Unit testing with JUnit

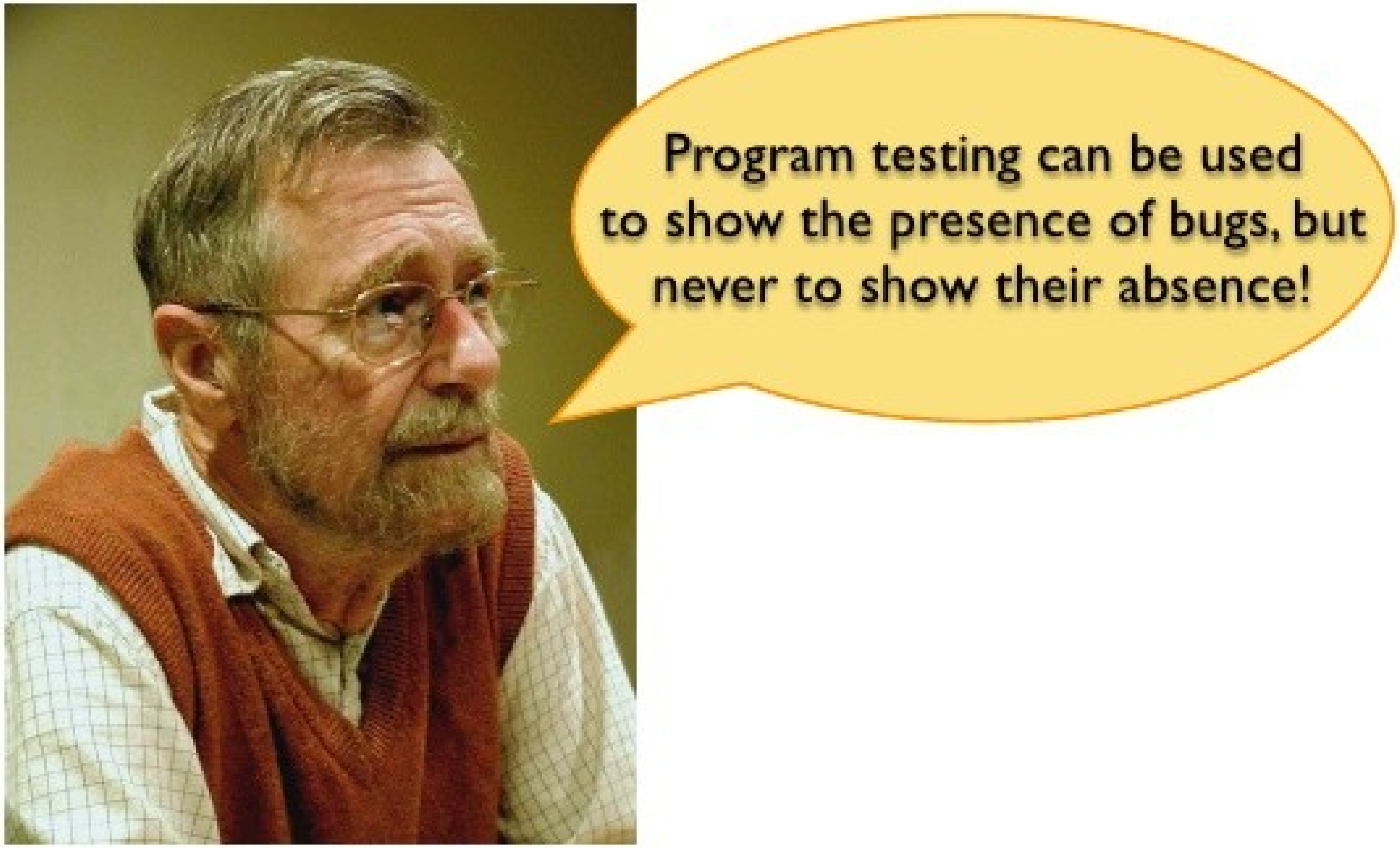
Tom Zimmermann

Testing

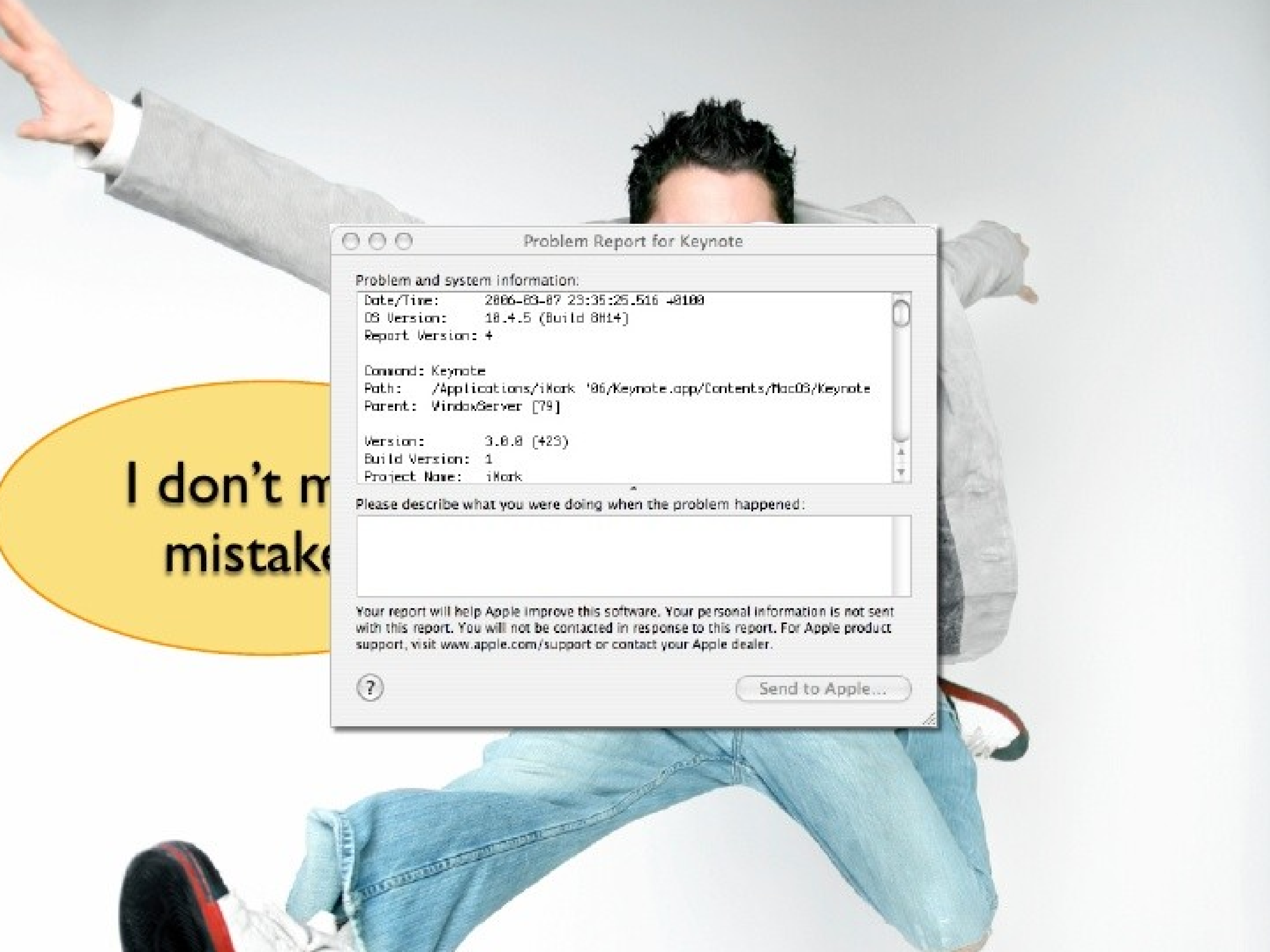
Testing is the activity of finding out whether a piece of code (a method, class, or program) produces the intended behavior.

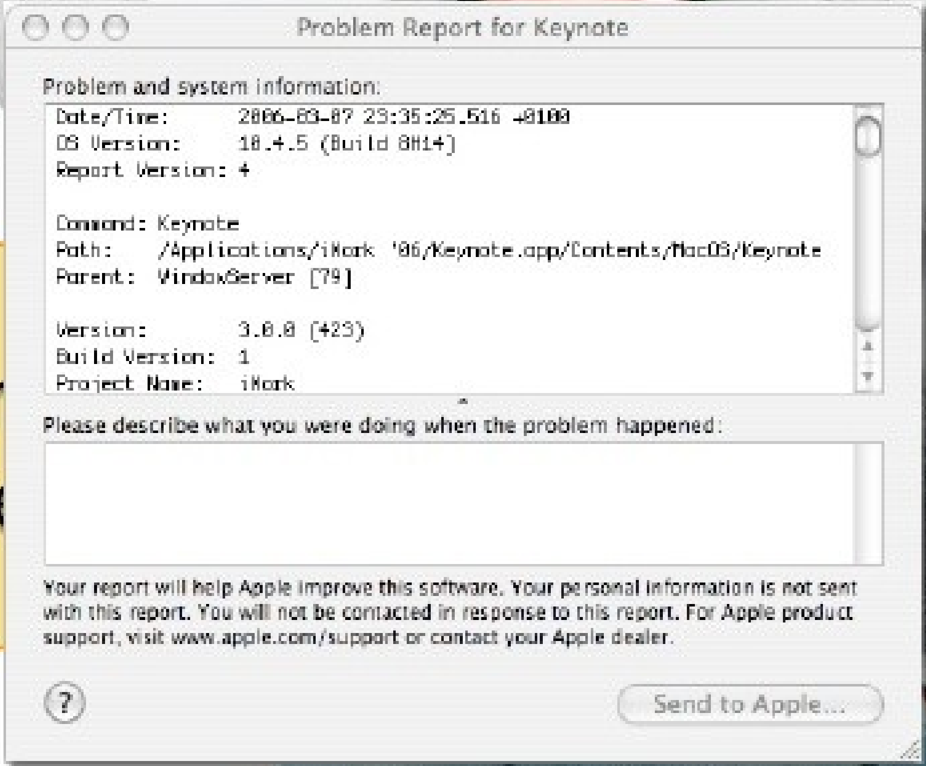
taken from "Objects First with Java"

Edsger Dijkstra


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

I don’t make mistakes!





Problem and system information:

```
Date/Time:      2006-03-07 23:35:25.516 +0100
OS Version:     10.4.5 (Build 8H14)
Report Version: 4

Command: Keynote
Path:    /Applications/iWork '06/Keynote.app/Contents/MacOS/Keynote
Parent:  WindowServer [79]

Version:        3.0.0 (423)
Build Version:  1
Project Name:   iWork
```

Please describe what you were doing when the problem happened:

Your report will help Apple improve this software. Your personal information is not sent with this report. You will not be contacted in response to this report. For Apple product support, visit www.apple.com/support or contact your Apple dealer.

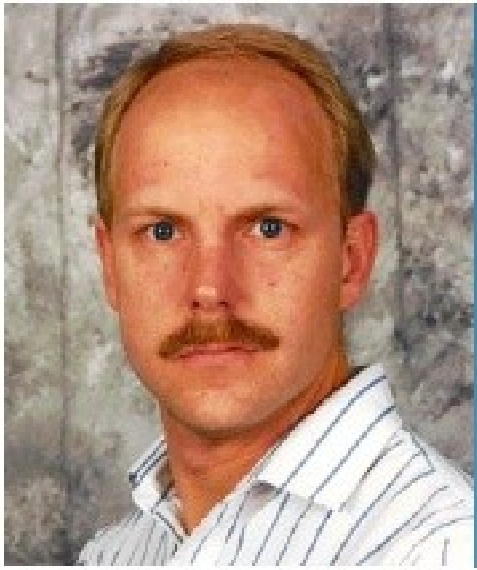
Test phases

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

Integration testing on groups of individual software modules.

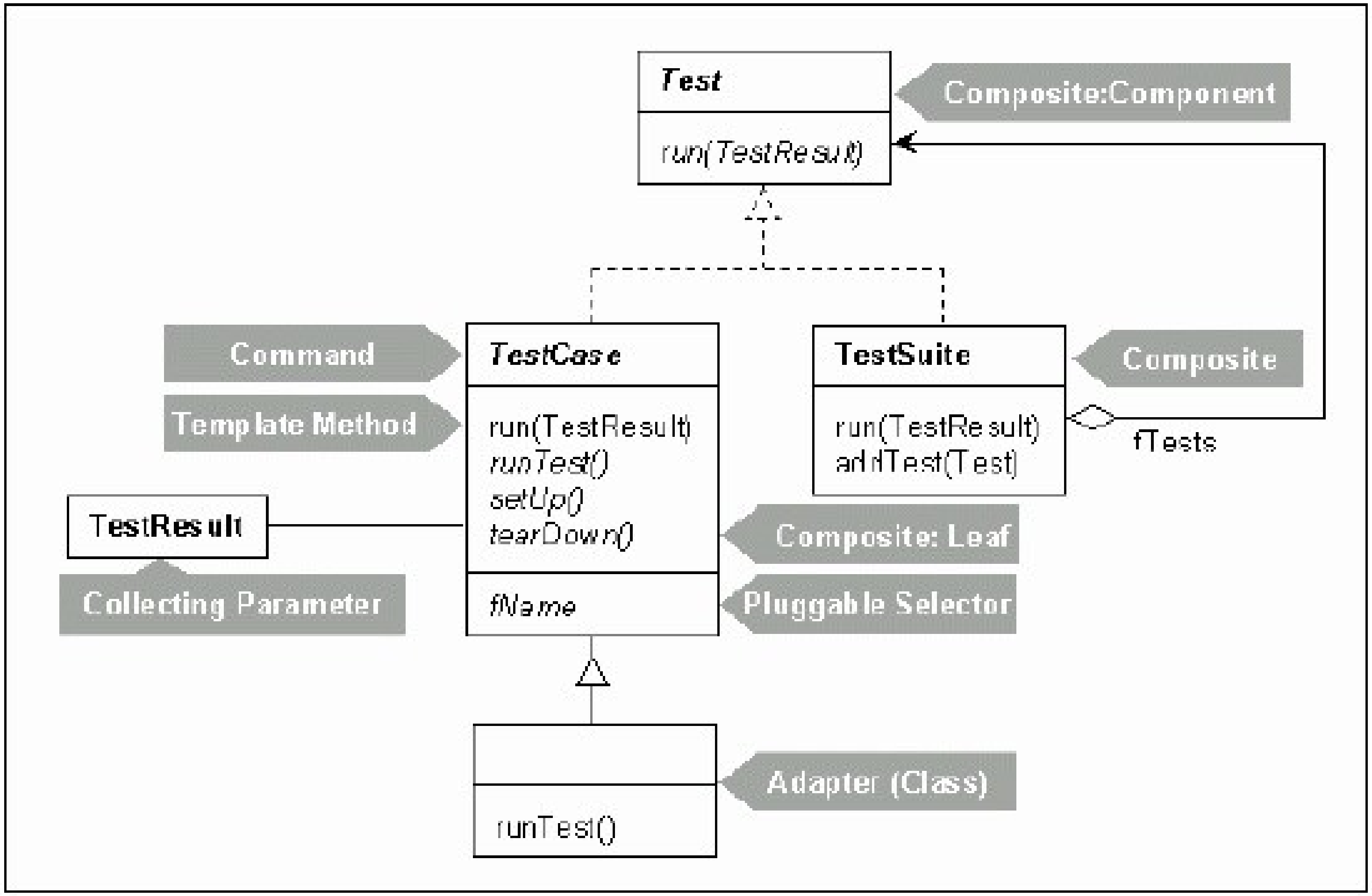
System testing on a complete, integrated system (evaluate compliance with requirements)

JUnit


Kent Beck

Erich Gamma

JUnit



taken from "JUnit: A Cook's Tour"

Tests in JUnit

Tests are realized as public void testX() methods.

A test typically calls a few methods, then checks if the state matches the expectation. If not, it fails.

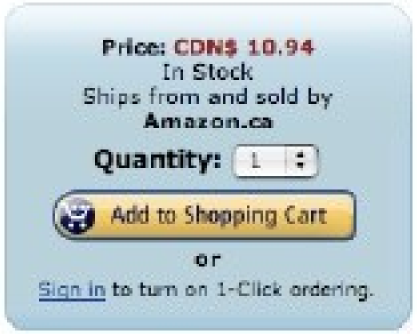
Example: Call *empty()* on the shopping cart and check the state with *isEmpty()*

```
// Tests the emptying of the cart.
public void testEmpty() {
    _bookCart.empty();
    assertTrue(_bookCart.isEmpty());
}
```

JUnit Checklist

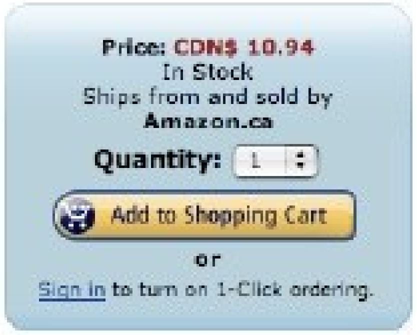
- ☐ Initialize
- ☐ Write test cases
- ☐ Write a test suite
- ☐ Execute test suite

Example: Shopping cart


Price: CDN$ 10.94
In Stock
Ships from and sold by
Amazon.ca
Quantity: 1

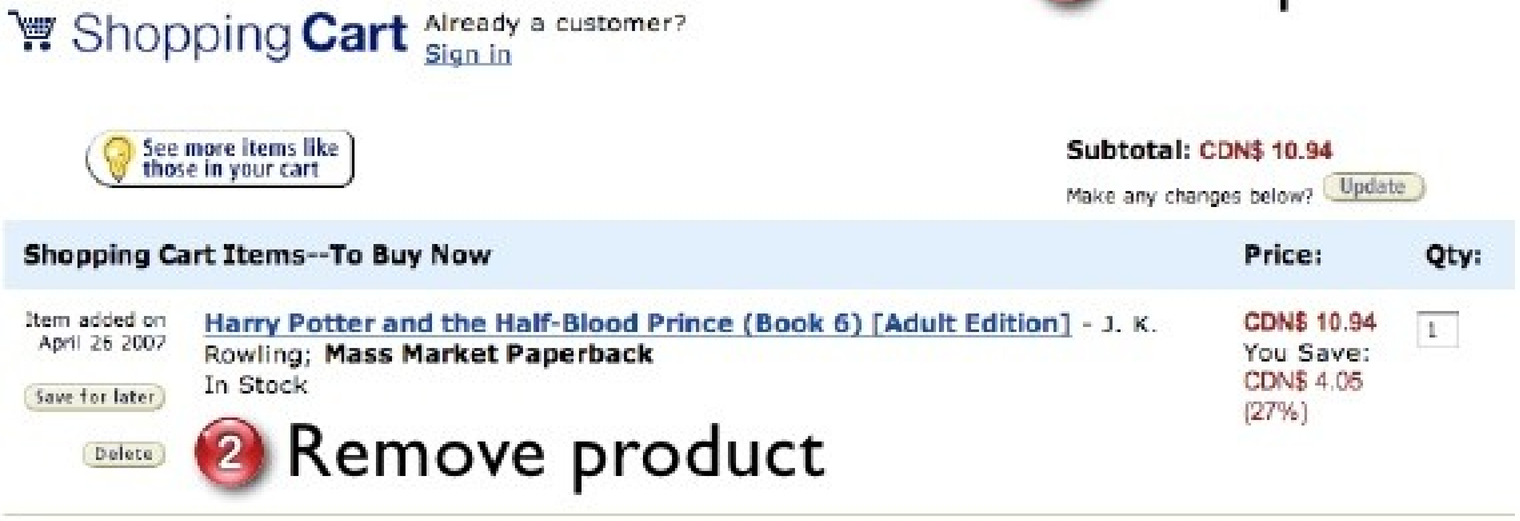
1 Add product

Example: Shopping cart



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

1 Add product



Shopping Cart
Already a customer?
Sign in
See more items like those in your cart

Subtotal: CDN$ 10.94

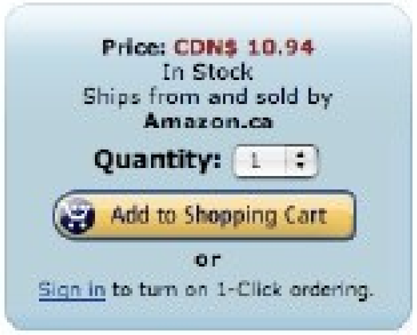
Make any changes below? Update

Shopping Cart Items--To Buy Now		Price:	Qty:
Item added on April 26 2007	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

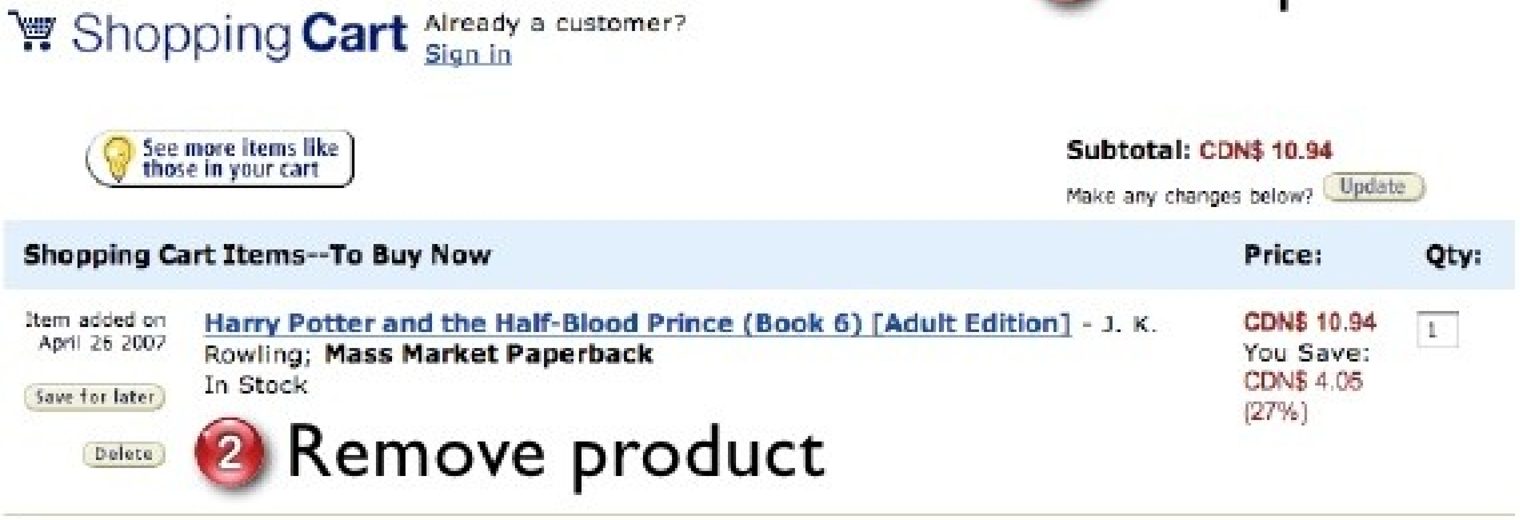
Example: Shopping cart

1 Set of products



Price: CDN$ 10.94
In Stock
Ships from and sold by
Amazon.ca
Quantity: 1
Add to Shopping Cart
or
Sign in to turn on 1-Click ordering.

1 Add product



Shopping Cart Already a customer? Sign in

See more items like those in your cart

Subtotal: CDN$ 10.94
Make any changes below? Update

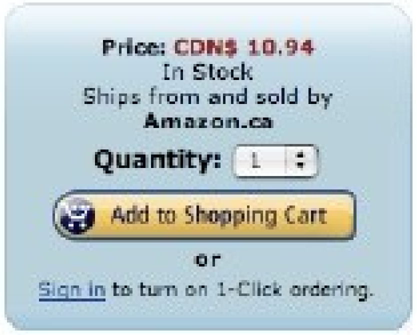
Shopping Cart Items--To Buy Now		Price:	Qty:
Item added on April 26 2007 Save for later Delete	Harry Potter and the Half-Blood Prince (Book 6) [Adult Edition] - J. K. Rowling; **Mass Market Paperback** In Stock	CDN$ 10.94 You Save: CDN$ 4.05 (27%)	1

2 Remove product

Example: Shopping cart

1 Set of products

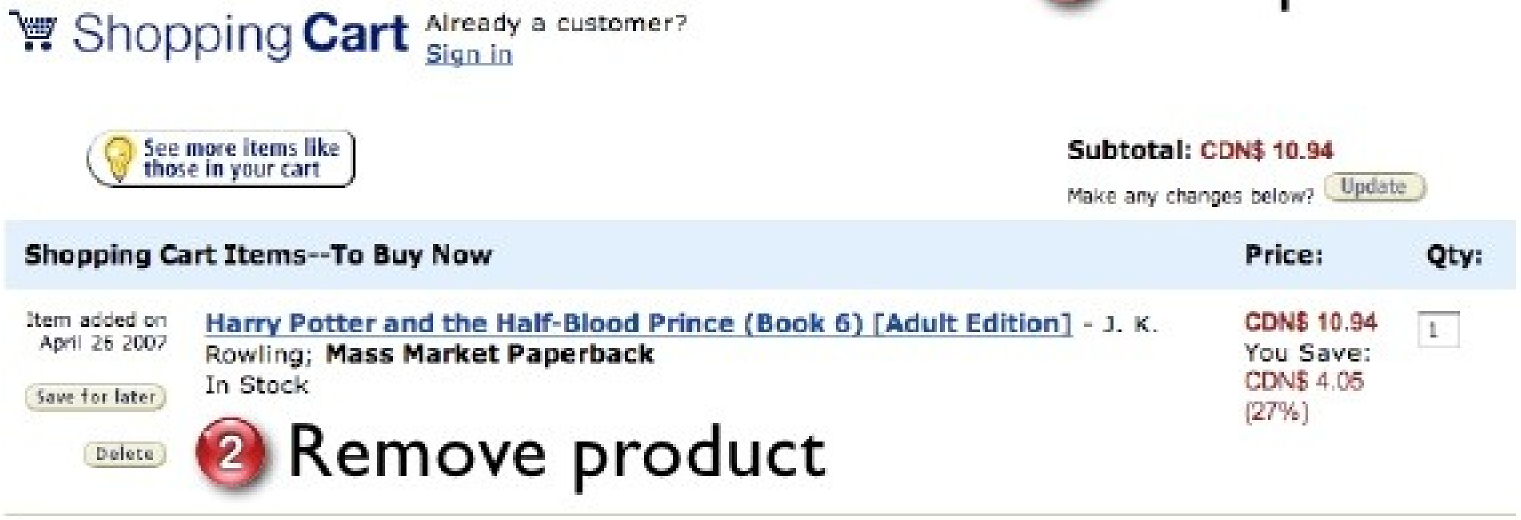
2 Number of products



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

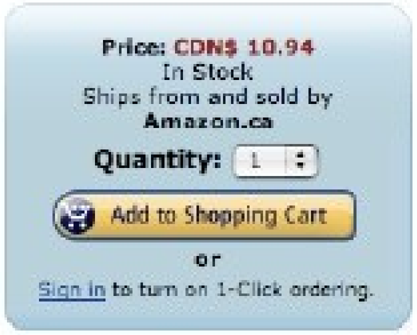
Make any changes below? Update

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

Example: Shopping cart

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



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

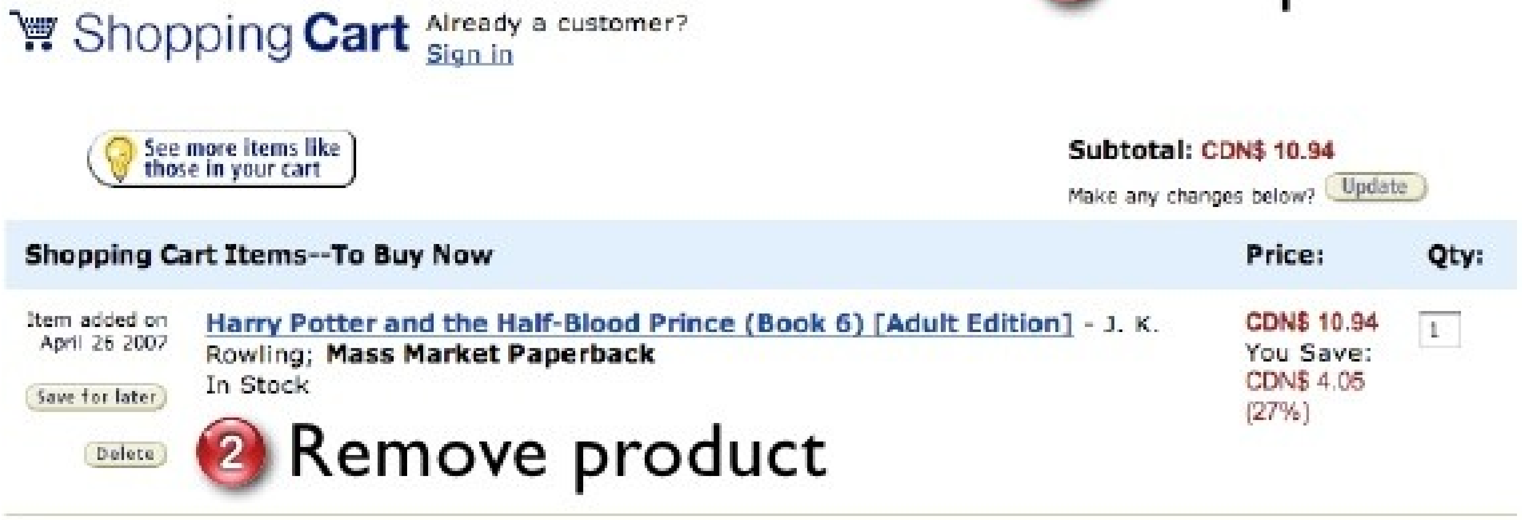
Quantity: 1

Add to Shopping Cart

or

Sign in to turn on 1-Click ordering.

1. Add product



Shopping Cart Already a customer? Sign in

See more items like those in your cart

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$ 4.06 (27%)	1

2. Remove product

Part 1: Initialize

Constructor + Set up and tear down of fixture.

```
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(String name) {
        super(name);
    }
```

Test fixture

```
// 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;
}
```

Part 2: Write test cases

Help methods inherited from *TestCase*:

`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*

The parameter *msg* is optional.

Test: Add product

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







}
```

Test: Add product

```
// 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);
}
```

Test: Remove product

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









}
```

Test: Remove product

```
// 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);
}
```

Test: Exception handling

```java
// Tests removing an unknown product from the cart.
public void testProductNotFound() {









}
```

Test: Exception handling

```
// Tests removing an unknown product from the cart.
public void testProductNotFound() {
     try {
        Product book = new Product("Bones", 4.95);
        _bookCart.removeItem(book);

        fail("Should raise a NotFoundException");
     } catch (NotFoundException nfe) {
        // Should always take this path
     }
}
```

Part 3: Write a test suite

The method *suite()* assembles the test methods into a test suite – using reflection all methods named *testX()* will be part of the test suite.

```
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;
}
```

Part 4: Execute test suite

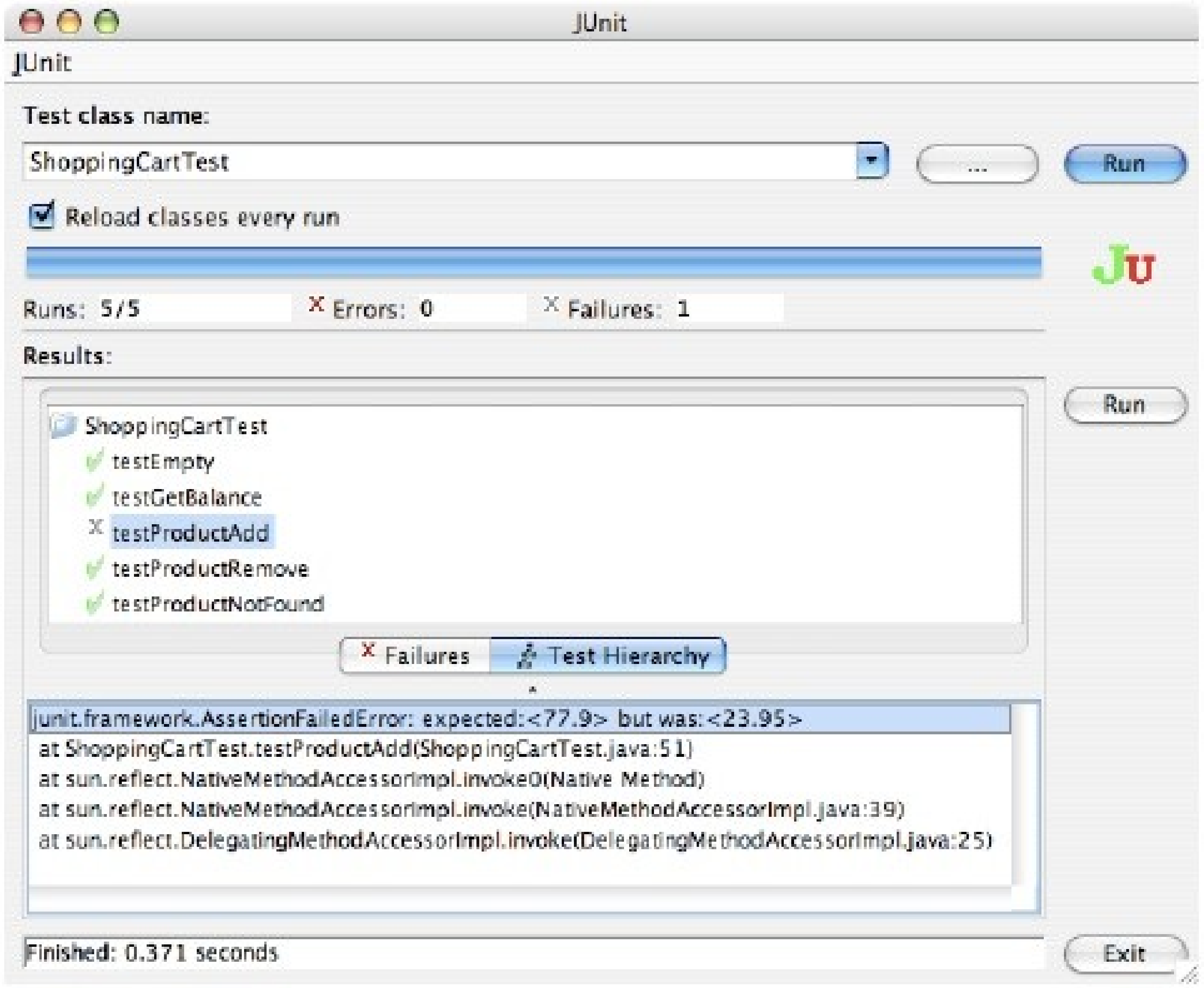
```
// Returns the name of the test case.
public String toString() {
   return getName();
}

// Main method: Call GUI
public static void main(String args[]) {
   String[] tcName = { ShoppingCartTest.class.getName() };
   junit.swingui.TestRunner.main(tcName);
   // junit.textui.TestRunner.main(tcName);
}
```

Execute the *main()* method to run the tests:

```
$ java ShoppingCartTest
```

Demo



```
ShoppingCartTest
  ✓ testEmpty
  ✓ testGetBalance
  ✗ testProductAdd
  ✓ testProductRemove
  ✓ testProductNotFound
```

```
junit.framework.AssertionFailedError: expected:<77.9> but was:<23.95>
 at ShoppingCartTest.testProductAdd(ShoppingCartTest.java:51)
 at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
 at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
 at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
```

Finished: 0.371 seconds

I know I should test, but everything?

Best practices

Tests should be written before the code.

Test everything that could reasonably break.

If it can't break on its own, it's too simple to break (like most get and set methods).

Run all your unit tests as often as possible.

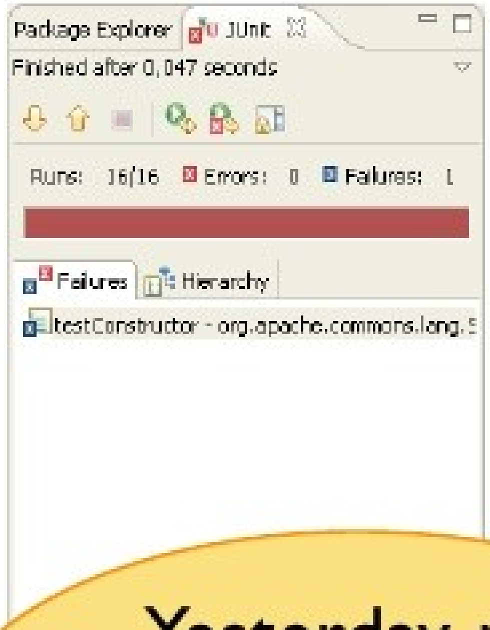
Best practices (cont.)

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

Martin Fowler

Bits of research

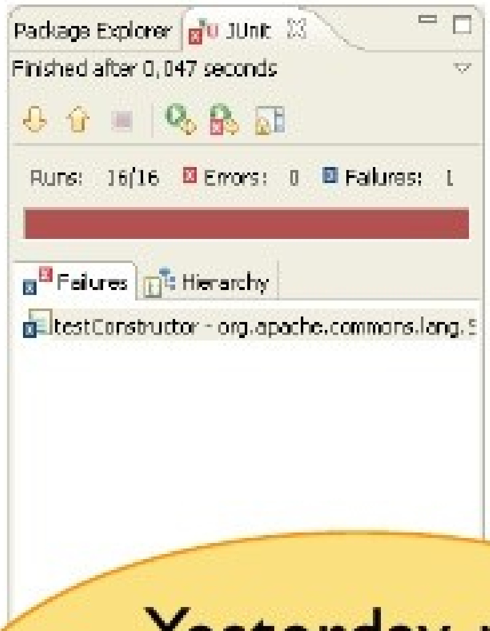
Delta debugging


Package Explorer
JUnit
Finished after 0,047 seconds
Runs: 16/16
Errors: 0
Failures: 1
Failures
Hierarchy
testConstructor - org.apache.commons.lang.S

Yesterday, my program worked. Today, it does not.

Zeller (ESEC/FSE 1999), Burger et al. (ETX 2003)

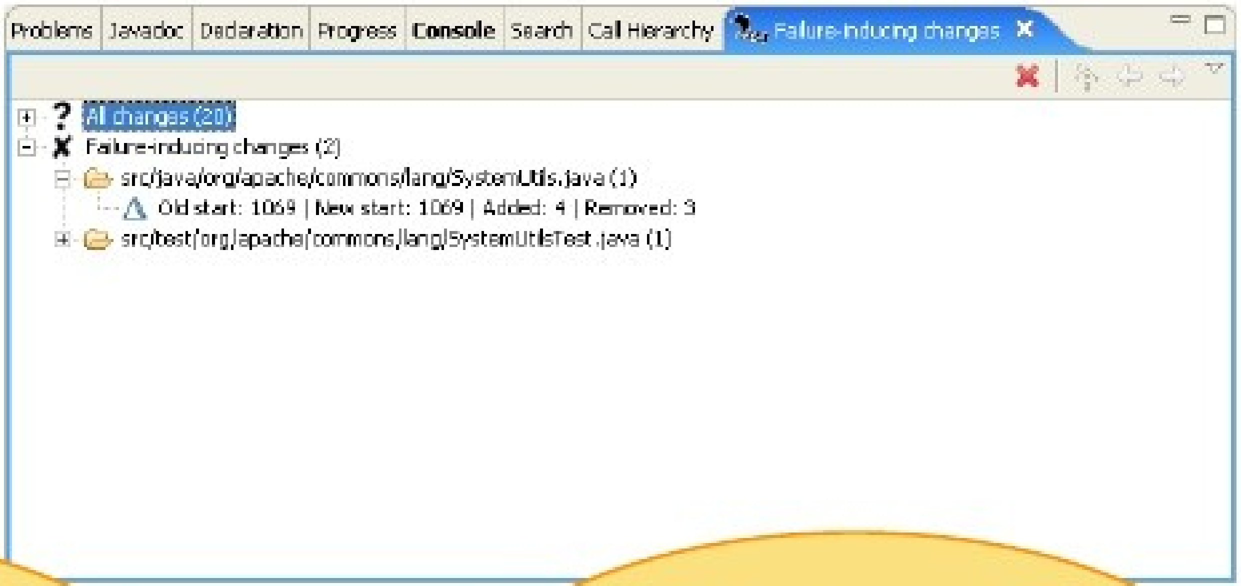
Delta debugging



Finished after 0,047 seconds

Runs: 16/16 Errors: 0 Failures: 1

testConstructor - org.apache.commons.lang.S



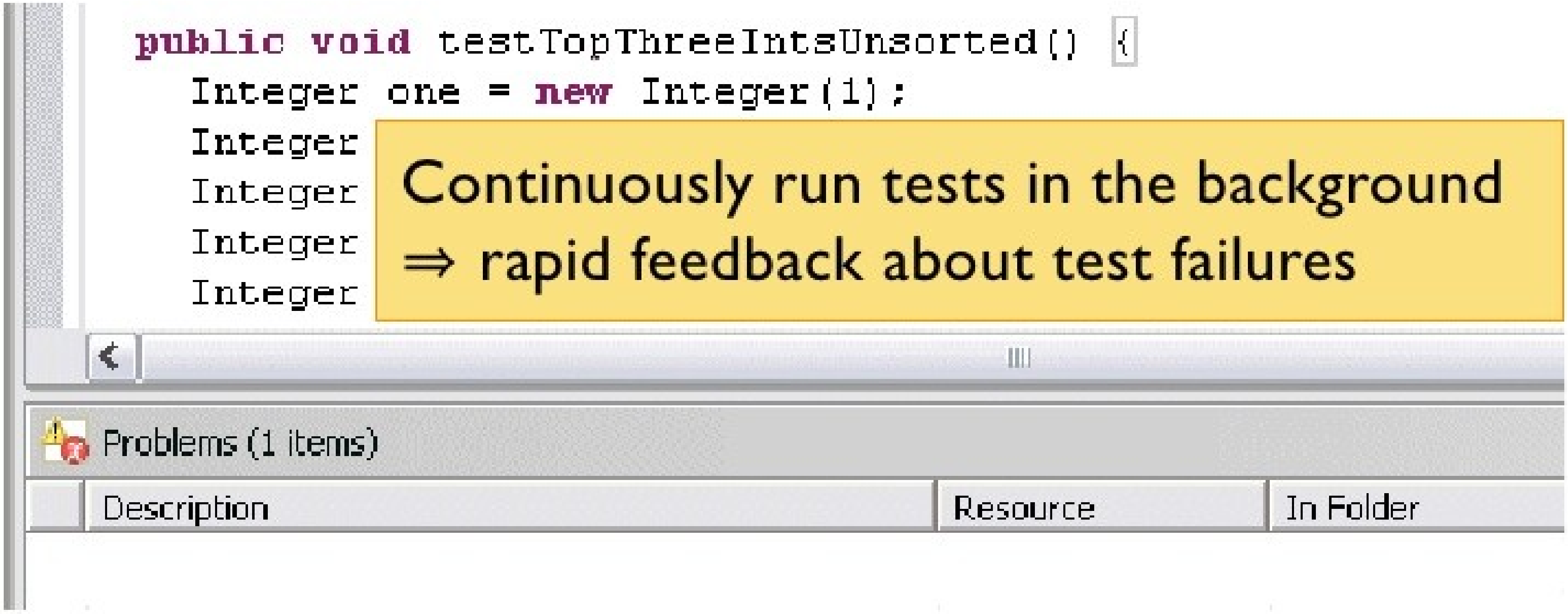
```
? All changes (20)
X Failure-inducing changes (2)
    src/java/org/apache/commons/lang/SystemUtils.java (1)
        Old start: 1069 | New start: 1069 | Added: 4 | Removed: 3
    src/test/org/apache/commons/lang/SystemUtilsTest.java (1)
```

Yesterday, my program worked. Today, it does not.

The change in Line 42 makes the program fail.

Zeller (ESEC/FSE 1999), Burger et al. (ETX 2003)

Continuous testing



```
public void testTopThreeIntsUnsorted() {
   Integer one = new Integer(1);
   Integer
   Integer
   Integer
   Integer
```

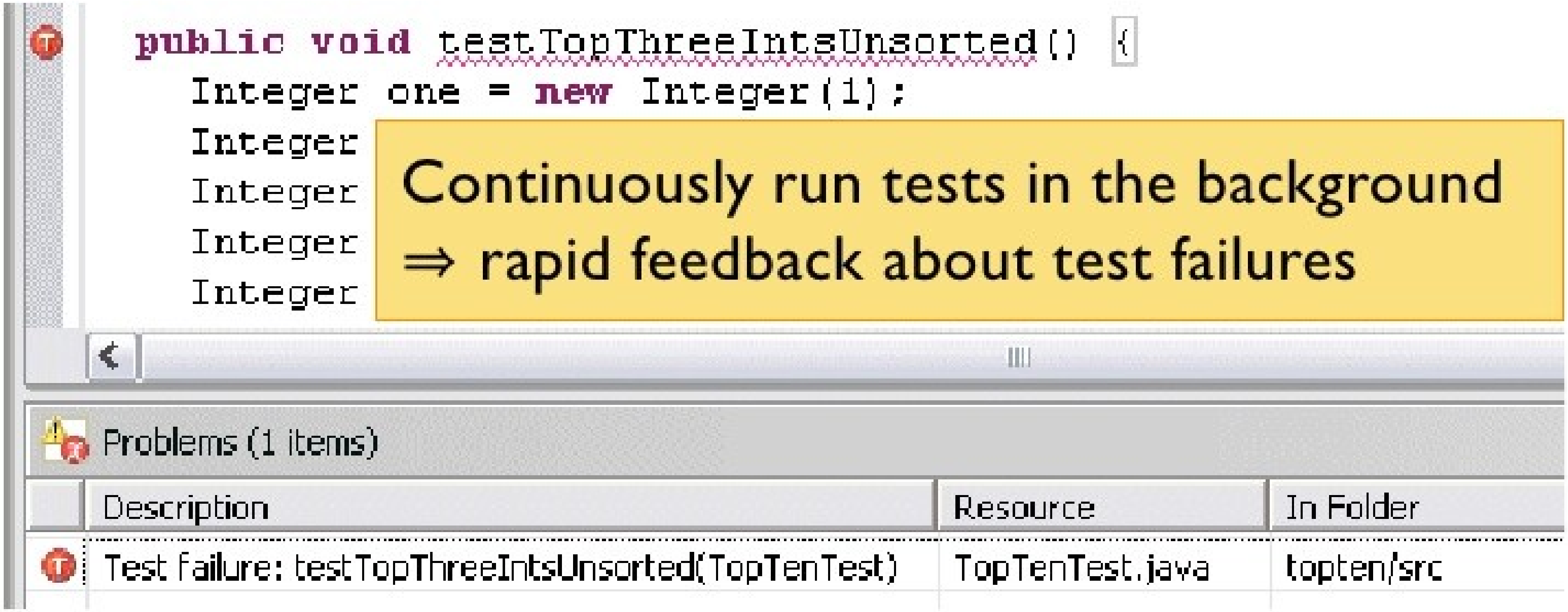
Continuously run tests in the background
⇒ rapid feedback about test failures

Problems (1 items)

Description	Resource	In Folder

Saff and Ernst (ISSRE 2003, ISSTA 2004)

Continuous testing



```
public void testTopThreeIntsUnsorted() {
    Integer one = new Integer(1);
    Integer
    Integer
    Integer
    Integer
```

Continuously run tests in the background
⇒ rapid feedback about test failures

Problems (1 items)

	Description	Resource	In Folder
	Test failure: testTopThreeIntsUnsorted(TopTenTest)	TopTenTest.java	topten/src

Saff and Ernst (ISSRE 2003, ISSTA 2004)

Summary

Testing

- Unit testing
- Integration testing
- System testing

JUnit

- Test cases
- Test suites
- Fixtures

Best practices

Literature @ junit.org

- JUnit Test Infected.
- JUnit A Cook's Tour.

Self-study questions

- How can we test protected and private methods?
- What distinguishes a good test suite from a bad one?
- Think of situations that are difficult to test.

FIM