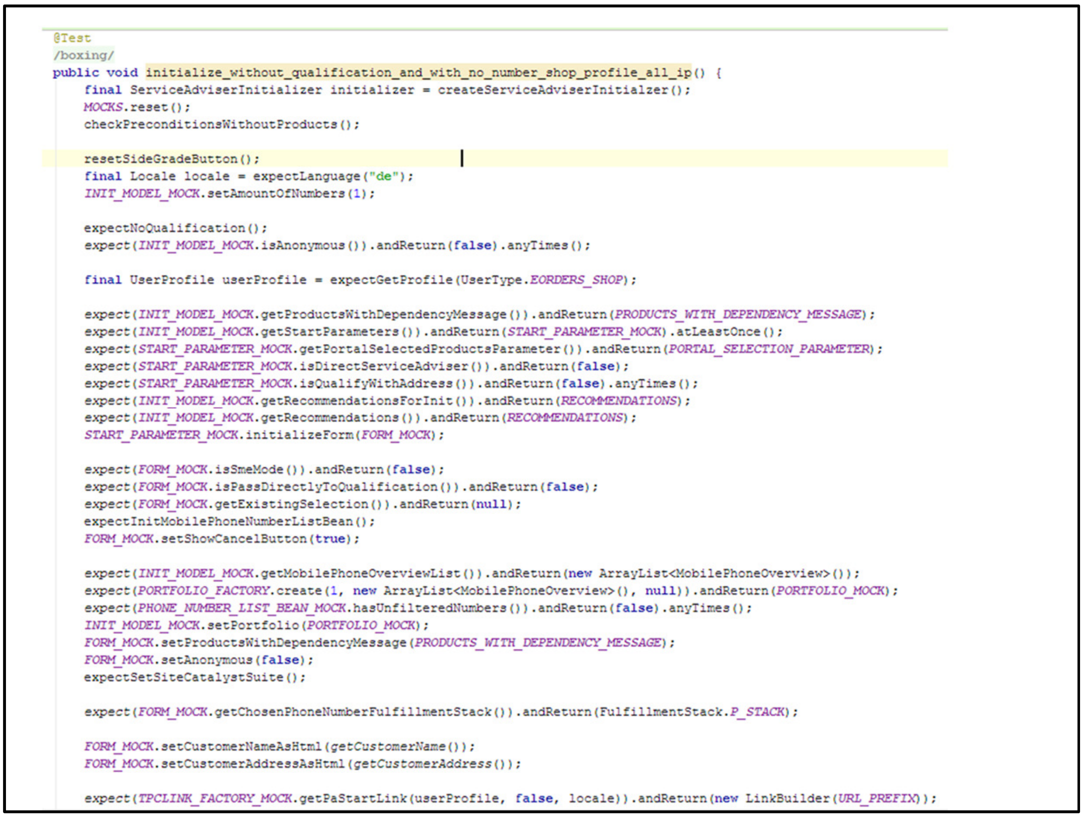
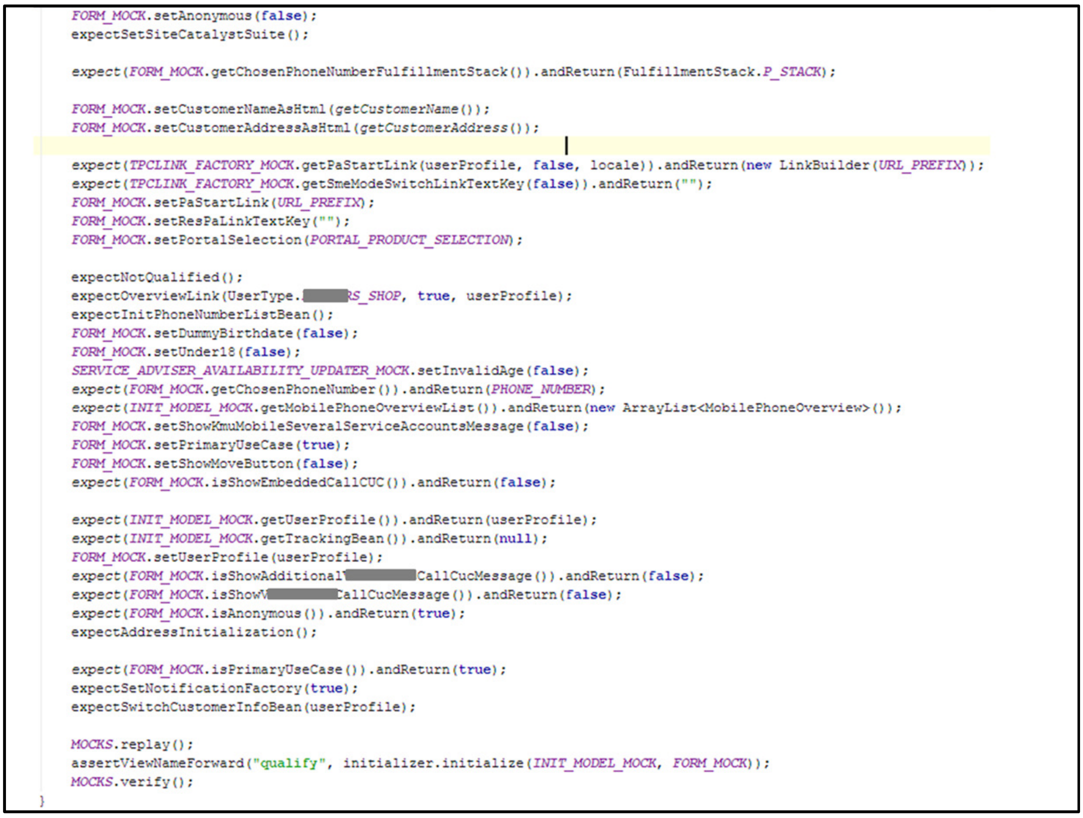
Stop mocking and get real

Rick.Janda@zuehlke.com

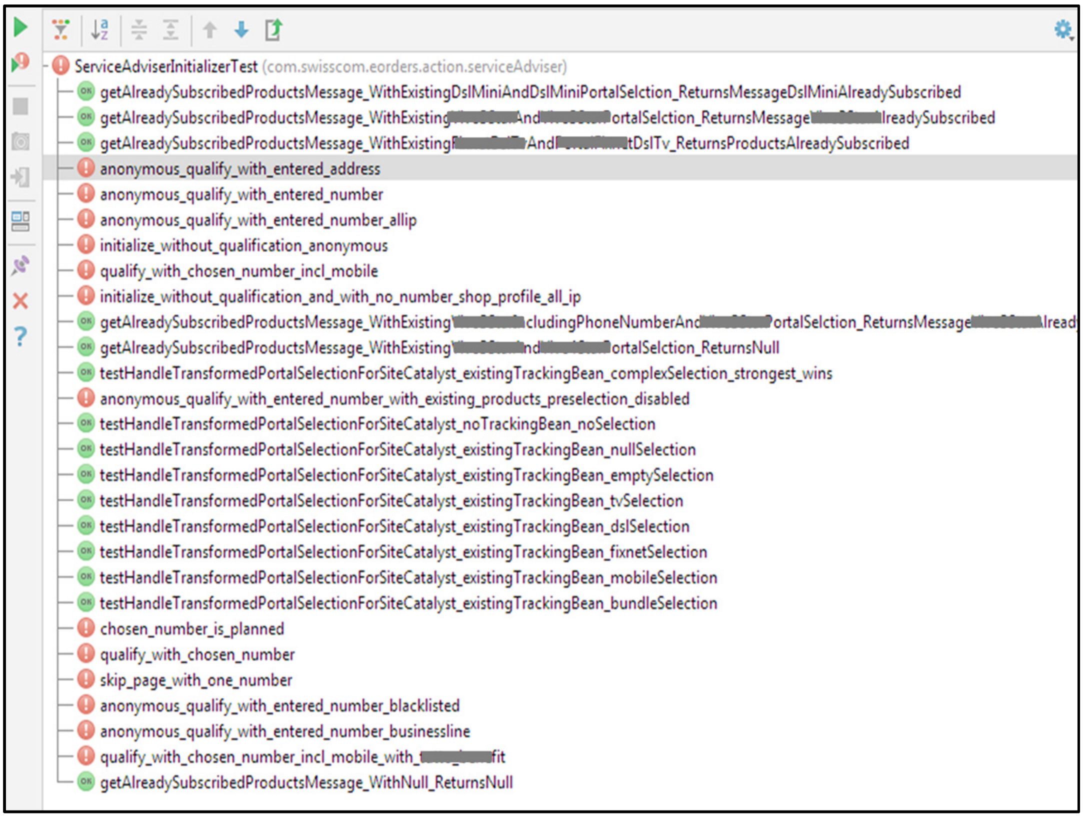


```
@Test
/boxing/
public void initialize_without_qualification_and_with_no_number_shop_profile_all_ip() {
    final ServiceAdviserInitializer initializer = createServiceAdviserInitialzer();
    MOCKS.reset();
    checkPreconditionsWithoutProducts();

    resetSideGradeButton();
    final Locale locale = expectLanguage("de");
    INIT_MODEL_MOCK.setAmountOfNumbers(1);

    expectNoQualification();
    expect(INIT_MODEL_MOCK.isAnonymous()).andReturn(false).anyTimes();

    final UserProfile userProfile = expectGetProfile(UserType.EORDERS_SHOP);

    expect(INIT_MODEL_MOCK.getProductsWithDependencyMessage()).andReturn(PRODUCTS_WITH_DEPENDENCY_MESSAGE);
    expect(INIT_MODEL_MOCK.getStartParameters()).andReturn(START_PARAMETER_MOCK).atLeastOnce();
    expect(START_PARAMETER_MOCK.getPortalSelectedProductsParameter()).andReturn(PORTAL_SELECTION_PARAMETER);
    expect(START_PARAMETER_MOCK.isDirectServiceAdviser()).andReturn(false);
    expect(START_PARAMETER_MOCK.isQualifyWithAddress()).andReturn(false).anyTimes();
    expect(INIT_MODEL_MOCK.getRecommendationsForInit()).andReturn(RECOMMENDATIONS);
    expect(INIT_MODEL_MOCK.getRecommendations()).andReturn(RECOMMENDATIONS);
    START_PARAMETER_MOCK.initializeForm(FORM_MOCK);

    expect(FORM_MOCK.isSmeMode()).andReturn(false);
    expect(FORM_MOCK.isPassDirectlyToQualification()).andReturn(false);
    expect(FORM_MOCK.getExistingSelection()).andReturn(null);
    expectInitMobilePhoneNumberListBean();
    FORM_MOCK.setShowCancelButton(true);

    expect(INIT_MODEL_MOCK.getMobilePhoneOverviewList()).andReturn(new ArrayList<MobilePhoneOverview>());
    expect(PORTFOLIO_FACTORY.create(1, new ArrayList<MobilePhoneOverview>(), null)).andReturn(PORTFOLIO_MOCK);
    expect(PHONE_NUMBER_LIST_BEAN_MOCK.hasUnfilteredNumbers()).andReturn(false).anyTimes();
    INIT_MODEL_MOCK.setPortfolio(PORTFOLIO_MOCK);
    FORM_MOCK.setProductsWithDependencyMessage(PRODUCTS_WITH_DEPENDENCY_MESSAGE);
    FORM_MOCK.setAnonymous(false);
    expectSetSiteCatalystSuite();

    expect(FORM_MOCK.getChosenPhoneNumberFulfillmentStack()).andReturn(FulfillmentStack.P_STACK);

    FORM_MOCK.setCustomerNameAsHtml(getCustomerName());
    FORM_MOCK.setCustomerAddressAsHtml(getCustomerAddress());

    expect(TPCLINK_FACTORY_MOCK.getPaStartLink(userProfile, false, locale)).andReturn(new LinkBuilder(URL_PREFIX));
```

Who knows the mocking hell?



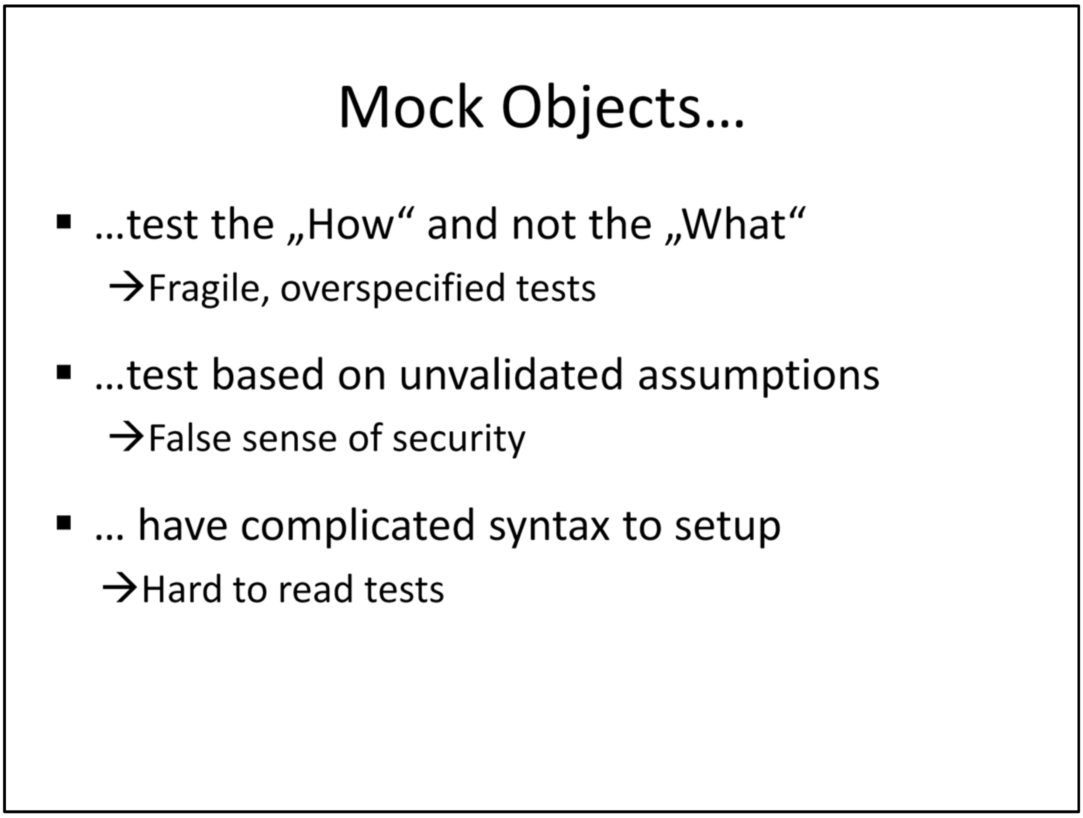
```java
    FORM_MOCK.setAnonymous(false);
    expectSetSiteCatalystSuite();

    expect(FORM_MOCK.getChosenPhoneNumberFulfillmentStack()).andReturn(FulfillmentStack.P_STACK);

    FORM_MOCK.setCustomerNameAsHtml(getCustomerName());
    FORM_MOCK.setCustomerAddressAsHtml(getCustomerAddress());

    expect(TPCLINK_FACTORY_MOCK.getPaStartLink(userProfile, false, locale)).andReturn(new LinkBuilder(URL_PREFIX));
    expect(TPCLINK_FACTORY_MOCK.getSmeModeSwitchLinkTextKey(false)).andReturn("");
    FORM_MOCK.setPaStartLink(URL_PREFIX);
    FORM_MOCK.setResPaLinkTextKey("");
    FORM_MOCK.setPortalSelection(PORTAL_PRODUCT_SELECTION);

    expectNotQualified();
    expectOverviewLink(UserType.[redacted]RS_SHOP, true, userProfile);
    expectInitPhoneNumberListBean();
    FORM_MOCK.setDummyBirthdate(false);
    FORM_MOCK.setUnder18(false);
    SERVICE_ADVISER_AVAILABILITY_UPDATER_MOCK.setInvalidAge(false);
    expect(FORM_MOCK.getChosenPhoneNumber()).andReturn(PHONE_NUMBER);
    expect(INIT_MODEL_MOCK.getMobilePhoneOverviewList()).andReturn(new ArrayList<MobilePhoneOverview>());
    FORM_MOCK.setShowKmuMobileSeveralServiceAccountsMessage(false);
    FORM_MOCK.setPrimaryUseCase(true);
    FORM_MOCK.setShowMoveButton(false);
    expect(FORM_MOCK.isShowEmbeddedCallCUC()).andReturn(false);

    expect(INIT_MODEL_MOCK.getUserProfile()).andReturn(userProfile);
    expect(INIT_MODEL_MOCK.getTrackingBean()).andReturn(null);
    FORM_MOCK.setUserProfile(userProfile);
    expect(FORM_MOCK.isShowAdditional[redacted]CallCucMessage()).andReturn(false);
    expect(FORM_MOCK.isShow[redacted]CallCucMessage()).andReturn(false);
    expect(FORM_MOCK.isAnonymous()).andReturn(true);
    expectAddressInitialization();

    expect(FORM_MOCK.isPrimaryUseCase()).andReturn(true);
    expectSetNotificationFactory(true);
    expectSwitchCustomerInfoBean(userProfile);

    MOCKS.replay();
    assertViewNameForward("qualify", initializer.initialize(INIT_MODEL_MOCK, FORM_MOCK));
    MOCKS.verify();
}
```



```
ServiceAdviserInitializerTest (com.swisscom.eorders.action.serviceAdviser)
  getAlreadySubscribedProductsMessage_WithExistingDslMiniAndDslMiniPortalSelction_ReturnsMessageDslMiniAlreadySubscribed
  getAlreadySubscribedProductsMessage_WithExisting[redacted]And[redacted]ortalSelction_ReturnsMessage[redacted]lreadySubscribed
  getAlreadySubscribedProductsMessage_WithExisting[redacted]And[redacted]tDslTv_ReturnsProductsAlreadySubscribed
  anonymous_qualify_with_entered_address
  anonymous_qualify_with_entered_number
  anonymous_qualify_with_entered_number_allip
  initialize_without_qualification_anonymous
  qualify_with_chosen_number_incl_mobile
  initialize_without_qualification_and_with_no_number_shop_profile_all_ip
  getAlreadySubscribedProductsMessage_WithExisting[redacted]cludingPhoneNumberAnd[redacted]PortalSelction_ReturnsMessage[redacted]lread
  getAlreadySubscribedProductsMessage_WithExisting[redacted]nd[redacted]ortalSelction_ReturnsNull
  testHandleTransformedPortalSelectionForSiteCatalyst_existingTrackingBean_complexSelection_strongest_wins
  anonymous_qualify_with_entered_number_with_existing_products_preselection_disabled
  testHandleTransformedPortalSelectionForSiteCatalyst_noTrackingBean_noSelection
  testHandleTransformedPortalSelectionForSiteCatalyst_existingTrackingBean_nullSelection
  testHandleTransformedPortalSelectionForSiteCatalyst_existingTrackingBean_emptySelection
  testHandleTransformedPortalSelectionForSiteCatalyst_existingTrackingBean_tvSelection
  testHandleTransformedPortalSelectionForSiteCatalyst_existingTrackingBean_dslSelection
  testHandleTransformedPortalSelectionForSiteCatalyst_existingTrackingBean_fixnetSelection
  testHandleTransformedPortalSelectionForSiteCatalyst_existingTrackingBean_mobileSelection
  testHandleTransformedPortalSelectionForSiteCatalyst_existingTrackingBean_bundleSelection
  chosen_number_is_planned
  qualify_with_chosen_number
  skip_page_with_one_number
  anonymous_qualify_with_entered_number_blacklisted
  anonymous_qualify_with_entered_number_businessline
  qualify_with_chosen_number_incl_mobile_with_[redacted]fit
  getAlreadySubscribedProductsMessage_WithNull_ReturnsNull
```

You don‘t care…
Until you have to make small code change…

What could be the problem with Mock Objects?



Mock Objects…

- …test the „How“ and not the „What“

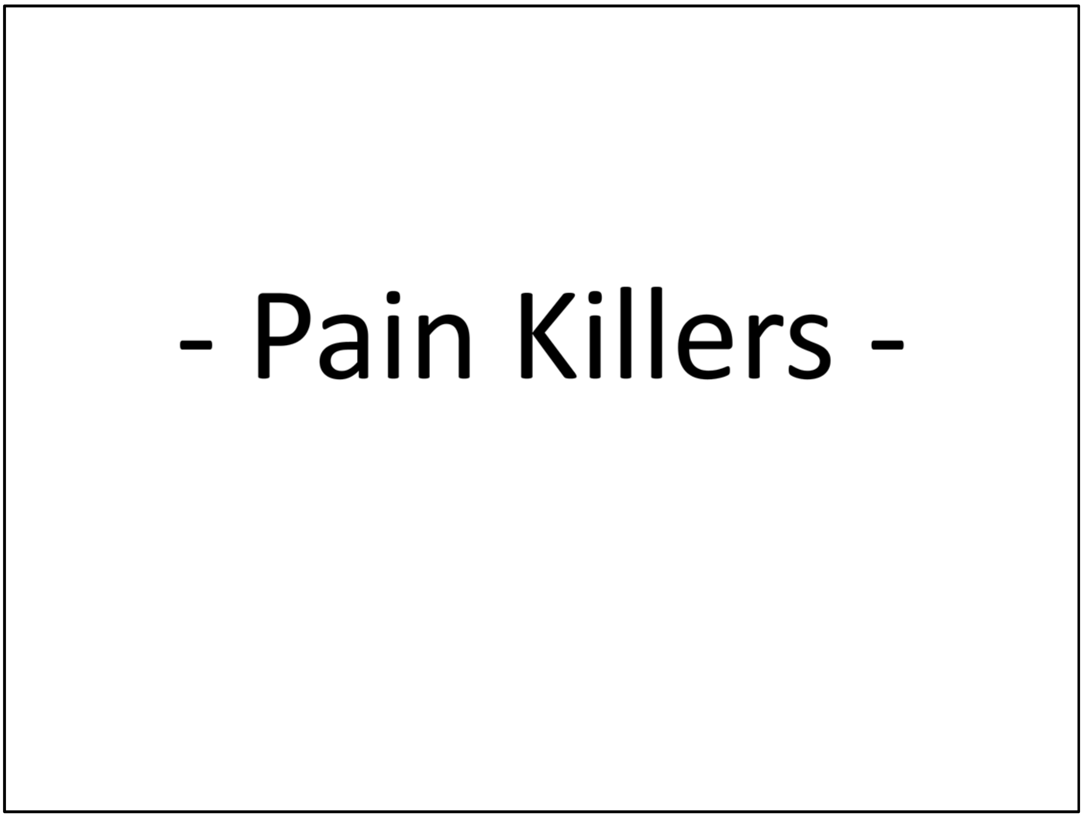
  →Fragile, overspecified tests

- …test based on unvalidated assumptions

  →False sense of security

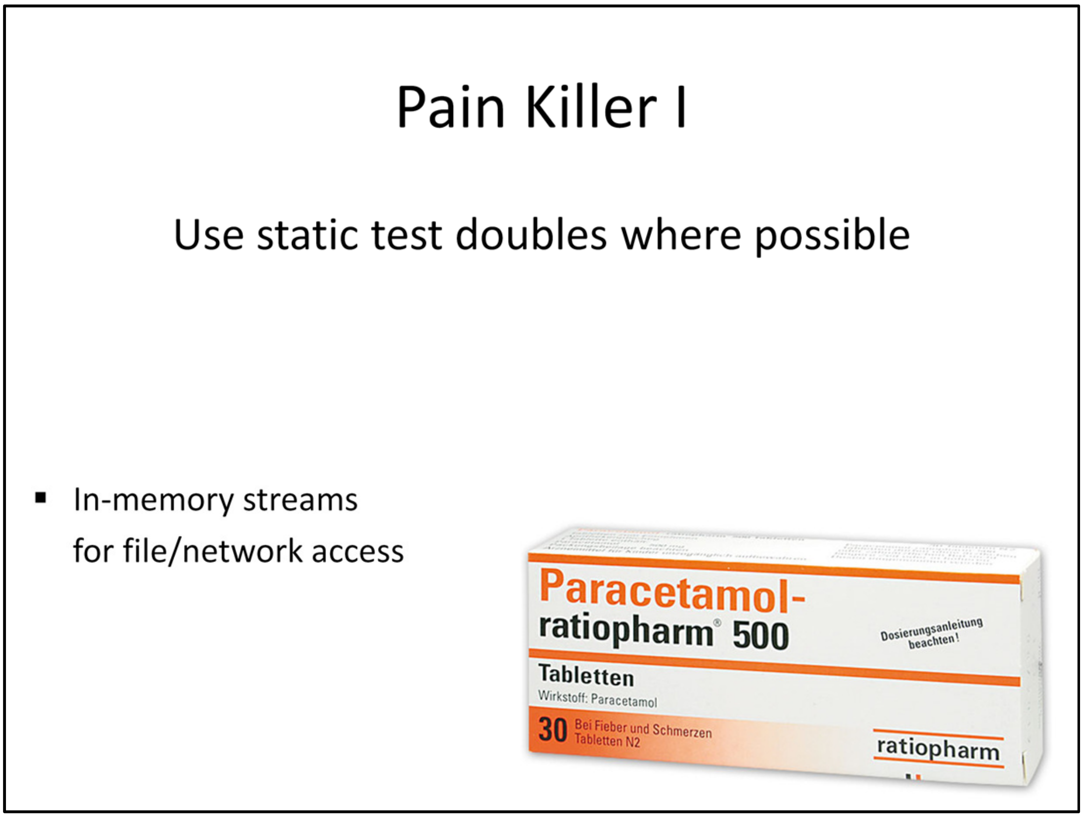
- … have complicated syntax to setup

  →Hard to read tests



- Pain Killers -

Changes on the test side, that make the tests less fragile and better readable.



Pain Killer I

Use static test doubles where possible

- In-memory streams for file/network access

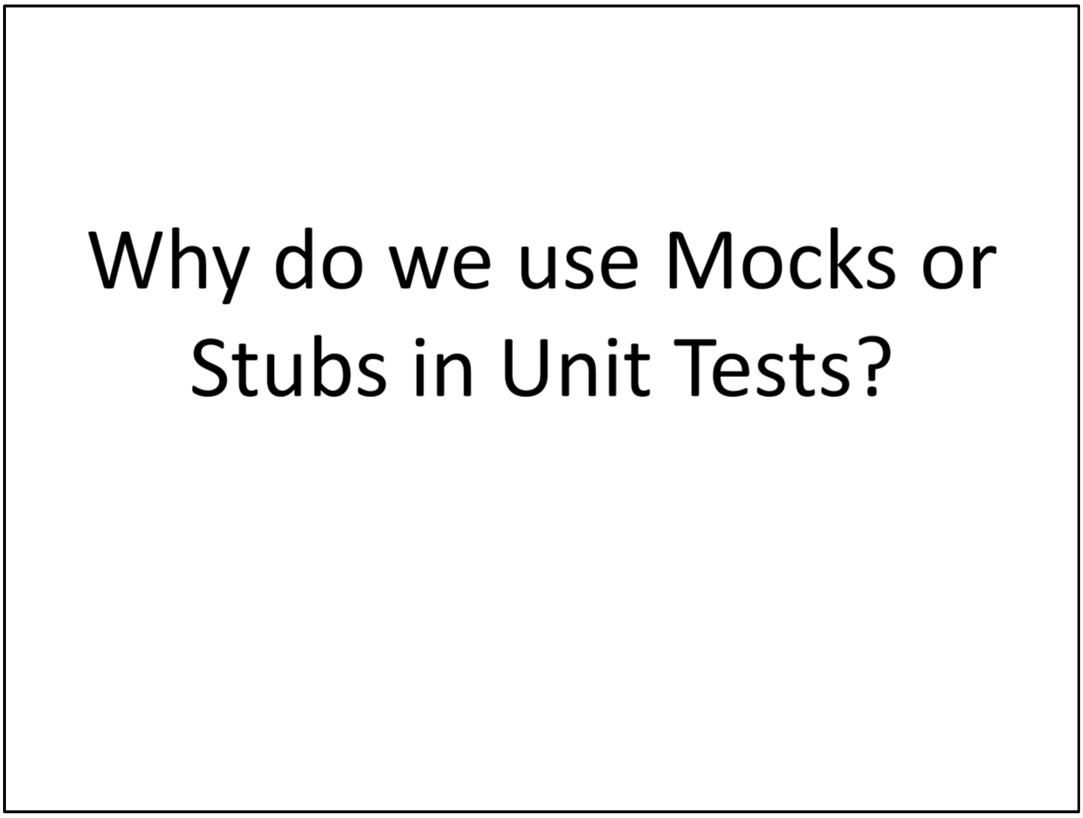
Examples:
- ByteArrayOutputStream ByteArrayInputStream for Streams and assert on the toString() result. → Allow changes in the how.
- CharArrayReader/-Writer for Reader/Writer
- MockedHttpServletRequest/-Response for Servlet API Interfaces (part of Spring Testing)

Pain Killer II

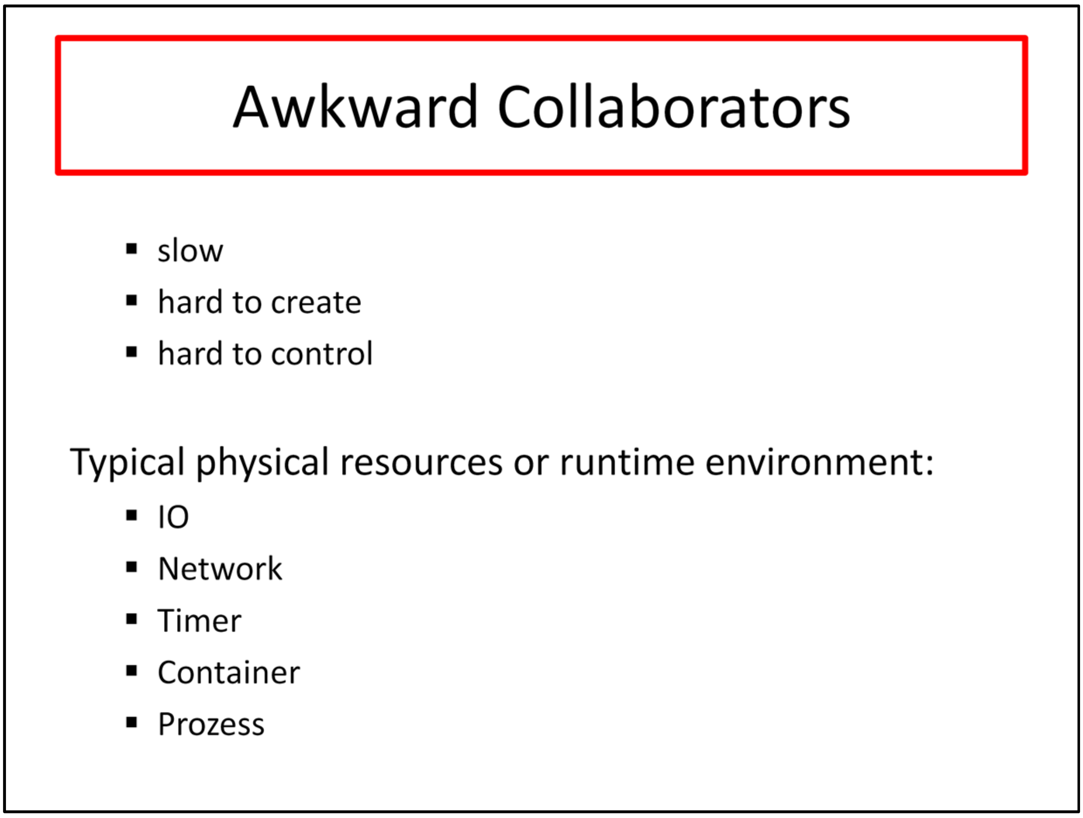
Make Unit Tests single responsible.

Bei Kopfschmerzen, Zahnschmerzen, Regelschmerzen
Ibuprofen 400
Wirkstoff: Ibuprofen 400 mg
20 Filmtabletten · N2
Ibuprofen 400
Sophien-Arzneimittel GmbH · www.sophien-arzneimittel.de

- Only one mock per test case



Why do we use Mocks or Stubs in Unit Tests?

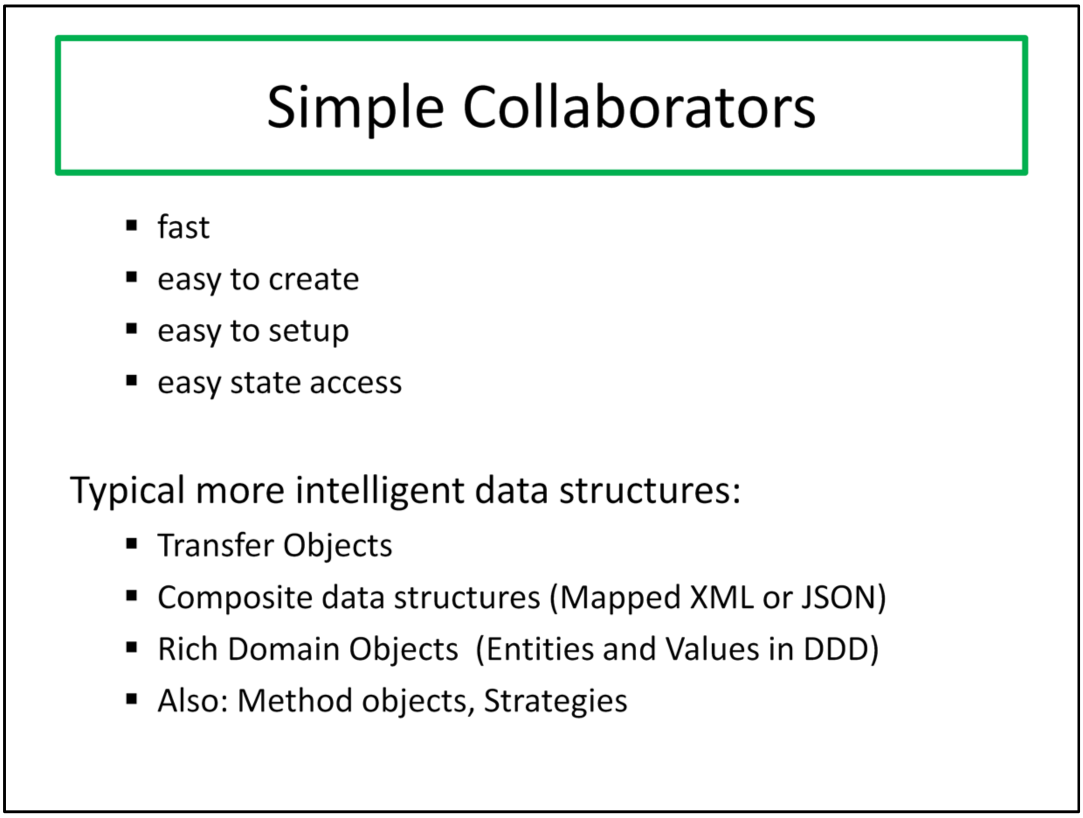


Awkward Collaborators

- slow
- hard to create
- hard to control

Typical physical resources or runtime environment:

- IO
- Network
- Timer
- Container
- Prozess



Simple Collaborators

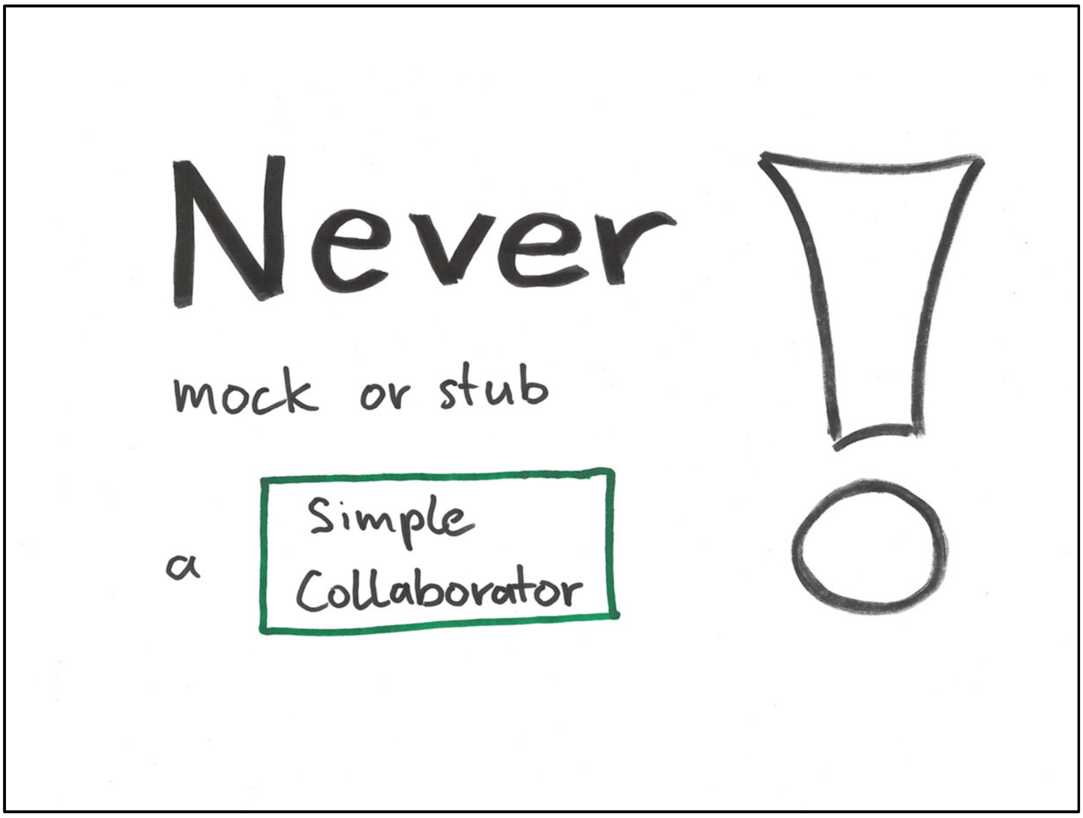
- fast
- easy to create
- easy to setup
- easy state access

Typical more intelligent data structures:

- Transfer Objects
- Composite data structures (Mapped XML or JSON)
- Rich Domain Objects (Entities and Values in DDD)
- Also: Method objects, Strategies

Do I have to mock/stub them? NOOOOOOO!!!! NEVER!!!!!

Using is not testing → Consider e.g. java.lang.String



Never

mock or stub

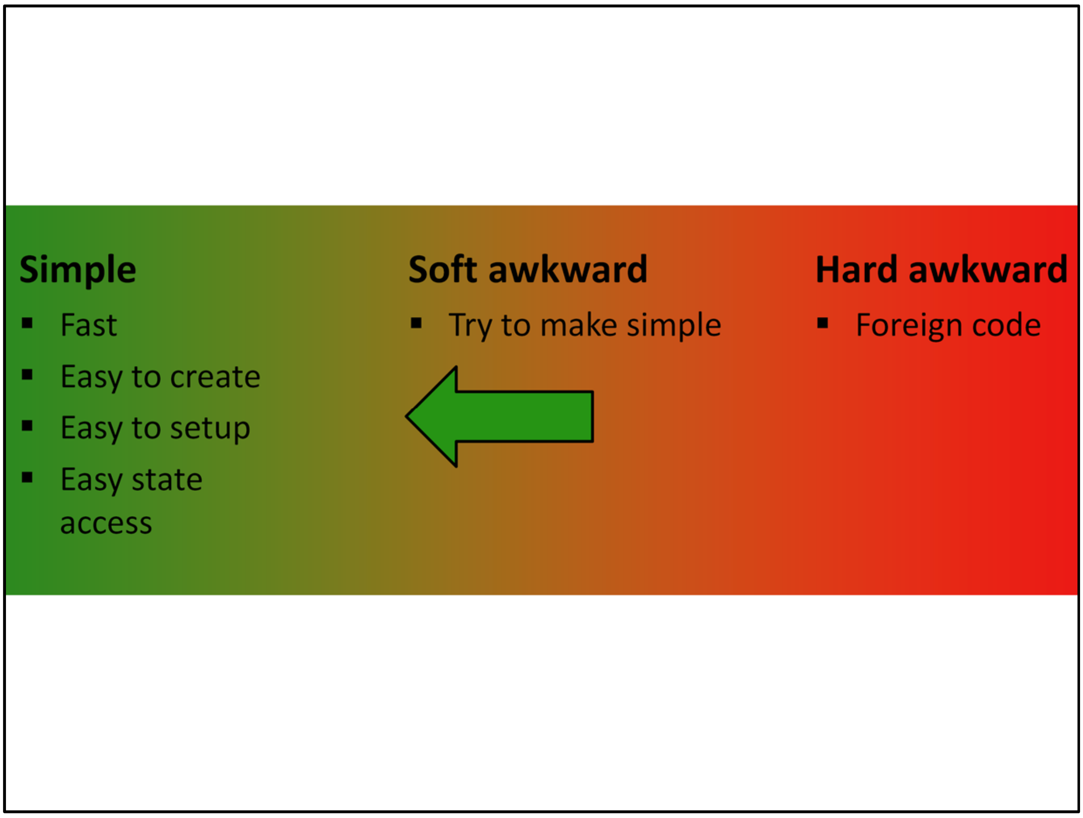
a Simple Collaborator

!

Gotteslästerung am Heiligen Unit-Test-Gott? → Hinweis auf Kai Zen

Hint on difference between testing an object or just using it during testing another object (consider String, Integer, HashSet, ArrayList…)

Und wichtig ist vor allem, das die Tests rasend schnell und unabhängig von irgendeinem ausserhalb vorbereiteten Umgebung sind.
→ Damit ich sie jederzeit nach jeder kleinen Code-Änderung lokal ausführen kann.



Simple

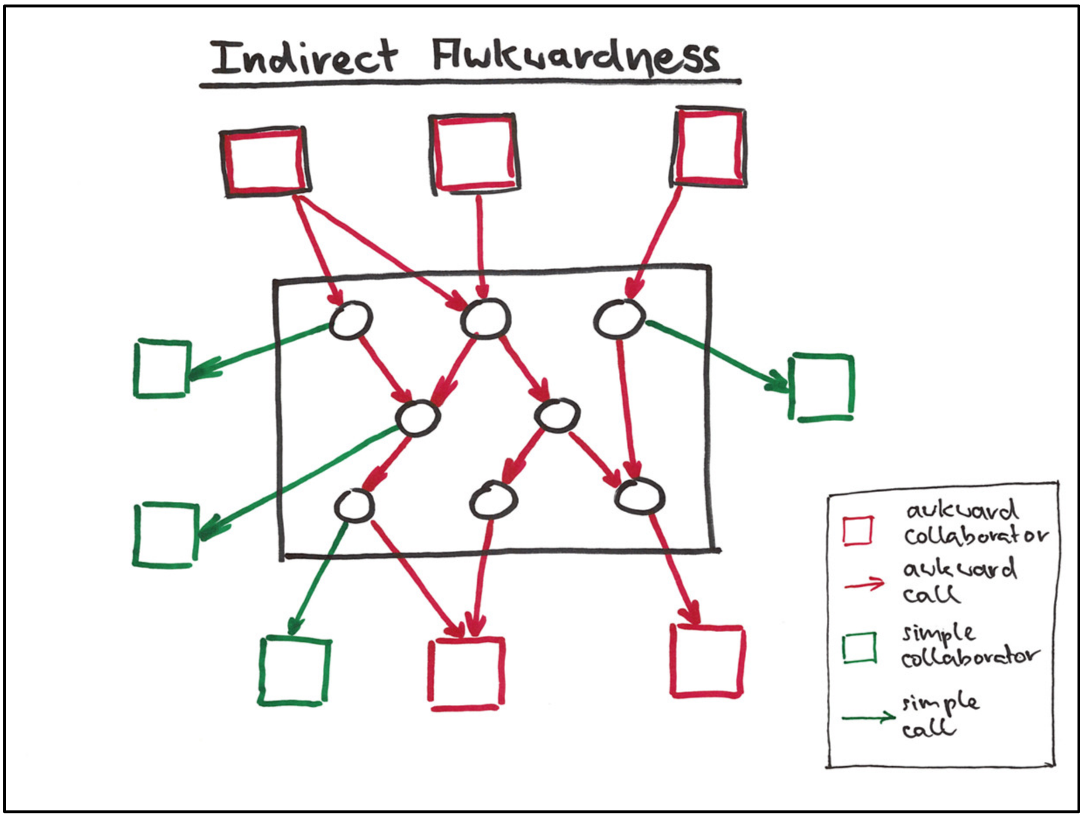
- Fast
- Easy to create
- Easy to setup
- Easy state access

Soft awkward

- Try to make simple

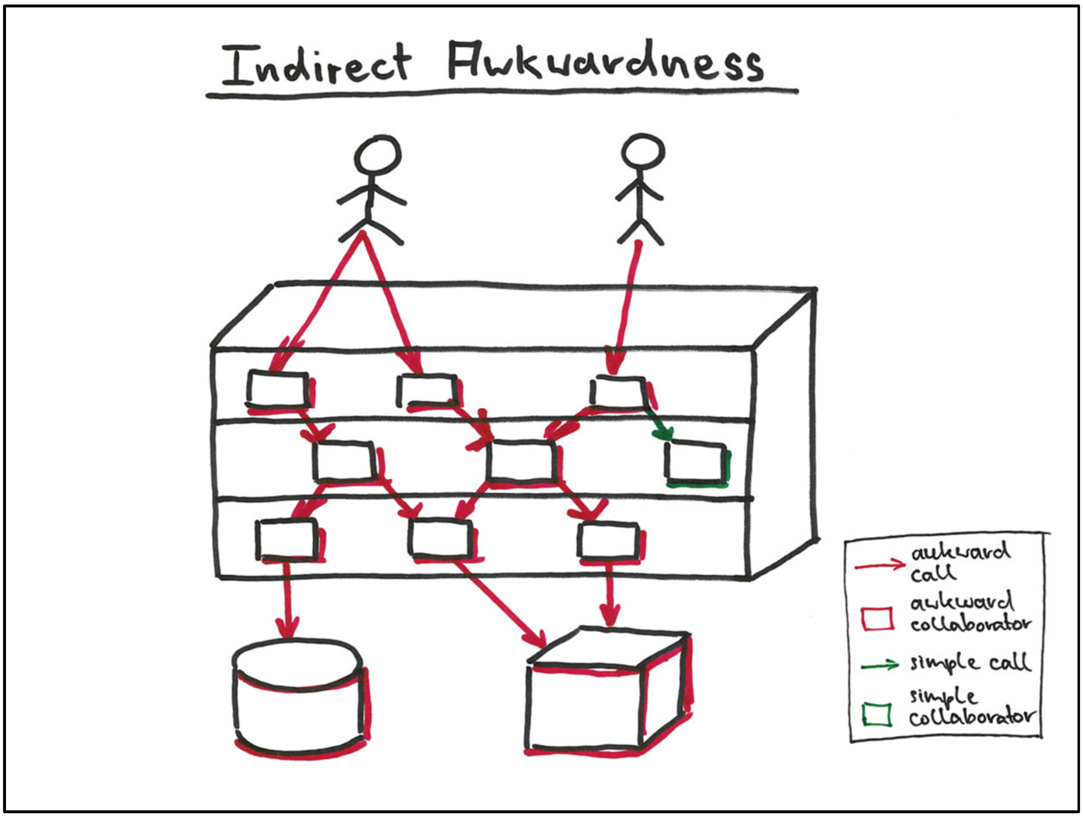
Hard awkward

- Foreign code



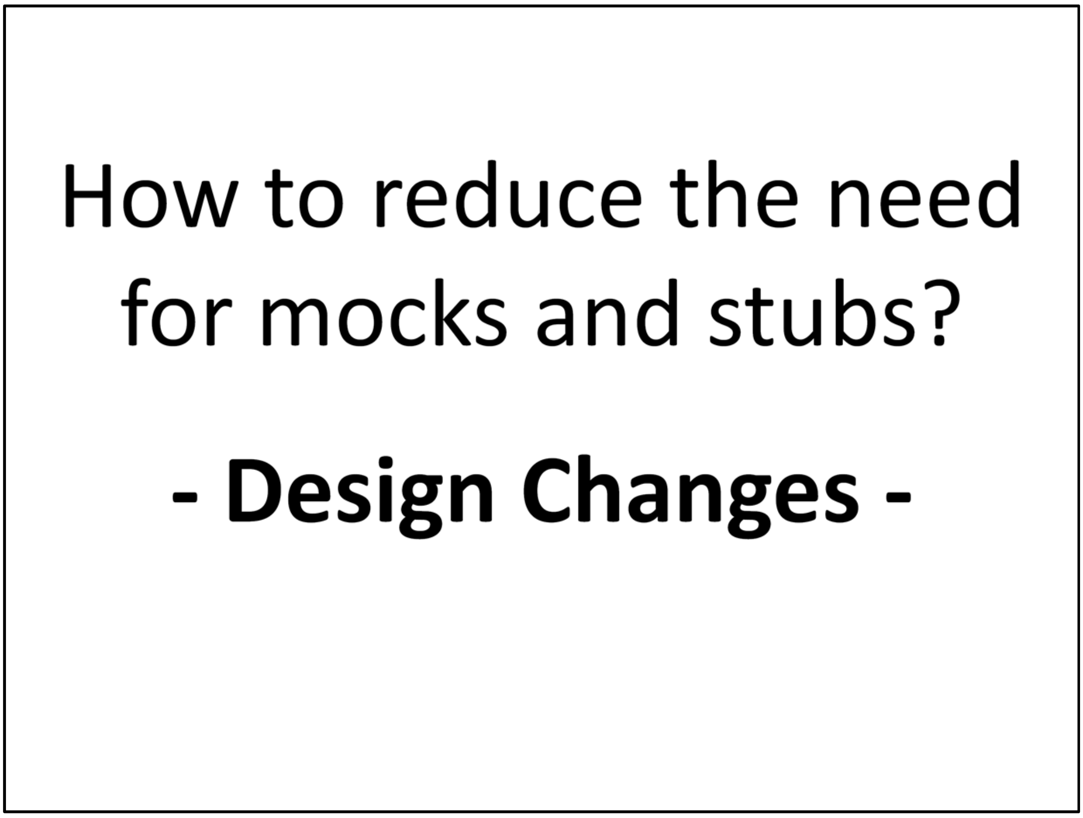
Awkwardness ist ansteckend…

Good guy but hangs out with the wrong persons… ;-)



Hochgradig anstecken…
Alles hängt irgendwie indirect an der Datenbank oder einem anderen Umsystem.

„But I need to mock, everything hangs directly or indirectly at the database“…
Well…

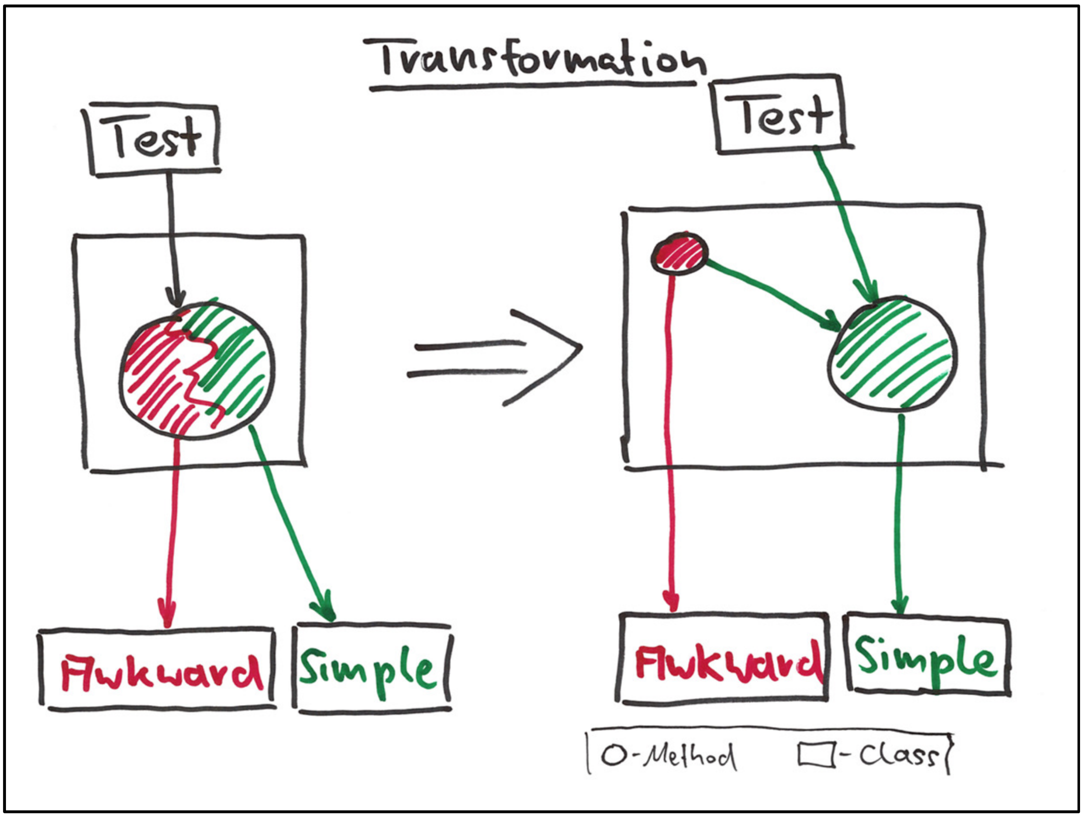


How to reduce the need for mocks and stubs?

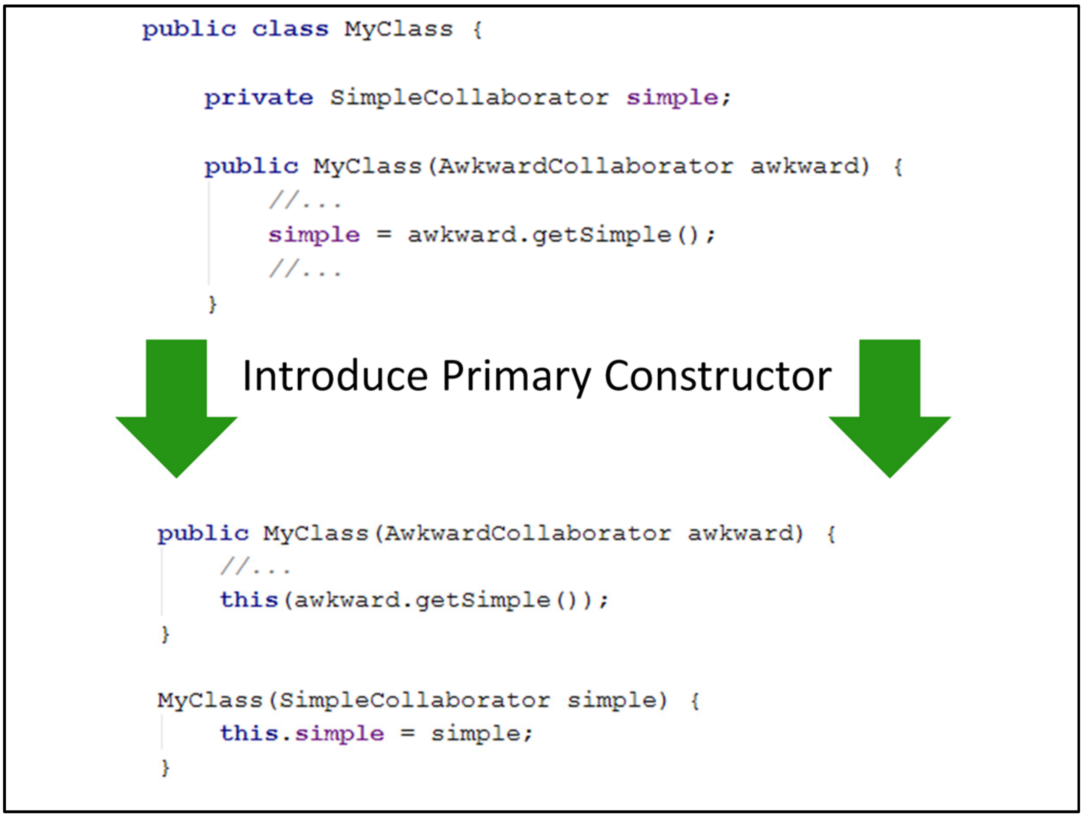
- Design Changes -

Extensive Stubbing and Mocking is a sypthome of two high coupling and missing separation of concerns.

The larger the setup of a unit test the more assuptions is it making and the more fragile it is potentially.



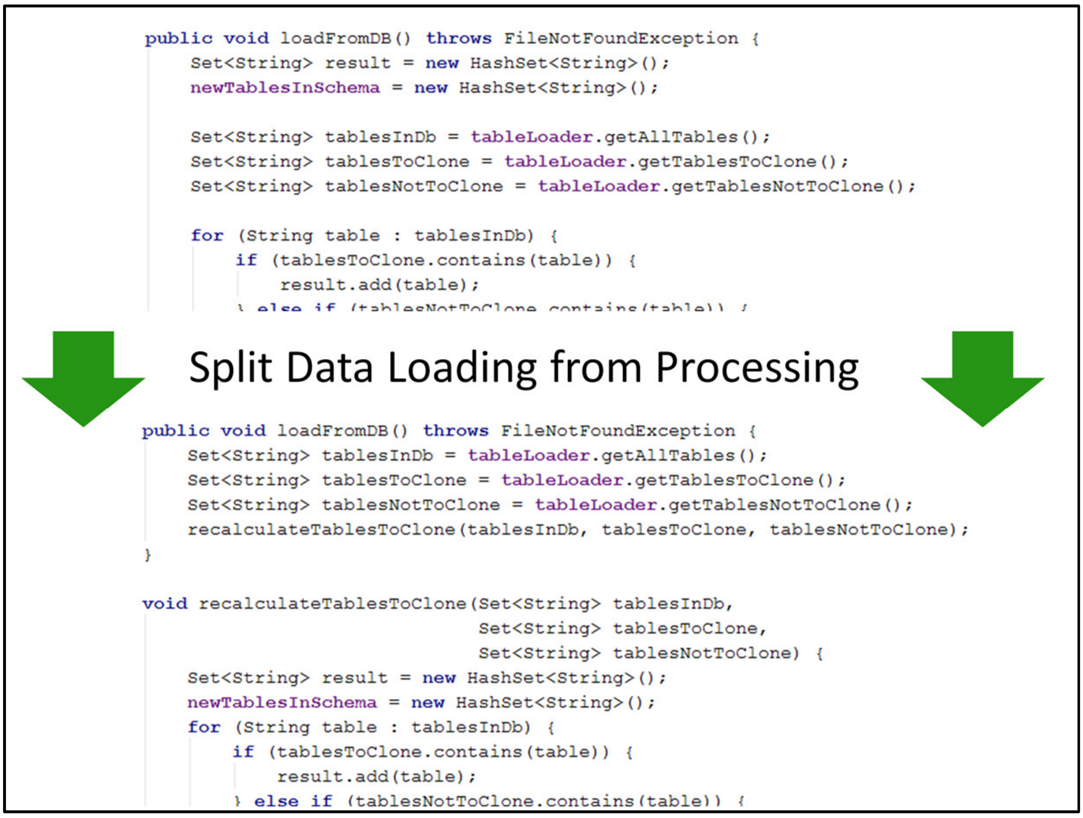
Split the awkward from the non awkward code parts on the method level.



```java
public class MyClass {

    private SimpleCollaborator simple;

    public MyClass(AwkwardCollaborator awkward) {
        //...
        simple = awkward.getSimple();
        //...
    }
```

Introduce Primary Constructor

```java
public MyClass(AwkwardCollaborator awkward) {
    //...
    this(awkward.getSimple());
}

MyClass(SimpleCollaborator simple) {
    this.simple = simple;
}
```

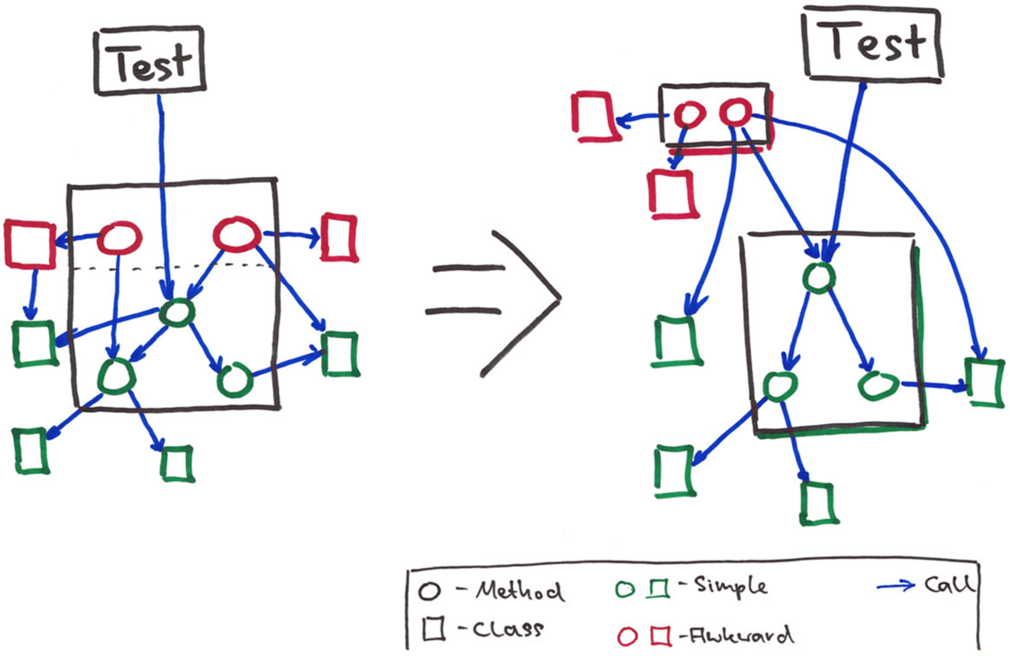


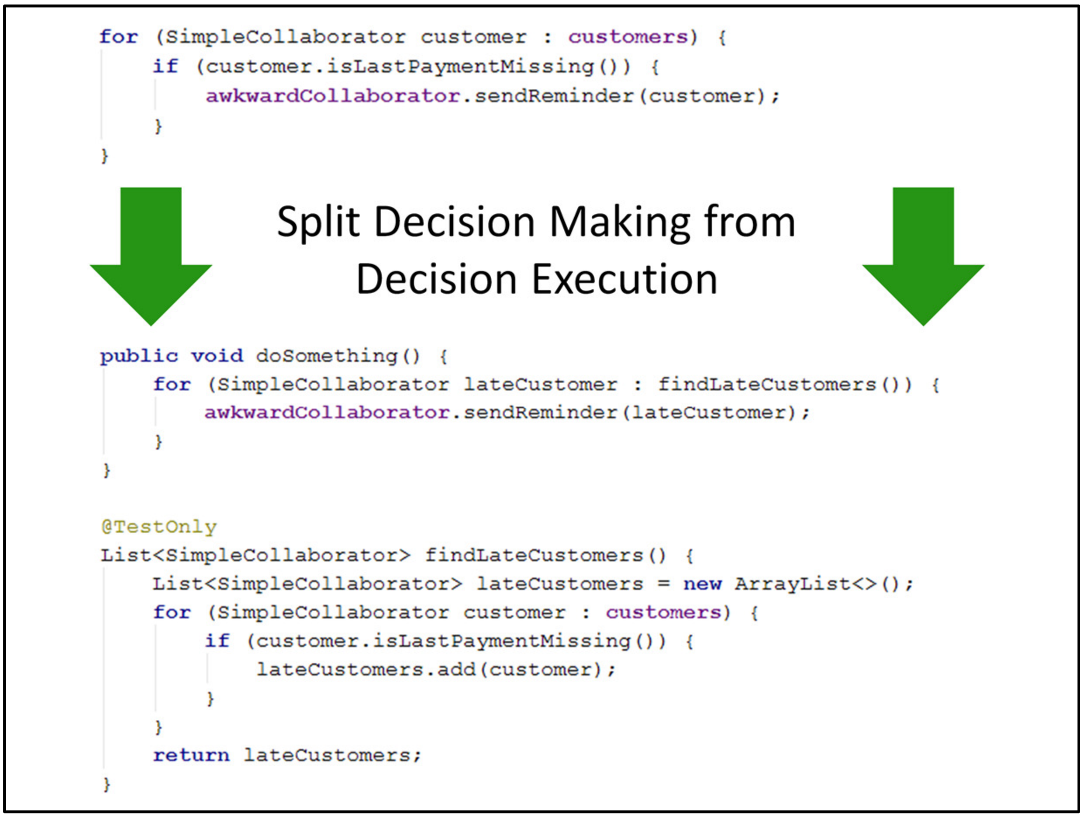
```
public void loadFromDB() throws FileNotFoundException {
    Set<String> result = new HashSet<String>();
    newTablesInSchema = new HashSet<String>();

    Set<String> tablesInDb = tableLoader.getAllTables();
    Set<String> tablesToClone = tableLoader.getTablesToClone();
    Set<String> tablesNotToClone = tableLoader.getTablesNotToClone();

    for (String table : tablesInDb) {
        if (tablesToClone.contains(table)) {
            result.add(table);
        } else if (tablesNotToClone.contains(table)) {
```

Split Data Loading from Processing

```
public void loadFromDB() throws FileNotFoundException {
    Set<String> tablesInDb = tableLoader.getAllTables();
    Set<String> tablesToClone = tableLoader.getTablesToClone();
    Set<String> tablesNotToClone = tableLoader.getTablesNotToClone();
    recalculateTablesToClone(tablesInDb, tablesToClone, tablesNotToClone);
}

void recalculateTablesToClone(Set<String> tablesInDb,
                              Set<String> tablesToClone,
                              Set<String> tablesNotToClone) {
    Set<String> result = new HashSet<String>();
    newTablesInSchema = new HashSet<String>();
    for (String table : tablesInDb) {
        if (tablesToClone.contains(table)) {
            result.add(table);
        } else if (tablesNotToClone.contains(table)) {
```

Transformation

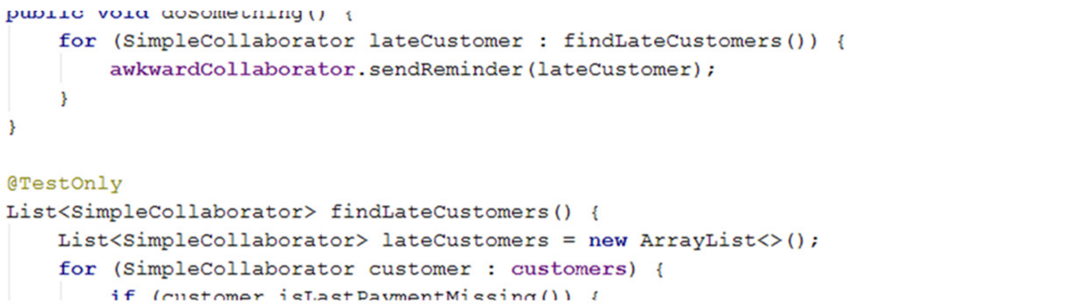




```
for (SimpleCollaborator customer : customers) {
    if (customer.isLastPaymentMissing()) {
        awkwardCollaborator.sendReminder(customer);
    }
}
```

Split Decision Making from Decision Execution

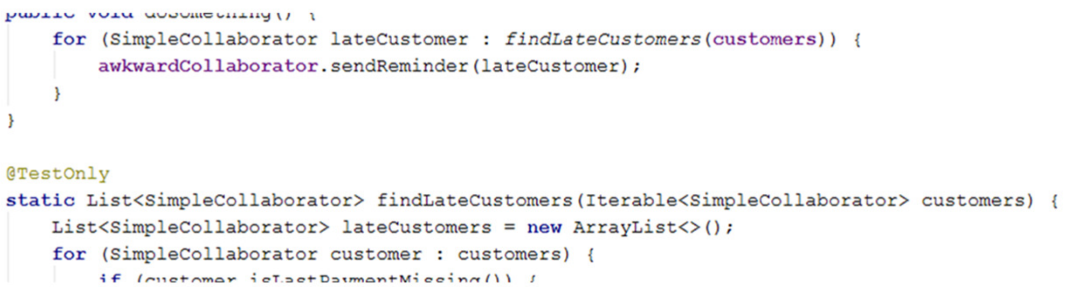
```
public void doSomething() {
    for (SimpleCollaborator lateCustomer : findLateCustomers()) {
        awkwardCollaborator.sendReminder(lateCustomer);
    }
}

@TestOnly
List<SimpleCollaborator> findLateCustomers() {
    List<SimpleCollaborator> lateCustomers = new ArrayList<>();
    for (SimpleCollaborator customer : customers) {
        if (customer.isLastPaymentMissing()) {
            lateCustomers.add(customer);
        }
    }
    return lateCustomers;
}
```



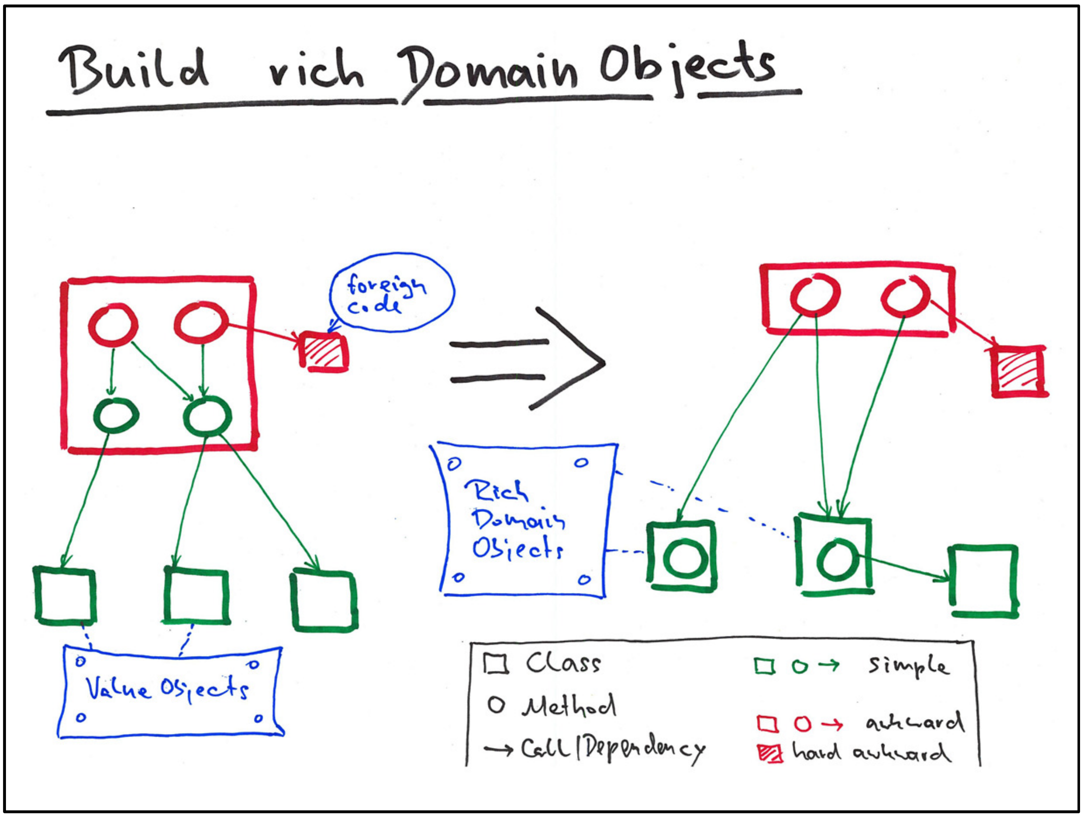
```
public void doSomething() {
    for (SimpleCollaborator lateCustomer : findLateCustomers()) {
        awkwardCollaborator.sendReminder(lateCustomer);
    }
}

@TestOnly
List<SimpleCollaborator> findLateCustomers() {
    List<SimpleCollaborator> lateCustomers = new ArrayList<>();
    for (SimpleCollaborator customer : customers) {
        if (customer.isLastPaymentMissing()) {
```

Make Static

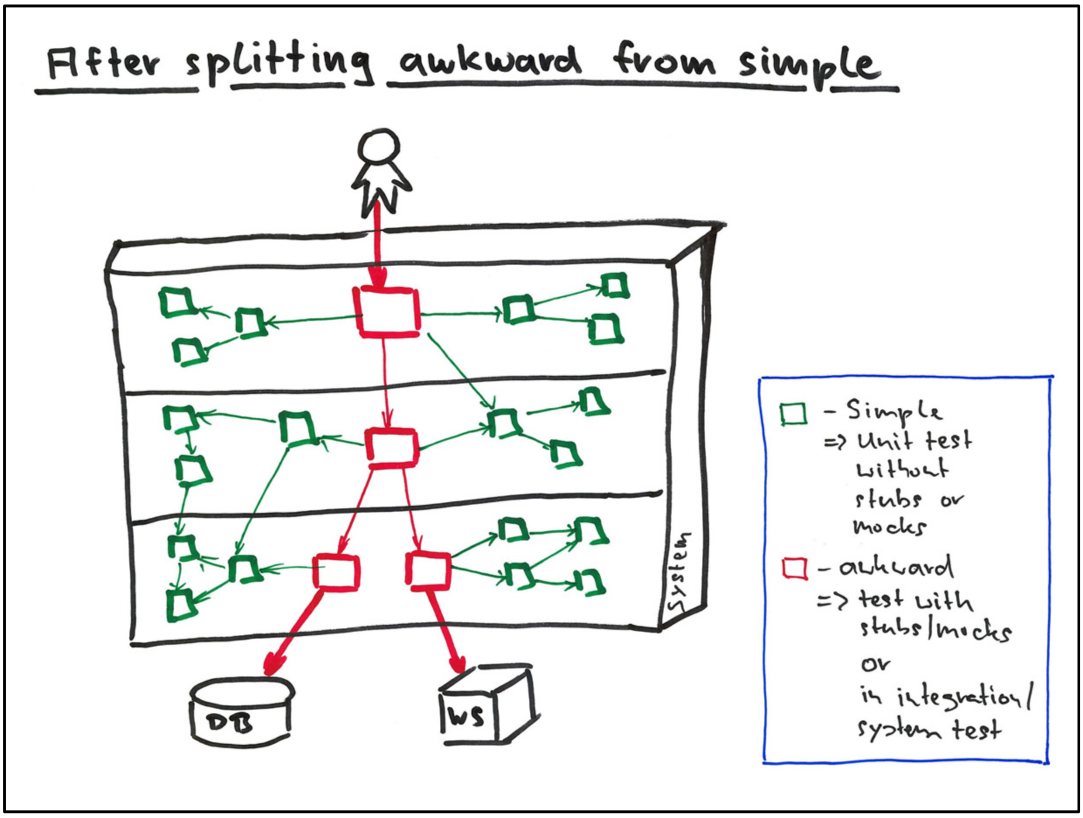
→ Objects above - Functions below



```
public void doSomething() {
    for (SimpleCollaborator lateCustomer : findLateCustomers(customers)) {
        awkwardCollaborator.sendReminder(lateCustomer);
    }
}

@TestOnly
static List<SimpleCollaborator> findLateCustomers(Iterable<SimpleCollaborator> customers) {
    List<SimpleCollaborator> lateCustomers = new ArrayList<>();
    for (SimpleCollaborator customer : customers) {
        if (customer.isLastPaymentMissing()) {
```



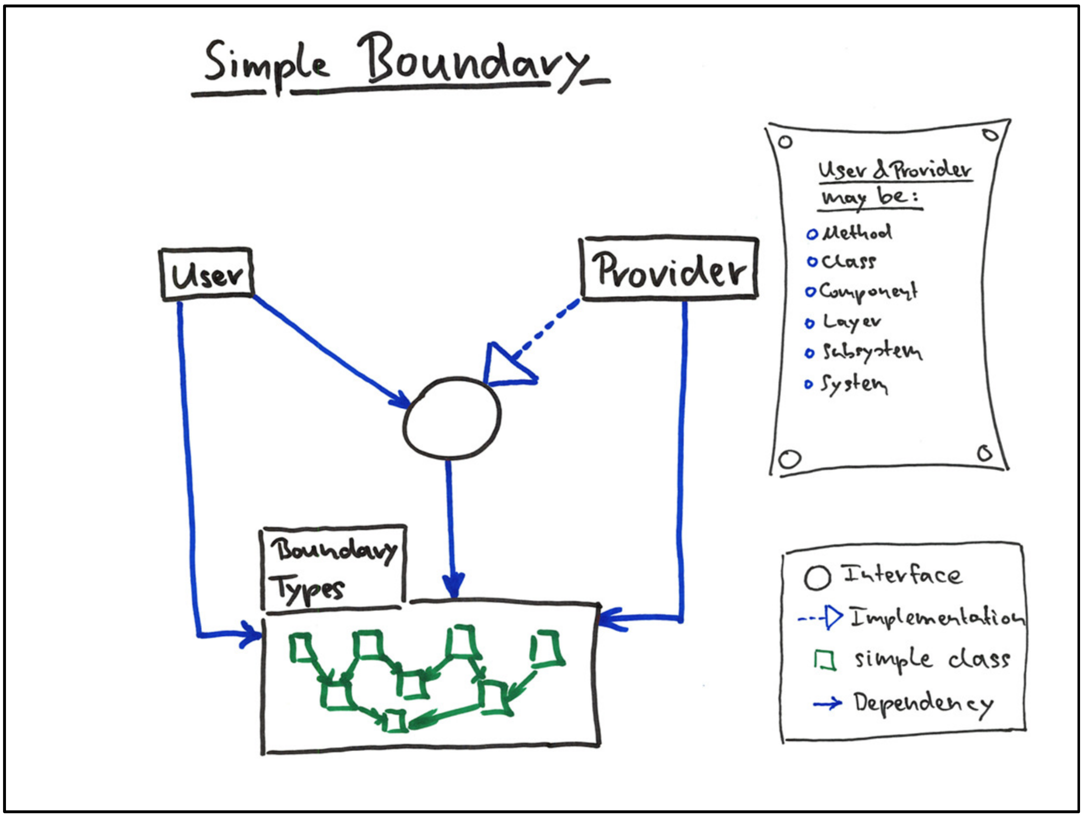
Build rich Domain Objects



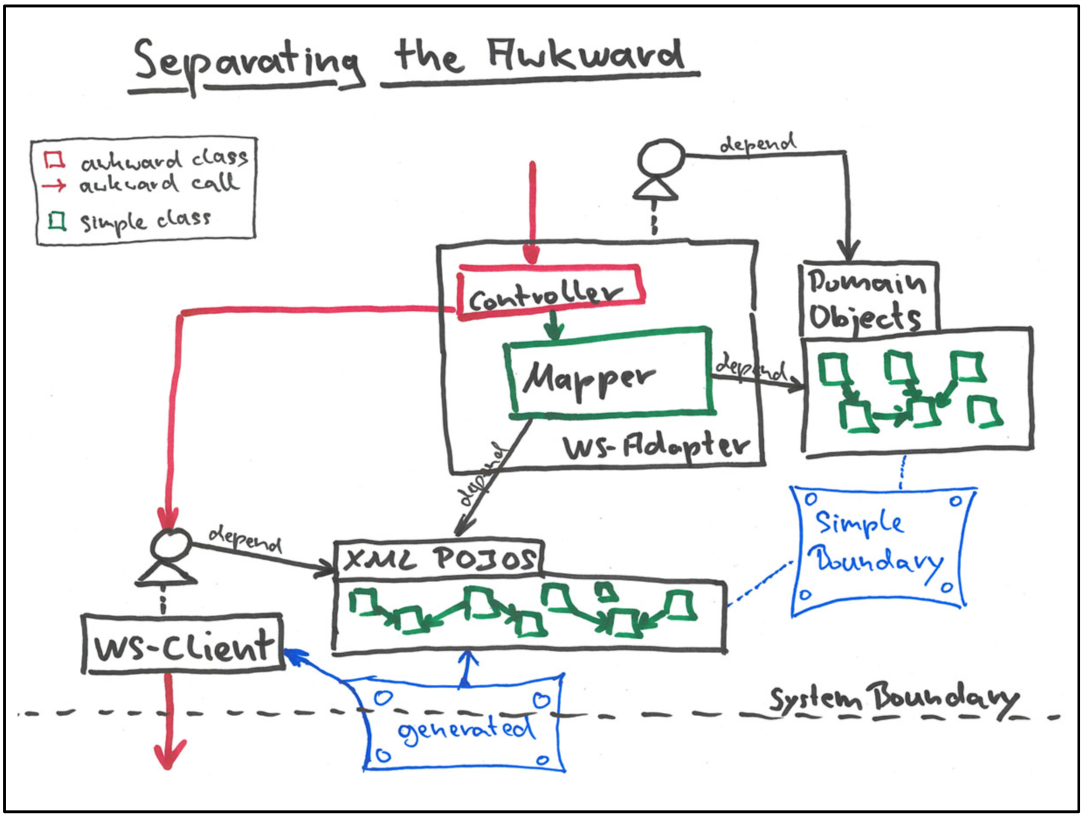
After splitting awkward from simple

🟩 - Simple
=> Unit test without stubs or mocks

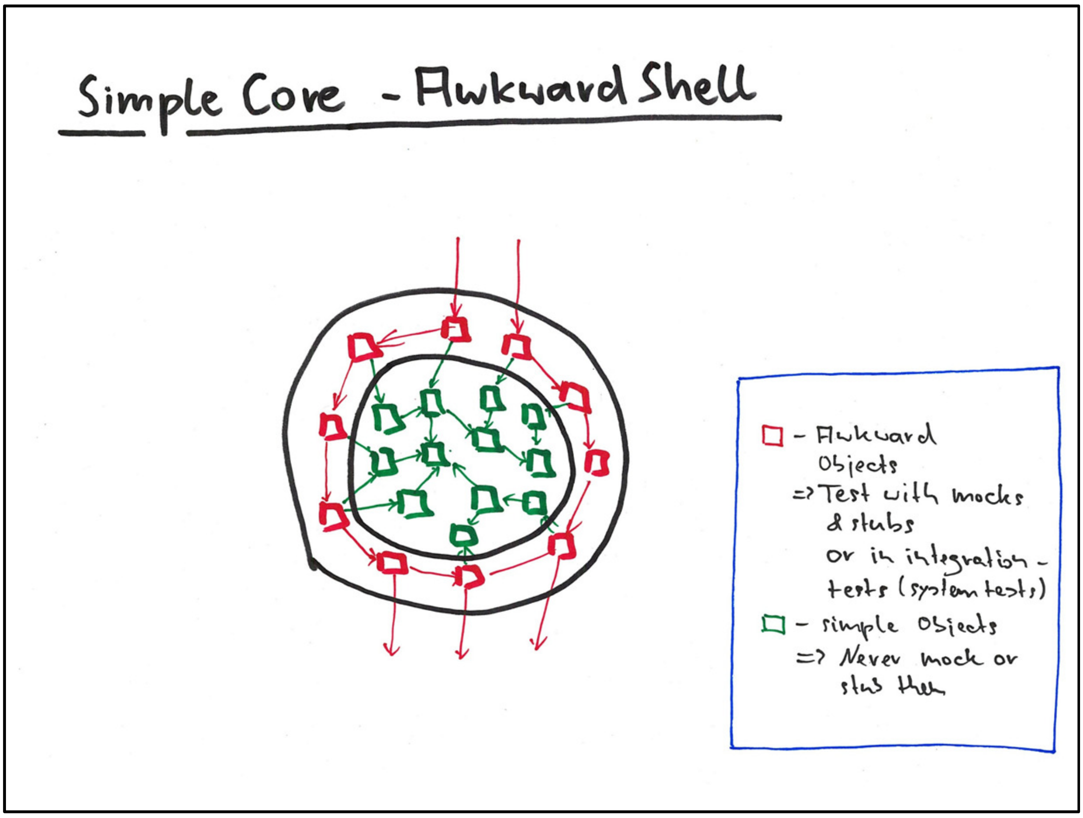
🟥 - awkward
=> test with stubs/mocks or in integration/system test



A simple boundary creates a natural means of isolation in the design without the need to use fakes and mocks.



No Mocks or stubs needed to unit-test the green parts.
Red (awkward) parts can be unit-tested with mocks or just in integration.



Simple Core - Awkward Shell

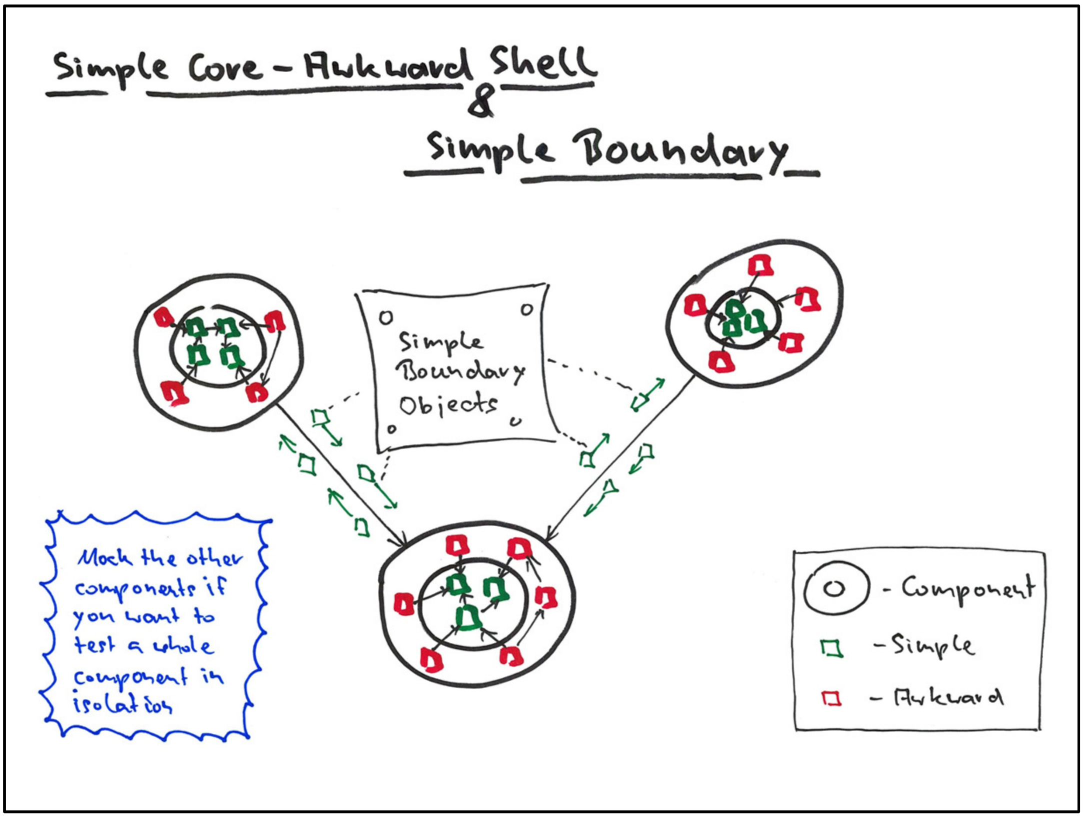
🟥 - Awkward Objects
=> Test with mocks & stubs
or in integration-tests (system tests)

🟩 - simple Objects
=> Never mock or stub them

Simple:
Domain Objects in the center
Application Logic around

Awkward:
Collaboration with the awkward world around in the shell.

Quiete similar to Domain Driven Design.

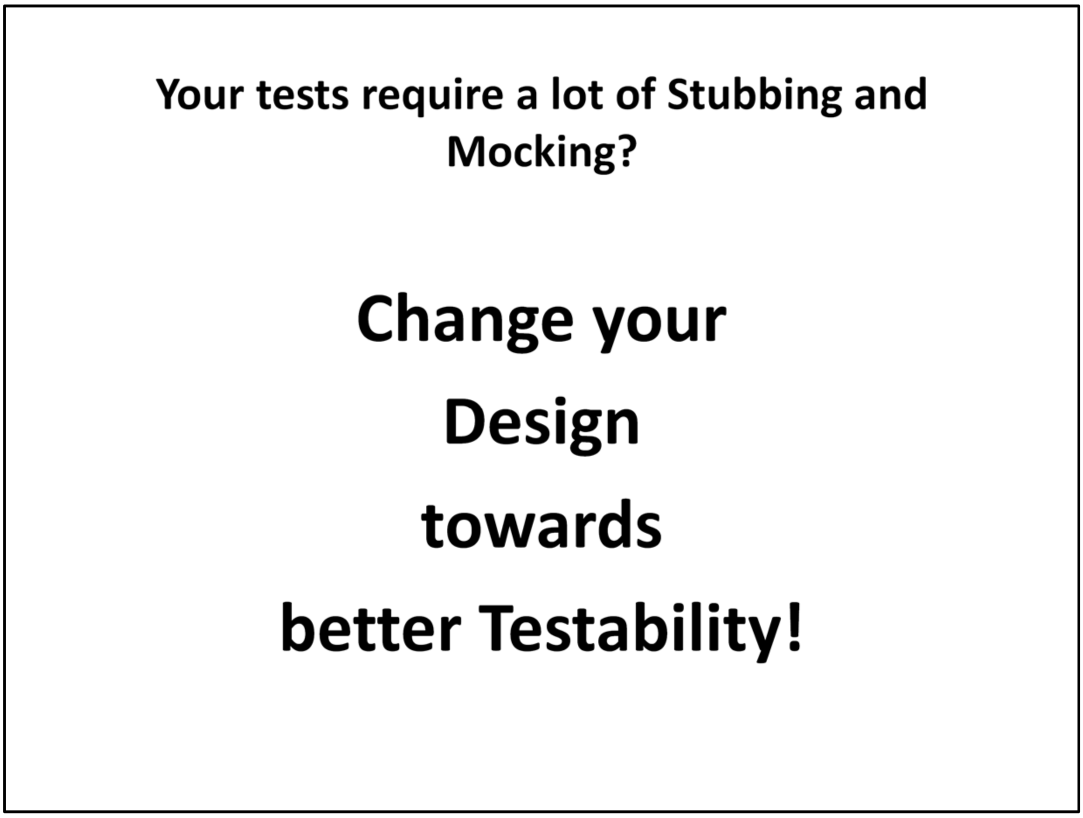


Simple Core - Awkward Shell & Simple Boundary

Mock the other components if you want to test a whole component in isolation

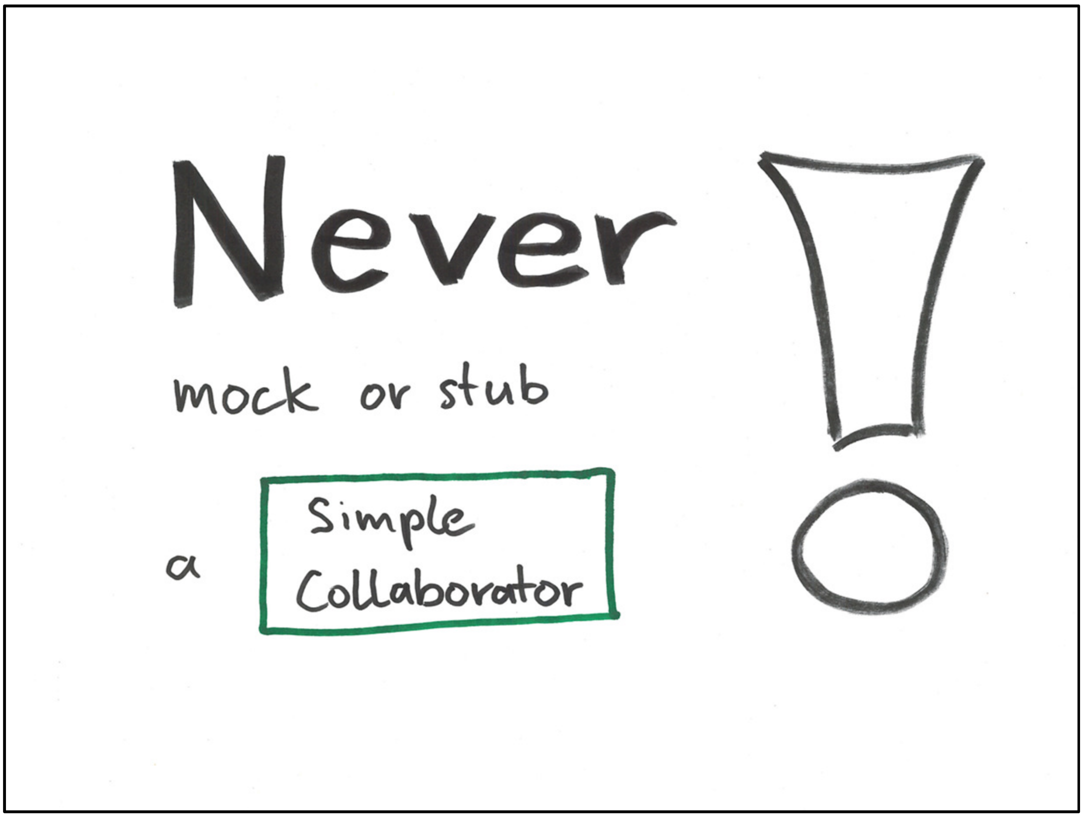
When still to use mock objects?

- Callbacks and Notifications (Fire&Forget)
- Component Testing
- Wrapper Testing (Decorator, Adapter, Proxy)
- Legacy Code Testing
- For Infrastructur Services in DDD, when testing Domain Objects



Your tests require a lot of Stubbing and Mocking?

Change your Design towards better Testability!



Never

mock or stub

a Simple Collaborator

!