Test-Driven Development
(that’s not what we meant)

steve.freeman@higherorderlogic.com
@sf105
Add a little test. Run all tests and fail. Refactor to remove duplication.

Make a little change. Run the tests and succeed.
- Write new code only if an automated test has failed
- Eliminate duplication

Add a little test. Make a little change.

Run the tests and fail. Run the tests and succeed.

Refactor to remove duplication.
public class ExchangeRateUploaderTest extends EasyMockTestCase {
    private Logger logger;
    private CurrencyManager mockCurrencyManager;
    private ExchangeRateManager mockExchangeRateManager;
    private PriceManagerFactory mockPriceManagerFactory;
    private PriceManager mockPriceManager;
    private GodObject mockGod;
    private DatabaseFacade mockPersistenceManager;
    private DatabaseFacade mockFrameworkPersistenceManager;
    private CyclicProcessManager mockCyclicProcessManager;
    private SystemVariableManager mockSystemVariableManager;
    private ScreenManager mockScreenManager;
    private Registry registry;
    private User adminUser;
    private Server server;

    private ExchangeRateUploader newExchangeRateUploader(CyclicProcessThread thread) {
        return new ExchangeRateUploader(thread) {
            @Override protected void initializeAction() throws FrameworkException {
                // Does nothing to prevent excessive mocking
            }
            @Override public Logger getLogger() { return logger; }
            @Override protected void setLogMDC() { }
            @Override protected User getUser() { return adminUser; }
            @Override protected CurrencyManager newCurrencyManager() { return mockCurrencyManager; }
            @Override protected PriceManagerFactory newPriceManagerFactory() { return mockPriceManagerFactory; }
            @Override protected CyclicProcessManager newCyclicProcessManager() { return mockCyclicProcessManager; }
            @Override protected DatabaseFacade newPersistenceManager() { return mockPersistenceManager; }
            @Override protected Registry newTaskPerformanceRegistry() { return registry; }
            @Override public PriceDataManager getPriceDataManager() { return null; }
            @Override protected ExchangeRateManager newExchangeRateManager() { return mockExchangeRateManager; }
        };
    }
}
public void testInternalAction() throws FrameworkException {
    mockGod = addMock(GodObject.class);
    expect(mockGod.getPriceDataManager()).andStubReturn(null);
    mockPersistenceManager = addMock(DatabaseFacade.class);
    registry = addMock(Registry.class);
    adminUser = new TestUser("Admin", "", "," , new TestCompany("company"), "");
    mockCyclicProcessManager();

    registry.finalizeThisThread(isA(String.class), isA(String.class));
    Date now = DateUtils.trimMinutesAndSecondsFromDate(new Date());

    mockSystemVariableManager();
    mockLogger();
    mockContextPersistenceManager();
    mockPriceManager();

    CyclicProcessThread thread = mockUserStateAndGetCyclicProcessThread();

    String primeName = "prime";
    String aName = "a";
    String otherName = "other";

    Currency primeCurrency = new TestCurrency(primeName, true);
    Currency aCurrency = new TestCurrency(aName, true);
    Currency otherCurrency = new TestCurrency(otherName, false);

    FXCurrencyPair aCurrencyPair = new FXCurrencyPair(primeCurrency, aCurrency);
    FXCurrencyPair otherCurrencyPair = new FXCurrencyPair(otherCurrency, primeCurrency);

    setupCurrencyManager(primeCurrency, aCurrency, otherCurrency);

    mockExchangeRateManager = addMock(ExchangeRateManager.class);
    mockGetFXRatesAtDatesForCurrencies(now, aCurrencyPair, otherCurrencyPair);

    FrameworkNumber aCurrencyValue = new FrameworkNumber("5");
    FrameworkNumber otherCurrencyValue = new FrameworkNumber("2");
    ExchangeRate aCurrencyRate = new ExchangeRate(primeCurrency, aCurrency, aCurrencyValue, now);
    ExchangeRate otherCurrencyRate = new ExchangeRate(otherCurrency, primeCurrency, otherCurrencyValue, now);

    expect(mockCurrencyManager.getParentToFractionalCurrencyMapForFractionalCurrency()).andStubReturn(newMap());

    expect(mockExchangeRateManager.saveExchangeRate(null, new FrameworkString(primeName), new FrameworkString(aName), aCurrencyValue, new FrameworkDate(now))).andReturn(null);
    expect(mockExchangeRateManager.saveExchangeRate(null, new FrameworkString(otherName), new FrameworkString(primeName), otherCurrencyValue, new FrameworkDate(now))).andReturn(null);

    Map<String, ExchangeRate> out = new HashMap<String, ExchangeRate>();
    out.put("primea", aCurrencyRate);
    out.put("otherprime", otherCurrencyRate);
    expect(mockPriceManager.getLatestExchangeRates(newList(aCurrencyPair, otherCurrencyPair))).andReturn(out);

    mockPMFactoryCleanup();
    replayMocks();

    ExchangeRateUploader uploader = newExchangeRateUploader(thread);
    uploader.initialise();
    uploader.run();
}
private void mockPMFactoryCleanup() {
    PersistenceFactory mockPersistenceFactory = addMock(PersistenceFactory.class);
    mockPersistenceFactory.purgeAllStateForThisThread();
    expect(mockGod.getPersistenceFactory()).andReturn(mockPersistenceFactory).anyTimes();
    expect(mockPersistenceFactory.getExceptionsInRequest()).andReturn(Collections.<Throwable>emptyList()).times(1);
}

private void mockCyclicProcessManager() throws CyclicProcessException {
    mockCyclicProcessManager = addMock(CyclicProcessManager.class);
    expect(mockGod.getCyclicProcessManager()).andStubReturn(mockCyclicProcessManager);
    mockCyclicProcessManager.updateServerCyclicProcessCurrentRunStatus(isA(String.class), isA(String.class), isA(ListENER_STATUS.class), isA(String.class), isA(Double.class), isA(Date.class));
    expectLastCall().anyTimes();
    server = addMock(Server.class);
    expect(mockCyclicProcessManager.getThisServer()).andStubReturn(server);
}

private void setupCurrencyManager(Currency primeCurrency, Currency aCurrency, Currency otherCurrency) {
    mockCurrencyManager = addMock(CurrencyManager.class);
    List<Currency> allCurrencies = new ArrayList<Currency>();
    allCurrencies.add(aCurrency);
    allCurrencies.add(primeCurrency);
    allCurrencies.add(otherCurrency);
    expect(mockCurrencyManager.getPrimeCurrency()).andReturn(primeCurrency).anyTimes();
    expect(mockCurrencyManager.getAllParentCurrencies()).andReturn(allCurrencies).times(2);
}

private void mockGetFXRatesAtDatesForCurrencies(Date now, FXCurrencyPair aCurrencyPair, FXCurrencyPair otherCurrencyPair) throws CurrencyException {
    FrameworkNumber originalACurrencyRate = new FrameworkNumber("1.23");
    Map<FXCurrencyPair, Collection<Date>> currencyPairAndDatesMap = new HashMap<FXCurrencyPair, Collection<Date>>();
    currencyPairAndDatesMap.put(aCurrencyPair, Arrays.asList(now));
    currencyPairAndDatesMap.put(otherCurrencyPair, Arrays.asList(now));
    FXCurrencyPairRates outputObj = addMock(FXCurrencyPairRates.class);
    expect(outputObj.rateMapSize()).andReturn(5).anyTimes();
    expect(outputObj.getActualPriceDateForCurrencyPair(aCurrencyPair, now)).andReturn(null).once();
    expect(outputObj.getRateFromFxRateMap(now, aCurrencyPair)).andReturn(originalACurrencyRate).once();
    expect(outputObj.getActualPriceDateForCurrencyPair(otherCurrencyPair, now)).andReturn(null).once();
    expect(outputObj.getRateFromFxRateMap(now, otherCurrencyPair)).andReturn(originalACurrencyRate); 
    expect(mockExchangeRateManager.getFXRatesAtDatesForCurrencies(currencyPairAndDatesMap)).andReturn(outputObj);
}
private CyclicProcessThread mockUserStateAndGetCyclicProcessThread() {
    Role mockAdminRole = addMock(Role.class);
    CyclicProcessThread thread = addMock(CyclicProcessThread.class);
    expect(thread.getAdminRole()).andReturn(mockAdminRole).anyTimes();
    expect(thread.getAdminUser()).andReturn(adminUser).anyTimes();
    thread.interrupt();
    expectLastCall();
    mockScreenManager = addMock(ScreenManager.class);
    expect(mockGod.getScreenManager()).andReturn(mockScreenManager).anyTimes();
    mockScreenManager.setThreadSignedOnState(new SignedOnState(adminUser, mockAdminRole, false));
    expectLastCall().anyTimes();
    expect(thread.getGod()).andReturn(mockGod).anyTimes();
    expect(thread.getShutdownInProgress()).andReturn(false).anyTimes();
    return thread;
}

private void mockContextPersistenceManager() {
    mockFrameworkPersistenceManager = addMock(DatabaseFacade.class);
    expect(mockGod.getDatabaseFacade()).andReturn(mockFrameworkPersistenceManager).anyTimes();
    mockFrameworkPersistenceManager.beginNewSession();
    expectLastCall().anyTimes();
}

private void mockPriceManager() throws PriceException {
    mockPriceManagerFactory = addMock(PriceManagerFactory.class);
    mockPriceManager = addMock(PriceManager.class);
    expect(mockPriceManagerFactory.newPriceManager(mockFrameworkPersistenceManager,
                                                     mockSystemVariableManager, null))
        .andReturn(mockPriceManager).once();
}

private void mockSystemVariableManager() {
    mockSystemVariableManager = addMock(SystemVariableManager.class);
    expect(mockGod.getSystemVariableManager()).andReturn(mockSystemVariableManager).anyTimes();
    expect(mockSystemVariableManager.getSystemVariable(CYCLIC_PROCESS_LISTENER_HEART_BEAT_TOLERANCE, "30000"))
        .andReturn("30000").anyTimes();
}

private void mockLogger() {
    logger = addMock(Logger.class);
    logger.info(isA(String.class)); expectLastCall().atLeastOnce();
    logger.debug(isA(String.class)); expectLastCall().atLeastOnce();
}

Think up a design

Write some tests that assert that the design exists

Implement a bunch of stuff

Test

Debug

All the tests pass!

Write TODOs to go back and refactor stuff later

Keith Braithwaite
Difficult to understand
Overspecified
Obscure
Brittle
Meaningless failures
“Security Theatre”
Testing Theatre
Should!
given I have a Foo when I call setBar with 3 then getBar should return 3()
What are the tests for TDD?
Steady, incremental progress
Constant positive reinforcement
I think before I code
Things break when they’re supposed to
Surprising designs emerge
an example
would be handy
right about now

www.exampler.com
BasketTest.add_adding_item()

sut = new Basket()

sut.add(ITEM)

assertEquals(
    ITEM,
    backdoor(sut, "itemList")[0])
Write readable code
is_empty_when_created()
    assertThat( new Basket().itemCount(), equals(0) )

returns_items_in_the_order_they_were_added()
    basket = new Basket()
        .add(pen).add(ink).add(paper)
    assertThat( basket,
        hasItems(pen, ink, paper) )

totals_up_the_cost_of_its_items()
fails_when_removing_an_absent_item()
...

Thursday, 17 October 2013
is_empty_when_created()
assertThat( new Basket().itemCount(),
equals(0) )

returns_items_in_the_order_they_were_added()
basket = new Basket()
   .add(pen).add(ink).add(paper)
assertThat( basket,
    hasItems(pen, ink, paper) )

totals_up_the_cost_of_its_items()
fails_when_removing_an_absent_item()
...

Interfaces, not internals
Protocols, not Interfaces

is_empty_when_created()
assertThat( new Basket().itemCount(),
equals(0) )

returns_items_in_the_order_they_were_added()
basket = new Basket()
.add(pen).add(ink).add(paper)
assertThat( basket,
  hasItems(pen, ink, paper) )

totals_up_the_cost_of_its_items()
fails_when_removing_an_absent_item()
...

steve.freeman@higherorderlogic.com ©2013
is_empty_when_created()
assertThat( new Basket().itemCount(),
equals(0) )
returns_items_in_the_order_they_were_added()
basket = new Basket()
  .add(pen).add(ink).add(paper)
assertThat( basket, hasItems(pen, ink, paper) )
totals_up_the_cost_of_its_items()
fails_when_removing_an_absent_item()
...

From simple to general
“It’s about explaining the domain, not about proving the correctness of the code.”

Andrew Parker
When you’re lost, slow down
The Addison-Wesley Signature Series

Test-Driven Development
By Example

Kent Beck
Test at the right level
Write a failing end-to-end test

Write a failing unit test

Make the test pass

Refactor

Deployable System
Traffic

Click-through Rate: 10.14%
PVs with Outbrain: 93,557
Clicks: 9,485
Organic: 8,432  Paid: 1,053

Revenue

RPM: $0.39
PVs with Paid Links: 93,557
Revenue: $36.75

Traffic Sources

<table>
<thead>
<tr>
<th>#</th>
<th>Source</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>62.86%</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>9.62%</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>5.08%</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>4.03%</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>18.41%</td>
</tr>
</tbody>
</table>

View detailed report
Why does TDD work?
Focussed
anyway, i had been using sunit for a couple of months when i remembered this passage from a programming book i'd read as a kid. it said the way to program is to look at the input tape and manually type in the output tape you expect. then you program until the actual and expected tapes match.

i thought, what a stupid idea. i want tests that pass, not tests that fail. why would i write a test when i was sure it would fail. well, i'm in the habit of trying stupid things out just to see what happens, so i tried it and it worked great.

i was finally able to separate logical from physical design. i'd always been told to do that but no one ever explained how.
Concrete
Japanese Short Code

Sierra JAPANESE SHORT name is built from the TSE code and always taken from GLOSS.

<table>
<thead>
<tr>
<th>sierra value for</th>
<th>JAPANESE SHORT name</th>
</tr>
</thead>
<tbody>
<tr>
<td>derives from</td>
<td>gloss classification 2200</td>
</tr>
<tr>
<td>derives from</td>
<td>gloss identifier EXCO</td>
</tr>
</tbody>
</table>

**Examples**

<table>
<thead>
<tr>
<th>Notes</th>
<th>sources</th>
<th>gloss classification</th>
<th>gloss identifier</th>
<th>expectedSierraValue()</th>
<th>expectedWarningMessage()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>gloss</td>
<td>JBGX</td>
<td>003100076</td>
<td>JFB310</td>
<td><em>NONE</em></td>
</tr>
<tr>
<td></td>
<td>bnd2</td>
<td>-</td>
<td>-</td>
<td><em>NONE</em></td>
<td><em>NONE</em></td>
</tr>
<tr>
<td></td>
<td>bnd2, gloss</td>
<td>JBGX</td>
<td>006400064</td>
<td>JNR64</td>
<td><em>NONE</em></td>
</tr>
<tr>
<td></td>
<td>gloss, bnd2</td>
<td>JBGX</td>
<td>000070031</td>
<td>JFWI</td>
<td><em>NONE</em></td>
</tr>
</tbody>
</table>

Empty TSE code

Unsupported MOF code

Supported MOF code, unsupported TSE code

**When Issued Inflation Linked JGB**

<table>
<thead>
<tr>
<th>Notes</th>
<th>sources</th>
<th>gloss classification</th>
<th>gloss identifier</th>
<th>expectedSierraValue()</th>
<th>expectedWarningMessage()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>gloss</td>
<td>JBGX</td>
<td>00090035</td>
<td>JSWI</td>
<td><em>NONE</em></td>
</tr>
<tr>
<td></td>
<td>gloss</td>
<td>JBGX</td>
<td>00100037</td>
<td>JBWI</td>
<td><em>NONE</em></td>
</tr>
<tr>
<td></td>
<td>gloss</td>
<td>JBGX</td>
<td>00070038</td>
<td>JXWI</td>
<td><em>NONE</em></td>
</tr>
<tr>
<td></td>
<td>gloss</td>
<td>JBGX</td>
<td>00090039</td>
<td>JLWI</td>
<td><em>NONE</em></td>
</tr>
</tbody>
</table>

**fit.Summary**

<table>
<thead>
<tr>
<th>counts</th>
<th>22 right, 0 wrong, 0 ignored, 0 exceptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>input file</td>
<td>C:\Projects\sierra\acceptance-tests\features\Importer\NamesAndIdentifiers\JapaneseShortCode.html</td>
</tr>
<tr>
<td>input update</td>
<td>Mon Oct 17 09:49:47 BST 2005</td>
</tr>
<tr>
<td>output file</td>
<td>C:\Projects\sierra\build-output\artifacts\tests\fit\features\Importer\NamesAndIdentifiers\JapaneseShortCode.html</td>
</tr>
<tr>
<td>run date</td>
<td>Mon Oct 31 11:16:30 GMT 2005</td>
</tr>
<tr>
<td>run elapsed time</td>
<td>0:00.40</td>
</tr>
</tbody>
</table>
Empirical
Write a failing test

Refactor

Make the test pass
Understand the problem → Broad-Brush Design (Architecture) → Production Release

- Automate
  - Build
  - Deployment
  - End-to-End Tests

Process:
- Write a failing end-to-end test
- Debug and fix
- Deploy and test
- Repeat
Test-Driven Development
(that’s not what we meant)

steve.freeman@higherorderlogic.com
@sf105
Test-Driven Development
(that’s not what we meant)

steve.freeman@higherorderlogic.com
@sf105

Remember the app