Unit Testing with JUnit Boot Camp

Peter Kofler, ‘Code Cop’
@codecopkofler
www.code-cop.org

geecon

Peter Kofler

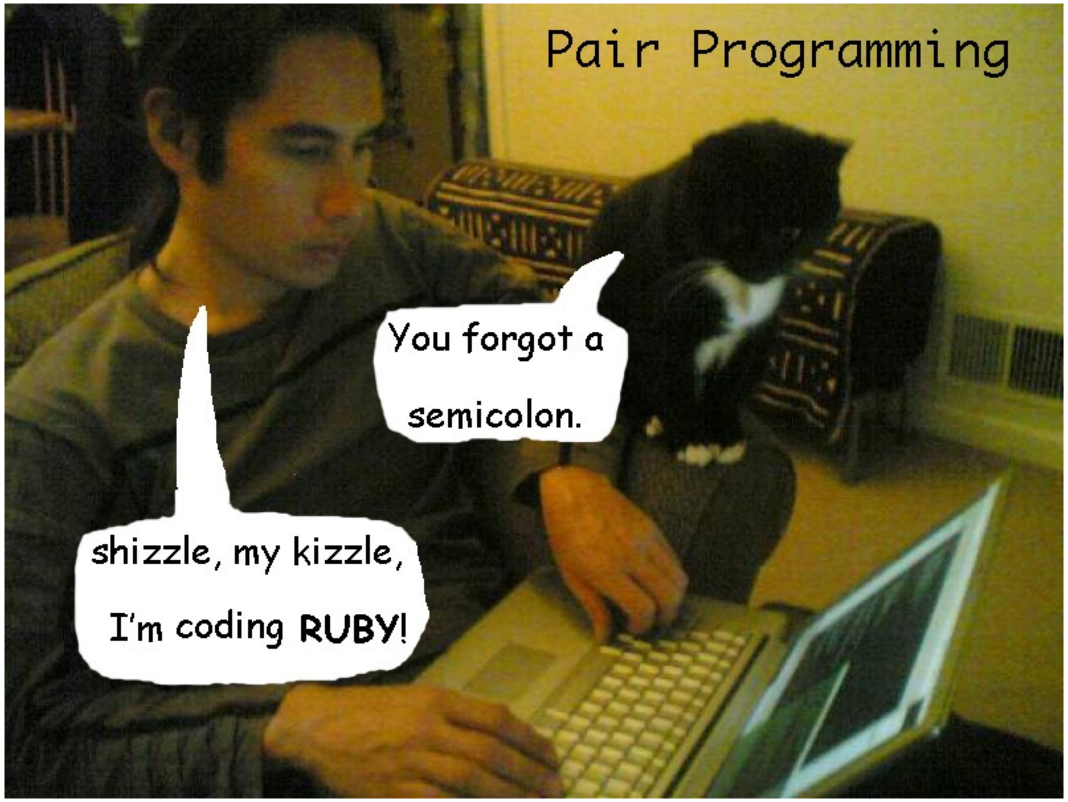
- Ph.D. (Appl. Math.)
- Professional Software Developer for 15+ years
- “fanatic about code quality”
- I help development teams

I help development teams with

- Professionalism
- Quality and Productivity
- Continuous Improvement

Mentoring

- Pair Programming
- Programming Workshops
- Deliberate Practice, e.g. Coding Dojos



Developing Quality Software Developers

Agenda

- JUnit
- Coding Exercise
- Unit Tests
- Lunch Break
- Clean Unit Tests
- Coding Exercise
- Retrospective

How to take advantage

- Slow down
- Focus on doing it right
- Get out of your comfort zone
- Embrace freedom of experimenting
- Pair with strangers you do not know
- What you learn is your responsibility

Introduce Yourself

- Your Name
- Who has heard of JUnit? (xUnit)
- Who has never written a unit test?
- Who has written a few?
- Who writes unit tests every day?

JUnit

JUnit

- http://junit.org/
- a (unit) testing framework
- tests are executed within a framework
- no output from your tests
- check expectations programmatically
- test runner reports failure on each test

Project – Class - Method

- Maven
    - `src/test/java`
- Test Class

```
public class SomeTestClass
{
     @Test
     public void someTestMethod()
     {
          ...
     }
}
```

JUnit Assertions

```java
import static org.junit.Assert.*;

assertEquals("SNEK", actual);
assertTrue(isSaved);
assertNotNull(customer);

assertArrayEquals(
  new String[] {"Berg", "Wien"},
  citiesArray);
```

Hamcrest Assertions

```
import static org.junit.Assert.assertThat;
import static org.hamcrest.core.IsCollectionContaining.hasItem;
import static org.hamcrest.core.IsInstanceOf.instanceOf;
import static org.hamcrest.core.StringContains.containsString;
import static org.hamcrest.core.StringEndsWith.endsWith;
```

```
assertThat(actual,instanceOf(Customer.class));

assertThat(message,
  containsString("abgelaufen"));

assertThat(message, endsWith("des Jahres."));

assertThat(citiesCollection, hasItem("Wien"));
```

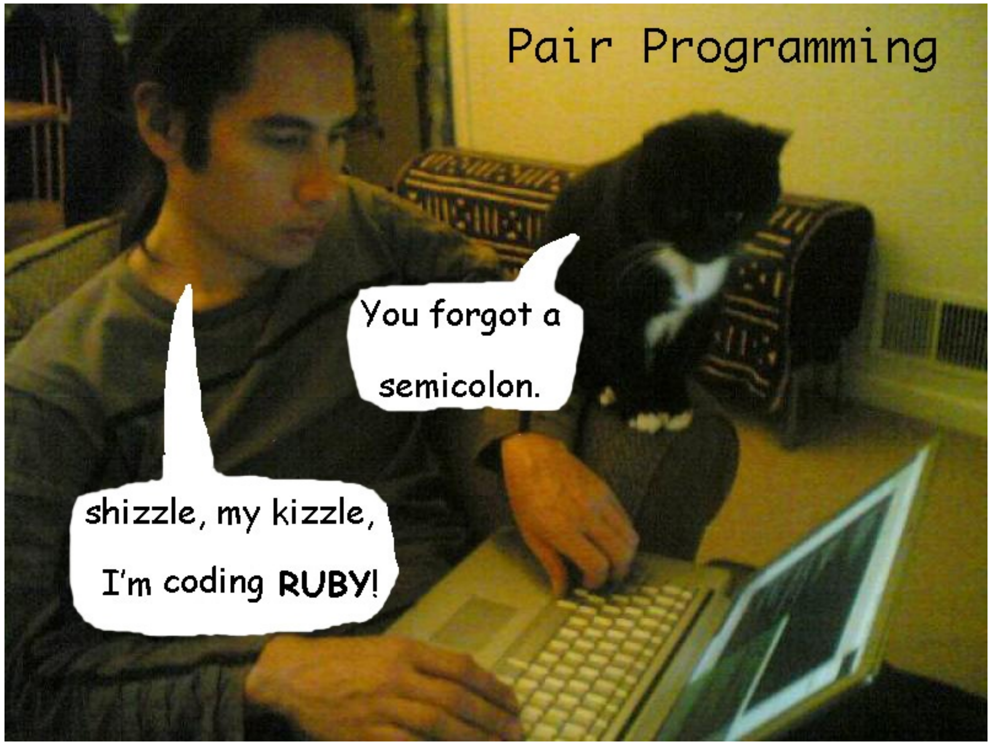
Try it yourself

Setup

- Find a pair.
- Get the code (https://bitbucket.org/pkofler/junit-koans)
- Run JUnit - should see no tests

Pair Programming

- Collaborative = Pair Programming
- do not talk for too long
- do not interrupt the other
- no “keyboard hugging“



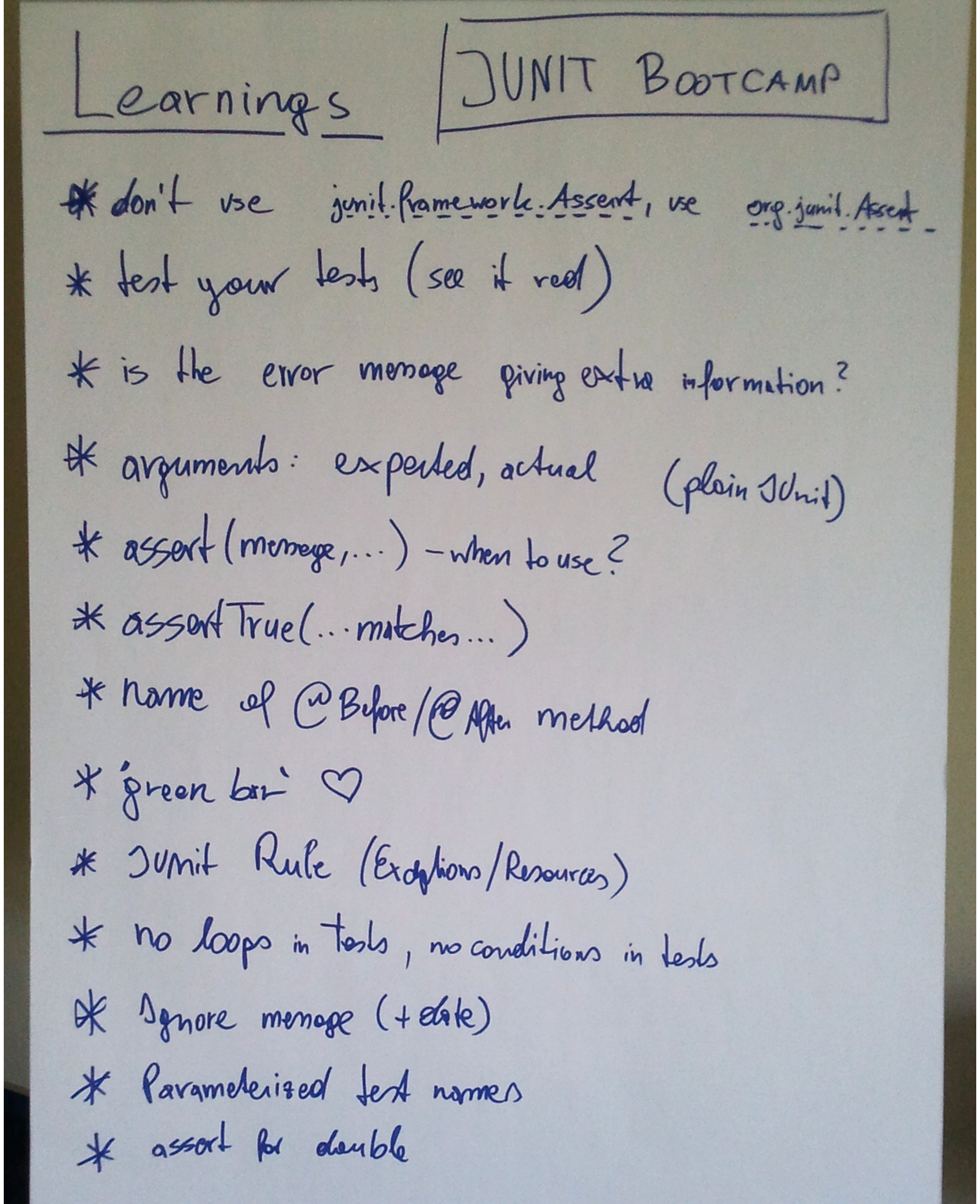
Pair Programming adds discussion & a second opinion to the practice.

Assignment

- Go through the test code
- Assertions are commented/ incomplete
- uncomment the assertions
- and complete them making tests pass

- Explore features of JUnit

→Practice



Learnings

JUNIT BOOTCAMP

* don't use junit.framework.Assert, use org.junit.Assert
* test your tests (see it red)
* is the error message giving extra information?
* arguments: expected, actual (plain JUnit)
* assert(message, ...) – when to use?
* assertTrue(...matches...)
* name of @Before/@After method
* 'green bar' ♡
* JUnit Rule (Exceptions/Resources)
* no loops in tests, no conditions in tests
* Ignore message (+ date)
* Parameterized test names
* assert for double

Keep the bar green to keep the code clean

Unit Tests

Why write tests?

“I write unit tests for one reason: so my coworkers don't f*** up my code.”

(David Angry)

Unit Test (Micro Test)

- code written by a developer
- tests an individual unit
    - isolate each part
- shows that the individual part is correct
- sort of living documentation

Unit of Work

- single logical functional use case
- invoked by some public interface
  - a single method,
  - a whole class or
  - multiple classes
- with one single logical purpose

Focus on Behaviour

- e.g. requirements are behaviour
- names from (problem) domain
- full sentences
  (long, descriptive test method names)
- expected behaviour **should**
- separation per business module

Consist of 3 Simple Steps

- Prepare Input – **A**rrange
- Call Method – **A**ct
- Check Output – **A**ssert
- Use in form of **Given-When-Then**

- No conditional logic (→ more test cases)
- No loops (→ parametrized test)

Only one aspect/feature

- tests work best with lots of small tests
- tests only fail because of one reason
- more than 1 assert in test
  → split into multiple tests

`testHighAndLowInput()` ✗ → `testHighInput()` `testLowInput()` ✓

- 1 test case – 1 method – 1 assertion

Attributes of a Good Unit Test?

F.I.R.S.T

- Fast
- Isolated (data, sequence)
- Repeatable
- Self-verifying
- Timely

Clean, Readable and Expressive?

http://theprofoundprogrammer.com/post/26561881517/text-single-letter-variables-who-the-fuck-do

Clean?

- **Free from dirt** or marks:
  e.g. a clean kitchen floor.
- **Without imperfections** or errors:
  e.g. a clean edge.

- What if all your tests would be nicely structured and consistent?

Readable?

- **Easily** read; **legible**:
  e.g. a readable typeface.
- **Enjoyable** or **interesting** to read:
  e.g. a readable story.

- What if a test suite would be a readable document at the same time?

Expressive?

- Full of expression; **meaningful**:
  e.g. an expressive shrug.
- **Effectively** conveying **thought**:
  e.g. an expressive glance.

- What if tests revealed their intend?
  Would express what should happen?

Clean Tests

- Are of same quality as production code.
- Are clean code, structured, consistent.
- Are a readable document.
- Reveal their intend and express what should happen.
- Give informative error message on failure.

Welcome to the Gilded Rose

The existing inventory system

- We have **items** to sell. Items degrade in quality the older they get.
- All items have a **SellIn value** which denotes the number of days we have to sell the item.
- All items have a **Quality value** which denotes how valuable the item is.

Requirements

- At the end of each day our system lowers both values for every item.
- Once the sell by date has passed, Quality degrades twice as fast.
- The Quality of an item is never negative.
- The Quality is never more than 50.

Special Item: Brie

- *Aged Brie* actually increases in Quality the older it gets.

Backstage Passes

- *A backstage pass* increases in Quality as it's SellIn value approaches (by a complex formula)

- but Quality drops to 0 after the concert.

Special Item

- *Sulfuras*, a legendary item, never has to be sold or decreases in Quality.

Setup

- Find a pair.
- Get the code. (https://github.com/emilybache/GildedRose-Refactoring-Kata)
- Run tests, should see single failing test.
- Read *GildedRoseRequirements.txt*
- Run `TextTestFixture` to see what it does.

Assignment

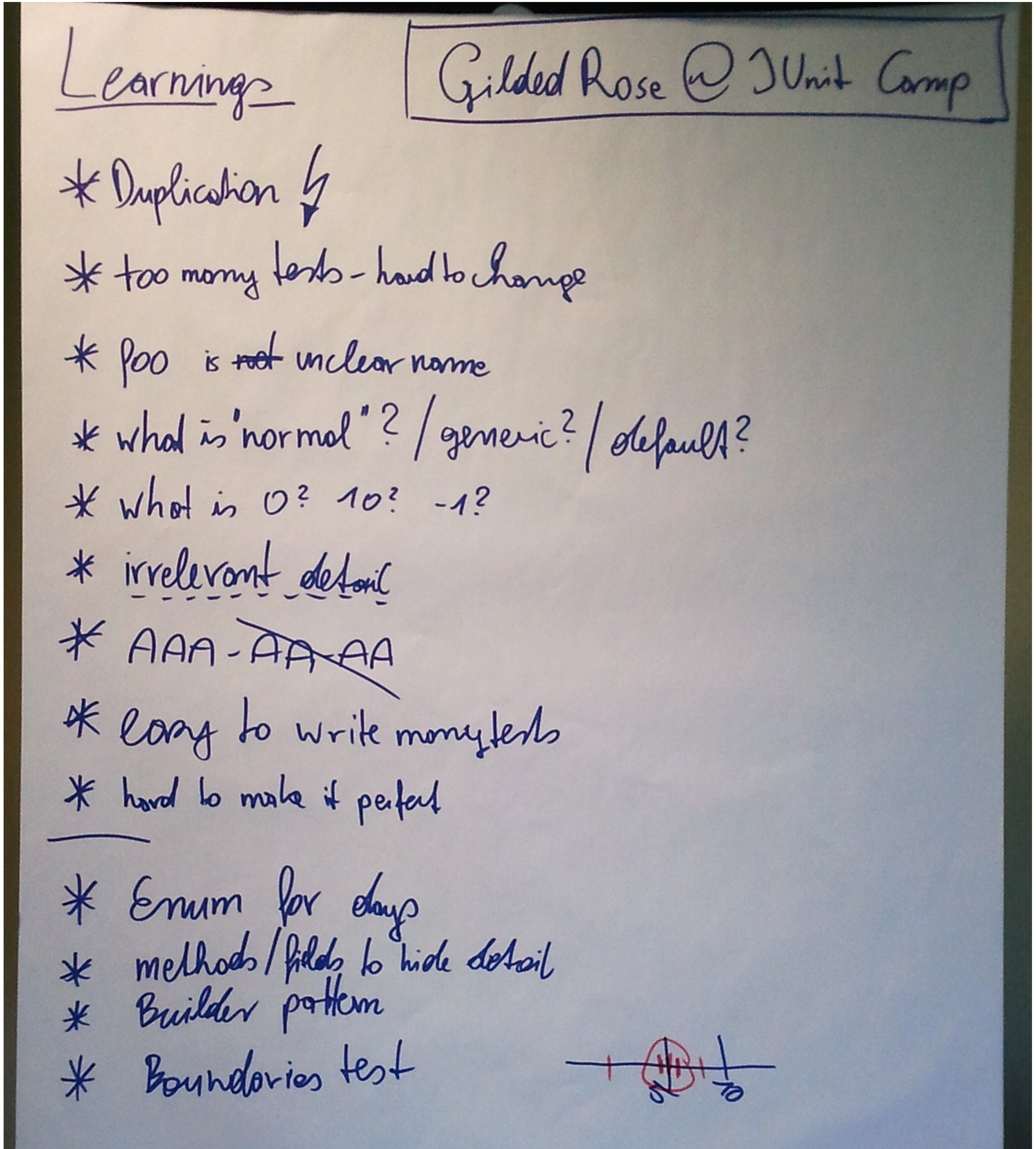
- Create “perfect” unit tests
  - derive test cases from requirements
  - cover all cases e.g. boundary conditions
  - readable, concise, free of duplication

- Experiment with different styles.

Create a test suite that is a readable document at the same time!

Don't Focus on Getting it Done. Focus on Doing It Perfectly.

→Practice



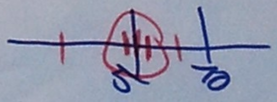
Learnings

Gilded Rose @ JUnit Camp

* Duplication ↯
* too many tests - hard to change
* foo is ~~not~~ unclear name
* what is "normal"? / generic? / default?
* what is 0? 10? -1?
* irrelevant detail
* AAA - ~~AA AA~~
* long to write many tests
* hard to make it perfect

---

* Enum for days
* methods/fields to hide detail
* Builder pattern
* Boundaries test



Closing Circle

- What did you learn today?
- What surprised you today?
- What will you do differently in the future?

Peter Kofler

@codecopkofler

www.code-cop.org

Kata by

Emily Bache

@emilybache

http://coding-is-like-cooking.info/2013/03/writing-good-tests-for-the-gilded-rose-kata/

CC Images

- Bruce http://www.flickr.com/photos/sherpas428/4350620602/
- pairing http://www.flickr.com/photos/dav/94735395/
- agenda http://www.flickr.com/photos/24293932@N00/2752221871/
- wants you http://www.flickr.com/photos/shutter/105497713/
- hands https://www.flickr.com/photos/ninahiironniemi/497993647/
- green http://www.flickr.com/photos/43442082@N00/296362882/
- inn http://www.flickr.com/photos/danielleblue/170496395/
- Brie http://www.flickr.com/photos/chez_loulou/2767503201
- pass http://www.flickr.com/photos/frf_kmeron/5556518514
- Sulfuras https://www.flickr.com/photos/sharelabs/11195626116