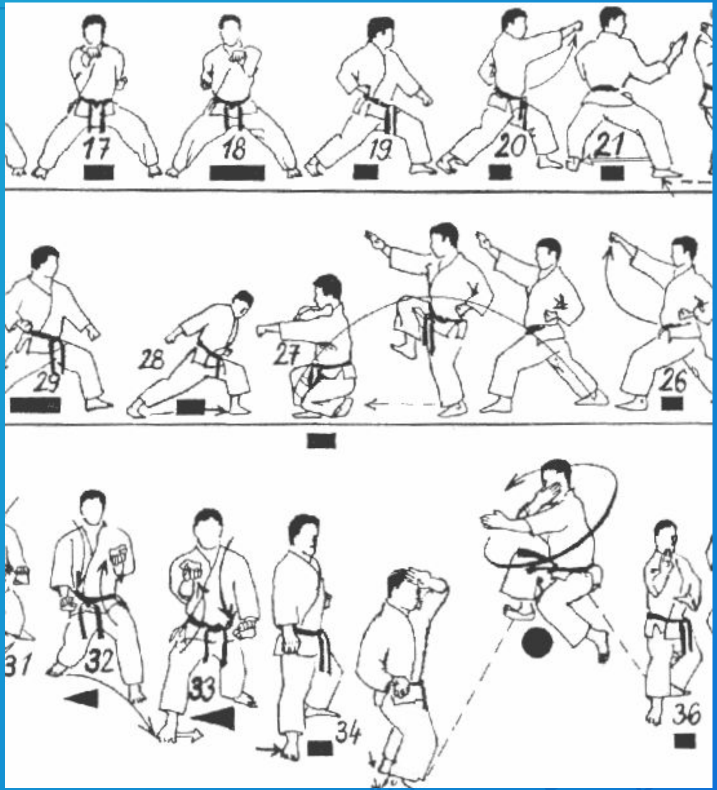
Code Kata:
Refactoring Legacy Code

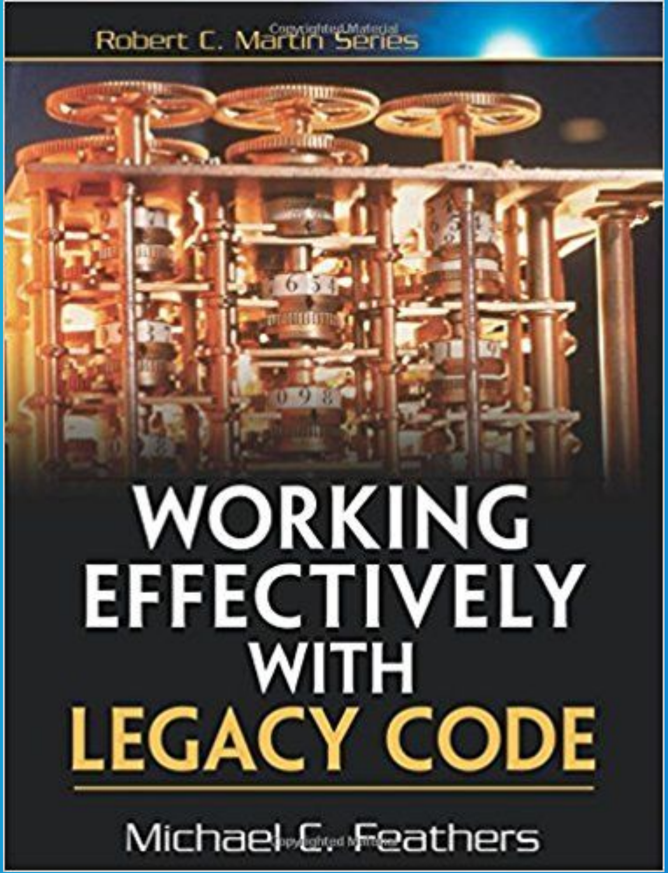
What’s Code Kata?

Dave Thomas

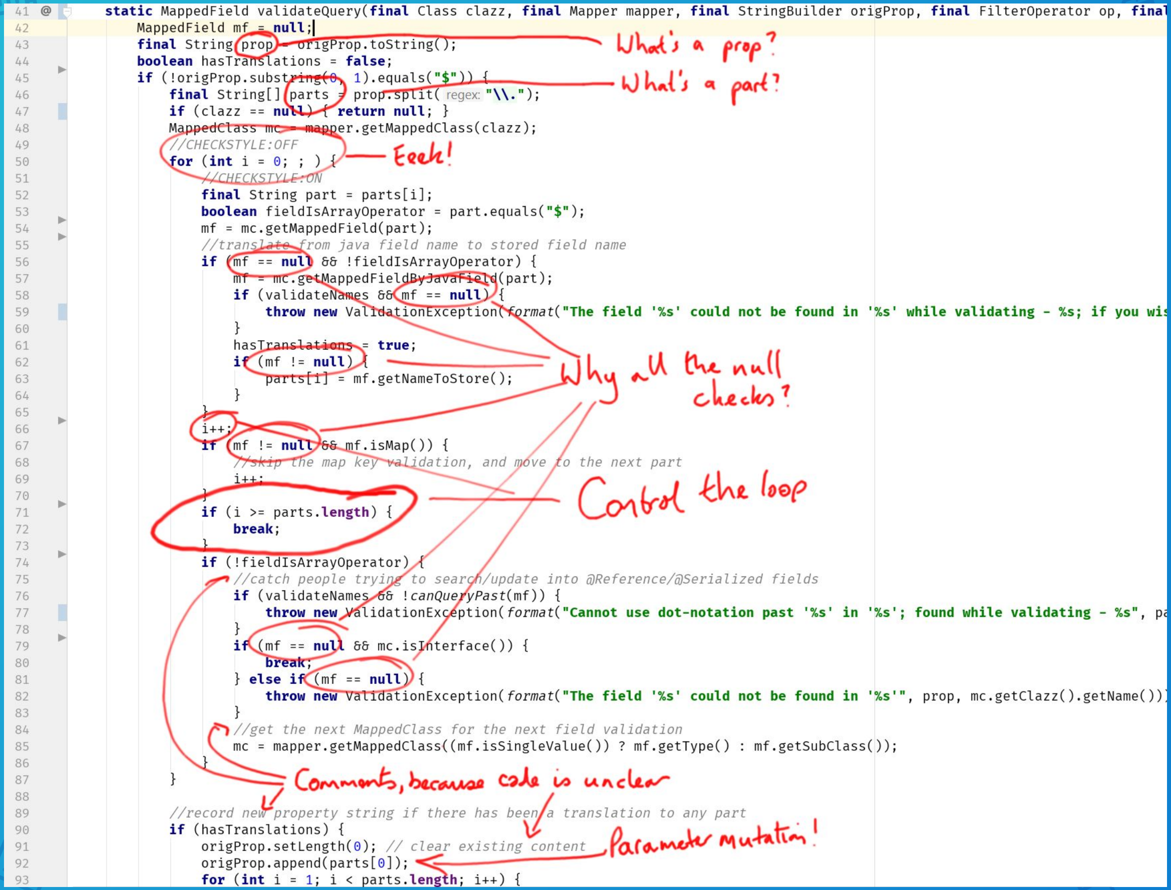
YOW!
AUSTRALIA SOFTW
DEVELOPER CONFERE
www.yowconference.co
1010101


17
18
19
20
21
29
28
27
26
31
32
33
34
36

What is legacy code?


Copyrighted Material
Robert C. Martin Series
WORKING EFFECTIVELY WITH LEGACY CODE
Michael C. Feathers
Copyrighted Material

What is refactoring?



```
static MappedField validateQuery(final Class clazz, final Mapper mapper, final StringBuilder origProp, final FilterOperator op, final
    MappedField mf = null;
    final String prop = origProp.toString();
    boolean hasTranslations = false;
    if (!origProp.substring(0, 1).equals("$")) {
        final String[] parts = prop.split( regex: "\\.");
        if (clazz == null) { return null; }
        MappedClass mc = mapper.getMappedClass(clazz);
        //CHECKSTYLE:OFF
        for (int i = 0; ; ) {
            //CHECKSTYLE:ON
            final String part = parts[i];
            boolean fieldIsArrayOperator = part.equals("$");
            mf = mc.getMappedField(part);
            //translate from java field name to stored field name
            if (mf == null && !fieldIsArrayOperator) {
                mf = mc.getMappedFieldByJavaField(part);
                if (validateNames && mf == null) {
                    throw new ValidationException(format("The field '%s' could not be found in '%s' while validating - %s; if you wis
                }
                hasTranslations = true;
                if (mf != null) {
                    parts[i] = mf.getNameToStore();
                }
            }
            i++;
            if (mf != null && mf.isMap()) {
                //skip the map key validation, and move to the next part
                i++;
            }
            if (i >= parts.length) {
                break;
            }
            if (!fieldIsArrayOperator) {
                //catch people trying to search/update into @Reference/@Serialized fields
                if (validateNames && !canQueryPast(mf)) {
                    throw new ValidationException(format("Cannot use dot-notation past '%s' in '%s'; found while validating - %s", pa
                }
                if (mf == null && mc.isInterface()) {
                    break;
                } else if (mf == null) {
                    throw new ValidationException(format("The field '%s' could not be found in '%s'", prop, mc.getClazz().getName())
                }
                //get the next MappedClass for the next field validation
                mc = mapper.getMappedClass((mf.isSingleValue()) ? mf.getType() : mf.getSubClass());
            }
        }

        //record new property string if there has been a translation to any part
        if (hasTranslations) {
            origProp.setLength(0); // clear existing content
            origProp.append(parts[0]);
            for (int i = 1; i < parts.length; i++) {
```

What's a prop?

What's a part?

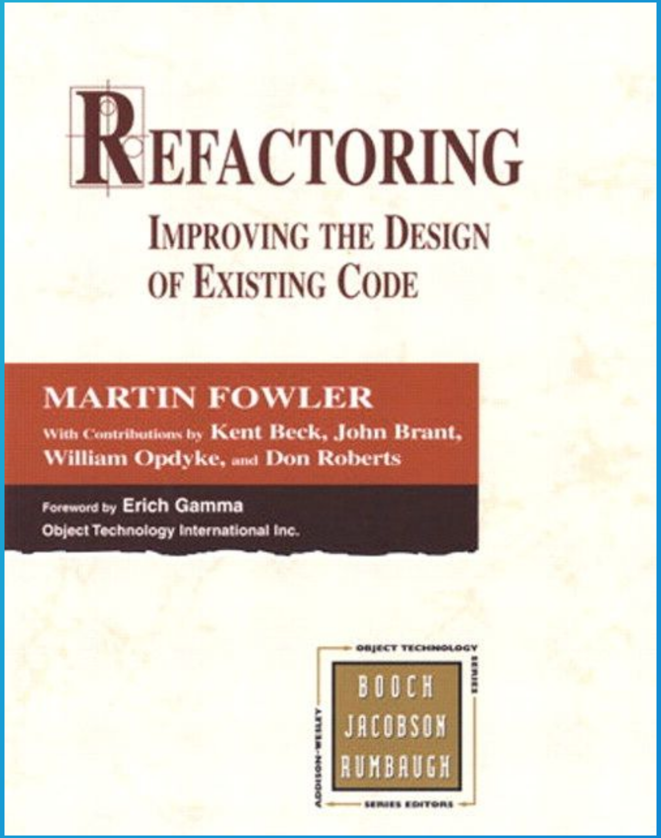
Eeeh!

Why all the null checks?

Control the loop

Comments, because code is unclear

Parameter mutation!



REFACTORING

IMPROVING THE DESIGN OF EXISTING CODE

MARTIN FOWLER

With Contributions by Kent Beck, John Brant, William Opdyke, and Don Roberts

Foreword by Erich Gamma

Object Technology International Inc.

OBJECT TECHNOLOGY SERIES

BOOCH JACOBSON RUMBAUGH

ADDISON-WESLEY

SERIES EDITORS

Challenges with Legacy Code

Convoluted Codes

The codes always grow more complex over time.

Missing Documentation

We need to make changes to it but we don't know enough about what it should do.

Missing Tests

We can't do refactoring without tests!!!

Not Testable

If you don't write tests before implementation, it could be very hard to add tests later

What should we do?

Characterization Tests

a.k.a Golden Master Test

Purpose of Characterization Tests

- Observe/Learn how the system currently works
- Build a safety net to catch changes in behavior
  - We can do refactoring after this!

Generate Characterization Test as Unit Test

- Assumptions:
  - The system under test can be isolated from its dependencies
  - The behavior is repeatable
- Steps:
  - Write an assertion that you know will fail.
  - Run the test and let the failure tell you what the actual behavior is.
  - Change the test so that it records the behavior that the code actually produces.
- Example
-

Generate Characterization Test as System Test

- Assumptions:
  - It's hard to write unit test in current codebase (so we have to test the whole system together)
  - The system logs it behavior to console/file
  - The behavior is repeatable for each fixed input
- Steps:
  - Redirect the logs to a file if it's not done yet
  - Find the set of possible inputs
  - Run the system with each input and record the behaviors in log files
  - Write tests to verify system behavior against the Golden Master Records

The Ugly Trivia

- https://github.com/jbrains/trivia
- https://github.com/songguoqiang/trivia_refactoring_kata

Uglified “Trivial Pursuit” Game

Refactor Legacy Code

- Understand the codebase
  - Run it and read the logs
  - Read the codes
- Build the Golden Master Test
- Refactor the codes
- Replace the Golden Master Test with proper Unit Tests along with refactoring

Sample Golden Master Test

- Java
- Java
- Java
- Python
- Javascript