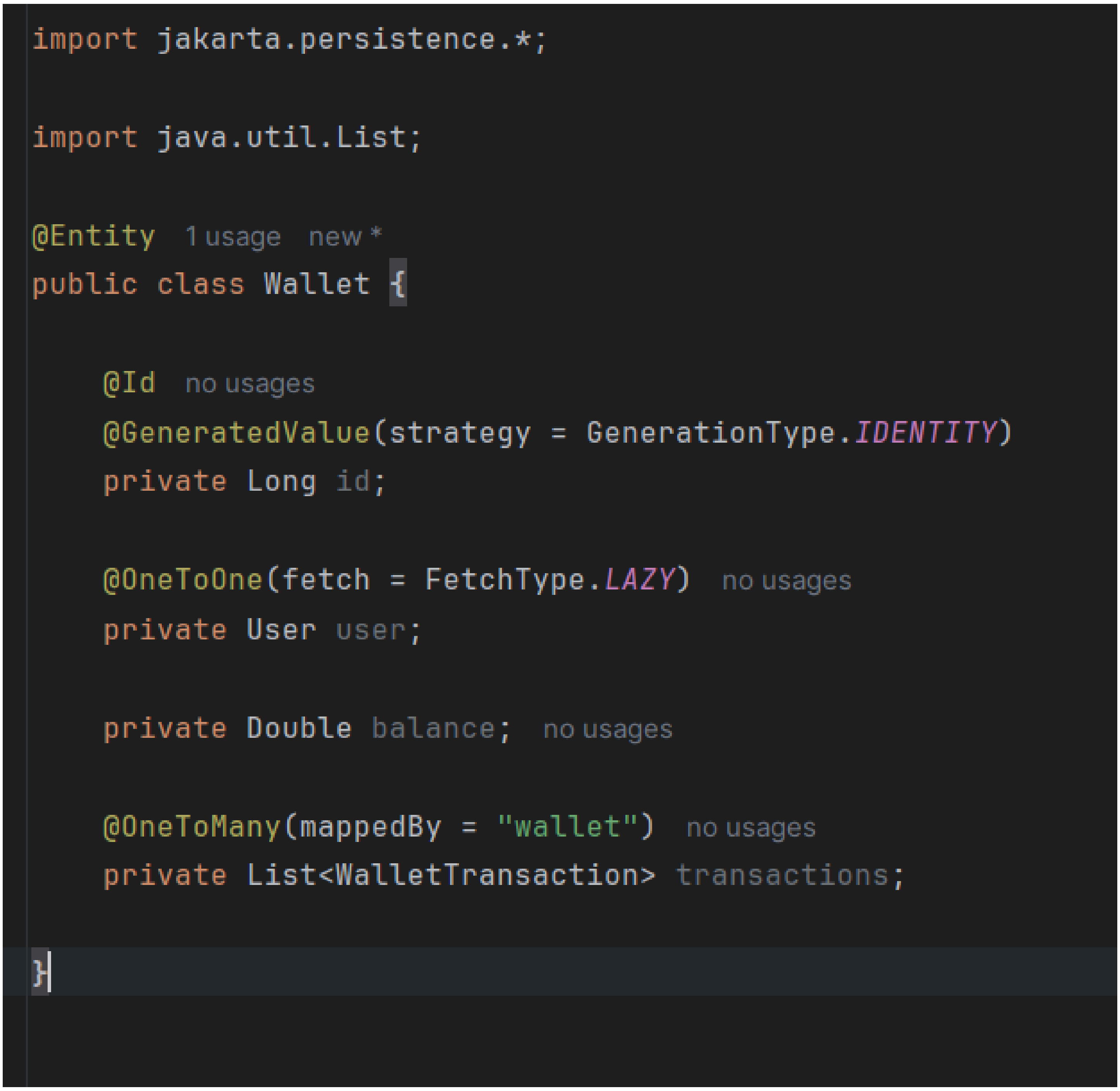
```
@Entity  2 usages  new *
@Getter
@Setter
public class RideRequest {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private  Long id;

    @Column(columnDefinition = "Geometry(Point,4326)")
    private Point pickUpLocation;

    @Column(columnDefinition = "Geometry(Point,4326)")
    private Point dropOffLocation;

    @CreationTimestamp
    private LocalDateTime requestedTime;

    @ManyToOne(fetch = FetchType.LAZY)
    private Rider rider;

    @Enumerated(EnumType.STRING)
    private PaymentMethod paymentMethod;

    @Enumerated(EnumType.STRING)
    private RideRequestStatus rideRequestStatus;

}
```



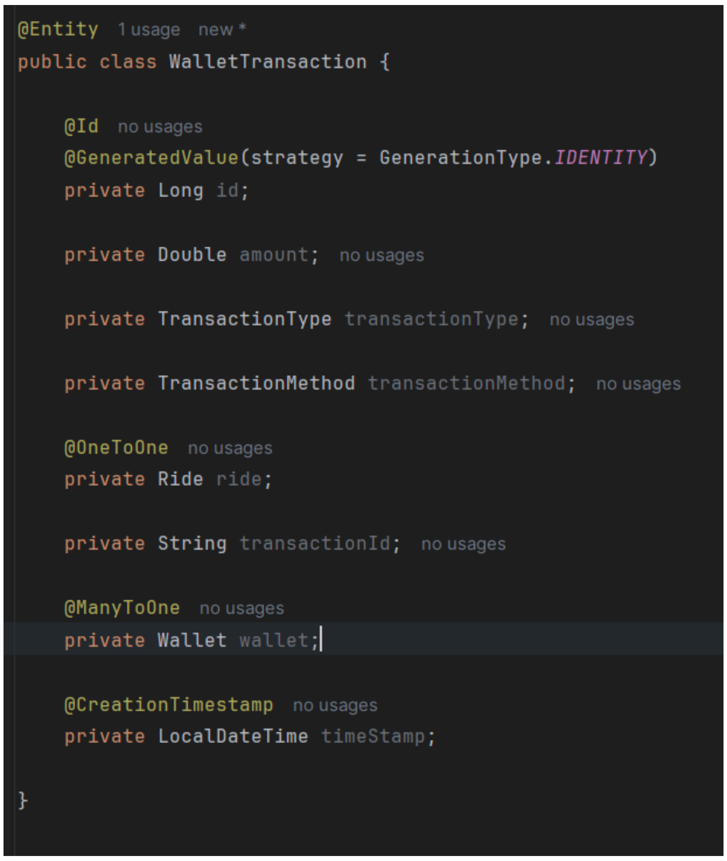
```
import ...

@Entity  4 usages  Yasif khan
@Getter
@Setter
public class Rider {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @OneToOne
    @JoinColumn(name = "user_id")
    private User user;

    private Double rating;

}
```



```
import ...

@Entity  5 usages  Yasif khan
@Table(name = "uber_user")
@Getter
@Setter
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY )
    private Long id;

    private String name;

    @Column(unique = true)
    private String email;

    private String password;

    @ElementCollection(fetch = FetchType.LAZY)
    @Enumerated(EnumType.STRING)
    private Set<Role> roles;

}
```

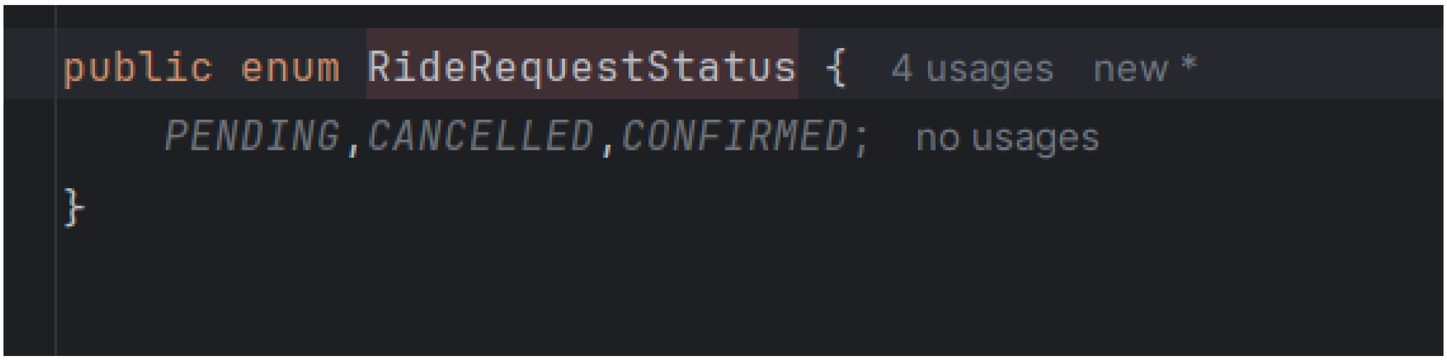


```
import jakarta.persistence.*;

import java.util.List;

@Entity  1 usage  new *
public class Wallet {

    @Id  no usages
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @OneToOne(fetch = FetchType.LAZY)  no usages
    private User user;

    private Double balance;  no usages

    @OneToMany(mappedBy = "wallet")  no usages
    private List<WalletTransaction> transactions;

}
```

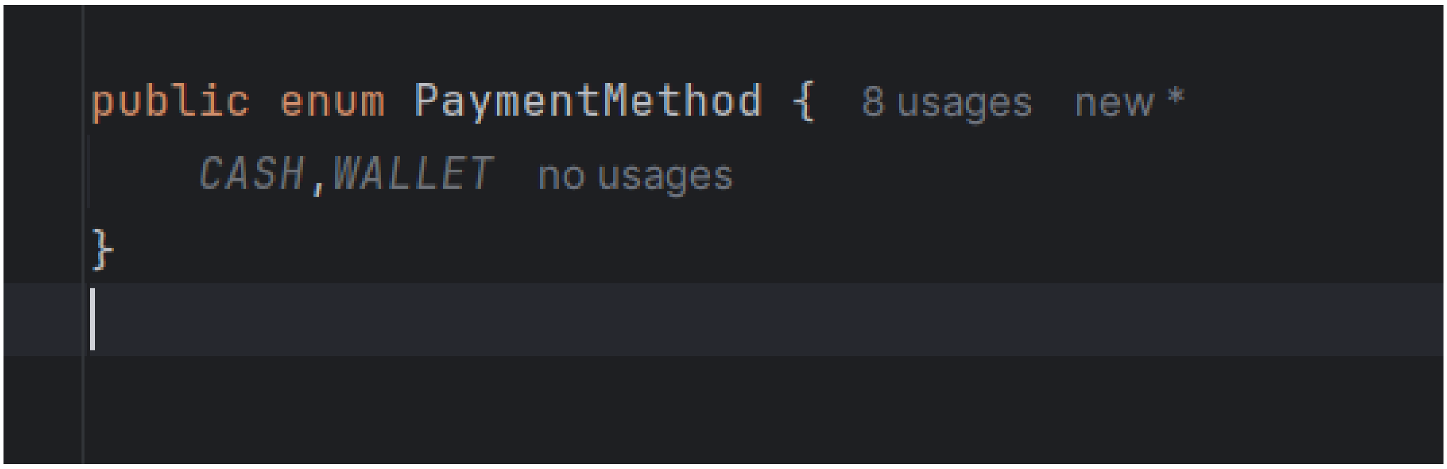


```
@Entity  1 usage  new *
public class WalletTransaction {

    @Id  no usages
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private Double amount;  no usages

    private TransactionType transactionType;  no usages

    private TransactionMethod transactionMethod;  no usages

    @OneToOne  no usages
    private Ride ride;

    private String transactionId;  no usages

    @ManyToOne  no usages
    private Wallet wallet;

    @CreationTimestamp  no usages
    private LocalDateTime timeStamp;

}
```

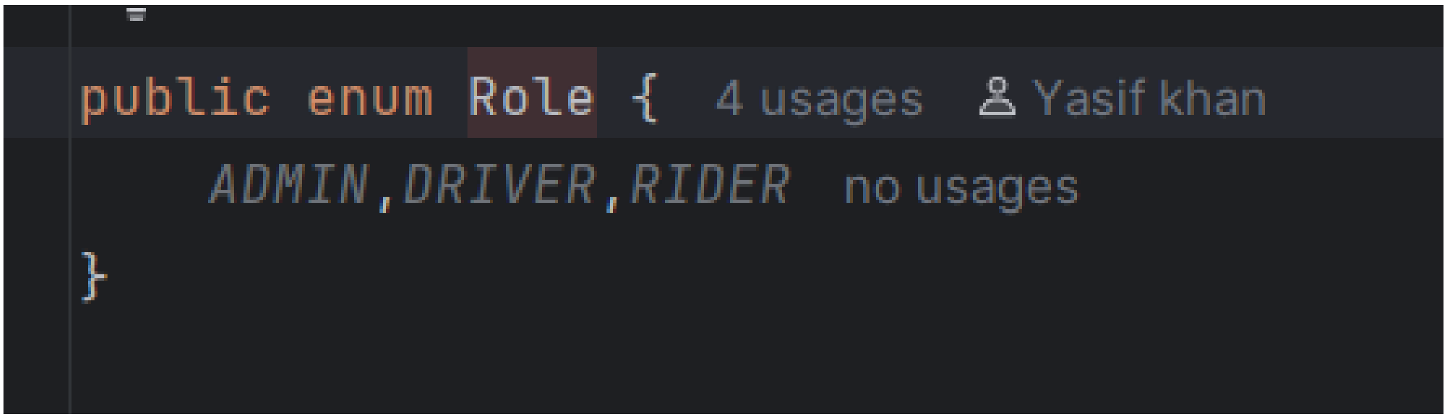
## Enums



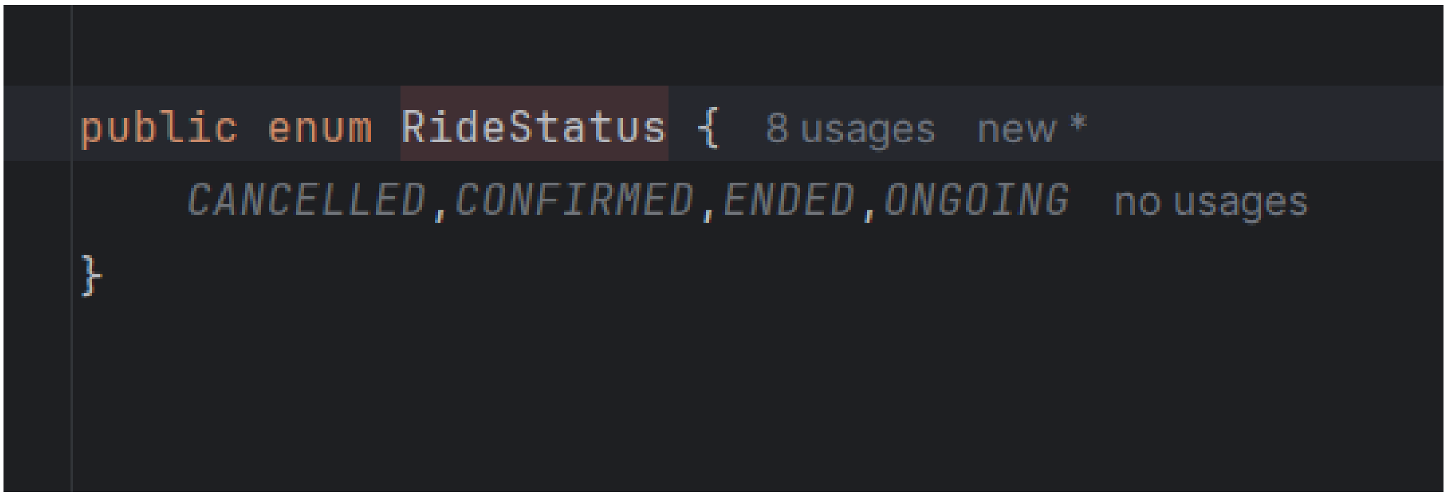
```
public enum RideRequestStatus {  4 usages  new *
    PENDING,CANCELLED,CONFIRMED;  no usages
}
```



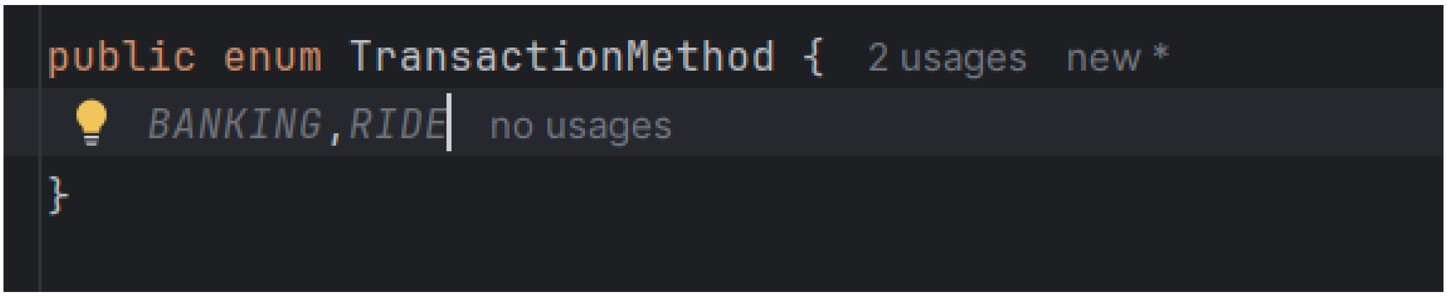
```
public enum PaymentMethod {  8 usages  new *
    CASH,WALLET  no usages
}
```



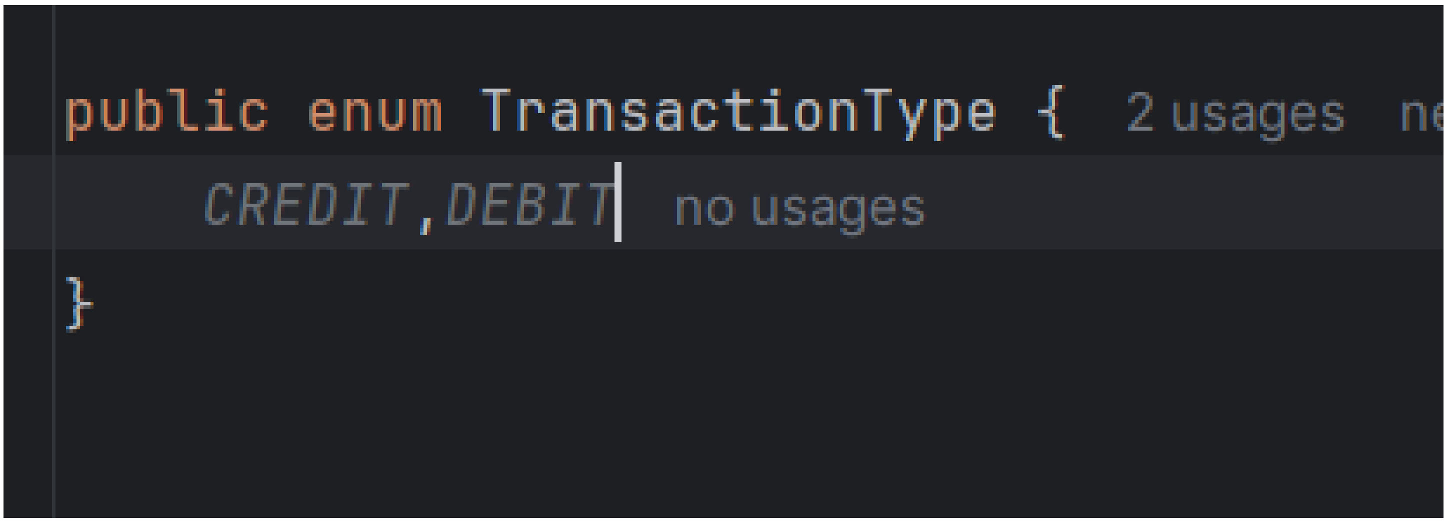
```
public enum Role {  4 usages   Yasif khan
    ADMIN,DRIVER,RIDER  no usages
}
```



```
public enum RideStatus {  8 usages  new *
    CANCELLED,CONFIRMED,ENDED,ONGOING  no usages
}
```

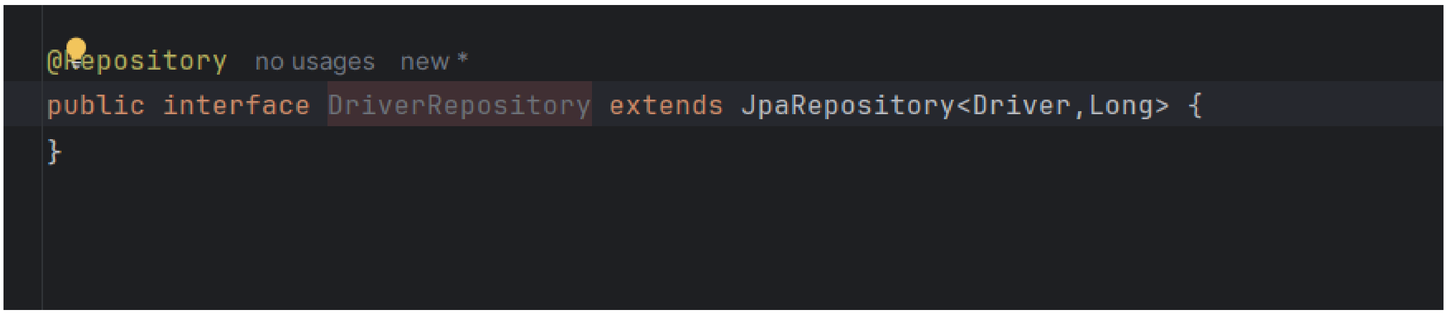


```
public enum TransactionMethod {  2 usages  new *
    BANKING,RIDE  no usages
}
```

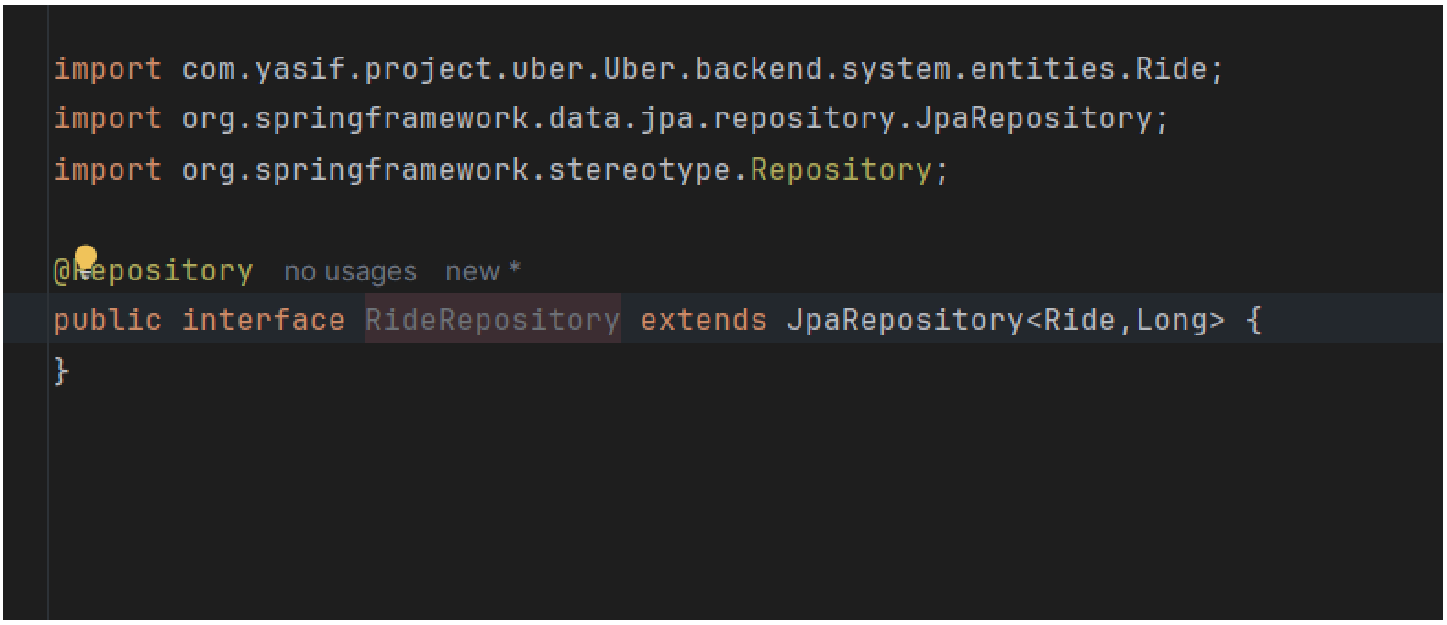


```
public enum TransactionType {  2 usages  ne
    CREDIT,DEBIT  no usages
}
```

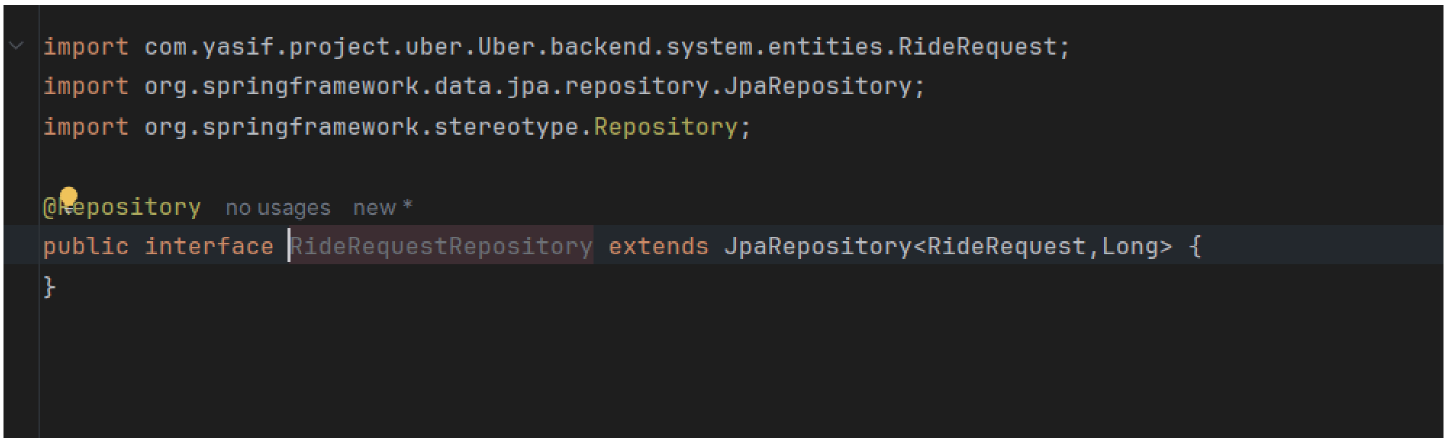
## Repositories



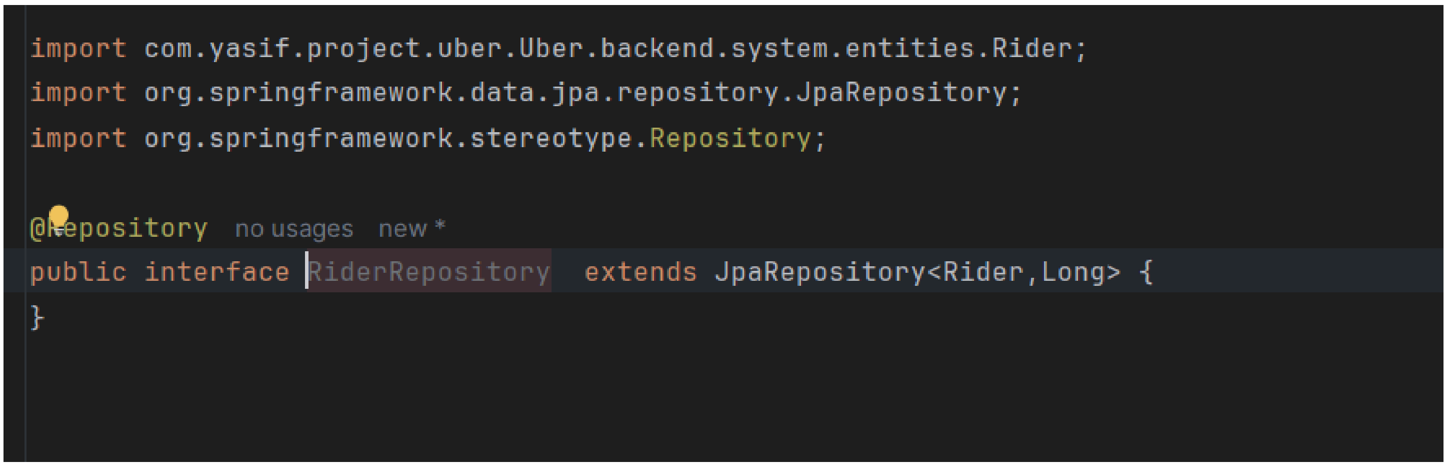
```java
@Repository  no usages  new *
public interface DriverRepository extends JpaRepository<Driver,Long> {
}
```



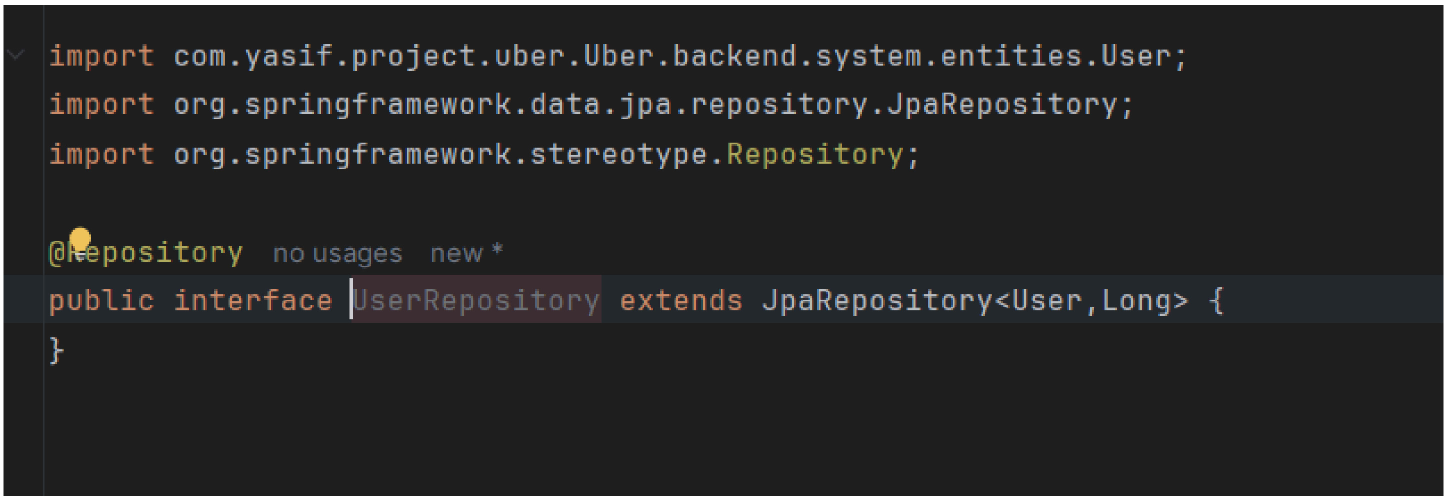
```java
import com.yasif.project.uber.Uber.backend.system.entities.Ride;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository  no usages  new *
public interface RideRepository extends JpaRepository<Ride,Long> {
}
```



```java
import com.yasif.project.uber.Uber.backend.system.entities.RideRequest;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository  no usages  new *
public interface RideRequestRepository extends JpaRepository<RideRequest,Long> {
}
```

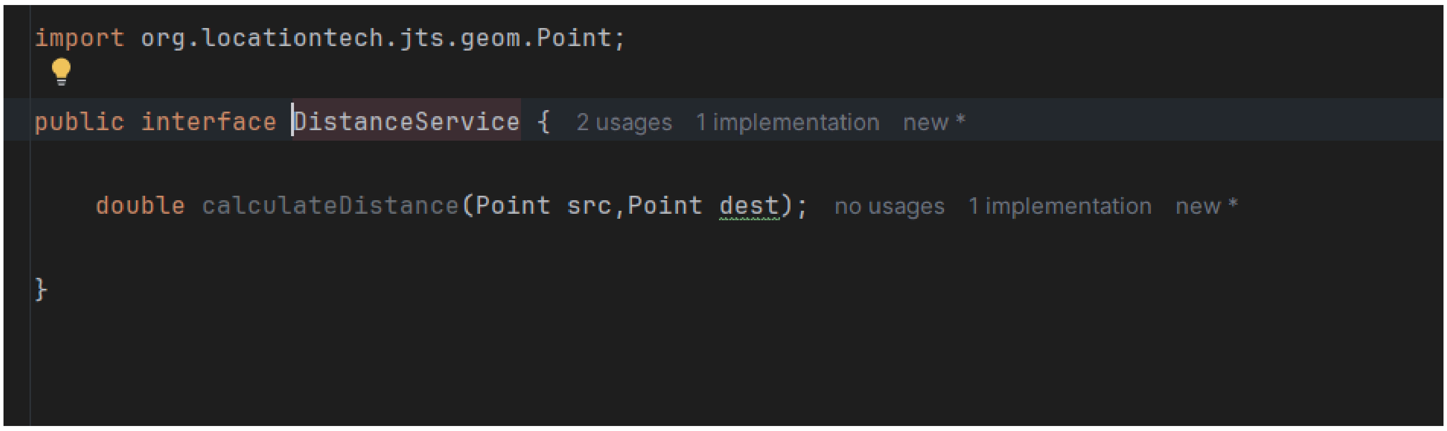


```java
import com.yasif.project.uber.Uber.backend.system.entities.Rider;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository  no usages  new *
public interface RiderRepository  extends JpaRepository<Rider,Long> {
}
```

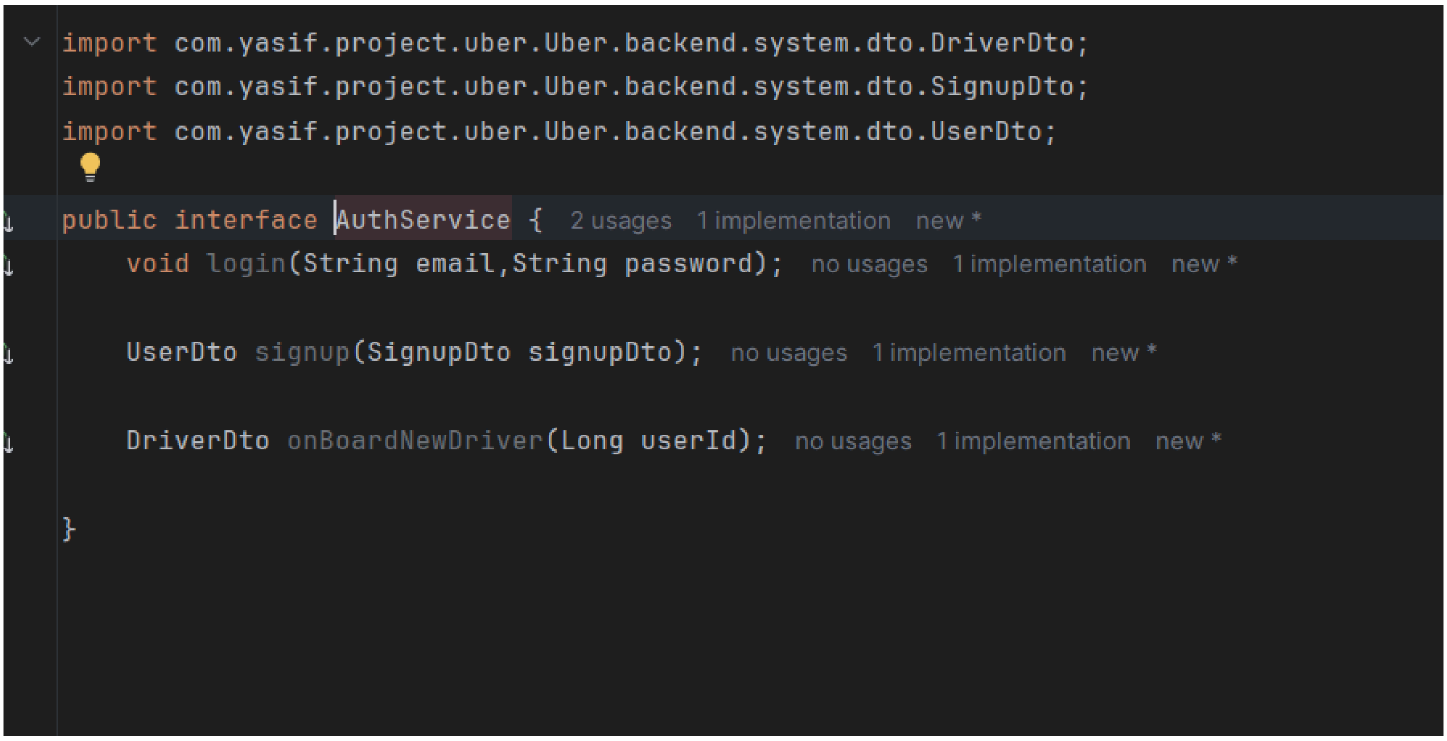


```
import com.yasif.project.uber.Uber.backend.system.entities.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository  no usages   new *
public interface UserRepository extends JpaRepository<User,Long> {
}
```

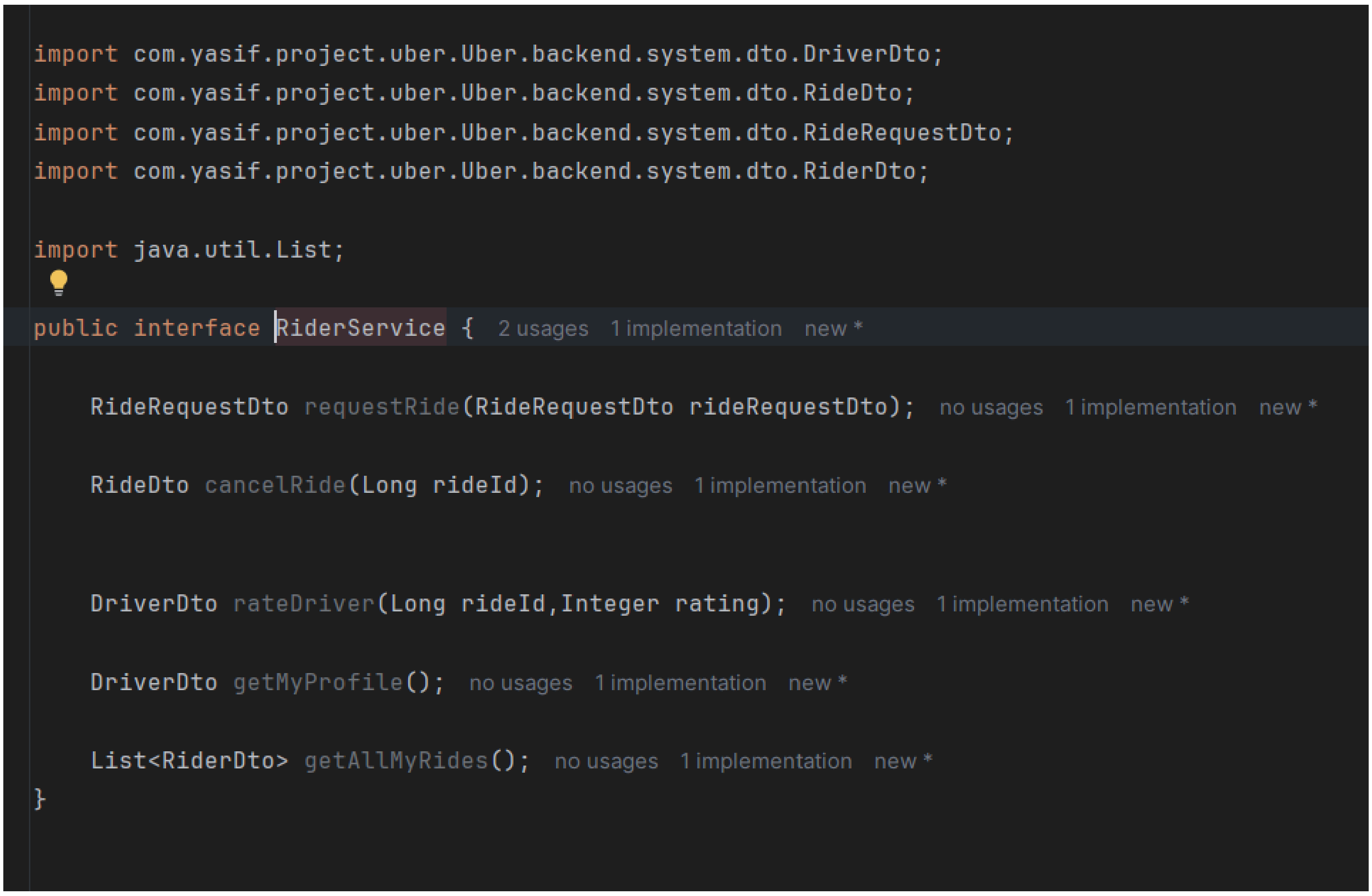
## Services



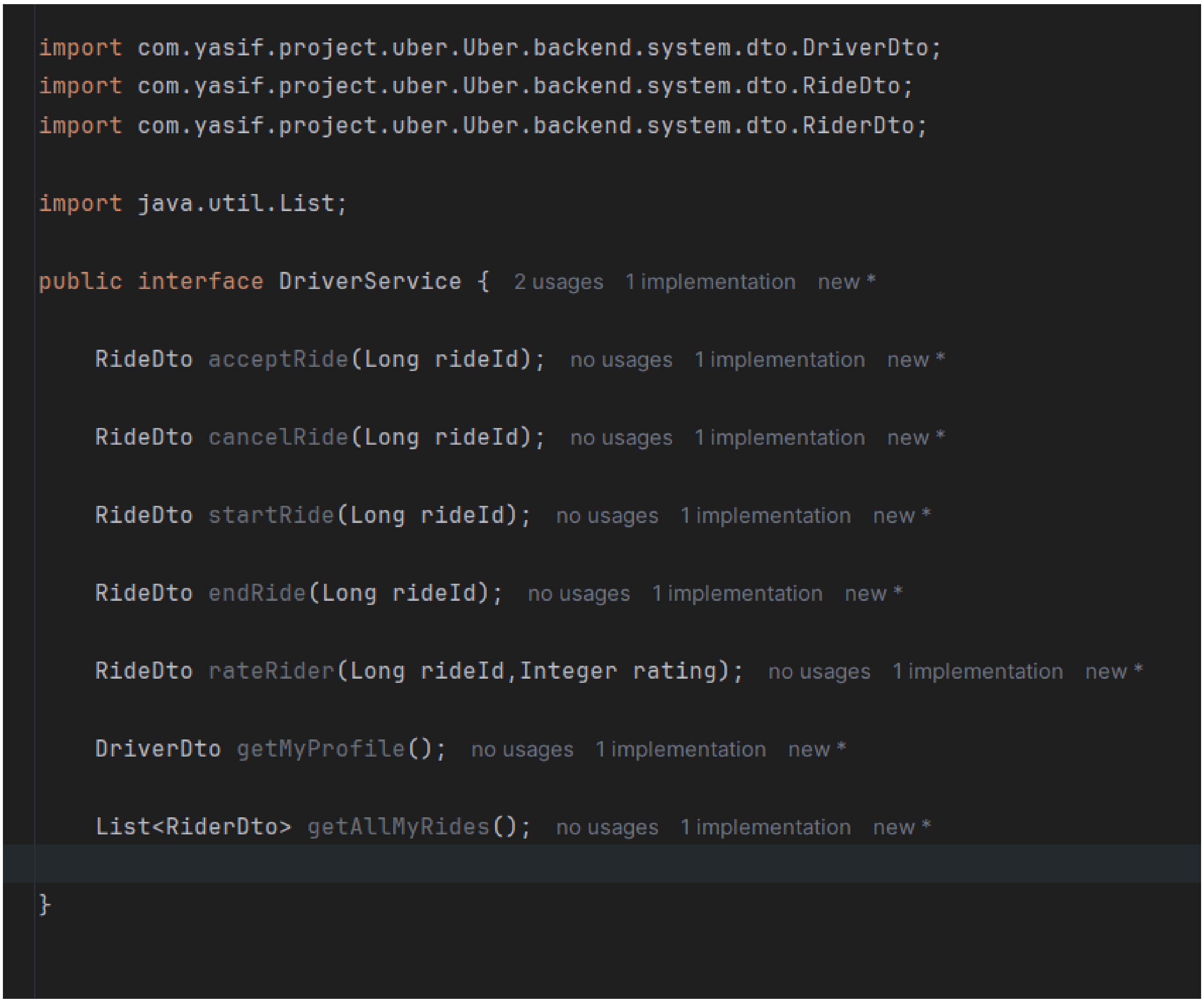
```
import org.locationtech.jts.geom.Point;

public interface DistanceService {  2 usages   1 implementation   new *

    double calculateDistance(Point src,Point dest);  no usages   1 implementation   new *

}
```



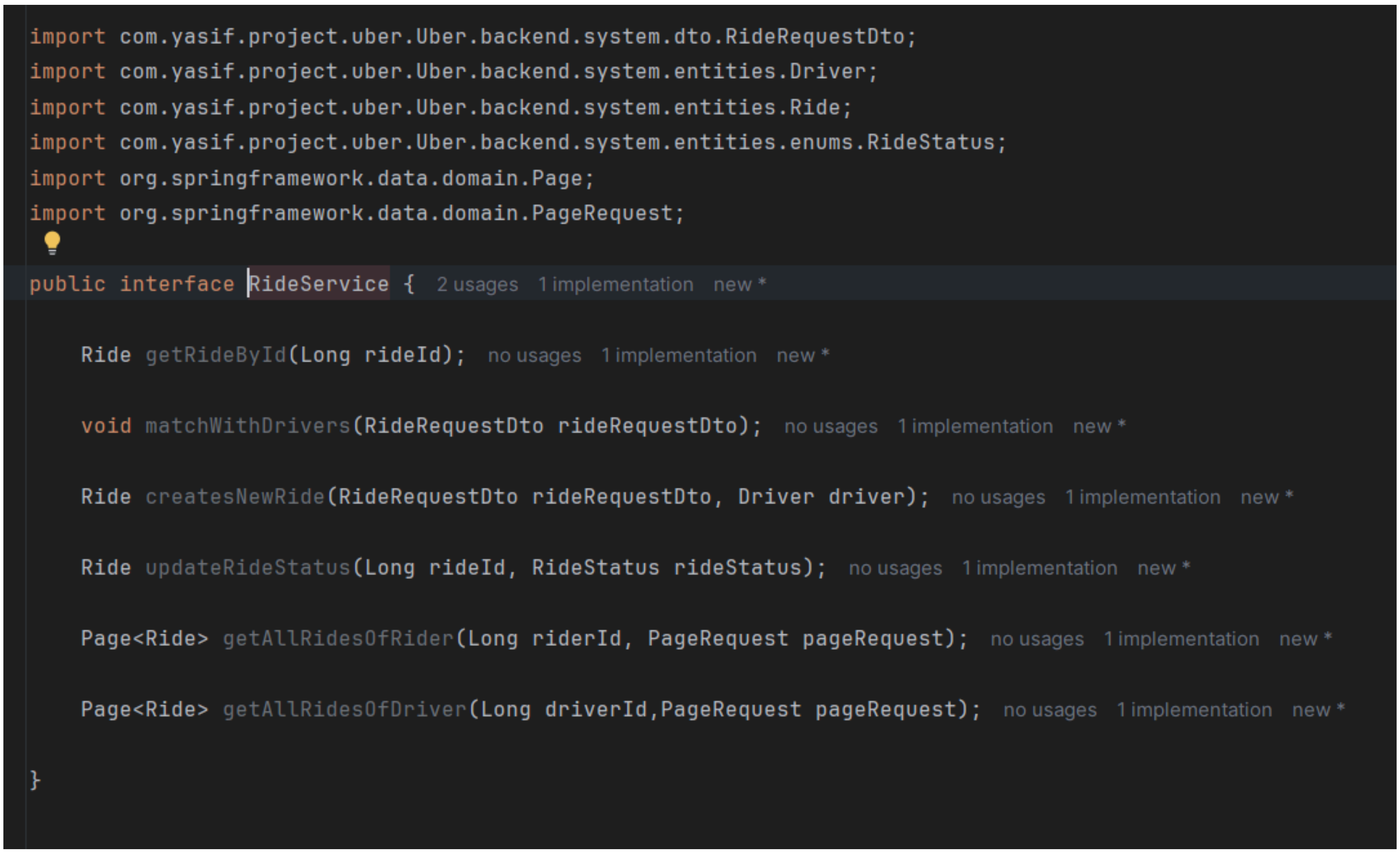
```
import com.yasif.project.uber.Uber.backend.system.dto.DriverDto;
import com.yasif.project.uber.Uber.backend.system.dto.SignupDto;
import com.yasif.project.uber.Uber.backend.system.dto.UserDto;

public interface AuthService {  2 usages   1 implementation   new *
    void login(String email,String password);  no usages   1 implementation   new *

    UserDto signup(SignupDto signupDto);  no usages   1 implementation   new *

    DriverDto onBoardNewDriver(Long userId);  no usages   1 implementation   new *

}
```



```java
import com.yasif.project.uber.Uber.backend.system.dto.DriverDto;
import com.yasif.project.uber.Uber.backend.system.dto.RideDto;
import com.yasif.project.uber.Uber.backend.system.dto.RideRequestDto;
import com.yasif.project.uber.Uber.backend.system.dto.RiderDto;

import java.util.List;

public interface RiderService {  2 usages   1 implementation   new *

    RideRequestDto requestRide(RideRequestDto rideRequestDto);  no usages   1 implementation   new *

    RideDto cancelRide(Long rideId);  no usages   1 implementation   new *


    DriverDto rateDriver(Long rideId,Integer rating);  no usages   1 implementation   new *

    DriverDto getMyProfile();  no usages   1 implementation   new *

    List<RiderDto> getAllMyRides();  no usages   1 implementation   new *
}
```



```java
import com.yasif.project.uber.Uber.backend.system.dto.DriverDto;
import com.yasif.project.uber.Uber.backend.system.dto.RideDto;
import com.yasif.project.uber.Uber.backend.system.dto.RiderDto;

import java.util.List;

public interface DriverService {  2 usages  1 implementation  new *

    RideDto acceptRide(Long rideId);  no usages  1 implementation  new *

    RideDto cancelRide(Long rideId);  no usages  1 implementation  new *

    RideDto startRide(Long rideId);  no usages  1 implementation  new *

    RideDto endRide(Long rideId);  no usages  1 implementation  new *

    RideDto rateRider(Long rideId,Integer rating);  no usages  1 implementation  new *

    DriverDto getMyProfile();  no usages  1 implementation  new *

    List<RiderDto> getAllMyRides();  no usages  1 implementation  new *

}
```



```java
import com.yasif.project.uber.Uber.backend.system.dto.RideRequestDto;
import com.yasif.project.uber.Uber.backend.system.entities.Driver;
import com.yasif.project.uber.Uber.backend.system.entities.Ride;
import com.yasif.project.uber.Uber.backend.system.entities.enums.RideStatus;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;

public interface RideService {  2 usages  1 implementation  new *

    Ride getRideById(Long rideId);  no usages  1 implementation  new *

    void matchWithDrivers(RideRequestDto rideRequestDto);  no usages  1 implementation  new *

    Ride createsNewRide(RideRequestDto rideRequestDto, Driver driver);  no usages  1 implementation  new *

    Ride updateRideStatus(Long rideId, RideStatus rideStatus);  no usages  1 implementation  new *

    Page<Ride> getAllRidesOfRider(Long riderId, PageRequest pageRequest);  no usages  1 implementation  new *

    Page<Ride> getAllRidesOfDriver(Long driverId,PageRequest pageRequest);  no usages  1 implementation  new *

}
```