HOW (NOT) TO BREAK THE PRODUCTION

A survival guide on how not to be eaten by the monster

Igor Kovacevic
21.10.2021

Inspired by the “ALCOR Code Renovation” course

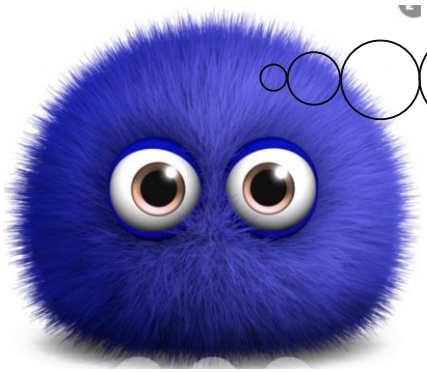
Feedback welcome: igor.kovacevic@css.ch

That is cool I just created some fancy pre-refactoring tests. Let's just run them and see what happens.

Our production system was broken.

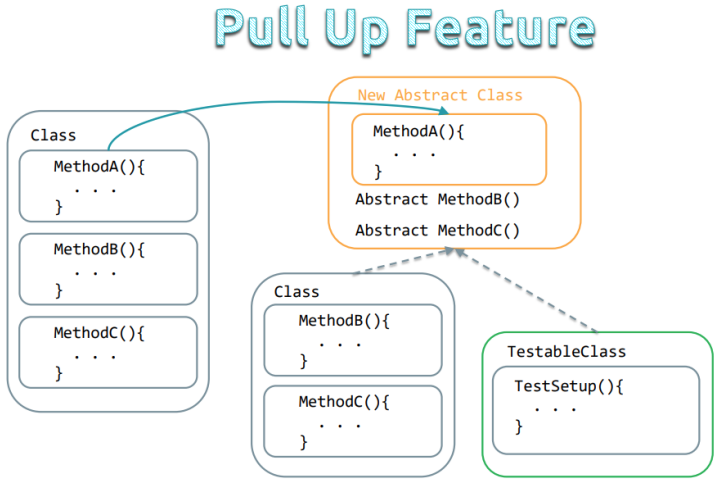
```
public class PeelAndSliceExample {
    @Test
    public void shouldTestMoreEasily(){
        MySystemUnderTest sut = new MySystemUnderTest();

        sut.doStuff( numbers: 1234);
    }
}
```

```
Tests failed: 1 of 1 test – 1 sec 14 ms
1 sec 14 ms   "C:\Program Files\Java\jdk-17\bin\java.exe" ...
1 sec 14 ms

java.lang.RuntimeException: This won't work in a test

    at PeelAndSlice.MySystemUnderTest.nastyExternalQuery(PeelAndSliceExample.java:23)
    at PeelAndSlice.MySystemUnderTest.doStuff(PeelAndSliceExample.java:9)
    at PeelAndSlice.PeelAndSliceExample.shouldTestMoreEasily(PeelAndSliceExample.java:44) <22 internal lines>
```



I saw this cool “Pull Up Feature” refactoring method. I have to try it with the most critical code of the company. It will solve all the issues.

- *when trying to test a method that doesn’t use the dependencies that makes the class difficult to instantiate*

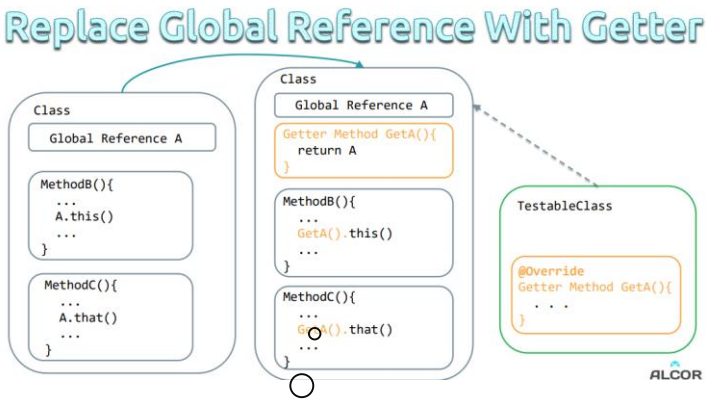


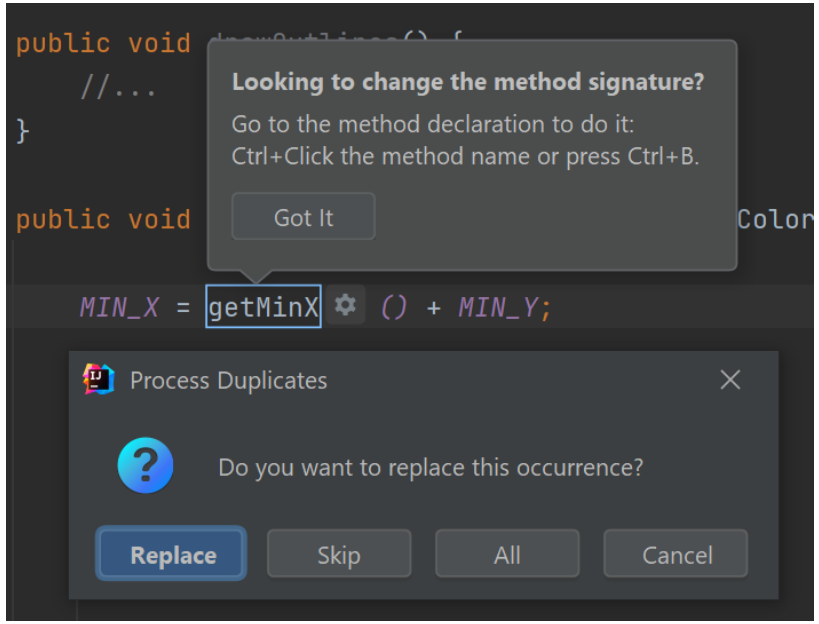

```
class TestablePager extends Pager {
    @Override
    public void formConnection() {
        // Do not Form a connection
    }
}

@Test
void sendMessagesWithoutComplaining(){
    Pager pager = new Pager();

    pager.sendMessage( address: "important@customer.com", message: "Hello, World!");
}
```

He sent a test message to our real customer.

Maybe I did send by mistake a message to the customer. But this "Replace Global References With Getter" refactoring method is dummy proof. Let's give it another shot on our critical code and save the world. And to be extra safe, I will do everything by hand and not use the IDE this time.

- *when trying to test a method that doesn't use the dependencies that makes the class difficult to instantiate*



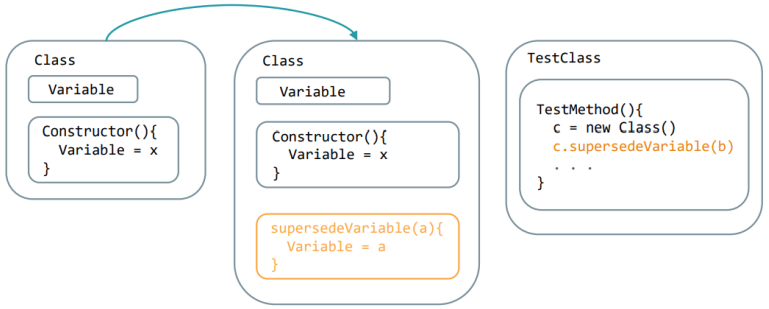


He forgot it here.

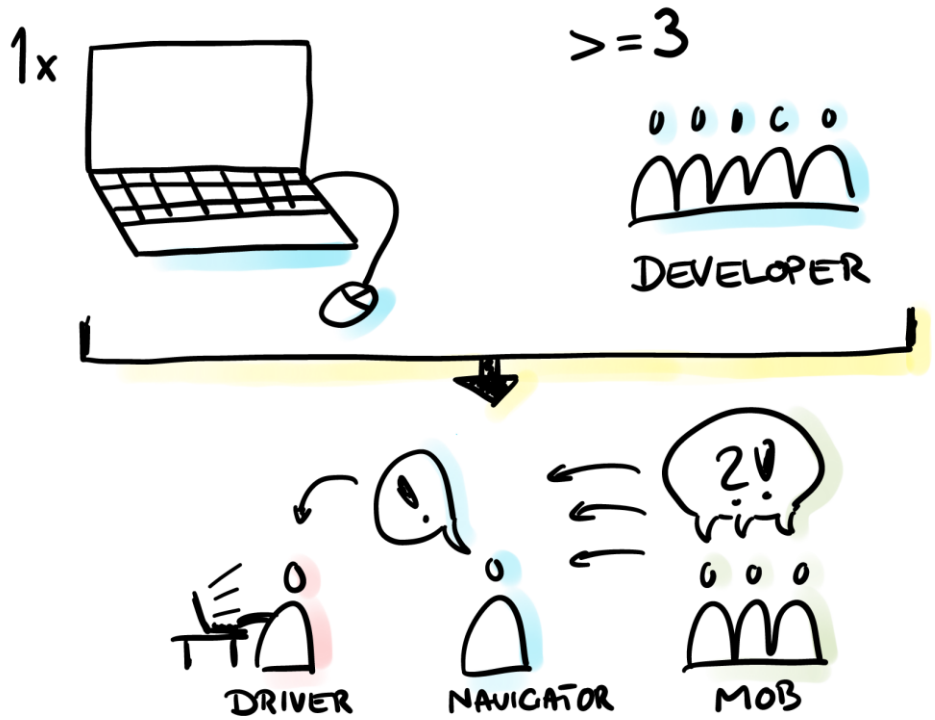
But this time it will work for sure. “Supersede Instance Variable” is easyyyy.

- *when you can’t override a virtual function call in the constructor*

Supersede Instance Variable



MOB PROGRAMMING IN A NUTSHELL



It is time to schedule a mob session.

APPROVAL TESTS

MAKES IT EASY

- *... for humans to evaluate results*
- *... to set tests up*
- *... to format test output for readability*
- *... to maintain tests*
- *... in multiple language*
- *... with visual results*
- *... to make characterization tests*

Wow. This Approval Test Framework allowed me very easy to test all combinations. My coverage is at 100%, I will save the golden master and finally, I can start refactoring.

```java
import org.approvaltests.combinations.CombinationApprovals;
import org.junit.jupiter.api.Test;

class ApprovalsExampleTest {

    @Test
    void mergeNameAndAgeWithApprovalsWithCombinations() {
        //do
        String[] names = new String[]{"alex"};
        Integer[] ages = new Integer[]{15, 25, 61};

        //verify
        CombinationApprovals.verifyAllCombinations(
                (name, age) -> {
                    String result = ApprovalsExample.mergeNameAndAge(name, age);
                    //returns a string
                    return result;
                }, names, ages);
    }
}
```

He missed some edge cases because he forgot the gold standard for coverage: 100% with mutation tests.

MUTATION TESTS

Traditional test coverage measures only which code is **executed** by your tests. It is therefore *only able to identify code that is definitely* ***not tested***.

As it is actually able to detect whether each statement is ***meaningfully*** tested, *mutation testing is the* ***gold standard*** against which all other types of *coverage* are measured.

Available mutators and groups

The following table list available mutators and whether or not they are part of a group :

Mutators	"OLD_DEFAULTS" group	"DEFAULTS" group	"STRONGER" group	"ALL" group
Conditionals Boundary	yes	yes	yes	yes
Increments	yes	yes	yes	yes
Invert Negatives	yes	yes	yes	yes
Math	yes	yes	yes	yes
Negate Conditionals	yes	yes	yes	yes
Return Values	yes			yes
Void Method Calls	yes	yes	yes	yes
Empty returns		yes	yes	yes
False Returns		yes	yes	yes
True returns		yes	yes	yes
Null returns		yes	yes	yes
Primitive returns		yes	yes	yes
Remove Conditionals			EQ_ELSE case	yes
Experimental Switch			yes	yes
Inline Constant				yes
Constructor Calls				yes
Non Void Method Calls				yes

```
6   1. negated conditional → SURVIVED
7   1. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED
10  1. Replaced integer subtraction with addition → KILLED
11  1. Replaced integer subtraction with addition → KILLED
    1. negated conditional → KILLED
12  2. negated conditional → SURVIVED
    3. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED
    4. replaced return of integer sized value with (x == 0 ? 1 : 0) → SURVIVED
```

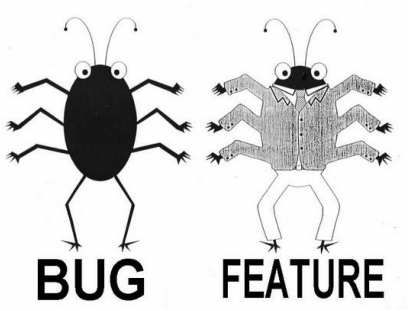
the list continues…

http://pitest.org/quickstart/mutators/

Look at this ugly bug. I will fix this annoying bug in the refactoring session.

You are a lucky bug. I'm seeing that you'll be shipped with the next five releases.

He forgot what "refactoring" means.



(CODE) REFACTORING

Refactoring: *(Definition) Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure.* [Fowler02]

- The **art** of safely improving the design of existing code [Fowler09]
- **Implications:** [Fowler09]
  - Refactoring does **not** include any change to the system
  - Refactoring is **not** "Rewriting from scratch"
  - Refactoring is **not** just any restructuring intended to improve the code

Now I think I have broken all the dependencies. Let's run the test.

Breaking dependencies means breaking dependencies. You need to be sure that you broke them.

Double Check

```
INFO: The bank transaction was executed. 1'000'000 was transfered.

Process finished with exit code 0
```

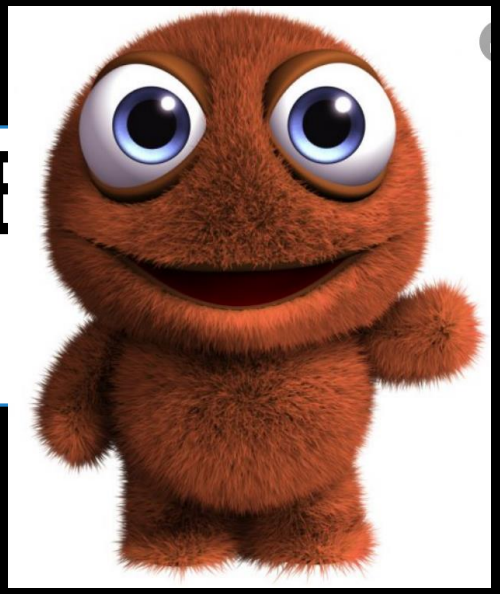
In case of fire 

 1. git commit

 2. git push

 3. leave building

GOOD NIGHT, SLEEP WELL ☺



RESOURCES

http://pitest.org/quickstart/mutators/

Code-Renovation course - Lesson-1.pdf (Contact Alessandro)

Images

https://society6.com/product/i-am-a-monster-and-i-am-gonna-eat-you_print

https://www.google.com/search?q=monster+under+bed&sxsrf=AOaemvKC9G2ogMhgCMvhJXzTqd5nLxf-WA:1634667527754&source=lnms&tbm=isch&sa=X&ved=2ahUKEwjuy8SKi9fzAhWk_7sIHbTmCpsQ_AUoAXoECAEQAw&biw=2195&bih=1035&dpr=1.75#imgrc=Y4xkRp2fweRasM

https://de.depositphotos.com/13301969/stock-photo-small-monster.html

https://github.com/hendrixroa/in-case-of-fire

https://quotesgram.com/img/software-bug-quotes/2905394/

https://www.dreams.co.uk/sleep-matters-club/the-history-of-that-monster-who-lives-under-your-bed/

HOW (NOT) TO BREAK THE PRODUCTION

A survival guide on how not to be eaten by the monster

Igor Kovacevic
21.10.2021

Inspired by the “ALCOR Code Renovation” course

Feedback welcome: igor.kovacevic@css.ch