Mutation Testing to the rescue of your tests

@nicolas_frankel

TestCon

hazelcast

Me, myself and I

- Former developer, team lead, architect, **whatever it takes**
- Developer Advocate
- Quality-obsessed

@nicolas_frankel

hazelcast

Hazelcast

HAZELCAST IMDG is an **operational, in-memory**, distributed computing platform that manages data using in-memory storage, and performs parallel execution for breakthrough application speed and scale.

HAZELCAST JET is the ultra fast, application embeddable, 3rd generation stream processing engine for low latency batch and stream processing.

@nicolas_frankel

hazelcast

Many kinds of testing

- Unit Testing
- Integration Testing
- End-to-end Testing
- Performance Testing
- Penetration Testing
- Exploratory Testing
- etc.

Their only single goal

Ensure the quality of the production code

The problem

How to check the quality of the testing code?

Code coverage

"Code coverage is a measure used to describe the degree to which the source code of a program is tested"

--Wikipedia
http://en.wikipedia.org/wiki/Code_coverage

Measuring Code Coverage

Check whether a source code line is executed during a test

- Or Branch Coverage

Computing Code Coverage

$$CC = \frac{L_{executed}}{L_{total}} * 100$$

- CC: Code Coverage (in percent)
- $L_{executed}$: Number of executed lines of code
- L_{total}: Number of total lines of code

Java Tools for Code Coverage

- JaCoCo
- Clover
- Cobertura
- etc.

100% Code Coverage?

“Is 100% code coverage realistic? Of course it is. If you can write a line of code, you can write another that tests it.”

Robert Martin (Uncle Bob)
https://twitter.com/unclebobmartin/status/559666209509667328

Assertless testing

```
@Test
public void add_should_add() {
  new Math().add(1, 1);
}
```

But, where is the assert?

As long as the Code Coverage is OK…

Code coverage as a measure of quality

- Any metric can be gamed!
- Code coverage is a metric…

⇒ Code coverage can be gamed

- On purpose
- Or by accident

Code coverage as a measure of quality

- Code Coverage lulls you into a false sense of security…

The problem still stands

- Code coverage cannot ensure test quality
- Is there another way?
- Mutation Testing to the rescue!

The Cast

William Stryker
Original Source Code

Jason Stryker
Modified Source Code
a.k.a "The Mutant"

```java
public class Math {
  public int add(int i1, int i2) {
    return i1 + i2;
  }
}
```

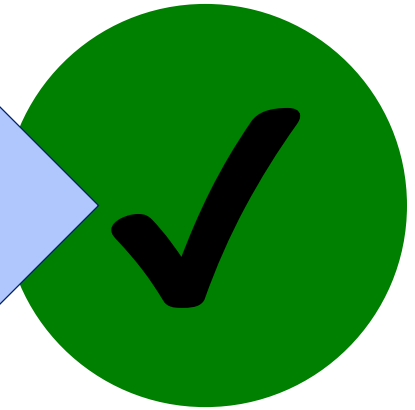
```java
public class Math {
  public int add(int i1, int i2) {
    return i1 - i2;
  }
}
```

@nicolas_frankel

hazelcast

Standard testing

Execute Test



Mutation testing

Mutation

Execute SAME Test

?

hazelcast

Mutation testing

Execute SAME Test

Mutant Killed

Execute SAME Test

Mutant Survived

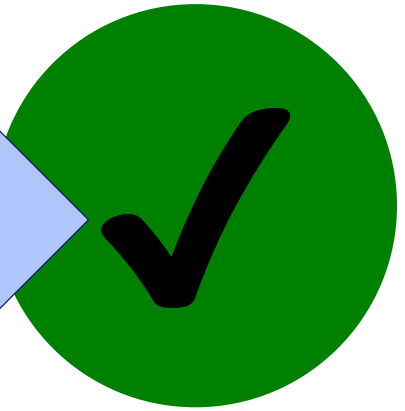
Killed or Surviving?

- Surviving means changing the source code did not change the test result → Bad!
- Killed means changing the source code changed the test result → Good!

Test the code

```
public class Math {
   public int add(int i1, int i2) {
      return i1 + i2;
   }
}
```

Execute Test



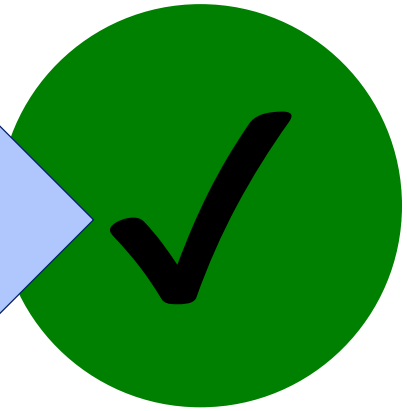
```
@Test
public void add_should_add() {
   new Math().add(1, 1);
}
```

hazelcast

Surviving mutant

```
public class Math {
  public int add(int i1, int i2) {
    return i1 - i2;
  }
}
```

Execute Test



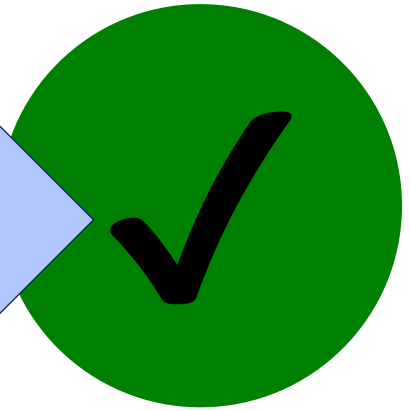
```
@Test
public void add_should_add() {
  new Math().add(1, 1);
}
```

hazelcast

Test the code

```
public class Math {
   public int add(int i1, int i2) {
      return i1 + i2;
   }
}
```

Execute Test

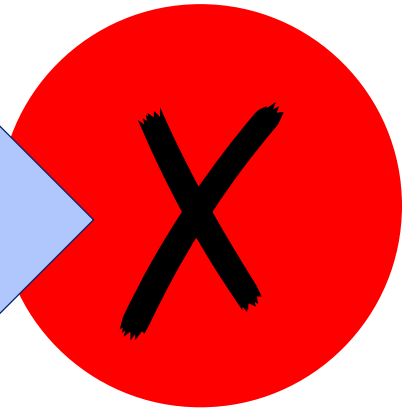


```
@Test
public void add_should_add() {
   new Math().add(1, 1);
   Assert.assertEquals(sum, 2);
}
```

Killed mutant

```
public class Math {
  public int add(int i1, int i2) {
    return i1 - i2;
  }
}
```

Execute SAME Test



```
@Test
public void add_should_add() {
  new Math().add(1, 1);
  Assert.assertEquals(sum, 2);
}
```

hazelcast

Mutation Testing in Java

- PIT is a tool for Mutation testing
- Available as
  - Command-line tool
  - Ant target
  - Maven plugin

Mutators

Mutators are patterns applied to source code to produce mutations

PIT mutators sample

Name	Example source	Result
Conditionals Boundary	>	>=
Negate Conditionals	==	!=
Remove Conditionals	foo == bar	true
Math	+	-
Increments	foo++	foo--
Invert Negatives	-foo	foo
Inline Constant	static final FOO= 42	static final FOO = 43
Return Values	return true	return false
Void Method Call	System.out.println("foo")	
Non Void Method Call	long t = System.currentTimeMillis()	long t = 0
Constructor Call	Date d = new Date()	Date d = null;

Important mutators

Conditionals Boundary

- Potential serious bug hiding there
- `if (foo > bar)`

Important mutators

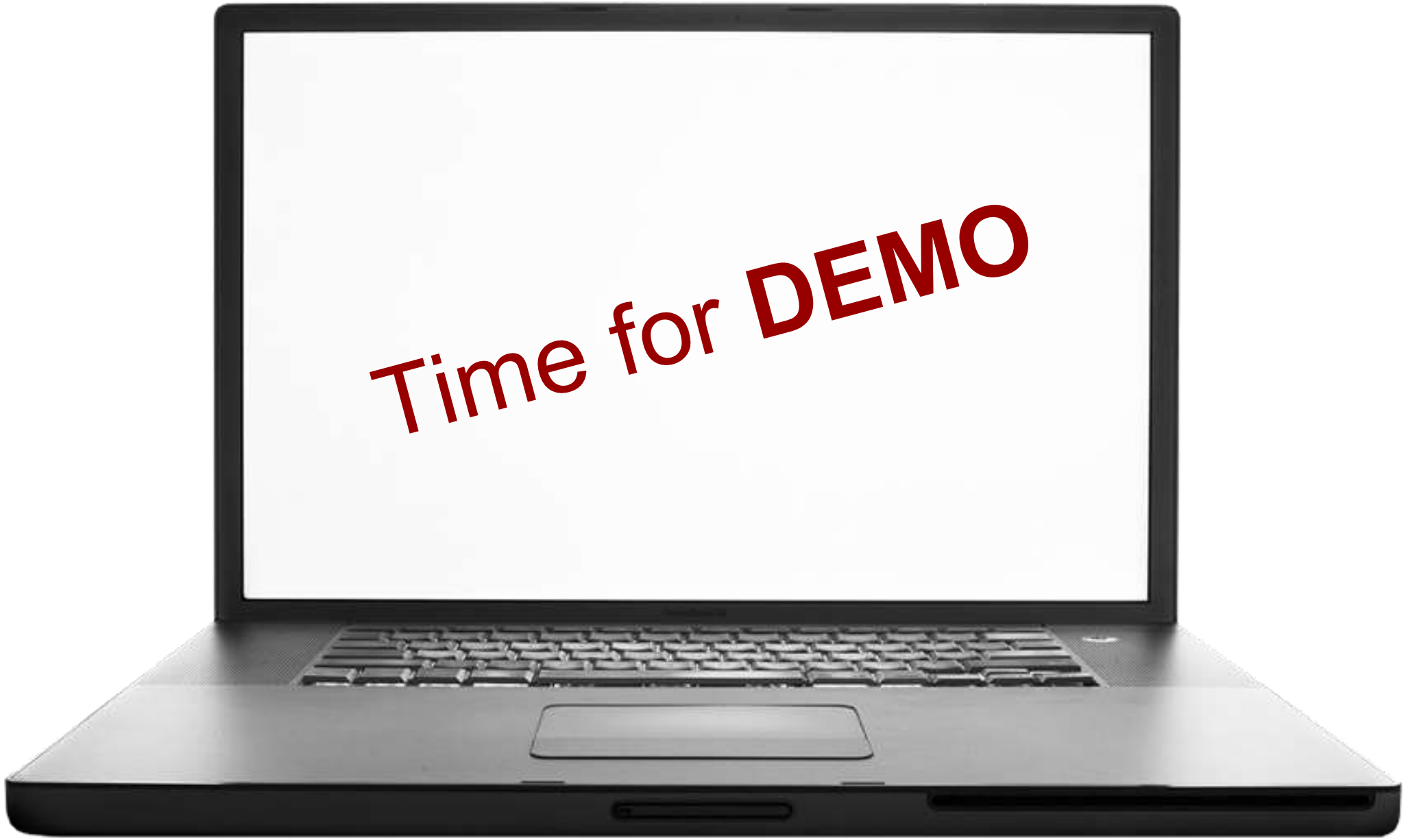
Void Method Call

- `Assert.checkNotNull()`
- `connection.close()`

Remember

It’s not because the IDE generates code safely that it will never change

- `equals()`
- `hashCode()`


Time for DEMO

Sh… (bad stuff) happens

- False positives
- Imperfect
- Slooooooooooooooow

Pit is imperfect

```
if (p < 0)
    ...
// changed condition boundary -> survived:
if (p > 0)
    ...
return 0;
```

@nicolas_frankel

hazelcast

Pit is imperfect

```
void rebootMachine() {
    // removed method call:
    checkUserPermissions();
    Runtime.getRuntime().exec("reboot");
}
```

Why so slow?

- Analyze test code
- For each class under test
  - For each mutator
    - Create mutation
  - For each mutation
    - Run test
    - Analyze result
    - Aggregate results

Workarounds

- Set a limited a set of mutators
- Limit scope of target classes
- Limit number of tests
- Limit dependency distance
- Increase number of threads → ☺
- Don’t bind to the test phase → ☺
- Use `scmMutationCoverage`
- Use incremental analysis → ☺

Is Mutation Testing the Silver Bullet?

- Checks the relevance of your unit tests
- Points out potential bugs

The rest is up to you

- Validate the assembled application
  - Integration Testing
- Check the performance
  - Performance Testing
- Look out for display bugs
  - End-to-end testing
- Etc.

Testing is about ROI

- Don’t test to achieve 100% coverage
- Test because it saves money in the long run
- Prioritize:
  - Business-critical code
  - Complex code

Thanks a lot!

- https://blog.frankel.ch/
- @nicolas_frankel
- https://git.io/vznQK
- https://git.io/fjw7Q