

## Refactoring the JwtService

In this exercise, you'll refactor the `JwtService` to take a more object-oriented approach.

OOP encourages bundling data and behavior, which leads to:

- Cleaner code
- Easier testing
- Better encapsulation
- More flexibility as the system evolves

### What You Need to Do

- Create a `Jwt` class that encapsulates token-related behavior.
- Refactor the `JwtService` to:
  - Return a `Jwt` object
  - Only expose high-level responsibilities like generating and parsing tokens.

JwtService
+ generateAccessToken(): Jwt + generateRefreshToken(): Jwt + parse(token: String): Jwt

Jwt
- claims: Claims - key: SecretKey
+ isValid(): boolean + getUserId(): Long + getRole(): Role + toString(): String