Presentation day Running

The Logistics hub

Introduction

- Bouvet developers may have heard about it
- A horrible monster that wear out developers
- Cost a lot of money
- Application for storing container data
- Tracking containers being shipped around
- It now dead
- Why?
  - It worked just fine
  - Nobody did use it
  - There was better on shelf software available

What was the logitics hub?

- Tracking solution for shipping containers
- Java application
- Oracle DB
- LDAP User directory
- Started development over 10 year ago.
- 1155 Java classes
- 79974 Lines of java code
- 2075 Unit tests (and some)
- And a LOT of stuff that is not java code.

What went wrong

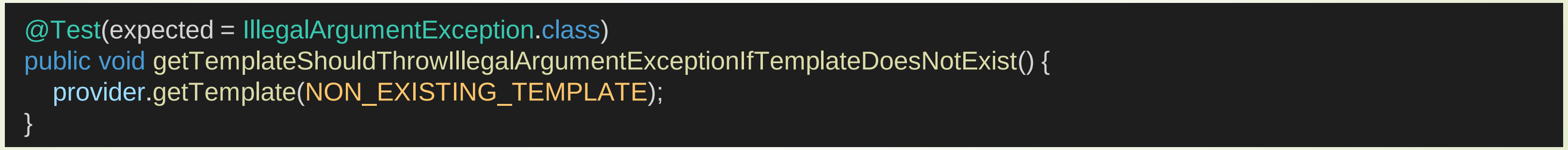
- Development was started by another company
- No time to do a proper handover when Bouvet got it
- Many developers involved
- Unclear specification
- Lots of changes
- Time pressure!!!

Why am i showing this?

- To show
  - Code smells
  - Violation of Object Calistenics rules
  - Bad pratices
- Will focus on the Java backend part

Unit tests

- Hundreds of tests failing because they rely on LDAP user directory.
- Unit test fail beacuse they are missing the SMTP server.
- Integration test inside unit test project. Failing beause the database is missing (By the way: There is a integration test project also)
- And a lot of tests fail because they rely on precompiled code. (java classes are generated from xsd files)
- Tests expect exceptions to be throwed



```
@Test(expected = IllegalArgumentException.class)
public void getTemplateShouldThrowIllegalArgumentExceptionIfTemplateDoesNotExist() {
    provider.getTemplate(NON_EXISTING_TEMPLATE);
}
```

What is the test smells

- Tests should be independent and small
- Exception swallowing
- Unclear failing reason
- There extremely many excepetion thown upon compile

Code smell: Large class

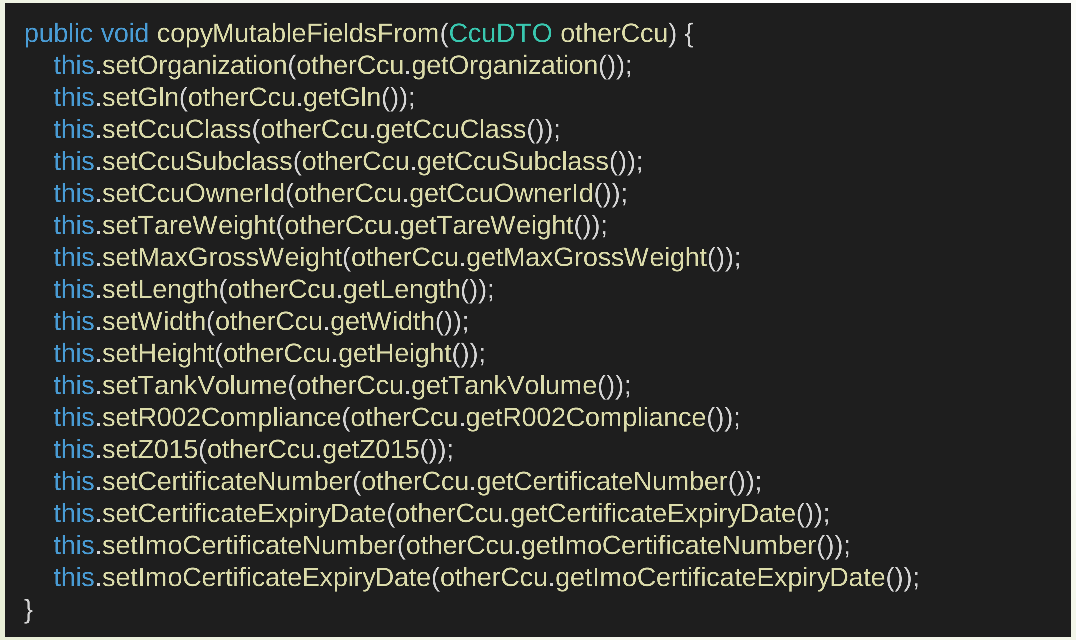
- The LdapSource class
- 1285 lines of code
- 74 Methods
- How to fix?
- @SuppressWarnings({ "PMD.GodClass", "PMD.TooManyMethods", "PMD.ExcessiveClassLength" })
- And another 17 classes that needs @SuppressWarnings PMD.GodClass.

Code smell: Man in the middle and Message Chain

- A request coming for the controller needs
- Ccu.Resource.getCcus ->
- QueryHandler.getCcu ->
- RelationalApplicationServer.getCcu ->
- CcuDAO.GetCcu ->
- CcuRepository.findByExtCcuId
- Finally This is inherited from Spring.CrudRepository.
- This long tree makes the application difficult to debug and hard to change.
- A change will be a shotgun surgery as all these classes needs to be updated with the change.

Long parameter list

- These long parmeter list are hard to read.
- Compiler is not very happy:
- @SuppressWarnings({ "PMD.ExcessivePublicCount", "PMD.GodClass", "PMD.TooManyFields", "PMD.ExcessiveParameterList" })

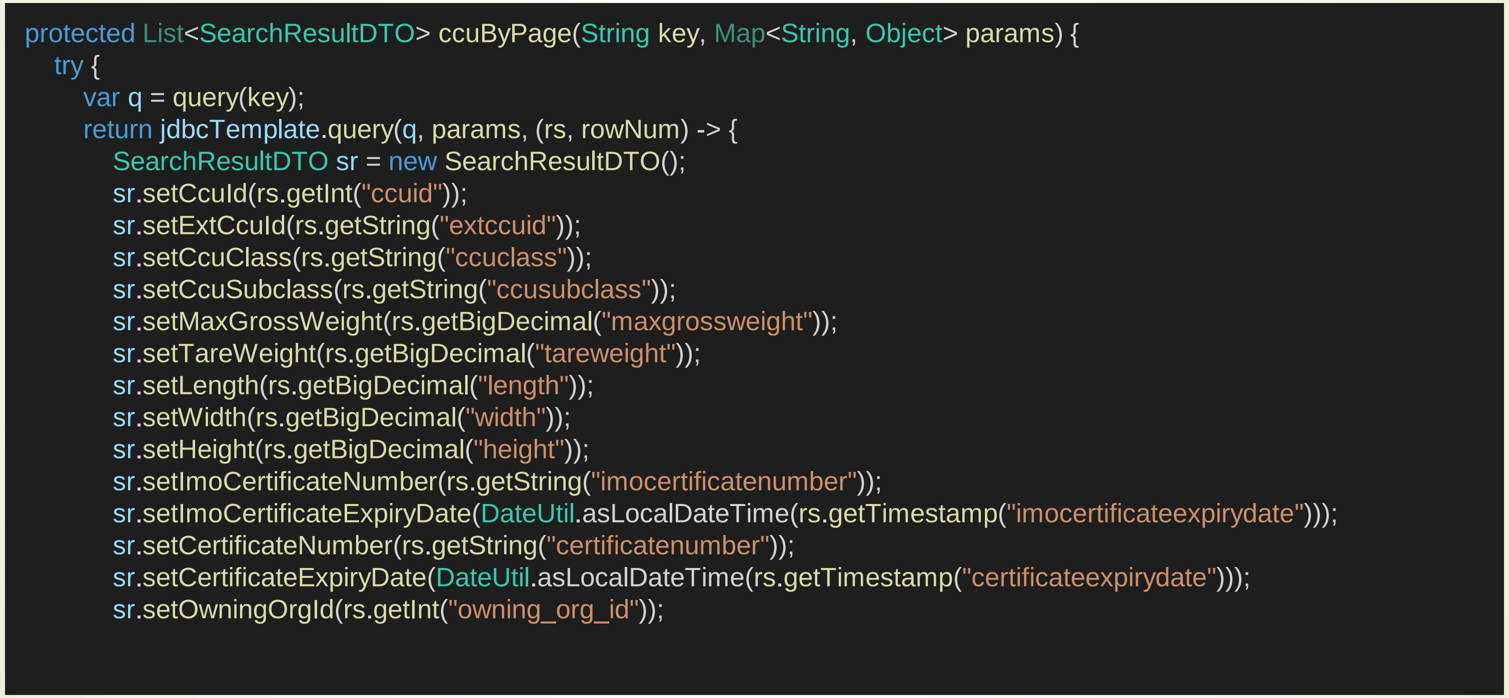


```java
public void copyMutableFieldsFrom(CcuDTO otherCcu) {
    this.setOrganization(otherCcu.getOrganization());
    this.setGln(otherCcu.getGln());
    this.setCcuClass(otherCcu.getCcuClass());
    this.setCcuSubclass(otherCcu.getCcuSubclass());
    this.setCcuOwnerId(otherCcu.getCcuOwnerId());
    this.setTareWeight(otherCcu.getTareWeight());
    this.setMaxGrossWeight(otherCcu.getMaxGrossWeight());
    this.setLength(otherCcu.getLength());
    this.setWidth(otherCcu.getWidth());
    this.setHeight(otherCcu.getHeight());
    this.setTankVolume(otherCcu.getTankVolume());
    this.setR002Compliance(otherCcu.getR002Compliance());
    this.setZ015(otherCcu.getZ015());
    this.setCertificateNumber(otherCcu.getCertificateNumber());
    this.setCertificateExpiryDate(otherCcu.getCertificateExpiryDate());
    this.setImoCertificateNumber(otherCcu.getImoCertificateNumber());
    this.setImoCertificateExpiryDate(otherCcu.getImoCertificateExpiryDate());
}
```

Duplicated Code

- The abstract server was copied to make a new webapi.
- The new webapi-server does much of the same as the abstract server
- A changed in the API or data structure need to be changed in both servers.
- Why?
- Time pressure from the customer.
- By the way. The abstract server is not very abstract.

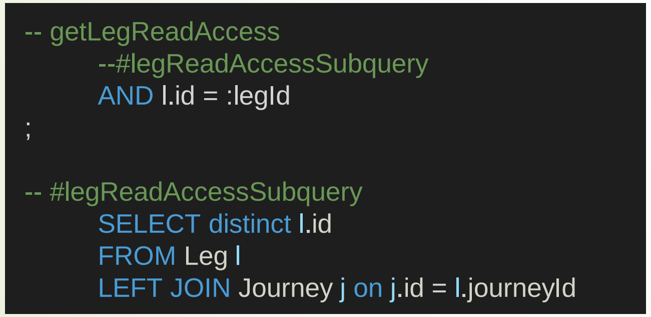
Primitive Obsession



```
protected List<SearchResultDTO> ccuByPage(String key, Map<String, Object> params) {
    try {
        var q = query(key);
        return jdbcTemplate.query(q, params, (rs, rowNum) -> {
            SearchResultDTO sr = new SearchResultDTO();
            sr.setCcuId(rs.getInt("ccuid"));
            sr.setExtCcuId(rs.getString("extccuid"));
            sr.setCcuClass(rs.getString("ccuclass"));
            sr.setCcuSubclass(rs.getString("ccusubclass"));
            sr.setMaxGrossWeight(rs.getBigDecimal("maxgrossweight"));
            sr.setTareWeight(rs.getBigDecimal("tareweight"));
            sr.setLength(rs.getBigDecimal("length"));
            sr.setWidth(rs.getBigDecimal("width"));
            sr.setHeight(rs.getBigDecimal("height"));
            sr.setImoCertificateNumber(rs.getString("imocertificatenumber"));
            sr.setImoCertificateExpiryDate(DateUtil.asLocalDateTime(rs.getTimestamp("imocertificateexpirydate")));
            sr.setCertificateNumber(rs.getString("certificatenumber"));
            sr.setCertificateExpiryDate(DateUtil.asLocalDateTime(rs.getTimestamp("certificateexpirydate")));
            sr.setOwningOrgId(rs.getInt("owning_org_id"));
```

Primitive Obsession

- If any of these string has a typo it the application will blow up.
- They are uses for parameters in SQL files.
- The sql files has also primitive obsession.



```
-- getLegReadAccess
        --#legReadAccessSubquery
        AND l.id = :legId
;

-- #legReadAccessSubquery
        SELECT distinct l.id
        FROM Leg l
        LEFT JOIN Journey j on j.id = l.journeyId
```

Summary

- There is a lot of good stuff also
- Still hard to read and hard to change
- Why?
- The sheer size of it
- The class hiearchy is hard to navigate
- A lot of the code is not obvbious written
- How to fix it? It not a easy task.
- Code smells, Object calistenics rules and TPP will help.

Thank you!

- Any Questions?

Eirik Dyrli
eirik.dyrli@bouvet.no