Patterns for Test Driven Development

Excerpted from "Test-Driven Development by Example" by Kent Beck, Addison-Wesley, ISBN 978-0-321-14653-3. Please refer to that volume for full details.

Chapter 25: Test-Driven Development Patterns

- Test (noun) –
- Isolated Test –
- Test List –
- Test First –
- Assert First –
- Test Data –
- Evident Data –

Chapter 26: Red Bar Patterns

- One Step Test –
- Starter Test –
- Explanation Test –
- Learning Test –
- Another Test –
- Regression Test –
- Break –
- Do Over –
- Cheap Desk, Nice Chair –

Chapter 27: Testing Patterns

- Child Test –
- Mock Object –
- Self Shunt –
- Log String –
- Crash Test Dummy –
- Broken Test –
- Clean Check-in –

Chapter 28: Green Bar Patterns

- Fake It ('Til You Make It) –
- Triangulate –

- Obvious Implementation –
- One to Many –

Chapter 29: xUnit Patterns

- Assertion –
- Fixture –
- External Fixture –
- Test Method –
- Exception Test –
- All Tests –

Chapter 30: Design Patterns

- These patterns are marked for use in Test Writing, TW, and/or Refactoring, R.

- Command (TW) –
- Value Object (TW) –
- Null Object (R) –
- Template Method (R) –
- Pluggable Object (R) –
- Pluggable Selector (R) –
- Factory Method (TW,R) –
- Imposter (TW,R) –
- Composite (TW,R) –
- Collecting Parameter (TW,R) -

Chapter 31: Refactoring

- Reconcile Differences –
- Isolate Change –
- Migrate Data –
- Extract Method –
- Inline Method –
- Extract Interface –
- Move Method –
- Method Object –
- Add Parameter –

Chapter 32: Mastering TDD

- This chapter is a series of questions to consider when incorporating TDD into a project.

Patterns for Test Driven Development

Excerpted from "Test-Driven Development by Example" by Kent Beck, Addison-Wesley, ISBN 978-0-321-14653-3. Please refer to that volume for full details.

Chapter 25: Test-Driven Development Patterns

AT ⇔ stress

- Test (noun) – Having (automated) tests ≠ doing testing
- Isolated Test – Independent + Fast
- Test List – Add to list as each discovered, then pick order
- Test First – lowers stress, controls scope
- Assert First – 1st step to writing a test, work backwards
- Test Data – simple unique values that reveal errors (1 ≠ 1)
- Evident Data – Expressions vs. constants to show meaning. 1.1 + 2.4

Chapter 26: Red Bar Patterns

- One Step Test – Pick a test moving 1 step towards goal.
- Starter Test – Some variation of no-op, establishes structure.
- Explanation Test – Explain things in terms of tests.
- Learning Test – Learn to use 3rd party API via testing
- Another Test – Stay on track by adding sidetrack ideas to test list
- Regression Test – When defects reported, write test that would have prevented them.
- Break – Take one when stuck - hourly, daily, weekly, yearly
- Do Over – Restart @ □ 1 if necessary
- Cheap Desk, Nice Chair – Lots of real estate, comfy for hours

Chapter 27: Testing Patterns

- Child Test – If a test won't run + is too big, create a smaller sub test
- Mock Object – www.mockobjects.com. Substitute for complex resource eg DB
- Self Shunt – Have object under test communicate with test, as mock
- Log String – Useful for recording order of operations/calls
- Crash Test Dummy – For testing error code, create crash dummies (mocks)
- Broken Test – Leave BT @ end of day, for obvious start next day
- Clean Check-in – When team relies on your work. When solo,

Chapter 28: Green Bar Patterns

- Fake It ('Til You Make It) – 1st pass: return a constant
- Triangulate – Abstract (generalize) only with 2 or more tests

- Obvious Implementation – Good, but not always simplest. 1/many tools.
- One to Many – Start w/ collections with single items. testing

Chapter 29: xUnit Patterns

- Assertion – Auto judge boolean. Use public API, not private data.
- Fixture – Factor out common initialization code into setup() routine
- External Fixture – Restore system, release resources, in teardown() "
- Test Method – Method within class named test— all tests of class same setup
- Exception Test – Code to fail when expected exception is not thrown,
- All Tests – Suite/subclass to run all tests.

```
try {
    func(); // throws
    fail();
} catch (expected only ex) {
    Exception
}
```

Chapter 30: Design Patterns

- These patterns are marked for use in Test Writing, TW, and/or Refactoring, R.
- Command (TW) – Encapsulate all params of method call, for later
- Value Object (TW) – An object encapsulating an unchangable constant
- Null Object (R) – Encapsulating null w. protocols helps error checking
- Template Method (R) –
- Pluggable Object (R) – Avoid lots of if-else cases w indirect access
- Pluggable Selector (R) – Eliminates switch via reflection
- Factory Method (TW,R) –
- Imposter (TW,R) – new class w. same interface replaces existing
- Composite (TW,R) –
- Collecting Parameter (TW,R) - Pass a container around to collect results.

Chapter 31: Refactoring

- Reconcile Differences –
- Isolate Change –
- Migrate Data –
- Extract Method –
- Inline Method –
- Extract Interface –
- Move Method –
- Method Object –
- Add Parameter –

Chapter 32: Mastering TDD

- This chapter is a series of questions to consider when incorporating TDD into a project.