TOOLS FOR (JAVA) DEVELOPMENT INDUSTRIALIZATION

MASTER 1 ICE, 2017-2018

BENOIT COMBEMALE
PROFESSOR, UNIV. TOULOUSE, FRANCE

HTTP://COMBEMALE.FR
BENOIT.COMBEMALE@IRIT.FR
@BCOMBEMALE

Outils et Méthodes

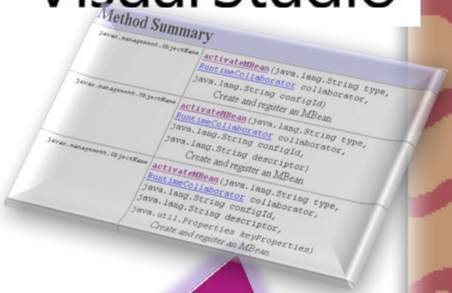
Software Engineering

Visual Basic

Code::Blocks Studio

eclipse

Microsoft® Visual Studio®



Method Summary

javax.management.ObjectName	activateMBean(java.lang.String type, RuntimeCollaborator collaborator, java.lang.String configId) Create and register an MBean
javax.management.ObjectName	activateMBean(java.lang.String type, RuntimeCollaborator collaborator, java.lang.String configId, java.lang.String descriptor) Create and register an MBean
javax.management.ObjectName	activateMBean(java.lang.String type, RuntimeCollaborator collaborator, java.lang.String configId, java.lang.String descriptor, java.util.Properties keyProperties) Create and register an MBean

<APACHE ANT>

JUnit.org

git

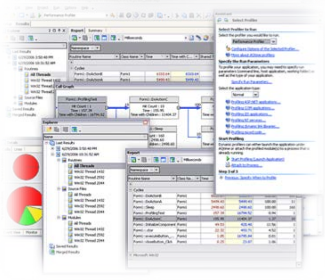
SUBVERSION

CVS

http://logging.apac

Logging Service

LOG4J™



Highlights

- Documentation
- Logging
- Testing
- Static analysis
- Refactoring
- Build automation
- Versioning
- Continuous integration

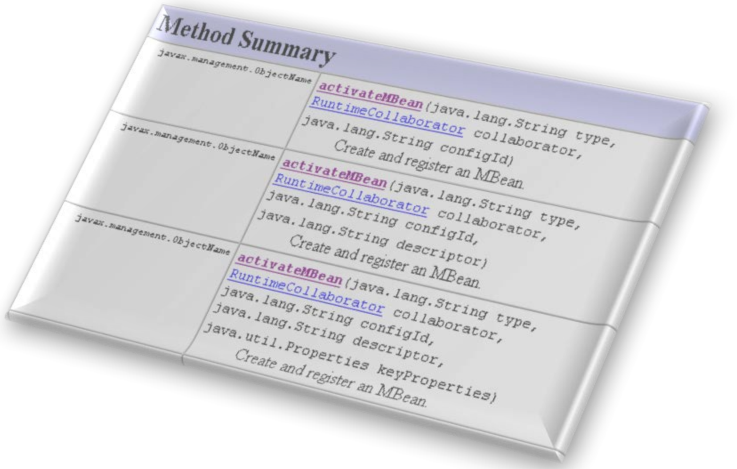
Documentation and Source Code

Documentation

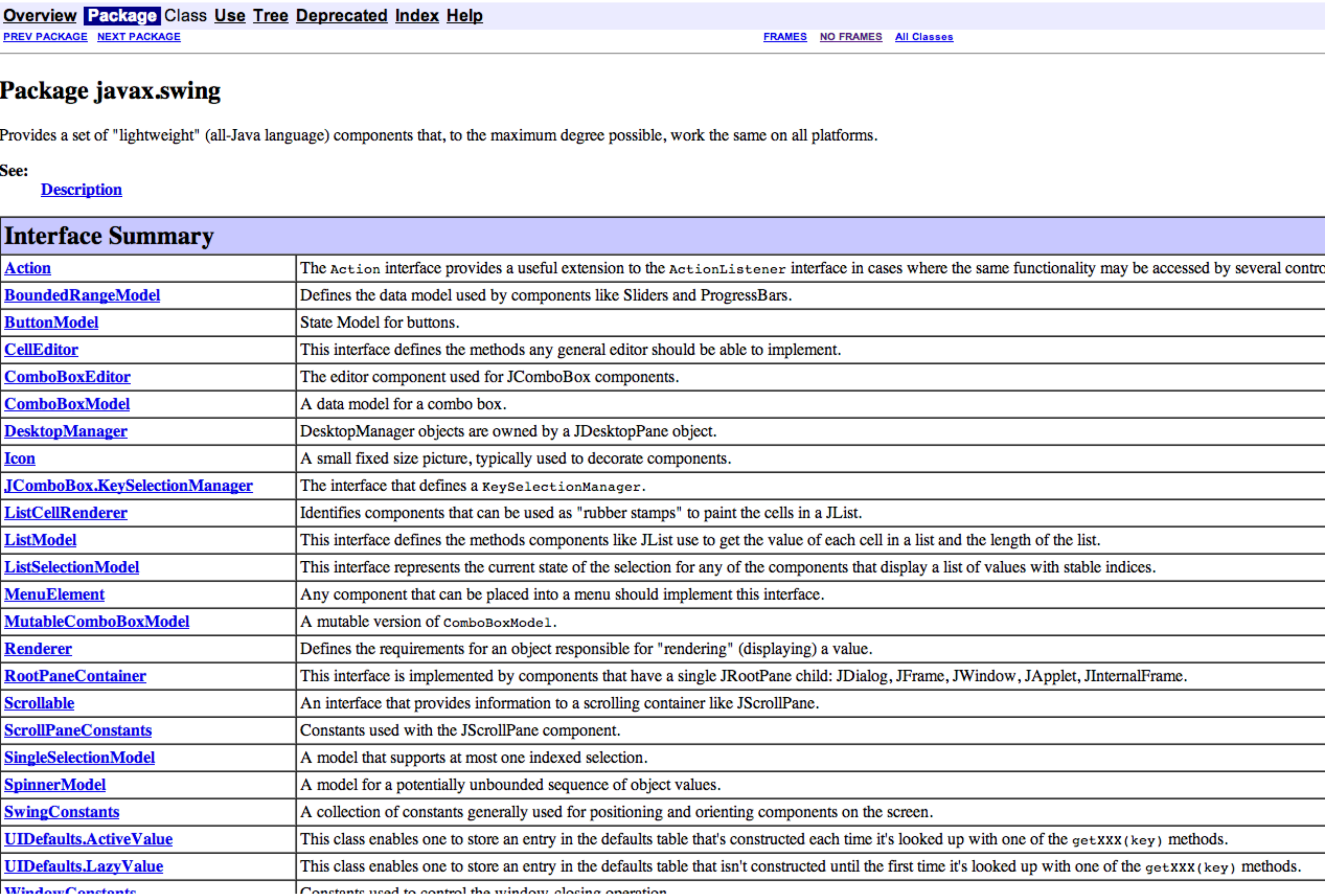
- Source code: one of the best artefact for documenting a project
- Javadoc (JDK)
  - Automatic **generation** of HTML documentation
  - Using comments in java files
- Syntax

```
/**
 * This is a <b>doc</b> comment.
 * @see java.lang.Object
 * @todo fix {@underline this !}
 */
```

- Includes
  - class hierarchy, interfaces, packages
  - detailed summary of class, interface, methods, attributes
- Note
  - Add doc generation to your favorite **compile chain**



Method Summary	
javax.management.ObjectName	activateMBean(java.lang.String type, RuntimeCollaborator collaborator, java.lang.String configId) Create and register an MBean.
javax.management.ObjectName	activateMBean(java.lang.String type, RuntimeCollaborator collaborator, java.lang.String configId, java.lang.String descriptor) Create and register an MBean.
javax.management.ObjectName	activateMBean(java.lang.String type, RuntimeCollaborator collaborator, java.lang.String configId, java.lang.String descriptor, java.util.Properties keyProperties) Create and register an MBean.



Overview **Package** Class **Use** **Tree** **Deprecated** **Index** **Help**

PREV PACKAGE NEXT PACKAGE FRAMES NO FRAMES All Classes

Package javax.swing

Provides a set of "lightweight" (all-Java language) components that, to the maximum degree possible, work the same on all platforms.

See:
Description

Interface Summary	
Action	The `Action` interface provides a useful extension to the `ActionListener` interface in cases where the same functionality may be accessed by several contro
BoundedRangeModel	Defines the data model used by components like Sliders and ProgressBars.
ButtonModel	State Model for buttons.
CellEditor	This interface defines the methods any general editor should be able to implement.
ComboBoxEditor	The editor component used for JComboBox components.
ComboBoxModel	A data model for a combo box.
DesktopManager	DesktopManager objects are owned by a JDesktopPane object.
Icon	A small fixed size picture, typically used to decorate components.
JComboBox.KeySelectionManager	The interface that defines a `KeySelectionManager`.
ListCellRenderer	Identifies components that can be used as "rubber stamps" to paint the cells in a JList.
ListModel	This interface defines the methods components like JList use to get the value of each cell in a list and the length of the list.
ListSelectionModel	This interface represents the current state of the selection for any of the components that display a list of values with stable indices.
MenuElement	Any component that can be placed into a menu should implement this interface.
MutableComboBoxModel	A mutable version of `ComboBoxModel`.
Renderer	Defines the requirements for an object responsible for "rendering" (displaying) a value.
RootPaneContainer	This interface is implemented by components that have a single JRootPane child: JDialog, JFrame, JWindow, JApplet, JInternalFrame.
Scrollable	An interface that provides information to a scrolling container like JScrollPane.
ScrollPaneConstants	Constants used with the JScrollPane component.
SingleSelectionModel	A model that supports at most one indexed selection.
SpinnerModel	A model for a potentially unbounded sequence of object values.
SwingConstants	A collection of constants generally used for positioning and orienting components on the screen.
UIDefaults.ActiveValue	This class enables one to store an entry in the defaults table that's constructed each time it's looked up with one of the `getXXX(key)` methods.
UIDefaults.LazyValue	This class enables one to store an entry in the defaults table that isn't constructed until the first time it's looked up with one of the `getXXX(key)` methods.
WindowConstants	Constants used to control the window closing operation.

public class **JFrame**
extends Frame
implements WindowConstants, Accessible, RootPaneContainer

An extended version of `java.awt.Frame` that adds support for the JFC/Swing component architecture. You can find task-oriented documentation about using `JFrame` in *The Java Tutorial*, in the section How to Make Frames.

The `JFrame` class is slightly incompatible with `Frame`. Like all other JFC/Swing top-level containers, a `JFrame` contains a `JRootPane` as its only child. The **content pane** provided by the root pane should, as a rule, contain all the nc
the AWT `Frame` case. For example, to add a child to an AWT frame you'd write:

```
        frame.add(child);
```

However using `JFrame` you need to add the child to the `JFrame's` content pane instead:

```
        frame.getContentPane().add(child);
```

The same is true for setting layout managers, removing components, listing children, and so on. All these methods should normally be sent to the content pane instead of the JFrame itself. The content pane will always be non-null. A
exception. The default content pane will have a BorderLayout manager set on it.

update

```
public void update(Graphics g)
```

Just calls `paint(g)`. This method was overridden to prevent an unnecessary call to clear the background.

Overrides:

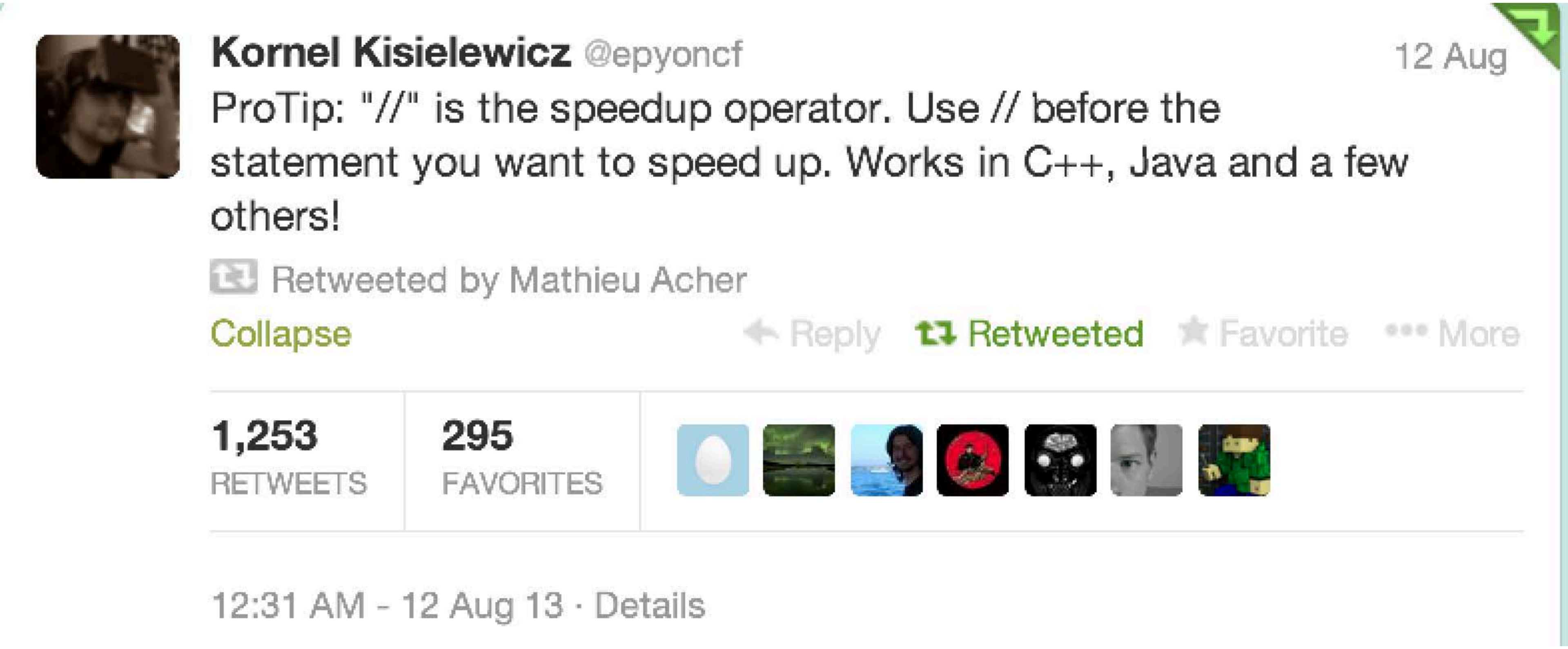
`update` in class `Container`

Parameters:

`g` - the Graphics context in which to paint

See Also:

`Component.update(Graphics)`



Kornel Kisielewicz @epyoncf 12 Aug

ProTip: "//" is the speedup operator. Use // before the statement you want to speed up. Works in C++, Java and a few others!

Retweeted by Mathieu Acher

1,253 RETWEETS 295 FAVORITES

12:31 AM - 12 Aug 13 · Details

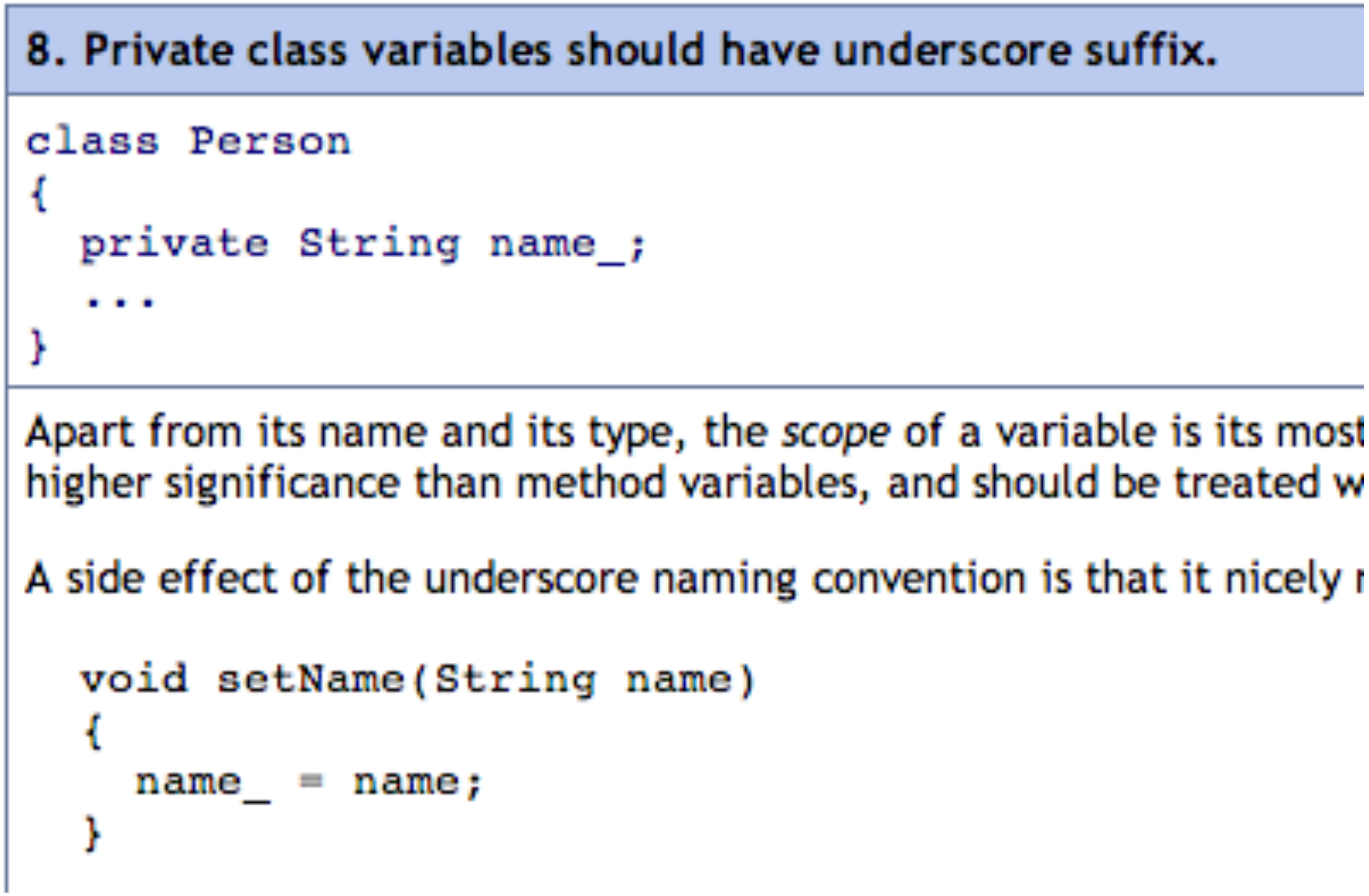
Coding Conventions

- Rules on the coding style :
  - Apache, Oracle and others template
    - e.g. http://www.oracle.com/technetwork/java/codeconv-138413.html
    - http://geosoft.no/development/javastyle.html
- Verification tools
  - CheckStyle, PMD, JackPot, Spoon Vsuite…
  - Some integrated into IDEs

Why Coding Standards are Important?

- Lead to greater **consistency** within your code and the code of your teammates
- Easier to **understand**
- Easier to **develop**
- Easier to **maintain**
- Reduces overall cost of application

Example



8. Private class variables should have underscore suffix.

```
class Person
{
  private String name_;
  ...
}
```

Apart from its name and its type, the *scope* of a variable is its most
higher significance than method variables, and should be treated w

A side effect of the underscore naming convention is that it nicely r

```
  void setName(String name)
  {
    name_ = name;
  }
```

Tools to Improve your Source code

- Formatting tools
  - Indenteurs (Jindent), beautifiers, stylers (JavaStyle), …
- « Bug fixing » tools
  - Spoon VSuite, Findbugs (sourceforge) …
- Quality report tools : code metrics
  - Number of Non Comment Code Source, Number of packages, Cyclomatic numbers, …
    - JavaNCCS, Eclipse Metrics …

Logging

Logging

- Logging is chronological and systematic record of data processing events in a program
  - e.g. the Windows Event Log
- Logs can be saved to a persistent medium to be studied at a later time
- Use logging in the development phase:
  - Logging can help you **debug** the code
- Use logging in the production environment:
  - Helps you **troubleshoot problems**

Logging, why? (claims)

- Logging is easier than debugging
- Logging is faster than debugging
- Logging can work in environments where debugging is not supported
- Can work in production environments
- Logs can be referenced anytime in future as the data is stored

Logging Methods, How?

- The evil System.out.println()

- Custom Solution to Log to various datastores, eg text files, db, etc…

- Use Standard APIs
  - Don't reinvent the wheel

Log4J

- Popular logging frameworks for Java
- Designed to be reliable, fast and extensible
- Simple to understand and to use API
- Allows the developer to control which log statements are output with arbitrary granularity
- Fully configurable at runtime using external configuration files

Log4J Architecture

- Log4J has three main components: loggers, appenders and layouts
  - Loggers
    - Channels for printing logging information
  - Appenders
    - Output destinations (console, File, Database, Email/SMS Notifications, Log to a socket, and many others…)
  - Layouts
    - Formats that appenders use to write their output
- Priorities

Logger

- Responsible for Logging
- Accessed through java code
- Configured Externally
- Every Logger has a name
- Prioritize messages based on level
  - TRACE < DEBUG < INFO < WARN < ERROR < FATAL
- Usually named following dot convention like java classes do.
  - Eg com.foo.bar.ClassName
- Follows inheritance based on name

Logger API

- Factory methods to get Logger
  - `Logger.getLogger(Class c)`
  - `Logger.getLogger(String s)`
- Method used to log message
  - `trace(), debug(), info(), warn(), error(), fatal()`
  - Details
    - `void debug(java.lang.Object message)`
    - `void debug(java.lang.Object message, java.lang.Throwable t)`
  - Generic Log method
    - `void log(Priority priority, java.lang.Object message)`

    -
      ```
      void log(Priority priority,
                    java.lang.Object message, java.lang.Throwable t)
      ```

Root Logger

- The root logger resides at the top of the logger hierarchy. It is exceptional in two ways:
  1. it always exists,
  2. it cannot be retrieved by name.

- Logger.getRootLogger()

Appender

- Appenders put the log messages to their actual destinations.
- No programatic change is require to configure appenders
- Can add multiple appenders to a Logger.
- Each appender has its Layout.
- ConsoleAppender, DailyRollingFileAppender, FileAppender, JDBCAppender, JMSAppender, NTEventLogAppender, RollingFileAppender, SMTPAppender, SocketAppender, SyslogAppender, TelnetAppender

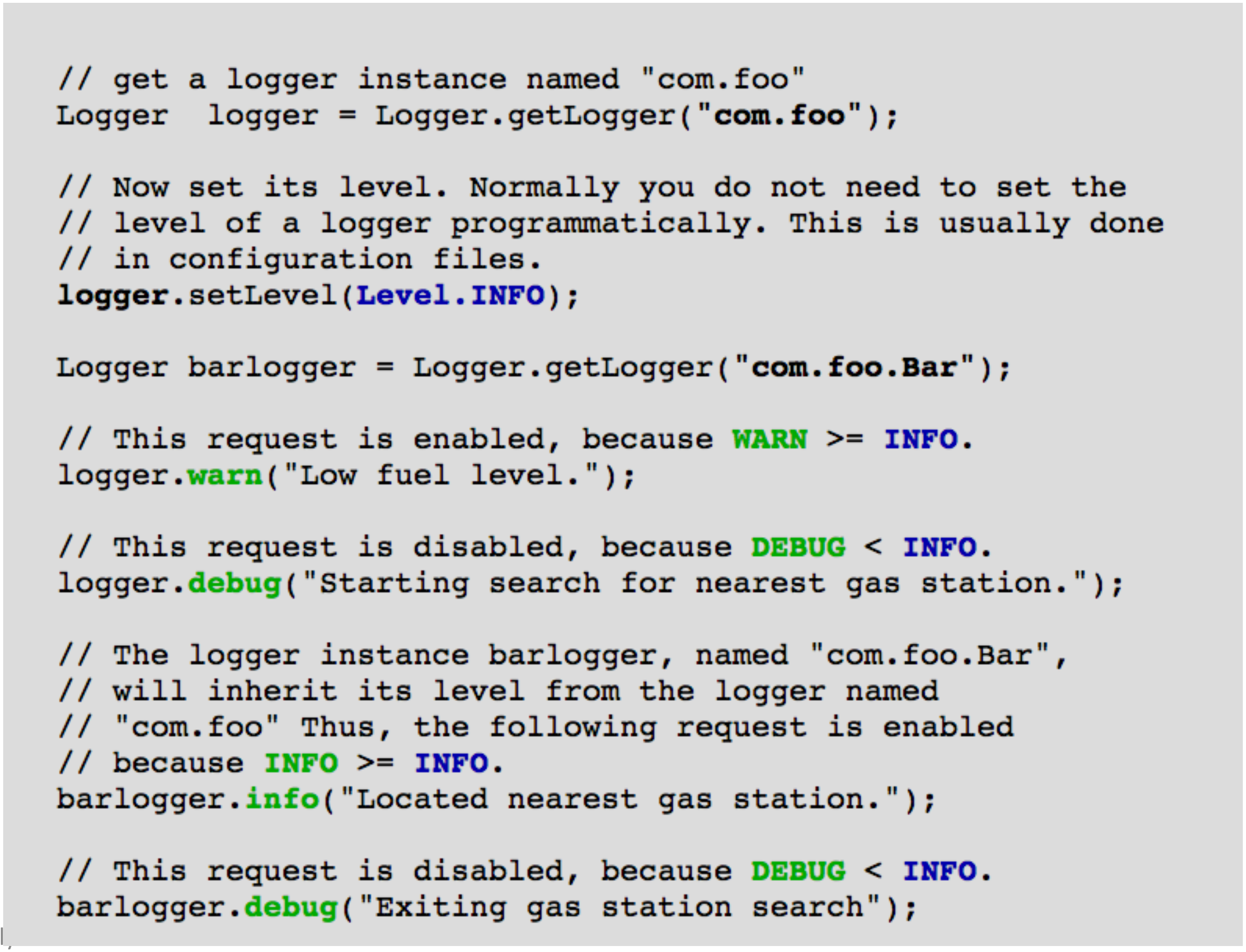
Layout

- Used to customize the format of log output.
- Eg. HTMLLayout, PatternLayout, SimpleLayout, XMLLayout
- Most commonly used is PatternLayout
  - Uses C-like syntax to format.
    - `Eg. "%-5p [%t]: %m%n`
    - `DEBUG [main]: Message 1 WARN [main]: Message 2`

Log4j Basics

- Who will log the messages?
  - The *Loggers*
- What decides the priority of a message?
  - *Level*
- Where will it be logged?
  - Decided by *Appender*
- In what format will it be logged?
  - Decided by *Layout*

Log4j in Action

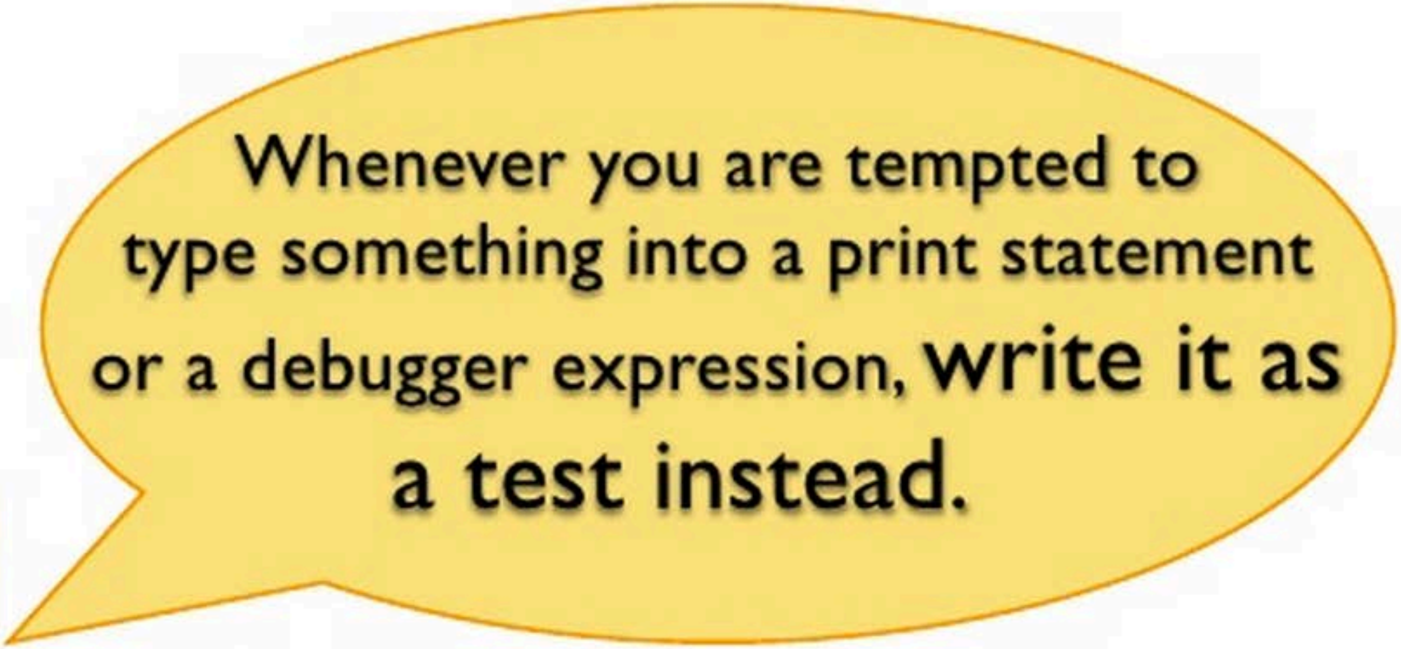


```
// get a logger instance named "com.foo"
Logger  logger = Logger.getLogger("com.foo");

// Now set its level. Normally you do not need to set the
// level of a logger programmatically. This is usually done
// in configuration files.
logger.setLevel(Level.INFO);

Logger barlogger = Logger.getLogger("com.foo.Bar");

// This request is enabled, because WARN >= INFO.
logger.warn("Low fuel level.");

// This request is disabled, because DEBUG < INFO.
logger.debug("Starting search for nearest gas station.");

// The logger instance barlogger, named "com.foo.Bar",
// will inherit its level from the logger named
// "com.foo" Thus, the following request is enabled
// because INFO >= INFO.
barlogger.info("Located nearest gas station.");

// This request is disabled, because DEBUG < INFO.
barlogger.debug("Exiting gas station search");
```

Log4j Optimization & Best Practises

- Use logger as private static variable
- Only one instance per class
- Name logger after class name
- Don’t use too many appenders
- Don’t use time-consuming conversion patterns
- Use Logger.isDebugEnabled() if need be
- Prioritize messages with proper levels

You can't test everything
(so one advice by Martin Fowler)


Whenever you are tempted to type something into a print statement or a debugger expression, write it as a test instead.

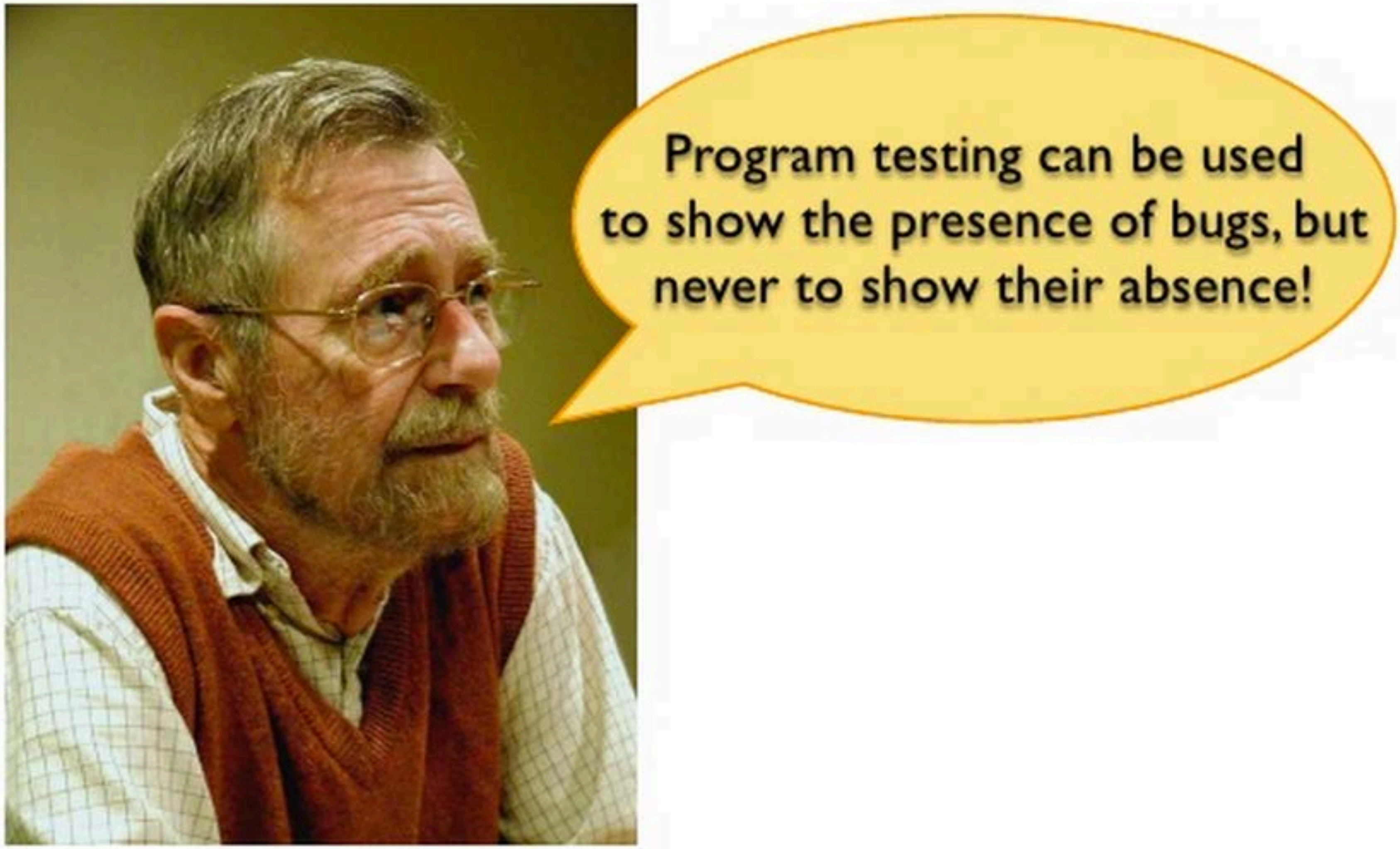
Testing and Static Analysis

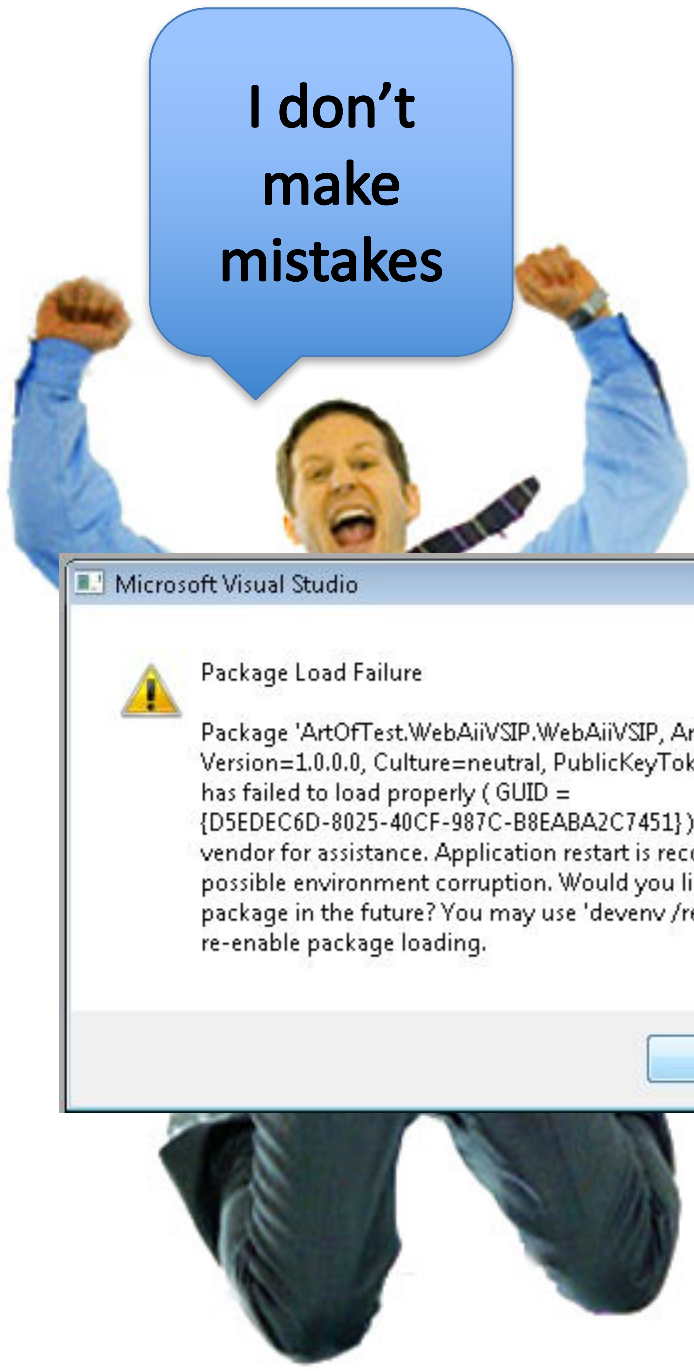
…the activity of finding out whether a piece of code (a method, class or program) produces the intended behavior

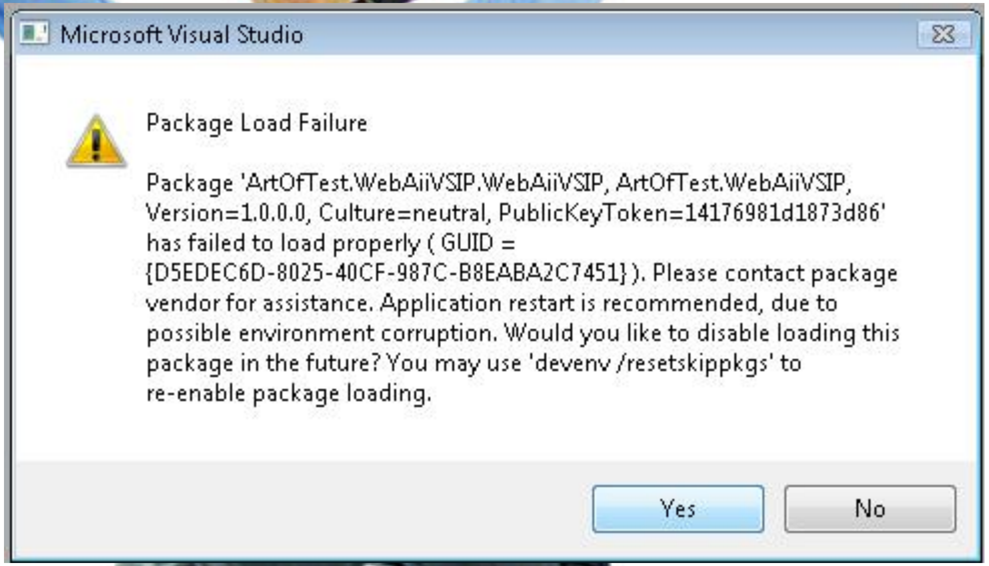
Your **hope** as a programmer

« A program does exactly what you expected to do »

Djikstra


Program testing can be used to show the presence of bugs, but never to show their absence!





Microsoft Visual Studio

Package Load Failure

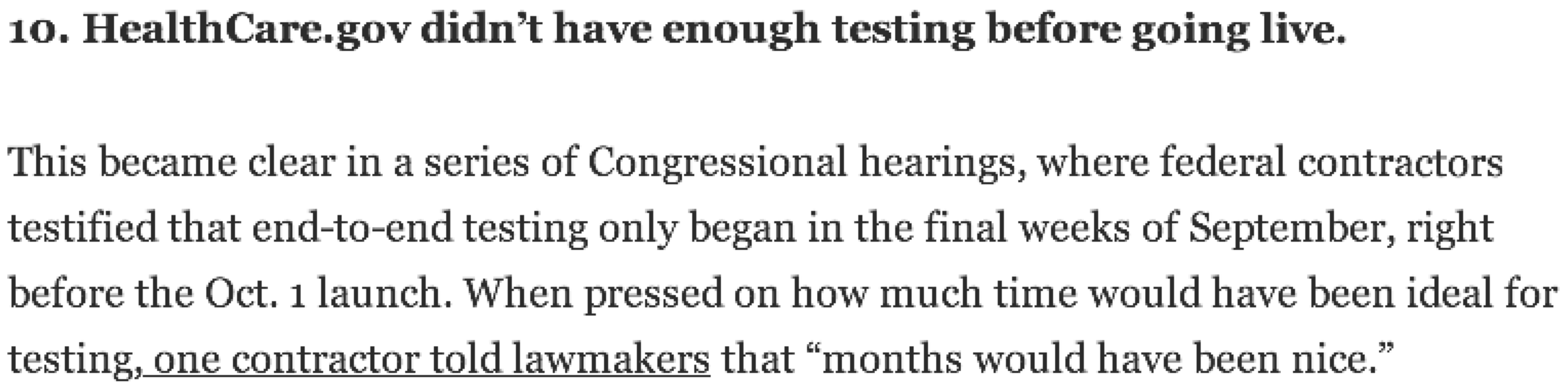
Package 'ArtOfTest.WebAiiVSIP.WebAiiVSIP, ArtOfTest.WebAiiVSIP, Version=1.0.0.0, Culture=neutral, PublicKeyToken=14176981d1873d86' has failed to load properly (GUID = {D5EDEC6D-8025-40CF-987C-B8EABA2C7451}). Please contact package vendor for assistance. Application restart is recommended, due to possible environment corruption. Would you like to disable loading this package in the future? You may use 'devenv /resetskippkgs' to re-enable package loading.

Master 2 (Apprentis)

15 « jobs », 15 aim at
Testing (critical or non critical) applications
Correcting anomalies and ensuring that they won’t appear in the future
Maintaining

« 1 day of producing code
= 3 days of testing code »

« 70% of a software project = maintainance »

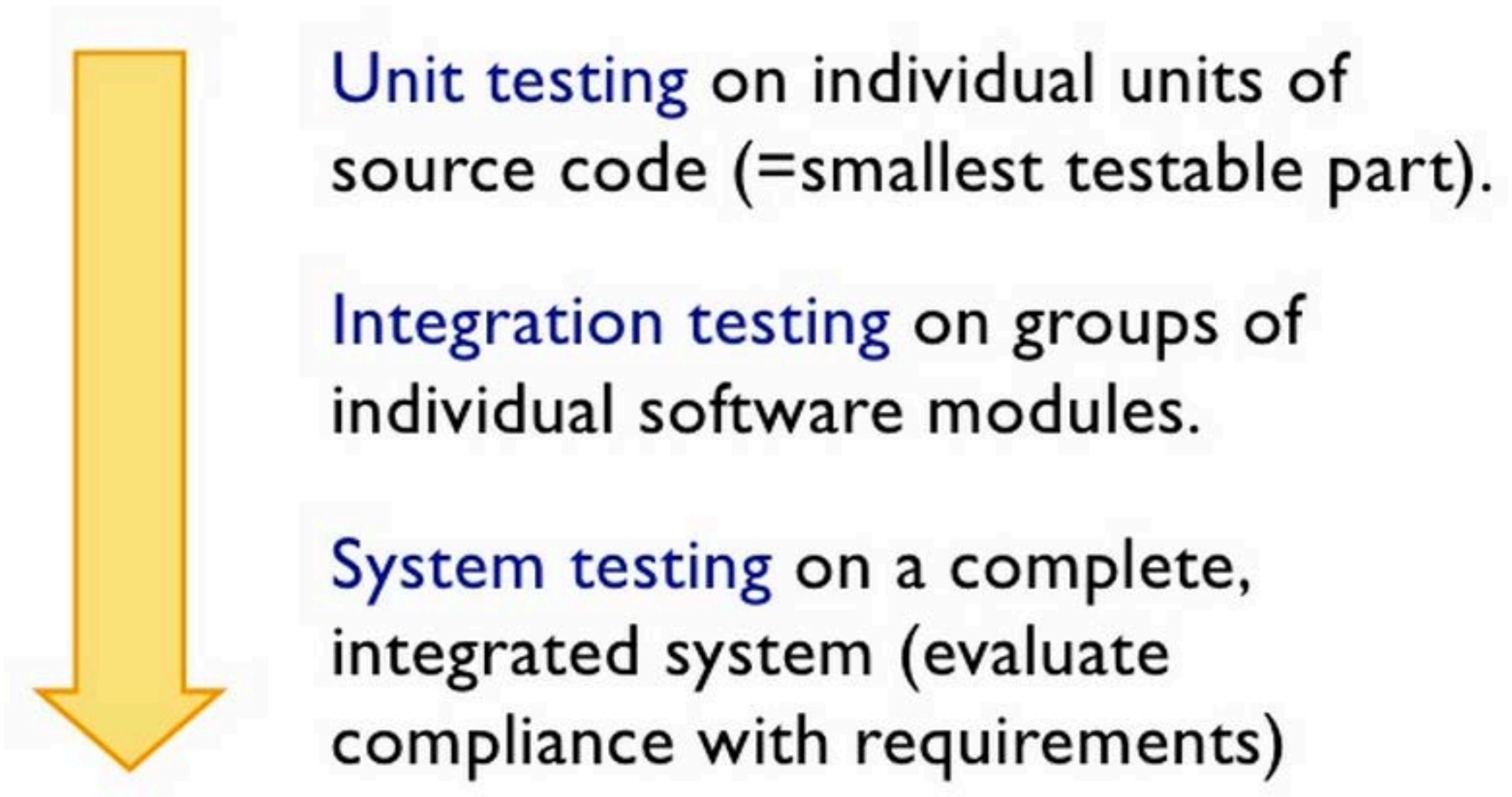


10. HealthCare.gov didn't have enough testing before going live.

This became clear in a series of Congressional hearings, where federal contractors testified that end-to-end testing only began in the final weeks of September, right before the Oct. 1 launch. When pressed on how much time would have been ideal for testing, one contractor told lawmakers that "months would have been nice."

http://www.washingtonpost.com/blogs/wonkblog/wp/2013/11/01/thirty-one-things-we-learned-in-healthcare-govs-first-31-days/

Test phases



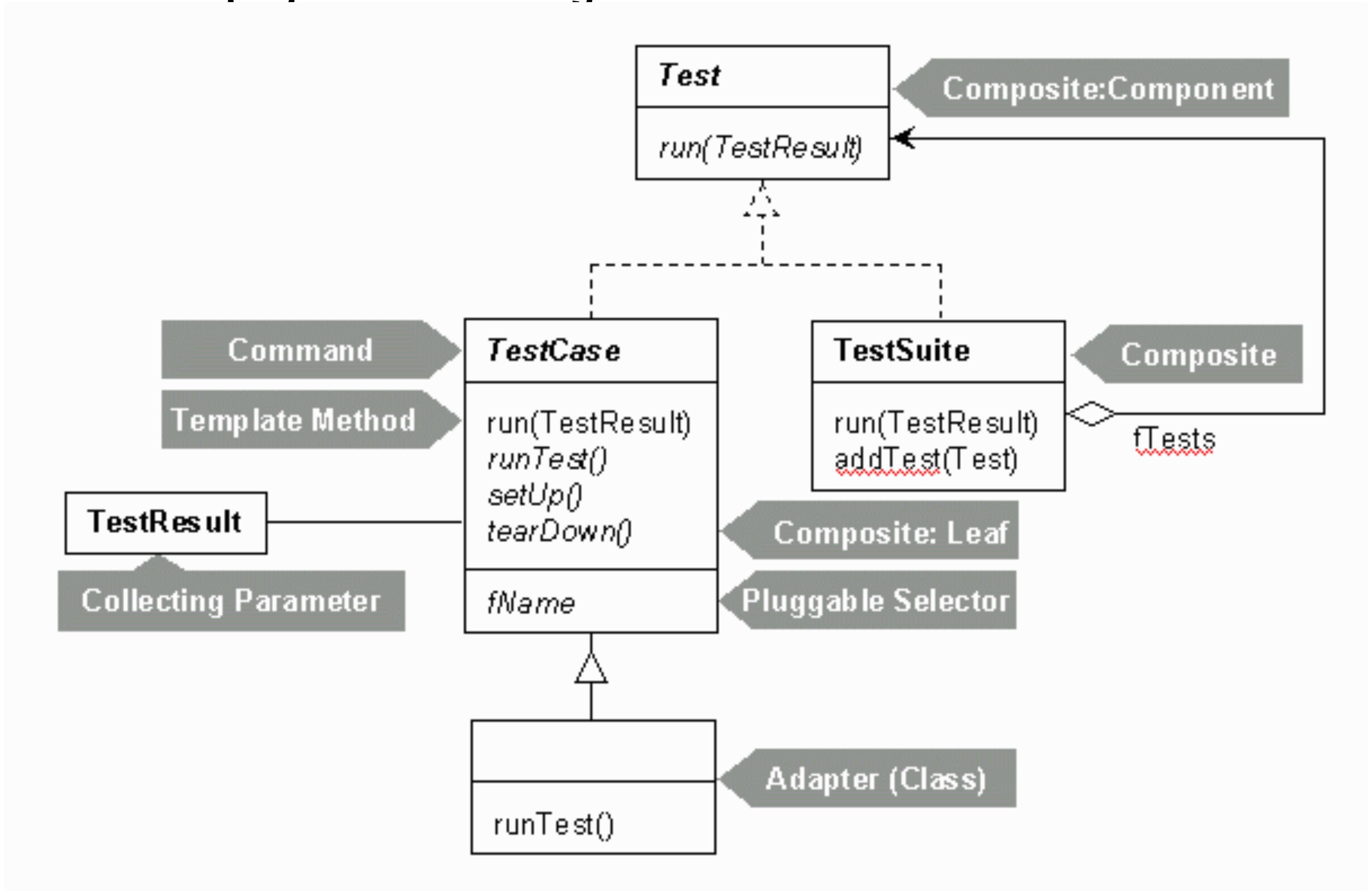
Unit testing on individual units of source code (=smallest testable part).

Integration testing on groups of individual software modules.

System testing on a complete, integrated system (evaluate compliance with requirements)

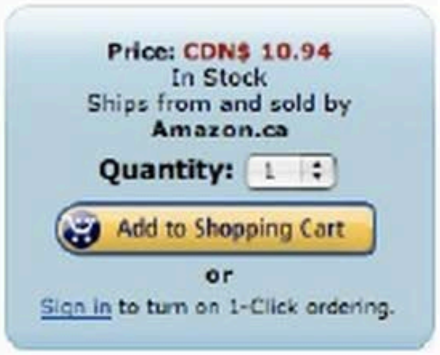
Junit and... Design Patterns

http://junit.sourceforge.net/doc/cookstour/cookstour.htm



Running example

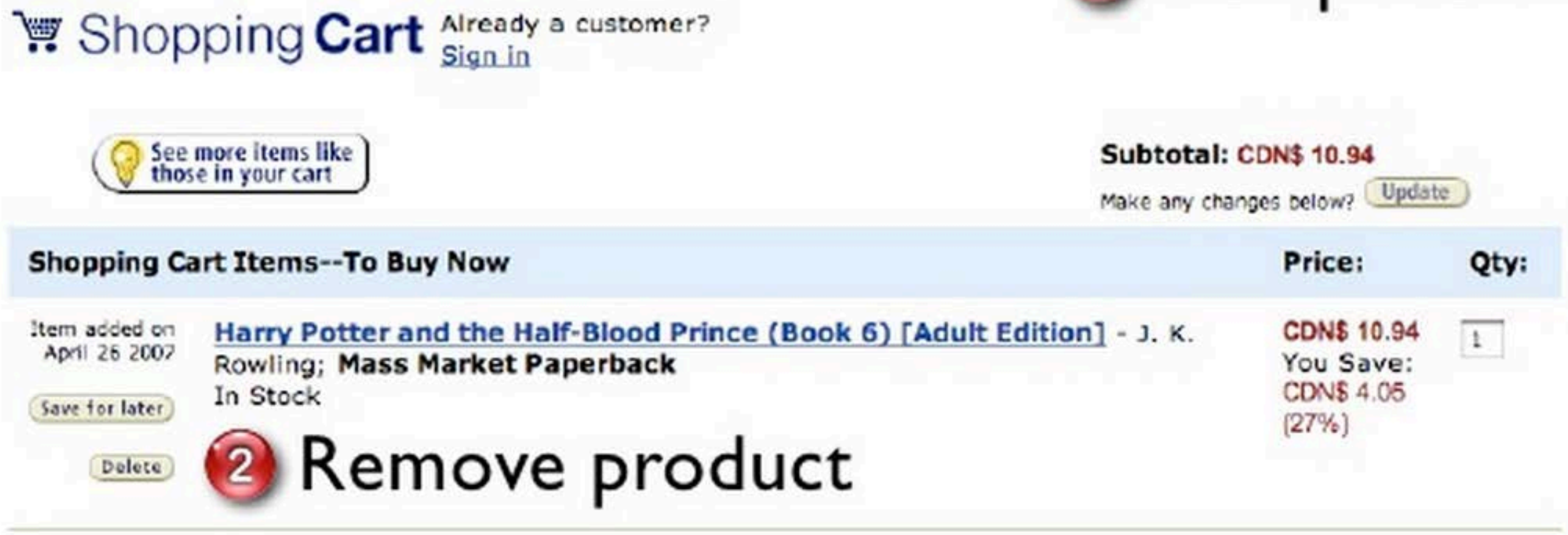
1. Set of products
2. Number of products
3. Balance



Price: CDN$ 10.94
In Stock
Ships from and sold by Amazon.ca

Quantity: 1

1 Add product



Shopping Cart Already a customer? Sign in

Subtotal: CDN$ 10.94

Make any changes below? Update

Shopping Cart Items--To Buy Now		Price:	Qty:
Item added on April 26 2007 Save for later Delete	Harry Potter and the Half-Blood Prince (Book 6) [Adult Edition] - J. K. Rowling; Mass Market Paperback In Stock	CDN$ 10.94 You Save: CDN$ 4.05 (27%)	1

2 Remove product

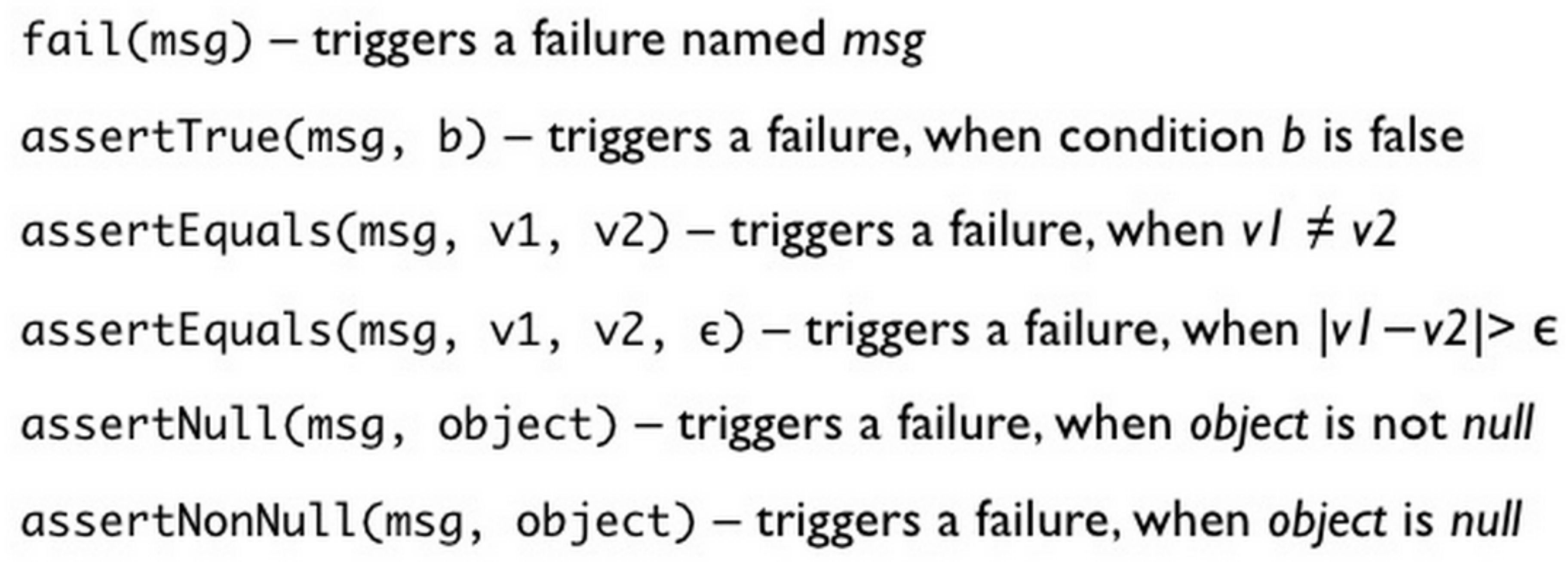
Init

Constructor + Set up and tear down of fixture.

```java
import junit.framework.Test;
import junit.framework.TestCase;
import junit.framework.TestSuite;

public class ShoppingCartTest extends TestCase {

    private ShoppingCart _bookCart;

    // Creates a new test case
    public ShoppingCartTest(St
        super(name);
    }
```

```java
// Creates test environment (fixture).
// Called before every testX() method.
protected void setUp() {
    _bookCart = new ShoppingCart();

    Product book = new Product("Harry Potter", 23.95);
    _bookCart.addItem(book);
}

// Releases test environment (fixture).
// Called after every testX() method.
protected void tearDown() {
    _bookCart = null;
}
```

Assertions



`fail(msg)` – triggers a failure named *msg*

`assertTrue(msg, b)` – triggers a failure, when condition *b* is false

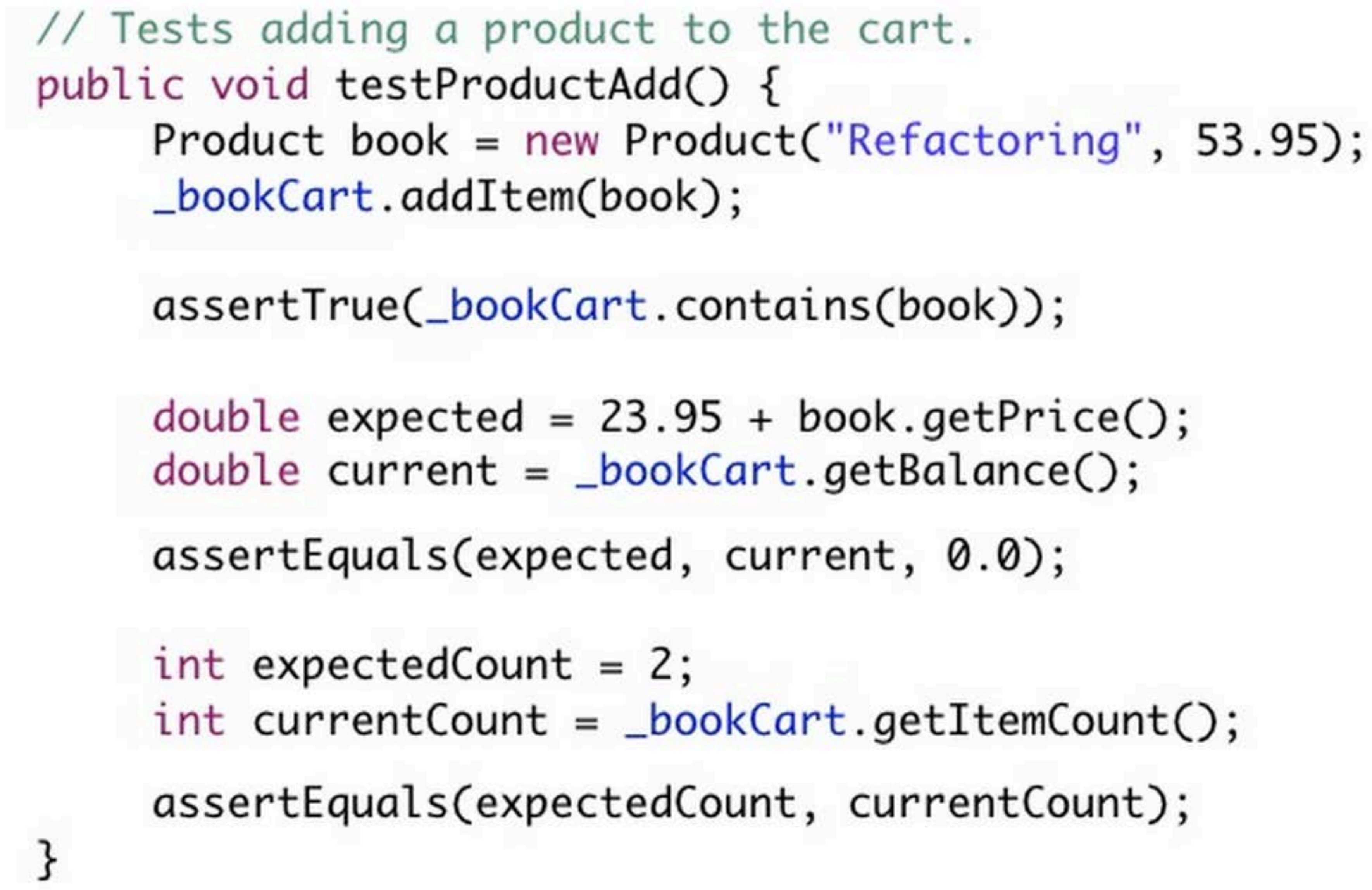
`assertEquals(msg, v1, v2)` – triggers a failure, when $v1 \neq v2$

`assertEquals(msg, v1, v2, ϵ)` – triggers a failure, when $|v1 - v2| > \epsilon$

`assertNull(msg, object)` – triggers a failure, when *object* is not *null*

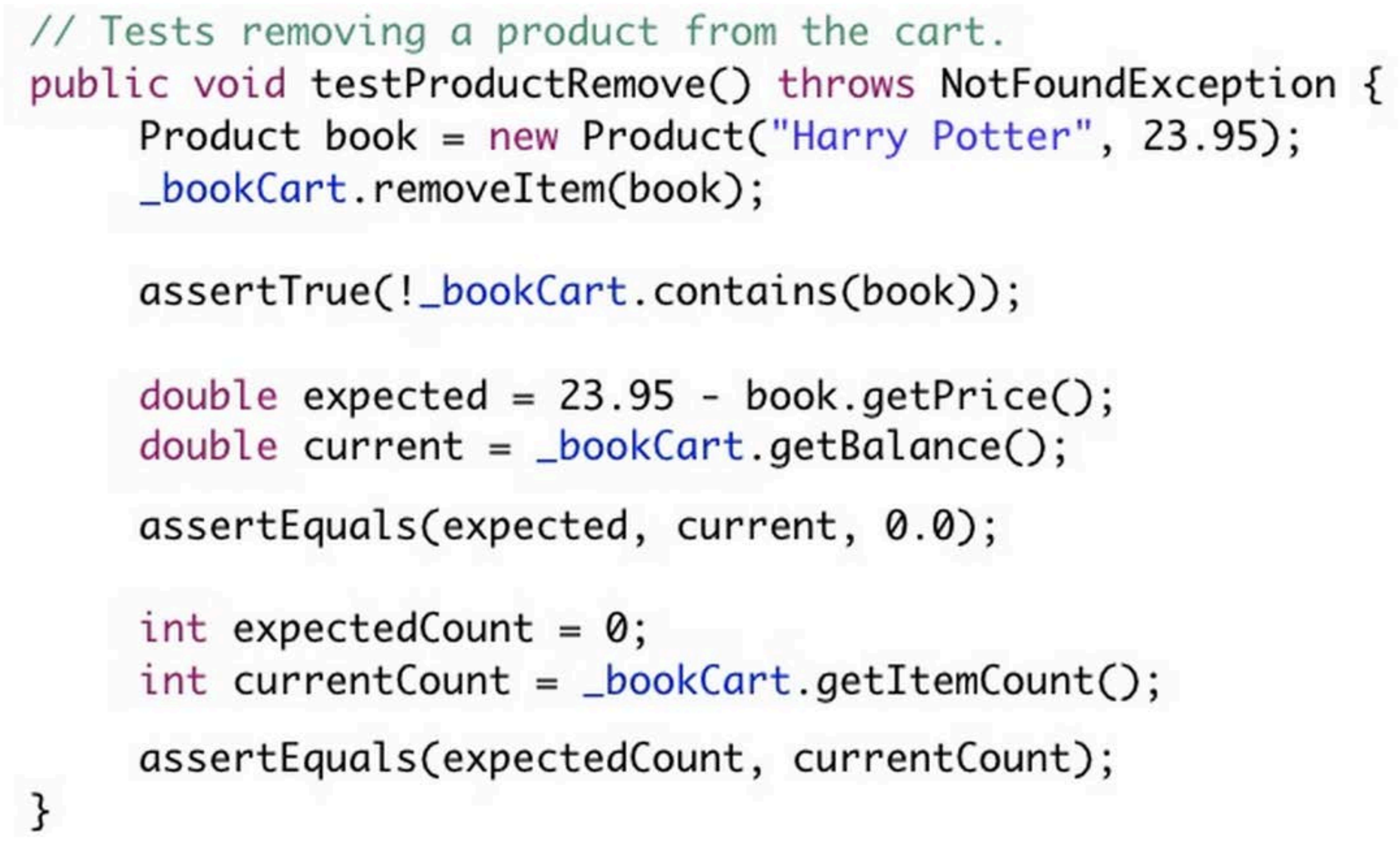
`assertNonNull(msg, object)` – triggers a failure, when *object* is *null*

Example #1



```
// Tests adding a product to the cart.
public void testProductAdd() {
    Product book = new Product("Refactoring", 53.95);
    _bookCart.addItem(book);

    assertTrue(_bookCart.contains(book));

    double expected = 23.95 + book.getPrice();
    double current = _bookCart.getBalance();

    assertEquals(expected, current, 0.0);

    int expectedCount = 2;
    int currentCount = _bookCart.getItemCount();

    assertEquals(expectedCount, currentCount);
}
```

Example #2

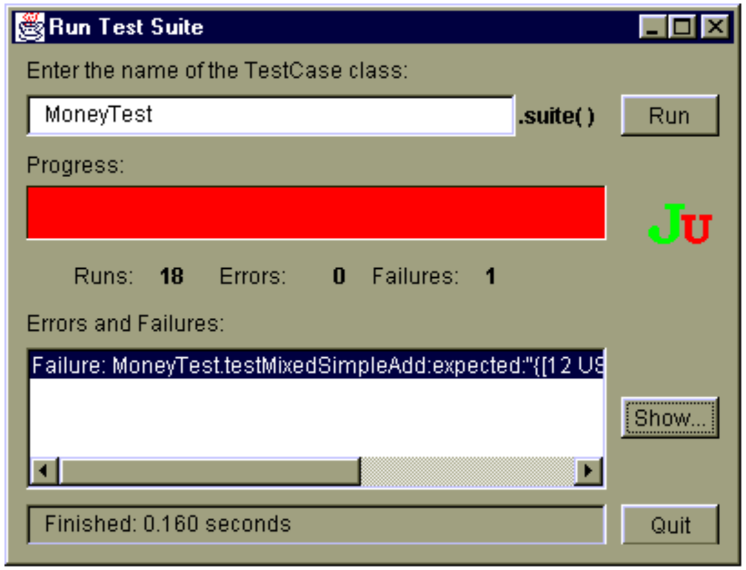


```
// Tests removing a product from the cart.
public void testProductRemove() throws NotFoundException {
    Product book = new Product("Harry Potter", 23.95);
    _bookCart.removeItem(book);

    assertTrue(!_bookCart.contains(book));

    double expected = 23.95 - book.getPrice();
    double current = _bookCart.getBalance();

    assertEquals(expected, current, 0.0);

    int expectedCount = 0;
    int currentCount = _bookCart.getItemCount();

    assertEquals(expectedCount, currentCount);
}
```

```java
public static Test suite() {

    // Here: add all testX() methods to the suite (reflection).
    TestSuite suite = new TestSuite(ShoppingCartTest.class);

    // Alternative: add methods manually (prone to error)
    // TestSuite suite = new TestSuite();
    // suite.addTest(new ShoppingCartTest("testEmpty"));
    // suite.addTest(new ShoppingCartTest("testProductAdd"));
    // suite.addTest(...);

    return suite;
}
```

Unit Test
JUnit 3, 4 & 5 http://www.junit.org

- Test pattern
  - Test, TestSuite, TestCase
  - Assertions (assertXX) that must be verified
- TestRunner
  - Chain tests and output a report.


Run Test Suite

Enter the name of the TestCase class:

MoneyTest .suite()

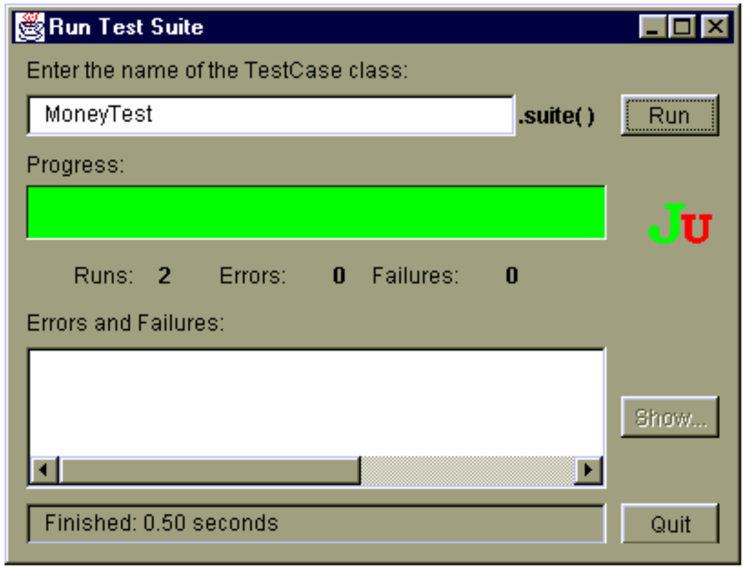
Progress:

Runs: 18 Errors: 0 Failures: 1

Errors and Failures:

Failure: MoneyTest.testMixedSimpleAdd:expected:"{[12 US

Finished: 0.160 seconds


Run Test Suite

Enter the name of the TestCase class:

MoneyTest .suite()

Progress:

Runs: 2 Errors: 0 Failures: 0

Errors and Failures:

Finished: 0.50 seconds

- See JUnit course:
  - **http://people.irisa.fr/Benoit.Combemale/teaching/vv**

Debugging

- Symbolic debugging
  - javac options: -g, -g:source,vars,lines
  - command-line debugger : jdb (JDK)
    - commands look like those of dbx
  - graphical « front-ends » for jdb (AGL)
  - Misc
    - Multi-threads, Cross-Debugging (-Xdebug) on remote VM , ...

Monitoring

- Tracer
  - TRACE options of the program
  - can slow-down .class with TRACE/←TRACE tests
    - solution : use a pre-compiler (excluding trace calls)
  - Kernel tools, like OpenSolaris DTrace (coupled with the JVM)

- Logger
  - Record events on a registry, to be used at execution time or later on (via some event handlers)
  - Tools
    - Apache Log4J, ObjectWeb MonoLog
    - Package java.util.logging since J2SE1.4
      - Logger, LogRecord, Handler

Validation (i)

- Assertion
  - Pre-Condition, Post-Condition, Invariant
    - EIFFEL, CLU ... built-in
    - Java since SE 1.4
- Tools for J2SE 1.3 et less (J2ME, JavaCard, ...)
  - AssertMate (Reliable Software Technologies)
    - http://www.ddj.com/articles/1998/9801d/9801d.htm#rel
  - JML (Java Modeling Language)
    - http://www.eecs.ucf.edu/~leavens/JML/
  - iContract (Reliable Systems)
  - ...

Design by Contract with JML (by Gary T. Leavens and Yoonsik Cheon)

Performances

- Measure/Analyze
  - Benchmark
  - java.awt.Robot (to build clients for testing)
  - Accounting : http://abone.unige.ch/jraf/index.htm
  - JProfiler, OptimiszeIt …
- Optimization
- See books:
  - Steve Wilson, Jeff Kesselman, « Java Platform Performance: Strategies and Tactics (The Java Series) », 1 edition (May 25, 2000), Addison-Wesley Pub Co; ISBN: 0-201-70969-4
  - Jack Shirazi, "Java Performance Tuning", Ed Oreilly, 2000, ISBN 0-596-00015-4

Choosing the JVM and JRE

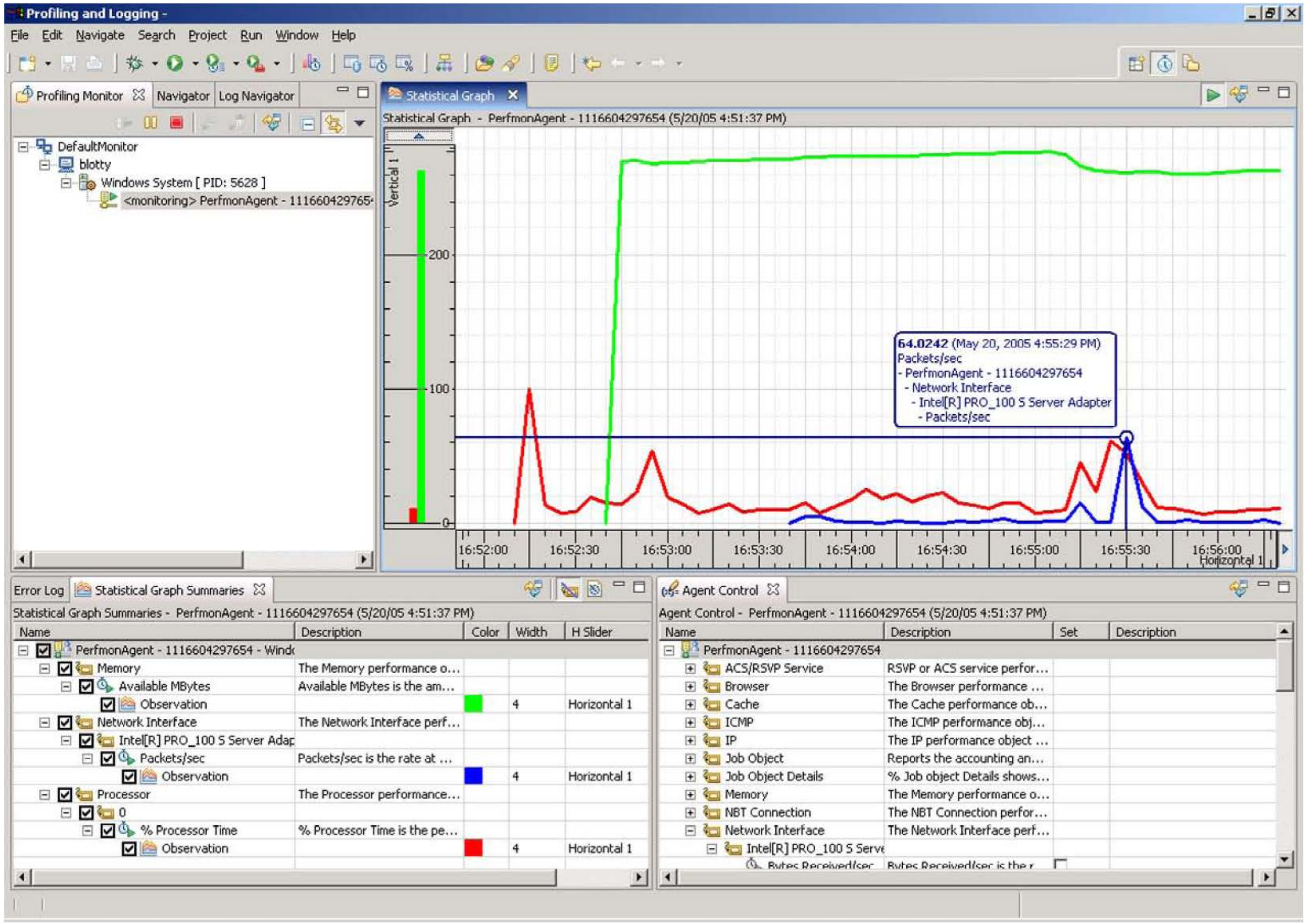
- Some criteria
  - License, redistribution, supports, performances, contraintes (embedded, servers, réal-time, …), runtimes, …
- Examples
  - Sun HotSpot JVM + JRE
  - MicroSoft JVM (deprecated)
  - IBM J9
  - BEA JRockit
  - HP Chai
  - Macromedia JRun
  - Blackdown JVM
  - Kaffe + GNU Classpath
  - CReME
  - Cacao http://www.cacaojvm.org/
  - Apache Harmony
  - JamVM
  - JRate
  - …

Measurements and Analysis of Performances

- Java Profiler
  - Use to profile an application
    - CPU usage, Memory, Network, time spent, and Garbage collection
    - In Total or by threads, or called methods

Exemple

Eclipse TPTP http://www.eclipse.org/tptp/


Profiling and Logging -
File Edit Navigate Search Project Run Window Help
Profiling Monitor
Navigator
Log Navigator
DefaultMonitor
blotty
Windows System [PID: 5628]
<monitoring> PerfmonAgent - 111660429765
Statistical Graph
Statistical Graph - PerfmonAgent - 1116604297654 (5/20/05 4:51:37 PM)
Vertical 1
200
100
0
16:52:00
16:52:30
16:53:00
16:53:30
16:54:00
16:54:30
16:55:00
16:55:30
16:56:00
Horizontal 1
64.0242 (May 20, 2005 4:55:29 PM)
Packets/sec
- PerfmonAgent - 1116604297654
- Network Interface
- Intel[R] PRO_100 S Server Adapter
- Packets/sec
Error Log
Statistical Graph Summaries
Agent Control

Statistical Graph Summaries - PerfmonAgent - 1116604297654 (5/20/05 4:51:37 PM)

Name	Description	Color	Width	H Slider
PerfmonAgent - 1116604297654 - Windc				
Memory	The Memory performance o...			
Available MBytes	Available MBytes is the am...			
Observation		🟩	4	Horizontal 1
Network Interface	The Network Interface perf...			
Intel[R] PRO_100 S Server Adap				
Packets/sec	Packets/sec is the rate at ...			
Observation		🟦	4	Horizontal 1
Processor	The Processor performance...			
0				
% Processor Time	% Processor Time is the pe...			
Observation		🟥	4	Horizontal 1

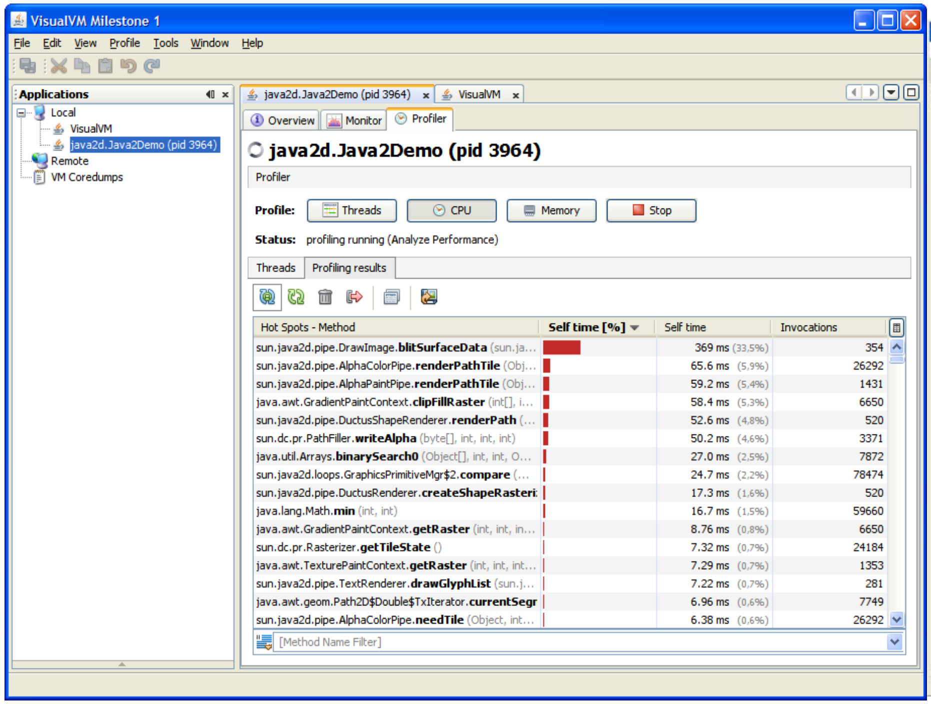
Agent Control - PerfmonAgent - 1116604297654 (5/20/05 4:51:37 PM)

Name	Description	Set	Description
PerfmonAgent - 1116604297654			
ACS/RSVP Service	RSVP or ACS service perfor...		
Browser	The Browser performance ...		
Cache	The Cache performance ob...		
ICMP	The ICMP performance obj...		
IP	The IP performance object ...		
Job Object	Reports the accounting an...		
Job Object Details	% Job object Details shows...		
Memory	The Memory performance o...		
NBT Connection	The NBT Connection perfor...		
Network Interface	The Network Interface perf...		
Intel[R] PRO_100 S Serve			
Bytes Received/sec	Bytes Received/sec is the r	☐	

VisualVM

https://visualvm.dev.java.net/

- « VisualVM is a visual tool that integrates several existing JDK software tools and lightweight memory and CPU profiling capabilities. This tool is designed for both production and development time use and further enhances the capability of monitoring and performance analysis for the Java SE platform.»
- **VisualVM includes the JConsole.**



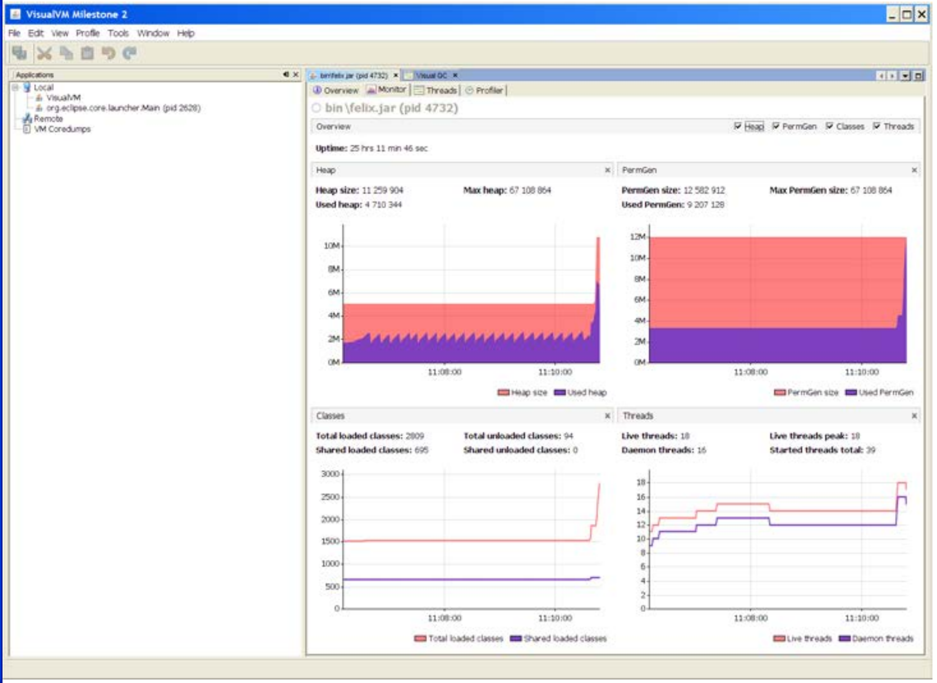
java2d.Java2Demo (pid 3964)

Profiler

Profile: Threads CPU Memory Stop

Status: profiling running (Analyze Performance)

Hot Spots - Method	Self time [%]	Self time	Invocations
sun.java2d.pipe.DrawImage.**blitSurfaceData** (sun.ja…		369 ms (33,5%)	354
sun.java2d.pipe.AlphaColorPipe.**renderPathTile** (Obj…		65.6 ms (5,9%)	26292
sun.java2d.pipe.AlphaPaintPipe.**renderPathTile** (Obj…		59.2 ms (5,4%)	1431
java.awt.GradientPaintContext.**clipFillRaster** (int[], i…		58.4 ms (5,3%)	6650
sun.java2d.pipe.DuctusShapeRenderer.**renderPath** (…		52.6 ms (4,8%)	520
sun.dc.pr.PathFiller.**writeAlpha** (byte[], int, int, int)		50.2 ms (4,6%)	3371
java.util.Arrays.**binarySearch0** (Object[], int, int, O…		27.0 ms (2,5%)	7872
sun.java2d.loops.GraphicsPrimitiveMgr$2.**compare** (…		24.7 ms (2,2%)	78474
sun.java2d.pipe.DuctusRenderer.**createShapeRasteri**		17.3 ms (1,6%)	520
java.lang.Math.**min** (int, int)		16.7 ms (1,5%)	59660
java.awt.GradientPaintContext.**getRaster** (int, int, in…		8.76 ms (0,8%)	6650
sun.dc.pr.Rasterizer.**getTileState** ()		7.32 ms (0,7%)	24184
java.awt.TexturePaintContext.**getRaster** (int, int, int…		7.29 ms (0,7%)	1353
sun.java2d.pipe.TextRenderer.**drawGlyphList** (sun.j…		7.22 ms (0,7%)	281
java.awt.geom.Path2D$Double$TxIterator.**currentSegr**		6.96 ms (0,6%)	7749
sun.java2d.pipe.AlphaColorPipe.**needTile** (Object, int…		6.38 ms (0,6%)	26292



Uptime: 25 hrs 11 min 46 sec

Heap size: 11 259 904 **Max heap:** 67 108 864
Used heap: 4 710 344

PermGen size: 12 582 912 **Max PermGen size:** 67 108 864
Used PermGen: 9 207 128

Total loaded classes: 2809 **Total unloaded classes:** 94
Shared loaded classes: 695 **Shared unloaded classes:** 0

Live threads: 18 **Live threads peak:** 18
Daemon threads: 16 **Started threads total:** 39

http://weblogs.java.net/blog/mandychung/archive/VisualVM-BOF-2007.pdf

Performance Optimizations

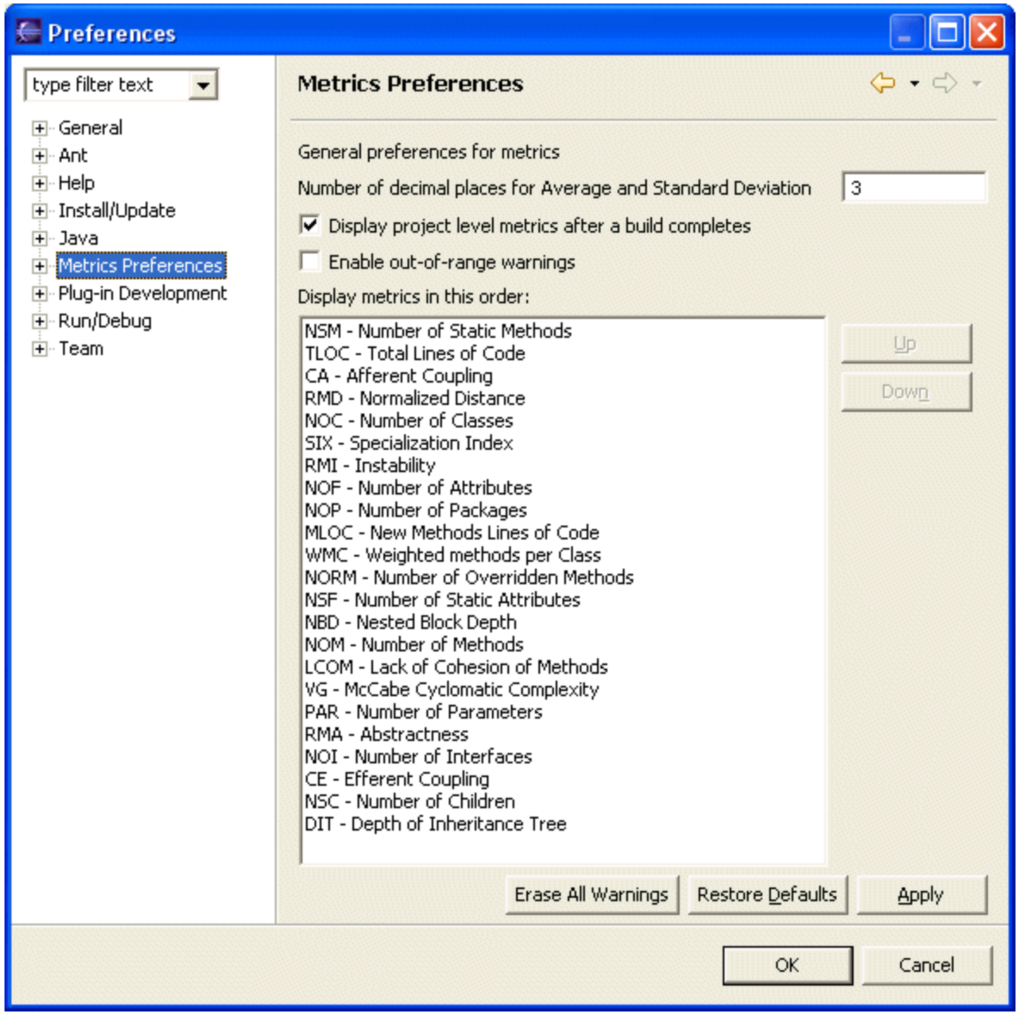
- Java’s Script Engine
  - Jython (jython.sourceforge.net), …
- Bytecode interpreter
- Native compiler (static)
  - .class to .c to .s to .exe
- On-the-fly compiler (dynamic)
  - Compilation JIT (Just-In-Time) de Symantec
- HotSpot™ Optimizer
  - garbage collector
  - « method inlining »
    - with load-time verification (dynamic) of class bytecode
- Benchmark de JVM

Code Quality Metrics

- Metrics on project source code to evaluate its quality (maintenance,reverse-engineering,evolution …)
  - and how good the development team is :-)
- Metrics
  - LOC, LOCC, McCabe Cyclomatic Complexity, …
- Lectures
  - Brian Henderson-Sellers , "Object-Oriented Metrics, measures of Complexity", Ed Prentice Hall, 1996
  - Robert Martin, "OO Design Quality Metrics, An Analysis of Dependencies", 1994, http://www.objectmentor.com/resources/articles/oodmetrc.pdf
  - Chidamber and Kemerer, A Metrics Suite for Object Oriented Design, http://www.pitt.edu/~ckemerer/CK%20research%20papers/MetricForOOD_ChidamberKemerer94.pdf
  - Mariano Ceccato and Paolo Tonella, Measuring the Effects of Software Aspectization, http://homepages.cwi.nl/~tourwe/ware/ceccato.pdf
  - Robert Martin, "Agile Software Development, Principles, Patterns and Practices", Prentice Hall, 1st edition, 2002, ISBN: 978-0135974445

Code Quality Metrics

- Example: Metrics (Tache ANT + Eclipse plugin)



Metrics Preferences

General preferences for metrics

Number of decimal places for Average and Standard Deviation: 3

[✓] Display project level metrics after a build completes

[ ] Enable out-of-range warnings

Display metrics in this order:

NSM - Number of Static Methods
TLOC - Total Lines of Code
CA - Afferent Coupling
RMD - Normalized Distance
NOC - Number of Classes
SIX - Specialization Index
RMI - Instability
NOF - Number of Attributes
NOP - Number of Packages
MLOC - New Methods Lines of Code
WMC - Weighted methods per Class
NORM - Number of Overridden Methods
NSF - Number of Static Attributes
NBD - Nested Block Depth
NOM - Number of Methods
LCOM - Lack of Cohesion of Methods
VG - McCabe Cyclomatic Complexity
PAR - Number of Parameters
RMA - Abstractness
NOI - Number of Interfaces
CE - Efferent Coupling
NSC - Number of Children
DIT - Depth of Inheritance Tree

Code Quality Metrics

- Others (standalone or as IDE plugins)
  - http://metrics.sourceforge.net/
  - http://qjpro.sourceforge.net/
  - http://www.geocities.com/sivaram_subr/index.htm
  - …

Refactoring

What’s Code Refactoring?

“A series of *small* steps, each of which changes the program’s *internal structure* without changing its *external behavior* “

Martin Fowler

Example

Which code segment is easier to read?

Sample 1:

```
if (markT>=0 && markT<=25 && markL>=0 && markL<=25){
      float markAvg = (markT + markL)/2;
      System.out.println("Your mark: " + markAvg);
}
```

Sample 2:

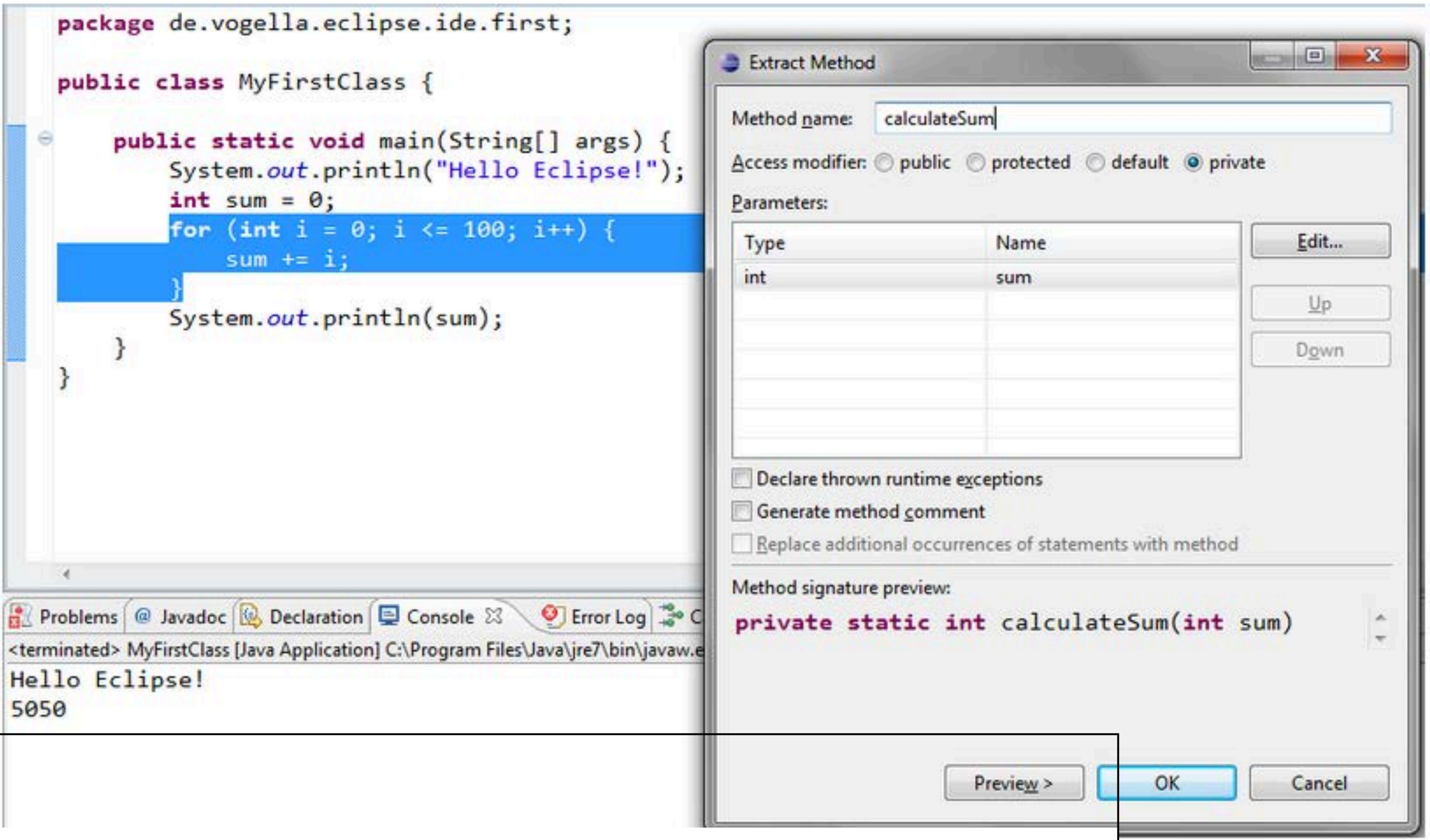
```
if (isValid(markT) && isValid(markL)){
      float markAvg = (markT + markL)/2;
      System.out.println("Your mark: " + mark);
}
```

Why do we Refactor?

- Improves the design of our software
  - Apply design pattern / remove anti pattern
- Minimizes technical debt
- Keep development at speed
- To make the software easier to understand
- To help find bugs
- To “Fix broken windows”

How do we Refactor?

- Manual Refactoring
  - Code Smells
- Automated/Assisted Refactoring
  - Refactoring by hand is time consuming and prone to error
  - Tools (IDE)
- In either case, **test your changes**



```
package de.vogella.eclipse.ide.first;

public class MyFirstClass {

    public static void main(String[] args) {
        System.out.println("Hello Eclipse!");
        int sum = 0;
        for (int i = 0; i <= 100; i++) {
            sum += i;
        }
        System.out.println(sum);
    }
}
```

```
<terminated> MyFirstClass [Java Application] C:\Program Files\Java\jre7\bin\javaw.e
Hello Eclipse!
5050
```

Extract Method

Method name: calculateSum

Access modifier: ( ) public ( ) protected ( ) default [✓ private]

Parameters:

Type	Name
int	sum

[ ] Declare thrown runtime exceptions

[ ] Generate method comment

[ ] Replace additional occurrences of statements with method

Method signature preview:

```
private static int calculateSum(int sum)
```

```
package de.vogella.eclipse.ide.first;

public class MyFirstClass {

  public static void main(String[] args) {
    System.out.println("Hello Eclipse!");
    int sum = 0;
    sum = calculateSum(sum);
    System.out.println(sum);
  }

  private static int calculateSum(int sum) {
    for (int i = 0; i <= 100; i++) {
      sum += i;
    }
    return sum;
  }
}
```

Typical refactoring patterns

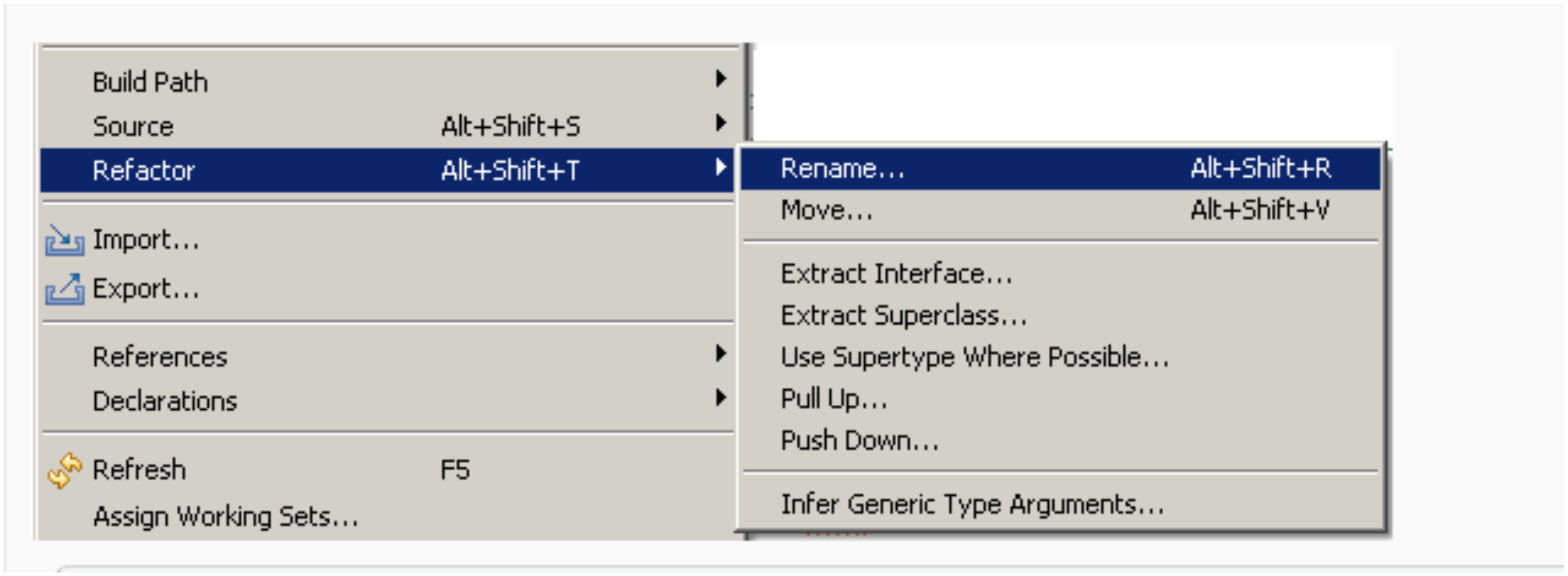
– Rename variable / class / method / member

– Extract method

– Extract constant

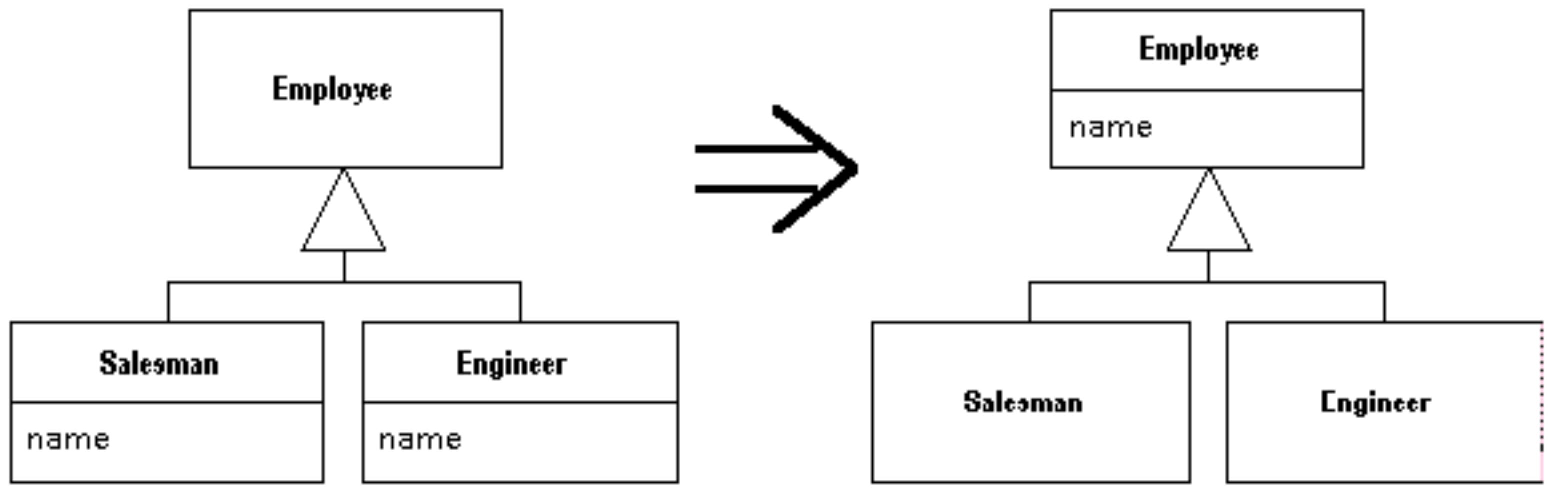
– Extract interface

– Encapsulate field



Two subclasses have the same field.

Move the field to the superclass.



You have a complicated expression.

Put the result of the expression, or parts of the expression, in a temporary variable with a name that explains the purpose.

```
if ( (platform.toUpperCase().indexOf("MAC") > -1) &&
      (browser.toUpperCase().indexOf("IE") > -1) &&
       wasInitialized() && resize > 0 )
 {
    // do something
 }
```

```
        final boolean isMacOs     = platform.toUpperCase().indexOf("MAC") > -1;
        final boolean isIEBrowser = browser.toUpperCase().indexOf("IE")  > -1;
        final boolean wasResized  = resize > 0;

        if (isMacOs && isIEBrowser && wasInitialized() && wasResized)
        {
                // do something
        }
```

Build automation

Compilation chain

- Tools
  - make, gmake, nmake (Win),
  - Apache ANT, Apache MAVEN, Freshmeat 7Bee ...
- To automate:
  - pre-compilation, obfuscation, verification
  - generation of .class and .jar
    - normal, tracing, debug, …
  - documentation generation
  - « stubs » generation (rmic, idl2java, javacard …)
  - test
  - …

Maven

- Goal
  - Separation of concerns applied to project build
    - Compilation, code generation, unit testing, documentation, …
  - Handle project dependencies with versions (artifacts)
- Project object model (POM)
  - abstract description of the project
  - Property inheritance from POM parents
- Tools (called plugin)
  - To compile, generate documentation, automate test …

- Note: more and more useful !

Maven Motivation

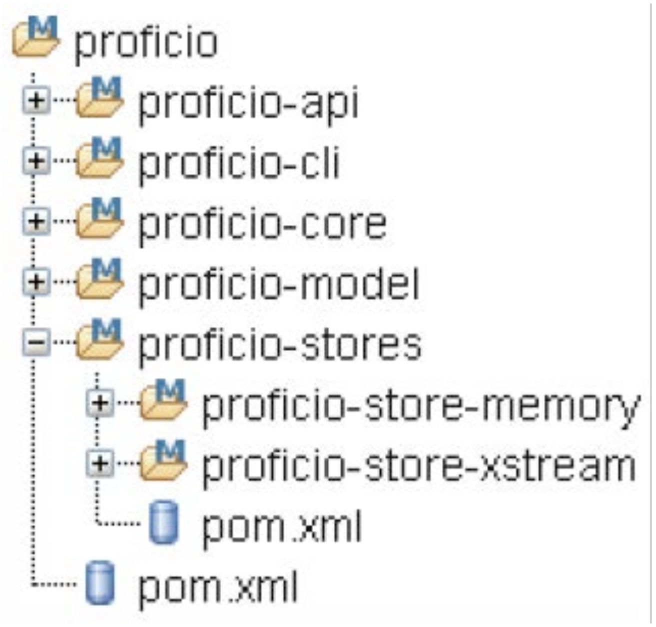
- Abstract project model (POM)
  - Object oriented, inheritance
  - Separation of concerns
- Default lifecycle
  - Default state (goals) sequence
    - plugins depend on states
- Give a project « standard » structure
  - Standard naming conventions
  - Standard lifecycle
- Automatic handling of dependencies between projects
  - Chargement des MAJ
- Project repositories
  - public or private, local or remotes
  - caching and proxy
- Extensible via external plugins

Maven plugins

- Core
  - clean, compiler, deploy, install, resources, site, surefire, verifier
- Packaging
  - ear, ejb, jar, rar, war, bundle (OSGi)
- Reporting
  - changelog, changes, checkstyle, clover, doap, docck, javadoc, jxr, pmd, project-info-reports, surefire-report
- Tools
  - ant, antrun, archetype, assembly, dependency, enforcer, gpg, help, invoker, one (interop Maven 1), patch, plugin, release, remote-resource, repository, scm
- IDEs
  - eclipse, netbeans, idea
- Others
  - exec, jdepend, castor, cargo, jetty, native, sql, taglist, javacc, obr …

Project hierarchies

- Motivations
  - Organize development in sub-projects
    - With N levels (N>=1)
- Technique
  - Create a super POM (type *pom*) for each nesting level
    - Place shared plugins/goals at the same level
  - Subprojects (called modules) inherit from this super pom
- Example



```
proficio
├─ proficio-api
├─ proficio-cli
├─ proficio-core
├─ proficio-model
├─ proficio-stores
│  ├─ proficio-store-memory
│  ├─ proficio-store-xstream
│  └─ pom.xml
└─ pom.xml
```

- Command
  - mvn clean install
    - Global construction

Maven plugin for JAVA IDE

- Maven plugins exists for
  - Eclipse
  - Intellij
  - NetBeans
  - …

Versioning

Versioning of source code

- Collaborative software engineering

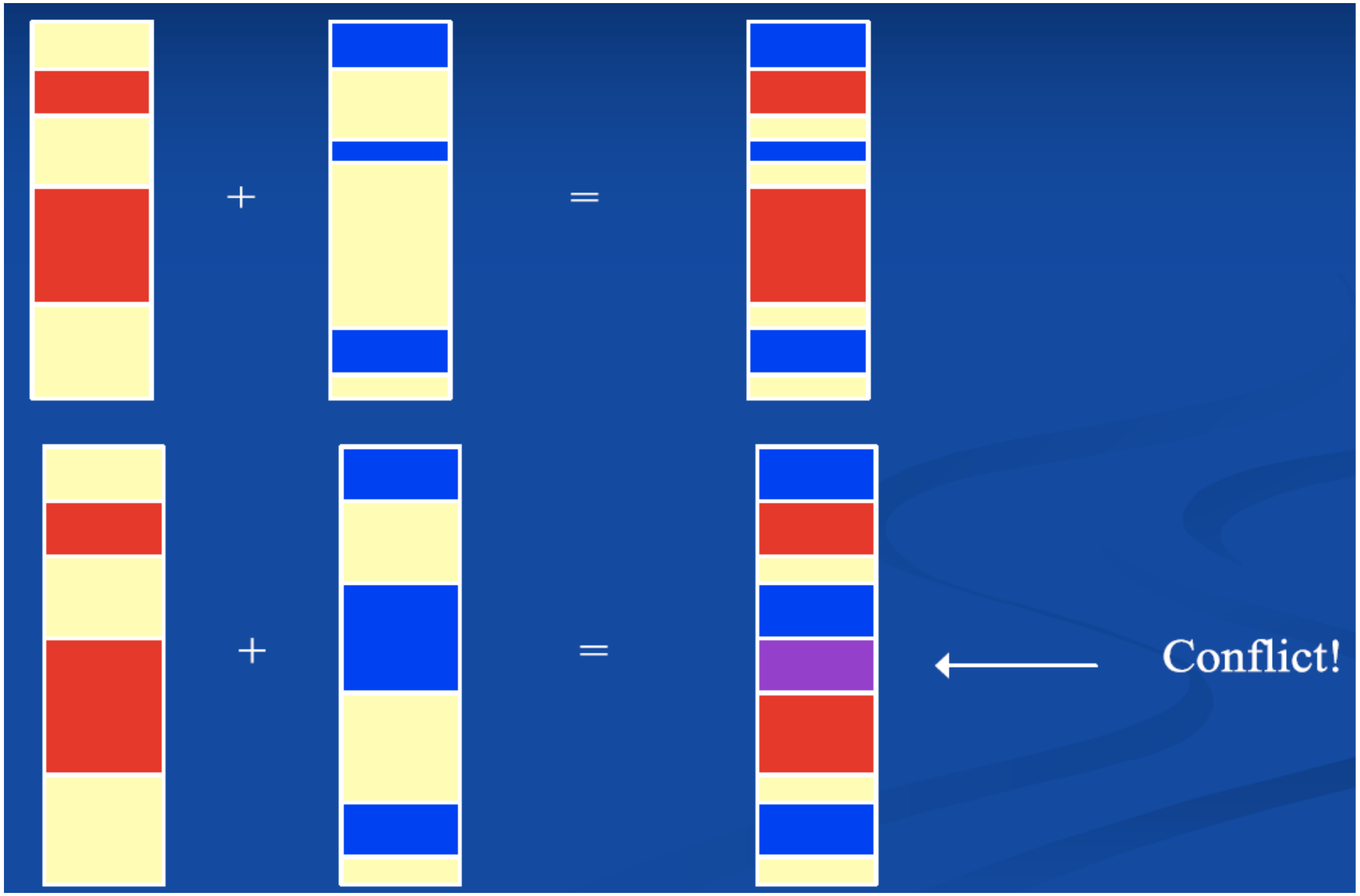
- To master
  - software development by very large developer teams
  - parallel implementations (experiments, vendors)
- Goals
  - Increase productivity of developers and software robustness
  - Low-down development costs

- Manage software system configuration
  - to control software system’s evolution
  - evolution tracking (time-machine)
  - issue and bug tracking

Versioning : What for?

- History of versions
  - back to an older version in case of errors
- Alternative versions (branching)
  - different design/implementations (maybe experimentals) for the same module
- Collaborative access by many developers
  - audit modification history
    - how many commits by X ?
    - when most of the commits are done?
    - ...

Concurrency management



Concurrency control

- Doing nothing!
- Lock-Modify-Unlock (Pessimistic)
  - SCCS, RCS
  - Decrease productivity
- Copy-Modify-Merge (Optimistic)
  - Conflicts resolution when concurrent modifications (which are actually rare)
    - Merge, Selection, …
  - CVS, SVN, Git : Client level resolution
- Policy-based
  - Merging and validation process for each code contribution

Concept of Version

- Trunk
  - main development
- Branches
  - Alternatives to trunk
    - Different design/implementation (experimental), vendor-specific
- Revisions
  - Sequence of versions
- Tags
  - Symbolic references to revisions (Tiger, LongHorn, …)
    - Represent a public release (R), a milestone (M)
- Branch merging

Tools

- Pioneers
  - SCCS, RCS, PVCS
- Current alternatives
  - CVS
  - SubVersion
  - Git
  - MS Visual SourceSafe
  - ChangeMan (Serena)
  - AllFusion Harvest (CA)
  - ClearCase (IBM Rational)
  - Perforce
  - CM Synergy (Telelogic)
  - Source Integrity (MKS)
  - PVCS (Merant)
  - TeamCode (Interwoven)
  - Surround CM (Seapine)
- Web-Oriented protocols
  - WebDAV/DeltaV

Git

- version control system
  - designed to handle very large projects with speed and efficiency
  - mainly for various open source projects, most notably the Linux kernel.
- https://git-scm.com (http://git.or.cz)

Tools to use Git

- SmartGit
- TortoiseGit
- CyberDuck
- Etc.

- GitHub => provide the whole forge with a GIT installed
  - Free for open-source project
  - Rich client: https://desktop.github.com

Centralized Version Control

- Traditional version control system
  - Server with database
  - Clients have a working version
- Examples
  - CVS
  - Subversion
  - Visual Source Safe
- Challenges
  - Multi-developer conflicts
  - Client/server communication

Distributed Version Control

- Authoritative server by convention only
- Every working checkout is a repository
- Get version control even when detached
- Backups are trivial

- Other distributed systems include
  - Mercurial
  - BitKeeper
  - Darcs
  - Bazaar

Git Advantages

- Resilience
  - No one repository has more data than any other
- Speed
  - Very fast operations compared to other VCS (I’m looking at you CVS and Subversion)
- Space
  - Compression can be done across repository not just per file
  - Minimizes local size as well as push/pull data transfers
- Simplicity
  - Object model is very simple
- Large userbase with robust tools

Git Architecture

- Index
  - Stores information about current working directory and changes made to it

- Object Database
  - Blobs (files)
    - Stored in .git/objects
    - Indexed by unique hash
    - All files are stored as blobs
  - Trees (directories)
  - Commits
    - One object for every commit
    - Contains hash of parent, name of author, time of commit, and hash of the current tree
  - Tags

Some Commands

- Getting a Repository
  - git init
  - git clone
- Commits
  - git add
  - git commit
- Get changes with
  - git fetch (fetches and merges)
  - git pull
- Propagate changes with
  - git push

Documenting,
Testing,
Design Patterns, bad smells
Refactoring,
Debugging, monitoring, logging

#1 What is the link?

- Documenting
  - Understanding (readability, maintainability)
- Refactoring
  - Improving the design (readability, maintainability, extensibility)
- The activity of documenting can somehow be replaced/automated by the activity of refactoring
  - if the code and architecture is comprehensible by itself

refactoring.com

Documentation and Refactoring

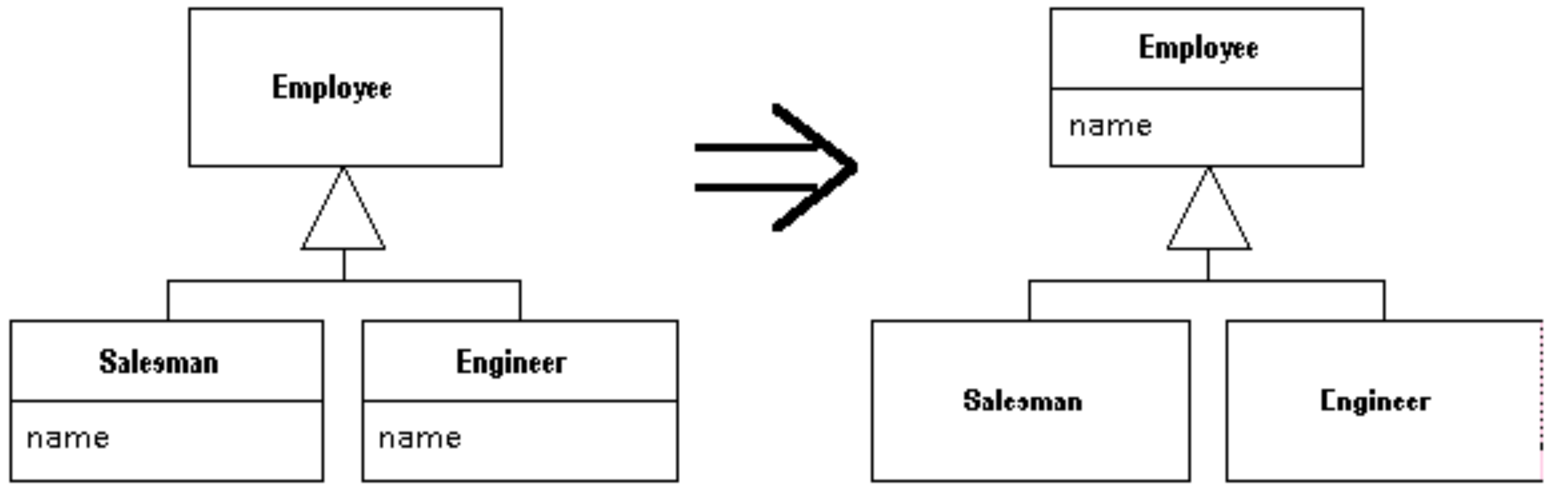
```
if ( (platform.toUpperCase().indexOf("MAC") > -1) &&  // platform is MacOS
      (browser.toUpperCase().indexOf("IE") > -1) && // brower is IE
       wasInitialized() && resize > 0 )
 {
   // do something
 }
```

```
final boolean isMacOs     = platform.toUpperCase().indexOf("MAC") > -1;
final boolean isIEBrowser = browser.toUpperCase().indexOf("IE")  > -1;
final boolean wasResized  = resize > 0;

if (isMacOs && isIEBrowser && wasInitialized() && wasResized)
{
        // do something
}
```

#2 What is the link?
Design patterns: there are refactorings

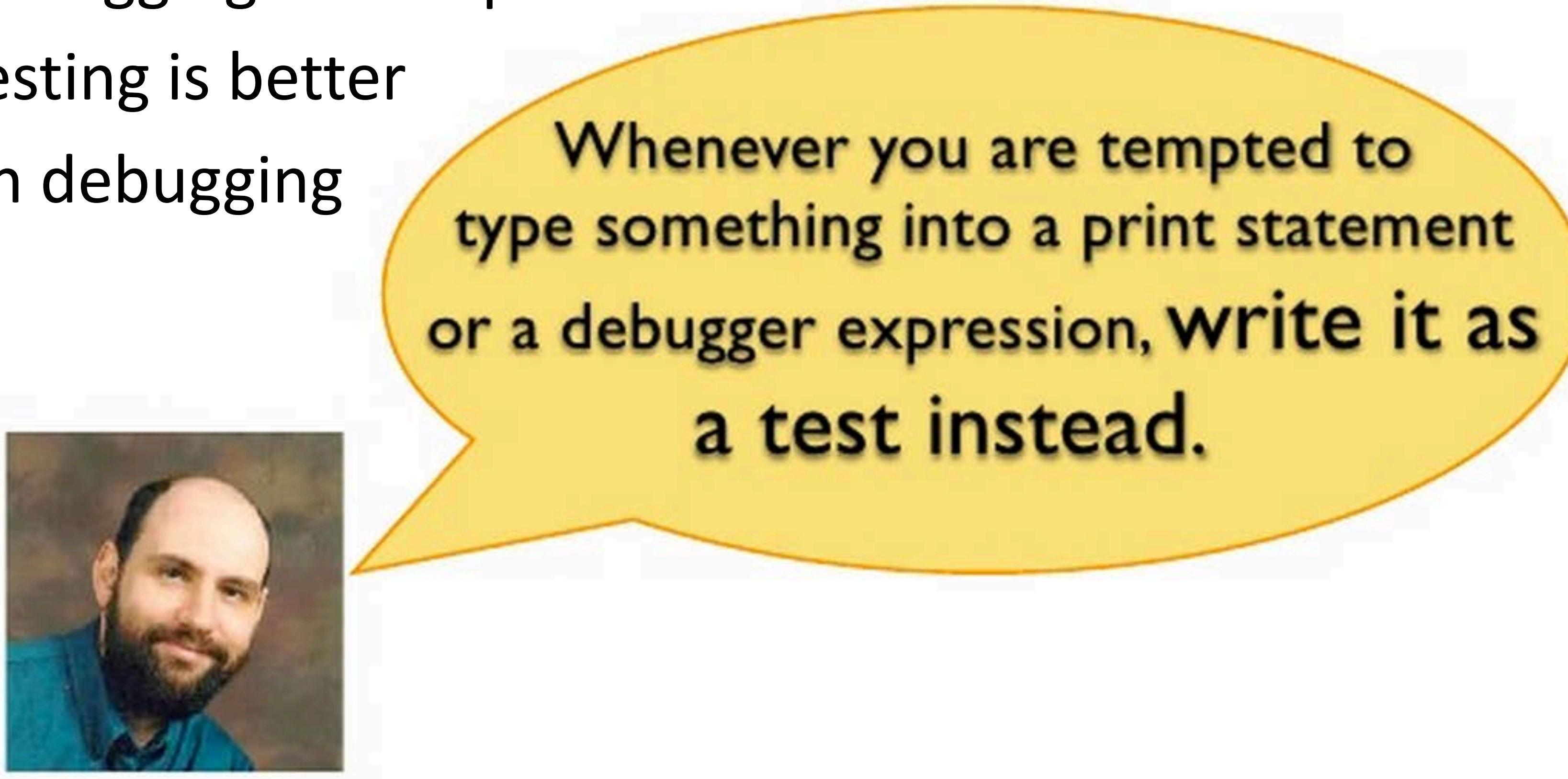
Two subclasses have the same field.

Move the field to the superclass.

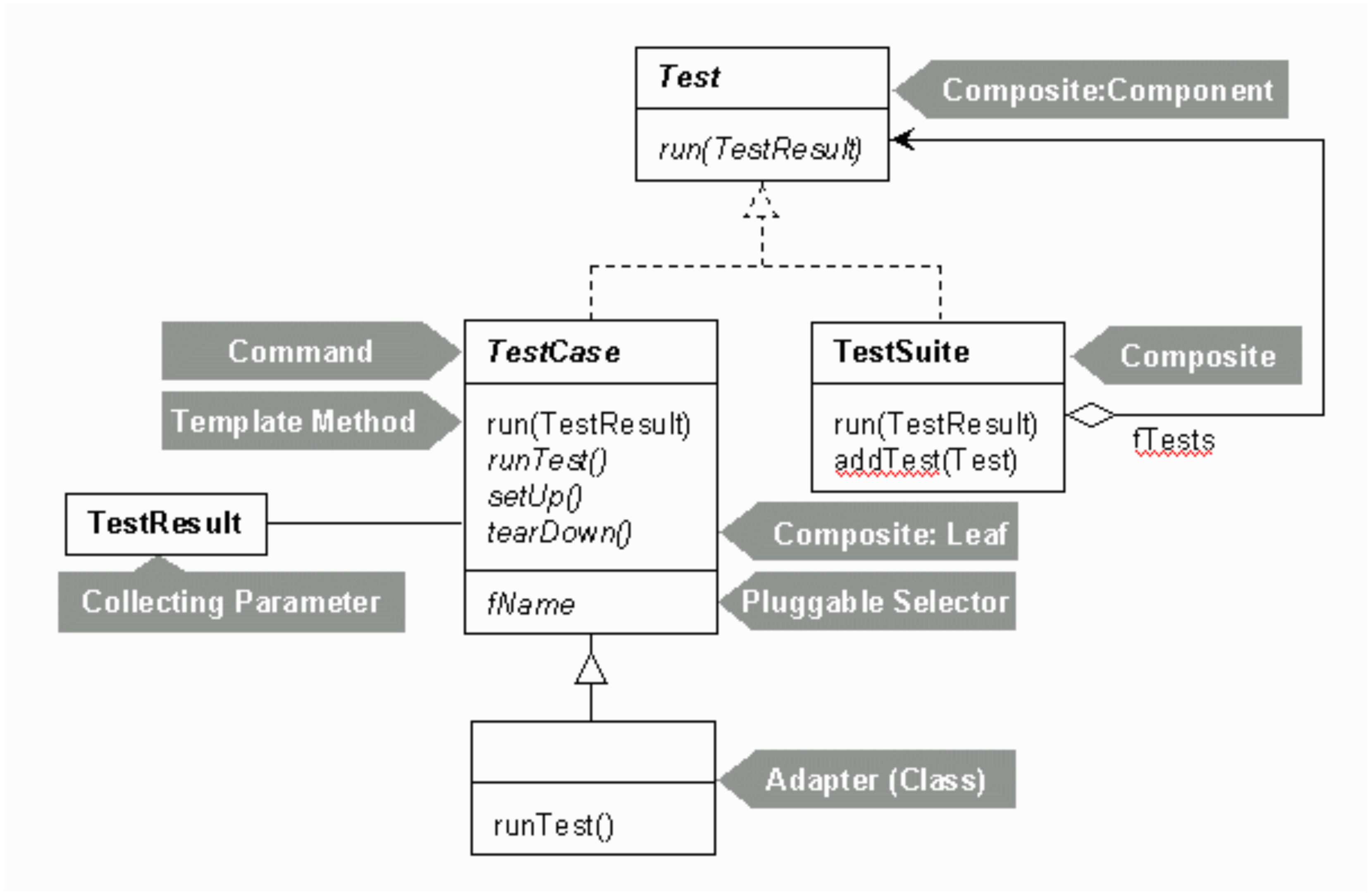


#3 What is the link?

- Testing: “the activity of finding out whether a piece of code produces the intended behavior”
  - Debugging can help
  - Testing is better than debugging


Whenever you are tempted to type something into a print statement or a debugger expression, write it as a test instead.

JUnit and… Design patterns



Worth reading!

http://junit.sourceforge.net/doc/cookstour/cookstour.htm

What is the link?

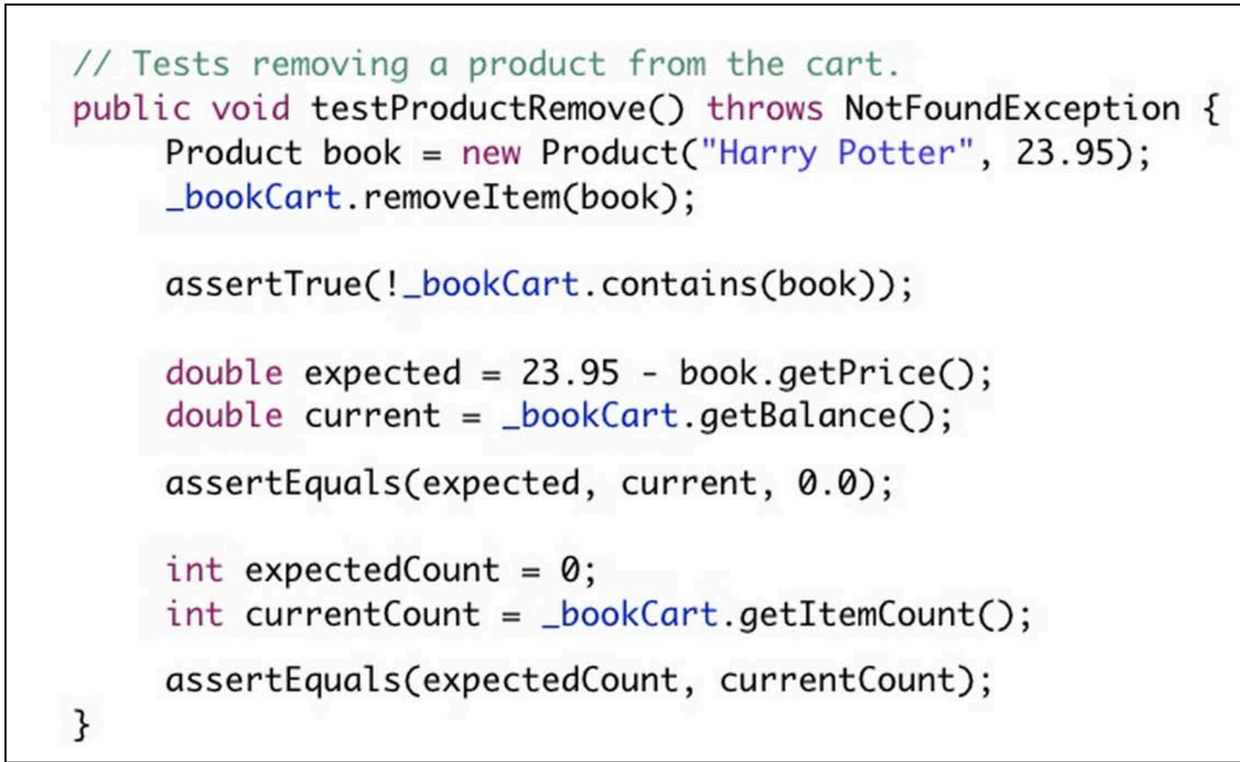
- Testability
  - degree to which a system or component **facilitates the establishment** of test criteria and the performance of tests to determine whether those criteria have been met.”
  - Controllability + Observability
- **Controllability** ability to manipulate the software’s input as well as to place this software into a particular state
- **Observability** deals with the possibility to observe the outputs and state changes
- How to improve Testability?
  - Refactoring, Design patterns

What is the link?

Testing/Refactoring/Design Patterns

- How to improve testability?
- Test-driven Development
  - Write tests first ~ Test-driven design

Let say your first piece of code is… a test



```java
// Tests removing a product from the cart.
public void testProductRemove() throws NotFoundException {
    Product book = new Product("Harry Potter", 23.95);
    _bookCart.removeItem(book);

    assertTrue(!_bookCart.contains(book));

    double expected = 23.95 - book.getPrice();
    double current = _bookCart.getBalance();

    assertEquals(expected, current, 0.0);

    int expectedCount = 0;
    int currentCount = _bookCart.getItemCount();

    assertEquals(expectedCount, currentCount);
}
```

What is the link?

- Testing
- Documenting

- Unit tests are one of the best source of documentation
  - One of the entry point to understand a framework
  - It documents the properties of methods, how objects collaborate, etc.

What is the link?

Documenting

Refactoring

Debugging

Testing

Readability
Understandibility
Maintainability

Design

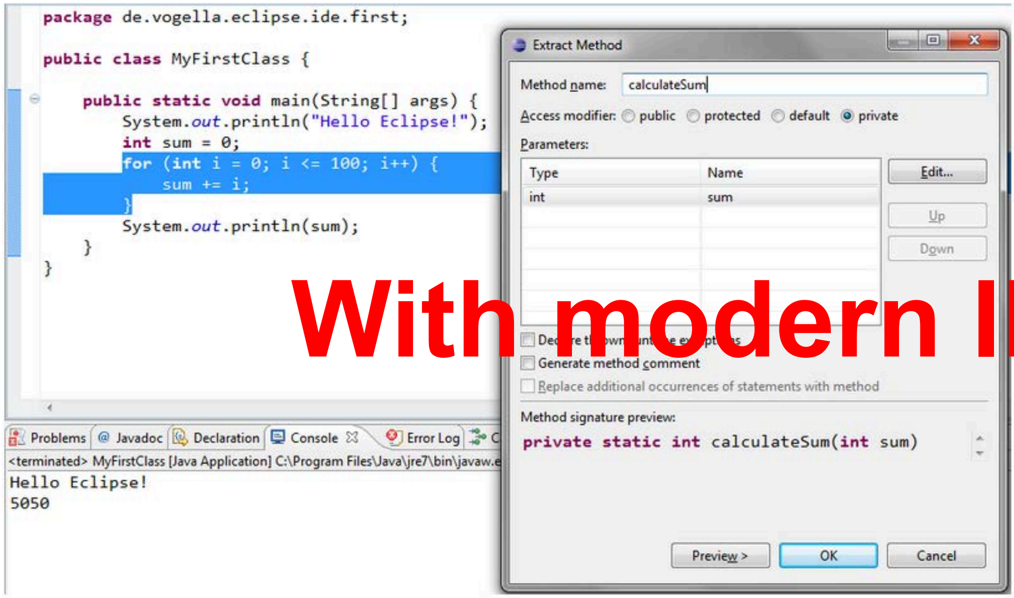
Document, refactor… Execute your tests… Debug.. Write test.. And so on!

Documenting

Debugging

Refactoring

Testing



```
package de.vogella.eclipse.ide.first;

public class MyFirstClass {

    public static void main(String[] args) {
        System.out.println("Hello Eclipse!");
        int sum = 0;
        for (int i = 0; i <= 100; i++) {
            sum += i;
        }
        System.out.println(sum);
    }
}
```

```
<terminated> MyFirstClass [Java Application] C:\Program Files\Java\jre7\bin\javaw.e
Hello Eclipse!
5050
```

Extract Method

Method name: calculateSum

Access modifier: public protected default [✓ private]

Parameters:

Type	Name
int	sum

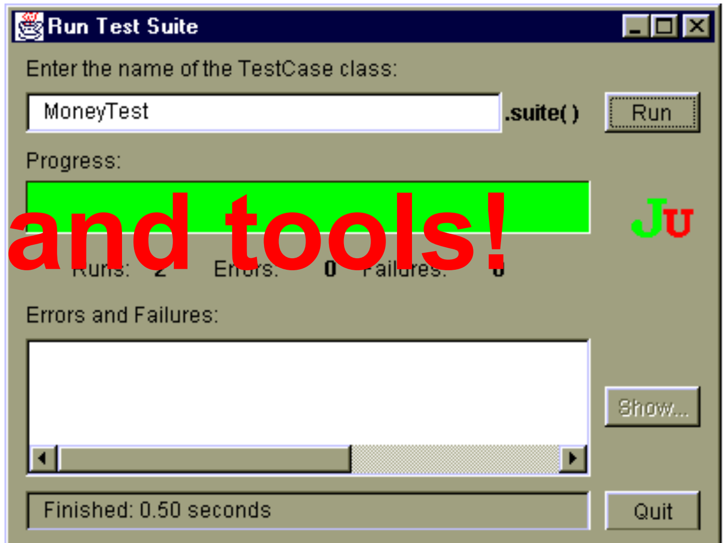
Dec re th wn unt e ex pt s

Generate method comment

Replace additional occurrences of statements with method

Method signature preview:

```
private static int calculateSum(int sum)
```



Run Test Suite

Enter the name of the TestCase class: MoneyTest .suite()

Progress:

Runs: 2 Errors: 0 Failures: 0

Errors and Failures:

Finished: 0.50 seconds

With modern IDE and tools!

INTEGRATION CONTINUE

INTRODUCTION

- ➢ Qu’est ce que l’intégration continue ?
- ➢ Pourquoi automatiser ?
- ➢ Par où commencer ?
- ➢ Le cycle vertueux de l’intégration continue

Définitions

"L'intégration continue est un ensemble de pratiques utilisées en génie logiciel. Elles consistent à vérifier à chaque modification de code source que le résultat des modifications ne produit pas de régression de l'application en cours de développement."

Wikipedia

"Une pratique considérant différemment l'intégration, habituellement connue comme pénible et peu fréquente, pour en faire une tâche simple faisant partie intégrante de l'activité quotidienne d'un développeur."

Documentation
CruiseControl.NET

Qu’est ce que l’intégration continue ?

- Technique puissante permettant dans le cadre du développement d’un logiciel en équipes de:
  - Garder en phase les équipes de dév
  - Limiter risques de dérive
  - Limiter la complexité

- A intervalles réguliers, vous allez construire (build) et tester la dernière version de votre logiciel.

- Parrallèlement, chaque développeur teste et valide (commit) son travail en ajoutant son code dans un lieu de stockage unique.

Pourquoi automatiser ?

- Gagner du temps
  - Vous ne faites pas de taches répétitives

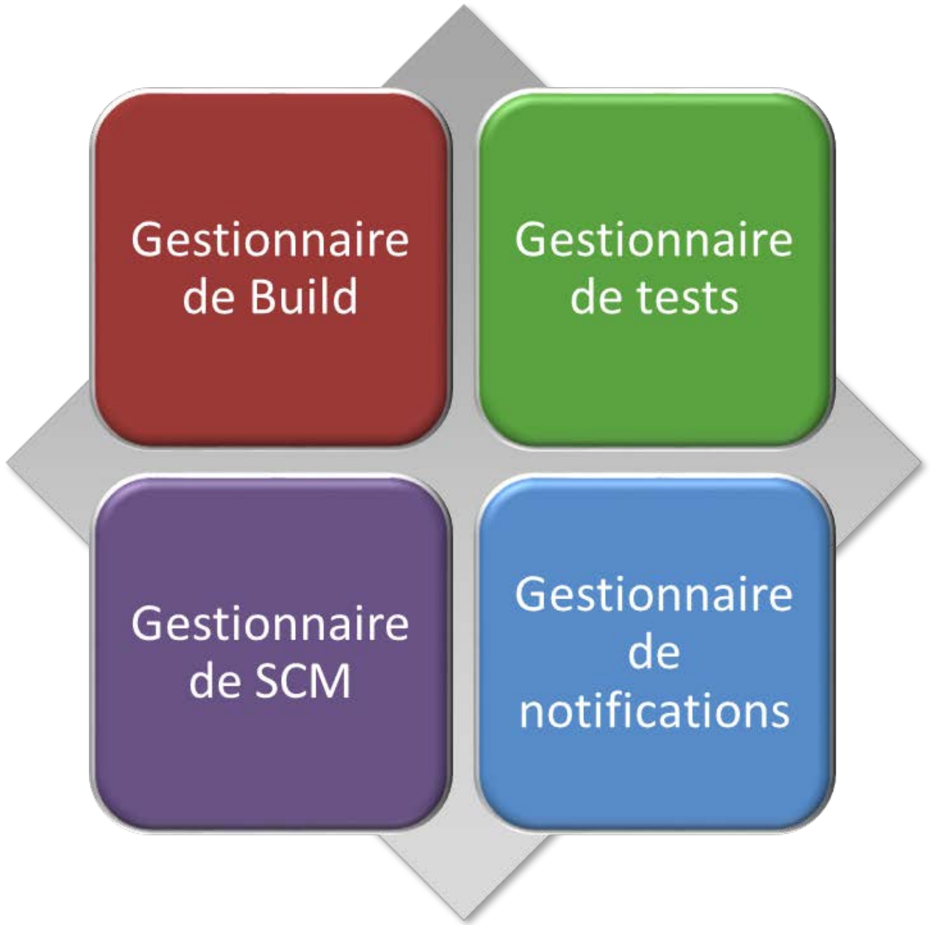
- Gagner en confiance
  - Indépendant de votre efficacité du moment
  - Procédures répétables

- Diminue le besoin de documentation
  - Pour nouveaux entrants projet, utiliser scripts !
  - … et + en analysant le script.

Par où commencer ?

- 1) Outil gestion versions code sources
  - Lieu unique de partage
  - Retour arrières, snapshots, branches…
- 2) Tests automatisés
  - Chaque développeur
- 3) Scripts
  - Coté serveur pour automatiser (Ex : crontab)
- 4)Outils de communication
  - Mail, Tél, Rss…

Architecture d’un logiciel d’intégration

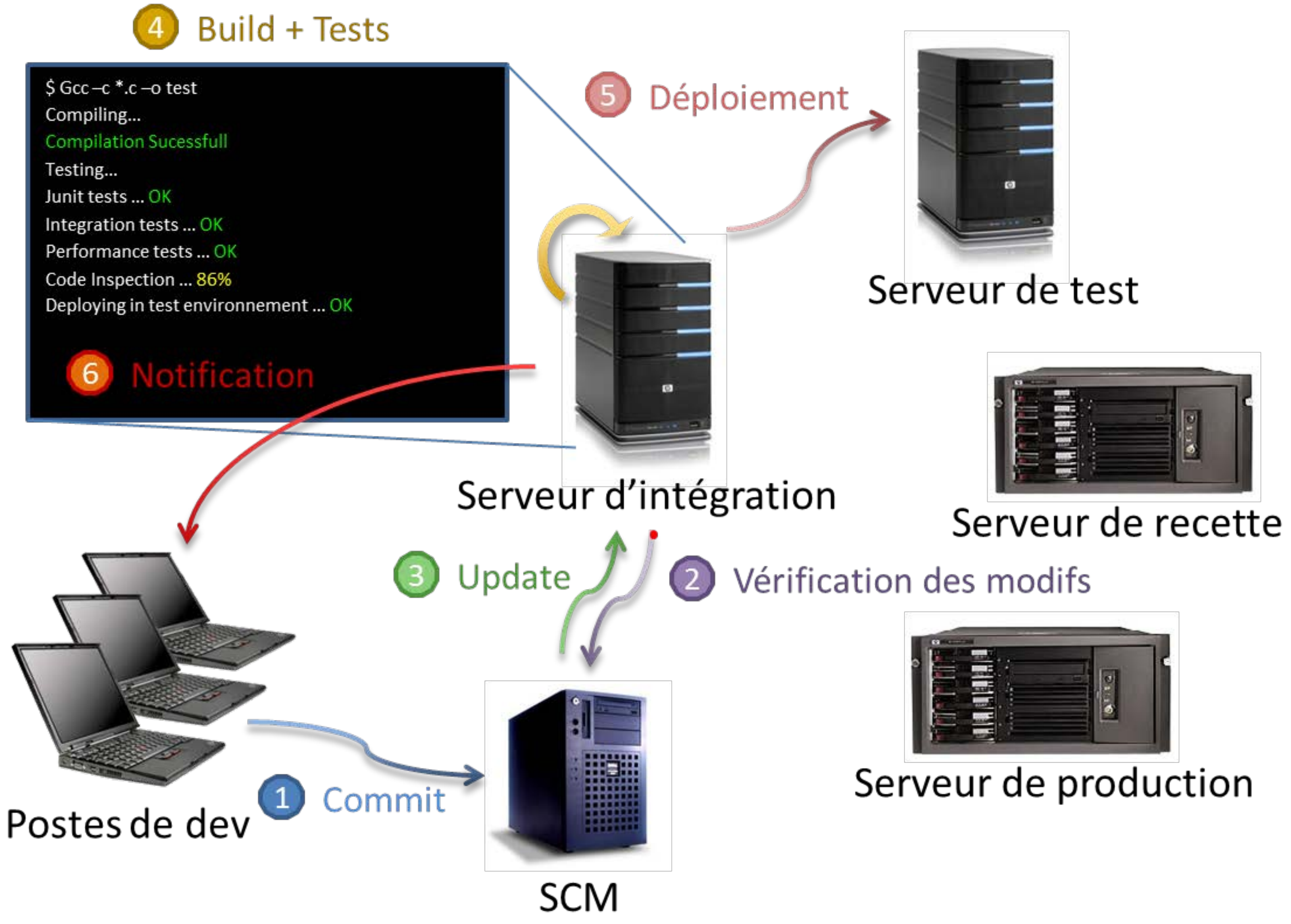


Un fonctionnement actif

- Les développeurs « committent »
- Le serveur d’intégration surveille le serveur SCM (Cron)

Cas d'utilisation

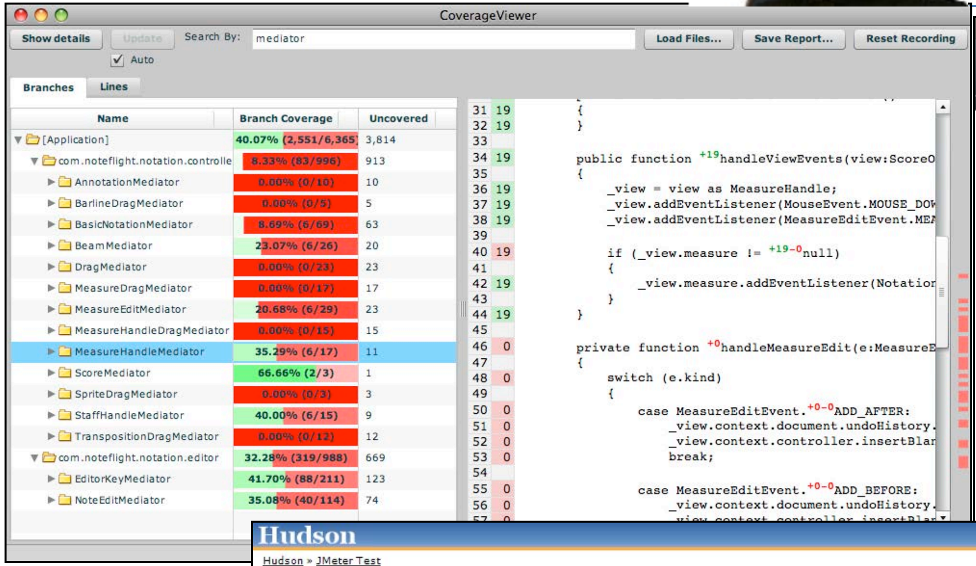
Le développeur soumet une modification



```
$ Gcc –c *.c –o test
Compiling…
Compilation Sucessfull
Testing…
Junit tests … OK
Integration tests … OK
Performance tests … OK
Code Inspection … 86%
Deploying in test environnement … OK
```

Cas d’utilisation

Le chef de projet analyse le reporting

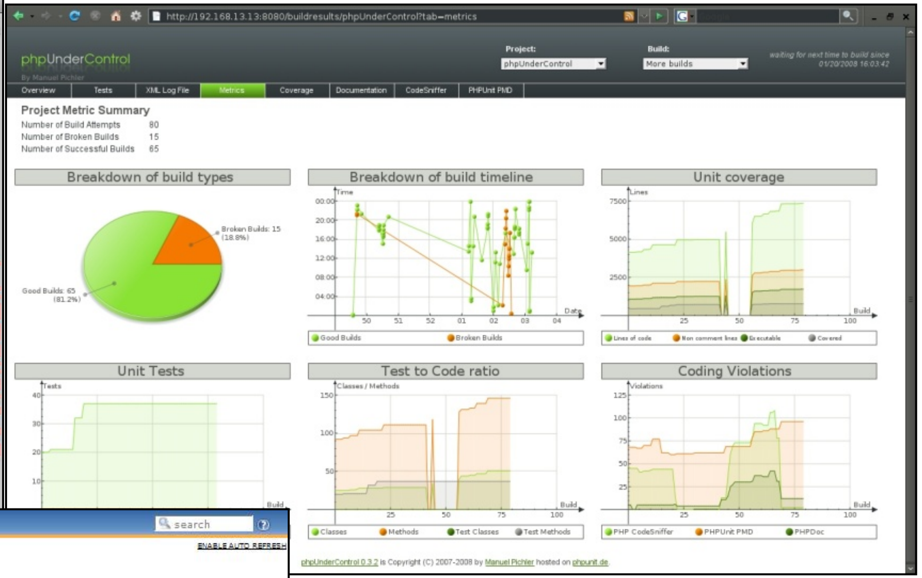


Search By: mediator

✓ Auto

Name	Branch Coverage	Uncovered
[Application]	40.07% (2,551/6,365)	3,814
com.noteflight.notation.controlle	8.33% (83/996)	913
AnnotationMediator	0.00% (0/10)	10
BarlineDragMediator	0.00% (0/5)	5
BasicNotationMediator	8.69% (6/69)	63
BeamMediator	23.07% (6/26)	20
DragMediator	0.00% (0/23)	23
MeasureDragMediator	0.00% (0/17)	17
MeasureEditMediator	20.68% (6/29)	23
MeasureHandleDragMediator	0.00% (0/15)	15
MeasureHandleMediator	35.29% (6/17)	11
ScoreMediator	66.66% (2/3)	1
SpriteDragMediator	0.00% (0/3)	3
StaffHandleMediator	40.00% (6/15)	9
TranspositionDragMediator	0.00% (0/12)	12
com.noteflight.notation.editor	32.28% (319/988)	669
EditorKeyMediator	41.70% (88/211)	123
NoteEditMediator	35.08% (40/114)	74

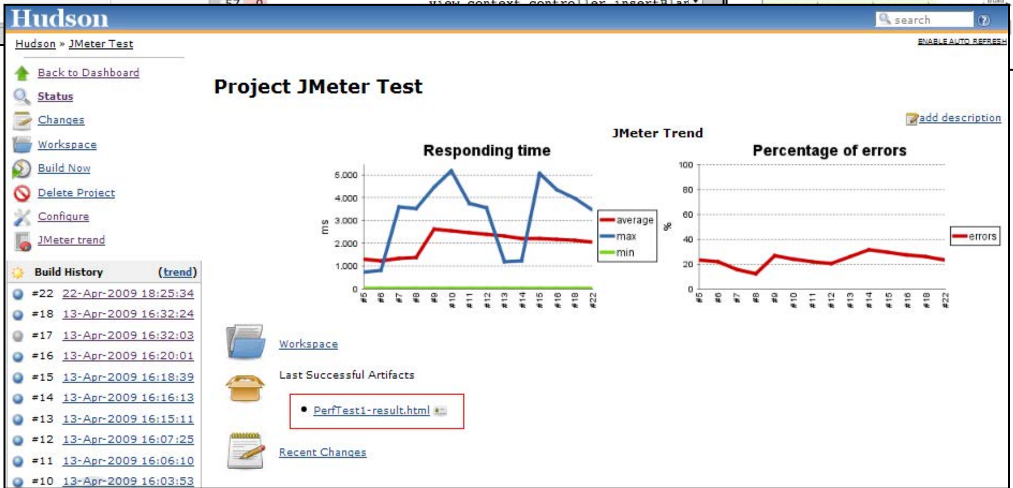
```
        {
        }

        public function +19handleViewEvents(view:ScoreO
        {
            _view = view as MeasureHandle;
            _view.addEventListener(MouseEvent.MOUSE_DOW
            _view.addEventListener(MeasureEditEvent.MEA

            if (_view.measure != +19-0null)
            {
                _view.measure.addEventListener(Notatior
            }
        }

        private function +0handleMeasureEdit(e:MeasureE
        {
            switch (e.kind)
            {
                case MeasureEditEvent.+0-0ADD_AFTER:
                    _view.context.document.undoHistory.
                    _view.context.controller.insertBlar
                    break;

                case MeasureEditEvent.+0-0ADD_BEFORE:
                    _view.context.document.undoHistory.
```



Project Metric Summary

Number of Build Attempts	80
Number of Broken Builds	15
Number of Successful Builds	65

phpUnderControl 0.3.2 is Copyright (C) 2007-2008 by Manuel Pichler hosted on phpunit.de



Hudson » JMeter Test

Project JMeter Test

Build History (trend)

#22	22-Apr-2009 18:25:34
#18	13-Apr-2009 16:32:24
#17	13-Apr-2009 16:32:03
#16	13-Apr-2009 16:20:01
#15	13-Apr-2009 16:18:39
#14	13-Apr-2009 16:16:13
#13	13-Apr-2009 16:15:11
#12	13-Apr-2009 16:07:25
#11	13-Apr-2009 16:06:10
#10	13-Apr-2009 16:03:53

Last Successful Artifacts

- PerfTest1-result.html

Les technologies existantes

- Hudson, jenkins
- CruiseControl / CruiseControl.NET
- Apache Continuum
- QuickBuild (open-source: LuntBuild)
- Et beaucoup d’autres …

Improving Your Productivity

- Continuous integration can help you go faster
  - Detect build breaks sooner
  - Report failing tests more clearly
  - Make progress more visible

Jenkins for Continuous Integration

- Jenkins – open source continuous integration server
- Jenkins (http://jenkins-ci.org/) is
  - Easy to install
  - Easy to use
  - Multi-technology
  - Multi-platform
  - Widely used
  - Extensible
  - Free

Jenkins for a Developer

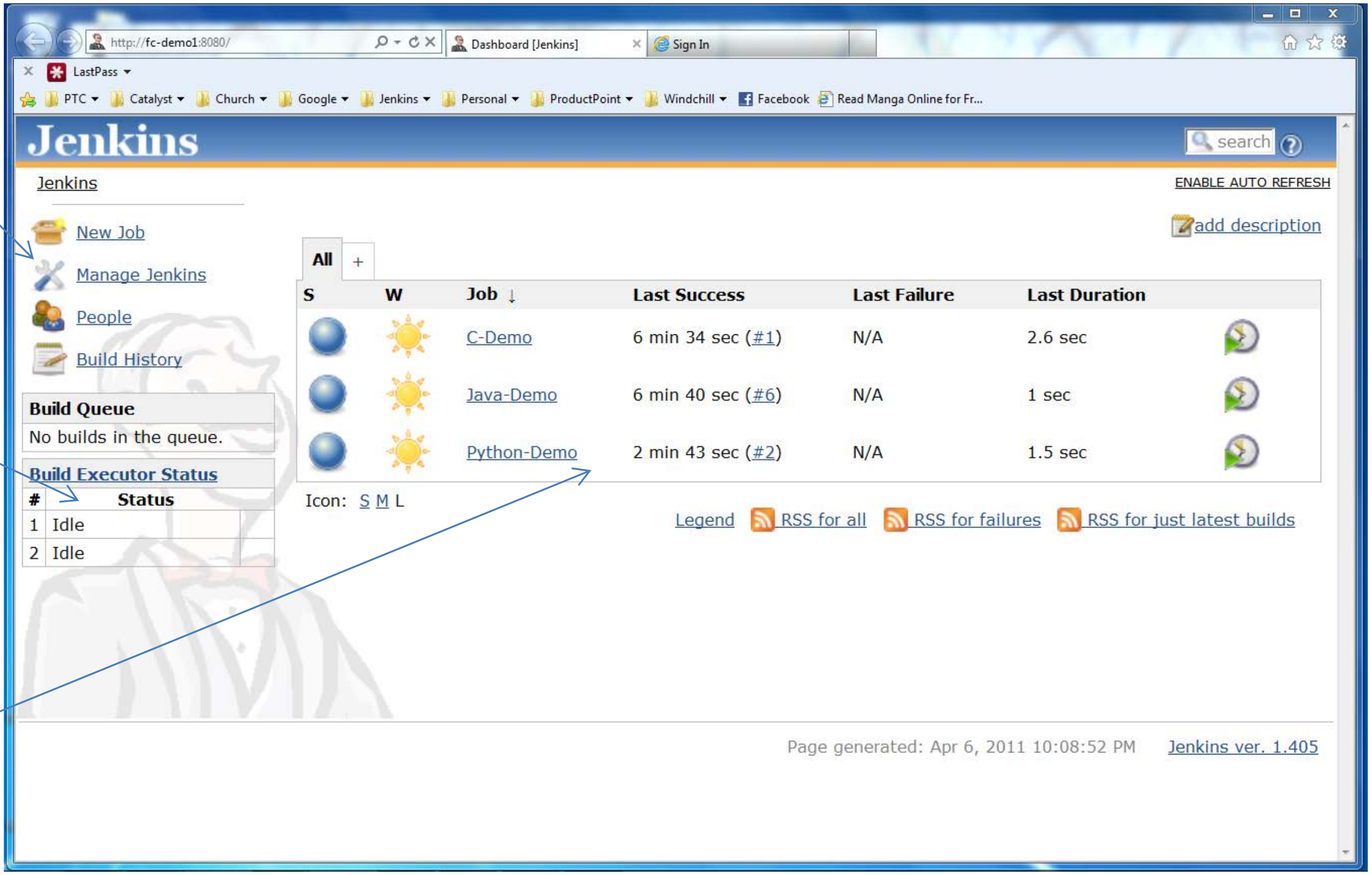
- Easy to install
  - Download one file – jenkins.war
  - Run one command – java –jar jenkins.war
- Easy to use
  - Create a new job – checkout and build a small project
  - Checkin a change – watch it build
  - Create a test – watch it build and run
  - Fix a test – checkin and watch it pass
- Multi-technology
  - Build C, Java, C#, Python, Perl, SQL, etc.
  - Test with Junit, Nunit, MSTest, etc.

Jenkins User Interface

Actions

Nodes

Jobs



Build Queue

No builds in the queue.

Build Executor Status

#	Status
1	Idle
2	Idle

S	W	Job ↓	Last Success	Last Failure	Last Duration
●	☀	C-Demo	6 min 34 sec (#1)	N/A	2.6 sec
●	☀	Java-Demo	6 min 40 sec (#6)	N/A	1 sec
●	☀	Python-Demo	2 min 43 sec (#2)	N/A	1.5 sec

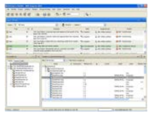
Icon: S M L

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

Page generated: Apr 6, 2011 10:08:52 PM Jenkins ver. 1.405

Jenkins Plugins - SCM

- Version Control Systems
  - Accurev
  - Bazaar
  - BitKeeper
  - ClearCase
  - Darcs
  - Dimensions
  - Git
  - Harvest
  - MKS Integrity
  - PVCS
  - StarTeam
  - Subversion
  - Team Foundation Server
  - Visual SourceSafe



Jenkins Plugins – Build & Test

- Build Tools
  - Ant
  - Maven
  - MSBuild
  - Cmake
  - Gradle
  - Grails
  - Scons
  - Groovy
- Test Frameworks
  - Junit
  - Nunit
  - MSTest
  - Selenium
  - Fitnesse

Jenkins Plugins – Analyzers

- Static Analysis
  - Checkstyle
  - CodeScanner
  - DRY
  - Crap4j
  - Findbugs
  - PMD
  - Fortify
  - Sonar
  - FXCop
- Code Coverage
  - Emma
  - Cobertura
  - Clover
  - GCC/GCOV

Jenkins Plugins – Other Tools

- Notification
  - Twitter
  - Campfire
  - Google Calendar
  - IM
  - IRC
  - Lava Lamp
  - Sounds
  - Speak
- Authorization
  - Active Directory
  - LDAP
- Virtual Machines
  - Amazon EC2
  - VMWare
  - VirtualBox
  - Xen
  - Libvirt