SOLI**D**

Dependency Inversion Principle

A deeper look with examples

History

- Around since early 90s
- Postulated by Robert Martin (aka Uncle Bob) 1994 in an article called Object Oriented Design Quality Metrics

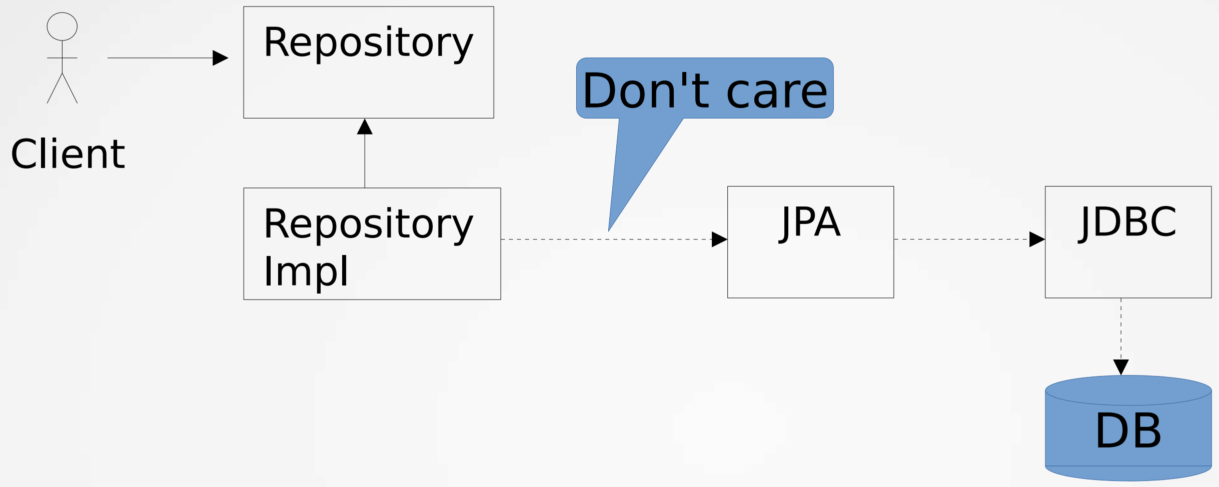
Key points

- High-level modules should not depend on low-level modules. Both should depend on abstractions
- Abstractions should not depend on details. Details should depend on abstractions
- → Use abstraction between high-level and low-level modules
- → see the following samples

Database access

- Evolution in DB technologies
  - JDBC
  - ORM/Hibernate
  - JPA
- Using these interfaces is violation of DIP
- DB Access on lower level than domain/business
- Complex interfaces
- They offer methods you will never use

Solution – Hide DB



- Domain related solution
- Use methods related to domain not to persistance layer
- e.g.: add(Person), findAllPersons(name)

Flexibility is costly

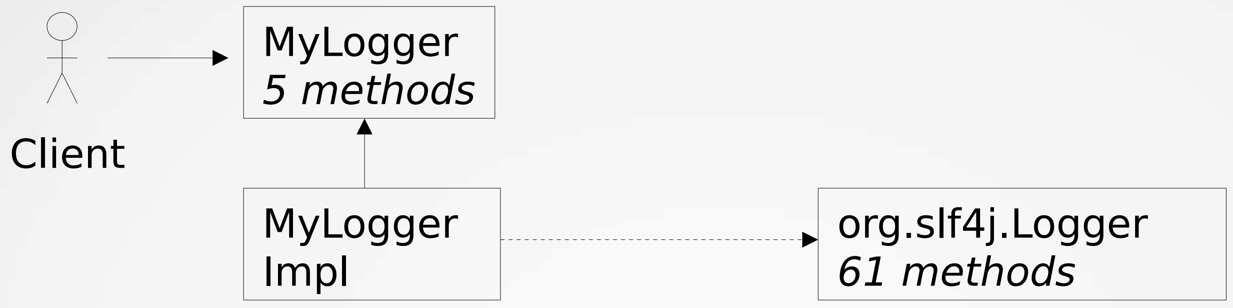
- Apache log4j Logger
- > 60 methods
- Which one do I use
- Which ones does the team use
- What do I / the team log on what level
- To much ↔ not enough information
- Consistant use is hard

Logger - Performance

-
  ```
  Logger logger = Logger.getLogger(getClass().getName();
  String message = String.format("Read of user: %s", user.getName());
  logger.log(Level.INFO, message);
  ```
- => String concatenation

-
  ```
  Logger logger = Logger.getLogger(getClass().getName();
  if (logger.isLoggable(Level.INFO) {
     String message = String.format("Read of user: %s", user.getName());
     logger.log(Level.INFO, message);
  }
  ```
- → decrease readability
  → discipline required
  → violates DRY, ...

Solution - gateway



- => more consistency
- => reduced set of relevant method
- => less discipline required
- => No DRY

-
  ```
  MyLogger logger = SystemLoggerFactory.get(getClass());
  logger.info("Read of user: %s", user.getName());
  ```

Advantages using DIP

- tame unwieldy APIs
- reduce complexity of interfaces
- remove mismatch between abstraction level of library and domain
- get better testing abilities
- reduce coupling

References

- Object Oriented Design Quality Metrics
  (1994 by Robert C. Martin)
- The Dependency Inversion Principle
  (1996 by Robert C. Martin)
- DIP in the Wild
  (2013 by Brett L. Schuchert)

Thanks for listening