CODE CAMP NYC

Unit Testing

The hard parts

October 10th, 2015

Shaun Abram

Blog: shaunabram.com
Email: shaun@abram.com
Twitter: @shaunabram
LinkedIn: linkedin.com/in/sabram

Test Obsessed?

How much do you know…

- Test coverage
- Dependency Injection
- Mock vs stubs
- Testing private or static methods
- Test driven development

What is a unit test?

A piece of code that executes a specific functionality (‘unit’) in the code, and

- Confirms behavior or result is as expected
- Determines if code is ‘fit for use’
- Does it do what the developer intended?

Before, we manually verified, but not easily repeatable.

Why unit test?

Economics $

but also…

- Drive design
- Create defensive code against bad input
- Act as safety buffers for regression bugs
- Provide documentation
- Clean code
- Less bugs

What is a unit?

- A class? A method? A single path through a method?
- The smallest testable part of an application.
- A single functional use case.

What makes a good unit test?

- Provides benefit!
- Readable, Understandable, Maintainable
- Independent
  - run in any order, no DB or File access
- Consistent / deterministic
- Runs fast
- Tests a single logical concept in the system

Unit testing limitations

- Can not prove the absence of bugs
- Lot’s of code (x3-5)
- Some things difficult to test

Dependency Injection

WidgetService.java — nycc

UNREGISTERED

WidgetService.java

```java
public class WidgetService {
    WidgetDao dao = new WidgetDaoImpl();

    public Widget getStandardizedWidget() {
        Widget widget = dao.getWidget();
        if (widget != null) {
            widget.setStandardized(true);
        }
        return widget;
    }

}
```

WidgetService.java

```java
public class WidgetService {
    WidgetDao dao = new WidgetDaoImpl();

    public Widget getStandardizedWidget() {
        Widget widget = dao.getWidget();
        if (widget != null) {
            widget.setStandardized(true);
        }
        return widget;
    }

}
```

WidgetService.java

```java
public class WidgetService {
    WidgetDao dao;

    public WidgetService(WidgetDao dao) {
        this.dao = dao;
    }

    public Widget getStandardizedWidget() {
        Widget widget = dao.getWidget();
        if (widget != null) {
            widget.setStandardized(true);
        }
        return widget;
    }

}
```

Dependency Injection

- Useful for testing – inject test doubles
- Also helps reduce coupling

Test Doubles

Mocks

Stubs

Dummies

Spies

Fakes

Test Doubles

Mocks

Stubs

Dummies

Spies

Fakes

Test Doubles: Mocks

Mocks: An overloaded term!

- uses behavior verification
- objects pre-programmed with expectations

WidgetService.java

UNREGISTERED

```java
public class WidgetService {
    final WidgetDao dao;

    public WidgetService(WidgetDao dao) {
        this.dao = dao;
    }

    public void createWidget(Widget widget) {
        //misc business logic
        //e.g. validating widget is valid

        dao.saveWidget(widget);
    }

}
```

WidgetService.java

```java
public class WidgetService {
    final WidgetDao dao;

    public WidgetService(WidgetDao dao) {
        this.dao = dao;
    }

    public void createWidget(Widget widget) {
        //misc business logic
        //e.g. validating widget is valid

        dao.saveWidget(widget);
    }

}
```

WidgetServiceTest.java

```java
public class WidgetServiceTest {

    //test fixtures
    WidgetDao widgetDao = mock(WidgetDao.class);
    WidgetService widgetService
            = new WidgetService(widgetDao);
    Widget widget = new Widget();

    @Test
    public void createWidget_saves_widget() {
        //call method under test
        widgetService.createWidget(widget);

        //verify expectation
        verify(widgetDao).saveWidget(widget);
    }

}
```

- Mocks use behavior verification
- Can specify how to respond when called
- Roll your own?

Test Doubles

Mocks

Stubs

Dummies

Spies

Fakes

Test Doubles: Stubs

‘stubs out’
or provides a simplified version
of the implementation
for the purposes of testing

WidgetService.java

UNREGISTERED

```java
public class WidgetService {
    final ManagerService manager;

    public WidgetService(ManagerService m) {
        this.manager = m;
    }

    public void standardize(Widget widget) {
        if (manager.isActive()) {
            widget.setStandardized(true);
        }
    }
}
```

WidgetService.java

```java
public class WidgetService {
    final ManagerService manager;

    public WidgetService(ManagerService m) {
        this.manager = m;
    }

    public void standardize(Widget widget) {
        if (manager.isActive()) {
            widget.setStandardized(true);
        }
    }
}
```

WidgetServiceTest.java

```java
public class WidgetServiceTest {

    class ManagerServiceStub extends ManagerService {
        @Override
        public boolean isActive() {
            return true;
        }
    }

    ManagerServiceStub managerServiceStub
            = new ManagerServiceStub();
    WidgetService widgetService
            = new WidgetService(managerServiceStub);
    Widget widget = new Widget();

    @Test
    public void standardize_widget_when_active() {
        //call method under test
        widgetService.standardize(widget);

        //verify state
        assertTrue(widget.isStandardized());
    }
}
```

- Stubs can use state or behavior verification
- Provides a useful approach for test fixture configurability

Test Doubles

Mocks

Stubs

Dummies

Spies

Fakes

Test Doubles: Dummies

- A very dumb class!
- Contains next to nothing - enough to compile
- Pass when you don’t expect to be used

```java
public class WidgetDaoDummy implements WidgetDao {

    @Override
    public Widget getWidget() {
        throw new RuntimeException("Not expected to be called");
    }

    @Override
    public void saveWidget(Widget widget) {
        throw new RuntimeException("Not expected to be called");
    }

}
```

- Use dummies with state or behavior verification
- Can create as inner class
- Can replicate with mocks

Unit Testing – the tricky parts

- Legacy code
- Privates
- Statics

General approach to testing legacy code

1) Start with coarse grained tests

- No modifications
- Strict and rigid
- Detect regressions
- Short term bridges - delete

General approach to testing legacy code

1) Start with coarse grained tests
2) Add finer grained tests

- Incrementally add more unit
- start gently refactoring
- Use TDD – if possible
- Test from 1) should protect you

General approach to testing legacy code

1) Start with coarse grained tests
2) Add finer grained tests
3) Continuously refactor

Use patterns such as

- Extract Method/Class
- Move Method/Field
- tease apart methods (>10 lines smells)

Unit Testing – the tricky parts

- Legacy code
- **Privates**
- Statics

How do you test private methods?

Indirectly! Best tested via public interface

But sometimes…

- Legacy code
  - With limited capability to refactor, or
  - adding a safety net before refactoring
- A public method calls several private methods, each with complex logic

Testing Private Methods

1. Refactor
2. Change the visibility
   - Bad practice
   - Public API? Very bad practice!
   - Internal & stepping stone -> lesser evil
   - Java private -> package; .Net protected or internal?
   - Document (@VisibleForTesting)

Testing Private Methods

1. Refactor
2. Change the visibility
3. Use Reflection

- No code modification, but…

4. Other options…
   - Testing frameworks
   - InternalsVisibleToAttribute
   - Private Accessors

Testing static methods

untitled UNREGISTERED

untitled

```java
public class WidgetService {

    public Widget getStandardizedWidget() {
        Widget widget = WidgetDao.getWidget();
        if (widget != null) {
            widget.setStandardized(true);
        }
        return widget;
    }

}
```

untitled

```java
public class WidgetService {

    public Widget getStandardizedWidget() {
        Widget widget = WidgetDao.getWidget();
        if (widget != null) {
            widget.setStandardized(true);
        }
        return widget;
    }

}
```

untitled

```java
public class WidgetService {
    final WidgetDao dao;

    public WidgetService(WidgetDao dao) {
        this.dao = dao;
    }

    public Widget getStandardizedWidget()
        Widget widget = dao.getWidget();
        if (widget != null) {
            widget.setStandardized(true);
        }
        return widget;
    }
}
```

```java
public class WidgetService {

    public Widget getStandardizedWidget() {
        Widget widget = WidgetDao.getWidget();
        if (widget != null) {
            widget.setStandardized(true);
        }
        return widget;
    }

}
```

```java
public class WidgetService {
    final WidgetDao dao;

    public WidgetService(WidgetDao dao) {
        this.dao = dao;
    }

    public Widget getStandardizedWidget()
        Widget widget = dao.getWidget();
        if (widget != null) {
            widget.setStandardized(true);
        }
        return widget;
    }
}
```

```java
public class WidgetServiceTest {

    WidgetDao widgetDaoMock= mock(WidgetDao.class);
    WidgetService widgetService = new WidgetService(widgetDaoMock);
    Widget unstandardizedWidget = new Widget();

    @Test
    public void getStandardizedWidget_returns_standardized_widget() {
        //set expectations on the mock
        when(widgetDaoMock.getWidget()).thenReturn(unstandardizedWidget);
        //call the method under test
        Widget widget = widgetService.getStandardizedWidget();
        //verify behavior
        assertTrue(widget.isStandardized());
    }

}
```

Testing static methods

- DI ideal
- But sometimes not pragmatic

untitled UNREGISTERED

untitled

```java
public class WidgetService {

    public Widget getStandardizedWidget() {
        Widget widget = WidgetDao.getWidget();
        if (widget != null) {
            widget.setStandardized(true);
        }
        return widget;
    }

}
```

```java
public class WidgetService {

    public Widget getStandardizedWidget() {
        Widget widget = WidgetDao.getWidget();
        if (widget != null) {
            widget.setStandardized(true);
        }
        return widget;
    }

}
```

```java
public class WidgetService {

    public Widget getStandardizedWidget()
        Widget widget = getWidget();
        if (widget != null) {
            widget.setStandardized(true);
        }
        return widget;
    }

    Widget getWidget() {
        return WidgetDao.getWidget();
    }
}
```

Refactor to wrap the static call in an instance method
Which can then be mocked…

untitled

```java
public class WidgetService {

    public Widget getStandardizedWidget() {
        Widget widget = WidgetDao.getWidget();
        if (widget != null) {
            widget.setStandardized(true);
        }
        return widget;
    }

}
```

untitled

```java
public class WidgetService {

    public Widget getStandardizedWidget()
        Widget widget = getWidget();
        if (widget != null) {
            widget.setStandardized(true);
        }
        return widget;
    }

    Widget getWidget() {
        return WidgetDao.getWidget();
    }
}
```

WidgetServiceTest.java

UNREGISTERED

```java
public class WidgetServiceTest {

    WidgetService service = partialMock(new WidgetService());
    Widget unstandardizedWidget = new Widget();

    @Test
    public void getStandardizedWidget_returns_standardized_widget() {
        //set expectations
        when(service.getWidget()).thenReturn(unstandardizedWidget);
        //cal method under test
        Widget widget = service.getStandardizedWidget();
        //verify behavior
        assertTrue(widget.isStandardized());
    }
}
```

WidgetServiceTest.java

```java
public class WidgetServiceTest {

    WidgetService service = partialMock(new WidgetService());
    Widget unstandardizedWidget = new Widget();

    @Test
    public void getStandardizedWidget_returns_standardized_widget() {
        //set expectations
        when(service.getWidget()).thenReturn(unstandardizedWidget);
        //cal method under test
        Widget widget = service.getStandardizedWidget();
        //verify behavior
        assertTrue(widget.isStandardized());
    }
}
```

WidgetServiceTest.java

```java
public class WidgetServiceTest {

    WidgetService service = partialMock(new WidgetService());
    Widget unstandardizedWidget = new Widget();

    @Test
    public void getStandardizedWidget_returns_standardized_widget() {
        //set expectations
        when(service.getWidget()).thenReturn(unstandardizedWidget);
        //cal method under test
        Widget widget = service.getStandardizedWidget();
        //verify behavior
        assertTrue(widget.isStandardized());
    }
}
```

WidgetServiceStubTest.java

```java
public class WidgetServiceStubTest {
    WidgetServiceStub service = new WidgetServiceStub();
    Widget unstandardizedWidget = new Widget();

    class WidgetServiceStub extends WidgetService {
        public Widget getStandardizedWidget() {
            return unstandardizedWidget;
        }
    }

    @Test
    public void getStandardizedWidget_returns_standardized_widget() {
        //call method under test
        Widget widget = service.getStandardizedWidget();
        //verify results
        assertTrue(widget.isStandardized());
    }
}
```

Testing static methods

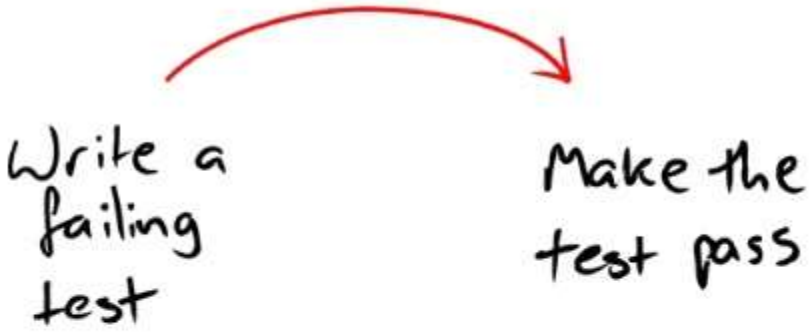
- Refactor – use DI
- Wrap static call in a instance method
- Use a mocking framework?

Test Driven Development

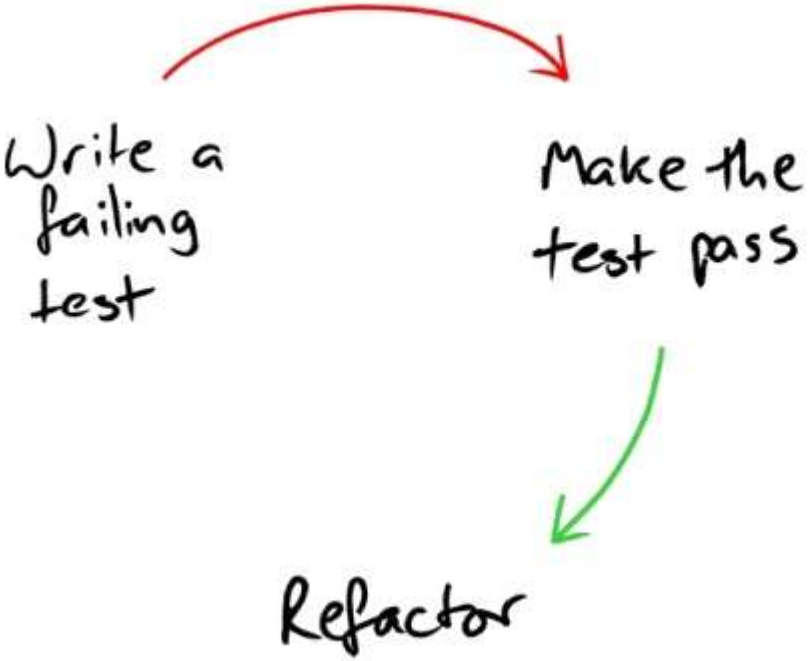
Test Driven Development

Write a
failing
test

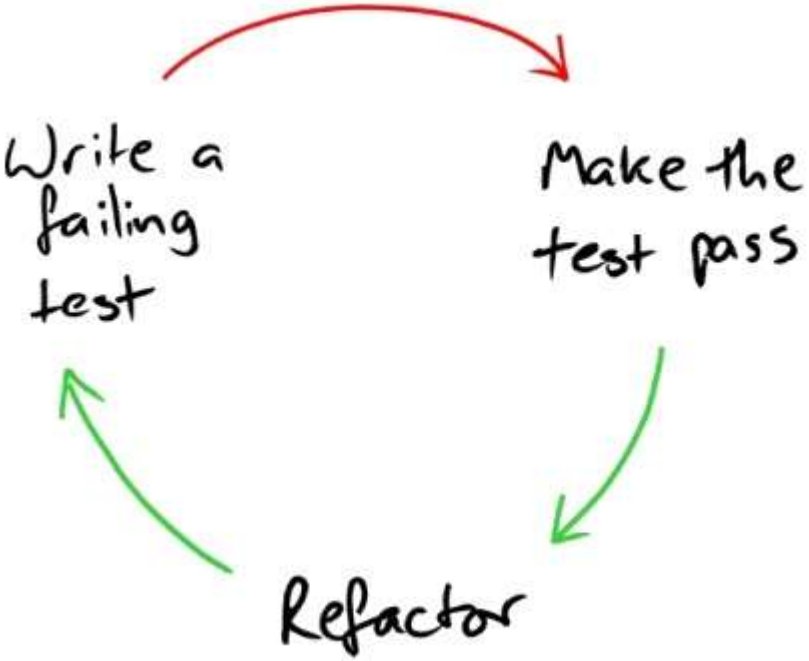
Test Driven Development



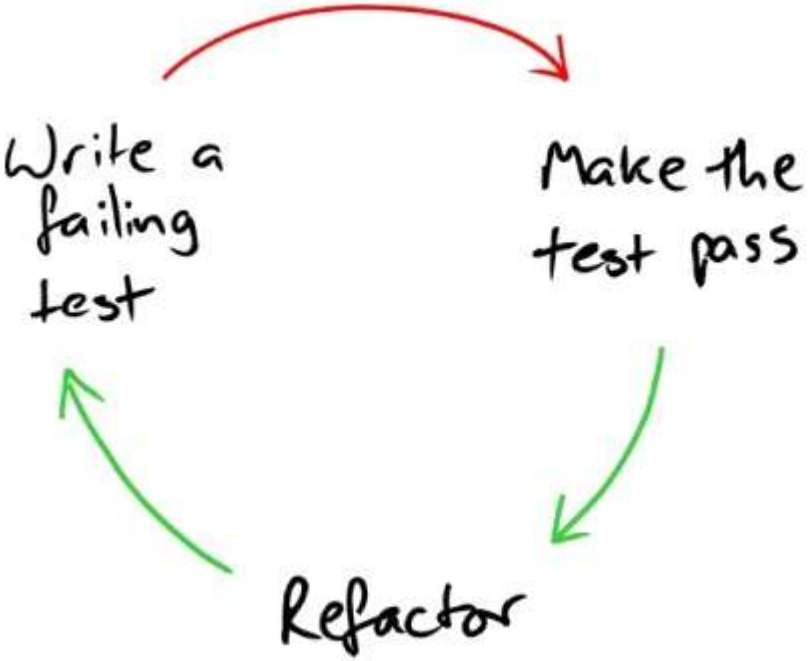
Test Driven Development



Test Driven Development



Test Driven Development



Red - Green – Refactor: the TDD Mantra
No new functionality without a failing test
No refactoring without passing tests

Test Driven Development Example…

Test Driven Development Example…

Create a **StringCalculator** class
with an **add** method
which takes a comma separate String of numbers
and returns their sum.

Example input	Result
	0
1	1
2	2
1,2	3
1,2,100	103

StringCalculatorTest.java

```java
import org.junit.Test;

public class StringCalculatorTest {

    @Test
    public void add_returns_0_for_empty_String() {

    }

}
```

4: Run 6: TODO Event Log

Tests Passed: 2 passed (6 minutes ago) 9:9 UTF-8 Insert 158M of 494M

StringCalculatorTest.java

```java
import org.junit.Test;

public class StringCalculatorTest {

    @Test
    public void add_returns_0_for_empty_String() {
        StringCalculator calculator = new StringCalculator();
        calculator.add("");
    }

}
```

4: Run 6: TODO Event Log

Tests Passed: 2 passed (9 minutes ago) 10:26 UTF-8 Insert 170M of 494M

StringCalculatorTest.java

```java
import ...

public class StringCalculatorTest {

    @Test
    public void add_returns_0_for_empty_String() {
        StringCalculator calculator = new StringCalculator();
        assertThat(calculator.add("")).isEqualTo(0);
    }

}
```

4: Run 6: TODO Event Log

Tests Passed: 2 passed (4 minutes ago) 15:2 UTF-8 Insert 121M of 494M

StringCalculatorTest.java

```java
import ...

public class StringCalculatorTest {

    @Test
    public void add_returns_0_for_empty_String() {
        StringCalculator calculator = new StringCalculator();
        assertThat(calculator.add("")).isEqualTo(0);
    }

}
```

StringCalculator.java

```java
public class StringCalculator {

    public Integer add(String numString) {
        return null;
    }
}
```

4: Run 6: TODO Event Log

Tests Failed: 0 passed, 1 failed (moments ago) 7:29 UTF-8 Insert 132M of 494M

StringCalculatorTest.java

```
1
2
3   import ...
6
7   public class StringCalculatorTest {
8
9       @Test
10      public void add_returns_0_for_empty_String() {
11          StringCalculator calculator = new StringCalculator();
12          assertThat(calculator.add("")).isEqualTo(0);
13      }
14
15  }
```

StringCalculator.java

```
1
2
3   public class StringCalculator {
4
5       public Integer add(String numString) {
6           return null;
7       }
8   }
9
```

Run StringCalculatorTest

Done: 1 of 1 Failed: 1 (0.028 s)

StringCalculatorTest (com.blackrock.tar.testing)
add_returns_0_for_empty_String

```
"C:\Program Files\Java\jdk1.7.0_11\bin\java" -ea -Didea.launcher.port=7534 "-Didea.launc

java.lang.AssertionError: expecting actual not to be null
    at com.blackrock.tar.testing.StringCalculatorTest.add_returns_0_for_empty_String(Str
```

4: Run 6: TODO Event Log

Tests Failed: 0 passed, 1 failed (a minute ago) 7:29 UTF-8 Insert 114M of 494M

StringCalculatorTest.java

```java
import ...

public class StringCalculatorTest {

    @Test
    public void add_returns_0_for_empty_String() {
        StringCalculator calculator = new StringCalculator();
        assertThat(calculator.add("")).isEqualTo(0);
    }

}
```

StringCalculator.java

```java
public class StringCalculator {

    public Integer add(String numString) {
        return 0;
    }
}
```

Run StringCalculatorTest

Done: 1 of 1 (0.02 s)

StringCalculatorTest (com.blackrock.tar.testing)
add_returns_0_for_empty_String

```
"C:\Program Files\Java\jdk1.7.0_11\bin\java" -ea -Didea.launcher.port=7533 "-Didea.launc

Process finished with exit code 0
```

4: Run 6: TODO Event Log

Tests Passed: 1 passed (moments ago) 6:17 UTF-8 Insert 193M of 494M

StringCalculatorTest.java

```java
import ...

public class StringCalculatorTest {

    @Test
    public void add_returns_0_for_empty_String() {
        StringCalculator calculator = new StringCalculator();
        assertThat(calculator.add("")).isEqualTo(0);
    }

}
```

StringCalculator.java

```java
public class StringCalculator {

    public Integer add(String numString) {
        return 0;
    }
}
```

Run StringCalculatorTest

Done: 1 of 1 (0.02 s)

StringCalculatorTest (com.blackrock.tar.testing)
add_returns_0_for_empty_String

```
"C:\Program Files\Java\jdk1.7.0_11\bin\java" -ea -Didea.launcher.port=7533 "-Didea.launc

Process finished with exit code 0
```

4: Run 6: TODO Event Log

Tests Passed: 1 passed (moments ago) 6:17 UTF-8 Insert 193M of 494M

StringCalculatorTest.java

```java
import ...

public class StringCalculatorTest {

    @Test
    public void add_returns_0_for_empty_String() {
        StringCalculator calculator = new StringCalculator();
        assertThat(calculator.add("")).isEqualTo(0);
    }

    @Test
    public void add_returns_integer_value_of_single_number_String() {
        StringCalculator calculator = new StringCalculator();
        assertThat(calculator.add("1")).isEqualTo(1);
    }

}
```

StringCalculator.java

```java
public class StringCalculator {

    public Integer add(String numString) {
        return 0;
    }
}
```

Tests Passed: 1 passed (6 minutes ago)

StringCalculatorTest.java

```java
import ...

public class StringCalculatorTest {

    @Test
    public void add_returns_0_for_empty_String() {
        StringCalculator calculator = new StringCalculator();
        assertThat(calculator.add("")).isEqualTo(0);
    }

    @Test
    public void add_returns_integer_value_of_single_number_String() {
        StringCalculator calculator = new StringCalculator();
        assertThat(calculator.add("1")).isEqualTo(1);
    }

}
```

StringCalculator.java

```java
public class StringCalculator {

    public Integer add(String numString) {
        return 0;
    }
}
```

Run StringCalculatorTest

Done: 2 of 2 Failed: 1 (0.054 s)

StringCalculatorTest (com.blackrock.tar.testing)
- add_returns_integer_value_of_single_numbe
- add_returns_0_for_empty_String

```
"C:\Program Files\Java\jdk1.7.0_11\bin\java" -ea -Didea.launcher.port=7534 "-Didea.launcl

org.junit.ComparisonFailure:
Expected :1
Actual   :0
```

Tests Failed: 1 passed, 1 failed (moments ago)

StringCalculatorTest.java

```java
import ...

public class StringCalculatorTest {

    @Test
    public void add_returns_0_for_empty_String() {
        StringCalculator calculator = new StringCalculator();
        assertThat(calculator.add("")).isEqualTo(0);
    }

    @Test
    public void add_returns_integer_value_of_single_number_String() {
        StringCalculator calculator = new StringCalculator();
        assertThat(calculator.add("1")).isEqualTo(1);
    }

}
```

StringCalculator.java

```java
public class StringCalculator {

    public Integer add(String numString) {
        if (numString.isEmpty()) return 0;
        return 1;
    }
}
```

Tests Failed: 1 passed, 1 failed (2 minutes ago)

StringCalculatorTest.java

```java
import ...

public class StringCalculatorTest {

    @Test
    public void add_returns_0_for_empty_String() {
        StringCalculator calculator = new StringCalculator();
        assertThat(calculator.add("")).isEqualTo(0);
    }

    @Test
    public void add_returns_integer_value_of_single_number_String() {
        StringCalculator calculator = new StringCalculator();
        assertThat(calculator.add("1")).isEqualTo(1);
    }

}
```

StringCalculator.java

```java
public class StringCalculator {

    public Integer add(String numString) {
        if (numString.isEmpty()) return 0;
        return 1;
    }
}
```

Run StringCalculatorTest

Done: 2 of 2 (0.02 s)

StringCalculatorTest (com.blackrock.tar.testing)
- add_returns_integer_value_of_single_numbe
- add_returns_0_for_empty_String

```
"C:\Program Files\Java\jdk1.7.0_11\bin\java" -ea -Didea.launcher.port=7533 "-Didea.launcl

Process finished with exit code 0
```

4: Run 6: TODO Event Log

Tests Passed: 2 passed (moments ago) 7:17 UTF-8 Insert 147M of 494M

StringCalculatorTest.java

```java
import ...

public class StringCalculatorTest {

    @Test
    public void add_returns_0_for_empty_String() {
        StringCalculator calculator = new StringCalculator();
        assertThat(calculator.add("")).isEqualTo(0);
    }

    @Test
    public void add_returns_integer_value_of_single_number_String() {
        StringCalculator calculator = new StringCalculator();
        assertThat(calculator.add("1")).isEqualTo(1);
    }

}
```

StringCalculator.java

```java
public class StringCalculator {

    public Integer add(String numString) {
        if (numString.isEmpty()) return 0;
        return 1;
    }
}
```

Refactor?

Run StringCalculatorTest

Done: 2 of 2 (0.02 s)

StringCalculatorTest (com.blackrock.tar.testing)
- add_returns_integer_value_of_single_numbe
- add_returns_0_for_empty_String

```
"C:\Program Files\Java\jdk1.7.0_11\bin\java" -ea -Didea.launcher.port=7533 "-Didea.launcl

Process finished with exit code 0
```

4: Run 6: TODO Event Log

Tests Passed: 2 passed (moments ago) 7:17 UTF-8 Insert 147M of 494M

StringCalculatorTest.java

```java
import ...

public class StringCalculatorTest {

    StringCalculator calculator;

    @Before
    public void setup() {
        calculator = new StringCalculator();
    }

    @Test
    public void add_returns_0_for_empty_String() {
        StringCalculator calculator = new StringCalculator();
        assertThat(calculator.add("")).isEqualTo(0);
    }

    @Test
    public void add_returns_integer_value_of_single_number_String() {
        StringCalculator calculator = new StringCalculator();
        assertThat(calculator.add("1")).isEqualTo(1);
    }

}
```

StringCalculator.java

```java
public class StringCalculator {

    public Integer add(String numString) {
        if (numString.isEmpty()) return 0;
        return 1;
    }
}
```

Tests Passed: 2 passed (3 minutes ago)

StringCalculatorTest.java

```java
import ...

public class StringCalculatorTest {

    StringCalculator calculator;

    @Before
    public void setup() {
        calculator = new StringCalculator();
    }

    @Test
    public void add_returns_0_for_empty_String() {
        assertThat(calculator.add("")).isEqualTo(0);
    }

    @Test
    public void add_returns_integer_value_of_single_number_String() {
        assertThat(calculator.add("1")).isEqualTo(1);
    }

}
```

StringCalculator.java

```java
public class StringCalculator {

    public Integer add(String numString) {
        if (numString.isEmpty()) return 0;
        return 1;
    }
}
```

Tests Passed: 2 passed (4 minutes ago)

StringCalculatorTest.java

```java
import ...

public class StringCalculatorTest {

    StringCalculator calculator;

    @Before
    public void setup() {
        calculator = new StringCalculator();
    }

    @Test
    public void add_returns_0_for_empty_String() {
        assertThat(calculator.add("")).isEqualTo(0);
    }

    @Test
    public void add_returns_integer_value_of_single_number_String() {
        assertThat(calculator.add("1")).isEqualTo(1);
    }

}
```

StringCalculator.java

```java
public class StringCalculator {

    public Integer add(String numString) {
        if (numString.isEmpty()) return 0;
        return 1;
    }
}
```

Run StringCalculatorTest

Done: 2 of 2 (0.017 s)

- StringCalculatorTest (com.blackrock.tar.testing)
  - add_returns_integer_value_of_single_numbe
  - add_returns_0_for_empty_String

```
"C:\Program Files\Java\jdk1.7.0_11\bin\java" -ea -Didea.launcher.port=7534 "-Didea.launcl

Process finished with exit code 0
```

Tests Passed: 2 passed (moments ago)

StringCalculatorTest.java

```java
import ...

public class StringCalculatorTest {

    StringCalculator calculator;

    @Before
    public void setup() {
        calculator = new StringCalculator();
    }

    @Test
    public void add_returns_0_for_empty_String() {
        assertThat(calculator.add("")).isEqualTo(0);
    }

    @Test
    public void add_returns_integer_value_of_single_number_String() {
        assertThat(calculator.add("1")).isEqualTo(1);
        assertThat(calculator.add("2")).isEqualTo(2);
    }

}
```

StringCalculator.java

```java
public class StringCalculator {

    public Integer add(String numString) {
        if (numString.isEmpty()) return 0;
        return 1;
    }
}
```

4: Run 6: TODO Event Log

Tests Passed: 2 passed (2 minutes ago) 25:37 UTF-8 Insert 103M of 494M

StringCalculatorTest.java

```java
import ...

public class StringCalculatorTest {

    StringCalculator calculator;

    @Before
    public void setup() {
        calculator = new StringCalculator();
    }

    @Test
    public void add_returns_0_for_empty_String() {
        assertThat(calculator.add("")).isEqualTo(0);
    }

    @Test
    public void add_returns_integer_value_of_single_number_String() {
        assertThat(calculator.add("1")).isEqualTo(1);
        assertThat(calculator.add("2")).isEqualTo(2);
    }

}
```

StringCalculator.java

```java
public class StringCalculator {

    public Integer add(String numString) {
        if (numString.isEmpty()) return 0;
        return 1;
    }
}
```

Run StringCalculatorTest

Done: 2 of 2 Failed: 1 (0.029 s)

- StringCalculatorTest (com.blackrock.tar.testing)
  - add_returns_integer_value_of_single_numbe
  - add_returns_0_for_empty_String

```
"C:\Program Files\Java\jdk1.7.0_11\bin\java" -ea -Didea.launcher.port=7533 "-Didea.launcl

org.junit.ComparisonFailure:
Expected :2
Actual   :1
```

Tests Failed: 1 passed, 1 failed (moments ago) | 25:37 | UTF-8 | Insert | 151M of 494M

StringCalculatorTest.java

```java
import ...

public class StringCalculatorTest {

    StringCalculator calculator;

    @Before
    public void setup() {
        calculator = new StringCalculator();
    }

    @Test
    public void add_returns_0_for_empty_String() {
        assertThat(calculator.add("")).isEqualTo(0);
    }

    @Test
    public void add_returns_integer_value_of_single_number_String() {
        assertThat(calculator.add("1")).isEqualTo(1);
        assertThat(calculator.add("2")).isEqualTo(2);
    }

}
```

StringCalculator.java

```java
public class StringCalculator {

    public Integer add(String numString) {
        if (numString.isEmpty()) return 0;
        return Integer.parseInt(numString);
    }
}
```

Tests Failed: 1 passed, 1 failed (a minute ago)

StringCalculatorTest.java

```java
import ...

public class StringCalculatorTest {

    StringCalculator calculator;

    @Before
    public void setup() {
        calculator = new StringCalculator();
    }

    @Test
    public void add_returns_0_for_empty_String() {
        assertThat(calculator.add("")).isEqualTo(0);
    }

    @Test
    public void add_returns_integer_value_of_single_number_String() {
        assertThat(calculator.add("1")).isEqualTo(1);
        assertThat(calculator.add("2")).isEqualTo(2);
    }

}
```

StringCalculator.java

```java
public class StringCalculator {

    public Integer add(String numString) {
        if (numString.isEmpty()) return 0;
        return Integer.parseInt(numString);
    }
}
```

Run StringCalculatorTest

Done: 2 of 2 (0.017 s)

StringCalculatorTest (com.blackrock.tar.testing)
- add_returns_0_for_empty_String
- add_returns_integer_value_of_single_numbe

```
"C:\Program Files\Java\jdk1.7.0_11\bin\java" -ea -Didea.launcher.port=7535 "-Didea.launcl

Process finished with exit code 0
```

4: Run 6: TODO Event Log

Tests Passed: 2 passed (moments ago) 7:42 UTF-8 Insert 99M of 494M

StringCalculatorTest.java

```java
import ...

public class StringCalculatorTest {

    StringCalculator calculator;

    @Before
    public void setup() {
        calculator = new StringCalculator();
    }

    @Test
    public void add_returns_0_for_empty_String() {
        checkAdd("", 0);
    }

    @Test
    public void add_returns_integer_value_of_single_number_String() {
        checkAdd("1", 1);
        checkAdd("1234", 1234);
    }

    @Test
    public void add_returns_integer_sum_of_multi_number_String() {
        checkAdd("1,2", 3);
    }

    private void checkAdd(String numString, int num) {
        assertThat(calculator.add(numString)).isEqualTo(num);
    }
}
```

StringCalculator.java

```java
public class StringCalculator {

    public Integer add(String numString) {
        if (numString.isEmpty()) return 0;
        String[] numbers = numString.split(",");
        return sum(numbers);
    }

    private int sum(String[] numbers) {
        int sum = 0;
        for (String number : numbers) {
            sum += Integer.parseInt(number);
        }
        return sum;
    }
}
```

Run StringCalculatorTest

Done: 3 of 3 (0.019 s)

- StringCalculatorTest (com.blackrock.tar.testing)
  - add_returns_0_for_empty_String
  - add_returns_integer_value_of_single_numbe
  - add_returns_integer_sum_of_multi_number_

```
"C:\Program Files\Java\jdk1.7.0_11\bin\java" -ea -Didea.launcher.port=7536 "-Didea.launcl

Process finished with exit code 0
```

4: Run 6: TODO Event Log

Tests Passed: 3 passed (moments ago) 18:2 UTF-8 Insert 113M of 494M

StringCalculatorTest.java

```java
import ...

public class StringCalculatorTest {

    StringCalculator calculator;

    @Before
    public void setup() {
        calculator = new StringCalculator();
    }

    @Test
    public void add_returns_0_for_empty_String() {
        checkAdd("", 0);
    }

    @Test
    public void add_returns_integer_value_of_single_number_String() {
        checkAdd("1", 1);
        checkAdd("1234", 1234);
    }

    @Test
    public void add_returns_integer_sum_of_multi_number_String() {
        checkAdd("1,2", 3);
    }

    private void checkAdd(String numString, int num) {
        assertThat(calculator.add(numString)).isEqualTo(num);
    }
}
```

Structure

- StringCalculatorTest
  - setup():void
  - add_returns_0_for_empty_String():void
  - add_returns_integer_value_of_single_number_String():void
  - add_returns_integer_sum_of_multi_number_String():void

These tests act like the original developer looking over your shoulder and advising you, long after that developer has left…

Tests Passed: 3 passed (6 minutes ago)

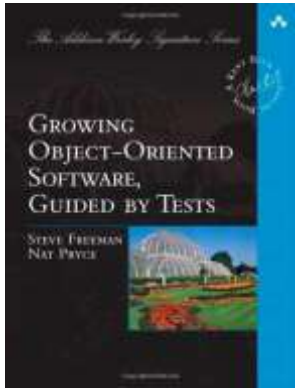
Recommended reading

Test Driven Development

Kent Beck

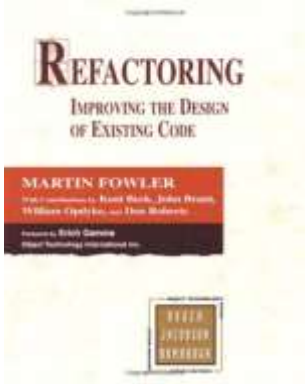
Growing Object-Oriented Software, Guided by Tests

Freeman & Pryce



Refactoring: Improving the Design of Existing Code

Martin Fowler, Kent Beck et. al.



Effective Unit Testing

Lasse Koskela

Recommended reading

Unit Test Best Practices -
https://wiki.tlcinternal.com/display/TD/Unit+Test+Best+Practices

Mocks Arent Stubs -
http://www.martinfowler.com/articles/mocksArentStubs.html

Questions?