Certified Scrum Developer

(Technical practices)

Learning Objectives

By the end of the course you will be able to:

- **Feel** the difference between a traditional and an agile team
- Understand the responsibilities of the team in Scrum
- Participate in Scrum Events such as Sprint Planning, Daily Scrum and Retrospective
- Apply team practices such as working agreements, decision making and pair programming
- Apply technical practices such as test-driven development, refactoring and continuous integration
- Apply agile design practices such as emergent design, SOLID principles and patterns

Certified Scrum Developer

- Requirements for Certified Scrum Developer:
  - **At least three days must be technical training through a three-day CSD-track technical skills course.**
  - At least one of the five days must be dedicated to the study of Scrum through a CSD-track Introduction to Scrum course or CSM/CSPO course.
  - **The fifth day is an elective. You can choose from among the CSD-track courses to develop the skills you feel would benefit you the most in your role**. The two-day CSM or CSPO course fulfills both Intro to Scrum and the elective requirements.

More info at:
http://www.scrumalliance.org/pages/certified_scrum_developer

Agenda (1)

Monday

- Requirement workshop / A-TDD
- SCM, Build Automation and other tools
- Definition of Done
- Sprint Planning
- Pair Programming

Tuesday

- Test-Driven Development
- Working in teams
- Continuous Integration and CI Systems
- Using your IDE
- Collective Code Ownership

Agenda (2)

Wednesday

- Code Smells & Refactoring
- Mocking
- Unit testing in JavaScript

Thursday

- Thinking about Design
- Good unit tests
- Working with Legacy Code

Friday

- Craftsmanship
- Retrospective

Typical day

- 9:00 Code Review and Feedback of previous day
- 9:30 Daily Scrum
- 10:30 Lecture #1
- 12:30 Lunch
- 15:00 Lecture #2
- 18:00? End

Course Contributors

Lv Yi

Steven Mak

Stanly Lau

Lasse Koskela

Ebacky

Bas Vodde

Stuart Turner

The product

Product Background

Inspector Gadget-a gadget salesman-has a list of contacts to be informed of the latest, greatest and hottest gadgets.

He wants to:

- send the “Gadget & Widget Newsletter”
- send personalized mail based on certain preferences of his contacts
- import/export his contacts from/to different sources
- use different mail servers such as gmail and his own provider

Products he looked at

http://www.falkon-ware.com/SerialMailer

http://www.emailarms.com/products/bulk_mac_mail.html

After careful inspection

None of the existing products fulfills his needs.

Some important missing functionality:

- Doing test-runs to avoid sending to the wrong people
- Being able to cc Penny
- Resending on failures
- Integration with Google
- Friendly to use, after all... he is... inspector gadget

So, he hired you (the team) to create a Personalized Mass Mailer

Project

Build a personalized mass mailer for Inspector Gadget

You:

- work as a team
- are guided through the planning & requirement events
- are provided with a Product Backlog
- will be introduced to technical practices before applying them
- have one week
- shall have fun!

Specification by Example

(or Acceptance Test Driven Development)

Have you ever...

built the wrong thing due to misunderstanding

had PO ask why this and that isn't supported

had to wait for clarifications mid-sprint

uncovered false assumptions in the sprint review

built too fancy things for what was needed

Collaboration is key

team gets feedback earlier

team understands what they're implementing

shared language and vocabulary is built

scope of work is clear and understood by all

team collaborates closely with product owner

A-TDD

Discuss
in workshop
story
Develop
in concurrence
Deliver
for acceptance
item

Odd-e

Discuss
in workshop

Develop
in concurrence

Deliver
for acceptance

Focus on customer collaboration and user engagement. Try to get as many of these people attend as you can.

Product Owner

Users

Dev Team

Tech Writers

Help Desk

IT operations

?

Odd-e

Discuss in workshop → Develop in concurrence → Deliver for acceptance

I want to show scores for the current pair.

OK. Can you give me an example of these scores?

So if the three judges gave scores 7, 11 and 2 then the displayed score should be 20.

OK. And would those individual judges' scores be shown? Or just the total?

Yes, individual scores. And they should be shown all the time, even before all judges have given their scores.

Like if first two judges give a 7 and an 11 then we'll show 18 as an intermediate

Exactly.

Odd-e

Discuss in workshop

Develop in concurrence

Deliver for acceptance

With 3 judges giving scores 4, 20, and 18, the displayed score should be 42.

When the first 2 judges have given their scores, e.g. 10 and 5, the intermediate score of 15 should be displayed already.

No scores displayed as a dash (–), not zero.

Maximum score from a judge is 20 points!

Judge A	B	C	Score
			—
10	5		15
4	20	18	42
21	0	-5	ERROR

Can you give me an example?

What if … ?

How frequently does this happen?

What other variations of this would be common?

What other types of users would typically do this?

How could this step go wrong?

Is there anything else that would happen?

What would be different after this has happened?

Tell me how this would work?

Where would that data be used?

What would you expect

And then…?

What would that look like?

Can you draw that for

Odd-e

Discuss in workshop

Develop in concurrence

Deliver for acceptance

The examples are translated into executable specification, allowing a computer to run them against the product.

Specification pyramid

Odd-e
Discuss
in workshop
story
Develop
in concurrence
Deliver
for acceptance
item
JOHN SMITH
MIKE DAVIS
Hello $FIRSTNAME,
How are you today?
Hello John,
How are you today?
Hello Mike,
How are you today?

www.craiglarman.com
www.odd-e.com

Copyright © 2010
C.Larman & B. Vodde
All rights reserved.

requirements workshop:
clarify the requirements by
writing example tests

iteration:
develop in concurrence

Requirements workshops are great opportunities to interact with all kinds of stakeholder groups:

- end users
- help desk
- technical support
- marketing
- sales

ATDD in summary

collaborative requirements discovery

tests as requirements, requirements as tests

workshops for clarifying requirements

concurrent engineering

prevention rather than detection

Odd-e

Acceptance Test Driven Development with

Robot Framework

Robot Framework

Odd-e

Discuss in workshop

Develop in concurrence

Deliver for acceptance

With 3 judges giving scores 4, 20, and 18, the displayed score should be 42.

When the first 2 judges have given their scores, e.g. 10 and 5, the intermediate score of 15 should be displayed already.

No scores displayed as a dash (–), not zero.

Maximum score from a judge is 20 points!

Judge A	B	C	Score
			—
10	5		15
4	20	18	42
21	0	-5	ERROR

Odd-e

Discuss
in workshop

Develop
in concurrence

Deliver
for acceptance

With 3 judges giving scores 4, 20, and 18, the displayed score should be 42.

When the first 2 judges have given their scores, e.g. 10 and 5, the intermediate score of 15 should be displayed already.

No scor
a dash

Maxim
a judge

Judge A

10

4

21

Robot tests are written in tables so that computers can read them

Invalid Login

Setting	Value	Value	
Force Tags	regression		
Suite Setup	Open Login Page		
Suite Teardown	Close Browser		
Test Teardown	Go To Login Page		
Resource	resource.html		

Variable	Value	Value	

Test Case	Action	Argument	A
Invalid Username	Login With Invalid Credentials Should Fail	invalid	mode
Invalid Password	Login With Invalid Credentials Should Fail	demo	invalid
Invalid Username And Password	Login With Invalid Credentials Should Fail	invalid	invalid
Empty Username	Login With Invalid Credentials Should Fail		invalid
Empty Password	Login With Invalid Credentials Should Fail	demo	\
Empty Username And Password	Login With Invalid Credentials Should Fail		\

Higher Level Login

Setting	Value	Value	Value
Force Tags	regression	smoke	
Test Setup	Open Login Page		
Test Teardown	Close Browser		
Resource	resource.html		

Variable	Value	Value	Value

Test Case	Action	Argument	Argume
Higher Level Valid Login	Input Username	demo	
	Input Password	mode	
	Click Login Button		
	Welcome Page Should Be Open		
Even Higher Level Valid Login	Login With Valid Credentials	demo	mode
Highest Level Login	Login As A Valid User		

Keyword	Action	Argument	Argume

It's all in the tables

Test Case	Step
Valid Login	Given login page is open
	When valid username and password are inserted
	and credentials are submitted
	Then welcome page should be open

Robot Architecture

Test Data (Tables)

Robot Framework

Test Library API

Test Libraries

Robot comes with a number of built-in test libraries and you can (should!) add your own.

Test Tools

Test libraries can use any test tool necessary to interact with the system under test.

application interfaces

System Under Test

Test Cases are composed of keyword-driven actions

this is the name of a *test case*

these *keywords* form the test case

Test Case	Action	Argument	Argument
User can create an account and log in	Create Valid User	fred	P4ssw0rd
	Attempt to Login with Credentials	fred	P4ssw0rd
	Status Should Be	Logged In	
User cannot log in with bad password	Create Valid User	betty	P4ssw0rd
	Attempt to Login with Credentials	betty	wrong
	Status Should Be	Access Denied	

keywords receive *arguments*

2 types of keywords

We can import keyword libraries for a *test case*

...and libraries may be configured, too.

Setting	Value	Value	Value
Library	Telnet	prompt=$	

This keyword comes from the imported library.

Test Case	Action	Argument	Argument	Argument
Example	Open connection	10.0.0.42	port=${25}	
	List files	options=-lh		
	List files	path=/tmp	options=-l	

Keyword	Action	Argument	Argument	Argument
List files	[Arguments]	${path}=.	${options}=	
	Execute command	ls ${options} ${path}		

This is a *user keyword*, implemented in table format.
(Think macros composed of other macros.)

Data-driven test cases

this is the name of a *test case*

these *keywords* form the test case

Test Case		Password	Expected error message
Too short password	Creating user with invalid password should fail	abCD5	${PWD INVALID LENGTH}
Too long password	Creating user with invalid password should fail	abCD567890123	${PWD INVALID LENGTH}
Password without lowercase letters	Creating user with invalid password should fail	123DEFG	${PWD INVALID CONTENT}
Password without capital letters	Creating user with invalid password should fail	abcd56789	${PWD INVALID CONTENT}
Password without numbers	Creating user with invalid password should fail	AbCdEfGh	${PWD INVALID CONTENT}
Password with special characters	Creating user with invalid password should fail	abCD56+	${PWD INVALID CONTENT}

keywords receive *arguments*

Variables

Variable	Value
${USERNAME}	janedoe
${PASSWORD}	J4n3D0e
${NEW PASSWORD}	e0D3n4J
${DATABASE FILE}	${TEMPDIR}${/}robotframework-quickstart-db.txt
${PWD INVALID LENGTH}	Password must be 7-12 characters long
${PWD INVALID CONTENT}	Password must be a combination of lowercase and uppercase letters and numbers

Specification pyramid

Principle of Symmetric Change

The best software is the one where the technical software model is aligned with the relevant business domain model. This ensures that one small change in the business domain (new requirements or changes to existing features) results in one small change in software. The same is true for the other artifacts produced for software — tests and documentation. When the tests are aligned with the business domain model, one small change in business will result in one small change in tests, making the tests easy to maintain.

Ref: The Secret Ninja Cucumber Scrolls

Test Data Syntax

Syntax Alternatives Galore

robot_test_case.html

```
<table class="example docutils" border="1">
    <thead valign="bot
        <tr>
            <th class=
                Settir
            </th>
            <th class=
```

```
*Setting*
Library Oper

*Variable*
${MESSAGE}

*Test Case*
My Test [Doc
    Log ${ME
    My Keywo

Another Test

*Keyword*
My Keyword
    Directory Should Exist  ${pat
```

Line: 23 Column: 5 Plain Text

Line: 241 Column: 1 HTML Tab Size:

robot_test_

```
***Settings***
Library     OperatingSystem

***Variables***
${MESSAGE}  Hell

***Test Cases***
My Test  [Docume
    Log
    My Keyword

Another Test
    Should Be Eq

***Keywords***
My Keyword  [Arg
    Directory Sh
```

Line: 21 Column: 1

```
 Variable
============
${MESSAGE}
============


============
 Test Case
============
My Test
\
\
\
Another Test
============
```

robot_test_case.psv

```
| *Setting*  |     *Value*     |
| Library    | OperatingSystem |

| *Variable* |     *Value*     |
| ${MESSAGE} | Hello, world!   |

| *Test Case*  | *Action*         | *Argument*   |
| My Test      | [Documentation]  | Example test |
|              | Log              | ${MESSAGE}   |
|              | My Keyword       | /tmp         |
| Another Test | Should Be Equal  | ${MESSAGE}   | Hello,
world!

| *Keyword*  |
| My Keyword | [Arguments] | ${path}
|            | Directory Should Exist | ${path}
```

Line: 20 Column: 1 Plain Text Tab Size: 4

```
============  =====================  ============  ==========
 Keyword            Action             Argument     Argument
============  =====================  ============  ==========
My Keyword    [Arguments]            ${path}
\             Directory Should Exist ${path}
============  =====================  ============  ==========
```

Line: 32 Column: 1 Plain Text Tab Size: 4

RIDE

RIDE

File Edit Tools Navigate Help

Login Tests
- *Higher Level Login
  - Higher Level Valid Login
  - Even Higher Level Valid
  - Highest Level Login
- Invalid Login
- Simple Login

Resources
- resource.html

Higher Level Valid Login

Documentation		Edit	Clear
Setup		Edit	Clear
Teardown		Edit	Clear
Tags		Edit	Clear
Timeout		Edit	Clear

Add Row Add Column

1	Input Username	demo			
2	Input Password	mode			
3	Click Login Button				
4	Welcome Page Should Be Open				
5	Should Be Em				
6	Should Be Empty	BuiltIn			
7					
8					
9					

Source: BuiltIn <test library>

Arguments: [item | msg=None]

Verifies that the given item is empty.

The length of the item is got using the `Get Length` keyword. The
default error message can be overridden with the

Built by:

```
<target name="robot" depends="init">
    <exec executable="jybot">
        <env key="CLASSPATH" value="${robot.test.classpath}"/>
        <arg line="-C off"/>
        <arg value="-d"/>
        <arg path="target/robot"/>
        <arg value="-v"/>
        <arg value="jetty.port:${jetty.port}"/>
        <arg path="src/test/acceptance/my_test_case.html"/>
    </exec>
</target>
```

Built by:

maven Robot Framework Maven Plugin

```
<plugin>
   <groupId>org.robotframework</groupId>
   <artifactId>robotframework-maven-plugin</artifactId>
   <version>1.2</version>
   <executions>
      <execution>
         <goals>
            <goal>run</goal>
         </goals>
      </execution>
   </executions>
</plugin>
```

http://robotframework.org/MavenPlugin

Selenium Library

Robot Architecture

Test Data (Tables)

Robot Framework

Test Library API

Test Libraries

Robot comes with a number of built-in test libraries and you can (should!) add your own.

Test Tools

Test libraries can use any test tool necessary to interact with the system under test.

application interfaces

System Under Test

Selenium

Selenium Remote Control v2.0 [a1] - Mozilla Firefox
File Edit View History Bookmarks Tools Help
chrome://src/content/RemoteRunner.html?sessionId=6500a2ada9b649ad85a3c1
Most Visited
Getting Started
Latest Headlines
Selenium Remote Control v2.0 ...
Selenium Functional Testing for Web Apps
Open Source From ThoughtWorks and Friends
Show Log
Slow Mode
6500a2ada9b649ad85a3c1627efa9ae2
Command History:
setContext(6500a2ada9b649ad85a3c1627efa9ae2)
open(/)
type(q, hello world)
click(btnG)
waitForPageToLoad(10000)
hello world - Google Search - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.google.co.in/#hl=en&source=hp&q=hello+world&btnG=Google+Se
Most Visited
Getting Started
Latest Headlines
hello world - Google Search
Web Images Maps News Orkut Books Gmail more
Web History | Search settings | Sign in
Google
hello world
Search
About 216,000,000 results (0.11 seconds)
Advanced search
Everything
News
Discussions
Shopping
Hello world program - Wikipedia, the free encyclopedia
A "Hello World" program is a computer program which prints out "Hello World!" on a display device. It is used in many introductory tutorials for teaching a ...
Purpose - History - Variations - See also
en.wikipedia.org/wiki/Hello_world_program - Cached - Similar
Done

Selenium launches two browsers: one that runs the browser-based Selenium Core and one for interacting with the application under test

Test speaks selenese to Selenium:
"open browser, http://www.google.com"

Test

Browser(s)

Selenium Server acts as an HTTP proxy for the browser

Selenium

Application

Choose Cancel On Next Confirmation · Choose File · Click Button · Click Element · Cli
Close All Browsers · Close Browser · Close Window · Confirm Action · Current Frame C
Current Frame Should Contain · Delete All Cookies · Delete Cookie · Drag And Drop ·
Element Should Be Visible · Element Should Contain · Element Should Not Be Visible · E
· Execute Javascript · Focus · Frame Should Contain · Frame Should Contain Text · Ge
Get All Links · Get Cookie Value · Get Cookies · Get Element Attribute · Get Horizontal
Get List Items · Get Location · Get Matching Xpath Count · Get Source · Get Table Cell
· Get Value · Get Vertical Position · Get Window Identifiers · Get Window Names · Get
Go Back · Go To · Input Password · Input Text · List Selection Should Be · List Should
Location Should Be · Location Should Contain · Log Source · Maximize Browser Window
Mouse Down On Image · ouse Over · Mouse Up
Open Context Menu · Button · Page Should
Page Should Contain E Should Contain Link ·
Page Should Contain L age Should Contain Tex
Page Should Not Conta Should Not Contain Cl
Page Should Not Contain Element · Page Should Not Contain Image · Page Should Not Co
Page Should Not Contain List · Page Should Not Contain Radio Button · Page Should Not
Press Key · Press Key Native · Radio Button Should Be Set To · Radio Button Should Not
Register Keyword To Run On Failure · Select All From List · Select Checkbox · Select Fr
· Select Radio Button · Select Window · Set Selenium Speed · Set Selenium Timeout · Si
Start Selenium Server · Stop Selenium Server · Submit Form · Switch Browser · Table C
Table Column Should Contain · Table Footer Should Contain · Table Header Should Cont
Table Row Should Contain · Table Should Contain · Textfield Should Contain · Textfield
Title Should Be · Unselect Checkbox · Unselect Frame · Unselect From List · Wait For C
Wait Until Page Contains · Wait Until Page Contains Element · Wait Until Page Loaded ·

Robot Framework hides (most of) this infrastructure behind simple keywords

Email Delivery Example

Email Delivered Acceptance Test

```
*** Settings ***
Library        check_mail.py

*** Variables ***
${MAIL_SERVER}     imap.gmail.com

*** Test Cases ***
Email Delivered Acceptance Rule
    [Template]  Confirm Email Delivered Workflow
    sender@mail.com       user@example.com  3asyp3asy  1
    sender@nonexist.com   user@example.com  3asyp3asy  0

*** Keywords ***
Confirm Email Delivered Workflow
    [Arguments]    ${sender}  ${recipient}  ${password}  ${number_of_emails_expected}
    Open Mail Box    ${MAIL_SERVER}  ${recipient}   ${password}
    Count Mail Received    ${sender}  ${number_of_emails_expected}

Count Mail Received
    [Arguments]    ${sender}  ${mail_expected}
    ${actual_amount_of_mail}=    Count Mail  UNSEEN  FROM  ${sender}
    Should Be Equal As Integers    ${mail_expected}  ${actual_amount_of_mail}
```

Wrong Account Acceptance Test

```
*** Settings ***
Library        check_mail.py

*** Variables ***
${MAIL_SERVER}     imap.gmail.com

*** Test Cases ***
Wrong Account Acceptance Rule
    [Template]  Wrong Account Workflow
    receiver@notexist.com  3asyp3asy       Could not connect to the mailbox
    user@example.com       wrong_password  Could not connect to the mailbox

*** Keywords ***
Wrong Account Workflow
    [Arguments]    ${recipient}  ${password}  ${expected_error}
    Open Mail Box    ${MAIL_SERVER}  ${recipient}   ${password}
    Check Mail Error Message    ${expected_error}

Check Mail Error Message
    [Arguments]    ${expected_error_message}
    ${actual_error_message}=    Get Mail Error Message
    Should Be Equal    ${actual_error_message}  ${expected_error_message}
```

Implement a Library

```
import imaplib

class check_mail:

    ROBOT_LIBRARY_SCOPE = 'GLOBAL'
    __version__ = '0.1'
    error_message = ""

    def open_mail_box(self, server, user, pwd):
        try:
            self.mailbox = imaplib.IMAP4_SSL(server)
            self.mailbox.login(user, pwd)
            self.mailbox.select()
        except imaplib.IMAP4.error:
            self.error_message = 'Could not connect to the mailbox'

    def count_mail(self, *args):
        r, self.item = self.mailbox.search(None, *args)
        return len(self.item[0].split())

    def get_mail_error_message(self):
        return self.error_message

    def close_mailbox(self):
        self.mailbox.close()
        self.mailbox.logout()
```

```java
public class MailLibrary {
	public static final String ROBOT_LIBRARY_SCOPE = "GLOBAL";
	public static final String ROBOT_LIBRARY_VERSION = "1.0";

	private Store store;
	private Folder folder;

	public MailLibrary() {
	}

	public void openMailbox(String host, String username, String password) throws MessagingException {
		Properties properties = new Properties();
		properties.setProperty("mail.imap.socketFactory.class", "javax.net.ssl.SSLSocketFactory");
		properties.setProperty("mail.imap.socketFactory.fallback", "false");
		properties.setProperty("mail.imap.socketFactory.port", "993");

		Session session = Session.getDefaultInstance(properties);
		store = session.getStore("imap");
		store.connect(host, username, password);
	}

	public void checkMailbox() throws MessagingException {
		FlagTerm flagUnseen = new FlagTerm(new Flags(Flags.Flag.SEEN), false);
		SearchTerm searchTerm = new AndTerm(flagUnseen, new FromStringTerm("sender@mail.com"));

		folder = store.getFolder("INBOX");
		folder.open(Folder.READ_ONLY);

		Message messages[] = folder.search(searchTerm);

		if (messages.length == 0) {
			throw new MessagingException("No mail found");
		}
	}

	public void closeMailbox() throws MessagingException {
		if (folder != null) {
			folder.close(false);
		}
		if (store != null) {
			store.close();
		}
	}
}
```

Given-when-then (BDD)

```
*** Test Cases ***
Addition
    Given calculator has been cleared
    When user types "1 + 1"
    and user pushes equals
    Then result is "2"

*** Keywords ***
Calculator has been cleared
    Push button    C

User types "${expression}"
    Push buttons    ${expression}

User pushes equals
    Push button    =

Result is "${result}"
    Result should be    ${result}
```

Useful Tips

- Use 4 spaces between keyword and arguments
- Open with Robotframework text editor for syntax highlight

Useful Tips (2)

- Indicate full qualified class name for libraries in Java

```
*** Settings ***
Library    com.mail.testlibrary.Server
```

- Use [tags] and set --noncritical option for tests under development

```
--noncritical not_ready
```

```
Wrong Account Acceptance Rule
    [Template]    Wrong Account Workflow
    [Tags]    not_ready
    receiver@notexist.com  3asyp3asy       Could not connect to the mailbox
    user@example.com       wrong_password  Could not connect to the mailbox
```

Pair Programming

Pair Programming

- Two people working together at one computer
  - Not just for programming

- Benefits:
  - Shared understanding of what code does
  - Continuous review helps spot mistakes quicker
  - Healthy peer pressure discourages bad habits
  - Learning from others through observation and discussions about practices and techniques
  - More time making progress and less time being ‘blocked’

How to pair program

How to pair program

Enough Space

How to pair program

Think aloud

Ask questions

Tips

- Avoid dominating
- Don’t point out trivial mistakes too quickly, e.g. missing semicolon
- Design with a flip chart or large piece of paper

Team doesn’t like to pair?

- Discuss with the team how much pair programming they feel is good
  - When do they feel it is appropriate?
  - What is expected?
  - When is it optional?
  - Create a working agreement?
- Seek to improve their pairing technique
- Try:
  - Share videos of how others pair program
  - Ping-pong programming to give structure and make it interesting

Ping-Pong Programming

1

Jim writes a failing test

2

Bob writes code to pass the test

3

Jim and Bob refactor the code **together**

4

Cycle starts again:
Bob writes a new failing test and hands the keyboard back to Jim

SCM System (Subversion)

Software Configuration Management Systems

Concept

- Records the evolution of changes in your project
- Govern how your team collaborates

Benefits

- Reproduce your project's state at any time in the past
- Easy for all team members to share source code
- Know when the change was made and by whom

SUBVERSION

HTML
JAVA
commit
Frequently with all files
JAVA
update
Repository
checkout
CI System
JAVA
HTML
checkout

Some SVN Commands

Update working copy

```
svn update
```

Commit changes

```
svn commit -m "Handled parsing error"
```

Display commit messages

```
svn log Foo.java
```

Checkout project into local dir.

```
svn co http://company/repos/mailer
```

Undo local edits

```
svn revert Foo.java
```

Create repository

```
svnadmin create /var/svn/repos
```

Some Steps You Might Need

- Repository url -> **https**://earth.local/svn
- Import your project skeleton into it
- Check out a working copy

For more details, http://svnbook.red-bean.com

Automation

Repeatable build and deployment

Automated Build

Automated Build

Automated Deployment

Tools

- First widely adopted build tool specially made for Java projects
- Build file written in XML
- Imperative programming
- Simple and verbose
  - You get *nothing* by default
- Dependency management through Apache Ivy
- Lots of 3rd party plugins

maven

- First build tool to incorporate built-in support for dependency management and a standard artifact repositories
- Build file written in XML
- Declarative programming
- Convention over configuration
- Lots of 3rd party plugins

maven

- Build file written in Groovy
- Imperative and declarative programming
- Exploits the Maven ecosystem:
  - Convention over configuration
  - Dependency management
- Gradle wrapper allows builds on machines where Gradle is not installed

Samples

build.xml

```xml
<?xml version="1.0"?>
<project xmlns:ivy="antlib:org.apache.ivy.ant" name="KillerApp" default="package" basedir=".">
    <path id="build.classpath">
        <fileset dir="lib/build" includes="**/*.jar"/>
    </path>
    <target name="init" depends="ivy-dependencies">
        <mkdir dir="target/classes/production"/>
        <mkdir dir="target/classes/test"/>
    </target>
    <target name="ivy-dependencies">
        <taskdef resource="org/apache/ivy/ant/antlib.xml" uri="antlib:o
        <ivy:settings file="ivysettings.xml"/>
        <ivy:retrieve/>
        <ivy:cachepath pathid="src.classpath" conf="production"/>
        <ivy:cachepath pathid="test.classpath" conf="development"/>
    </target>
    <target name="compile" depends="init" description="compile the sour
        <javac srcdir="src" destdir="target/classes/production" classpa
        <javac srcdir="test" destdir="target/classes/test" classpathref
    </target>
    <target name="test" depends="compile" description="run unit tests">
        <mkdir dir="target/reports"/>
        <junit printsummary="yes" haltonfailure="yes" showoutput="yes">
            <classpath>
                <path refid="test.classpath"/>
                <pathelement location="target/classes/test"/>
            </classpath>
            <batchtest fork="yes" todir="target/reports">
                <formatter type="xml"/>
                <fileset dir="test" includes="**/*Test.java,**/Test*.ja
            </batchtest>
        </junit>
    </target>
    <target name="package" depends="compile, test" description="generat
        <mkdir dir="target/package"/>
        <jar jarfile="target/package/killerapp.jar" basedir="target/cla
    </target>
    <target name="clean" description="clean up">
        <delete dir="target"/>
    </target>
</project>
```

```
$ ant clean package
Buildfile: /Users/lkoskela/Desktop/CSD/Automation/ant/build.xml

clean:
  [delete] Deleting directory /Users/lkoskela/Desktop/CSD/Automation/ant/target

ivy-dependencies:
[ivy:retrieve] :: Ivy 2.2.0 - 20100923230623 :: http://ant.apache.org/ivy/ ::
[ivy:retrieve] :: loading settings :: file = /Users/lkoskela/Desktop/CSD/Automation/ant/
ivysettings.xml
[ivy:retrieve] :: resolving dependencies :: com.odde#killerapp;working@1162.local
[ivy:retrieve]      confs: [production, development]
[ivy:retrieve]      found junit#junit;4.8.2 in ibiblio
[ivy:retrieve] :: resolution report :: resolve 187ms :: artifacts dl 4ms
        ---------------------------------------------------------------------
        |                  |            modules            ||   artifacts   |
        |       conf       | number| search|dwnlded|evicted|| number|dwnlded|
        ---------------------------------------------------------------------
        |   production     |   0   |   0   |   0   |   0   ||   0   |   0   |
        |   development    |   1   |   0   |   0   |   0   ||   1   |   0   |
        ---------------------------------------------------------------------
[ivy:retrieve] :: retrieving :: com.odde#killerapp
[ivy:retrieve]      confs: [production, development]
[ivy:retrieve]      0 artifacts copied, 1 already retrieved (0kB/7ms)

init:
   [mkdir] Created dir: /Users/lkoskela/Desktop/CSD/Automation/ant/target/classes/production
   [mkdir] Created dir: /Users/lkoskela/Desktop/CSD/Automation/ant/target/classes/test

compile:
   [javac] Compiling 1 source file to /Users/lkoskela/Desktop/CSD/Automation/ant/target/classes/
production
   [javac] Compiling 1 source file to /Users/lkoskela/Desktop/CSD/Automation/ant/target/classes/
test

test:
   [mkdir] Created dir: /Users/lkoskela/Desktop/CSD/Automation/ant/target/reports
   [junit] Running com.odde.build.SampleTest
   [junit] Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0.084 sec

package:
   [mkdir] Created dir: /Users/lkoskela/Desktop/CSD/Automation/ant/target/package
     [jar] Building jar: /Users/lkoskela/Desktop/CSD/Automation/ant/target/package/killerapp.jar

BUILD SUCCESSFUL
Total time: 4 seconds
```

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.odde.build</groupId>
  <artifactId>webapp</artifactId>
  <packaging>war</packaging>
  <version>1.2</version>
  <name>Killer App</name>
  <url>http://www.killerapp.com</url>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.8.2</version>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>servlet-api</artifactId>
      <version>2.4</version>
    </dependency>
  </dependencies>

  <build>
    <finalName>killer</finalName>
  </build>
</project>
```

```
$ mvn clean package
[INFO] Scanning for projects...
[INFO] ----------------------------------------------------------------------
[INFO] Building Killer App
[INFO]    task-segment: [clean, package]
[INFO] ----------------------------------------------------------------------
[INFO] [clean:clean {execution: default-clean}]
[INFO] ...
[INFO] [compiler:compile {execution: default-compile}]
[INFO] ...
[INFO] [compiler:testCompile {execution: default-testCompile}]
[INFO] ...
[INFO] [surefire:test {execution: default-test}]
[INFO] ...
[INFO] [war:war {execution: default-war}]
[INFO] ...
maven2/src/main/webapp]
[INFO] Webapp assembled in [82 msecs]
[INFO] Building war: ./target/killer.war
[INFO] ----------------------------------------------------------------------
[INFO] BUILD SUCCESSFUL
[INFO] ----------------------------------------------------------------------
[INFO] Total time: 3 seconds
[INFO] Finished at: Wed Nov 24 11:30:26 SGT 2010
[INFO] Final Memory: 16M/81M
[INFO] ----------------------------------------------------------------------
$
```

build.gradle

```
apply plugin: 'eclipse'
apply plugin: 'war'
apply plugin: 'jetty'

repositories {
   mavenCentral()
}

dependencies {
   testCompile 'junit:junit:4.11'
   compile group: 'commons-io', name: 'commons-io', version: '1.4'
   compile group: 'log4j', name: 'log4j', version: '1.2.15', ext: 'jar'
}

httpPort = 8080
stopPort = 9451
stopKey = 'foo'
```

```
$ gradle clean build
:clean
:compileJava
:processResources
:classes
:war
:assemble
:compileTestJava
:processTestResources UP-TO-DATE
:testClasses
:test
:check
:build

BUILD SUCCESSFUL

Total time: 4.503 secs
ompleted in 5.588s
$
```

More about Maven

Before Maven

build-main.bat

different

turbine

turbine-build.sh

Before Maven

what does the project need to build?

what libraries do I download?

where do I put them?

what should I execute on the build?

Consistency

1. checkout source
2. `mvn package`

dependency management and reuse of build logic

free!

Maven Lifecycle

1. validate
2. compile
3. test
4. package
5. integration-test
6. verify
7. install
8. deploy

```
mvn package
```

will invoke validate, compile **and** test

Conventions

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-
v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.odde.build</groupId>
  <artifactId>webapp</artifactId>
  <packaging>war</packaging>
  <version>1.2</version>
  <name>Killer App</name>
  <url>http://www.killerapp.com</url>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.8.2</version>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>servlet-api</artifactId>
      <version>2.4</version>
    </dependency>
  </dependencies>

  <build>
    <finalName>killer</finalName>
  </build>
</project>
```

assumes

src/test/java

src/main/java

src/main/webapp/WEB-INF/web.xml

target/killer.war

Configuration & Repository

~/.m2/settings.xml

user-specific configuration for authentication, repositories and other Maven customization

~/.m2/repository/

your local Maven repository. Downloaded dependencies are stored here.

How to start?

Command line

1. create pom.xml
2. `mvn eclipse:eclipse`
3. Search libraries in http://search.maven.org

http://maven.apache.org/guides/mini/guide-ide-eclipse.html

Resources

Multi-module

Archetypes

http://www.sonatype.com/books/mvnref-book

Definition of Done

Definition of “done”

- Team, together with the product owner, defines what “done” means
- Examples of “done”
  - Design, coding, unit testing, integrated
  - Static analysis, refactored
  - Acceptance tested, deployable

- Done defines the current technical capability of the team
  - Over time “done” should include everything needed before deployment
- Done defines amount of risk/delay due to non-transparency for the PO

- Not done backlog items should not be reviewed
  - Only done items are calculated in velocity

TODO: Before we can ship

Need to be done before shipping

- Design & Code
- Unit test
- Functional test
- Performance test
- Usability test
- Test in sandbox production environment
- Update customer documentation
- Localize in 10 languages
- Update marketing plan
- Update deployment scripts
- Actually deploy

Weak “Definition of Done”

Can do during the sprint:

- Design & Code
- Unit test
- Functional test

Won’t do during the sprint:

- Performance test
- Usability test
- Test in sandbox production environment
- Update customer documentation
- Localize in 10 languages
- Update marketing plan
- Update deployment scripts
- Actually deploy

Planned, Undone, Risk

Can do during the sprint:

- Design & Code
- Unit test
- Functional test

Won't do during the sprint:

- Performance test
- Usability test
- Test in sandbox production environment
- Update customer documentation
- Localize in 10 languages
- Update marketing plan
- Update deployment scripts
- Actually deploy

Definition of Done

This is what is planned for during the Sprint Planning

Undone work

Not planned but must be done before the release in either
1) a 'release sprint' or
2) a different department.

This causes a lack of transparency increasing delay and risk in the release!

Strong “Definition of Done”

Can do during the sprint:

- Design & Code
- Unit test
- Functional test
- Performance test
- Usability test
- Test in sandbox production environment
- Update customer documentation
- Localize in 10 languages

Won’t do during the sprint:

- Update marketing plan
- Update deployment scripts
- Actually deploy

Extending “done”

www.craiglarman.com
www.odd-e.com

Copyright © 2010
C.Larman & B. Vodde
All rights reserved.

Sprint Planning

Sprint Planning Overview

www.craiglarman.com
www.odd-e.com

Copyright © C.Larman &
B. Vodde 2009.
All rights reserved.

Sprint Planning

Clarify Requirements

Sprint Planning Part 1

Team Capabilities

Business Conditions

Technology Stability

Previous Sprint Product Increment

Initial Design and Plan

Sprint Planning Part 2

Updated Product Backlog

Sprint Backlog

Sprint Goal

Sprint Commitment

- Made after Sprint Planning part 2
- Should be taken seriously by the team
  - Creates trust between the PO and the team
  - If not taken seriously -> problems with the team
- Software development is new product development, which means that stuff comes up!
  - Even good teams might deliver on their sprint commitment perhaps 80% of the time.

Part 1

Typical Part 1

- Agenda:
  - Introduce Product Backlog
  - Tentatively select backlog items
  - Clarify unclear items
- Tips:
  - Can use a projector, but printed or wall backlogs work well
  - Should be short due to Backlog Refinement

Exercise:
Do Sprint Planning Part 1

Part 2

Typical Agenda for Part 2

- Create initial design through a short Agile Modeling session
- Create the team's sprint calendar
- Estimate team's hour/capacity budget

- Until capacity is reached:
  - Pick an item from the product backlog
  - Generate implementation tasks
  - Estimate the tasks

- Make an explicit sprint commitment – this is what we will deliver!

Meeting Tips for Part 2

- Sprint planning is time-boxed. Do not let it flow over.
- Avoid computers, projectors, screens – use inclusive media
  - A computer might be necessary for quick research or checking facts about the current code base
- Keep up the pace – try not to dwell on an individual item for too long
- Team owns the meeting – beware of a dominating Scrum Master
- Establish a collaborative environment
- The goal is a sufficient level of certainty for making a commitment
  - You will never uncover tasks *exhaustively*
  - Estimates will never be *accurate*
  - The design will never be *perfect*

Design Tips for Part 2

- Time-box the design activity
- Involve everyone, avoid one person dominating
  - Team should reach a *common* understanding
  - Hand *everyone* a marker pen, stickies, etc.
- Model at whiteboards or flip charts
  - Focus on the essential, avoid extraneous detail
  - Try to be useful, not precise
  - Forget full-blown UML diagrams – Boxes, lines, arrows and ugly stick figures work best!

Steps in Part 2

Modeling

Sprint calendar

Good table (even though it is not round :( )

Use Index Cards

Use the Table Space

Planning Poker

Poker in Action

Calculations

Task Board (Cork)

Task Board (Post-it)

Updating Estimates

Sprint Burndown

hours remaining

Exercise:
Do Sprint Planning Part 2

Test-Driven Development

Odd-e

TESTING

I FIND YOUR LACK OF TESTS DISTURBING.

Quick Intro

TDD Cycle

- Short
- Rhythmic
- Incremental
- Design-focused
- **Disciplined**

TDD Rules

- The one rule to rule them all
  - Only ever write code to fix a failing test

- 3 Steps:
  - Write a test (which fails “red”)
  - Code (to make test pass “green”)
  - Refactor (test still pass “green”)
  - Repeat forever

Write test

TemplateEngineTest.java

```
public class TemplateEngineTest {

    @Test
    public void templatesWithoutPlaceHoldersDoNotChange() throws Exception {
        Template template = new Template();
        TemplatePlaceholderValues replacementValues = new TemplatePlaceholderValues();
        template.set("Nothing here");
        assertEquals("Nothing here", template.replaceValues(replacementValues));
    }
}
```

This does not compile!
Template and TemplatePlaceholderValues do **not** exist.

Step 2 -> Code

Template.java

```
public class Template {

    private String template;

    public void set(String template) {
        this.template = template;
    }

    public String replaceValues(TemplatePlaceholderValues replacementValues) {
        return template;
    }

}
```

TemplatePlaceholderValues.java

```
public class
TemplatePlaceholderValues {

}
```

Run the test!

Refactor

New Test

TemplateEngineTest.java

```
@Test
public void templateWithOnePlaceholderReplacesItToValue() throws Exception {
    Template template = new Template();
    TemplatePlaceholderValues replacementValues = new TemplatePlaceholderValues();
    template.set("Hi, $name");
    replacementValues.placeholder("$name").hasValue("John");
    assertEquals("Hi, John", template.replaceValues(replacementValues));
}
```

Again, won't compile!

... and implementation

Template.java

```
public String replaceValues(TemplatePlaceholderValues replacementValues) {
    return template.replace("$name", "John");
}
```

TemplatePlaceholderValues.java

```
public class TemplatePlaceholderValues {

    public Placeholder placeholder(String name) {
        return new Placeholder();
    }
}
```

Placeholder.java

```
public class Placeholder {

    public void hasValue(String value) {
    }
}
```

Run!

Refactor (test)

Pull out common objects

TemplateEngineTest.java

```
Template template = new Template();
TemplatePlaceholderValues replacementValues = new TemplatePlaceholderValues();

@Test
public void templateWithoutPlaceholderDoesNotChange() throws Exception {
Template template = new Template();
TemplatePlaceholderValues replacementValues = new TemplatePlaceholderValues();
    template.set("Nothing here");
    assertEquals("Nothing here", template.replaceValues(replacementValues));
}

@Test
public void templateWhereOnePlaceholderIsReplaced() throws Exception {
    template.set("Hi, $name");
    replacementValues.placeholder("$name").hasValue("John");
    assertEquals("Hi, John", template.replaceValues(replacementValues));
}
```

New Test -> Drive out logic

TemplateEngineTest.java

```
@Test
public void templateWhereOnePlaceholderIsReplacedAndOneIsNot() throws Exception {
    template.set("Hi, $name $last_name");
    replacementValues.placeholder("$last_name").hasValue("Doe");
    assertEquals("Hi, $name Doe", template.replaceValues(replacementValues));
}
```

More implementation

Template.java

```
public String replaceValues(TemplatePlaceholderValues replacementValues) {
    return template.replace(replacementValues.firstPlaceholder().name(),
            replacementValues.firstPlaceholder().value());
}
```

TemplatePlaceholderValues.java

```
private Placeholder placeholder = new Placeholder("");

public Placeholder placeholder(String name) {
    placeholder = new Placeholder(name);
    return placeholder;
}

public Placeholder firstPlaceholder() {
    return placeholder;
}
```

And more implementation

Placeholder.java

```
public class Placeholder {

    private String name;
    private String value = new String();

    public Placeholder(String name) {
        this.name  = name;
    }

    public void hasValue(String value) {
        this.value = value;
    }

    public String name() {
        return name;
    }

    public String value() {
        return value;
    }
}
```

Run!

Refactoring

TemplatePlaceholderValues.java

```
    private Placeholder placeholder = new Placeholder();

    public Placeholder placeholder(String name) {
placeholder = new Placeholder(name);
        placeholder.setName(name);
        return placeholder;
    }
```

Placeholder.java

```
    private String name = new String();
    private String value = new String();

    public void setName(String name) {
        this.name = name;

    }
```

New Test

TemplateEngineTest.java

```
@Test
public void templateWherePlaceholdersAreReplaced() throws Exception {
    template.set("Hi, $first_name $last_name");
    replacementValues.placeholder("$first_name").hasValue("John");
    replacementValues.placeholder("$last_name").hasValue("Doe");
    assertEquals("Hi, John Doe", template.replaceValues(replacementValues));
}
```

Compiling passed,
but test fails

org.junit.ComparisonFailure: expected:<Hi, [John] Doe> but was:<Hi, [$first_name] Doe>

Backtrack for refactoring

```
// @Test
// public void templateWherePlaceholdersAreReplaced() throws Exception {
//     template.set("Hi, $first_name $last_name");
//     replacementValues.placeholder("$first_name").hasValue("John");
//     replacementValues.placeholder("$last_name").hasValue("Doe");
//     assertEquals("Hi, John Doe", template.replaceValues(replacementValues));
// }
```

Implementation would be too complicated.
Prepare the code for the implementation first!

Template.java

```
public String replaceValues(TemplatePlaceholderValues replacementValues) {
    return template.replace(replacementValues.firstPlaceholder().name(),
            replacementValues.firstPlaceholder().value());
}
```

Would require iteration over replacementValues.
And it already smells!

Refactor before...

Template.java

```
    public String replaceValues(TemplatePlaceholderValues replacementValues) {
return template.replace(replacementValues.firstPlaceholder().name(),
                replacementValues.firstPlaceholder().value());
        return replacementValues.replaceValuesIn(template);
    }
```

TemplatePlaceholderValues.java

```
 public String replaceValuesIn(String template) {
     return template.replace(placeholder.name(), placeholder.value());
 }
```

TemplatePlaceholderValues.java

```
public Placeholder firstPlaceholder() {
    return placeholder;
}
```

And more...

TemplatePlaceholderValues.java

```
public String replaceValuesIn(String template) {
    return template.replace(placeholder.name(), placeholder.value());
    return placeholder.replaceValuesIn(template);
}
```

Placeholder.java

```
public String replaceValuesIn(String template) {
    return template.replace(name, value);
}
```

Placeholder.java

```
public String name() {
    return name;
}

public String value() {
    return value;
}
```

Still green!

Try again!

TemplateEngineTest.java

```
@Test
public void templateWherePlaceholdersAreReplaced() throws Exception {
    template.set("Hi, $first_name $last_name");
    replacementValues.placeholder("$first_name").hasValue("John");
    replacementValues.placeholder("$last_name").hasValue("Doe");
    assertEquals("Hi, John Doe", template.replaceValues(replacementValues));
}
```

Implementation

TemplatePlaceholderValues.java

```
private List<Placeholder> placeholders = new ArrayList<Placeholder>();

public Placeholder placeholder(String name) {
    Placeholder placeholder = new Placeholder();
    placeholders.add(placeholder);
    return placeholder;
}

public String replaceValuesIn(String template) {
    String replacedTemplate = template;
    for (Placeholder placeholder : placeholders) {
        replacedTemplate = placeholder.replaceValuesIn(replacedTemplate);
    }
    return replacedTemplate;
}
```

Run!

org.junit.ComparisonFailure:
expected:<[Hi,]John>
but was:<[JohnHJohniJohn,John John$JohnnJohnaJohnmJohne]John>

Mistake!

TemplatePlaceholderValues.java

```
public Placeholder placeholder(String name) {
    Placeholder placeholder = new Placeholder();
    placeholder.setName(name);
    placeholders.add(placeholder);
    return placeholder;
}
```

Forgot to set the placeholder name!

Run!

- Ability to make changes
  - Cheat if you can
  - Diving into details
- Confidence and trust in the code
  - Treading the happy path
- Easy to experiment with designs
  - Exploring uncertain territory
  - Going for the highest value
- Encourages good design
  - Clean it up later
- “If it doesn’t have to work, I can get it done a lot faster!”
- “Cost of bug fixing is lower if it is found earlier”

SYNTK (stuff you need to know)

Refactoring

Structured code transformation to prepare the code for change

Key points:

- Doesn't adjust functionality
- Small and disciplined
- Well defined
- Keeps code healthy

Code Smells

It smells!

It stinks!

A sign there is probably something wrong with your code

Example: Duplicate code

```
public DataSet navigateFileUp()
{
    int fileCount = this.db.getFileCount();
    int x = 0;
    if(fileCount <= 0) return null;
    x = (this.db.getCurrentFileIndex() + 1) % fileCount;
    if(this.db.getFileSize(x) == 0)//file still empty
        return this.db.setNewCurrentSet(x);
    return this.db.setCurrentSet(x, this.db.getCurrentCorrectAnsweredIndex(), 0);
}

public DataSet navigateFileDown()
{
    int fileCount = this.db.getFileCount();
    int x = 0;
    if(fileCount <= 0) return null;
    x = (this.db.getCurrentFileIndex() - 1 + fileCount) % fileCount;
    if(this.db.getFileSize(x) == 0)//file still empty
        return this.db.setNewCurrentSet(x);
    return this.db.setCurrentSet(x, this.db.getCurrentCorrectAnsweredIndex(), 0);
}
```

Only difference!

Refactoring visualized

Without refactoring:

Original program:

Making changes:

More changes:

With refactoring:

Small change

Refactor

Etc

Cost of change increases rapidly!

Cost of change does not increase

Example refactoring - Original Code

```
if (andNode.getFirstNode().getNodeType().alwaysTrue()) {
    // compile first node as non-expr and then second node
    compile(andNode.getFirstNode(), context, false);
    compile(andNode.getSecondNode(), context, expr);
} else if (andNode.getFirstNode().getNodeType().alwaysFalse()) {
    // compile first node only
    compile(andNode.getFirstNode(), context, expr);
} else {
    compile(andNode.getFirstNode(), context, true);
    BranchCallback longCallback = new BranchCallback() {
                public void branch(BodyCompiler context) {
                    compile(andNode.getSecondNode(), context, true);
                }
            };

    context.performLogicalAnd(longCallback);
    if (!expr) context.consumeCurrentValue();
}
```

Refactoring Example
Extract Local Variable

```
Node firstNode = andNode.getFirstNode();
if (firstNode.getNodeType().alwaysTrue()) {
    // compile first node as non-expr and then second node
    compile(firstNode, context, false);
    compile(andNode.getSecondNode(), context, expr);
 } else if (firstNode.getNodeType().alwaysFalse()) {
    // compile first node only
    compile(firstNode, context, expr);
} else {
    compile(firstNode, context, true);
    BranchCallback longCallback = new BranchCallback() {
                public void branch(BodyCompiler context) {
                    compile(andNode.getSecondNode(), context, true);
                }
            };

    context.performLogicalAnd(longCallback);
    if (!expr) context.consumeCurrentValue();
}
```

Alt-Shift-L

Refactoring Example
Extract Method

```
public void compileAnd(Node node, BodyCompiler context, final boolean expr) {
    final AndNode andNode = (AndNode) node;

    ...
    } else {
        compile(andNode.getFirstNode(), context, true);
        BranchCallback longCallback = createBranchCallbackForNode(andNode);

        context.performLogicalAnd(longCallback);
        if (!expr) context.consumeCurrentValue();
    }
}

private BranchCallback createBranchCallbackForNode(final AndNode andNode) {
    return new BranchCallback() {
                public void branch(BodyCompiler context) {
                    compile(andNode.getSecondNode(), context, true);
                }
            };
}
```

Alt-Shift-M

Refactoring Example
Inline Variable

```
public void compileAnd(Node node, BodyCompiler context, final boolean expr) {
    final AndNode andNode = (AndNode) node;

    if (andNode.getFirstNode().getNodeType().alwaysTrue()) {
        // compile first node as non-expr and then second node
        compile(andNode.getFirstNode(), context, false);
        compile(andNode.getSecondNode(), context, expr);
    } else if (andNode.getFirstNode().getNodeType().alwaysFalse()) {
        // compile first node only
        compile(andNode.getFirstNode(), context, expr);
    } else {
        compile(andNode.getFirstNode(), context, true);
        context.performLogicalAnd(createBranchCallbackForNode(andNode));
        if (!expr) context.consumeCurrentValue();
    }
}
```

Alt-Shift-I

Refactor before, Refactor after, but do **not** refactor during

```
void func1() {
  lfjlfjsd
  fsdllfsdjfds
  lkfkjdslfjd
  jfndslknds
  fldlfs
}

void func2() {
  ljfdsljd
  lkfjdslkjfs
  pweuiporewi
  mlcdmdsl
  lcdld
}

void func3() {
  lkjfsdlj
  piejiewpj
  lncdslnds
  pejwpjewp
}

void func4() {
  lnwlenw
  powjofij
  cklndslkdsn
  pewopwe
}
```

```
void func1() {
  lfjlfjsd
  fsdllfsdjfds
  fldlfs
}

void func2() {
  ljfdsljd
  mlcdmdsl
  lcdld
}

void func3() {
  lkjfsdlj
  piejiewpj
  lncdslnds
  pejwpjewp
}

void func4() {
  lnwlenwn
  pewopwe
}

void func5() {
  fsdfdsfds
  ewoihoidwneo
  kncwenkjen
  oihcdoinewn
  lkmdlcmddc
}
```

```
void func1() {
  lfjlfjsd
  fsdllfsdjfds
  fldlfs
}

void func2() {
  ljfdsljd
  mlcdmdsl
  lcdld
}

void func3() {
  lkjfsdlj
  piejiewpj
  lncdslnds
  pejwpjewp
}

void func4() {
  lnwlenwn
  pewopwe
}

void func5() {
  fsdfdsfds
  ewoihoidwneo
  kncwenkjen
  bdkbdskbdkjb
  kndckjjndskjnc
  cndkndsknw
  puwepewuuew
  weijdioejwoijde
  oinenenkekwjn
  nkjnwekjndkjw
  oihcdoinewn
  lkmdlcmddc
}
```

```
void func1() {
  lfjlfjsd
  fsdllfsdjfds
  fldlfs
}

void func2() {
  ljfdsljd
  mlcdmdsl
  lcdld
}

void func3() {
  lkjfsdlj
  piejiewpj
  lncdslnds
  pejwpjewp
}

void func4() {
  lnwlenwn
  pewopwe
}

void func5() {
  fsdfdsfds
  ewoihoidwneo
  kncwenkjen
  bdkbdskbdkjb
  kndckjjndskjnc
  cndkndsknw
  puwepewuuew
  weijdioejwoijde
  oinenenkekwjn
  nkjnwekjndkjw
  oihcdoinewn
  lkmdlcmddc
}
```

```
void func1() {
  lfjlfjsd
  fsdllfsdjfds
  fldlfs
}

void func2() {
  ljfdsljd
  mlcdmdsl
  lcdld
}

void func3() {
  lkjfsdlj
  piejiewpj
  lncdslnds
  pejwpjewp
}

void func4() {
  lnwlenwn
  pewopwe
}

void func5() {
  fsdfdsfds
  ewoihoidwneo
  kncwenkjen
}

void func6 () {
  nkjnwekjndkjw
  oihcdoinewn
  lkmdlcmddc
}
```

Doing multiple things at the same time results in a mess

1. Multiple change points
   Need to prepare for the change
2. Refactor before
   Change point now at one place
3. Make the change
   Do not refactor
4. Identify smells
   opportunities to clean up
5. Refactor after
   Clean up to keep the code in a good state

It is not a unit test when:

- It talks to the database
- It communicates across the network
- It touches the file system
- It can't run at the same time as any of your other unit tests
- You have to do special things to your environment (such as editing config files) to run it.

Michael Feathers

JUnit 4

Test class containing tests

Import JUnit package

Import assertion methods

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

public class PersonTest {

    @Test
    public void defaultNameOfThePersonIsJohn() throws Exception {
        Person john = new Person();
        assertEquals("John", john.name());
    }
}
```

Test method

Annotation for test methods

Assertion

Test Fixture

Data needed in each test.

Import Before/ After annotations

```
import org.junit.After;
import org.junit.Before;

public class PersonTest {

    Person john = new Person();

    @Before
     public void initializePerson() {
        john.initialize();
    }

    @After
    public void cleanUpPerson() {
        john.clean();
    }

    @Test
    public void defaultNameOfThePersonIsJohn() throws Exception {
        assertEquals("John", john.name());
    }
}
```

Run before each test

Run after each test

The data and setup / teardown (before, after) is also sometimes called: **test fixture**

JUnit Runtime Model

```
public class PersonTest {

    Person john = new Person();

    @Test
    public void defaultNameOfThePersonIsJohn() {
        assertEquals("John", john.name());
    }

    @Test
    public void defaultAgeOfThePersonIsTwentyTwo() {
        assertEquals(22, john.age());
    }

    @Before
     public void initializePerson() {
        john.initialize();
    }

    @After
    public void cleanUpPerson() {
        john.clean();
    }

public class AccountTest {

    @Test
    public void defaultTotalIsZero() {
        Account account = new Account();
        assertEquals(0, account.total());
    }
```

First create all objects

Then do the execution

Its design, stupid!

Modularity == Testability

Focus on tests first ensures modularity
and other good design principles

YAGNI

Developers often ask questions such as:

- “What if...”
- “Perhaps x will change”
- “How can I make it more flexible?”
- “This would be nice!”

Assume that:

You Ain’t Gonna Need It

and prevent designs from becoming complex!

Simplicity is the ultimate sophistication

Flexible complex design on paper → Something unexpected **always** changes → More complexity than needed. Hard to maintain. Legacy

---

Simple design (not simplistic) → Something unexpected **always** changes → Easier to adopt. Simple design turns out to be easier to change. Less complexity.

Design from the perspective of use rather than implementation

What is design?

"After reviewing the software development lifecycle as I understood it, I concluded that the only software documentation that actually seems to satisfy the criteria of an engineering design is the source code listings."

– Jack Reeves,
"What is Software Design", C++ Journal,
1992.

"In the end, the source code is the design."
– Robert C. Martin,
"Agile Software Development",

"Design"

Reality

Getting Started

Deciding what test to write

- Small steps, avoid long red
- Make progress
- Depth-first vs. breadth-first
- Drive out wrong code
- Where is the risk?

Your test-writing order will influence your design

Why is it difficult to sustain?

TDD:

- Steady pace
- Speed limits
- No traffic lights (bugs)
- Might **feel** slow!!

Traditional:

- Spurts
- Fast when no problems!
- Debugging
- **Feels** fast! But often slow

Sustainability

Time developers do not notice nor plan for

Traditional development

Speculate	Code	Test	Debug

Time vs

Feels slower

Test-driven development

Spec ulate	Test and Code	Debug

Collective Code Ownership

What is Collective Code Ownership?

- Anyone can make necessary changes anywhere
- Expects everyone to fix any problems they find
- Everyone shares the responsibility for the quality of the code
- Everyone communicates through code

Supporting Practices

Test Driven Development

Pair Programming

Refactoring

Continuous Integration

Benefits of Collective Code Ownership

Reduce complex code

Lesser frustration, never stuck with someone else's stupidity

Spread knowledge

Continuous Integration

What is Continuous Integration?

CI System **NOT**

Automated **the point!**

CI is a developer practice

Discipline to integrate frequently (hourly, at least daily)

CI is a developer practice

Strive to make small changes

CI is a developer practice

Strive for fast feedback

Scenario

Hey! I'm changing the method that conflicts with your changes!

Roger!

... done and committed!

Ok! I'll update and check the merged changes! ... Looks good!

Great!

CI is a practice...

it is about **what people do**, **not about what tools** they use.

As a project starts to scale, it is **easy to be deceived** into thinking that the **team is practicing continuous integration** just because **all of the tools are set up and running**.

If developers do not have the **discipline to integrate** their changes on a **regular basis** or to maintain the **integration environment in a good working order** they are **not practicing continuous integration**. Full stop.

R. Owen Rogers

Odd-e

Continuous Integration System

CI System

www.craiglarman.com
www.odd-e.com

Copyright © 2010
C.Larman & B. Vodde
All rights reserved.

Jenkins

Jenkins

search

Jenkins

ENABLE AUTO REFRESH

New Job

add description

Manage Jenkins

People

Build History

Build Queue

No builds in the queue.

Build Executor Status

#	Status
1	Idle
2	Idle

All +

S	W	Job ↓	Last Success	Last Failure	Last Duration
		Apache Maven	0.74 sec (#1)	N/A	31 ms
		Commons annotations	1 min 27 sec (#6)	N/A	0.37 sec
		Jenkins	81 ms (#1)	N/A	27 ms

Icon: S M L

Legend RSS for all RSS for failures RSS for just latest builds

Page generated: Jul 6, 2011 10:33:35 AM Jenkins ver. 1.414

Setting up Jenkins

Go to

Download

Start up

```
java -jar jenkins.war
```

For more details, http://wiki.jenkins-ci.org/display/JENKINS/Use+Jenkins

Some useful plugins

Jenkins Sounds

Build Monitor

Robotframework

Green Balls

https://wiki.jenkins-ci.org/display/JENKINS/Plugins

Some Steps You Might Need

1. Go to http://earth.local:8000

2. Create Jenkins job

Eclipse IDE
Getting productive

```
    for (int i = 0
        copy[i] =
    }
    return copy;
}
```

Programming is a text based activity

Instead of precise mousing...

prefer the keyboard when coding

Learning methods

Do not memorize a long list

Learning methods

Short-cut keys with Shift+Ctrl+L

Key promoter tools like MouseFeed

Keyboard Shortcuts

Quick Fix	Ctrl+1
General Short-cut	Ctrl+3
Delete a line	Ctrl+D
Search outline of the class	Ctrl+O
Open resource (file)	Ctrl+Shift+R
Content assist	Ctrl+Space
Extract Method (Refactor)	Shift+Alt+M
Inline (Refactor)	Shift+Alt+I
Rename (Refactor)	Shift+Alt+R
Search and open for Type	Shift+Ctrl+T
Run a new JUnit test	Shift+Alt+X T
Run last unit test	Ctrl+F11

Eclipse plugins in Marketplace

Subversive or Subclipse — SUBVERSION clients

MouseFeed — feeds you keyboard shortcuts.

M2Eclipse — a maven client

Working in team

Teams

Team

- A team comprises a group of people linked in a common purpose.

- Appropriate for conducting complex and interdependent tasks
- Team members have complementary skills and generate synergy
- Team members help one another

- A team has:
  - Shared work product
  - Interdependent work
  - Shared responsibility
  - Set of working agreements
  - Responsibility for managing the outside-team relationships
  - Distributed leadership

Real Team

- Appropriate task for teamwork
- Clear but moderately permeable membership boundaries
- Substantial but clearly delimited authority for managing its work
- Reasonably stable over time

Four Types of Work Teams

Scrum teams are here

© J. Richard Hackman: Leading Teams

Scrum Team

- Self-organizing;
- Cross-functional with no roles;
- Seven plus or minus two;
- Responsible for committing to work;
- Authority to do whatever is needed to meet commitment;
- Open, collocated space;
- Resolution of conflicts;
- Rules of Etiquette:

The Five Dysfunctions of a Team

Chartering

- At the very start of product development:
  - Clarify the vision
  - Team values
  - Establish project community
  - Create working agreements
- Not part of Scrum, still useful.

Working agreements

- Explicit statement of team accountability
- Create new team habits

- Categories:
  - Decision making
  - Timing
  - Respect
  - Technical
  - Social

- Related
  - Coding standards
  - Definition of “done”

Decision making

- How does the team decide?
  - Consensus
  - Voting
  - Specialist decides
- Without an agreement on how to make decisions, team might have lots of problems.

Participatory Decision Making

Encourage full participation

Promote mutual understanding

Foster inclusive solutions

Cultivate shared responsibility

Fist of Five

Exercise:
Create Working Agreement

Code Smells & Refactoring

Refactor based on smells

Why? How?

opportunity for
refactoring
amount of code
smells
indicates
motivation
developers
amount of bad
code
quick hacks
# of bugs
time spend on
bug fixing
panic

Refactoring

How do you know, when to refactor?

When your code smells!

Good code an opinion?

NO!

Good code does
not smell!

What about performance?

Not an excuse to
write smelly code.

Refactoring Buddi!

Messy - Debug Info

```
public long getAmount(Date startDate, Date endDate){
     if (startDate.after(endDate))
          throw new RuntimeException("Start date cannot be before End Date!");

     Logger.getLogger(this.getClass().getName()).info("Starting to calculate the budgeted amount for " + getFullName() + " between " + star
and " + endDate + ".");

     //If Start and End are in the same budget period
     if (getBudgetPeriodType().getStartOfBudgetPeriod(startDate).equals(
               getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
//        Logger.getLogger().info("Start Date and End Date are in the same period.");
          long amount = getAmount(startDate);
//        Logger.getLogger().info("Amount = " + amount);
          long daysInPeriod = getBudgetPeriodType().getDaysInPeriod(startDate);
//        Logger.getLogger().info("Days in Period = " + daysInPeriod);
          long daysBetween = DateUtil.getDaysBetween(startDate, endDate, true);
//        Logger.getLogger().info("Days Between = " + daysBetween);

//        Logger.getLogger().info("Returning " + (long) (((double) amount / (double) daysInPeriod) * daysBetween));
//        Logger.getLogger().info("Finished calculating the budget amount.\n\n");
          return (long) (((double) amount / (double) daysInPeriod) * daysBetween);
     }

     //If the area between Start and End overlap at least two budget periods.
     if (getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1).equals(
               getBudgetPeriodType().getStartOfBudgetPeriod(endDate))
               || getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1).before(
                         getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
//        Logger.getLogger().info("Start Date and End Date are in different budget periods.");
          long amountStartPeriod = getAmount(startDate);
//        Logger.getLogger().info("Amount Start Period = " + amountStartPeriod);
          long daysInStartPeriod = getBudgetPeriodType().getDaysInPeriod(startDate);
//        Logger.getLogger().info("Days in Start Period = " + daysInStartPeriod);
          long daysAfterStartDateInStartPeriod = DateUtil.getDaysBetween(startDate, getBudgetPeriodType().getEndOfBudgetPeriod(startDate),
//        Logger.getLogger().info("Days After Start Date in Start Period = " + daysAfterStartDateInStartPeriod);
          double totalStartPeriod = (((double) amountStartPeriod / (double) daysInStartPeriod) * daysAfterStartDateInStartPeriod);
//        Logger.getLogger().info("Total in Start Period = " + totalStartPeriod);

          double totalInMiddle = 0;
          for (String periodKey : getBudgetPeriods(
                    getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1),
                    getBudgetPeriodType().getBudgetPeriodOffset(endDate, -1))) {
               totalInMiddle += getAmount(getPeriodDate(periodKey));
               Logger.getLogger(this.getClass().getName()).info("Added " + getAmount(getPeriodDate(periodKey)) + " to total for one period
between; current value is " + totalInMiddle);
          }
//        Logger.getLogger().info("Total in Middle = " + totalInMiddle);

          long amountEndPeriod = getAmount(endDate);
//        Logger.getLogger().info("Amount End Period = " + amountEndPeriod);
```

Long Method

Large number of line of code
(anything over 5-10)

Class BudgetCategoryImpl

Too long!

```
public long getAmount(Date startDate, Date endDate){
     if (startDate.after(endDate))
          throw new RuntimeException("Start date cannot be before End Date!");

     //If Start and End are in the same budget period
     if (getBudgetPeriodType().getStartOfBudgetPeriod(startDate).equals(
               getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
          long amount = getAmount(startDate);
          long daysInPeriod = getBudgetPeriodType().getDaysInPeriod(startDate);
          long daysBetween = DateUtil.getDaysBetween(startDate, endDate, true);

          return (long) (((double) amount / (double) daysInPeriod) * daysBetween);
     }

     //If the area between Start and End overlap at least two budget periods.
     if (getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1).equals(
               getBudgetPeriodType().getStartOfBudgetPeriod(endDate))
               || getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1).before(
                         getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
          long amountStartPeriod = getAmount(startDate);
          long daysInStartPeriod = getBudgetPeriodType().getDaysInPeriod(startDate);
          long daysAfterStartDateInStartPeriod = DateUtil.getDaysBetween(startDate, getBudgetPeriodType().getEndOfBudgetPeriod(startDate), true);
          double totalStartPeriod = (((double) amountStartPeriod / (double) daysInStartPeriod) * daysAfterStartDateInStartPeriod);

          double totalInMiddle = 0;
          for (String periodKey : getBudgetPeriods(
                    getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1),
                    getBudgetPeriodType().getBudgetPeriodOffset(endDate, -1))) {
               totalInMiddle += getAmount(getPeriodDate(periodKey));
          }

          long amountEndPeriod = getAmount(endDate);
          long daysInEndPeriod = getBudgetPeriodType().getDaysInPeriod(endDate);
          long daysBeforeEndDateInEndPeriod = DateUtil.getDaysBetween(getBudgetPeriodType().getStartOfBudgetPeriod(endDate), endDate, true);
          double totalEndPeriod = (long) (((double) amountEndPeriod / (double) daysInEndPeriod) * daysBeforeEndDateInEndPeriod);

          return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
     }

     throw new RuntimeException("You should not be here.  We have returned all legitimate numbers from getAmount(Date, Date) in BudgetCategoryImpl.
Please contact Wyatt Olson with details on how you got here (what steps did you perform in Buddi to get this error message).");
}
```

Long method
often indicates
other smells

Safety first!

Class BudgetCategoryTest

Are there tests?
Are they any good?

```
public class BudgetCategoryTest {

        @Test
        public void testBudgetCategory(){
                try {
                        BudgetCategoryType bct = ModelFactory.getBudgetCategoryType(BudgetCategoryTypes.BUDGET_CATEGORY_TYPE_MONTH);
                        BudgetCategory bc = ModelFactory.createBudgetCategory("Test", bct, false);
                        bc.setAmount(DateUtil.getDate(2007, Calendar.APRIL, 1), 100);
                        bc.setAmount(DateUtil.getDate(2007, Calendar.MAY, 1), 200);
                        bc.setAmount(DateUtil.getDate(2007, Calendar.JUNE, 1), 240);
                        bc.setAmount(DateUtil.getDate(2007, Calendar.JULY, 1), 10);
                        bc.setAmount(DateUtil.getDate(2007, Calendar.AUGUST, 1), 130);
                        bc.setAmount(DateUtil.getDate(2007, Calendar.SEPTEMBER, 1), 13);
                        bc.setAmount(DateUtil.getDate(2007, Calendar.OCTOBER, 1), 333);
                        bc.setAmount(DateUtil.getDate(2007, Calendar.NOVEMBER, 1), 331);

                        assertEquals((double) 100, bc.getAmount(DateUtil.getDate(2007, Calendar.APRIL, 1)), 1);
                        assertEquals((double) 100, bc.getAmount(DateUtil.getDate(2007, Calendar.APRIL, 10)), 1);
                        assertEquals((double) 100, bc.getAmount(DateUtil.getDate(2007, Calendar.APRIL, 28)), 1);

                        assertEquals((double) 300, bc.getAmount(DateUtil.getDate(2007, Calendar.APRIL, 1), DateUtil.getDate(2007, Calendar.MAY, 31)), 1);
                        assertEquals((double) 149, bc.getAmount(DateUtil.getDate(2007, Calendar.APRIL, 15), DateUtil.getDate(2007, Calendar.MAY, 15)), 1);

                }
                catch (Exception e){
                        fail("Exception: " + e);
                }
        }

        @Test
        public void budgetPeriodWeekly() throws Exception {
                BudgetCategoryType bct = ModelFactory.getBudgetCategoryType(BudgetCategoryTypes.BUDGET_CATEGORY_TYPE_WEEK);
                BudgetCategory bc = ModelFactory.createBudgetCategory("Weekly Test", bct, false);

                bc.setAmount(DateUtil.getDate(2007, Calendar.SEPTEMBER, 10), 100);
                bc.setAmount(DateUtil.getDate(2007, Calendar.SEPTEMBER, 17), 100);
                bc.setAmount(DateUtil.getDate(2007, Calendar.SEPTEMBER, 24), 100);
                bc.setAmount(DateUtil.getDate(2007, Calendar.OCTOBER, 1), 100);
                bc.setAmount(DateUtil.getDate(2007, Calendar.OCTOBER, 8), 100);
                bc.setAmount(DateUtil.getDate(2007, Calendar.OCTOBER, 15), 100);
                bc.setAmount(DateUtil.getDate(2007, Calendar.OCTOBER, 22), 100);
                bc.setAmount(DateUtil.getDate(2007, Calendar.OCTOBER, 29), 100);
                bc.setAmount(DateUtil.getDate(2007, Calendar.NOVEMBER, 5), 100);
                bc.setAmount(DateUtil.getDate(2007, Calendar.NOVEMBER, 12), 100);

                assertEquals(100l,
                                bc.getAmount(DateUtil.getDate(2007, Calendar.OCTOBER, 1)));
                assertEquals(14l,
                                bc.getAmount(
                                                DateUtil.getDate(2007, Calendar.OCTOBER, 1),
                                                DateUtil.getDate(2007, Calendar.OCTOBER, 1)));
                assertEquals(28l,
                                bc.getAmount(
                                                DateUtil.getDate(2007, Calendar.OCTOBER, 1),
                                                DateUtil.getDate(2007, Calendar.OCTOBER, 2)));
                assertEquals(100l,
                                bc.getAmount(
                                                bct.getStartOfBudgetPeriod(DateUtil.getDate(2007, Calendar.OCTOBER, 1)),
                                                bct.getEndOfBudgetPeriod(DateUtil.getDate(2007, Calendar.OCTOBER, 1))));
                assertEquals(200l,
                                bc.getAmount(
                                                bct.getStartOfBudgetPeriod(DateUtil.getDate(2007, Calendar.OCTOBER, 1)),
                                                bct.getEndOfBudgetPeriod(bct.getBudgetPeriodOffset(DateUtil.getDate(2007, Calendar.OCTOBER, 1), 1))));
                assertEquals(400l,
                                bc.getAmount(
                                                bct.getStartOfBudgetPeriod(DateUtil.getDate(2007, Calendar.OCTOBER, 1)),
                                                bct.getEndOfBudgetPeriod(bct.getBudgetPeriodOffset(DateUtil.getDate(2007, Calendar.OCTOBER, 1), 3))));
```

Some tests,
not very good ones.

Add tests

```
@Test (expected=RuntimeException.class)
public void testBeginDateIsLaterThanStartDateThrowsAnException() throws Exception {
    bc.getAmount(fourthOfJuly2011 , secondOfJuly2011);
}

@Test
public void getAmountOfARangeInsideABudgetPeriod() throws Exception {
    bc.setAmount(secondOfJuly2011, 31);
    assertEquals(3, bc.getAmount(secondOfJuly2011, fourthOfJuly2011));
}

@Test
public void getAmountOfARangeTwoNeigbouringBudgetPeriods() throws Exception {
    bc.setAmount(twentyJune2011, 300);
    bc.setAmount(secondOfJuly2011, 31);
    assertEquals(114, bc.getAmount(twentyJune2011, fourthOfJuly2011));
}

@Test
public void getAmountOfARangeWithManyBudgetPeriodsInBetween() throws Exception {
    bc.setAmount(tenthMarch2011, 31);
    bc.setAmount(twentyApril2011, 3000);
    bc.setAmount(twentyMay2011, 3100);
    bc.setAmount(twentyJune2011, 3000);
    bc.setAmount(secondOfJuly2011, 31);
    assertEquals(9100 + 2 + 22, bc.getAmount(tenthMarch2011, secondOfJuly2011));
}
```

Buckled? Ready?

Duplicate code

Two pieces of code that are conceptually the same

Class BudgetCategoryImpl

```
        //If Start and End are in the same budget period
        if (getBudgetPeriodType().getStartOfBudgetPeriod(startDate).equals(
                getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
            long amount = getAmount(startDate);
            long daysInPeriod = getBudgetPeriodType().getDaysInPeriod(startDate);
            long daysBetween = DateUtil.getDaysBetween(startDate, endDate, true);

            return (long) (((double) amount / (double) daysInPeriod) * daysBetween);
        }

        //If the area between Start and End overlap at least two budget periods.
        if (getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1).equals(
                getBudgetPeriodType().getStartOfBudgetPeriod(endDate))
                || getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1).before(
                        getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
            long amountStartPeriod = getAmount(startDate);
            long daysInStartPeriod = getBudgetPeriodType().getDaysInPeriod(startDate);
            long daysAfterStartDateInStartPeriod = DateUtil.getDaysBetween(startDate,
getBudgetPeriodType().getEndOfBudgetPeriod(startDate), true);
            double totalStartPeriod =
(((double) amountStartPeriod / (double) daysInStartPeriod) * daysAfterStartDateInStartPeriod);

            double totalInMiddle = 0;
            for (String periodKey : getBudgetPeriods(
                    getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1),
                    getBudgetPeriodType().getBudgetPeriodOffset(endDate, -1))) {
                totalInMiddle += getAmount(getPeriodDate(periodKey));
            }

            long amountEndPeriod = getAmount(endDate);
            long daysInEndPeriod = getBudgetPeriodType().getDaysInPeriod(endDate);
            long daysBeforeEndDateInEndPeriod =
DateUtil.getDaysBetween(getBudgetPeriodType().getStartOfBudgetPeriod(endDate), endDate, true);
            double totalEndPeriod =
(long) (((double) amountEndPeriod / (double) daysInEndPeriod) * daysBeforeEndDateInEndPeriod);

            return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
        }
```

First make duplication obvious (same)

```
long amount = getAmount(startDate);
long daysInPeriod = getBudgetPeriodType().getDaysInPeriod(startDate);
long daysBetween = DateUtil.getDaysBetween(startDate, endDate, true);

return (long) (((double) amount / (double) daysInPeriod) * daysBetween);
```

Extract Local Variable (Alt-Shift-L)

```
long amount = getAmount(startDate);
long daysInPeriod = getBudgetPeriodType().getDaysInPeriod(startDate);
long daysBetween = DateUtil.getDaysBetween(startDate, endDate, true);
double amountInPeriod = (long) (((double) amount / (double) daysInPeriod) * daysBetween);

return (long) amountInPeriod;
```

```
            long amountStartPeriod = getAmount(startDate);
            long daysInStartPeriod = getBudgetPeriodType().getDaysInPeriod(startDate);
            long daysAfterStartDateInStartPeriod =
DateUtil.getDaysBetween(startDate, getBudgetPeriodType().getEndOfBudgetPeriod(startDate), true);
            double totalStartPeriod =
(((double) amountStartPeriod / (double) daysInStartPeriod) * daysAfterStartDateInStartPeriod);
```

Extract Local Variable (Alt-Shift-L)

```
            Date endOfBudgetPeriod = getBudgetPeriodType().getEndOfBudgetPeriod(startDate);
            long amountStartPeriod = getAmount(startDate);
            long daysInStartPeriod = getBudgetPeriodType().getDaysInPeriod(startDate);
            long daysAfterStartDateInStartPeriod = DateUtil.getDaysBetween(startDate, endOfBudgetPeriod, true);
            double totalStartPeriod =
(((double) amountStartPeriod / (double) daysInStartPeriod) * daysAfterStartDateInStartPeriod);
```

Class BudgetCategoryImpl

Remove useless parenthesis

```
double totalStartPeriod =
(((double) amountStartPeriod / (double) daysInStartPeriod) * daysAfterStartDateInStartPeriod);
```

```
double totalStartPeriod =
((double) amountStartPeriod / (double) daysInStartPeriod) * daysAfterStartDateInStartPeriod;
```

```
double totalEndPeriod =
(long) (((double) amountEndPeriod / (double) daysInEndPeriod) * daysBeforeEndDateInEndPeriod);
```

Casts without rounding?

```
double totalEndPeriod =
((double) amountEndPeriod / (double) daysInEndPeriod) * daysBeforeEndDateInEndPeriod;
```

Without this, Eclipse can't detect the duplication :(

```
            long amountEndPeriod = getAmount(endDate);
            long daysInEndPeriod = getBudgetPeriodType().getDaysInPeriod(endDate);
            long daysBeforeEndDateInEndPeriod =
DateUtil.getDaysBetween(getBudgetPeriodType().getStartOfBudgetPeriod(endDate), endDate, true);
            double totalEndPeriod =
(long) ((double) amountEndPeriod / (double) daysInEndPeriod) * daysBeforeEndDateInEndPeriod;
```

Extract Local Variable (Alt-Shift-L)

```
            Date startOfBudgetPeriod = getBudgetPeriodType().getStartOfBudgetPeriod(endDate);
            long amountEndPeriod = getAmount(startOfBudgetPeriod);
            long daysInEndPeriod = getBudgetPeriodType().getDaysInPeriod(startOfBudgetPeriod);
            long daysBeforeEndDateInEndPeriod =
ateUtil.getDaysBetween(startOfBudgetPeriod, endDate, true);
            double totalEndPeriod =
long) ((double) amountEndPeriod / (double) daysInEndPeriod) * daysBeforeEndDateInEndPeriod;
```

Class BudgetCategoryImpl

Use start instead of end of period.

```
long amountEndPeriod = getAmount(endDate);
long daysInEndPeriod = getBudgetPeriodType().getDaysInPeriod(endDate);
long daysBeforeEndDateInEndPeriod = DateUtil.getDaysBetween(startOfBudgetPeriod, endDate, true);
```

```
long amountEndPeriod = getAmount(startOfBudgetPeriod);
long daysInEndPeriod = getBudgetPeriodType().getDaysInPeriod(startOfBudgetPeriod);
long daysBeforeEndDateInEndPeriod = DateUtil.getDaysBetween(startOfBudgetPeriod, endDate, true);
```

Note: Parameter of getAmount and getDaysInPeriod is a Period, not a Date.
Therefore both start and end date of period will work!

Class BudgetCategoryImpl

Duplication is exactly the same now.

```
if (getBudgetPeriodType().getStartOfBudgetPeriod(startDate).equals(
          getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
     long amount = getAmount(startDate);
     long daysInPeriod = getBudgetPeriodType().getDaysInPeriod(startDate);
     long daysBetween = DateUtil.getDaysBetween(startDate, endDate, true);
     double amountInPeriod = ((double) amount / (double) daysInPeriod) * daysBetween;
     return (long) amountInPeriod;
}

//If the area between Start and End overlap at least two budget periods.
if (getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1).equals(
     getBudgetPeriodType().getStartOfBudgetPeriod(endDate))
     || getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1).before(
                getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
     Date endOfBudgetPeriod = getBudgetPeriodType().getEndOfBudgetPeriod(startDate);
     long amountStartPeriod = getAmount(startDate);
     long daysInStartPeriod = getBudgetPeriodType().getDaysInPeriod(startDate);
     long daysAfterStartDateInStartPeriod = DateUtil.getDaysBetween(startDate, endOfBudgetPeriod, true);
     double totalStartPeriod = ((double) amountStartPeriod / (double) daysInStartPeriod) * daysAfterStartDateInStartPeriod;

     double totalInMiddle = 0;
     for (String periodKey : getBudgetPeriods(
               getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1),
               getBudgetPeriodType().getBudgetPeriodOffset(endDate, -1))) {
          totalInMiddle += getAmount(getPeriodDate(periodKey));
     }

     Date startOfBudgetPeriod = getBudgetPeriodType().getStartOfBudgetPeriod(endDate);
     long amountEndPeriod = getAmount(startOfBudgetPeriod);
     long daysInEndPeriod = getBudgetPeriodType().getDaysInPeriod(startOfBudgetPeriod);
     long daysBeforeEndDateInEndPeriod = DateUtil.getDaysBetween(startOfBudgetPeriod, endDate, true);
     double totalEndPeriod = ((double) amountEndPeriod / (double) daysInEndPeriod) * daysBeforeEndDateInEndPeriod;
     return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
}
```

Then extract duplication

Extract Method
Alt-Shift-M

```
//If Start and End are in the same budget period
if (getBudgetPeriodType().getStartOfBudgetPeriod(startDate).equals(
        getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
    double amountInPeriod = getAmountInPeriod(startDate, endDate);
    return (long) amountInPeriod;
}

//If the area between Start and End overlap at least two budget periods.
if (getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1).equals(
        getBudgetPeriodType().getStartOfBudgetPeriod(endDate))
    || getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1).before(
            getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
    Date endOfBudgetPeriod = getBudgetPeriodType().getEndOfBudgetPeriod(startDate);
    double totalStartPeriod = getAmountInPeriod(startDate, endOfBudgetPeriod);

    double totalInMiddle = 0;
    for (String periodKey : getBudgetPeriods(
            getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1),
            getBudgetPeriodType().getBudgetPeriodOffset(endDate, -1))) {
        totalInMiddle += getAmount(getPeriodDate(periodKey));
    }

    Date startOfBudgetPeriod = getBudgetPeriodType().getStartOfBudgetPeriod(endDate);
    double totalEndPeriod = getAmountInPeriod(startOfBudgetPeriod, endDate);
    return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
}
```

Good Principles

DRY
Don’t Repeat Yourself

Once And Only Once

DO NOT COPY PASTE

Magic Numbers

A constant appears in the code

1 - 1

```
//If the area between Start and End overlap at least two budget periods.
if (getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1).equals(
        getBudgetPeriodType().getStartOfBudgetPeriod(endDate))
        || getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1).before(
                getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
    Date endOfBudgetPeriod = getBudgetPeriodType().getEndOfBudgetPeriod(startDate);
    double totalStartPeriod = getAmountInPeriod(startDate, endOfBudgetPeriod);

    double totalInMiddle = 0;
    for (String periodKey : getBudgetPeriods(
            getBudgetPeriodType().getBudgetPeriodOffset(startDate, 1),
            getBudgetPeriodType().getBudgetPeriodOffset(endDate, -1))) {
        totalInMiddle += getAmount(getPeriodDate(periodKey));
    }

    Date startOfBudgetPeriod = getBudgetPeriodType().getStartOfBudgetPeriod(endDate);
    double totalEndPeriod = getAmountInPeriod(startOfBudgetPeriod, endDate);
    return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
}
```

Class BudgetCategoryImpl

Extract to method

```
//If the area between Start and End overlap at least two budget periods.
if (getBudgetPeriodType().getStartOfNextBudgetPeriod(startDate).equals(
        getBudgetPeriodType().getStartOfBudgetPeriod(endDate))
        || getBudgetPeriodType().getStartOfNextBudgetPeriod(startDate).before(
                getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
    Date endOfBudgetPeriod = getBudgetPeriodType().getEndOfBudgetPeriod(startDate);
    double totalStartPeriod = getAmountInPeriod(startDate, endOfBudgetPeriod);

    double totalInMiddle = 0;
    for (String periodKey : getBudgetPeriods(
            getBudgetPeriodType().getStartOfNextBudgetPeriod(startDate),
            getBudgetPeriodType().getStartOfPreviousBudgetPeriod(startDate))) {
        totalInMiddle += getAmount(getPeriodDate(periodKey));
    }

    Date startOfBudgetPeriod = getBudgetPeriodType().getStartOfBudgetPeriod(endDate);
    double totalEndPeriod = getAmountInPeriod(startOfBudgetPeriod, endDate);
    return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
}
```


Extracted to class BudgetCategoryType

Class BudgetCategoryImpl

Additional cleanup

```
//If Start and End are in the same budget period
if (getBudgetPeriodType().getStartOfBudgetPeriod(startDate).equals(
        getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
    return (long) getAmountInPeriod(startDate, endDate);
}

//If the area between Start and End overlap at least two budget periods.
if (getBudgetPeriodType().getStartOfNextBudgetPeriod(startDate).equals(
        getBudgetPeriodType().getStartOfBudgetPeriod(endDate))
        || getBudgetPeriodType().getStartOfNextBudgetPeriod(startDate).before(
                getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
    double totalStartPeriod = getAmountInPeriod(startDate, getBudgetPeriodType().getEndOfBudgetPeriod(startDate));

    double totalInMiddle = 0;
    for (String periodKey : getBudgetPeriods(
            getBudgetPeriodType().getStartOfNextBudgetPeriod(startDate),
            getBudgetPeriodType().getStartOfPreviousBudgetPeriod(endDate))) {
        totalInMiddle += getAmount(getPeriodDate(periodKey));
    }

    double totalEndPeriod = getAmountInPeriod(getBudgetPeriodType().getStartOfBudgetPeriod(endDate), endDate);
    return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
}
```

Inline Temp - Alt-Shift-I

Data Clumps

A couple of items frequently appear together

Class BudgetCategoryImpl

Note: Also earlier, we saw the smell that Date was used as Period

```
public long getAmount(Date startDate, Date endDate){
    if (startDate.after(endDate))
        throw new RuntimeException("Start date cannot be before End Date!");

    //If Start and End are in the same budget period
    if (getBudgetPeriodType().getStartOfBudgetPeriod(startDate).equals(
            getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
        return (long) getAmountInPeriod(startDate, endDate);
    }

    //If the area between Start and End overlap at least two budget periods.
    if (getBudgetPeriodType().getStartOfNextBudgetPeriod(startDate).equals(
            getBudgetPeriodType().getStartOfBudgetPeriod(endDate))
            || getBudgetPeriodType().getStartOfNextBudgetPeriod(startDate).before(
                    getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
        double totalStartPeriod = getAmountInPeriod(startDate,
                getBudgetPeriodType().getEndOfBudgetPeriod(startDate));

        double totalInMiddle = 0;
        for (String periodKey : getBudgetPeriods(
                getBudgetPeriodType().getStartOfNextBudgetPeriod(startDate),
                getBudgetPeriodType().getStartOfPreviousBudgetPeriod(endDate))) {
            totalInMiddle += getAmount(getPeriodDate(periodKey));
        }

        double totalEndPeriod = getAmountInPeriod(getBudgetPeriodType().getStartOfBudgetPeriod(endDate),
                endDate);
        return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
    }

    throw new RuntimeException("You should not be here.  We have returned all legitimate numbers from getAmount(Dat
BudgetCategoryImpl.  Please contact Wyatt Olson with details on how you got here (what steps did you perform in Buddi to
error message).");
}
```

Class BudgetCategoryImpl

Add new abstraction

```
public long getAmount(Date startDate, Date endDate){
    if (startDate.after(endDate))
        throw new RuntimeException("Start date cannot be before End Date!");

    Period period = new Period(startDate, endDate);
```

```
public class Period {

    private final Date start;
    private final Date end;

    public Period(Date start, Date end) {
        this.start = start;
        this.end = end;
    }

}
```

Abstraction not yet used...

Class Period

Class BudgetCategoryImpl

Use new abstraction

```
//If Start and End are in the same budget period
if (getBudgetPeriodType().getStartOfBudgetPeriod(period.getStartDate()).equals(
        getBudgetPeriodType().getStartOfBudgetPeriod(period.getEndDate()))){
    return (long) getAmountInPeriod(period.getStartDate(), period.getEndDate());
}

//If the area between Start and End overlap at least two budget periods.
if (getBudgetPeriodType().getStartOfNextBudgetPeriod(period.getStartDate()).equals(
        getBudgetPeriodType().getStartOfBudgetPeriod(period.getEndDate()))
        || getBudgetPeriodType().getStartOfNextBudgetPeriod(period.getStartDate()).before(
                getBudgetPeriodType().getStartOfBudgetPeriod(period.getEndDate()))){
    double totalStartPeriod = getAmountInPeriod(period.getStartDate(),
            getBudgetPeriodType().getEndOfBudgetPeriod(period.getStartDate()));

    double totalInMiddle = 0;
    for (String periodKey : getBudgetPeriods(
            getBudgetPeriodType().getStartOfNextBudgetPeriod(period.getStartDate()),
            getBudgetPeriodType().getStartOfPreviousBudgetPeriod(period.getEndDate()))) {
        totalInMiddle += getAmount(getPeriodDate(periodKey));
    }

    double totalEndPeriod =
getAmountInPeriod(getBudgetPeriodType().getStartOfBudgetPeriod(period.getEndDate()),
            period.getEndDate());
    return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
}
```

Makes the code uglier....

Class BudgetCategoryImpl

Extract Method

Extract Method
Alt-Shift-M

```
private long getAmount(Period period) {
    //If Start and End are in the same budget period
    if (getBudgetPeriodType().getStartOfBudgetPeriod(period.getStartDate()).equals(
            getBudgetPeriodType().getStartOfBudgetPeriod(period.getEndDate()))){
        return (long) getAmountInPeriod(period.getStartDate(), period.getEndDate());
    }

    //If the area between Start and End overlap at least two budget periods.
    if (getBudgetPeriodType().getStartOfNextBudgetPeriod(period.getStartDate()).equals(
            getBudgetPeriodType().getStartOfBudgetPeriod(period.getEndDate()))
            || getBudgetPeriodType().getStartOfNextBudgetPeriod(period.getStartDate()).before(
                    getBudgetPeriodType().getStartOfBudgetPeriod(period.getEndDate()))){
        double totalStartPeriod = getAmountInPeriod(period.getStartDate(),
                getBudgetPeriodType().getEndOfBudgetPeriod(period.getStartDate()));

        double totalInMiddle = 0;
        for (String periodKey : getBudgetPeriods(
                getBudgetPeriodType().getStartOfNextBudgetPeriod(period.getStartDate()),
                getBudgetPeriodType().getStartOfPreviousBudgetPeriod(period.getEndDate()))) {
            totalInMiddle += getAmount(getPeriodDate(periodKey));
        }

        double totalEndPeriod = getAmountInPeriod(getBudgetPeriodType().getStartOfBudgetPeriod(period.getEndDate()),
                period.getEndDate());
        return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
    }

    throw new RuntimeException("You should not be here.  We have returned all legitimate numbers from
getAmount(Date, Date) in BudgetCategoryImpl.  Please contact Wyatt Olson with details on how you got here (what steps
did you perform in Buddi to get this error message).");
}
```

And the original method

```
public long getAmount(Date startDate, Date endDate){
    if (startDate.after(endDate))
        throw new RuntimeException("Start date cannot be before End Date!");

    return getAmount(new Period(startDate, endDate));
}
```

Requires no changes in the calling code.. yet

Data Clumps and other smells often suggest a common problem:

Missing domain objects!

Uncommunicative Name & Inconsistent Names

Ambiguous names

```
private double getAmountInPeriod(Date startDate, Date endDate) {
    long amount = getAmount(startDate);
    long daysInPeriod = getBudgetPeriodType().getDaysInPeriod(startDate);
    long daysBetween = DateUtil.getDaysBetween(startDate, endDate, true);
    double amountInPeriod = ((double) amount / (double) daysInPeriod) * daysBetween;
    return amountInPeriod;
}
```

The BudgetCategory class now contains:

3 methods called getAmount
1 method called getAmounts
1 method called getAmountInPeriod

(and we caused 2 of these!)

Rename

```
public Map<String, Long> getAmounts()  ⟶  public Map<String, Long> getBudgetPeriods()

public void setAmounts(Map<String, Long> amounts)  ⟶  public void setBudgetPeriods(Map<String, Long> amounts)

public long getAmount(Date periodDate)  ⟶  public long getAmountFromBudgetPeriodContainingDate(Date periodDate)

public long getAmount(Date startDate, Date endDate) ⟶ public long getTotalAmountPeriod(Date startDate, Date endDate)

private long getAmount(Period period)  ⟶  private long getTotalAmountPeriod(Period period)

private double getAmount(Date startDate, Date endDate)

        ⟶  private double getAmountForPeriodWithinBudgetPeriodOfStartDate(Date startDate, Date endDate)
```

New names suggest new smells and domain objects

Comments

Explain how bad code works

```java
public long getTotalAmountPeriod(Period period){
    Date startDate = period.getStartDate();
    Date endDate = period.getEndDate();

    //If Start and End are in the same budget period
    if (getBudgetPeriodType().getStartOfBudgetPeriod(startDate).equals(
            getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
        return (long) getAmountInPeriod(startDate, endDate);
    }

    //If the area between Start and End overlap at least two budget periods.
    if (getBudgetPeriodType().getStartOfNextBudgetPeriod(startDate).equals(
            getBudgetPeriodType().getStartOfBudgetPeriod(endDate))
            || getBudgetPeriodType().getStartOfNextBudgetPeriod(startDate).before(
                    getBudgetPeriodType().getStartOfBudgetPeriod(endDate))){
        double totalStartPeriod = getAmountInPeriod(startDate, getBudgetPeriodType().getEndOfBudgetPeriod(startDate));

        double totalInMiddle = 0;
        for (String periodKey : getBudgetPeriods(
                getBudgetPeriodType().getStartOfNextBudgetPeriod(startDate),
                getBudgetPeriodType().getStartOfPreviousBudgetPeriod(endDate))) {
            totalInMiddle += getAmount(getPeriodDate(periodKey));
        }

        double totalEndPeriod = getAmountInPeriod(getBudgetPeriodType().getStartOfBudgetPeriod(endDate), endDate);
        return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
    }


    throw new RuntimeException("...");
}
```

Comments are deodorant

They hide other smells

Not all comments are bad.

Comments that describe code because it is unreadable by itself -> Bad

Comments that describe why the code works this way -> Sometimes ok!

Lazy Class

A class isn’t doing much

Class BudgetCategoryTypeMonthly

```
public class BudgetCategoryTypeMonthly extends BudgetCategoryType {

    public Date getStartOfBudgetPeriod(Date date) {
        return DateUtil.getStartOfMonth(date);
    }

    public Date getEndOfBudgetPeriod(Date date) {
        return DateUtil.getEndOfMonth(date);
    }

    public Date getBudgetPeriodOffset(Date date, int offset) {
        return getStartOfBudgetPeriod(DateUtil.addMonths(DateUtil.getStartOfMonth(date), 1 * offset));
    }

    public long getDaysInPeriod(Date date) {
        return DateUtil.getDaysInMonth(date);
    }

    public String getDateFormat() {
        return "MMM yyyy";
    }

    public String getName() {
        return BudgetCategoryTypes.BUDGET_CATEGORY_TYPE_MONTH.toString();
    }
}
```

Type in the classname also smells

Class BudgetCategoryImpl

Welcome BudgetPeriod!

```
private long getTotalAmountPeriod(Period period) {

    BudgetPeriod firstBudgetPeriod = createBudgetPeriodFromDate(period.getStartDate());

    //If Start and End are in the same budget period
    if (getBudgetPeriodType().getStartOfBudgetPeriod(period.getStartDate()).equals(
            getBudgetPeriodType().getStartOfBudgetPeriod(period.getEndDate()))){
        return (long) getAmountForPeriodWithinBudgetPeriodOfStartDate(period.getStartDate(), period.getEndDate());
    }
```

Abstraction suggested by:

- new method names
- lazy class
- comments

Use BudgetPeriod

```
BudgetPeriod firstBudgetPeriod = createBudgetPeriodFromDate(period.getStartDate());
BudgetPeriod lastBudgetPeriod = createBudgetPeriodFromDate(period.getEndDate());

if (firstBudgetPeriod.equals(lastBudgetPeriod)){
    return (long) getAmountForPeriodWithinBudgetPeriodOfStartDate(period.getStartDate(), period.getEndDate());
}
```

Now useless comment deleted

And first implementation

```
public class BudgetPeriod {

    private Period period;

    public BudgetPeriod(BudgetCategoryType type, Date date) {
        period = new Period (type.getStartOfBudgetPeriod(date), type.getEndOfBudgetPeriod(date));
    }

    @Override
    public boolean equals(Object object) {
        BudgetPeriod otherBudgetPeriod = (BudgetPeriod) object;
        return otherBudgetPeriod.period.equals(this.period);
    }

}
```

Yes, BudgetCategoryType is still there!
Removing will be gradual

Replace getBudgetPeriodType with BudgetPeriod

```
private long getTotalAmountPeriod(Period period) {

    BudgetPeriod firstBudgetPeriod = createBudgetPeriodFromDate(period.getStartDate());
    BudgetPeriod lastBudgetPeriod = createBudgetPeriodFromDate(period.getEndDate());

    if (firstBudgetPeriod.equals(lastBudgetPeriod))
        return (long) getAmountForPeriodWithinBudgetPeriodOfStartDate(period.getStartDate(), period.getEndDate());

    //If the area between Start and End overlap at least two budget periods.
    if (firstBudgetPeriod.nextBudgetPeriod().getStartDate().equals(lastBudgetPeriod.getStartDate())
            || firstBudgetPeriod.nextBudgetPeriod().getStartDate().before(lastBudgetPeriod.getStartDate())){
            double totalStartPeriod = getAmountForPeriodWithinBudgetPeriodOfStartDate(period.getStartDate(),
                    firstBudgetPeriod.getEndDate());

            double totalInMiddle = 0;
            for (String periodKey : getBudgetPeriods(
                    firstBudgetPeriod.nextBudgetPeriod().getStartDate(),
                    lastBudgetPeriod.previousBudgetPeriod().getStartDate())) {
                totalInMiddle += getAmountFromBudgetPeriodContainingDate(getPeriodDate(periodKey));
            }

            double totalEndPeriod = getAmountForPeriodWithinBudgetPeriodOfStartDate(lastBudgetPeriod.getStartDate(),
period.getEndDate());
            return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
        }

        throw new RuntimeException("You should not be here.  We have returned all legitimate numbers from
getAmount(Date, Date) in BudgetCategoryImpl.  Please contact Wyatt Olson with details on how you got here (what steps
did you perform in Buddi to get this error message).");
    }
```

Added implementation

```
public BudgetPeriod nextBudgetPeriod() {
    return new BudgetPeriod (type, type.getBudgetPeriodOffset(period.getStartDate(), 1));
}

public Date getStartDate() {
    return period.getStartDate();
}

public Date getEndDate() {
    return period.getEndDate();
}

public BudgetPeriod previousBudgetPeriod() {
    return new BudgetPeriod (type, type.getBudgetPeriodOffset(period.getStartDate(), -1));
}
```

And removed nextBudgetPeriod which was added to BudgetCategoryType while removing magic numbers.
Functionality has moved to BudgetCategory

Dead code

Code that is not used or not useful

Periods and BudgetPeriods

First if-statement

Second if-statement

Second if-statement

No other options!

Class BudgetCategoryImpl

```
private long getTotalAmountPeriod(Period period) {

    BudgetPeriod firstBudgetPeriod = createBudgetPeriodFromDate(period.getStartDate());
    BudgetPeriod lastBudgetPeriod = createBudgetPeriodFromDate(period.getEndDate());

    if (firstBudgetPeriod.equals(lastBudgetPeriod))
        return (long) getAmountForPeriodWithinBudgetPeriodOfStartDate(period.getStartDate(), period.getEndDate());

    //If the area between Start and End overlap at least two budget periods.
    if (firstBudgetPeriod.nextBudgetPeriod().getStartDate().equals(lastBudgetPeriod.getStartDate())
            || firstBudgetPeriod.nextBudgetPeriod().getStartDate().before(lastBudgetPeriod.getStartDate())){
        double totalStartPeriod = getAmountForPeriodWithinBudgetPeriodOfStartDate(period.getStartDate(),
                firstBudgetPeriod.getEndDate());

        double totalInMiddle = 0;
        for (String periodKey : getBudgetPeriods(
                firstBudgetPeriod.nextBudgetPeriod().getStartDate(),
                lastBudgetPeriod.previousBudgetPeriod().getStartDate())) {
            totalInMiddle += getAmountFromBudgetPeriodContainingDate(getPeriodDate(periodKey));
        }

        double totalEndPeriod = getAmountForPeriodWithinBudgetPeriodOfStartDate(lastBudgetPeriod.getStartDate(),
eriod.getEndDate());
        return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
    }

    throw new RuntimeException("You should not be here.  We have returned all legitimate numbers from
etAmount(Date, Date) in BudgetCategoryImpl.  Please contact Wyatt Olson with details on how you got here (what steps
id you perform in Buddi to get this error message).");
}
```

TRUE! It cannot possibly come here :)

Removed dead code

```
private long getTotalAmountPeriod(Period period) {

    BudgetPeriod firstBudgetPeriod = createBudgetPeriodFromDate(period.getStartDate());
    BudgetPeriod lastBudgetPeriod = createBudgetPeriodFromDate(period.getEndDate());

    if (firstBudgetPeriod.equals(lastBudgetPeriod))
        return (long) getAmountForPeriodWithinBudgetPeriodOfStartDate(period.getStartDate(), period.getEndDate());

    double totalStartPeriod = getAmountForPeriodWithinBudgetPeriodOfStartDate(period.getStartDate(),
            firstBudgetPeriod.getEndDate());

    double totalInMiddle = 0;
    for (String periodKey : getBudgetPeriods(
            firstBudgetPeriod.nextBudgetPeriod().getStartDate(),
            lastBudgetPeriod.previousBudgetPeriod().getStartDate())) {
        totalInMiddle += getAmountFromBudgetPeriodContainingDate(getPeriodDate(periodKey));
    }

    double totalEndPeriod = getAmountForPeriodWithinBudgetPeriodOfStartDate(lastBudgetPeriod.getStartDate(),
period.getEndDate());
    return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
}
```

Class BudgetCategoryImpl

Remove another Data Clump and Odd Name

```
private long getTotalAmountPeriod(Period period) {

    BudgetPeriod firstBudgetPeriod = createBudgetPeriodFromDate(period.getStartDate());
    BudgetPeriod lastBudgetPeriod = createBudgetPeriodFromDate(period.getEndDate());

    if (firstBudgetPeriod.equals(lastBudgetPeriod))
        return (long) getAmountForPeriodWithinBudgetPeriod(period, firstBudgetPeriod);

    double totalStartPeriod = getAmountForPeriodWithinBudgetPeriod(new Period (period.getStartDate(),
            firstBudgetPeriod.getEndDate()), firstBudgetPeriod);

    double totalInMiddle = 0;
    for (String periodKey : getBudgetPeriods(
            firstBudgetPeriod.nextBudgetPeriod().getStartDate(),
            lastBudgetPeriod.previousBudgetPeriod().getStartDate())) {
        totalInMiddle += getAmountFromBudgetPeriodContainingDate(getPeriodDate(periodKey));
    }

    double totalEndPeriod = getAmountForPeriodWithinBudgetPeriod(new Period(lastBudgetPeriod.getStartDate(),
            period.getEndDate()), lastBudgetPeriod);
    return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
}
```

Yes, still ugly...

Feature Envy

More interested in another class than its own

Requesting too much information from Periods

```
private double getAmountForPeriodWithinBudgetPeriod(Period period, BudgetPeriod firstBudgetPeriod) {
    long amount = getAmountFromBudgetPeriodContainingDate(firstBudgetPeriod.getStartDate());
    long daysInPeriod = getBudgetPeriodType().getDaysInPeriod(firstBudgetPeriod.getStartDate());
    long daysBetween = DateUtil.getDaysBetween(period.getStartDate(), period.getEndDate(), true);
    double amountInPeriod = ((double) amount / (double) daysInPeriod) * daysBetween;
    return amountInPeriod;
}
```

Move functionality to Period and BudgetPeriod

```
private double getAmountForPeriodWithinBudgetPeriod(Period period, BudgetPeriod firstBudgetPeriod) {
    long amount = getAmountFromBudgetPeriodContainingDate(firstBudgetPeriod.getStartDate());
    long daysInPeriod = firstBudgetPeriod.getAmountOfDays();
    long daysBetween = period.getAmountOfDays();
    double amountInPeriod = ((double) amount / (double) daysInPeriod) * daysBetween;
    return amountInPeriod;
}
```

Additional Cleanup

```
private double getAmountForPeriodWithinBudgetPeriod(Period period, BudgetPeriod firstBudgetPeriod) {
    long amount = getAmountFromBudgetPeriodContainingDate(firstBudgetPeriod.getStartDate());
    long daysInPeriod = firstBudgetPeriod.getAmountOfDays();
    long daysBetween = period.getAmountOfDays();
    double amountInPeriod = ((double) amount / (double) daysInPeriod) * daysBetween;
    return amountInPeriod;
}
```

Inline Temp - Alt-Shift-I

```
private double getAmountForPeriodWithinBudgetPeriod(Period period, BudgetPeriod firstBudgetPeriod) {
    long amount = getAmountFromBudgetPeriod(firstBudgetPeriod);
    long daysInPeriod = firstBudgetPeriod.getAmountOfDays();
    long daysBetween = period.getAmountOfDays();
    return ((double) amount / (double) daysInPeriod) * daysBetween;
}
```

Primitive Obsession

Use of primitives in higher-level abstraction methods

```
    double totalStartPeriod = getAmountForPeriodWithinBudgetPeriod(new Period (period.getStartDate(),
            firstBudgetPeriod.getEndDate()), firstBudgetPeriod);

    double totalInMiddle = 0;
    for (String periodKey : getBudgetPeriods(
            firstBudgetPeriod.nextBudgetPeriod().getStartDate(),
            lastBudgetPeriod.previousBudgetPeriod().getStartDate())) {
        totalInMiddle += getAmountFromBudgetPeriodContainingDate(getPeriodDate(periodKey));
    }

    double totalEndPeriod = getAmountForPeriodWithinBudgetPeriod(new Period(lastBudgetPeriod.getStartDate(),
            period.getEndDate()), lastBudgetPeriod);
    return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
}
```

```
double totalInMiddle = 0;
for (BudgetPeriod budgetPeriod :
        getBudgetPeriods(firstBudgetPeriod.nextBudgetPeriod(), lastBudgetPeriod.previousBudgetPeriod())) {
    totalInMiddle += getAmountFromBudgetPeriod(budgetPeriod);
}
```

String -> BudgetPeriod

```
public List<String> getBudgetPeriods(Date startDate, Date endDate){
    List<String> budgetPeriodKeys = new LinkedList<String>();

    Date temp = getBudgetPeriodType().getStartOfBudgetPeriod(startDate);

    while (temp.before(getBudgetPeriodType().getEndOfBudgetPeriod(endDate))){
        budgetPeriodKeys.add(getPeriodKey(temp));
        temp = getBudgetPeriodType().getBudgetPeriodOffset(temp, 1);
    }

    return budgetPeriodKeys;
}
```

```
private List<BudgetPeriod> getBudgetPeriods(BudgetPeriod firstBudgetPeriod, BudgetPeriod lastBudgetPeriod) {
    List<BudgetPeriod> budgetPeriodKeys = new LinkedList<BudgetPeriod>();

    BudgetPeriod currentBudgetPeriod = firstBudgetPeriod;

    while (currentBudgetPeriod.getStartDate().before(lastBudgetPeriod.getEndDate())){
        budgetPeriodKeys.add(currentBudgetPeriod);
        currentBudgetPeriod = currentBudgetPeriod.nextBudgetPeriod();
    }

    return budgetPeriodKeys;
}
```

Class BudgetPeriod!!!!

And move it to BudgetPeriod (feature envy)

```
public List<BudgetPeriod> createBudgetPeriodListTill(BudgetPeriod lastBudgetPeriod) {
    List<BudgetPeriod> budgetPeriodKeys = new LinkedList<BudgetPeriod>();

    BudgetPeriod currentBudgetPeriod = this;

    while (currentBudgetPeriod.getStartDate().before(lastBudgetPeriod.getEndDate())){
        budgetPeriodKeys.add(currentBudgetPeriod);
        currentBudgetPeriod = currentBudgetPeriod.nextBudgetPeriod();
    }

    return budgetPeriodKeys;
}
```

Eclipse
Move Method

And the call

```
double totalInMiddle = 0;
for (BudgetPeriod budgetPeriod :
        firstBudgetPeriod.nextBudgetPeriod().createBudgetPeriodListTill(lastBudgetPeriod.previousBudgetPeriod()))

    totalInMiddle += getAmountFromBudgetPeriod(budgetPeriod);
}
```

Class BudgetCategoryImpl

Domain objects

Primitive Obsession is one of the most common and important smells.

It often suggests missing domain objects!

Abstraction Distraction

Use of different abstraction levels in the same code

```java
private long getTotalAmountPeriod(Period period) {

    BudgetPeriod firstBudgetPeriod = createBudgetPeriodFromDate(period.getStartDate());
    BudgetPeriod lastBudgetPeriod = createBudgetPeriodFromDate(period.getEndDate());

    if (firstBudgetPeriod.equals(lastBudgetPeriod))
        return (long) getAmountForPeriodWithinBudgetPeriod(period, firstBudgetPeriod);

    double totalStartPeriod = getAmountForPeriodWithinBudgetPeriod(new Period (period.getStartDate(),
            firstBudgetPeriod.getEndDate()), firstBudgetPeriod);

    double totalInMiddle = 0;
    for (BudgetPeriod budgetPeriod :
            firstBudgetPeriod.nextBudgetPeriod().createBudgetPeriodListTill(lastBudgetPeriod.previousBudgetPeriod())) {
        totalInMiddle += getAmountFromBudgetPeriod(budgetPeriod);
    }

    double totalEndPeriod = getAmountForPeriodWithinBudgetPeriod(new Period(lastBudgetPeriod.getStartDate(),
            period.getEndDate()), lastBudgetPeriod);
    return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
}
```

```java
BudgetPeriod firstBudgetPeriod = createFirstBudgetPeriod(period);
BudgetPeriod lastBudgetPeriod = createLastBudgetPeriod(period);
```

Class BudgetCategoryImpl

```
double totalStartPeriod = getAmountForPeriodWithinBudgetPeriod(new Period (period.getStartDate(),
        firstBudgetPeriod.getEndDate()), firstBudgetPeriod);
```

```
double totalStartPeriod = getAmountForOverlappingDays(period, firstBudgetPeriod);
```

And new getAmountForOverlappingDays

```
private double getAmountForOverlappingDays(Period period, BudgetPeriod firstBudgetPeriod) {
    long amount = getAmountFromBudgetPeriod(firstBudgetPeriod);
    long daysInPeriod = firstBudgetPeriod.getAmountOfDays();
    long daysBetween = period.getNumberOfOverlappingDays(firstBudgetPeriod.getPeriod());
    return ((double) amount / (double) daysInPeriod) * daysBetween;
}
```

And new getNumberOfOverlappingDays in Period

```
public long getNumberOfOverlappingDays(Period period) {
    Date largestStartDate = (start.after(period.start)) ? start : period.start;
    Date smallestEndDate = (end.before(period.end)) ? end : period.end;

    if (smallestEndDate.before(largestStartDate))
        return 0;
    return new Period(largestStartDate, smallestEndDate).getAmountOfDays();
}
```

Class Period

Use more generic version

```
if (firstBudgetPeriod.equals(lastBudgetPeriod))
    return (long) getAmountForPeriodWithinBudgetPeriod(period, firstBudgetPeriod);

double totalStartPeriod = getAmountForOverlappingDays(period, firstBudgetPeriod);

double totalInMiddle = 0;
for (BudgetPeriod budgetPeriod :
        firstBudgetPeriod.nextBudgetPeriod().createBudgetPeriodListTill(lastBudgetPeriod.previousBudgetPeriod()))

    totalInMiddle += getAmountFromBudgetPeriod(budgetPeriod);
}

double totalEndPeriod = getAmountForPeriodWithinBudgetPeriod(new Period(lastBudgetPeriod.getStartDate(),
        period.getEndDate()), lastBudgetPeriod);
```

And delete getAmountForPeriodWithinBudgetPeriod

```
if (firstBudgetPeriod.equals(lastBudgetPeriod))
    return (long) getAmountForOverlappingDays(period, firstBudgetPeriod);

double totalStartPeriod = getAmountForOverlappingDays(period, firstBudgetPeriod);

double totalInMiddle = 0;
for (BudgetPeriod budgetPeriod :
        firstBudgetPeriod.nextBudgetPeriod().createBudgetPeriodListTill(lastBudgetPeriod.previousBudgetPeriod())) {
    totalInMiddle += getAmountForOverlappingDays(period, budgetPeriod);
}

double totalEndPeriod = getAmountForOverlappingDays(period, lastBudgetPeriod);
```

Hmm...

```
private long getTotalAmountPeriod(Period period) {

    BudgetPeriod firstBudgetPeriod = createFirstBudgetPeriod(period);
    BudgetPeriod lastBudgetPeriod = createLastBudgetPeriod(period);

    if (firstBudgetPeriod.equals(lastBudgetPeriod))
        return (long) getAmountForOverlappingDays(period, firstBudgetPeriod);

    double totalStartPeriod = getAmountForOverlappingDays(period, firstBudgetPeriod);

    double totalInMiddle = 0;
    for (BudgetPeriod budgetPeriod :
            firstBudgetPeriod.nextBudgetPeriod().createBudgetPeriodListTill(lastBudgetPeriod.previousBudgetPeriod())) {
        totalInMiddle += getAmountForOverlappingDays(period, budgetPeriod);
    }

    double totalEndPeriod = getAmountForOverlappingDays(period, lastBudgetPeriod);
    return (long) (totalStartPeriod + totalInMiddle + totalEndPeriod);
}
```

Last refactoring made the first and last budgetPeriod the same, so we can delete the checks

Final version (?)

```
private long getTotalAmountPeriod(Period period) {

    BudgetPeriod firstBudgetPeriod = createFirstBudgetPeriod(period);
    BudgetPeriod lastBudgetPeriod = createLastBudgetPeriod(period);

    double total = 0;
    for (BudgetPeriod budgetPeriod :  firstBudgetPeriod.createBudgetPeriodListTill(lastBudgetPeriod)) {
        total += getAmountForOverlappingDays(period, budgetPeriod);
    }

    return (long) total;
}
```

Probably not. Possible future directions:

- Introduce Budget class (does not exist!)
- Introduce Money class (does not exist!)
- Remove the BudgetCategoryType
- Many more primitive obsessions

PAUSE

Refactorings so far

- Extract Local Variable (alt-shift-L)
  - Introduce a local variable and initialize it with a piece of code.
- Extract Method (alt-shift-M)
  - Replace a piece of code with a call to a new method
- Inline Temp (alt-shift-I)
  - Replace a variable with the code that initialized it
  - Opposite of Extract Local Variable
- Extract Class
  - Extract a piece of code into a separate class
- Preserve Whole Object
  - Pass a whole object rather than parts of it
- Move Method (alt-shift-V)
  - Move a method from one class to another

Divergent Change

Need to change the same class for different reasons

Whatever changes

↓

We'll need to edit **this** code.

Shotgun Surgery

Responsibility is split among several classes

Whatever changes

↓

We'll always need to edit in **many different** places.

Do your tests require shotgun surgery?

Managers

Words in the name that do not tell much about its responsiblity

What do managers do?

```
class GlobalDataHandler extends DefaultHandler {
    private GlobalData data;
    // used to save temporary object data
    private Stack<Object> tempData;
    ...

}
```

GlobalDataHandler with GlobalData and a Stack of Objects????

Long Parameter List

More than one or two parameters

```
    boolean addSubject(String pId, String userid, String code, String title, String comment) throws Exception {
        CollabDb db = null;
        String sql = "";
        try {
            db = new CollabDb();
            Statement stmt = db.getStatement();
            SQLRenderer r = new SQLRenderer();
...
    }
```

Often indicates
other smells, such as
Data Clumps

Arrows

Nested if statements generate an arrow shape

```java
private static ArrayList<DataSet>[] parseDBFile(BufferedReader readFile) throws DBFileParseException
{
    if(readFile != null)
    {
        String line = readLine(readFile);
        String expression = "";
        if(line != null)
        {
            if(line.startsWith("<TRAINERDB>")) {
                // About 150 LOC deleted
            }

            else throw new DBFileParseException("Fehler in erster Zeile");
        }
        else throw new DBFileParseException("Datei leer");
    }
    throw new DBFileParseException("Datei fehlt");
}
```

Guard Clause

```
private static ArrayList<DataSet>[] parseDBFile2(BufferedReader readFile) throws DBFileParseException
{
    if(readFile == null)
        throw new DBFileParseException("Datei leer");

    String line = readLine(readFile);
    String expression = "";
    if(line == null)
        throw new DBFileParseException("Fehler in erster Zeile");
    if(line.startsWith("<TRAINERDB>")) {
        // About 150 LOC deleted
    }
}
```

Arrows can often be resolved with a revert-if refactoring and introducing a Guard Clause

They often also indicate other smells such as Long Method

Single entry - Single Exit

One major cause of arrows is developers strictly following the “single-entry, single-exit” principles (often taught at schools)

When keeping methods small, single-exit is not very relevant.

Null Checks

Many occurrences of checks for null

Remember: Design by Contract

```
public boolean accept(File f) {
     if(f != null) {
         if(f.isDirectory()) {
         return true;
         }
         String extension = getExtension(f);
         if(extension != null && mExtensions.contains(extension)) {
         return true;
         };
     }
     return false;
}
```

Null checks such as above ought to be ensured by the caller.

Design by Contract!

Primitive Obsession

Use of primitives in higher-level abstraction methods

```java
public static Card createCard(List row, int rowNum) throws InvalidCsvException, IOException {

    if (row.size() == 1 && row.get(0).equals("")) {
        // ignore blank row
        return null;
    }

    if (row.size() < 5) {
        throw new InvalidCsvException("Expected 5 columns: Reserved, Reserved, Display Name, " +
                "Reserved, and Email Address.  Found " + row.size() + " columns on row " + rowNum);
    }

    String displayName = row.get(2).toString().trim();
    String address = row.get(4).toString().trim();

    if (address.indexOf("@") == -1) {
        throw new InvalidCsvException("Error found on row " + rowNum + "The following string is not " +
                "an email address in column 5: " + address);
    }

    if ("".equals(displayName)) {
        displayName = address;
    }

    Card card = new Card();
    card.setDisplayName(displayName);
    card.setEmailAddress(address);

    return card;
}
```

Domain objects

Primitive Obsession is one of the most common and important smells.

It often suggests missing domain objects!

```java
public static Card createCard(CardFileRow row) throws InvalidCsvException, IOException {

    if (row.isValid()) {
        return null;
    }

    if (!row.emailAddressIsValid()) {
        throw new InvalidCsvException("Error found on row " + row.number() + "The following string is not " +
                "an email address in column 5: " + row.getEmailAddress().toString());
    }

    String displayName = row.getDisplayName();
    if (row.getDisplayName().isEmpty()) {
        displayName = row.getEmailAddress().toString();
    }

    Card card = new Card();
    card.setDisplayName(displayName);
    card.setEmailAddress(row.getEmailAddress());

    return card;
}
```

Introduced domain object
CardFileRow and EmailAddress

Upfront Design

A common cause of missing domain objects is to do design in “the design phase”

During implementation, new things are discovered, but “the design is already done”..
Causing... primitive obsession

Data Class

A class consisting of only data

```java
public class SubTransaction implements Serializable, Cloneable {
    private static final long serialVersionUID = 1L;

    private double amount;
    private String description;
    private Category category;

    public SubTransaction(double amount, String description, Category category) {
        super();
        this.amount = amount;
        this.description = description;
        this.category = category;
    }
    public double getAmount() {
        return amount;
    }
    public String getDescription() {
        return description;
    }
    public Category getCategory() {
        return category;
    }
    @Override
    public Object clone() {
        return new SubTransaction(amount, description, category);
    }

    @Override
    public String toString() {
        return "[{"+description+"}{"+category+"}{"+amount+"}]";
    }
}
```

Temporary!

Data classes are common when the domain objects start to emerge. Over time they ought to get logic, otherwise it is a smell.

Boredom

Feeling bored doing a task

```java
private void  fillBookToChapters() {
    bookToChapters.put(new Integer(1), new Integer(50)); //genesis
    bookToChapters.put(new Integer(2), new Integer(40)); //exodus
    bookToChapters.put(new Integer(3), new Integer(27)); //leviticus
    bookToChapters.put(new Integer(4), new Integer(36)); //numbers
    bookToChapters.put(new Integer(5), new Integer(34)); //deuteronomy
    bookToChapters.put(new Integer(6), new Integer(24)); //joshua
    bookToChapters.put(new Integer(7), new Integer(21)); //judges
    bookToChapters.put(new Integer(8), new Integer(4)); //ruth
    bookToChapters.put(new Integer(9), new Integer(31)); //1 samuel
    bookToChapters.put(new Integer(10), new Integer(24)); //2 samuel
    bookToChapters.put(new Integer(11), new Integer(22)); //1 kings
    bookToChapters.put(new Integer(12), new Integer(25)); //2 kings
    bookToChapters.put(new Integer(13), new Integer(29)); //1 chronicles
    bookToChapters.put(new Integer(14), new Integer(36)); //2 chronicles
    bookToChapters.put(new Integer(15), new Integer(10)); //ezra
    bookToChapters.put(new Integer(16), new Integer(13)); //nehemia
    bookToChapters.put(new Integer(17), new Integer(10)); //esther
    bookToChapters.put(new Integer(18), new Integer(42)); //job
    bookToChapters.put(new Integer(19), new Integer(150)); //psalms
    bookToChapters.put(new Integer(20), new Integer(31)); //proverbs
    bookToChapters.put(new Integer(21), new Integer(12)); //ecclesiastes
    bookToChapters.put(new Integer(22), new Integer(8)); //song of songs
    bookToChapters.put(new Integer(23), new Integer(66)); //isaiah
    bookToChapters.put(new Integer(24), new Integer(52)); //jeremiah
    bookToChapters.put(new Integer(25), new Integer(5)); //lamentations
    bookToChapters.put(new Integer(26), new Integer(48)); //ezekiel
    bookToChapters.put(new Integer(27), new Integer(12)); //daniel
    bookToChapters.put(new Integer(28), new Integer(14)); //hosea
    bookToChapters.put(new Integer(29), new Integer(3)); //joel
    bookToChapters.put(new Integer(30), new Integer(9)); //amos
    bookToChapters.put(new Integer(31), new Integer(0)); //obadiah
    bookToChapters.put(new Integer(32), new Integer(4)); //jonah
    bookToChapters.put(new Integer(33), new Integer(7)); //micah
    bookToChapters.put(new Integer(34), new Integer(3)); //nahum
    bookToChapters.put(new Integer(35), new Integer(3)); //habakkuk
    bookToChapters.put(new Integer(36), new Integer(3)); //zephaniah
```

```java
/**
 * Fills the HashMap of bookToChapters
 */
private void  fillBookToChapters() {
    bookToChapters.put(new Integer(1), new Integer(50)); //genesis
    bookToChapters.put(new Integer(2), new Integer(40)); //exodus
    bookToChapters.put(new Integer(3), new Integer(27)); //leviticus
    bookToChapters.put(new Integer(4), new Integer(36)); //numbers
    bookToChapters.put(new Integer(5), new Integer(34)); //deuteronomy
    bookToChapters.put(new Integer(6), new Integer(24)); //joshua
    bookToChapters.put(new Integer(7), new Integer(21)); //judges
    bookToChapters.put(new Integer(8), new Integer(4)); //ruth
    bookToChapters.put(new Integer(9), new Integer(31)); //1 samuel
    bookToChapters.put(new Integer(10), new Integer(24)); //2 samuel
    bookToChapters.put(new Integer(11), new Integer(22)); //1 kings
    bookToChapters.put(new Integer(12), new Integer(25)); //2 kings
    bookToChapters.put(new Integer(13), new Integer(29)); //1 chronicles
    bookToChapters.put(new Integer(14), new Integer(36)); //2 chronicles
    bookToChapters.put(new Integer(15), new Integer(10)); //ezra
    bookToChapters.put(new Integer(16), new Integer(13)); //nehemia
    bookToChapters.put(new Integer(17), new Integer(10)); //esther
    bookToChapters.put(new Integer(18), new Integer(42)); //job
    bookToChapters.put(new Integer(19), new Integer(150)); //psalms
    bookToChapters.put(new Integer(20), new Integer(31)); //proverbs
    bookToChapters.put(new Integer(21), new Integer(12)); //ecclesiastes
    bookToChapters.put(new Integer(22), new Integer(8)); //song of songs
    bookToChapters.put(new Integer(23), new Integer(66)); //isaiah
    bookToChapters.put(new Integer(24), new Integer(52)); //jeremiah
    bookToChapters.put(new Integer(25), new Integer(5)); //lamentations
    bookToChapters.put(new Integer(26), new Integer(48)); //ezekiel
    bookToChapters.put(new Integer(27), new Integer(12)); //daniel
    bookToChapters.put(new Integer(28), new Integer(14)); //hosea
    bookToChapters.put(new Integer(29), new Integer(3)); //joel
    bookToChapters.put(new Integer(30), new Integer(9)); //amos
    bookToChapters.put(new Integer(31), new Integer(0)); //obadiah
    bookToChapters.put(new Integer(32), new Integer(4)); //jonah
    bookToChapters.put(new Integer(33), new Integer(7)); //micah
    bookToChapters.put(new Integer(34), new Integer(3)); //nahum
    bookToChapters.put(new Integer(35), new Integer(3)); //habakkuk
    bookToChapters.put(new Integer(36), new Integer(3)); //zephaniah
    bookToChapters.put(new Integer(37), new Integer(2)); //haggai
    bookToChapters.put(new Integer(38), new Integer(14)); //zechariah
    bookToChapters.put(new Integer(39), new Integer(4)); //malachi
    bookToChapters.put(new Integer(40), new Integer(28)); //matthew
    bookToChapters.put(new Integer(41), new Integer(16)); //mark
    bookToChapters.put(new Integer(42), new Integer(24)); //luke
    bookToChapters.put(new Integer(43), new Integer(21)); //john
    bookToChapters.put(new Integer(44), new Integer(28)); //acts
    bookToChapters.put(new Integer(45), new Integer(16)); //romans
    bookToChapters.put(new Integer(46), new Integer(16)); //1 corinthians
    bookToChapters.put(new Integer(47), new Integer(13)); //2 corinthians
    bookToChapters.put(new Integer(48), new Integer(6)); //galatians
    bookToChapters.put(new Integer(49), new Integer(6)); //ephesians
    bookToChapters.put(new Integer(50), new Integer(4)); //phillipians
    bookToChapters.put(new Integer(51), new Integer(4)); //colossians
    bookToChapters.put(new Integer(52), new Integer(5)); //1 thessalonians
    bookToChapters.put(new Integer(53), new Integer(3)); //2 thessalonians
    bookToChapters.put(new Integer(54), new Integer(6)); //1 timothy
    bookToChapters.put(new Integer(55), new Integer(4)); //2 timothy
    bookToChapters.put(new Integer(56), new Integer(3)); //titus
    bookToChapters.put(new Integer(57), new Integer(0)); //philemon
    bookToChapters.put(new Integer(58), new Integer(13)); //hebrews
    bookToChapters.put(new Integer(59), new Integer(5)); //james
    bookToChapters.put(new Integer(60), new Integer(5)); //1 peter
    bookToChapters.put(new Integer(61), new Integer(3)); //2 peter
    bookToChapters.put(new Integer(62), new Integer(5)); //1 john
    bookToChapters.put(new Integer(63), new Integer(0)); //2 john
    bookToChapters.put(new Integer(64), new Integer(0)); //3 john
    bookToChapters.put(new Integer(65), new Integer(0)); //jude
    bookToChapters.put(new Integer(66), new Integer(22)); //revelation
}
```

```java
/**
 * Fills the Vector with booknames, dependant on the language
 */
private void fillBooks() {
    books = new Vector();
    //WATCH OUT, ORDER OF PUTTING THESE IN THE VECTOR MATTERS FOR
    //TRANSLATION IN SAVING AND OPENING THE VERSE FILES!!!!
    books.add(BibleStudy.getMessages().getString("genesis"));
    books.add(BibleStudy.getMessages().getString("exodus"));
    books.add(BibleStudy.getMessages().getString("leviticus"));
    books.add(BibleStudy.getMessages().getString("numbers"));
    books.add(BibleStudy.getMessages().getString("deuteronomy"));
    books.add(BibleStudy.getMessages().getString("joshua"));
    books.add(BibleStudy.getMessages().getString("judges"));
    books.add(BibleStudy.getMessages().getString("ruth"));
    books.add(BibleStudy.getMessages().getString("1samuel"));
    books.add(BibleStudy.getMessages().getString("2samuel"));
    books.add(BibleStudy.getMessages().getString("1kings"));
    books.add(BibleStudy.getMessages().getString("2kings"));
    books.add(BibleStudy.getMessages().getString("1chronicles"));
    books.add(BibleStudy.getMessages().getString("2chronicles"));
    books.add(BibleStudy.getMessages().getString("ezra"));
    books.add(BibleStudy.getMessages().getString("nehemiah"));
    books.add(BibleStudy.getMessages().getString("esther"));
    books.add(BibleStudy.getMessages().getString("job"));
    books.add(BibleStudy.getMessages().getString("psalms"));
    books.add(BibleStudy.getMessages().getString("proverbs"));
    books.add(BibleStudy.getMessages().getString("ecclesiastes"));
    books.add(BibleStudy.getMessages().getString("songofsongs"));
    books.add(BibleStudy.getMessages().getString("isaiah"));
    books.add(BibleStudy.getMessages().getString("jeremiah"));
    books.add(BibleStudy.getMessages().getString("lamentations"));
    books.add(BibleStudy.getMessages().getString("ezekiel"));
    books.add(BibleStudy.getMessages().getString("daniel"));
    books.add(BibleStudy.getMessages().getString("hosea"));
    books.add(BibleStudy.getMessages().getString("joel"));
    books.add(BibleStudy.getMessages().getString("amos"));
    books.add(BibleStudy.getMessages().getString("obadiah"));
    books.add(BibleStudy.getMessages().getString("jonah"));
    books.add(BibleStudy.getMessages().getString("micah"));
    books.add(BibleStudy.getMessages().getString("nahum"));
    books.add(BibleStudy.getMessages().getString("habakkuk"));
    books.add(BibleStudy.getMessages().getString("zephaniah"));
    books.add(BibleStudy.getMessages().getString("haggai"));
    books.add(BibleStudy.getMessages().getString("zechariah"));
    books.add(BibleStudy.getMessages().getString("malachi"));
    books.add(BibleStudy.getMessages().getString("matthew"));
    books.add(BibleStudy.getMessages().getString("mark"));
    books.add(BibleStudy.getMessages().getString("luke"));
    books.add(BibleStudy.getMessages().getString("john"));
    books.add(BibleStudy.getMessages().getString("acts"));
    books.add(BibleStudy.getMessages().getString("romans"));
    books.add(BibleStudy.getMessages().getString("1corinthians"));
    books.add(BibleStudy.getMessages().getString("2corinthians"));
    books.add(BibleStudy.getMessages().getString("galatians"));
    books.add(BibleStudy.getMessages().getString("ephesians"));
    books.add(BibleStudy.getMessages().getString("philippians"));
    books.add(BibleStudy.getMessages().getString("colossians"));
    books.add(BibleStudy.getMessages().getString("1thessalonians"));
    books.add(BibleStudy.getMessages().getString("2thessalonians"));
    books.add(BibleStudy.getMessages().getString("1timothy"));
    books.add(BibleStudy.getMessages().getString("2timothy"));
    books.add(BibleStudy.getMessages().getString("titus"));
    books.add(BibleStudy.getMessages().getString("philemon"));
    books.add(BibleStudy.getMessages().getString("hebrews"));
    books.add(BibleStudy.getMessages().getString("james"));
    books.add(BibleStudy.getMessages().getString("1peter"));
    books.add(BibleStudy.getMessages().getString("2peter"));
    books.add(BibleStudy.getMessages().getString("1john"));
    books.add(BibleStudy.getMessages().getString("2john"));
    books.add(BibleStudy.getMessages().getString("3john"));
    books.add(BibleStudy.getMessages().getString("jude"));
    books.add(BibleStudy.getMessages().getString("revelation"));
}
```

```java
/**
 * Fills the HashMap with booknames, mapped to the BibleGateway.org URL
 * dependant on the language
 */
private void fillBookToURLs() {
    bookNameToURLs = new HashMap();
    bookNameToURLs.put(BibleStudy.getMessages().getString("genesis"), "genesis");
    bookNameToURLs.put(BibleStudy.getMessages().getString("exodus"), "exodus");
    bookNameToURLs.put(BibleStudy.getMessages().getString("leviticus"), "leviticus");
    bookNameToURLs.put(BibleStudy.getMessages().getString("numbers"), "numbers");
    bookNameToURLs.put(BibleStudy.getMessages().getString("deuteronomy"), "deuteronomy");
    bookNameToURLs.put(BibleStudy.getMessages().getString("joshua"), "joshua");
    bookNameToURLs.put(BibleStudy.getMessages().getString("judges"), "judges");
    bookNameToURLs.put(BibleStudy.getMessages().getString("ruth"), "ruth");
    bookNameToURLs.put(BibleStudy.getMessages().getString("1samuel"), "1sam");
    bookNameToURLs.put(BibleStudy.getMessages().getString("2samuel"), "2sam");
    bookNameToURLs.put(BibleStudy.getMessages().getString("1kings"), "1kgs");
    bookNameToURLs.put(BibleStudy.getMessages().getString("2kings"), "2kgs");
    bookNameToURLs.put(BibleStudy.getMessages().getString("1chronicles"), "1chron");
    bookNameToURLs.put(BibleStudy.getMessages().getString("2chronicles"), "2chron");
    bookNameToURLs.put(BibleStudy.getMessages().getString("ezra"), "ezra");
    bookNameToURLs.put(BibleStudy.getMessages().getString("nehemiah"), "neh");
    bookNameToURLs.put(BibleStudy.getMessages().getString("esther"), "esth");
    bookNameToURLs.put(BibleStudy.getMessages().getString("job"), "job");
    bookNameToURLs.put(BibleStudy.getMessages().getString("psalms"), "psalms");
    bookNameToURLs.put(BibleStudy.getMessages().getString("proverbs"), "proverbs");
    bookNameToURLs.put(BibleStudy.getMessages().getString("ecclesiastes"), "ecc");
    bookNameToURLs.put(BibleStudy.getMessages().getString("songofsongs"), "song");
    bookNameToURLs.put(BibleStudy.getMessages().getString("isaiah"), "isa");
    bookNameToURLs.put(BibleStudy.getMessages().getString("jeremiah"), "jer");
    bookNameToURLs.put(BibleStudy.getMessages().getString("lamentations"), "lam");
    bookNameToURLs.put(BibleStudy.getMessages().getString("ezekiel"), "ezek");
    bookNameToURLs.put(BibleStudy.getMessages().getString("daniel"), "dan");
    bookNameToURLs.put(BibleStudy.getMessages().getString("hosea"), "hosea");
    bookNameToURLs.put(BibleStudy.getMessages().getString("joel"), "joel");
    bookNameToURLs.put(BibleStudy.getMessages().getString("amos"), "amos");
    bookNameToURLs.put(BibleStudy.getMessages().getString("obadiah"), "obad");
    bookNameToURLs.put(BibleStudy.getMessages().getString("jonah"), "jona");
    bookNameToURLs.put(BibleStudy.getMessages().getString("micah"), "micah");
    bookNameToURLs.put(BibleStudy.getMessages().getString("nahum"), "nahum");
    bookNameToURLs.put(BibleStudy.getMessages().getString("habakkuk"), "habakuk");
    bookNameToURLs.put(BibleStudy.getMessages().getString("zephaniah"), "zeph");
    bookNameToURLs.put(BibleStudy.getMessages().getString("haggai"), "hag");
    bookNameToURLs.put(BibleStudy.getMessages().getString("zechariah"), "zech");
    bookNameToURLs.put(BibleStudy.getMessages().getString("malachi"), "malachi");
    bookNameToURLs.put(BibleStudy.getMessages().getString("matthew"), "matthew");
    bookNameToURLs.put(BibleStudy.getMessages().getString("mark"), "mark");
    bookNameToURLs.put(BibleStudy.getMessages().getString("luke"), "luke");
    bookNameToURLs.put(BibleStudy.getMessages().getString("john"), "john");
    bookNameToURLs.put(BibleStudy.getMessages().getString("acts"), "acts");
    bookNameToURLs.put(BibleStudy.getMessages().getString("romans"), "romans");
    bookNameToURLs.put(BibleStudy.getMessages().getString("1corinthians"), "1cor");
    bookNameToURLs.put(BibleStudy.getMessages().getString("2corinthians"), "2cor");
    bookNameToURLs.put(BibleStudy.getMessages().getString("galatians"), "galatians");
    bookNameToURLs.put(BibleStudy.getMessages().getString("ephesians"), "ephesians");
    bookNameToURLs.put(BibleStudy.getMessages().getString("philippians"), "phil");
    bookNameToURLs.put(BibleStudy.getMessages().getString("colossians"), "colossians");
    bookNameToURLs.put(BibleStudy.getMessages().getString("1thessalonians"), "1thes");
    bookNameToURLs.put(BibleStudy.getMessages().getString("2thessalonians"), "2thes");
    bookNameToURLs.put(BibleStudy.getMessages().getString("1timothy"), "1tim");
    bookNameToURLs.put(BibleStudy.getMessages().getString("2timothy"), "2tim");
    bookNameToURLs.put(BibleStudy.getMessages().getString("titus"), "titus");
    bookNameToURLs.put(BibleStudy.getMessages().getString("philemon"), "philem");
    bookNameToURLs.put(BibleStudy.getMessages().getString("hebrews"), "heb");
    bookNameToURLs.put(BibleStudy.getMessages().getString("james"), "james");
    bookNameToURLs.put(BibleStudy.getMessages().getString("1peter"), "1pet");
    bookNameToURLs.put(BibleStudy.getMessages().getString("2peter"), "2pet");
    bookNameToURLs.put(BibleStudy.getMessages().getString("1john"), "1john");
    bookNameToURLs.put(BibleStudy.getMessages().getString("2john"), "2john");
    bookNameToURLs.put(BibleStudy.getMessages().getString("3john"), "3john");
    bookNameToURLs.put(BibleStudy.getMessages().getString("jude"), "jude");
    bookNameToURLs.put(BibleStudy.getMessages().getString("revelation"), "revelation");
}
```

Inappropriate Intimacy

One class accesses internal information of another class

Casting interface to implementation...

```
for (BudgetCategory bc : document.getBudgetCategories()) {
    boolean changed = false;
    Map<String, Long> newAmountMap = new HashMap<String, Long>();

    for (String key : ((BudgetCategoryImpl) bc).getBudgetPeriods().keySet()) {
        String[] splitKey = key.split(":");
        if (splitKey.length == 2){
            changed = true;
            Date dateKey = new Date(Long.parseLong(splitKey[1]));
            Logger.getLogger(ModelFactory.class.getName()).finest("dateKey: " + dateKey);
            long amount = ((BudgetCategoryImpl) bc).getBudgetPeriods().get(key);
            String newDateKey =
                bc.getBudgetPeriodType().getName()
                + ":" + DateUtil.getYear(dateKey)
                + ":" + DateUtil.getMonth(dateKey)
                + ":" + DateUtil.getDay(dateKey);
            newAmountMap.put(newDateKey, amount);
        }
        else if (splitKey.length == 4){
            newAmountMap.put(key, ((BudgetCategoryImpl) bc).getBudgetPeriods().get(key));
        }
    }

    if (changed){
        ((BudgetCategoryImpl) bc).setBudgetPeriods(newAmountMap);
    }
}
```

Speculative Generality

Code that is more complicated than to fulfill the current requirements

Asymmetrical Code

Asymmetrical in naming or location of functionality

Mocking

Mocking Basics

What is mocking?

Use mock objects to substitute the real collaborators

When to mock?

it depends on classes that have side effects (e.g. network connection, database connection).

we have no control over the source code of 3rd party libraries with which it is collaborating.

Speeds up test execution

Some real collaborators are expensive to create/execute.
E.g. resource intensive algorithm, complex creation.

Mock objects help return canned results which can make the tests run much faster.

You can mock by hand

```
@Test
public void sendEmailOnMakingPayment() throws Exception {
    MailServiceStub mailService = new MailServiceStub();
    ShoppingCart shoppingCart = buildShoppingCart();

    shoppingCart.setMailService(mailService);
    shoppingCart.makePayment(paymentMethod);

    assertEquals(1, mailService.numberOfMessagesSent());
}

public interface MailService {
    public void send(Message msg);
}
public class MailServiceStub implements MailService {
    List<Message> list = new ArrayList<Message>();
    public void send(Message msg) {
list.add(msg);
    }
    public int numberOfMessagesSent() {
        return list.size();
    }
}
```

execution

verify

write stubs

Or use a mocking framework

```
import static org.mockito.Mockito.*;

@Test
public void sendEmailOnMakingPayment() throws Exception {
    final MailService mailService = mock(MailService.class);

    ShoppingCart shoppingCart = buildShoppingCart();
    shoppingCart.setMailService(mailService);


    shoppingCart.makePayment(paymentMethod);


    verify(mailService).send(any(Message.class));
}
```

mock the object

execution

verify

Also known as interaction-based testing

Using mockito

Argument Matchers

```
verify(mailService).send(any(Message.class));


anyString(), anyInt(), anyList() ...

eq(boolean, String, int, Object ...)


verify(mailService).send(eq(expectedMessage));

verify(mailService).set(eq("localhost"), portNumber);
```

all arguments have to be provided by matchers

Stubbing

stub value

```
when(userRepository.findEmail(anyString())).thenReturn(expectedUser);
```

stub exception

```
when(userRepository.findEmail(anyString())).thenThrow(new UserNotFound());
```

stub exception on void method

```
doThrow(new MailException()).when(mailService).send(any(Message.class));
```

More mockito

http://docs.mockito.googlecode.com/hg/latest/org/mockito/Mockito.html

JavaScript Unit Testing
with Jasmine

Odd-e

We have to unit test JavaScript??
Yes!

Why?

Refactoring

"Rich-client"

Regression Testing

Firefox, Safari, Chrome...

Frameworks

QUnit
jQuery

YAHOO!
YUI Test

PhantomJS

js-test-driver
Remote javascript console

mocha

Jasmine

mock & async included

pure-JS, no DOM

BDD style

Code Coverage

CI server friendly mvn plugin

Setting up Jasmine

Built by:

maven

jasmine-maven-plugin

```
<plugin>
    <groupId>com.github.searls</groupId>
    <artifactId>jasmine-maven-plugin</artifactId>
    <version>1.3.1.2</version>
    <executions>
        <execution>
            <goals>
                <goal>test</goal>
            </goals>
        </execution>
    </executions>
    <configuration>
        <!-- configuration properties will go here -->
    </configuration>
</plugin>
```

https://github.com/searls/jasmine-maven-plugin

Default Paths

src/**test**/javascript

src/**main**/javascript

changeable in Maven configuration ‘jsSrcDir’

http://searls.github.com/jasmine-maven-plugin

How to run - developing

1

```
mvn jasmine:bdd
```

2

How to run - CI

Bind to test phase in maven plugin

```
mvn jasmine:test
```

uses HtmlUnit for GUI-less execution

How to run - Code Coverage

file:///Users/hugues/proj/testable-javascript/my-jasmine-project/target/coverage/total-report.html#6fc0c978938

Total coverage report

File	Statements	Executed	Coverage
Total	**15**	**12**	**80 %**
src/**authentication-form.js**	12	9	75 %

Missed lines: 6-7, 29

```
1  1  function AuthenticationForm() {
2  2          this.username = $("#username").val();
3     }
4
5  1  AuthenticationForm.prototype.onSubmit = function(event) {
6  !          event.preventDefault();
7  !          this.checkAuthentication(this.username, this.password);
8     };
9
10 1  AuthenticationForm.prototype.checkAuthentication = function(username, password, callback) {
11 1          $.ajax({
12                type: "POST",
13                url: "/authenticate_user",
14                data: {
15                  username: username,
16                  password: password
17                },
18                done: function(resp) {
19 1               var user = null;
20 1               if (resp.success) {
21 1                 user = {
22                     username: resp.username,
23                     userid: resp.userid
24                   };
25                 }
26 1               callback && callback(null, user);
27                },
28                fail: function(jqXHR, textStatus, errorThrown) {
29 !               callback && callback(errorThrown);
30                }
31            });
32    };
```

src/**jquery-v1.9.1.min.js**	3	3	100 %

Generated using saga-core version 1.5.0

uses saga-maven-plugin

Using Jasmine

```
describe("CookBook Photo", function() {
    it("should pass a png extension", function() {
        expect(isPhoto('abc.png')).toBeTruthy();
    });

    it("should fail if there is no suffix", function() {
        expect(isPhoto('abc')).toBeFalsy();
    });

    it("should fail if it is not a known picture", function() {
        expect(isPhoto('abc.xls')).toBeFalsy();
    });
});
```

#Suite

#Spec

Jasmine 1.0.2 revision 1298837858 Show ☑ passed ☐ skipped

3 specs, 0 failures in 0.034s Finished at Tue Dec 06 2011 23:00:56 GMT+0800 (SGT) run all

CookBook Photo run

- should pass a png extension run
- should fail if there is no suffix run
- should fail if it is not a known picture run

Example: Login Form

```
<form id="authentication_form" class="form-signin">
  <h2 class="form-signin-heading">Please sign in</h2>
  <input type="text" id="username"
    class="input-block-level" placeholder="Email address">
  <input type="password" id="password"
    class="input-block-level" placeholder="Password">
  <label class="checkbox">
    <input type="checkbox" value="remember-me"> Remember me
  </label>
  <button class="btn btn-large btn-primary" type="submit">Sign in</button>
</form>
```

Untestable

```
$('#authentication_form').on('submit', function(e) {
  event.preventDefault();
  var username = $('#username').val();
  var password = $('#password').val();
  var messageContainer = $('#message');
  $.ajax({
    url: '/doLogin',
    type: 'POST',
    done: function() {
      console.log('Great success');
      window.location('/dashboard');
    },
    fail: function(e) {
      console.log('Issue submitting to the server',
      messageContainer.append('<div>Login failed: ' +
                e.message + '</div>');
    }
  });
});
```

#anonymous function

#UI coupling

#Backend Service

#Routing

#UI in-line

TDD Step 1

src/test/javascript/test-authentication-form.js

```
describe("When loading the login form", function() {

    var authenticationForm;

    it("Must find an empty username", function() {
        expect(authenticationForm.username).toEqual('');
    });

});
```

src/main/javascript/authentication-form.js

```
function AuthenticationForm() {
    this.username = $("#username").val();
}
```

Good Practices

- All javascript externalized
- Use objects and instantiate them
- No singletons
- No global variables

TDD: Step 2

```
describe("When loading the login form", function() {

    var rootId = 'testContainer';
    var authenticationFormMarkup = '<form>'+
          '<input type="text" id="username" name="username"></input>' +
          '<input type="password" id="password" name="password"></input>' +
          '</form>';

    beforeEach(function() {
        var container = document.createElement('div');
        container.setAttribute('id',rootId);
        document.body.appendChild(container);
        container.innerHTML = authenticationFormMarkup;
        authenticationForm = new AuthenticationForm();
    });

    afterEach(function() {
        var container = document.getElementById(rootId);
        container.parentNode.removeChild(container);
    });

    it("Must find an empty username", function() {
        expect(authenticationForm.username).toEqual('');
    });

});
```

#Fixture

#Instantiate Object

Ajax Service Code

```
AuthenticationForm.prototype.checkAuthentication = function(username, password, callback) {
    $.ajax({
        type: "POST",
        url: "/authenticate_user",
        data: {
          username: username,
          password: password
        },
        done: function(resp) {
          var user = null;
          if (resp.success) {
            user = {
              username: resp.username,
              userid: resp.userid
            };
          }
          callback && callback(null, user);
        },
        fail: function(jqXHR, textStatus, errorThrown) {
          callback && callback(errorThrown);
        }
    });
};
```

Ajax Service Code Test

```
describe("When checking the username and password", function() {
    var formSubmitted;
    var originalAjaxFct;
    beforeEach(function() {
        formSubmitted = new AuthenticationForm();
        originalAjaxFct = $.ajax;
    });
    afterEach(function() {
        $.ajax = originalAjaxFct;
    });
    it("Must call jquery ajax", function() {
        var usernameTested = 'joe';
        var passwordTested = 'abc';
        $.ajax = function(options) {
            expect(typeof options).toEqual('object');
            expect(options.data.username).toEqual(usernameTested);
            expect(typeof options.done).toEqual('function');
            options.done({ success: true, username: usernameTested, userid: 123 });
        };
        formSubmitted.checkAuthentication(usernameTested, passwordTested, function(err, res) {
            expect(err).toEqual(null);
            expect(res.username).toEqual(usernameTested);
            expect(res.userid).toEqual(123);
        });
    });
});
```

Conclusion / Q&A

- Linting with JSHint: a syntax checker

- TDD and Javascript: just do it

Thinking about Design

Emergent Design

Traditional Design

Ref: peterme.com

Reality

Not designed!

Forced design

Do not walk there!

Reality

Larger problems!

Beginning of a new path

Emergent design

Ref: Discovering Urbanism

Reality

Traditional vs Emergent

Traditional

Emergent

Traditional vs Emergent

Traditional design is about decision

Decide the whole blueprint

Emergent design is about direction

Make sure we are going in the right direction

Traditional vs Emergent

Mindset:
We can think of most things beforehand. That will be most efficient

Mindset:
Premature speculation is probably wrong. Gradual discovery and learning leads to better solutions

Emergent Design

Steering the design

Steering the direction of the design based on:

- Order of test-driving
- Principles
- Patterns

Principles

From: losTechnies.com

SOLID

Software Development is not a Jenga game

SOLID

Principles for Object-Oriented Design collected by Uncle Bob Martin:

- S - Single Responsibility Principle
- O - Open-Closed Principle
- L - Liskov Substitution Principle
- I- Interface Segregation Principle
- D - Dependency Inversion Principle

SINGLE RESPONSIBILITY PRINCIPLE

Just Because You Can, Doesn't Mean You Should

SRP - Single-Responsibility Principle

```
public final class CurrencyConverter {
    private static final String ECB_RATES_URL = "http://www.ecb.int/stats/eurofxref/eurofxref-daily.xml";

    transient private File cacheFile = null;
    private String cacheFileName = null;
    private HashMap<String, Long> fxRates = new HashMap<String, Long>(40);
    private Proxy proxy = Proxy.NO_PROXY;

    public void setProxy(Proxy proxy) { ... }

    public double convert(double amount, String fromCurrency, String toCurrency) { ... }

    public String getCacheFileName() { ... }

...
```

"A class should have only one reason to change"

This class breaks the SRP:

- Converts Currencies
- Updates exchange rates via the web
- Caches rates locally

OPEN CLOSED PRINCIPLE

Open Chest Surgery Is Not Needed When Putting On A Coat

OCP - Open-Closed Principle

```
public final class CurrencyConverter {
    private static final String ECB_RATES_URL = "http://www.ecb.int/stats/eurofxref/eurofxref-daily.xml";

    transient private File cacheFile = null;
    private String cacheFileName = null;
    private HashMap<String, Long> fxRates = new HashMap<String, Long>(40);
    private Proxy proxy = Proxy.NO_PROXY;

    public void setProxy(Proxy proxy) { ... }

    public double convert(double amount, String fromCurrency, String toCurrency) { ... }

    public String getCacheFileName() { ... }

...
```

Software entities should be open for extension, but closed for modification

This class breaks the OCP:

- It doesn't allow for extension, such as retrieving the currencies from a non-web source.
- Even getting the currencies from a different URL will require modification.

LISKOV SUBSTITUTION PRINCIPLE

If It Looks Like A Duck, Quacks Like A Duck, But Needs Batteries - You Probably Have The Wrong Abstraction

LSP - Liskov Substitution Principle

```
public class SortedArrayList<T extends Comparable<? super T>> extends AbstractList<T> {
    private ArrayList<T> backingList;

    public void add(int location, T object) {
        //We don't care about what position you try for, foo!
        add(arg1);
    }

    public boolean add(T object) {
        int index = Collections.binarySearch(backingList, arg0);
        if (index < 0)
            index = (index + 1) * -1;
        backingList.add(index, arg0);
        return true;
    }
```

Subtypes must be substitutable for their base types

This class breaks the LSP:

- The add to a position in the AbstractList breaks the base type interface.
- After adding it to position x, the get from position x will return a different object.

INTERFACE SEGREGATION PRINCIPLE

You Want Me To Plug This In, Where?

ISP - Interface Segregation Principle

```
public interface Document extends ModelObject, StandardDocument {

    public static final int RESET_PASSWORD = 1;  //Should we change the current password?
    public static final int CHANGE_PASSWORD = 2; //Should we prompt for a password?

    public void addAccount(Account account) throws ModelException;
    public void addAccountType(AccountType type) throws ModelException;
    public void addBudgetCategory(BudgetCategory budgetCategory) throws ModelException;
    public void addScheduledTransaction(ScheduledTransaction scheduledTransaction) throws ModelException;
    public void addTransaction(Transaction transaction) throws ModelException;
    public String doSanityChecks();
    public Account getAccount(String name);
    public List<Account> getAccounts();
    public AccountType getAccountType(String name);
    public List<AccountType> getAccountTypes();
    public List<BudgetCategory> getBudgetCategories();
    public BudgetCategory getBudgetCategory(String fullName);
    public long getNetWorth(Date date);
    public ModelObject getObjectByUid(String uid);
    public List<ScheduledTransaction> getScheduledTransactions();
    public List<Source> getSources();
    public List<Transaction> getTransactions();
    public List<Transaction> getTransactions(Date startDate, Date endDate);
    public List<Transaction> getTransactions(Source source);
    public List<Transaction> getTransactions(Source source, Date startDate, Date endDate);
    public void refreshUidMap() throws ModelException;
    public void removeAccount(Account account) throws ModelException;
    public void removeAccountType(AccountType type) throws ModelException;
    public void removeBudgetCategory(BudgetCategory budgetCategory) throws ModelException;
    public void removeScheduledTransaction(ScheduledTransaction scheduledTransaction) throws ModelException;
    public void removeTransaction(Transaction transaction) throws ModelException;
    public void save() throws DocumentSaveException;
    public void saveAs(File file) throws DocumentSaveException;
    public void saveToStream(OutputStream os) throws DocumentSaveException;
    public void setFlag(int flag, boolean set);
    public void updateAllBalances();
    public void updateScheduledTransactions();
    public Document clone() throws CloneNotSupportedException;
}
```

Clients should not be forced to depend on methods that they do not use

DEPENDENCY INVERSION PRINCIPLE

Would You Solder A Lamp Directly To The Electrical Wiring In A Wall?

DIP - Dependency Inversion Principle

```
public class users implements IStructuredContentProvider {
    private Vector<user> usersList=new Vector<user>();

    private user activeUser=null;


    public void getUsersFromDB() {
        usersList.clear();

        List retrievals = DB.getSession().createQuery("from user WHERE outdated IS NULL").list();
        for (Iterator iter = retrievals.iterator(); iter.hasNext();) {
            user currentlyRetrieved =  (user)iter.next();
            usersList.add(currentlyRetrieved);
        }
```

Abstractions should not depend on details
Details should depend on abstractions

This class breaks the DIP:

- getUsersFromDB details ought to depend on an abstraction rather than direct SQL queries

Other principles

DRY. Don't Repeat Yourself!

Law of Demeter

Don't Ask, Tell.

Design by Contract

DRY - Don’t Repeat Yourself

Every piece of knowledge must have a single, unambiguous, authoritative representation within a system

DRY principle covers more than obvious duplication, focus on duplication of knowledge

Law of Demeter

Any method of an object should only call methods belonging to:

- Itself
- Its parameters
- Any objects it creates
- Its direct component objects

```
public String replaceValues(TemplatePlaceholderValues replacementValues) {
    return template.replace(replacementValues.firstPlaceholder().name(),
            replacementValues.firstPlaceholder().value());
}
```

This class breaks the law of Demeter:

- The call to name() doesn't belong to any of these four groups

Don’t ask, tell!

You should endeavor to tell objects what you want them to do instead of asking for their data

Design by Contract

Relationship between a method and it's caller can be viewed as a contract between them containing:

- Pre-condition
- Post-condition
- Class invariant

Design Contract

Client

```
public class User {

    private Printer printer = new Printer();
    private String content;

    public void displayEverything() {
        printer.printLines(content, 10);
    }
}
```

Supplier

```
public class Printer {

    public void printLines(String content, int i) {

    }
}
```

Design Contract: The pre-condition

Client

```java
public class User {

    private Printer printer = new Printer();
    private String content;

    public void displayEverything1() {
        printer.printLines1(content, 10);
    }
}
```

Supplier

```java
public class Printer {

    public int printLines1(String content, int linesToPrint) {
        if (content == null)
            throw new IllegalArgumentException();
        if (linesToPrint < 0)
            throw new IllegalArgumentException();

        int amountOfLinesPrinted = 0;

        // Do the work

        return amountOfLinesPrinted;
    }
}
```

Contract:

Precondition: <anything>

Whatever the client passes in, Printer should work correctly.

Design Contract: The pre-condition

Client

```
private String content2 = new String();

public void displayEverything2() {
    printer.printLines2(content2, 10);
}
```

Supplier

```
public int printLines2(String content, int linesToPrint) {
    if (linesToPrint < 0)
        throw new IllegalArgumentException();

    int amountOfLinesPrinted = 0;

    // Do the work

    return amountOfLinesPrinted;
}
```

Contract:

Precondition:

- content is a valid String

Now, client is responsible for ensuring that content is a valid String. If it is not a valid String then the bug is in Client and not in Supplier!

Design Contract: The pre-condition

Client

```
private String content3 = new String();

public void displayEverything3() {
    printer.printLines3(content3, 10);
}
```

Supplier

```
public int printLines3(String content, int linesToPrint) {

    int amountOfLinesPrinted = 0;

    // Do the work

    return amountOfLinesPrinted;
}
```

Contract:

Precondition:
- content is a valid String
- linesToPrint must be >= 0

More expectations move to the client by stricter pre-conditions

Design Contract: The post-condition

Client

```
public class User {

    private Printer printer = new Printer();
    private String content;

    public void displayEverything4() {
        printer.printLines4(content, 10);
    }
}
```

Supplier

```
    public int printLines4(String content, int linesToPrint) {
        return (int) ((Math.random() - 0.5)
            * (double) Integer.MAX_VALUE) * 2;
    }
```

Contract:

Postcondition:

- printLines returns a valid int

Client cannot assume anything!

Design Contract: The post-condition

Client

```
public class User {

    private Printer printer = new Printer();
    private String content;

    public void displayEverything5() {
        printer.printLines5(content, 10);
    }
}
```

Supplier

```
    public int printLines5(String content, int linesToPrint) {
        int amountOfLinesPrinted = 0;

        // Do the work

        return amountOfLinesPrinted;
    }
```

Contract:

Postcondition:
- printLines returns >= 0
- printLines won't throw an exception

Client can assume a sane return value

Design Contract: The post-condition

Client

```
public class User {

    private Printer printer = new Printer();
    private String content;

    public void displayEverything() {
        printer.printLines(content, 10);
    }
}
```

Supplier

```
    public int printLines5(String content, int linesToPrint) {
        int amountOfLinesPrinted = 0;

        // Do the work

        return amountOfLinesPrinted;
    }
```

Contract:

Postcondition:
- printLines returns >= 0
- printLines won't throw an exception
- the sizeOfBufferedText() == 0

The post-condition expresses what happened to the state of the object!

Design Contract: Class Invariant

Supplier

```
public class Printer {

    public int capacityOfPrintBuffer() {
        return capacity;
    }

    public int sizeOfBufferedText() {
        return sizeOfBufferedText;
    }
```

Contract:

Class Invariant:
-sizeOfBufferedText <= capacityOfPrintBuffer

Client can at any time assume the supplier's class invariant

Design Contract: Example contracts

Contract:

Precondition:
-<anything>
Postcondition:
- printLines returns >= 0
- printLines can throw IllegalArgumentException
Class Invariant:
-sizeOfBufferedText <= capacityOfPrintBuffer

With a weak pre-condition, the supplier can't guarantee much... (garbage-in garbage-out)

Often leads to ugly code

With a strong pre-condition, the supplier can guarantee more which leads to less error handling in client

Contract:

Precondition:
- content is a valid String
- linesToPrint >= 0
Postcondition:
- printLines returns >= 0
- the sizeOfBufferedText() == 0
Class Invariant:
-sizeOfBufferedText <= capacityOfPrintBuffer

Design by Contract

Strong/weak pre/post conditions are not good/bad, they just are.

Always make expectations and assumptions clear by thinking in contracts between client/supplier

Liskov substitution principle

- Pre-conditions for derived types must be the same or weaker than base type
- Post-conditions for derived types must be the same or stronger than base type

Design by Contract Summary

Pre-condition: must be true to execute correctly:

- The supplier assumes it to be true
- The user ensures it is true!

Post-condition: a state prevails upon completion i.e. a description of the results of the actions performed.

- The supplier must make it so!
- The user assumes it will be true

Modeling in Agile

First law of diagramming

The primary value in diagrams is in
the discussion while diagramming—
we model to have a conversation

Model on whiteboards

Ref: Ravenbook

Model together

Ref: Technical University Dortmund

Principles

- Software is your primary goal
- Assume simplicity
- Model with a purpose
- Multiple models
- Content is more important than representation
- Everyone can learn from everyone else

Ref: Agile Modeling

Avoid electronic tools

Avoid electronic modeling tools. They limit the space and its hard to model together.

Even for documentation, photos often work better

Object Creation

Hard dependencies

```
void clear(double initialBalance) {
    this.currentBalanceDate = DateUtils.dateToInteger(new Date());
    this.balanceHistory = new BalanceHistory(initialBalance);
    this.finalBalance = initialBalance;
    this.currentBalance = initialBalance;
    this.checkedBalance = initialBalance;
}

void enableEvents(boolean enabled) {
    super.setEventsEnabled(enabled);
    if (enabled) fireEvent(new EverythingChangedEvent(this));
}
```

Created Objects are hard dependencies. They need to be separated

Factories

Move creation to separate classes

Dependency Injection

Pass objects in rather than create them yourself

- Setter injection
- Constructor injection

Patterns

Design Patterns
Elements of Reusable
Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Cover art © 1994 M.C. Escher / Cordon Art - Baarn - Holland. All rights reserved.
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Design Pattern

"descriptions of communicating objects and classes that are customized to solve a general design problem in a particular context"

-- Gang of four

Common patterns

- Composite
- Bridge
- Decorator
- Observer
- Null Object
- Builder
- Chain of responsibility
- Template method

Avoid Singleton

(well, no need to avoid this Singleton)

Good unit tests

Test Structure

Non-Mock test structure

Also called: Arrange-Act-Assert

Consistent code, Clear Structure

- Arrange-Act-Assert / Given-When-Then
- One assertion per test / Test one behavior per class

```
@Test
public void makesReservationThroughReservationService() {
    // ARRANGE
    final Reservation reservation = new Reservation();
    Reservations reservations = stubReservationsToReturn(reservation);
    HouseKeeping houseKeeping = stubHouseKeeping();
    HotelRoom room = new HotelRoom(reservations, houseKeeping);


    // ACT
    Reservation reservationReturned = room.bookFor(new Customer());

    // ASSERT
    assertSame(reservation, reservationReturned);
}
```

Mock test structure

Tests with mocks first specify the expectations and then execute.

Test <-> Production?

- one-to-one mapping of test class to production class is nearly always a bad idea...
- No one right approach, different strategies:
  - Test class per class
  - Test class per ‘feature’
  - Test class per fixture
- Sometimes, refactorings such as extract class lead to accidental test-production strategies. This is ok, just needs to evolve over time.

Properties of good tests

It is not a unit test when:

- It talks to the database
- It communicates across the network
- It touches the file system
- It can't run at the same time as any of your other unit tests
- You have to do special things to your environment (such as editing config files) to run it.

Michael Feathers

Odd-e

GOALS

- ✓ Tests should reduce risk, not introduce it.
- ✓ Tests should be easy to run again and again.
- ✓ Tests should be easy to maintain as time passes.

HOW TO DO IT?

- ✓ Tests should only fail because of one reason.
- ✓ Tests should only test one thing
- ✓ Minimize test dependencies (no dependencies on databases, files, ui etc.)

✓ Fast! (seconds)

✓ A-TRIP

- ★ Automated
- ★ Thorough
- ★ Repeatable
- ★ Independent
- ★ Professional

✓ Readable

NOT

*l*Eg*i*b*l*E *l*Et*t*e*R*s

BUT

Readable Words

✓ small & simple

Odd-e

✓ To the .

```
@Test
public void iteratorReturnsAllObjectsOfGivenTypeInObjectSpace() {
        IRubyObject o1 = runtime.newFixnum(10);
        IRubyObject o2 = runtime.newFixnum(20);
        IRubyObject o3 = runtime.newFixnum(30);
        IRubyObject o4 = runtime.newString("hello");
        objectSpace.add(o1);
        objectSpace.add(o2);
        objectSpace.add(o3);
        objectSpace.add(o4);

        List storedFixnums = new ArrayList(3);
        storedFixnums.add(o1);
        storedFixnums.add(o2);
        storedFixnums.add(o3);

        Iterator strings = objectSpace.iterator(runtime.getString());
        assertSame(o4, strings.next());
        assertNull(strings.next());

        Iterator numerics = objectSpace.iterator(runtime.getNumeric());
        for (int i = 0; i < 3; i++) {
                Object item = numerics.next();
                assertTrue(storedFixnums.contains(item));
        }
        assertNull(numerics.next());
}
```

```
@Test
public void iteratorReturnsAllObjectsOfGivenTypeInObjectSpace() {
        givenObjectSpaceContains(1, 2, 3);
        givenObjectSpaceContains("string");
        assertThat(iteratorFor(runtime.getString()), containsExactly("hello"));
        assertThat(iteratorFor(runtime.getNumeric()), containsExactly(1, 2, 3));
}
```

✓ Focused

```
public class HotelRoom {
    public Reservation bookFor(Customer customer) {
        Reservation reservation = reservations.reserveFor(customer, this);
        housekeeping.notifyAbout(reservation);
        return reservation;
    }
}

@Test
public void makesReservationThroughReservationService() {
    final Reservation reservation = new Reservation();
    Reservations reservations = stubReservationsToReturn(reservation);
    HouseKeeping houseKeeping = stubHouseKeeping();

    HotelRoom room = new HotelRoom(reservations, houseKeeping);

    assertSame(reservation, room.bookFor(new Customer()));
}
```

Whether housekeeping gets notified is another test's concern!

✓ Named well

```
@Test
public void booksRoomThroughReservations()
```

✓ Potty trained

GOOD TESTS

- ✓ Clean up after themselves.
- ✓ Don't leave the lid up.

Odd-e

✓ Expressive

Literals vs. Variables

```
@Test
public void alternative1() {
   int votes = 2;
   assertEquals(WEIGHT * votes * votes, new Topic(votes).popularity());
}

@Test
public void alternative2() {
   assertEquals(4.56, new Topic(votes).popularity());
}

@Test
public void alternative3() {
   int votes = 2;
   assertEquals(1.14 * votes * votes, new Topic(votes).popularity());
}
```

Which of these tests breaks most easily?

Which of these tests expresses its intent best?

Sometimes it's *good* if a test breaks easily.

Test Smells

Incidental detail

Expensive setup

Legacy Code

Legacy Overview

What is legacy code?

Code without unit test

The Legacy Code Dilemma

When we change code, we should have tests in place. To put tests in place, we often have to change code.

Legacy code algorithm

1. Identify change point
2. Find test points
3. Break dependencies
4. Write tests
5. Make changes and refactor

Change point and Test point

System boundaries – e.g. Network, database

Close proximity test points tend to provide a more localized checkpoint without too much noise around.

Distant test points are more likely to catch side effects

Seam

A seam is a place where you can alter behavior in your program without editing in that place

- Link Seam
- Preprocessor Seam
- Meta-programming Seam
  (e.g function pointer)
- Object (polymorphism) Seam

Enabling point is a place where you can make the decision to use one behavior or another in each seam.

I don’t want to break anything

- Automatic Refactoring tool
- Lean on Compiler
  - Let your compiler tell where you need to fix
- Signature Preservation
- Single Goal Editing
  - Do one thing at a time
- Pair Programming

Understanding the code

Characterization Tests

Tests that record the actual behavior of a piece of code

1. Use a piece of code in a test harness
2. Write an assertion that you know will fail
3. Let the failure tell you what the behavior is
4. Change the test so that it expects the behavior that the code produces

Sensing Variable

Add a variable to a class and use it to sense conditions in the method that you want to refactor

Example:

```
public static final CharSequence getDisplayLabel(Context context, int kind, int type, CharSequence label) {
    CharSequence display = "";
    if (type != People.ContactMethods.TYPE_CUSTOM) {
        CharSequence[] labels = context.getResources().getTextArray(emailAddressTypes);
        try {
            display = labels[type - 1];
        } catch (ArrayIndexOutOfBoundsException e) {
            display = labels[ContactMethods.TYPE_HOME - 1];
        }
    } else {
        if (!TextUtils.isEmpty(label)) {
            if (label.toString().equals(MOBILE_EMAIL_TYPE_NAME)) {
                display = context.getString(mobileEmailTypeName);
            } else {
                display = label;
            }
        }
    }
    return display;
}
```

Introduce sensing variable

```
public static boolean STANDARD_TYPE = false;

public static final CharSequence getDisplayLabel(Context context, int kind, int type, CharSequence label) {
    CharSequence display = "";
    if (type != People.ContactMethods.TYPE_CUSTOM) {
        CharSequence[] labels = context.getResources().getTextArray(emailAddressTypes);
        try {
            display = labels[type - 1];
        } catch (ArrayIndexOutOfBoundsException e) {
            display = labels[ContactMethods.TYPE_HOME - 1];
        }
    } else {
        if (!TextUtils.isEmpty(label)) {
            if (label.toString().equals(MOBILE_EMAIL_TYPE_NAME)) {
                display = context.getString(mobileEmailTypeName);
            } else {
                display = label;
            }
        }
        STANDARD_TYPE = true;
    }
    return display;
}
```

Add a test that covers the condition

```
public void testGetDisplayLabelForStandardType() throws Exception {

    Context context = new Context();

    Contacts.getDisplayLabel(context , standardKind, standardType, label);
    assertTrue(Contacts.STANDARD_TYPE);
}
```

Refactor to understand

- Refactoring helps understanding legacy code.

- Specific refactorings:
  - Rename attributes to convey meaning
  - Rename methods to convey intent
  - Rename classes to convey purpose
  - Remove duplicated code
  - Replace condition branches by methods
  - Refactor method bodies to consistent level of abstraction (remove Abstraction Distraction code smell)

Scratch Refactoring

- Refactor it whatever way you want to get a better understanding of it
- Don’t check it in again
- Throw that code away

Effect Propagation

- Identify a method that will change
- If the method has a return value, look at its callers
- See if the method modifies any values, if it does, look at the methods use those values, and the methods that use those methods
- Make sure to look for superclasses and subclasses for instance variables
- Look at parameters to the methods to see if they are used by the code that you want to change
- Look for global variables and static data that is modified

So we have layers of layers of dependencies?

Mikado Method

- It's not mean to construct the big whole graph at one step
- Avoid "Analysis Paralysis"
- "Undo" and revert whenever needed

Further reading: http://mikadomethod.wordpress.com/book/

Dependency Breaking

Common Strategies

- Add
  - Extra Constructor
  - Static Setter
  - Parameterize
- Replace
  - Test specific subclass
  - Static instance
- Extract
  - Method
  - Interface / Implementation
  - Method to Class
  - Globals to Class

You may need to use them in combination!

Parameterize functions

- Create a new method with the internally created object as an argument
- Copy the code from the original method into the old method, deleting the creation code
- Cut the code from the original method and replace it with a call to the new method, using a “new expression” in the argument list

```
void run() {
   m_result = new TestResult();
   runTest(m_result);
}
```

```
void run() {
   run(new TestResult());
}

void run(TestResult result) {
   m_result = result;
   runTest(m_result);
}
```

Extra Constructor

```
public class LogFileMerge {
  private URL logFileA, logFileB;

  public LogFileMerge() {
    this(new URL("http://server1/system.log"),
    new URL("http://server2/system.log"));
  }

  LogFileMerge(URL a, URL b) {
    this.logFileA = a;
    this.logFileB = b;
  }

  ...
}
```

Original delegates to extra constructor

Dependencies exposed to tests

Test-Specific SubClass

```
public class CreditCardProcessing {

    public boolean isValid(String cardnumber) {
        return validationCodeMatches(cardnumber)
            && cardIsActive(cardnumber);
    }

    protected boolean validationCodeMatches(String cardnumber) {
        // validation logic omitted for brevity...
        return false;
    }

    protected boolean cardIsActive(String cardnumber) {
        // access to merchant system's web service
        // omitted for brevity...
        return false;
    }
}
```

Extract and Override Call

Break dependence in a local method

Dependencies

1.

```
public void rebindStyles() {
   styles = StyleMaster.formStyles(template, id);
   ...
}
```

2.

```
public void rebindStyles() {
   styles = formStyles(template, id);
   ...
}

protected List formStyles(StyleTemplate template, int id) {
    return StyleMaster.formStyles(template, id);
}
```

Create a new method and copy the call

3.

```
public class TestingPageLayout extends PageLayour {

    protected List formStyles(StyleTemplate template, int id) {
        ...
        return new ArrayList();
    }
}
```

Introduce a testing subclass and override this new method

Extract Interface

```
Transaction t = new PaydayTransaction(getTestingDatabase(), new TransactionLog());
```

dependencies that we want to get rid of

```
void testPayday()
{
    FakeTransactionLog aLog = new FakeTransactionLog();
    Transaction t = new PaydayTransaction(getTestingDatabase(), aLog);
    t.run();

    assertEquals(getSampleCheck(12), getTestingDatabase().findCheck(12));
}
```

1. Extract Interface of TransactionLog
2. Implement fake class

Seeing Responsibilities

Seeing Responsibilities

Annoying: God Class

Classes that assume too many responsibilities is hard to maintain:

- Difficult to read
- Difficult to maintain
- Difficult to identify the impact of change

Remember: Single Responsibility Principle!

Responsibilities

- Doing:
  - Doing something itself:
    - Creating an object or doing a calculation
  - Initiating action in other objects
  - Controlling and co-ordinating activities in other projects
- Knowing:
  - about private data
  - about related objects
  - about things it derive or calculate

- Bigger responsibilities take several classes
  - Domain model helps with “knowing”
  - Interaction diagrams help with “doing”

Heuristic: Group methods

- Look for similar method names
- Write down all of the methods on a class, along with their access types
- Try to find ones that seem to go together

Heuristic: Hidden methods

- If a class has many of private and protected methods, it often indicates that there is another class in the class
- If you have the urge to test a private method, the method should not be private; and
- if making the method public bothers you, chances are, it is because it is part of a separate responsibility.

Heuristic: Decisions that can change

- Is there some way of doing something that seems hard-coded? for example:
    - Talking to a database
- Can you imagine it changing?

Heuristic: Internal relationships

- Are certain instance variables used by some methods and not others?
- Can we identify cohesive subsets of instance variables?

Craftsmanship

Software engineering?

- When you look at software, do you get the feeling that it has been developed without adult supervision?
- Is software more critical to your business and yet becoming bloated and buggier?
- Are projects taking longer and delivering less than promised?
- Is licensing software professionals really going to make any difference?
- The “software crisis” has existed for years and there is still a shortage of good software developers, because mechanical metaphors for software development don’t work

What is craftsmanship

- Nature of learn programming
  - Innate knowledge
  - Learned skills
  - Hard-won rules of thumb

The only way to learn is by learning from those more experience than you
This is craftsmanship

Craftsman model

Master

Journeyman

Apprentice

Software craftsmanship Manifesto

Manifesto for Software Craftsmanship

Raising the bar.

As aspiring Software Craftsmen we are raising the bar of professional software development by practicing it and helping others learn the craft. Through this work we have come to value:

Not only working software,
but also **well-crafted software**

Not only responding to change,
but also **steadily adding value**

Not only individuals and interactions,
but also **a community of professionals**

Not only customer collaboration,
but also **productive partnerships**

That is, in pursuit of the items on the left we have found the items on the right to be indispensable.

Software Craftsman

- Take responsibility
- No broken windows
- Quality is not an option
- Growing yourself
- Communicate!
- Professionalism

Practising

- Coding Dojo
  - Kata
  - Randoori
- Code Retreat

- Further broadening with OSS

Coding Dojo in Rio 2009 http://tinyurl.com/3pjyprt

Are you professional?

Don’t write bad code

Never be blocked

Use Good Tools

Avoid Debugging

Apprenticeship

Short Iterations

Incremental Improvement

Avoid Turgid Viscous Architectures

100% Code Coverage

Clean Code

QA should find nothing

Test through right interface

TDD

Abstract away volatility

Progressive Deepening

De-couple from others

Definition of Done

No Grand Redesigns

Manual Test Scripts are Immoral

Retrospectives

Sprint Retrospective

- Process improvement at end of every Sprint
- Facilitated by ScrumMaster
- What went well, what could be improved.
- Team devises solution to most vexing problems
- “Project Retrospectives,” Norman Kerth

Retrospective Techniques

- Check previous actions first. If not done, retrospect on them.
- Only select a couple of actions and really DO them.
- Focus on a constructive attitude.
  - “What can WE do”.

- Action planning: http://www.scrumalliance.org/articles/61-plan-of-action

Action

Long term goal:

Now-action:

Things to check in Retrospectives

- Actions:
  - Do we only have a few actions?
  - Are they considered useful?
  - Are they implemented?

- Is “done” extended?
- Updating our working agreements?
- Do we require:
  - Tasks in the Sprint backlog?
  - Items in the Product backlog?

What will you do on Monday?

References

Test-Driven Development

Refactoring

Object-Oriented Design

Acceptance-TDD

Extreme Programming

Automation

The Addison-Wesley Signature Series
A Martin Fowler Signature Book
Continuous Delivery
Reliable Software Releases through Build, Test, and Deployment Automation
Jez Humble,
David Farley

Pragmatic Project Automation

COVERS ANT 1.7
Steve Loughran
Erik Hatcher
ANT IN ACTION
Second Edition of
JAVA DEVELOPMENT WITH ANT
MANNING

Continuous Integration
Paul M. Duvall
Steve Matyas
Andrew Glover
Forewords by Martin Fowler and Paul Julius

Patterns

Not software

Code

Robert C. Martin Series
PRENTICE HALL
Clean Code
A Handbook of Agile Software Craftsmanship
Foreword by James O. Coplien
Robert C. Martin

Microsoft
CODE COMPLETE
2
Second Edition
A practical handbook of software construction
Steve McConnell

WRITING SOLID CODE
Microsoft's Techniques for Developing Bug-Free C Programs
STEVE MAGUIRE
Foreword by Dave Moore
Director of Development, Microsoft Corporation
Microsoft

Old

Attitude!

The Pragmatic Programmer
Andrew Hunt
David Thomas

The Pragmatic Bookshelf
PRAGMATIC LIFE
THE PASSIONATE PROGRAMMER
CREATING A REMARKABLE CAREER IN SOFTWARE DEVELOPMENT
CHAD FOWLER

The Productive Programmer
O'REILLY
Neal Ford

Robert C. Martin Series
PRENTICE HALL
The Clean Coder
A Code of Conduct for Professional Programmers
Robert C. Martin

Books - Teams

LEADING SELF-DIRECTED WORK TEAMS
A Guide to Developing New Team Leadership Skills
The Classic Book on Team Leadership Now Revised and Expanded
KIMBALL FISHER

LEADING TEAMS
SETTING THE STAGE FOR GREAT PERFORMANCES
J. RICHARD HACKMAN

NEW YORK TIMES BEST-SELLER
The FIVE DYSFUNCTIONS of a TEAM
A LEADERSHIP FABLE
PATRICK LENCIONI

Second Edition
FACILITATOR'S GUIDE TO PARTICIPATORY DECISION-MAKING
SAM KANER

TEAMWORK Is An INDIVIDUAL SKILL
Getting Your Work Done When Sharing Responsibility

Software that got harmed during this course

- **Card Me** http://sourceforge.net/projects/cardme/
- **Church Software** http://sourceforge.net/projects/churchsoftware/
- **Buddi** http://sourceforge.net/projects/buddi/
- **Bulkmailer** http://sourceforge.net/projects/bulkmailer/
- **GnuAccounting** http://sourceforge.net/projects/gnuaccounting/
- **StudyTrainer** http://sourceforge.net/projects/studytrainer/
- **University ERP System** http://sourceforge.net/projects/university/
- **Yapbam** http://sourceforge.net/projects/yapbam/
- **Yamm** http://sourceforge.net/projects/yammjava/

We mean no harm to the open source software movement, these open source projects, nor their authors. We are happy that they've contributed their source code to the software development community at large and that we have the opportunity to learn from others.