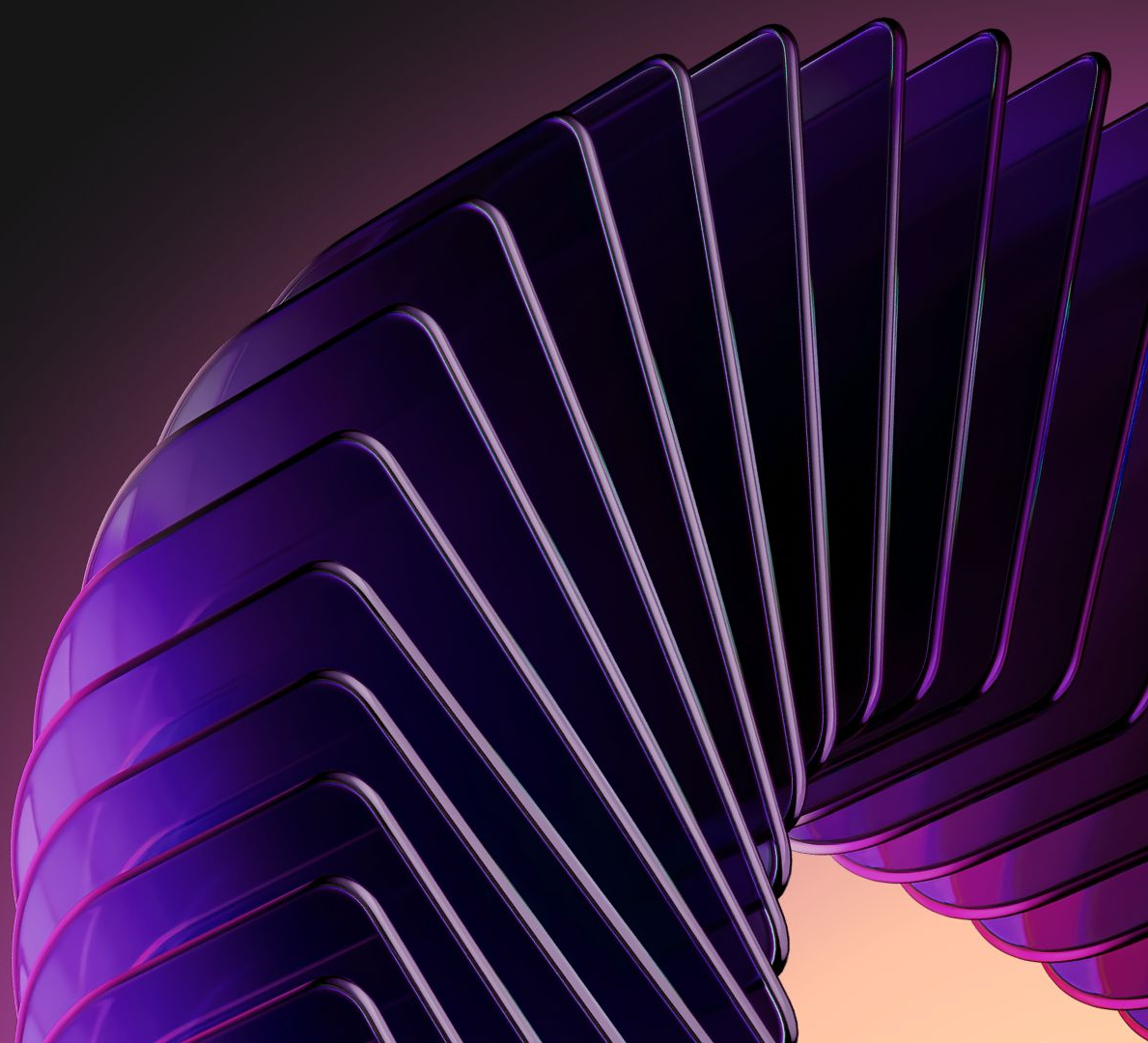
dodo engineering

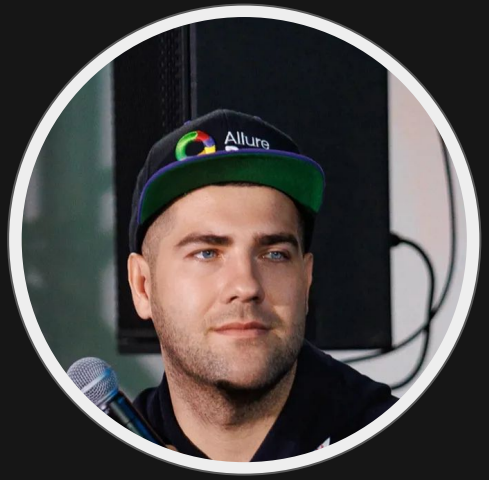
# **JUnit**: прошлое, настоящее и будущее

Москва, Heisenbug Spring`26 HB



# Дима Тучс

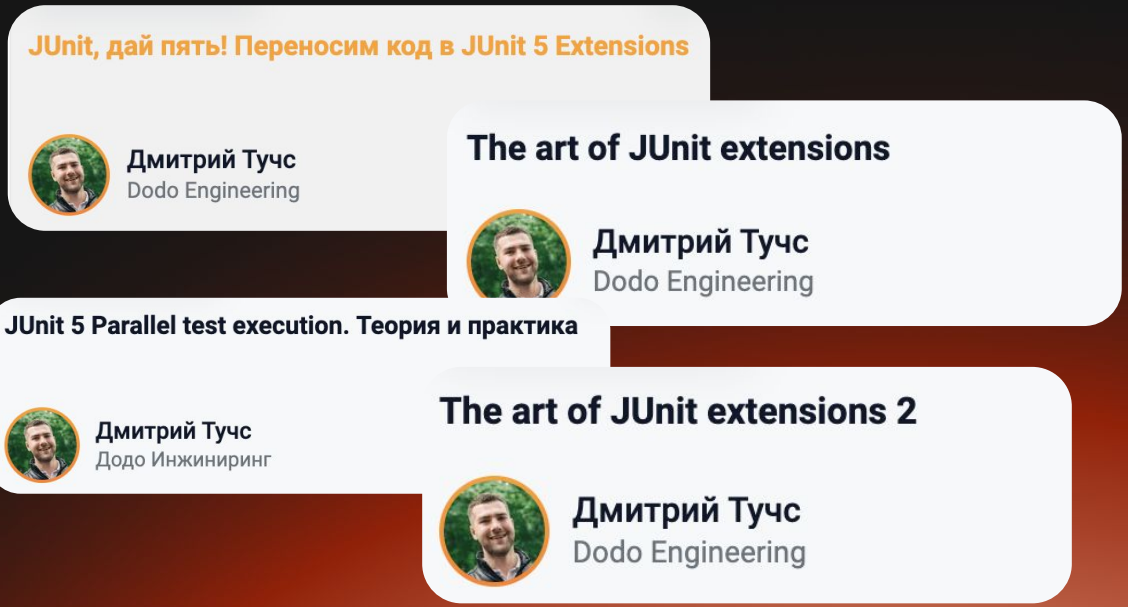
Head of QA # DODO Engineering




2009 → сегодня

Backend — Project — QA — Manager

# Опять про JUnit?!




...НО

[JUnit, дай пять! Переносим код в JUnit 5 Extensions]

[The art of JUnit extensions]

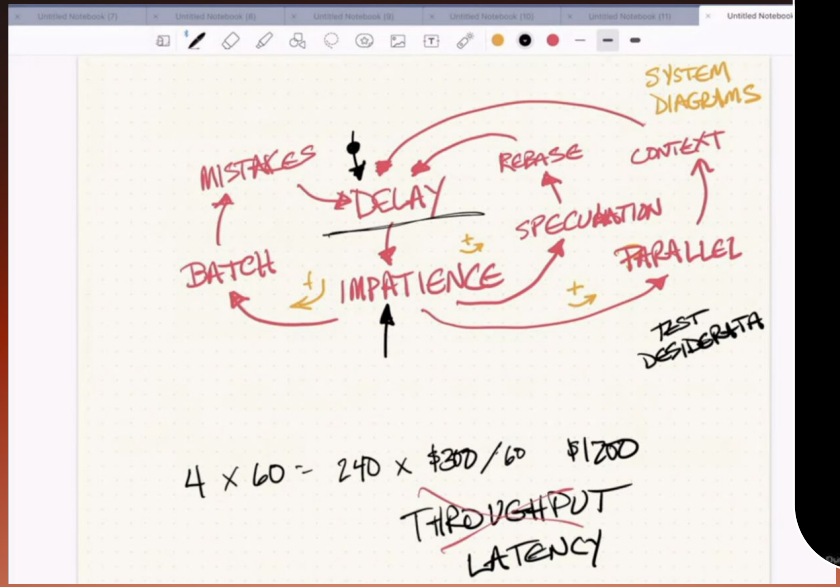
[The art of JUnit extensions 2]

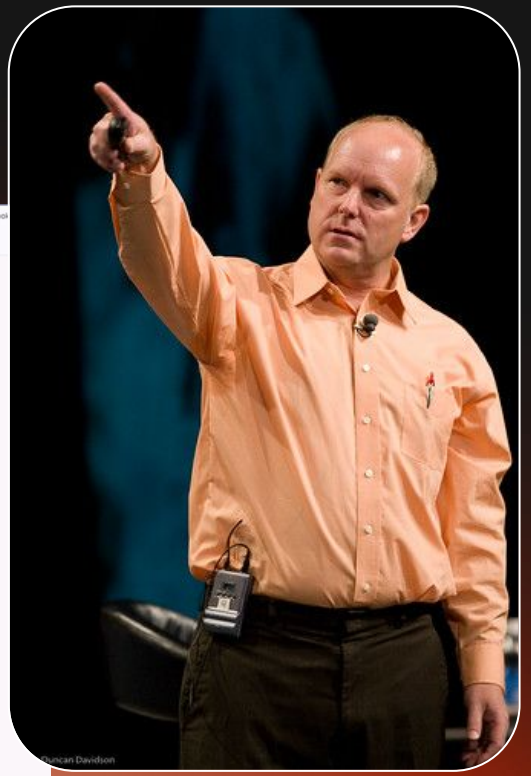
[JUnit 5 Parallel test execution. Теория и практика]

» Как часто вы смотрите на код своего *тестового фреймворка* и не понимаете, почему это написано *так*?

# Kent Beck

- OOP Patterns
- XP + TDD
- SUnit -> jUnit

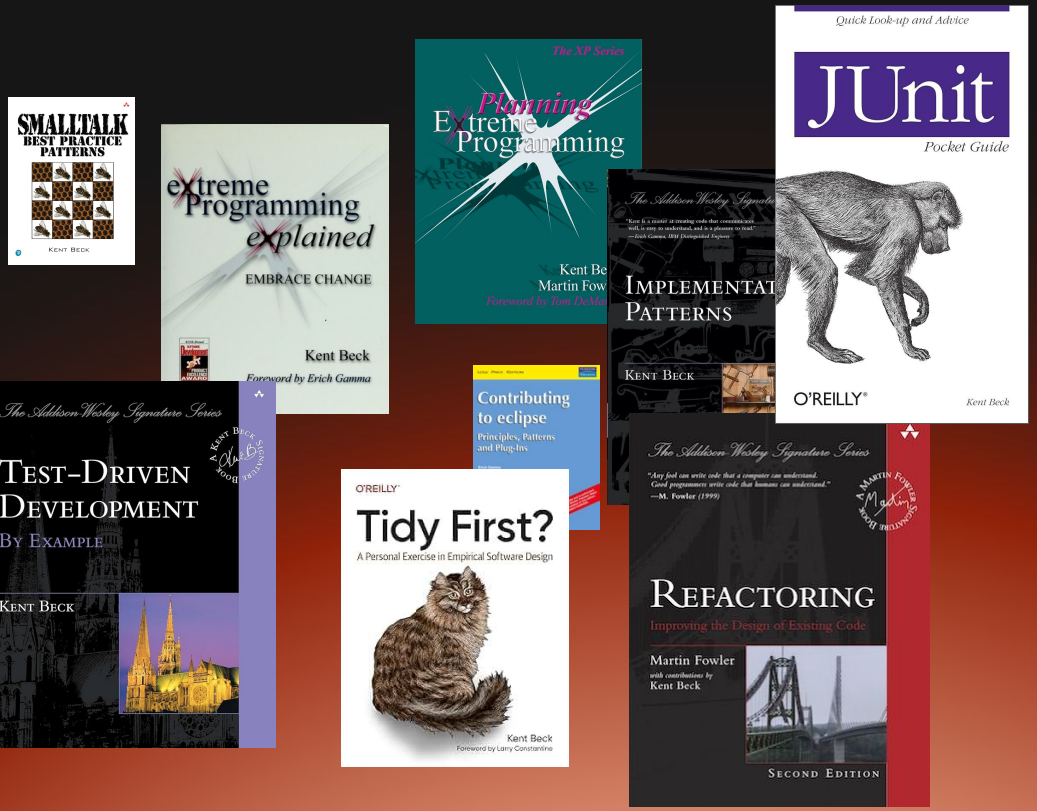


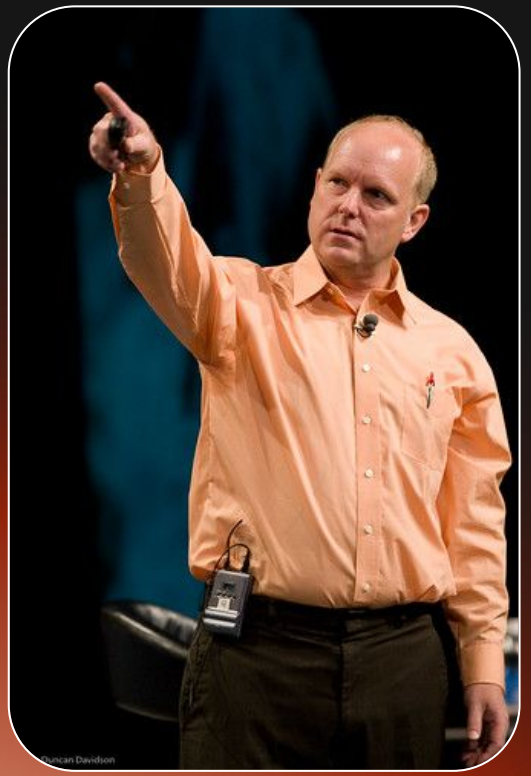



[Kent Beck — Test Run Latency]

https://en.wikipedia.org/wiki/Kent_Beck

# Kent Beck



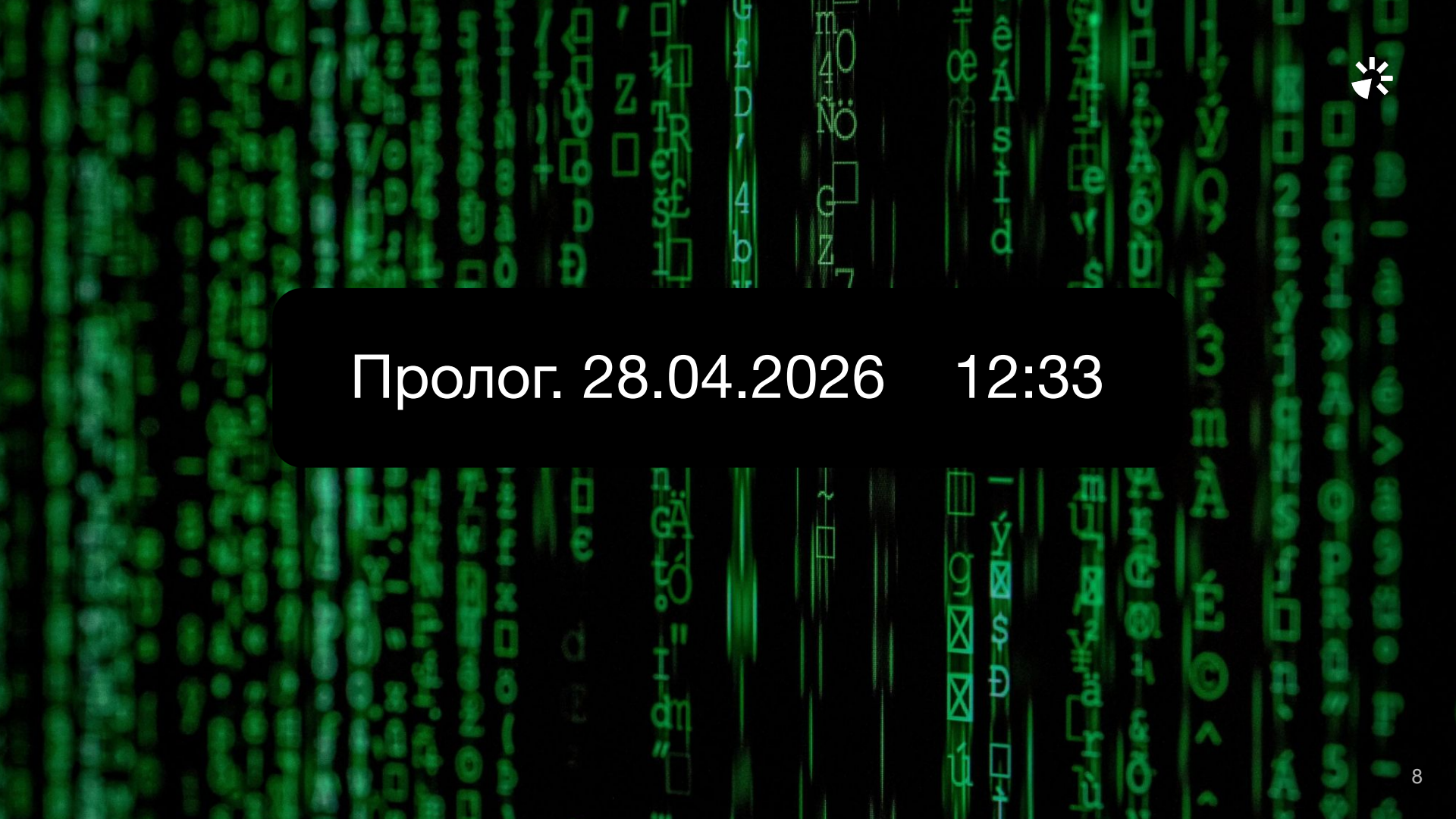



https://en.wikipedia.org/wiki/Kent_Beck

» Каждый тест должен **рассказывать историю**. У него должны быть начало, середина и конец.

» Every test should tell a story. It should have a beginning, a middle, and an end.

SE Radio 167: The History of JUnit and the Future of Testing with Kent Beck


Пролог. 28.04.2026 12:33

# Пролог


AI Best Practices

AI Tools/Frameworks

AI

AI

AI

AI

AI Tools/Frameworks

AI

AI

AI

Mobile AI

AI Tools/Frameworks

AI

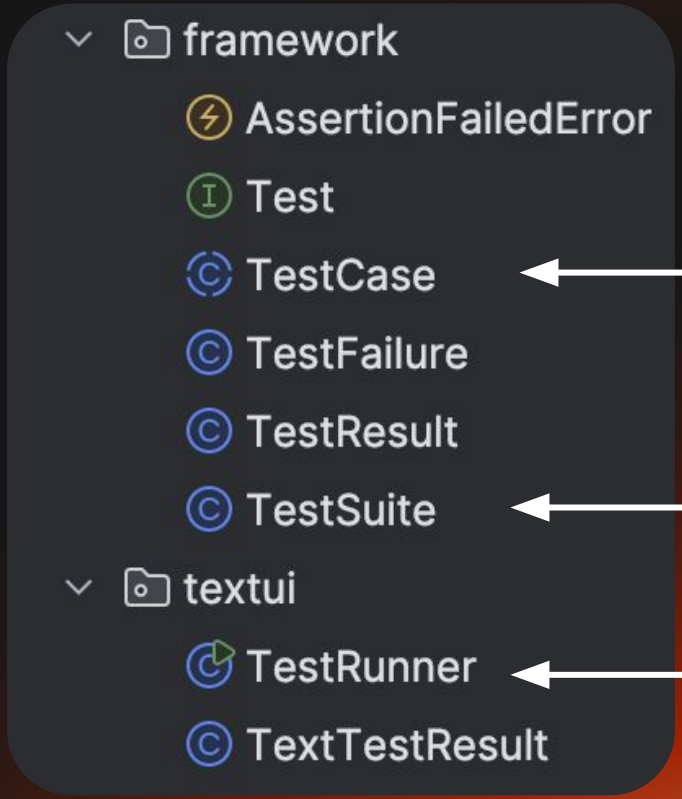
# **Глава 1**: Сколько времени нужно на написание тестового фреймворка?

» Я сказал ему: «Ну хорошо, научи меня Java». Мы сели в самолёте, и я спросил: «Какую программу напишем?»

» I said “Okay, teach me Java.” So we sat down on the airplane, and I said, “Well, what program should we write?”

Kent Beck on creating JUnit

# 1997 год: JUnit 1




API написания теста

Список `TestCase`

Метод `main`

https://github.com/dtuchs/junit-history

# JUnit 1: API TestCase

```
  /**
   * Sets up the fixture, f. e., open a network connection.
   * This method is called before a test is executed.
   */
protected void setUp() {}

  /**
   * Tears down the fixture, f. e., close a network connection.
   * This method is called after a test is executed.
   */
protected void tearDown() {}
```

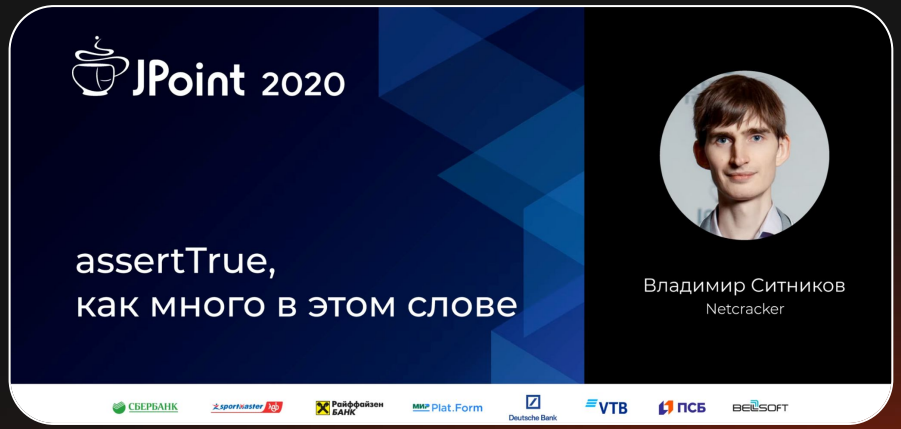
https://github.com/dtuchs/junit-history

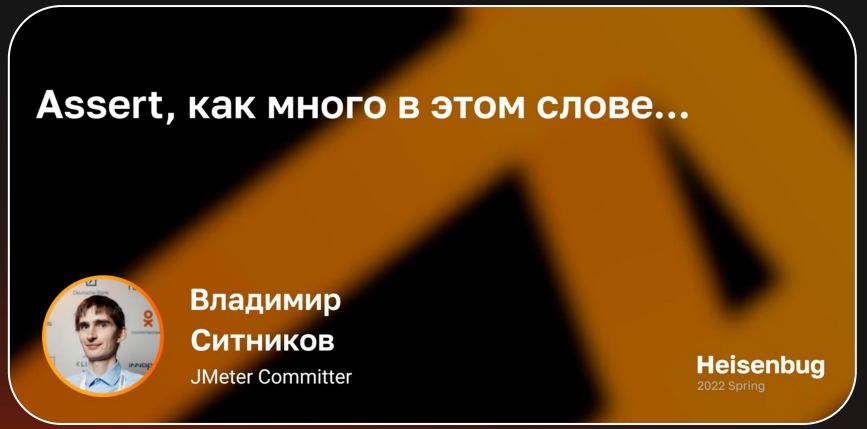
# JUnit 1: API TestCase

```
protected void assert(String message, boolean condition) {
 if (!condition)
  throw new AssertionFailedError(message);
}


protected void assertEquals(Object expected, Object actual) {
   if (!expected.equals(actual))
    assert(notEqualsMessage("", expected, actual), false);
}
```

https://github.com/dtuchs/junit-history

# Ассерты




[Владимир Ситников — assertTrue, как много в этом слове...]
[Владимир Ситников — Assert, как много в этом слове...]

# JUnit 1: Тест

```
public class VectorTest extends TestCase {
   protected Vector fEmpty = new Vector();

   public VectorTest(String name) {
       super(name);
   }

//@Override
   protected void setUp() {
     …
   }

   public void testContains() {
      assert(!fEmpty.contains(new Integer(1)));
   }
```

https://github.com/dtuchs/junit-history

# JUnit 1: Тест

```
public class VectorTest extends TestCase {
   protected Vector fEmpty = new Vector();

   public VectorTest(String name) {
       super(name);
   }

   protected void setUp() {
     …
   }

//@Test
   public void testContains() {
      assert(!fEmpty.contains(new Integer(1)));
   }
```

https://github.com/dtuchs/junit-history

# JUnit 1: Список тестов для запуска

```
public static Test suite() {
    TestSuite suite= new TestSuite();

    suite.addTest(new VectorTest("testContains"));
    suite.addTest(new VectorTest("testClone"));
    suite.addTest(new VectorTest("testRemoveAll"));
    return suite;
}
```

//Главное, не опечататься

https://github.com/dtuchs/junit-history

# JUnit 1: Поиск сьюта…

```
public static void main(String argv[]) {
    String testCase= argv[i];
    Class testClass= Class.forName(testCase);

    Method suiteMethod=
              testClass.getMethod("suite", new Class[0]);

    Test suite= (Test)suiteMethod.invoke(
                                        null, new Class[0]);
    run(suite);
}
```

https://github.com/dtuchs/junit-history

# JUnit 1: … и запуск

```
public void run(TestResult result) {
 result.startTest(this);
 setUp();
 try {
  runTest();
 }
 catch (AssertionFailedError e) {
  result.addFailure(this, e);
 }
 catch (Throwable e) {
  result.addError(this, e);
 }
 tearDown();
 result.endTest(this);
}
```

https://github.com/dtuchs/junit-history

# JUnit 1: … с разными результатами

```
public void run(TestResult result) {
 result.startTest(this);
 setUp();
 try {
  runTest();
 }
 catch (AssertionFailedError e) {
  result.addFailure(this, e);
 }
 catch (Throwable e) {
  result.addError(this, e);
 }
 tearDown();
 result.endTest(this);
}
```

https://github.com/dtuchs/junit-history

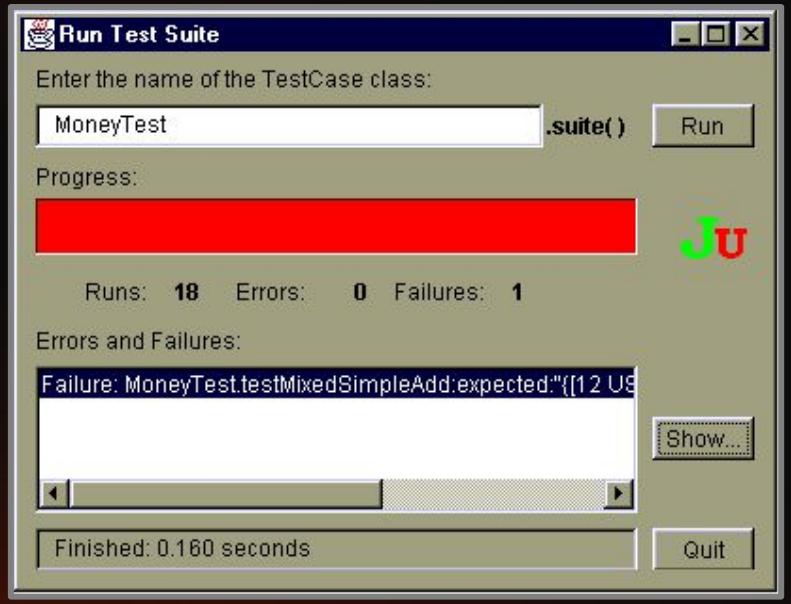
# JUnit 1: Отчет



```
public synchronized void print() {
    printHeader();
    printErrors();
    printFailures();
}
```



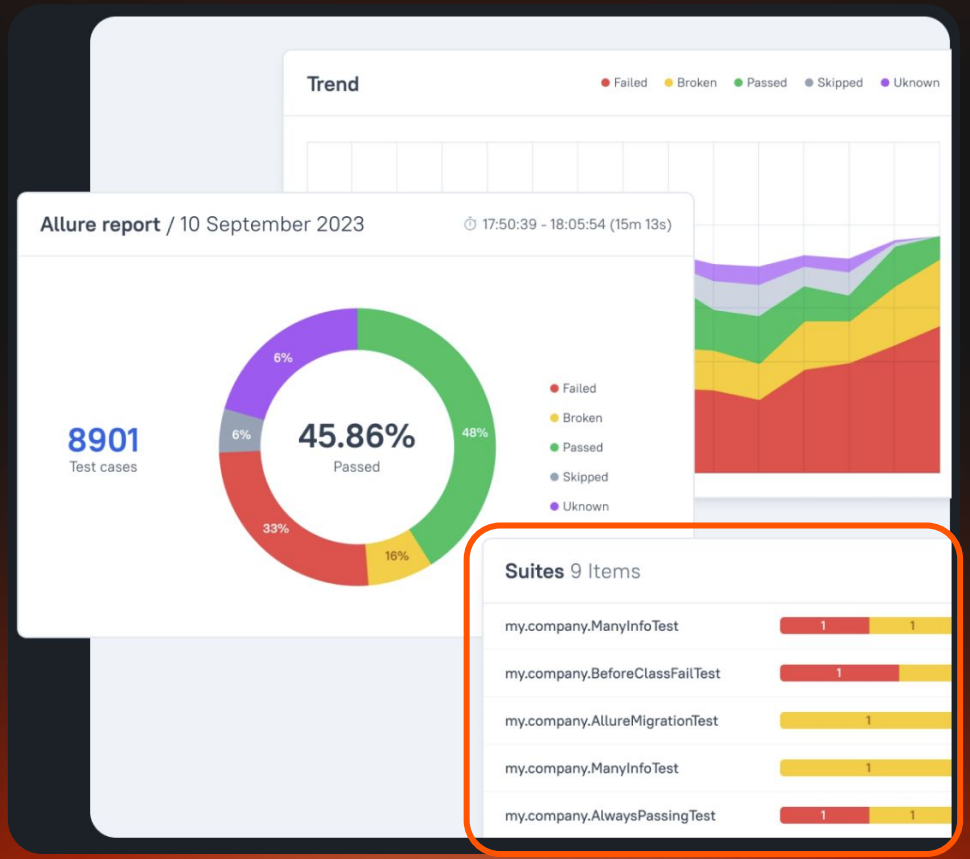




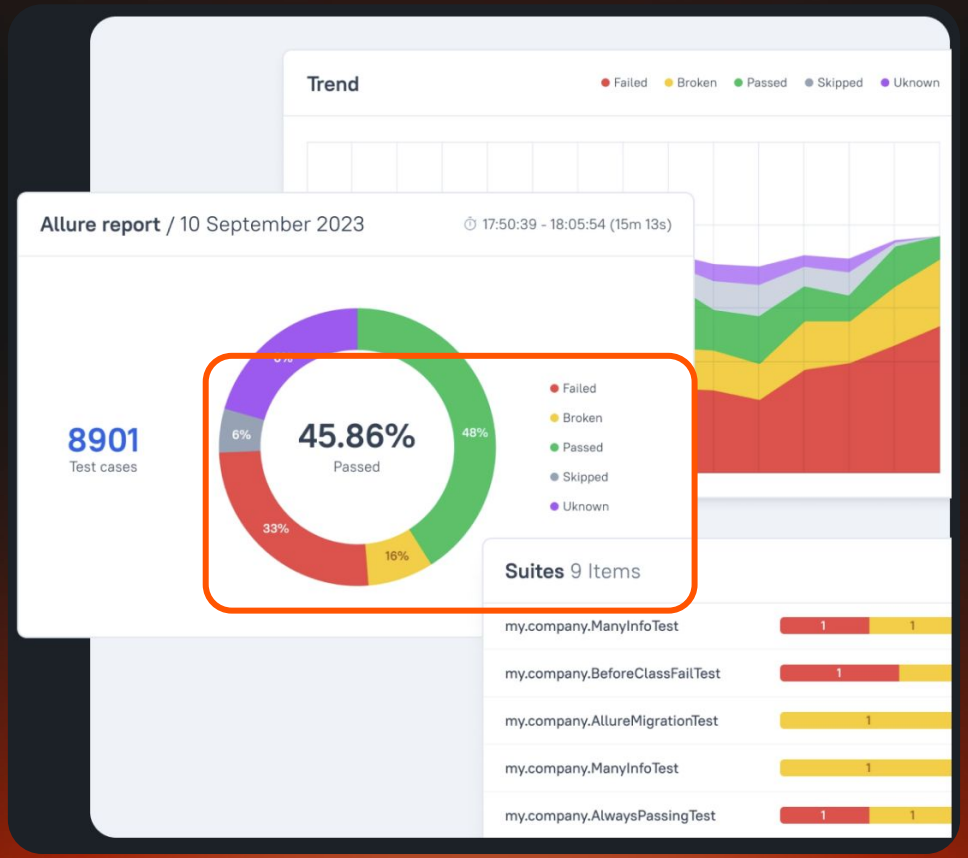

```
public void printFailures() {
    if (testFailures() != 0) {
        System.out.println(
                "There were "+testFailures()+" failures:"
                );
```

https://github.com/dtuchs/junit-history

# Отчет сейчас



# Отчет сейчас



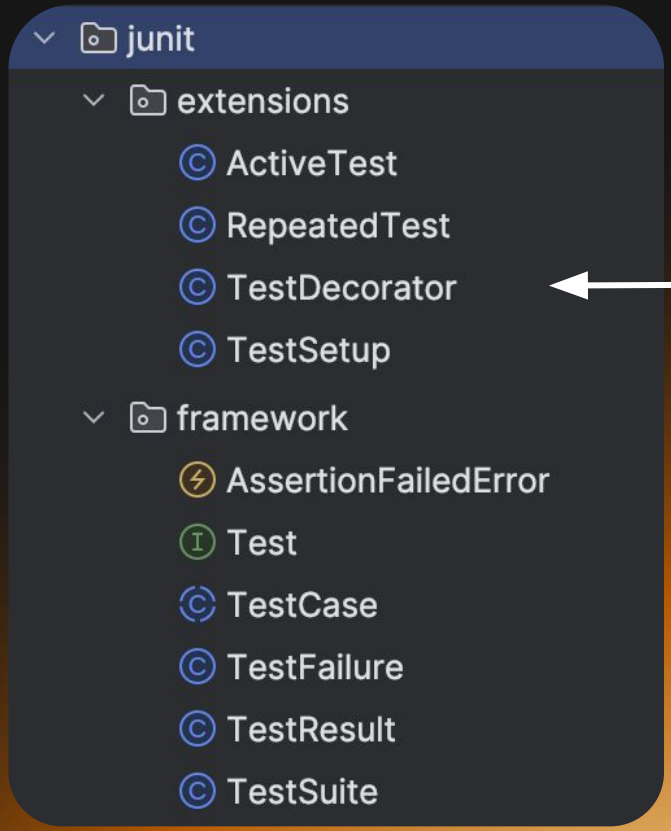
» Мы дали её Мартину Фаулеру в Атланте, он сразу попробовал и сказал: «Это круто»

» We gave it to Martin Fowler. He tried it out immediately and said, this is cool

SE Radio 167: The History of JUnit and the Future of Testing with Kent Beck

# **Глава 2**, в которой перезапуск flaky понадобился задолго до изобретения Selenium

# JUnit 2: Маленькое обновление…




Не совсем

Почти то же самое?

https://github.com/dtuchs/junit-history

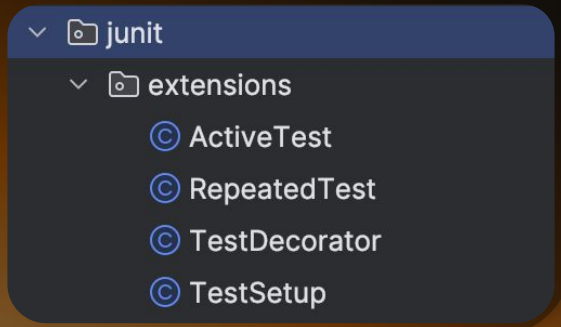
» Конечно, мы не удержались и добавили несколько функций, но постарались аккуратно вынести их в отдельный пакет расширений (test.extensions)

» Sure, we couldn’t resist adding some features but we were careful to put them into their own extensions package (test.extensions)

A Cook’s Tour of JUnit

# JUnit 2: Первая версия extensions API - декораторы

```java
/**
* A Decorator for Tests. Use TestDecorator as the base class
* for defining new test decorators. Test decorator subclasses
* can be introduced to add behaviour before or after a test
* is run.
*
*/
public class TestDecorator implements Test {

    protected Test fTest;
```




https://github.com/dtuchs/junit-history

# JUnit 2: Первая версия extensions API - декораторы

```
public class RepeatedTest extends TestDecorator {
 private int fTimesRepeat;

 public RepeatedTest(Test test, int repeat) {
  super(test);
  fTimesRepeat= repeat;
 }

 public void run(TestResult result) {
  for (int i= 0; i < fTimesRepeat; i++) {
   super.run(result);
  }
 }
```

https://github.com/dtuchs/junit-history

# JUnit 2: Первая версия extensions API - декораторы

```
@ApiLogin
@User(income = 5, outcome = 3)
@Test
void paginationTest() { .. }
```

```
new TestSuite(PaginationTest.class)
```

# JUnit 2: Первая версия extensions API - декораторы

```
@ApiLogin
@User(income = 5, outcome = 3)
@Test
void paginationTest() { .. }
```

```
new LoggedIn(
  new TestSuite(PaginationTest.class)
)
```

# JUnit 2: Первая версия extensions API - декораторы

```
@ApiLogin
@User(income = 5, outcome = 3)
@Test
void paginationTest() { .. }
```

```
new WithIncome(
  new LoggedIn(
    new TestSuite(PaginationTest.class)
  ), 1)
```

# JUnit 2: Первая версия extensions API - декораторы

```
@ApiLogin
@User(income = 5, outcome = 3)
@Test
void paginationTest() { .. }
```

```
new WithOutcome(
  new WithIncome(
    new LoggedIn(
      new TestSuite(PaginationTest.class)
    ), 1), 1)
```

# JUnit 2: Первая версия extensions API - декораторы

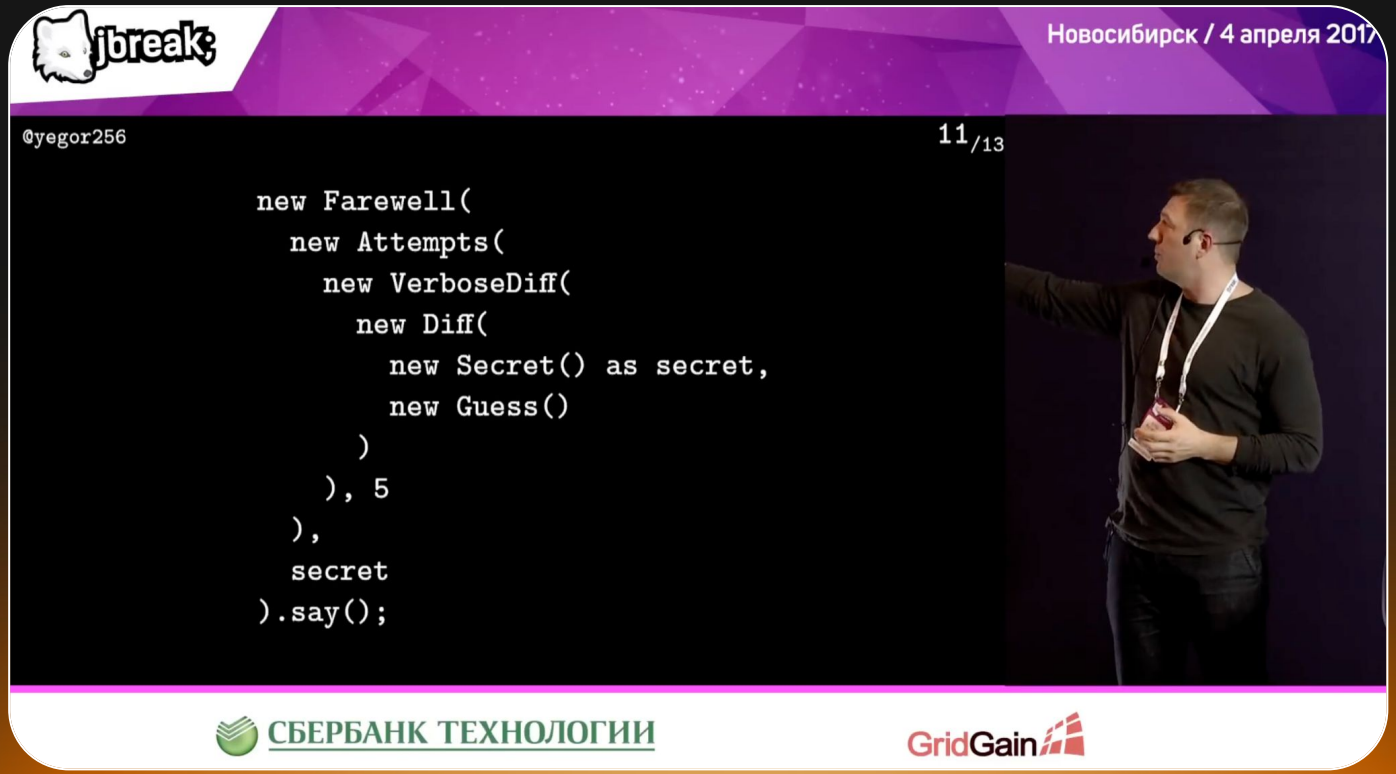
```
@ApiLogin
@User(income = 5, outcome = 3)
@Test
void paginationTest() { .. }
```

```
new WithUser(
  new WithOutcome(
    new WithIncome(
      new LoggedIn(
        new TestSuite(PaginationTest.class)
      ), 1), 1));
```

# JUnit 2: Первая версия extensions API - декораторы

```
@ApiLogin
@User(income = 5, outcome = 3)
@RepeatedTest(3)
void paginationTest() { .. }

      new RepeatedTest(
        new WithUser(
          new WithOutcome(
            new WithIncome(
              new LoggedIn(
                new TestSuite(PaginationTest.class)
              ), 1), 1)), 3);
```

# JUnit 2: ООП




Егор Бугаенко — Utility-классы нас убивают

» **Тем не менее**, проблема не только в этом:

```
), 1), 1)), 3);
```

# **Глава 3**, в которой тестовых фреймворков становится много

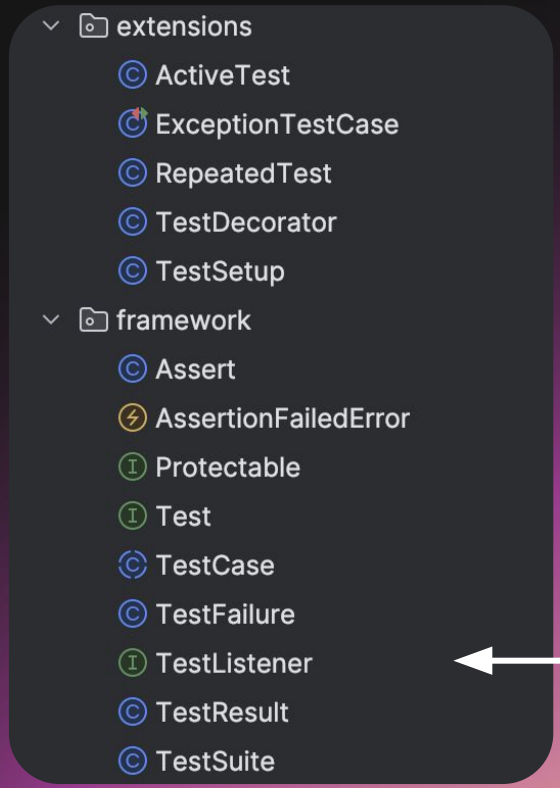
# 1998: TEST-INFECTED: PROGRAMMERS LOVE WRITING TESTS

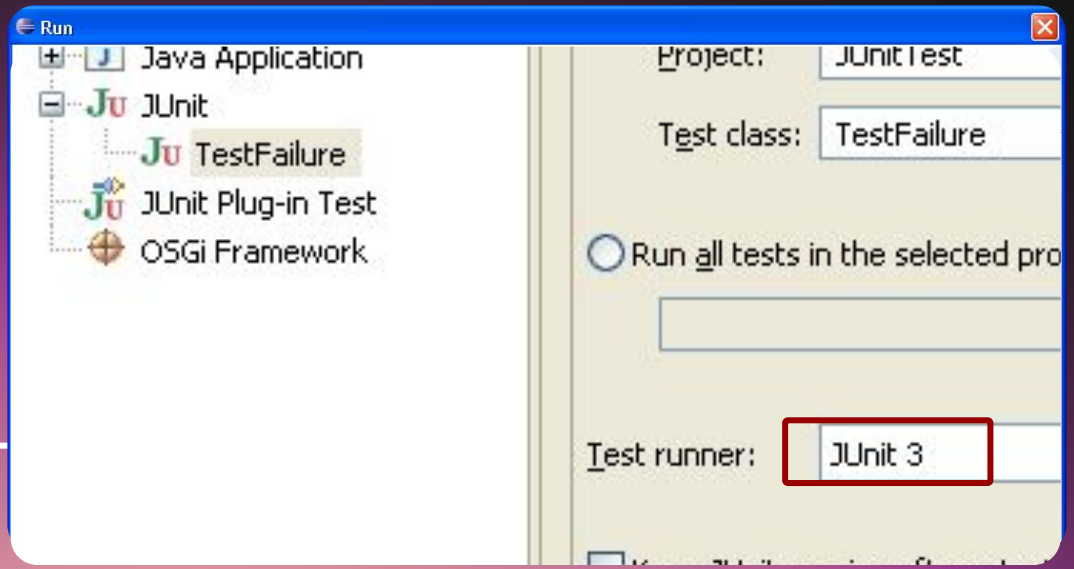
» Тестирование **не тесно** интегрировано с разработкой.

» **Testing is not** closely integrated with development.

Test Infected: Programmers Love Writing Tests, 1998

# JUnit 3: История успеха




https://help.eclipse.org/latest/index.jsp?topic=%2Forg.eclipse.jdt.doc.user%2FgettingStarted%2Fqs-junit.htm

» Чтобы писать тесты, вам нужна последняя версия JUnit (или написать **собственный эквивалент** - это не так уж сложно)

» To write tests you need to get the latest copy JUnit (or write **your own equivalent** - it's not so much work).

Test Infected: Programmers Love Writing Tests, 1998

# JUnit 3: Влияние

| PyUnit / unittest | CppUnit | NUnit / CsUnit | PHPUnit |
|---|---|---|---|
| 1999 | 2000 | 2001 | 2001 |

» То, насколько широко распространилась сама архитектура, стало для меня большим сюрпризом

» The fact that the architecture spread so far has been a big surprise to me...

Kent Beck on creating JUnit

# JUnit 3: Стандарт на многие годы

- Стандартная модель Runner + Listener, открывшая путь к IDE
- Интеграция с Ant и Maven
- Ant же создал стандарт xml-отчета (JUnit report) на 25 лет вперед

# **Глава 4**, в которой уже хотелось больше Extension-ов

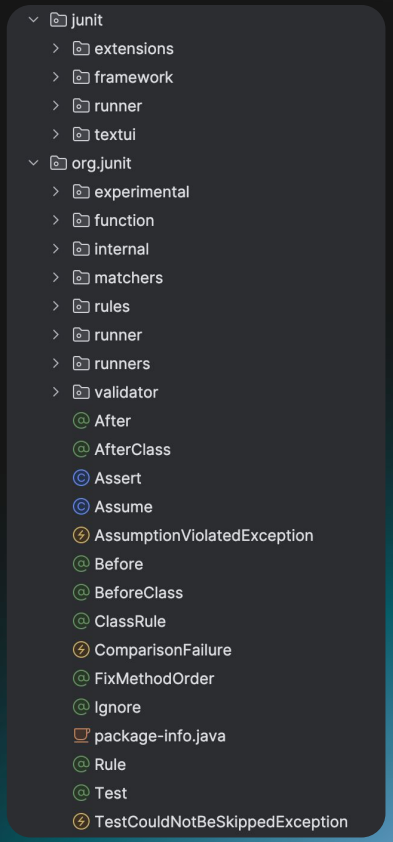
# 2006: JavaOne

» Тесты - заманчивая цель для расширений

» Test is a tempting target for extensions.

We thought we were just programming on an airplane

# JUnit 4



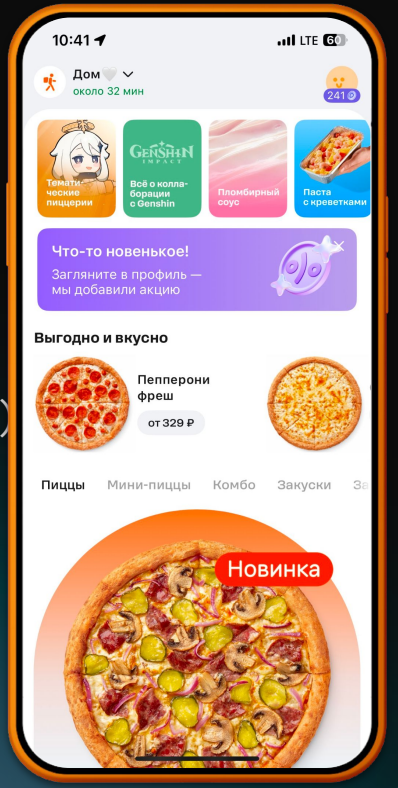

» Я очень рано увидел NUnit… увидел аннотации и подумал: это классная идея!

» I had seen NUnit very early … and seen the annotations and thought, this is a cool idea

SE Radio 167: The History of JUnit and the Future of Testing with Kent Beck

# JUnit 4: Вполне современно?

```
@RunWith(AndroidJUnit4::class)
class LoyaltyCartTest {

 @Test
 @AllureId("737278")
 @WithUser(cart =[PIZZA_PEPPERONI])
 fun deliveryRewardSum() = test(startOn =Screen.Cart)
   ShoppingCartScreen {
     verifyLoyaltyRewardSum(expected = "+90")
   }
 }
}
```



# JUnit 4: С нами и сегодня

**mannodermaus** on Oct 1, 2025

JUnit 6 for Android instrumentation tests is probably not going to happen for a long time. We are only slowly arriving at minSdk 26 (the requirement for JUnit 5) being reasonable for applications. JUnit 6 will require minSdk 35, a pretty steep ask in 2025. You'll be better off with 5 if you want to use it for on-device tests.

👍 3

## 1. Library & Tool Requirements

- **Jetpack/AndroidX Libraries:** Since June 2025, many libraries require **API Level 23 (Android 6.0)** or higher.
- **Google Play Services:** Version 24.28+ requires **API Level 23** or higher. Older versions (v23.30.99+) previously required API Level 21.
- **Flutter:** As of the 3.22 stable release, Flutter no longer supports Android KitKat (API 19) and requires at least **API Level 21 (Android 5.0)**. Newer Flutter updates (starting June 2025) are moving toward a minimum of **API Level 24**. Android API Levels +4

# JUnit 4: Rules

» Нужно придумать простой способ объяснить: Когда запускается тест, создается цепочка объектов... И с помощью rules ты можешь вставлять объекты в эту цепочку.

» When your test runs this chain of objects is created ... Well, you can insert, using rules, you can insert objects into this chain.

SE Radio 167: The History of JUnit and the Future of Testing with Kent Beck

# JUnit 4: Rules

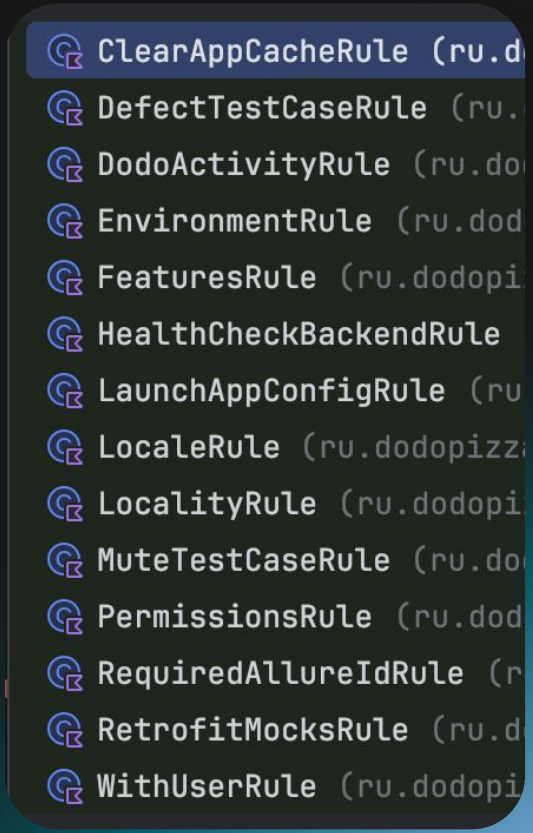
```
public interface TestRule {

    Statement apply(Statement base, Description description);
}
```

```
//Тест или сьют, запуск которого будет
передан далее через вызов метода
evaluete();
```

# JUnit 4: Rules

```
public abstract class TestWatcher implements TestRule {
   public Statement apply(Statement base, Description description) {
       return new Statement() {
           @Override public void evaluate() throws Throwable {
               starting(description, errors);
               try {
                   base.evaluate();

                   succeeded(description, errors);
               } catch (AssumptionViolatedException  e) {
                   skipped(e, description, errors);
               } catch (Throwable e) {
                   failed(e, description, errors);
               } finally {
                   finished(description, errors);
               }
           }};}
```

# JUnit 4: Rules




- Очищаем кэш приложения
- Ищем дефект в TestOps и фэйлим тест
- Решаем, с какого экрана стартовать
- На каком окружении тестируем
- На каких тоглах
- Проверяем, жив ли бэкенд перед тестами
- Кастомный DSL конфигурации запуска
- Язык и локация в приложении
- Скипаем тест, если он в карантине
- Доступ к системным API Android
- Просто проверка, есть ли аннотация
- Замокать конкретный ответ
- Создать пользователя…

# JUnit 4: Rules

» Я всегда стремлюсь к декларативной форме тестов.

» I always strive for a kind of declarative expression in my tests

SE Radio 167: The History of JUnit and the Future of Testing with Kent Beck

# JUnit 4: Rules

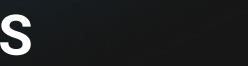
```
@WithMock(GET_CART_SHARING_PREVIEW, code = 404)
fun openOutdatedSharedCart() = test(startOn = Cart) {
```

```
@WithMock(
    endpoint = POST_ACTUALIZE,
    paths = [Path(FILE, path = "$.delivery", value =
"delivery.json")]
)
class DeliveryIncreaseCartTest
```

# JUnit 4: Rules

```
class WithUserRule : BaseRule() {

 private companion object {
   private const val WAIT_AFTER_RESET_MAPI_CACHE_MILLIS = 2_000L
 }

 private val user = UserStorage
 private val settings = SettingsStorage
```

# TEST-INFECTED: PROGRAMMERS LOVE WRITING TESTS



» Тестирование ~~не~~ **тесно** интегрировано с разработкой.

» **Testing is not** closely integrated with development.

JUnit

[Test Infected: Programmers Love Writing Tests, 1998]

# **Глава 5**: вы находитесь здесь

# 2015 год: сбор средств на разработку JUnit 5

# JUnit 5: Предпосылки

» Правила созданы для того, чтобы их нарушать

» Rules… are meant to be broken

JUnit 5: New Opportunities for Testing on the JVM

# JUnit Extensions

» Фреймворк должен быть пересечением возможностей, а не объединением всех возможностей

» The framework should be the intersection of features, not the union of features

Kent Beck on creating JUnit

# JUnit 4 - объединения не получилось

```java
@ContextConfiguration(locations =
{"classpath:context-test.xml"})
@RunWith(SpringJUnit4ClassRunner.class)
public class SpringJUnit4Test {
 @Resource(name="testBean")
 private TestBean bean;

 @Test
 void beanTest() {...}
}
```

```kotlin
@RunWith(Parameterized::class)
class CancelOrderTest(
 private val canBeCancelled: Boolean,
) : BaseTest() {

 companion object {
   @JvmStatic @Parameters(name = "{0}")
   fun data() = listOf(true, false)
 }
```

# JUnit 4 - объединения не получилось

```
@ContextConfiguration(locations =
{"classpath:context-test.xml"})
@RunWith(SpringJUnit4ClassRunner.class)
public class SpringJUnit4Test {
 @Resource(name="testBean")
 private TestBean bean;

 @Test
 void beanTest() {...}
}
```

OR

```
@RunWith(Parameterized::class)
class CancelOrderTest(
 private val canBeCancelled: Boolean,
) : BaseTest() {

 companion object {
   @JvmStatic @Parameters(name = "{0}")
   fun data() = listOf(true, false)
 }
```

# 2016 год: первые RC JUnit 5

```
class SimpleTest {

    @Test
    @DisplayName("1 + 1 = 2")
    void myFirstTest() {
        assertEquals(2, 1 + 1);
    }

    @Test
    @Disabled("for some reason")
    void anotherTest() {
        assertEquals("", 1 + 1);
     }
```

JUnit 5 at Devoxx US 2017

# 2016 год: первые RC JUnit 5

```
@TestFactory
@DisplayName("all Fibonacci numbers are odd")
Stream<DynamicTest> testFactory() {
    return IntStream.range(1, 20)
            .map(this::fibonacci)
            .mapToObj(aFibonacci ->
                dynamicTest("Fibonacci = " + aFibonacci, () ->
isOdd(aFibonacci)));
 }
```

JUnit 5 M1

# 2016 год: первые RC JUnit 5

## Extension Points

- **Conditional Test Execution**
  - `ContainerExecutionCondition`
  - `TestExecutionCondition`
- **General Purpose**
  - `TestInstancePostProcessor`
  - `TestExecutionExceptionHandler`
  - `ParameterResolver`
- **Test Lifecycle Callbacks**
  - `BeforeAllCallback`
    - `BeforeEachCallback`
      - `BeforeTestExecutionCallback`
      - `AfterTestExecutionCallback`
    - `AfterEachCallback`
  - `AfterAllCallback`

JUnit 5 at Devoxx US 2017

# 2016 год: первые RC JUnit 5

```
public class DisabledOnConference implements TestExecutionCondition {
 @Override
 public ConditionEvaluationResult evaluate(TestExtensionContext context){
    return LocalDate.now().getDayOfWeek() == THURSDAY
        ? disabled("You're at a conference")
        : enabled("Get some work done!!!");
    }
 }
```

JUnit 5 at Devoxx US 2017

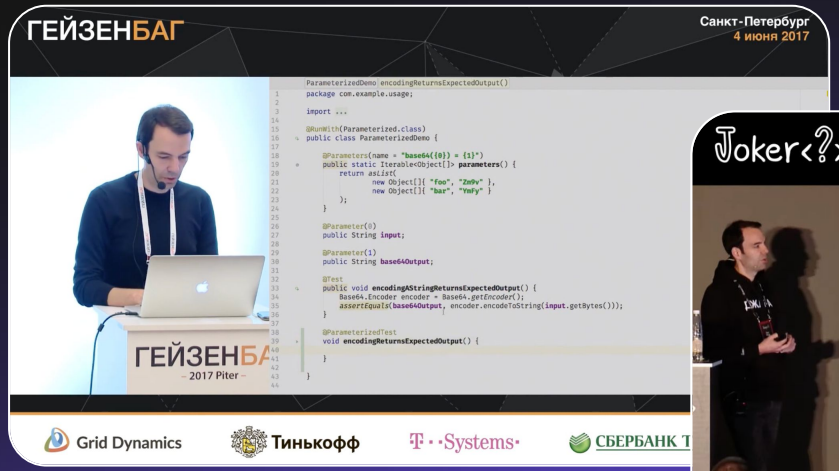
# 2016 год: первые RC JUnit 5

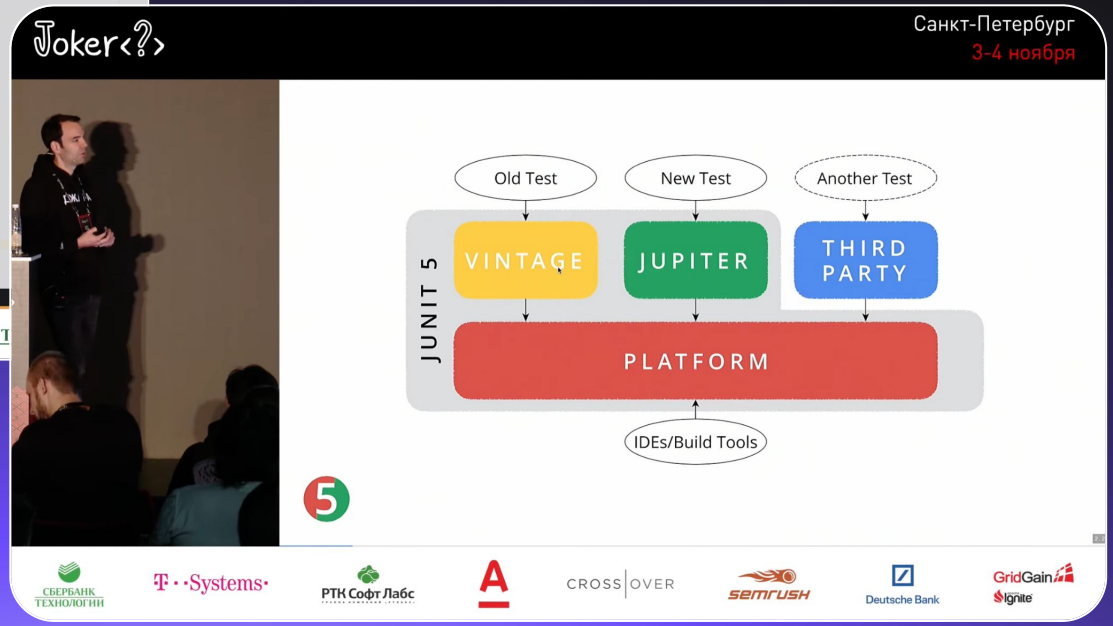
```
public class TmpDirEx implements ParameterResolver, AfterEachCallback {
  @Override
  public boolean supports(ParameterContext parameterCtx,
                          ExtensionContext extensionCtx) {
    return parameterCtx.getParameter().getType()
                       .equals(Path.class);
  }

  @Override
  public Object resolve(ParameterContext parameterCtx,
                        ExtensionContext extensionCtx) {
    return extensionCtx.getStore()
                       .getOrComputeIfAbsent("root", key -> createTmpDir());
  }
```

JUnit 5 at Devoxx US 2017

# 2017 год: релиз и быстрый успех



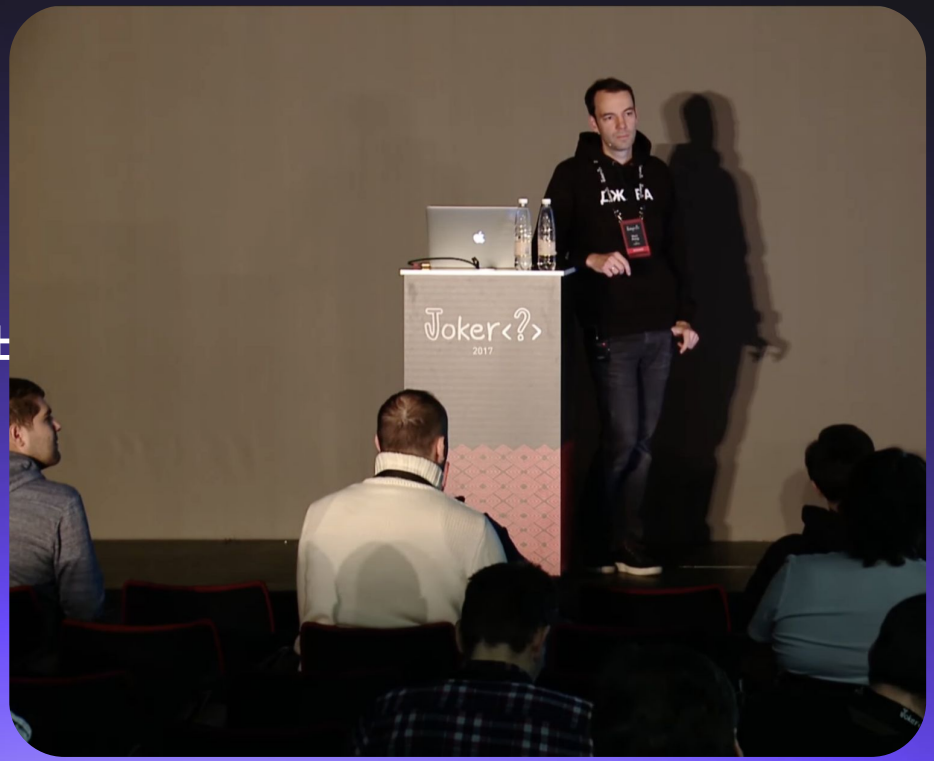

[JUnit 5 — The New Testing Framework for Java and Platform for the JVM]

[JUnit 5 extensions: from conditional test execution to test templates]

# 2017 год: в этой истории появляюсь я



» В этом что-то есть



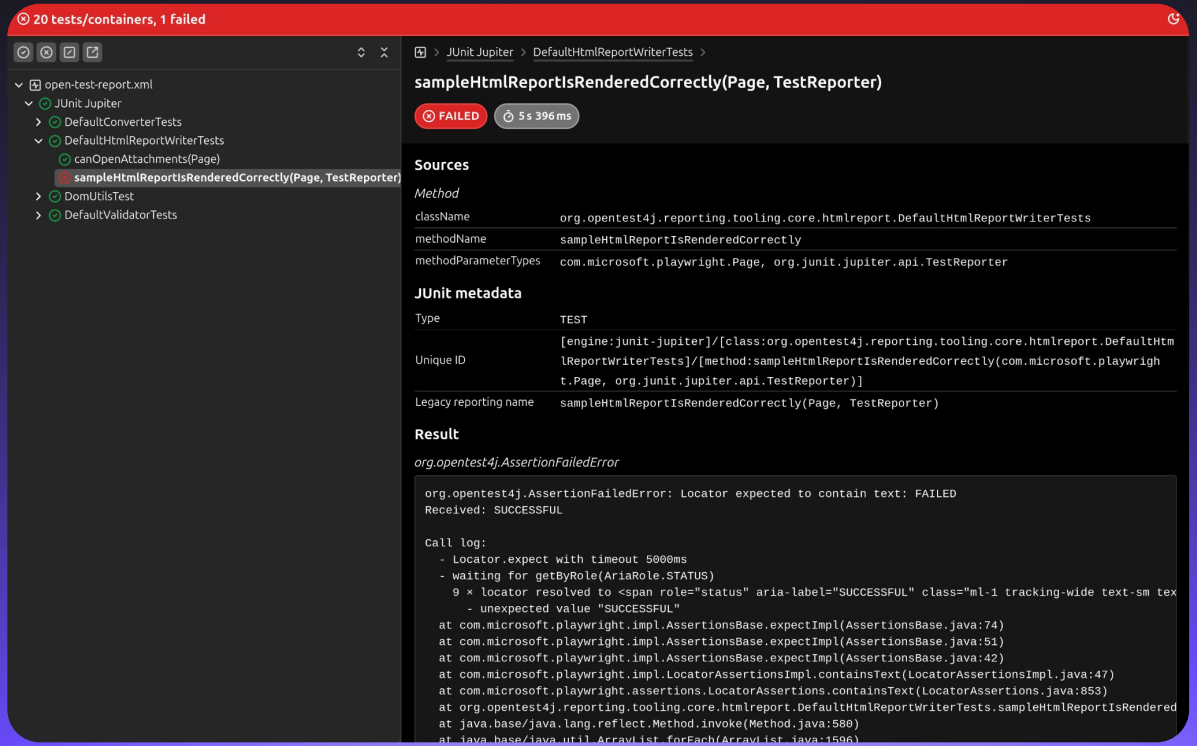

JUnit 5 extensions: from conditional test execution to test templates

# JUnit 5 теперь с нами надолго

- 2017 - первая версия spring-test-junit5;
- **Mockito** - MockitoExtension;
- **Testcontainers** - testcontainers-junit-jupiter;
- **WireMock** - WireMockExtension;
- **Playwright** - @UsePlaywright;
- **Selenide**, **Allure**, и т.д.

# JUnit 5: open-test-reporting

`junit.platform.reporting.open.xml.enabled=true`



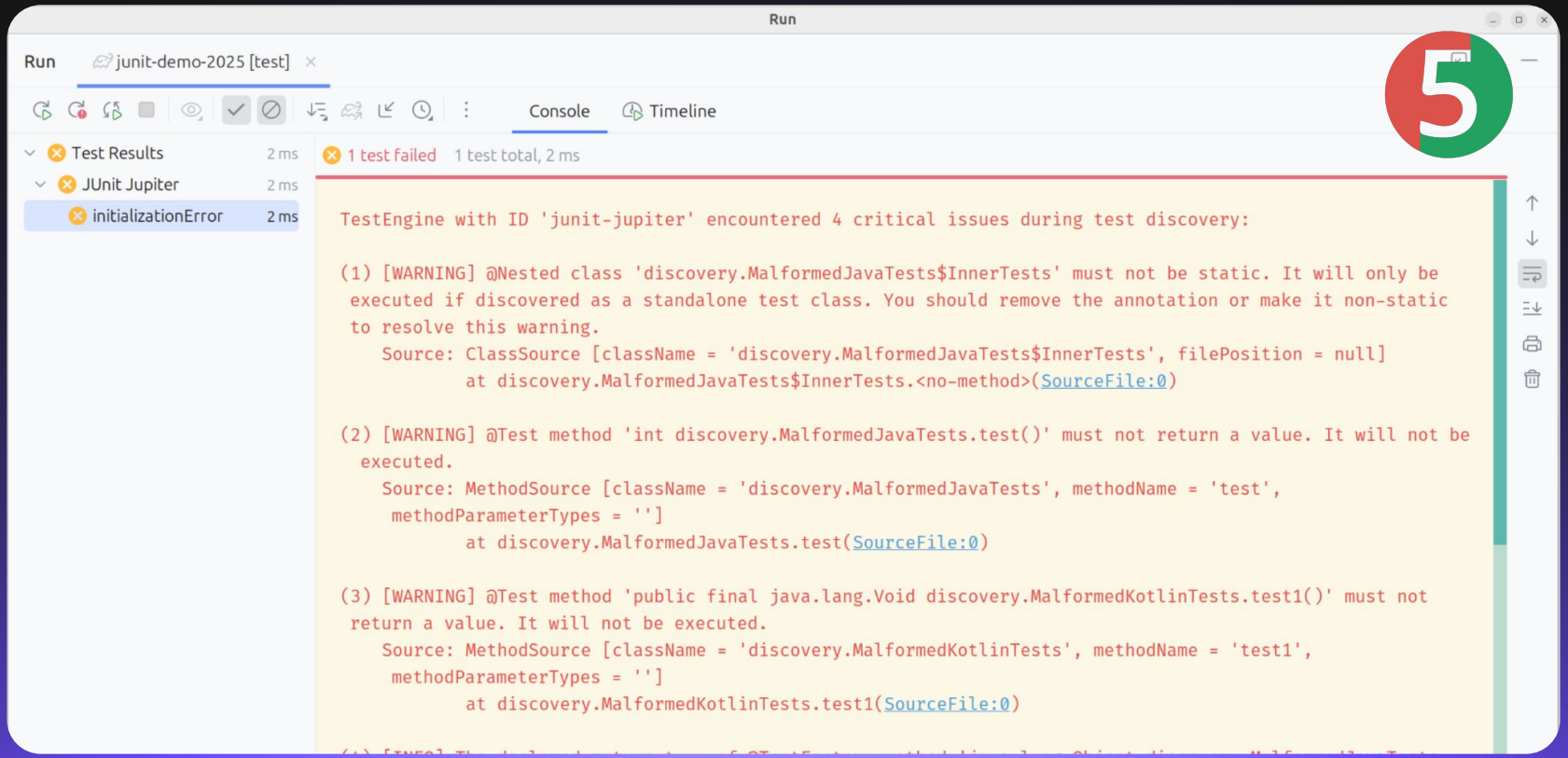

https://github.com/ota4j-team/open-test-reporting

# JUnit 5 сегодня: Discovery issues

```
fail("Method \""+fName+"\" should be public");

warning("No tests found in "+theClass.getName())
```

# JUnit 5 сегодня: Discovery issues



# JUnit 5 сегодня: Suite классы

```
@Suite
@SuiteDisplayName("Smoke сценарии")
@SelectClasses({
    LoginTest.class,
    OrderTest.class,
    MenuTest.class
})
class SmokeSuite {
}

@SelectPackages("com.example.tests")
@IncludeTags("smoke")
@IncludeClassNamePatterns(".*IT|.*SmokeTest")
```

JUnit Platform Suite Engine

# JUnit 5 сегодня: Suite методы

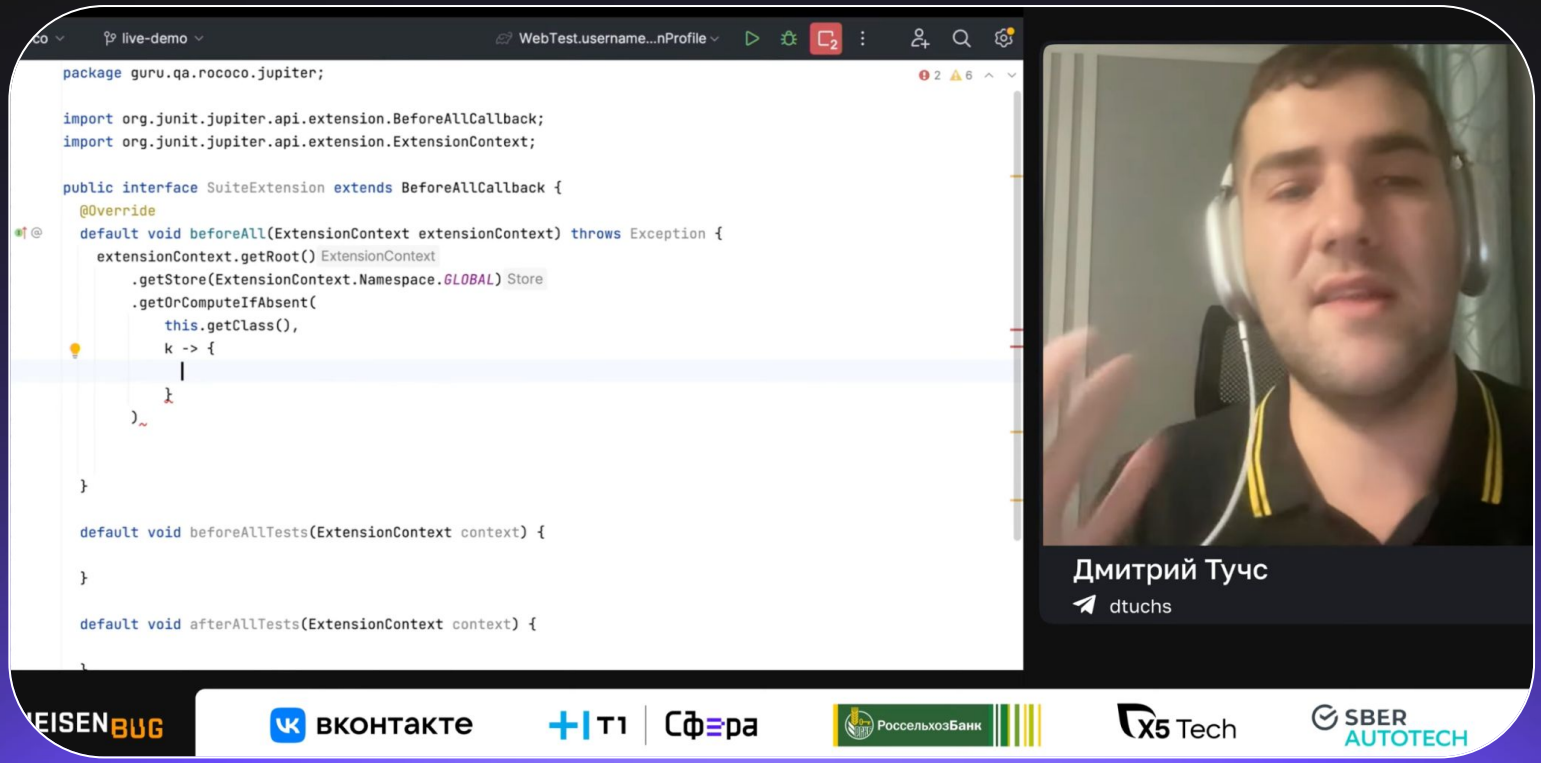
```
@Suite
@SelectPackages("com.example.tests")
class RegressSuite {

  @BeforeSuite
  static void beforeSuite() {
  }

  @AfterSuite
  static void afterSuite() {
  }
}
```

JUnit Platform Suite Engine

# JUnit 5 сегодня: Suite методы




[The art of JUnit extensions]

# JUnit 5 сегодня: ClassTemplate

```
@ClassTemplate
@ExtendWith(UserRepositoryInvocationProvider.class)
class UserRepositoryTest {

  UserRepository repository;

  @Test
  void shouldSaveUser() { }

  @Test
  void shouldUpdateUser() { }
}
```

Providing Invocation Contexts for Class Templates

# JUnit 5 сегодня: ParametrizedClass

```
@ParameterizedTest
@ValueSource(strings = { "racecar", "radar" })
void palindromes(String candidate) {}


@ParameterizedTest
@ValueSource(strings = { "racecar", "radar" })
void reversePalindrome(String candidate) {}
```

Parameterized Classes and Tests

# JUnit 5 сегодня: ParametrizedClass

```
@ParameterizedClass
@ValueSource(strings = { "racecar", "radar" })
class PalindromeTests {

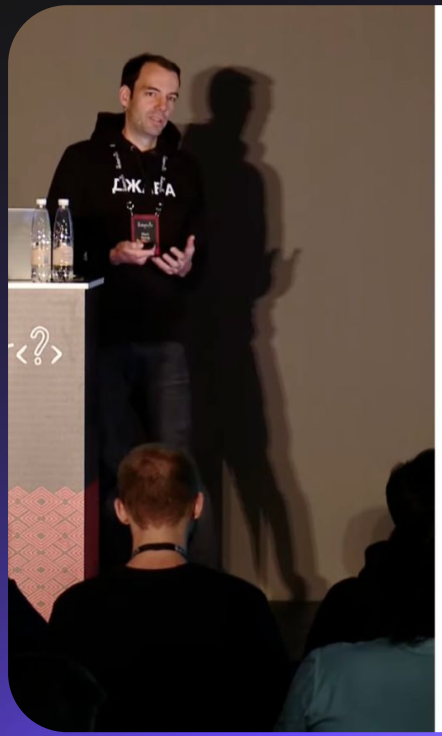
  @Parameter
  String candidate;

  @Test
  void palindrome() { }

  @Test
  void reversePalindrome() { }
}
```

Parameterized Classes and Tests

# Невозможно переоценить роль Extensions сегодня



## JUNIT JUPITER API

- programming mode for test authors
- extension model for extension authors

JUnit 5 extensions: from conditional test execution to test templates

# JUnit 6

- Представлен в июне 2025;
- Java 17+, Kotlin 2.2+;
- API не претерпело изменений;
- Единые версии всех артефактов;
- Как и ранее с 5, Spring быстро объявил о переходе на JUnit 6

# JUnit 6

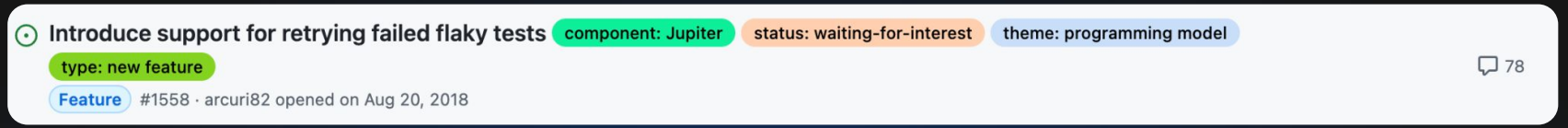
» Поскольку изменение интерфейсов ломает клиентский код, после публикации следует считать их **неизменяемыми**.

» Since changing interfaces breaks clients you should consider them as immutable once you've published them.

Design Principles from Design Patterns A Conversation with Erich Gamma

# JUnit - Будущее

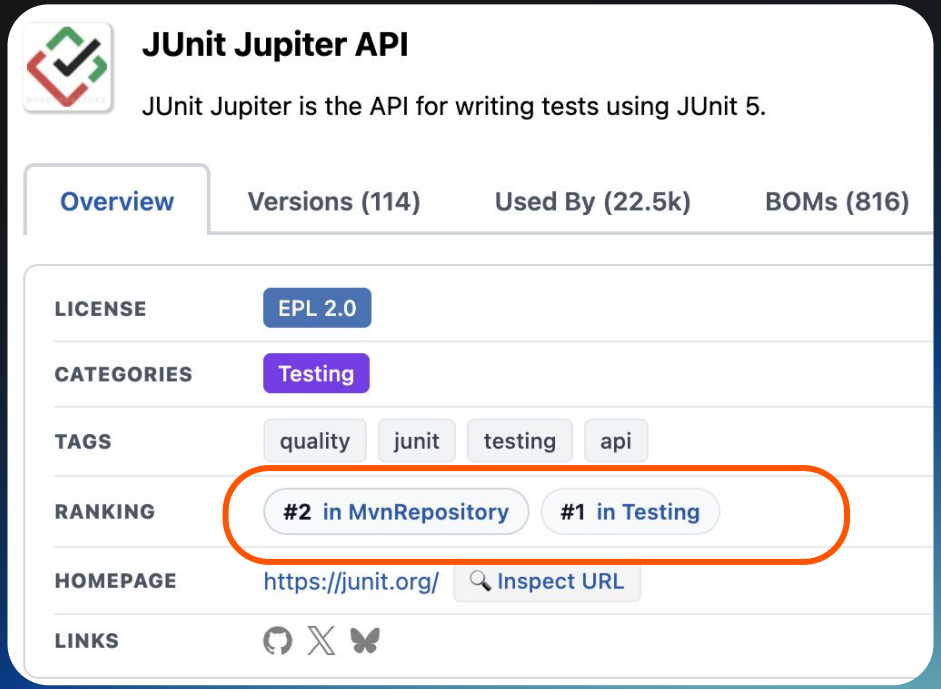
Новый ThreadPool для параллельного выполнения тестов;




Новый отчет станет стандартом (не скоро);

JUnit MCP

# JUnit - Будущее




JUnit

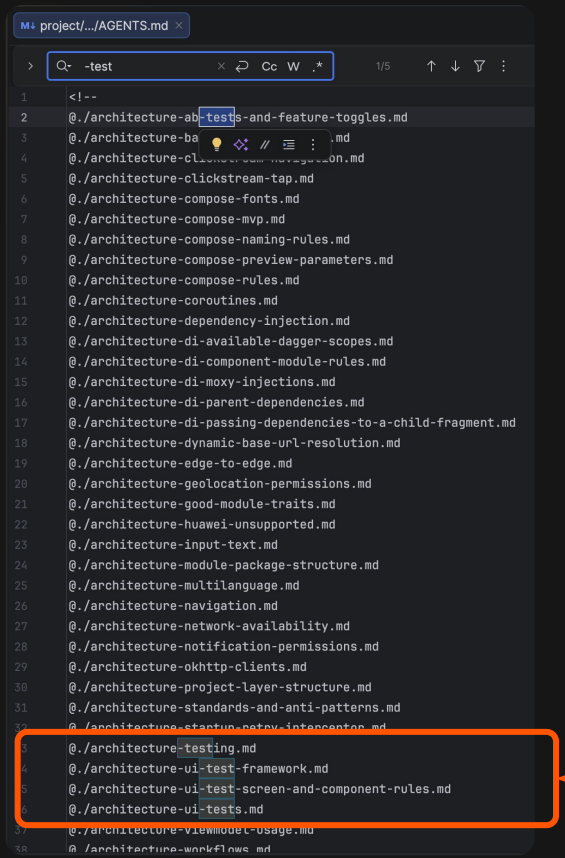
#7 in MvnRepository #2 in Testing

TestNG

#44 in MvnRepository #8 in Testing

# **Эпилог:** любая история может иметь два финала

# Тесты сегодня




```
### Доступ к автоматически созданным мокам
`FixtureRule` неявно создает моки с помощью `MockitoRule`, который также применяется к классу те

Для того чтобы получить доступ к этим автоматически созданным мокам (чтобы задать им поведение ч

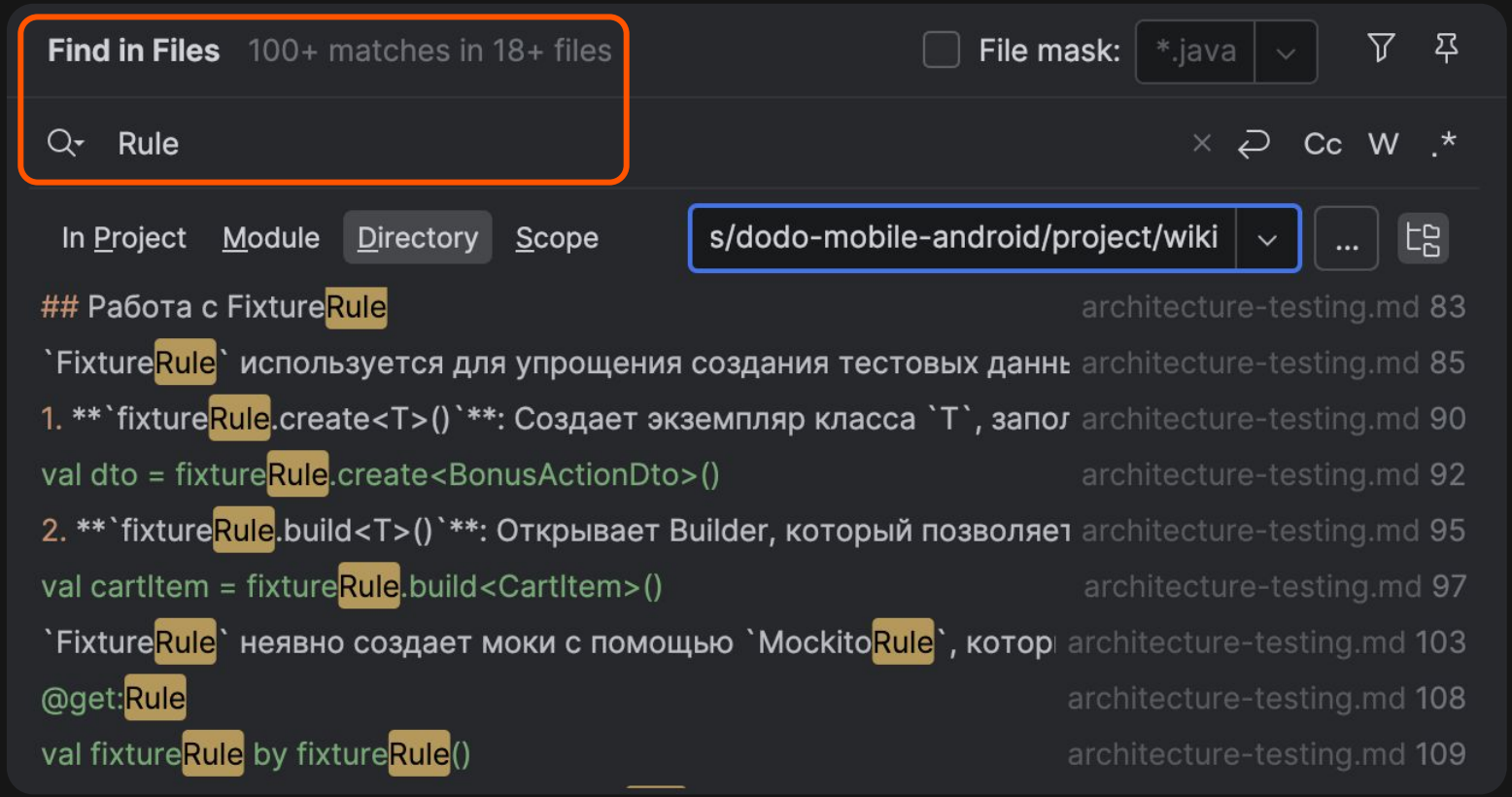
```kotlin
@get:Rule
val fixtureRule by fixtureRule()

// Автоматически созданный мок, который будет использован в конструкторе SUT
@Mock
private lateinit var api: SomeApi

// При создании sut фикстура подставит мок api, объявленный выше
private val sut: SomeInteractor by fixtureRule.creator()

@Test
fun test() {
    whenever(api.getData()).thenReturn(...)
    // ...
}
```

### Переопределение инъекций (Scoping)
Иногда бывает необходимо подменить конкретную зависимость на реальный объект (например, `Corouti

Вы можете вызывать `inject` в двух местах:
1. **При создании правила:** Применяется ко всем тестам в классе.
   ```kotlin
   @get:Rule
   val fixtureRule by fixtureRule {
       inject<CoroutineDispatcher>(testCoroutineRule.dispatcher)
   }
   ```
2. **Внутри тестового метода:** Применяется только в рамках конкретного теста. Важно отметить, ч

   ```kotlin
   @Test
```
```

# Тесты сегодня



» Написание тестов для меня связано с ответственностью. Это способ сказать: «Вот о чем я думал». Каждый тест - это запись о том, о чем я подумал, пока программировал.

» Testing, to me, is about saying, “Here’s what I was thinking.” Each test is a record of something I thought about while I was programming

Kent Beck on creating JUnit

# Эпилог

История про то, о чем **думал автор теста**

```
+
+ @SetFeatures(
+     enabled = [ALWAYS_SHOW_PROMOCODE_IN_CART, ALWAYS_SHOW_PROMOCODE_IN_CART_ENABLED],
+     disabled = [PROMO_FOR_ROOKIE_WITHOUT_EXPERIMENT, SHARE_CART, UPSELL_REDESIGN],
+ )
+ @SetFirebaseSegments(
+     segments = [
+       TestFirebaseSegment(
+           segmentKey = ALWAYS_SHOW_PROMOCODE_IN_CART_SEGMENT,
+           segment = VARIANT_2,
+       )
+     ]
+ )
+ class AlwaysShowPromocodeInCartTest : BaseTest() {
+
+   @Test
+   @AllureId("887887")
+   @WithUser(needTestActions = true, cart = [PIZZA_HAM_AND_CHEESE_MEDIUM])
+   @DisplayName("Поле промокода отображается в корзине при примененной акции")
+   fun actionAndPromocode() = test(startOn = Screen.Profile) {
+     ProfileScreen {
+       applyOffer(offer = PIZZA_ARRIVA_AND_COLA_PROMOTION)
+       clickOnExitButton()
+     }
+
+     FoodMenuScreen { clickOnCartButton() }
+     ShoppingCartScreen {
+       verifyOfferHasBeenApplied(expected = PIZZA_ARRIVA_AND_COLA_PROMOTION)
+       verifyPromocodeButtonIsDisplayed()
+       clickOnPromocodeButton()
+       promocode { verifyIsLoaded() }
```

# Эпилог

История про то, о чем думал **создатель Extensions**

История про то, о чем **думала LLM**

```
+ @SetFeatures(
+     enabled = [ALWAYS_SHOW_PROMOCODE_IN_CART, ALWAYS_SHOW_PROMOCODE_IN_CART_ENABLED],
+     disabled = [PROMO_FOR_ROOKIE_WITHOUT_EXPERIMENT, SHARE_CART, UPSELL_REDESIGN],
+ )
+ @SetFirebaseSegments(
+     segments = [
+       TestFirebaseSegment(
+           segmentKey = ALWAYS_SHOW_PROMOCODE_IN_CART_SEGMENT,
+           segment = VARIANT_2,
+       )
+     ]
+ )
+ class AlwaysShowPromocodeInCartTest : BaseTest() {
+
+   @Test
+   @AllureId("887887")
+   @WithUser(needTestActions = true, cart = [PIZZA_HAM_AND_CHEESE_MEDIUM])
+   @DisplayName("Поле промокода отображается в корзине при примененной акции")
+   fun actionAndPromocode() = test(startOn = Screen.Profile) {
+     ProfileScreen {
+       applyOffer(offer = PIZZA_ARRIVA_AND_COLA_PROMOTION)
+       clickOnExitButton()
+     }
+
+     FoodMenuScreen { clickOnCartButton() }
+     ShoppingCartScreen {
+       verifyOfferHasBeenApplied(expected = PIZZA_ARRIVA_AND_COLA_PROMOTION)
+       verifyPromocodeButtonIsDisplayed()
+       clickOnPromocodeButton()
+       promocode { verifyIsLoaded() }
```

# » **Какую историю** рассказывает ваш автотест?

Спасибо