# AUTOMATIC TESTS

## YOU WON'T FIND IT IN A BOOK

Jacek Milewski

looksok.wordpress.com

jacek.milewski.k@gmail.com

https://github.com/yacekmm/testingDemo

# WHY?

It's important

There are many ways to do it well

# BE AWARE OF THE OPTIONS

Choose the right tool for the problem you have

# HOW DOES IT ALL START

Unit testing at its basics

Branch: 1/initialTest/start

# INTERNAL DEPENDENCY

Mocking at its basics

Branch: 2/internalDependency/start

# ANY ISSUES?

Test duplicates implementation

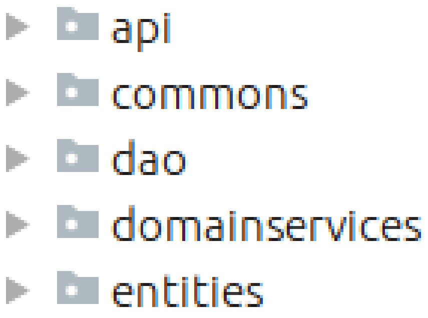
Tough refactoring

# USE REAL IMPLEMENTATIONS

What is my unit?

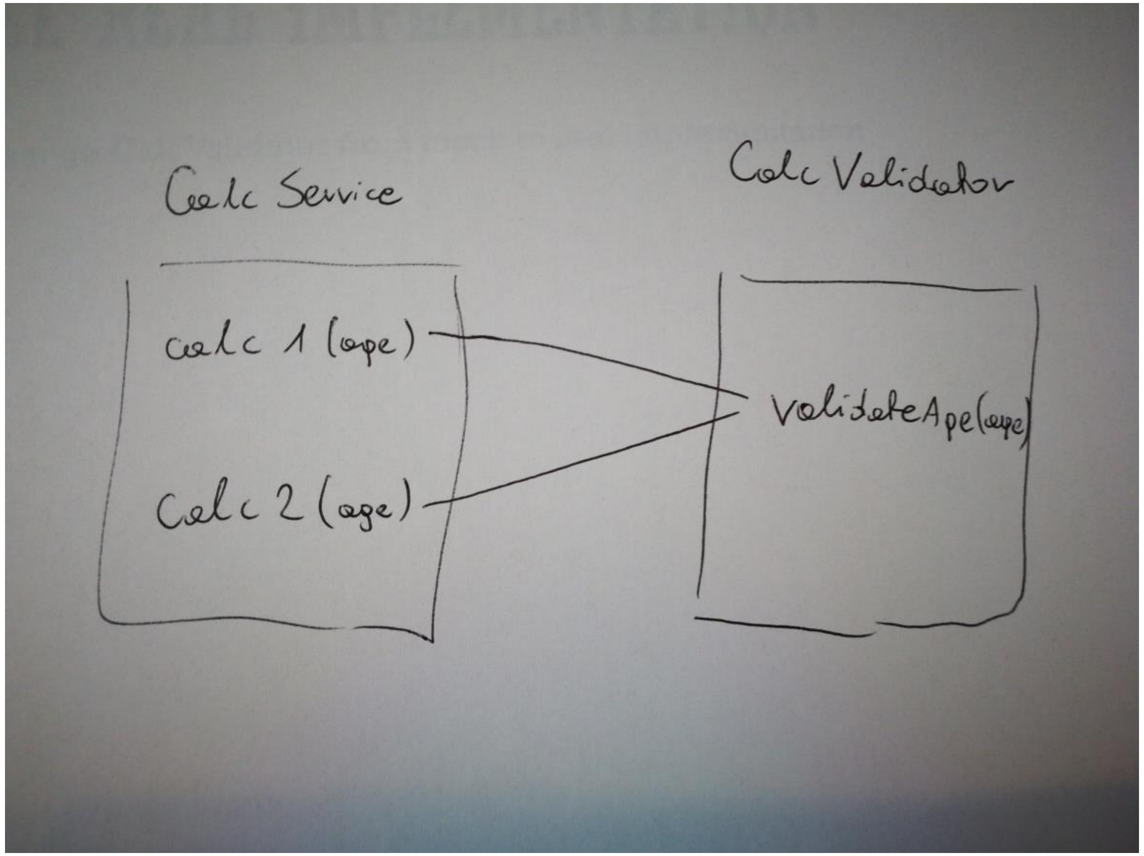
Branch: 3/intDepRealImplementation/done

# TAKE A CLOSER LOOK AT MODULES



- api
- commons
- dao
- domainservices
- entities

# VALIDATOR COMPLEXITY



# MODULE TEST

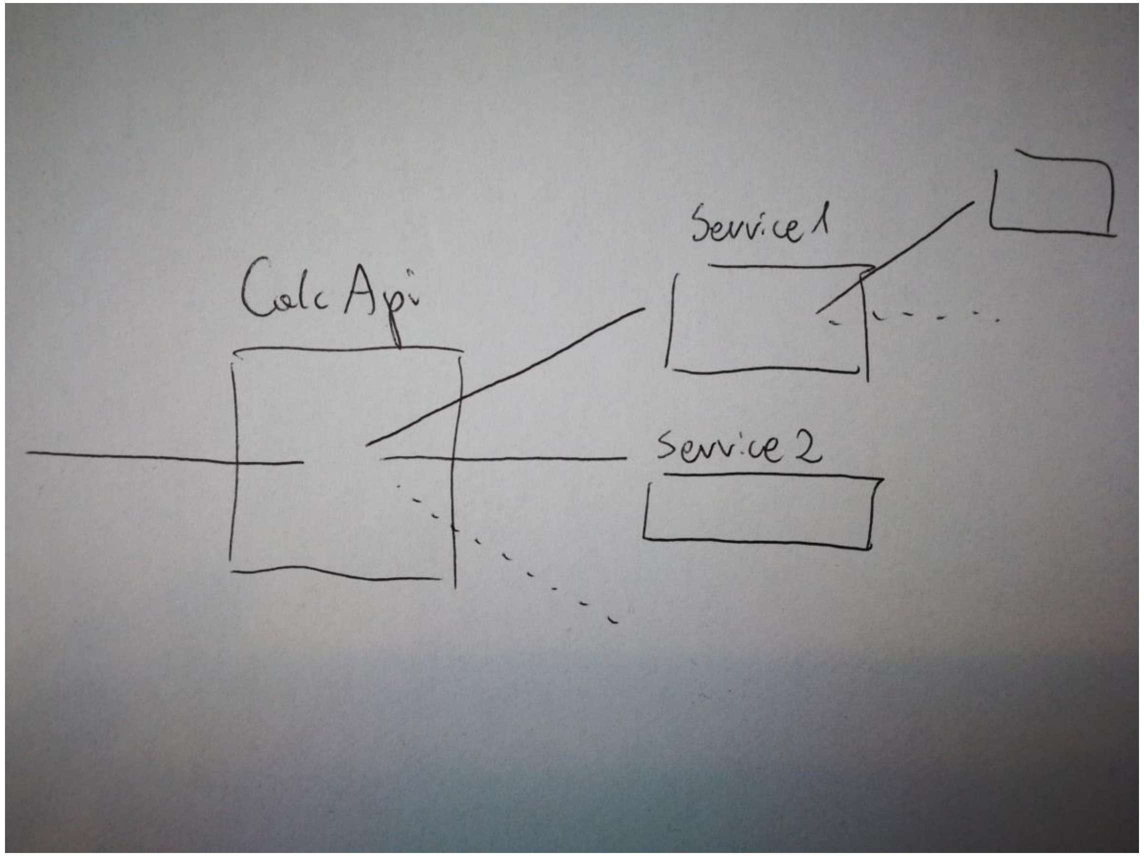
CalcApi

Branch: 4/moduleApi/start

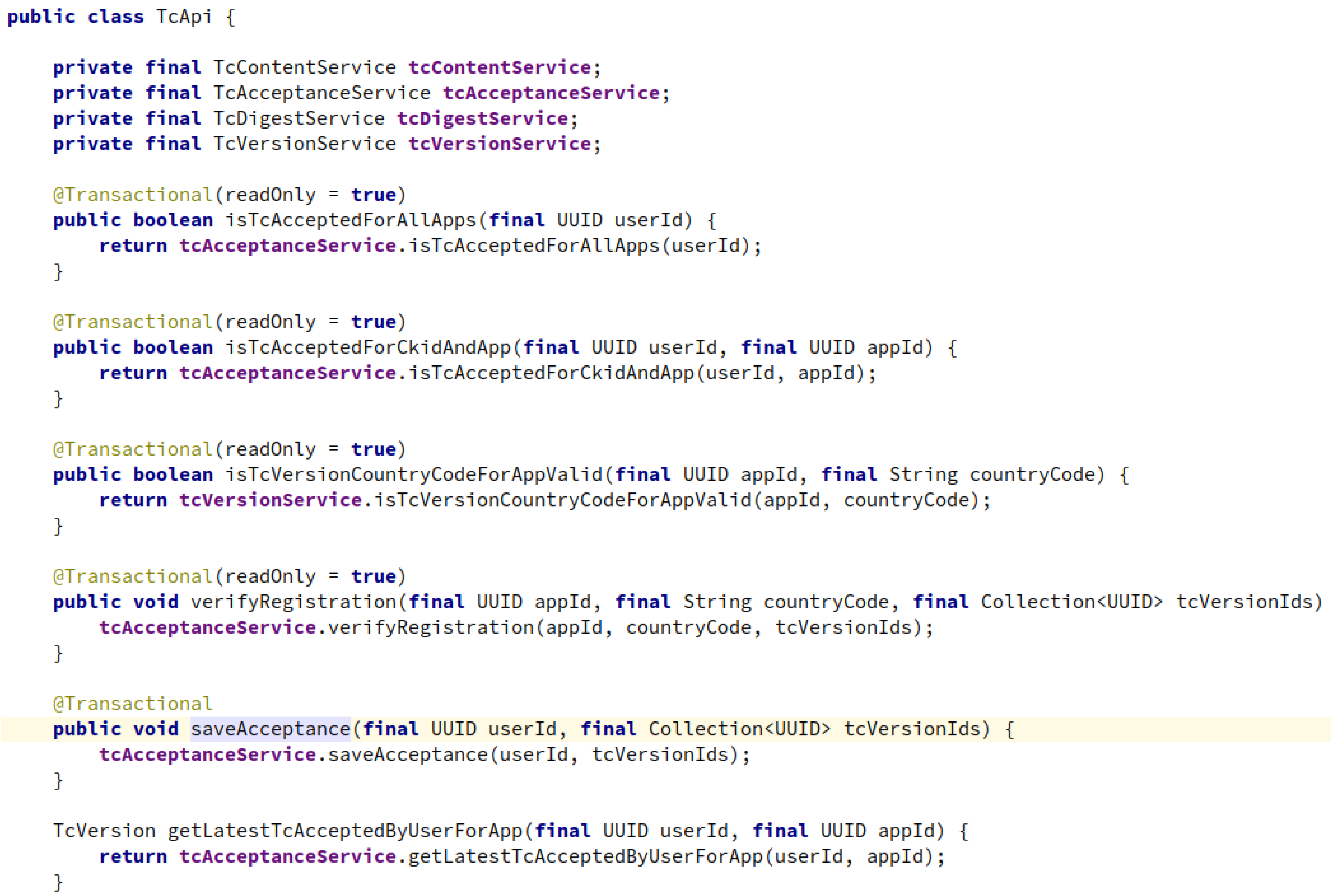
# THE UNIT

A module that exposes API

# API



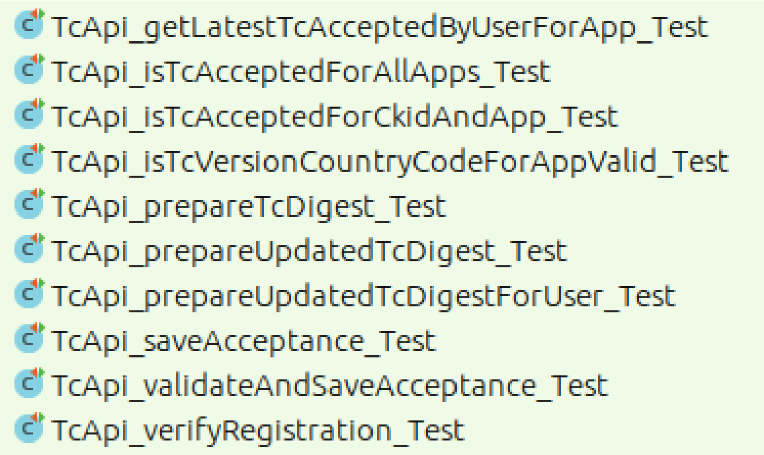
# API



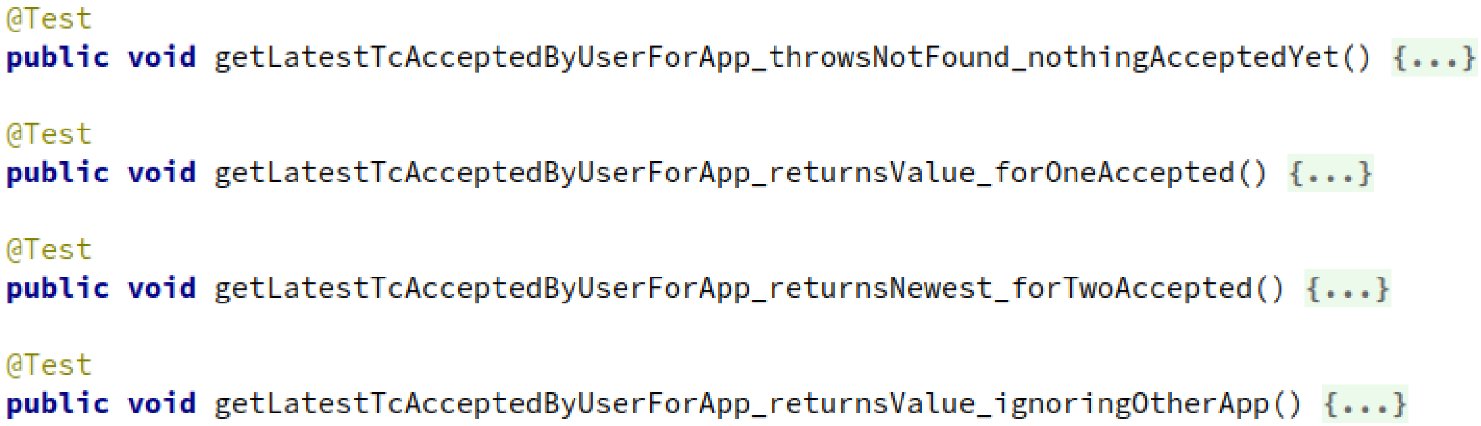
```
public class TcApi {

    private final TcContentService tcContentService;
    private final TcAcceptanceService tcAcceptanceService;
    private final TcDigestService tcDigestService;
    private final TcVersionService tcVersionService;

    @Transactional(readOnly = true)
    public boolean isTcAcceptedForAllApps(final UUID userId) {
        return tcAcceptanceService.isTcAcceptedForAllApps(userId);
    }

    @Transactional(readOnly = true)
    public boolean isTcAcceptedForCkidAndApp(final UUID userId, final UUID appId) {
        return tcAcceptanceService.isTcAcceptedForCkidAndApp(userId, appId);
    }

    @Transactional(readOnly = true)
    public boolean isTcVersionCountryCodeForAppValid(final UUID appId, final String countryCode) {
        return tcVersionService.isTcVersionCountryCodeForAppValid(appId, countryCode);
    }

    @Transactional(readOnly = true)
    public void verifyRegistration(final UUID appId, final String countryCode, final Collection<UUID> tcVersionIds)
        tcAcceptanceService.verifyRegistration(appId, countryCode, tcVersionIds);
    }

    @Transactional
    public void saveAcceptance(final UUID userId, final Collection<UUID> tcVersionIds) {
        tcAcceptanceService.saveAcceptance(userId, tcVersionIds);
    }

    TcVersion getLatestTcAcceptedByUserForApp(final UUID userId, final UUID appId) {
        return tcAcceptanceService.getLatestTcAcceptedByUserForApp(userId, appId);
    }
```

# MODULE UNIT TESTS

- Test Only Api class
- One test file per API method
- Testing all edge cases
- Not starting app context (framework independent)



```
TcApi_getLatestTcAcceptedByUserForApp_Test
TcApi_isTcAcceptedForAllApps_Test
TcApi_isTcAcceptedForCkidAndApp_Test
TcApi_isTcVersionCountryCodeForAppValid_Test
TcApi_prepareTcDigest_Test
TcApi_prepareUpdatedTcDigest_Test
TcApi_prepareUpdatedTcDigestForUser_Test
TcApi_saveAcceptance_Test
TcApi_validateAndSaveAcceptance_Test
TcApi_verifyRegistration_Test
```



```java
@Test
public void getLatestTcAcceptedByUserForApp_throwsNotFound_nothingAcceptedYet() {...}

@Test
public void getLatestTcAcceptedByUserForApp_returnsValue_forOneAccepted() {...}

@Test
public void getLatestTcAcceptedByUserForApp_returnsNewest_forTwoAccepted() {...}

@Test
public void getLatestTcAcceptedByUserForApp_returnsValue_ignoringOtherApp() {...}
```

# EXTERNAL MODULE DEPENDENCIES

Interaction with UserApi

Branch: 5/externalModuleDep/done

# DATABASE

Fake the Repository

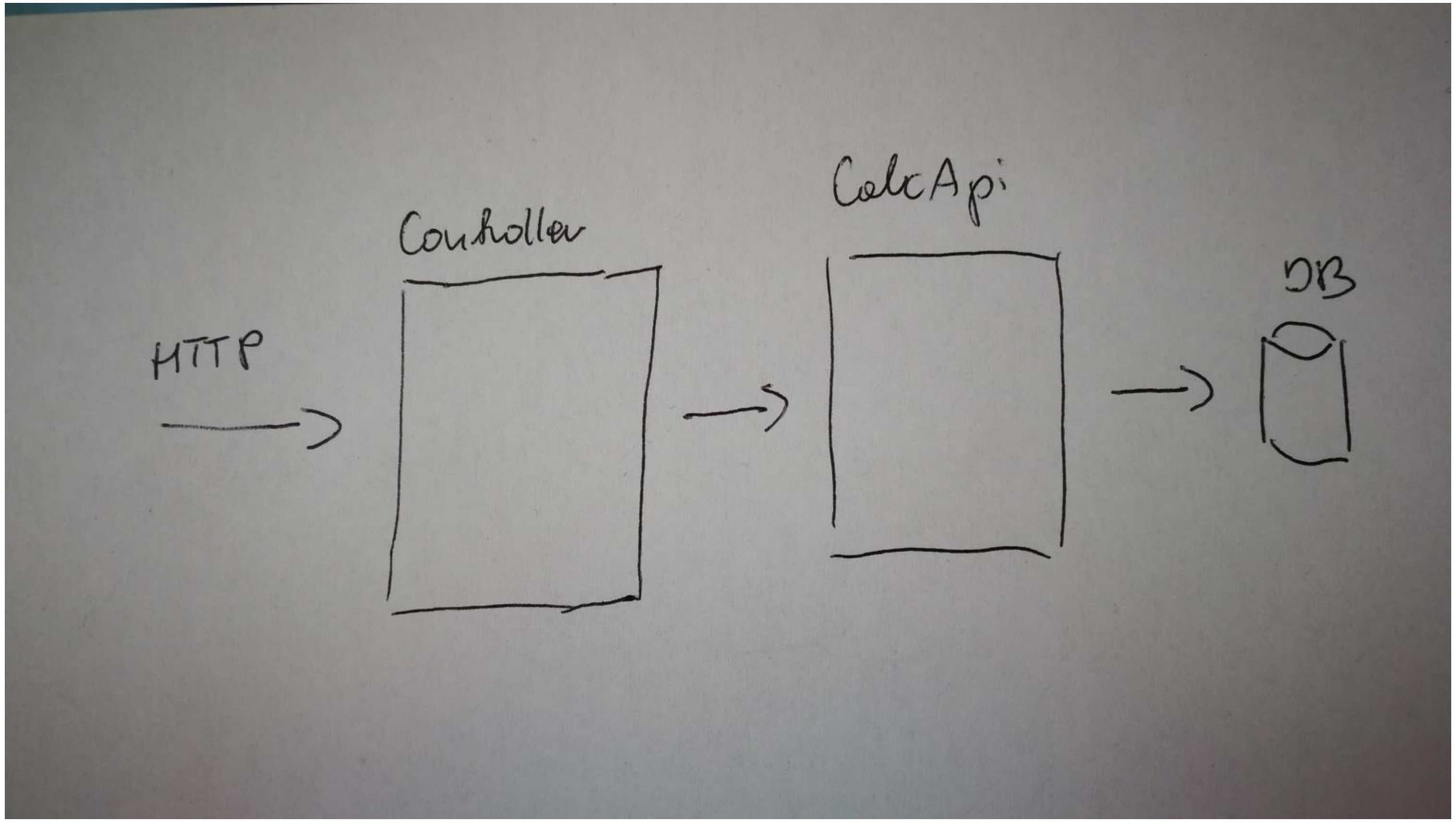
Branch: 6/repository/start

# INTEGRATION TESTS

Confirm it all plays well together

Branch: 7/integrationTest/start

# INTEGRATION TESTS

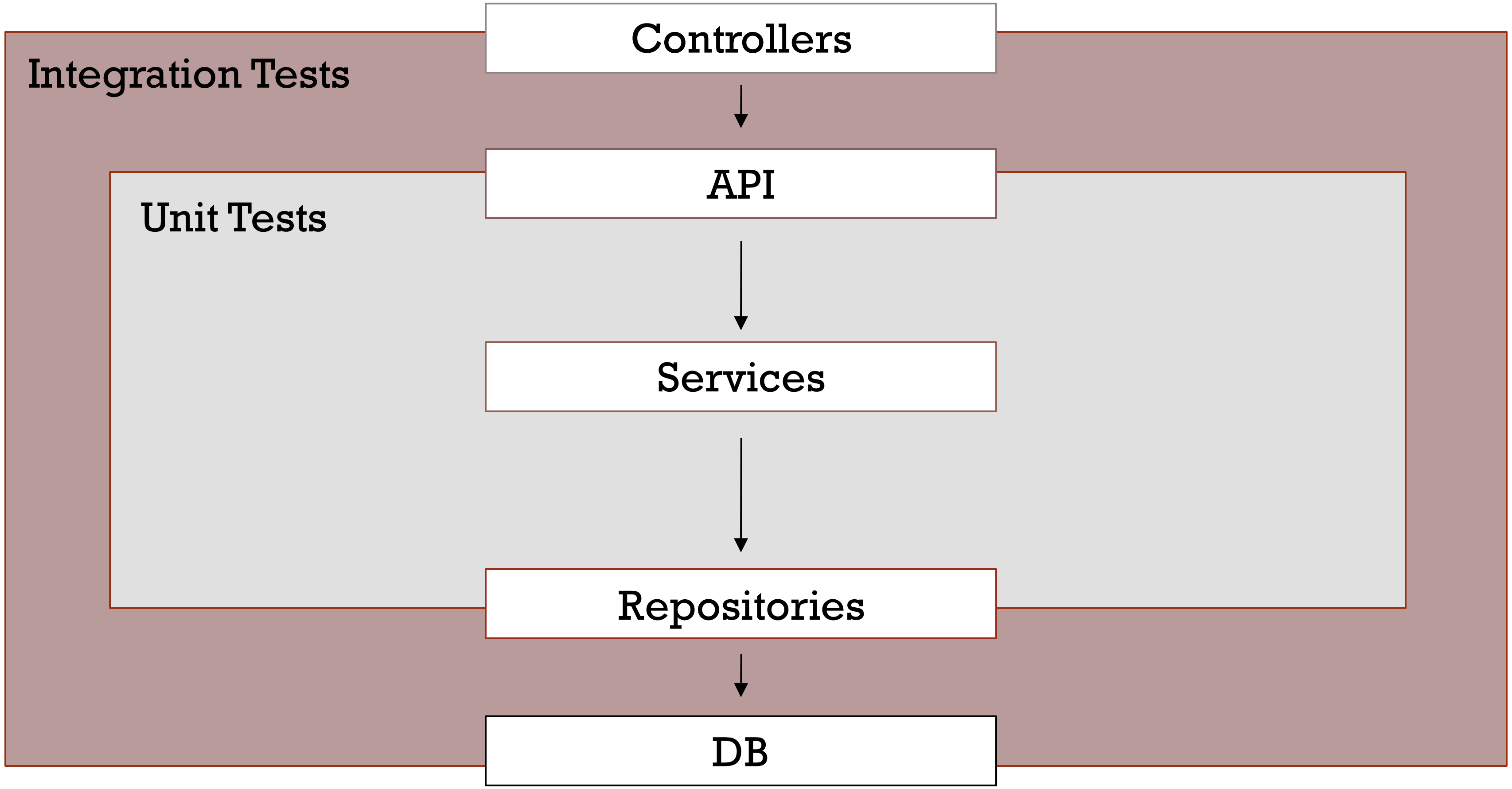


# INTEGRATION TESTS

Confirm that

- REST API is exposed
- API is secured
- Faked / mocked dependencies are working
- Exceptions are handled
- API contract is documented

# MODULE INTERNALS



# TESTING STRATEGIES

How Much of these do you need:

Integration tests

Unit tests

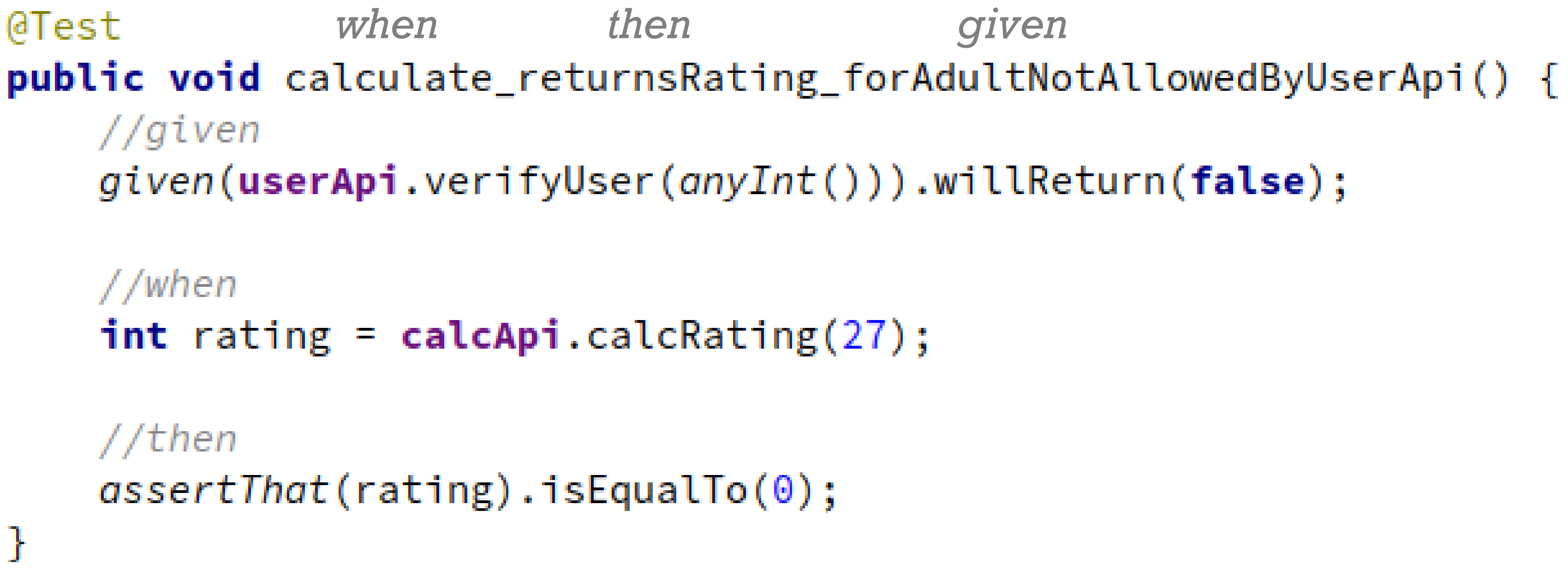
Micro tests

No tests

# TOOLBOX

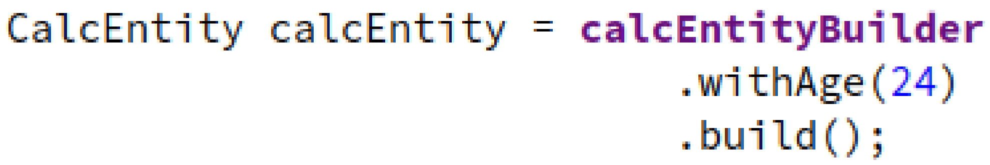
# NAMING

1. Body: *given – when - then*
2. Name: *behavior_expectedResult_conditions*

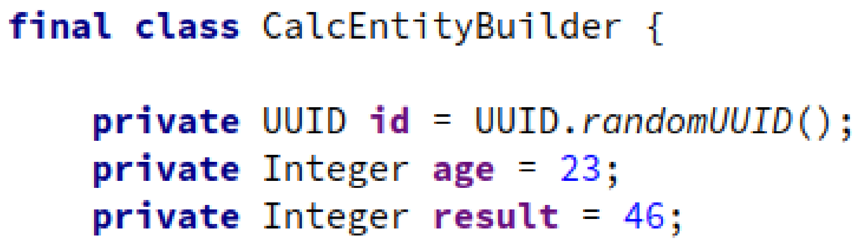


```
@Test         when        then            given
public void calculate_returnsRating_forAdultNotAllowedByUserApi() {
    //given
    given(userApi.verifyUser(anyInt())).willReturn(false);

    //when
    int rating = calcApi.calcRating(27);

    //then
    assertThat(rating).isEqualTo(0);
}
```

# BUILDERS

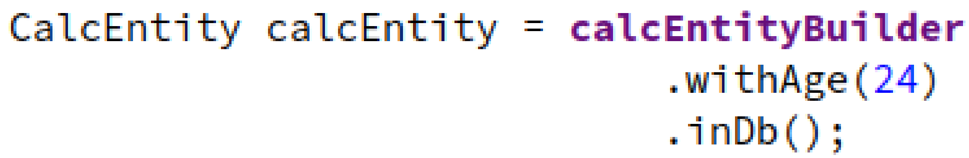


```java
CalcEntity calcEntity = calcEntityBuilder
                            .withAge(24)
                            .build();
```

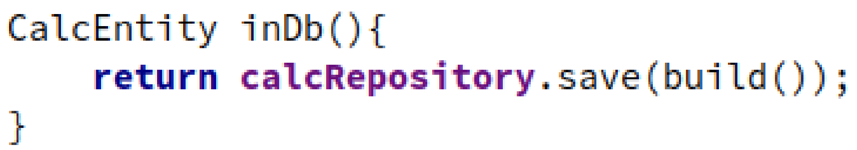


```java
final class CalcEntityBuilder {

    private UUID id = UUID.randomUUID();
    private Integer age = 23;
    private Integer result = 46;
```

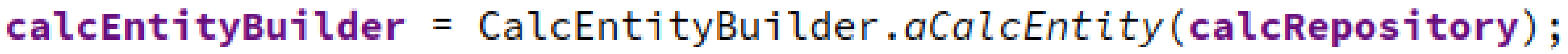
# SET STATE WITH BUILDERS



```java
CalcEntity calcEntity = calcEntityBuilder
                            .withAge(24)
                            .inDb();
```

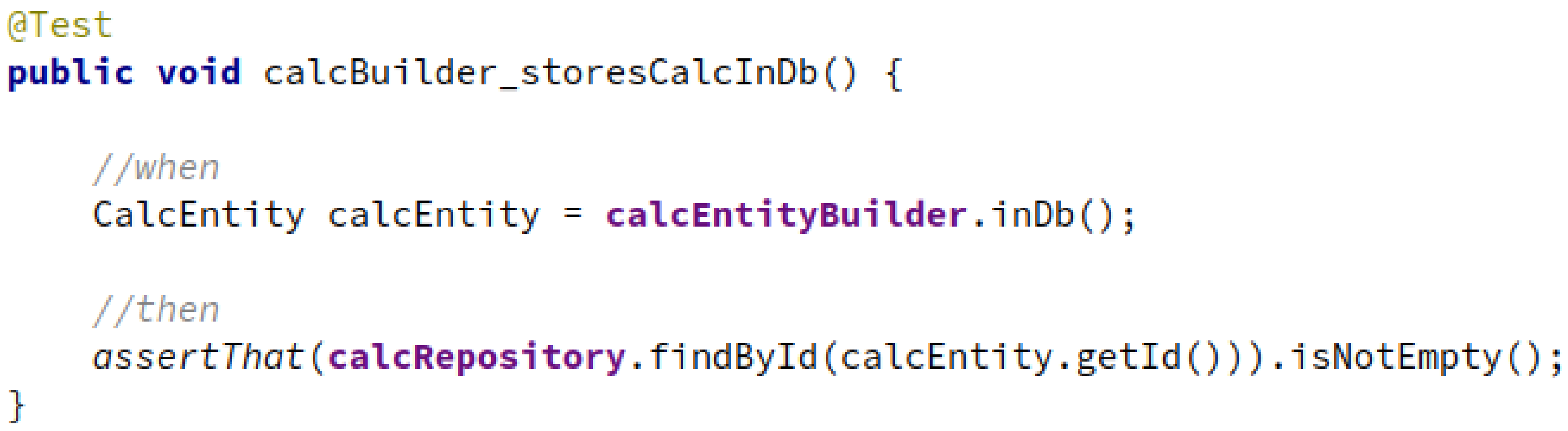


```java
CalcEntity inDb(){
    return calcRepository.save(build());
}
```



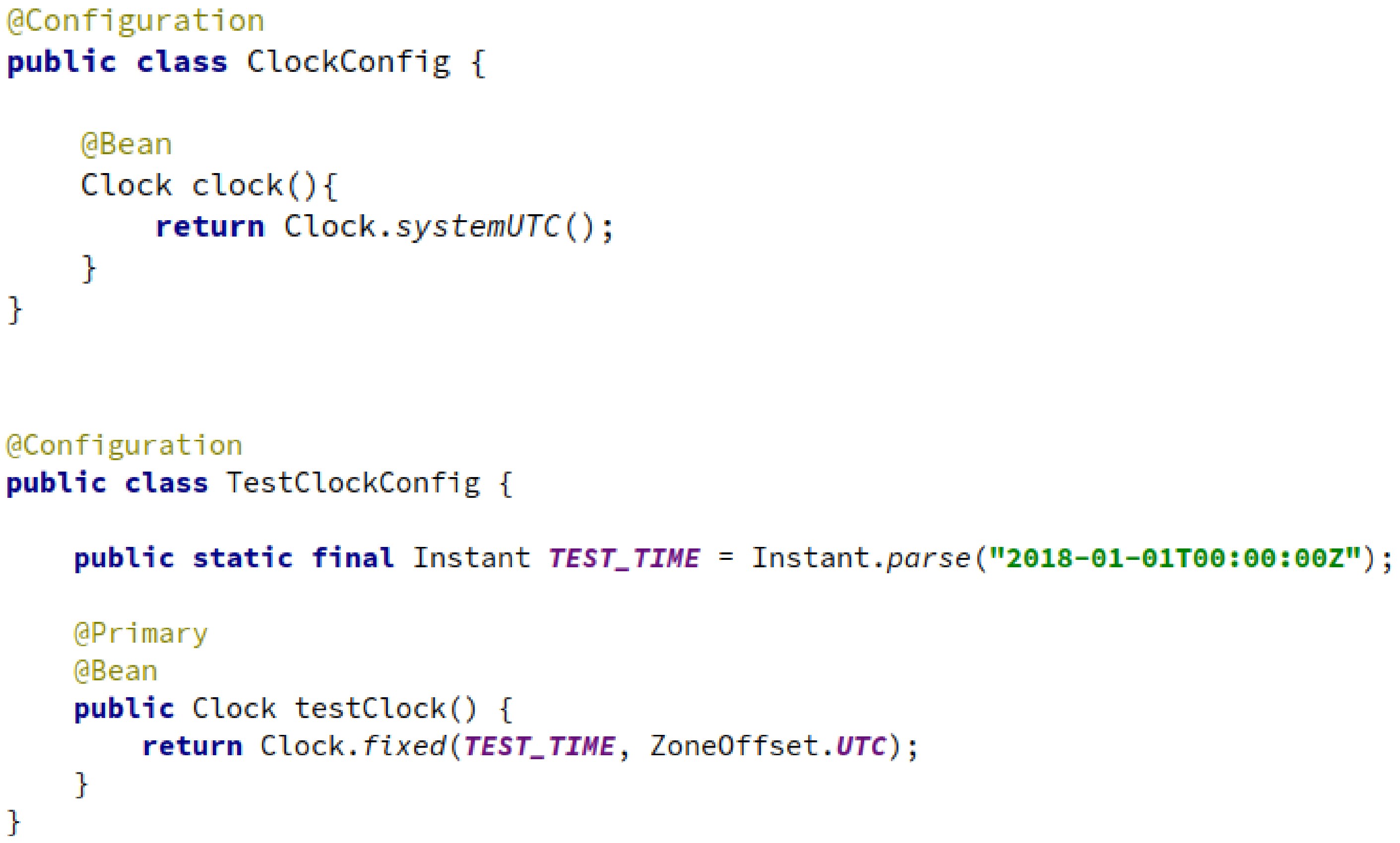
```java
calcEntityBuilder = CalcEntityBuilder.aCalcEntity(calcRepository);
```

# SAME MINDSET IN ALL TEST LAYERS



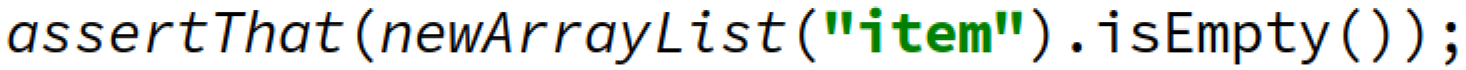
```
@Test
public void calcBuilder_storesCalcInDb() {

    //when
    CalcEntity calcEntity = calcEntityBuilder.inDb();

    //then
    assertThat(calcRepository.findById(calcEntity.getId())).isNotEmpty();
}
```

# TIME



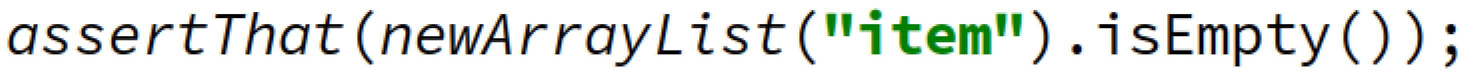
```java
@Configuration
public class ClockConfig {

    @Bean
    Clock clock(){
        return Clock.systemUTC();
    }
}


@Configuration
public class TestClockConfig {

    public static final Instant TEST_TIME = Instant.parse("2018-01-01T00:00:00Z");

    @Primary
    @Bean
    public Clock testClock() {
        return Clock.fixed(TEST_TIME, ZoneOffset.UTC);
    }
}
```

# TDD

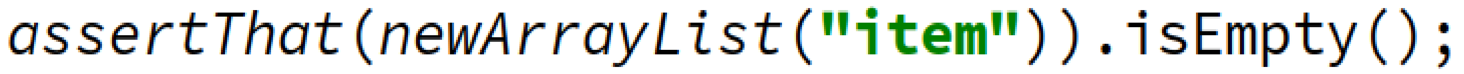


```
assertThat(newArrayList("item").isEmpty());
```

# TDD



```
assertThat(newArrayList("item").isEmpty());
```

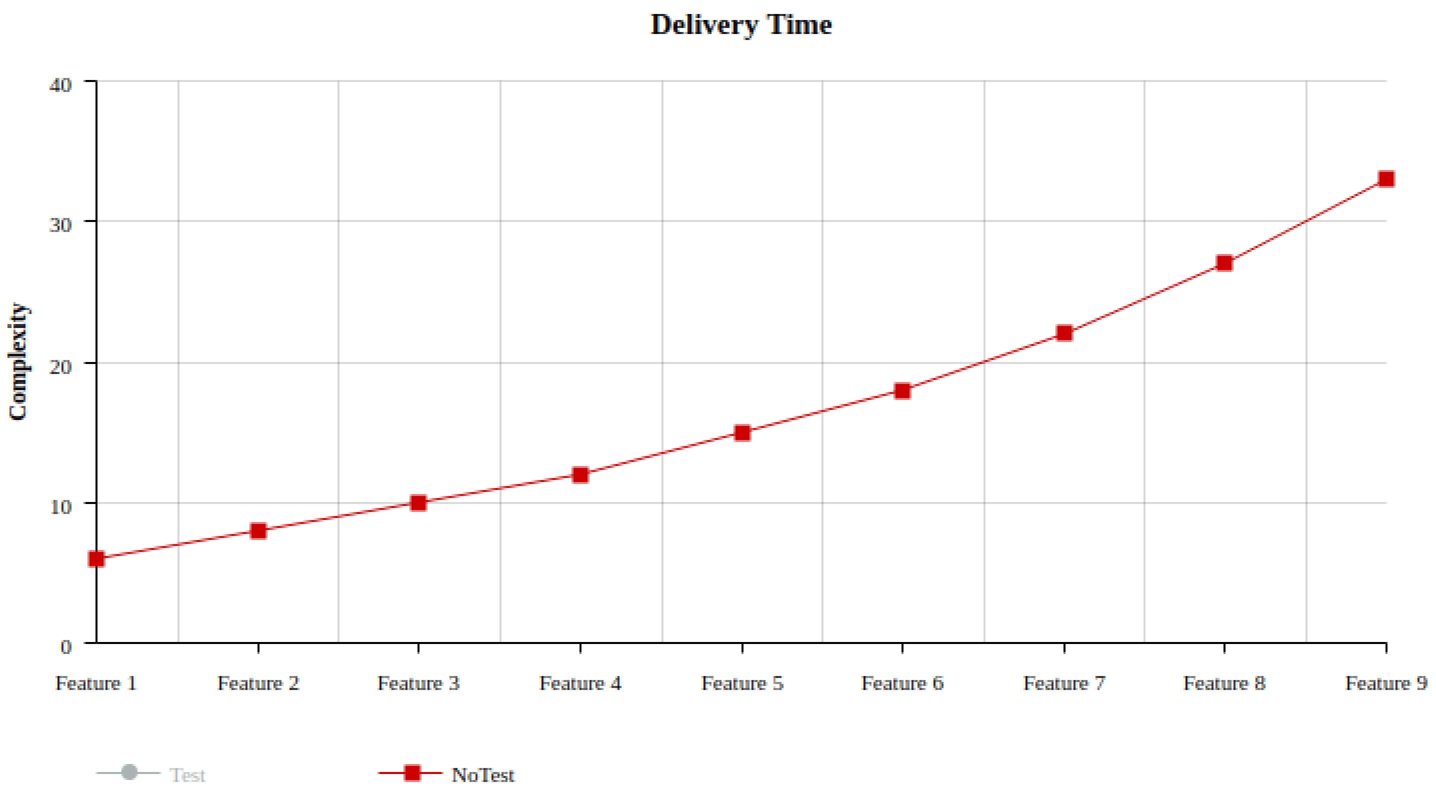


```
assertThat(newArrayList("item")).isEmpty();
```

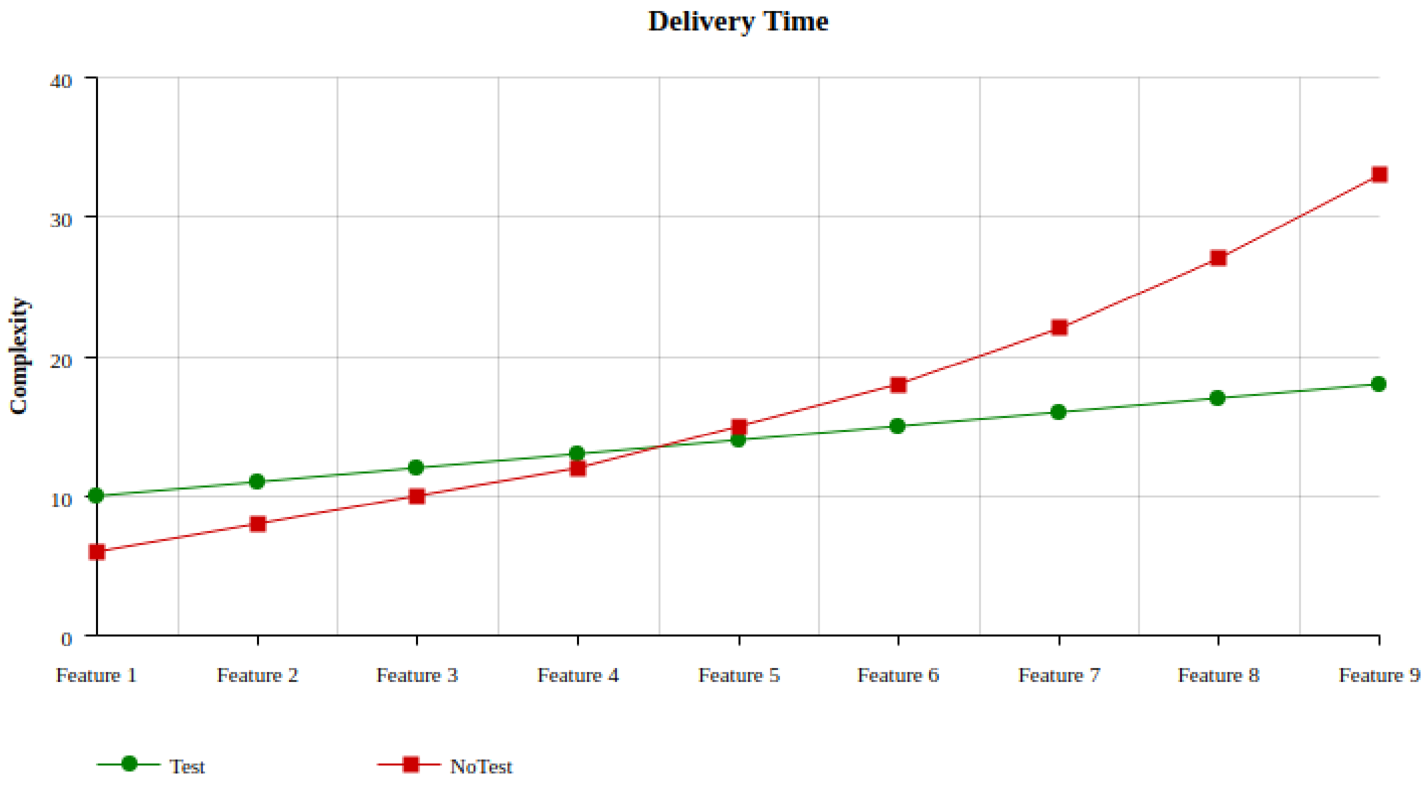
# HOW DO YOU TEST YOUR CODE?

Any answer can be correct if you can answer 'Why?'

# TESTING SPEEDS UP DEVELOPMENT



# TESTING SPEEDS UP DEVELOPMENT



# THANK YOU!

https://github.com/yacekmm/testingDemo

4Developers 2018: Modularity – the final frontier (Łukasz Szydło)

Confitura 2017: Keep IT clean: mid-sized building blocks and hexagonal architecture (Jakub Nabrdalik)

"Test Driven Development", Kent Beck