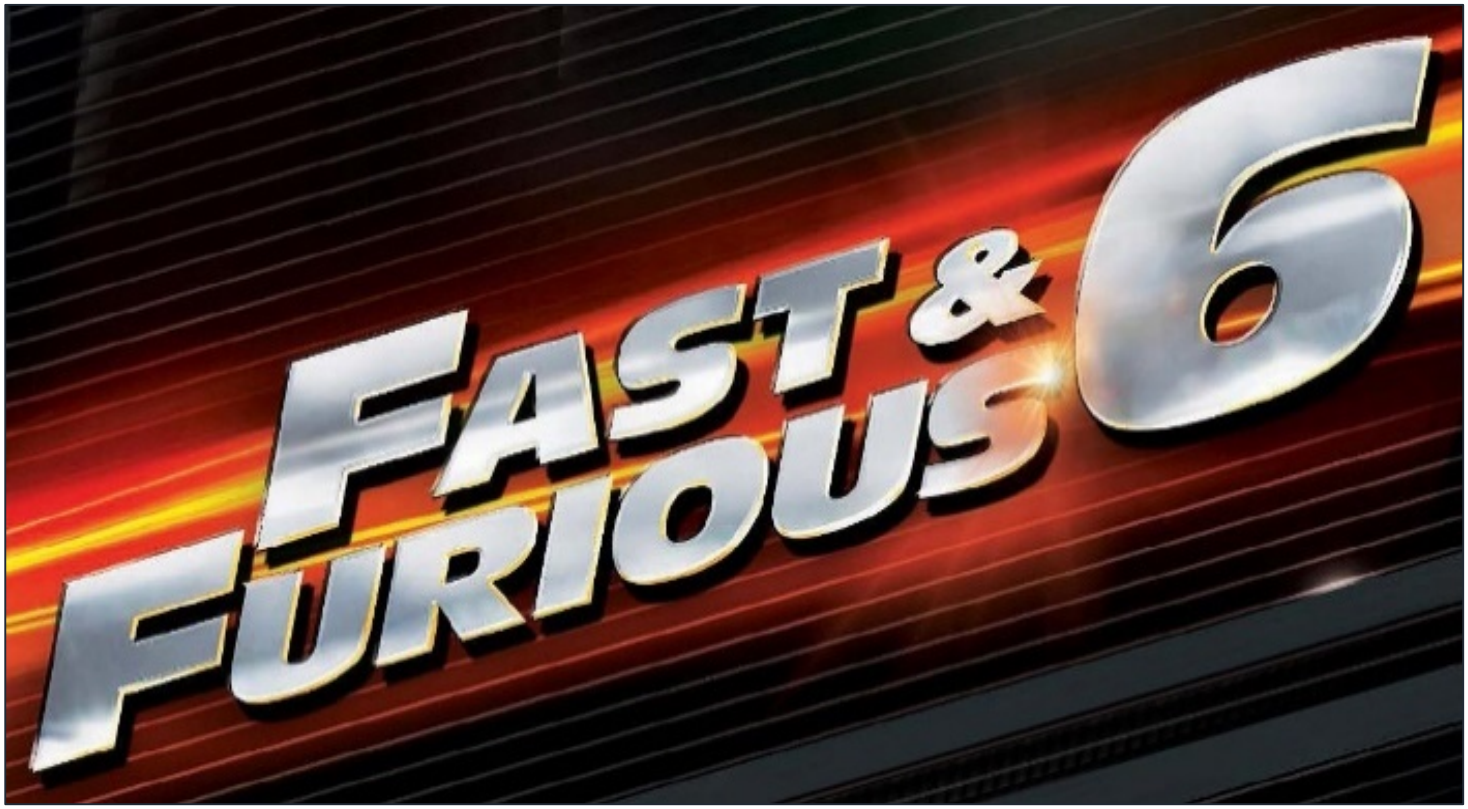
Increase your capacity
to deliver more value

UNIT TESTING

Aims at creating a system that is easy
to maintain and extend over time

FAST FEED BACK

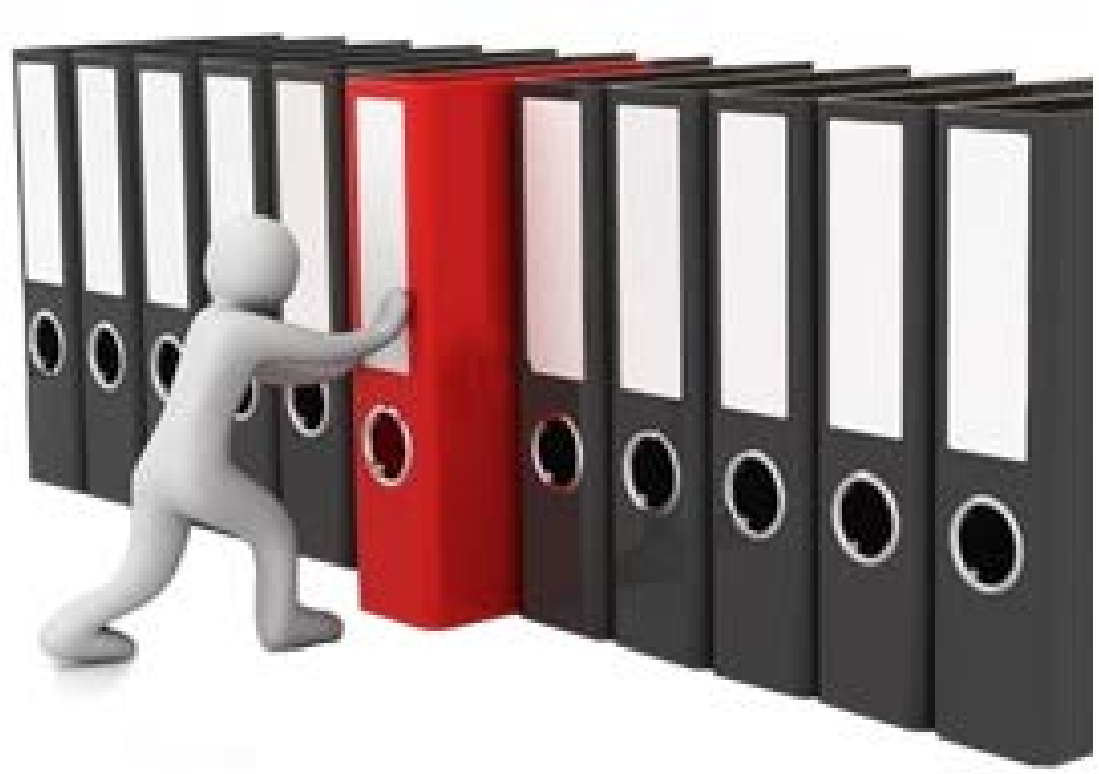
Run in IDE and Software factories as well



42skillz

BEST ENTRY POINT FOR NEWCOMMER

Like a live documentation



42skillz

SAFETY NET

No fear to break something

42skillz

Fast *many hundreds or thousands tests per second*

Isolates *failure*

reasons become obvious

FIRST properties ...

Repeatable *run repeatedly in any order, any time*

Self-validating

no manual evaluation required

Timely Fashion

written before the code

THE SECRET ? DO NOT TEST …

~~METHODS~~

Arrange all necessary preconditions and inputs

3 A's-rule

Act on the object or method under test

Assert that the expected results have occurred

3 A'S RULE

In real life no comment is need

```
[Test]
public static void Should_return_Statement_when_rental_one_regular_movie_during_less_than_2_days()
{
    // Arrange
    var customerName = “Thomas”;
    var movieName = "Dracula Untold“;
    var customer = MakeCustomerWithRental(movieName, KindOfMovie.Regular, 1, customerName);

    // Act
    var actual = customer.Statement();

    // Assert
    Check.That(actual)
      .Equals(string.Format(
      "Rental Record for {0}\n\t{1}\t{2}\nAmount owed is {2}\nYou earned 1 frequent renter points",
      customerName, movieName, amount));
}
```

FAST FEED BACK

Remove external dependencies

```
public class AlarmShould
{
  [Test]
  public void Raise_alarm_when_pressure_is_under_lowPressureTreshold()
  {
     var sensor = Substitute.For<ISensor>();
     sensor.PopNextPressurePsiValue().Returns(10);
     var alarm = new Alarm(sensor);

     alarm.Check();

     Check.That(alarm.AlarmOn).IsTrue();
  }
}
```

REVEAL BEHAVIORS

Each test must give a clear intention

```
public class TripServiceTests
{
  [Test]
  public void Shoud_raise_exception_when_logged_user_is_not_connected()
  {
    ...
  }
}
```

```
public class TripServiceShould
{
  @Test
  public void raise_exception_when_logged_user_is_not_connected()
  {
    ...
  }
}
```

UNIT TEST STRUCTURE

```
public class TripServiceShould
 {
    [TestFixtureSetUp]
    public void FixtureSetUp()
    { … }

    [SetUp]
    public void SetUp()
    { … }

    [Test]
    public void do_when_conditions()
    { … }

    [TearDown]
    public void TearDown()
    { … }

    [TestFixtureTearDown]
    public void FixtureTearDown()
    { … }
}
```


```
public class TripServiceShould
 {
    @BeforeClass
    public void fixtureSetUp()
    { … }

    @Before
    public void setUp()
    { … }

    @Test
    public void do_when_conditions()
    { … }

    @After
    public void tearDown()
    { … }

    @AfterClass
    public void fixtureTearDown()
    { … }
}
```

TO SUM-UP

Each unit test

- ✓ Must respect **FIRST** principles
- ✓ Must follow clean code principles

  as your production code
- ✓ Must **reflect your intention**

  Technical or business

  Test a behaviors not methods
- ✓ Must be readable **as human language**