Из чего же только сделаны тесты?: **генерация тестов для Spring**

Денис Фокин

Huawei

Егор Куликов

Huawei

Joker

Денис Фокин

levantarse

Bio

- Huawei эксперт
- IntelliJ платформа
- Sun Microsystems, Oracle
- OpenJDK contributor

Егор Куликов

egor_k_kulikov

Bio

- К. ф.–м. н.
- Анализ кода

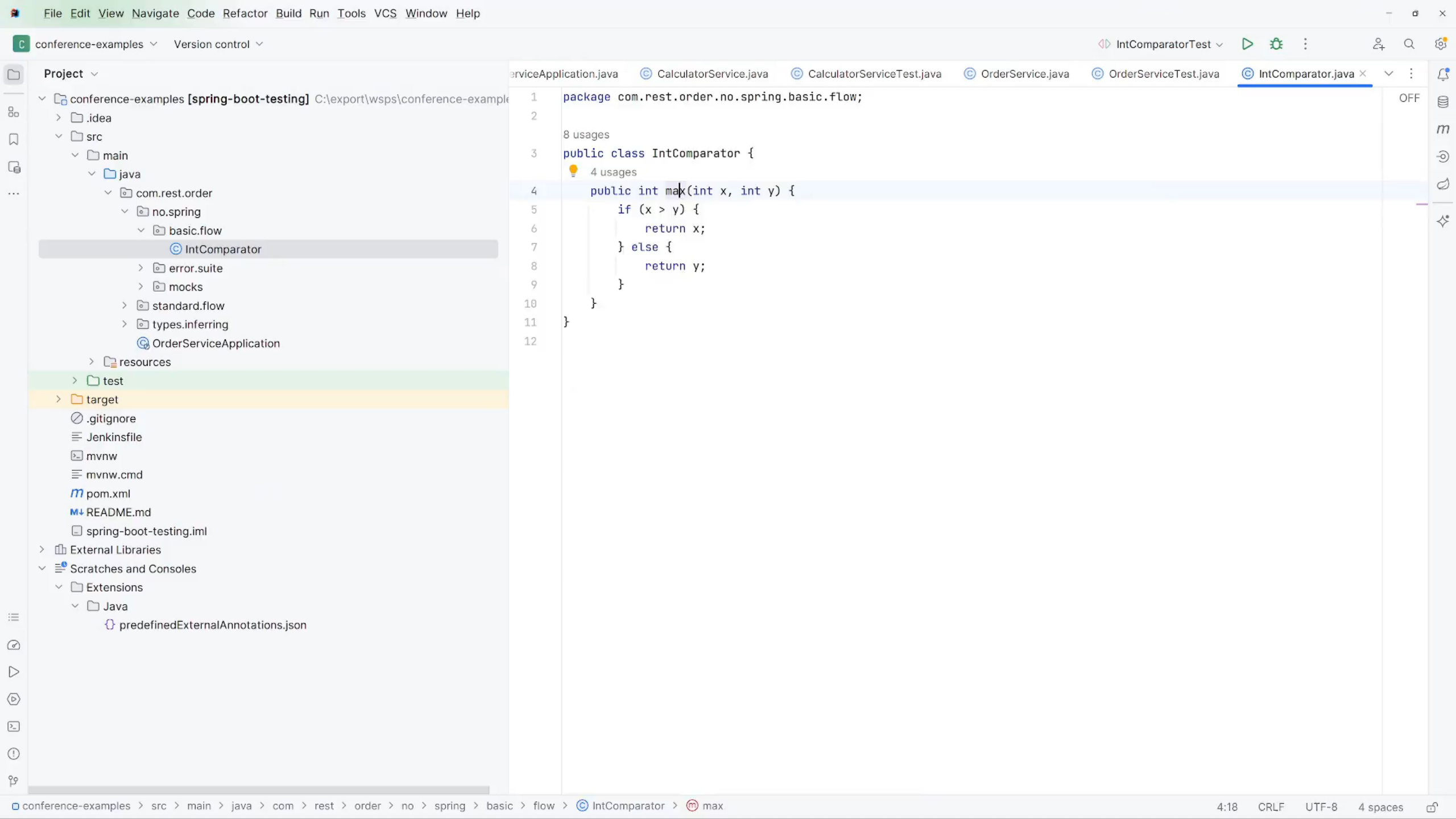
- Методы обработки сигналов
- Цифровая криминалистика
- Разработка ERP-систем

План доклада

- Что мы ожидаем от тестов?
  - Включая автоматически сгенерированных
- Генерация тестов
  - Что под капотом?
- Что особенного в Spring тестах?
- Как Spring помогает в генерации?
  - Вспомогательная информация
- Анализ кода
  - Безопасно?

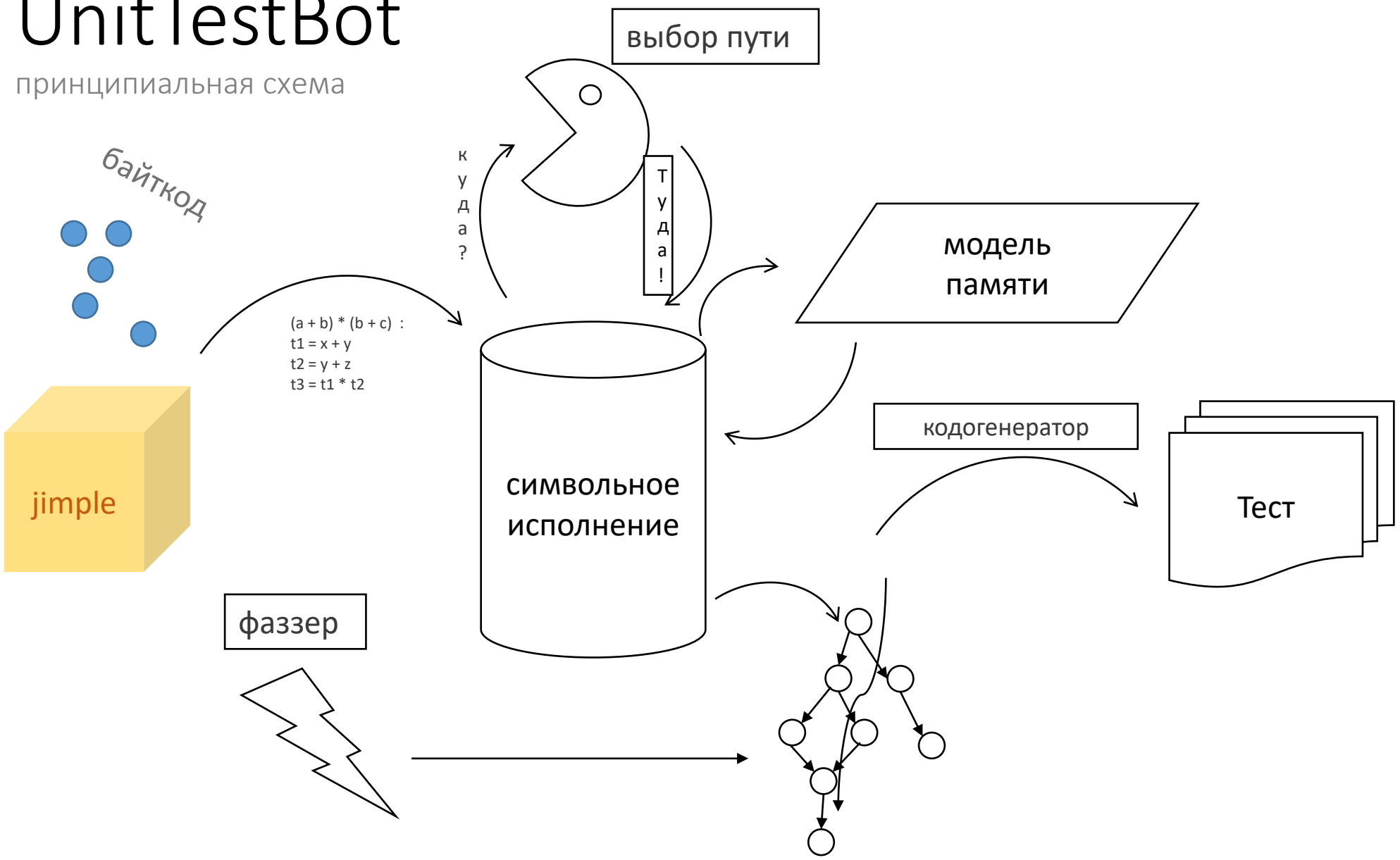
Что мы ждём от тестов?

```
public class IntComparator {
    public int max(int a, int b) {
        if (a > b) {
            return a;
        } else {
            return b;
        }
    }
}
```


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
conference-examples
Version control
IntComparatorTest
Project
conference-examples [spring-boot-testing] C:\export\wsps\conference-example
.idea
src
main
java
com.rest.order
no.spring
basic.flow
IntComparator
error.suite
mocks
standard.flow
types.inferring
OrderServiceApplication
resources
test
target
.gitignore
Jenkinsfile
mvnw
mvnw.cmd
pom.xml
README.md
spring-boot-testing.iml
External Libraries
Scratches and Consoles
Extensions
Java
predefinedExternalAnnotations.json
erviceApplication.java
CalculatorService.java
CalculatorServiceTest.java
OrderService.java
OrderServiceTest.java
IntComparator.java
OFF
1 package com.rest.order.no.spring.basic.flow;
2
8 usages
3 public class IntComparator {
4 usages
4 public int max(int x, int y) {
5 if (x > y) {
6 return x;
7 } else {
8 return y;
9 }
10 }
11 }
12
conference-examples > src > main > java > com > rest > order > no > spring > basic > flow > IntComparator > max
4:18
CRLF
UTF-8
4 spaces

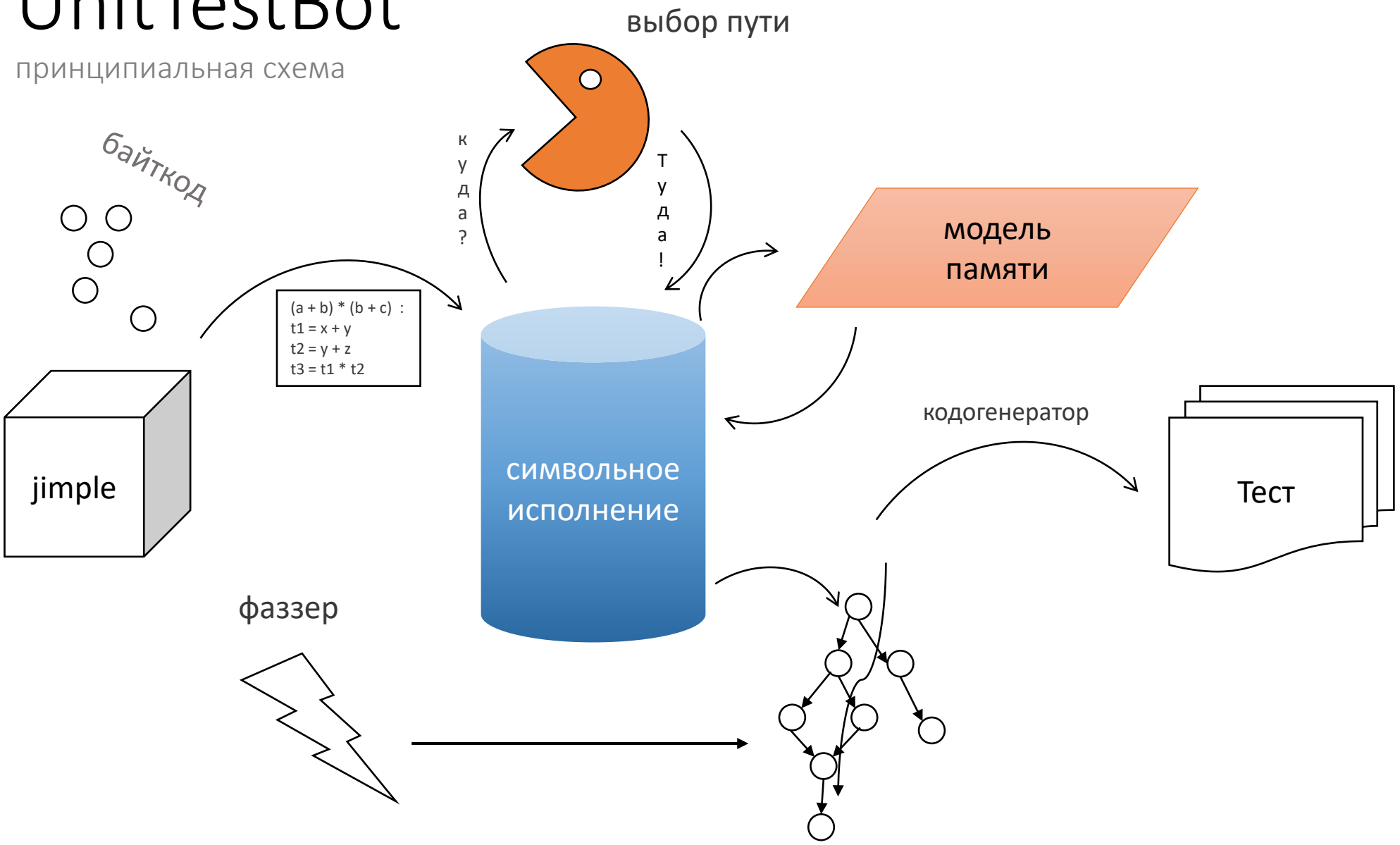
UnitTestBot

принципиальная схема



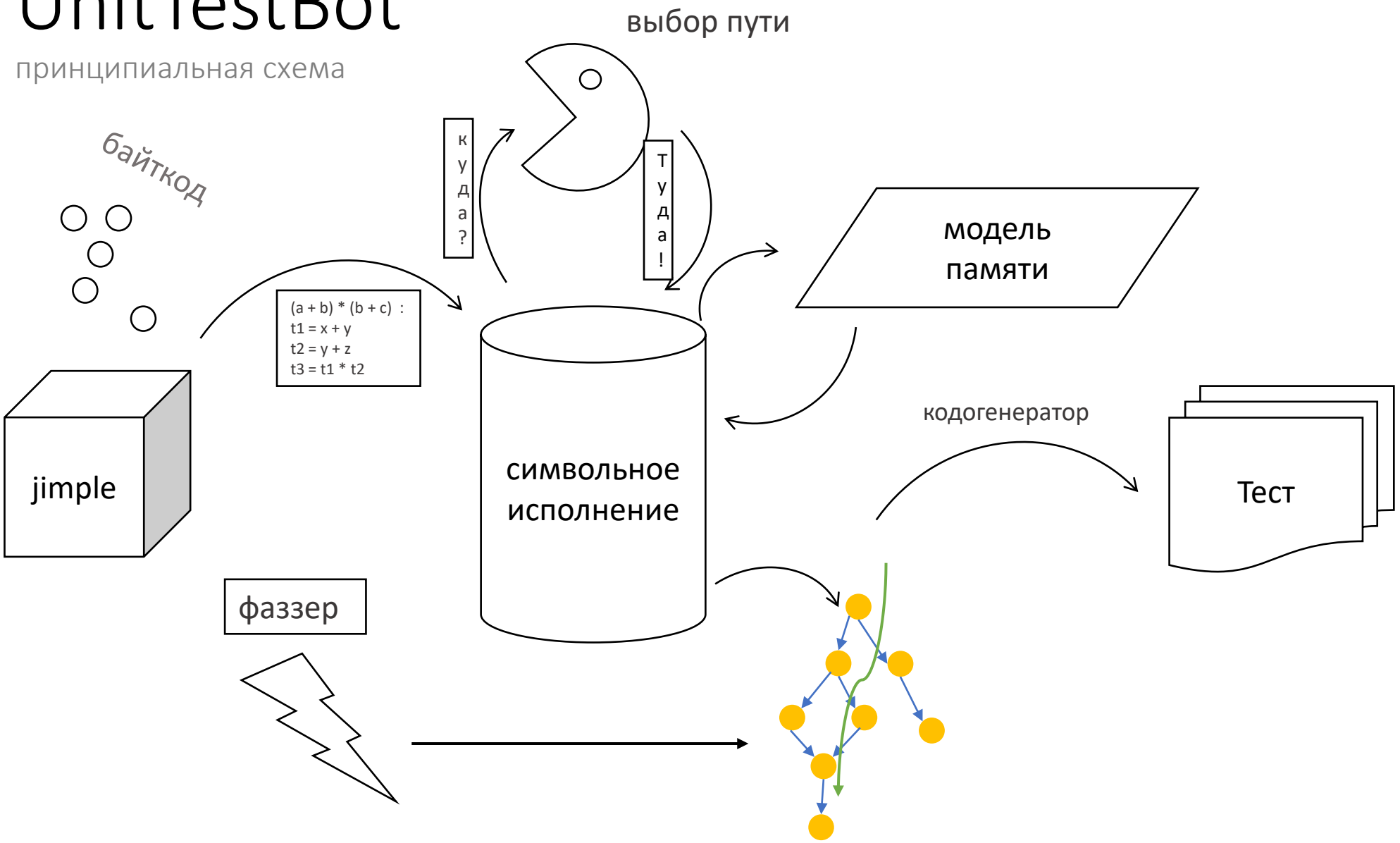
UnitTestBot

принципиальная схема



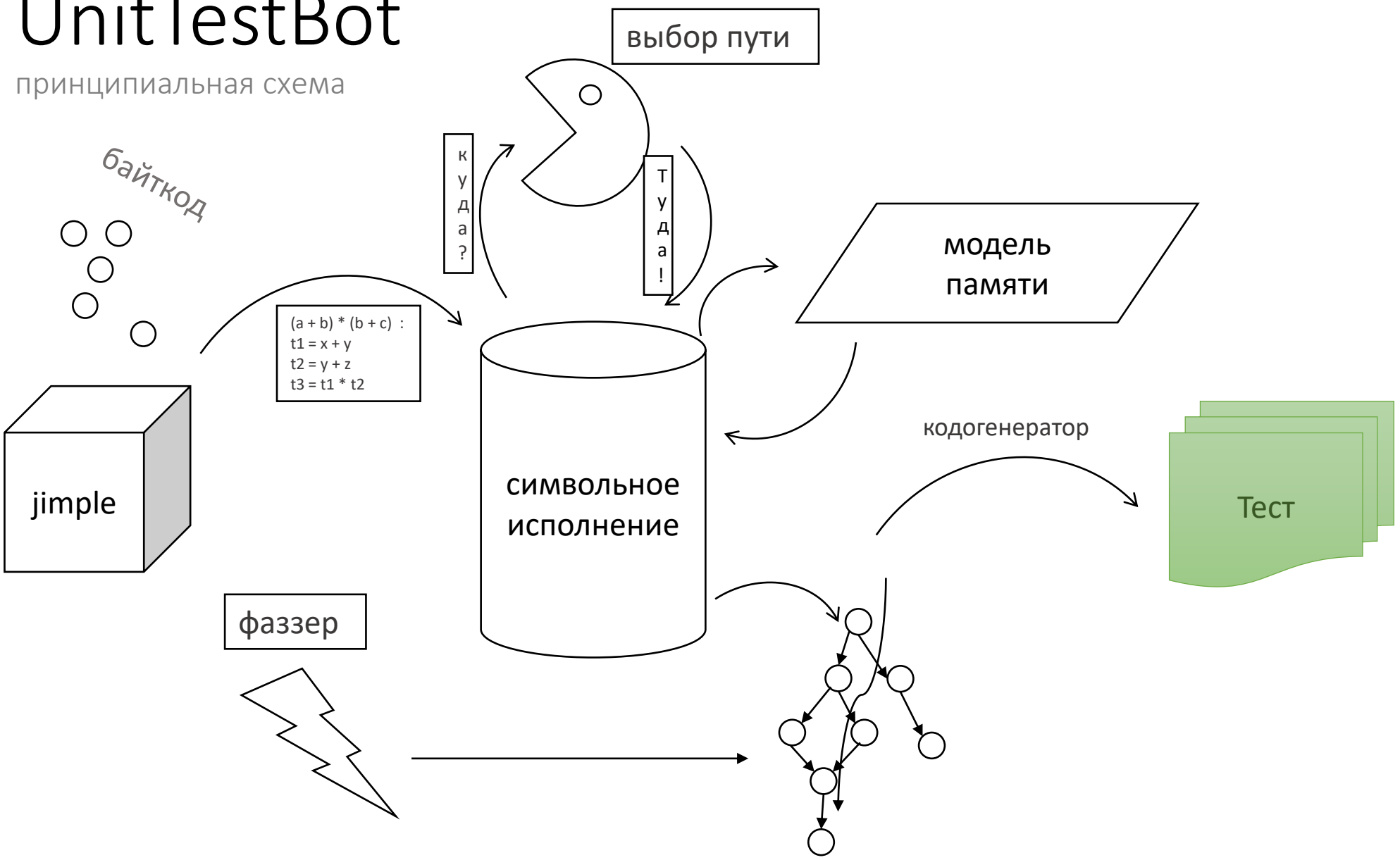
UnitTestBot

принципиальная схема



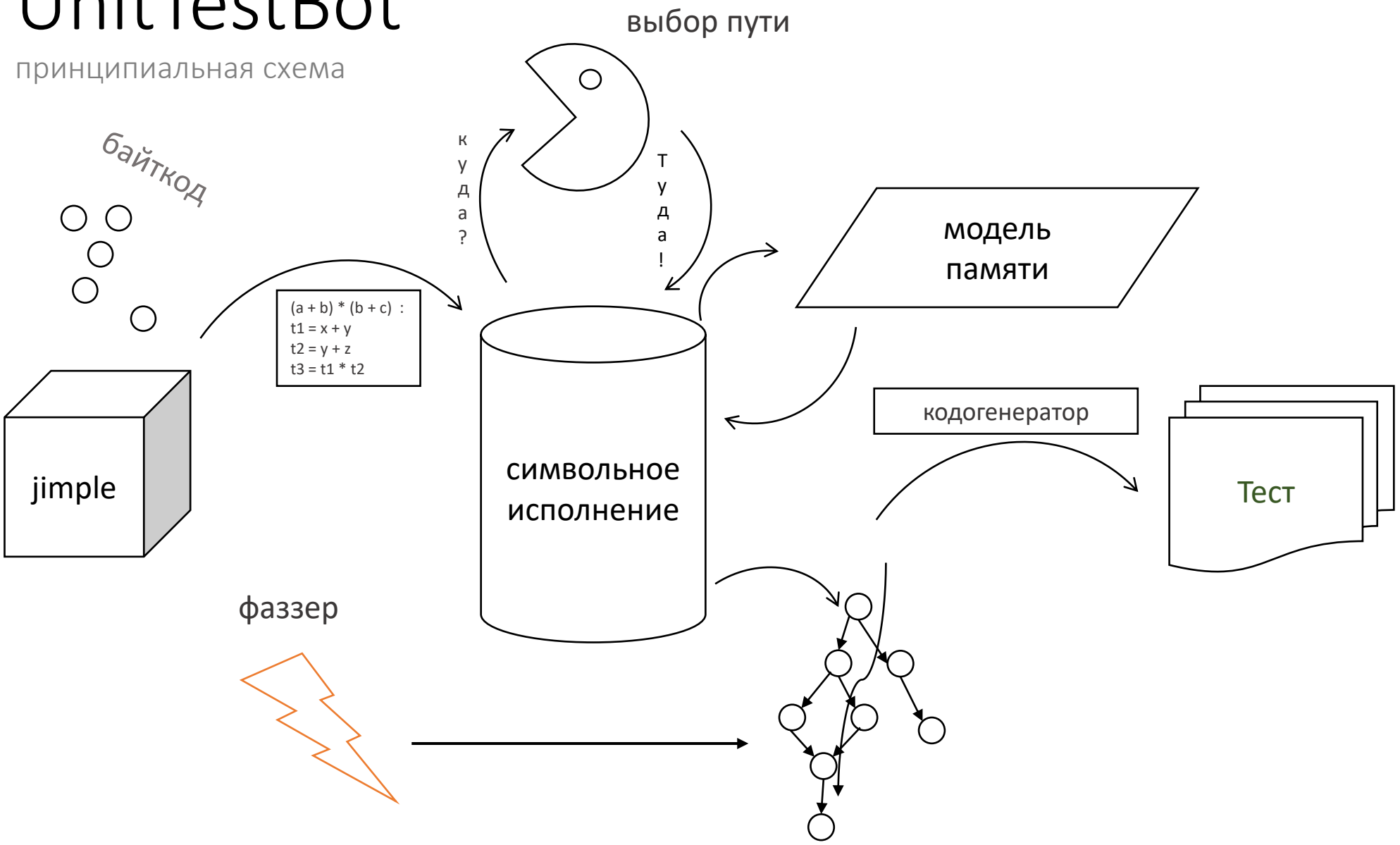
UnitTestBot

принципиальная схема



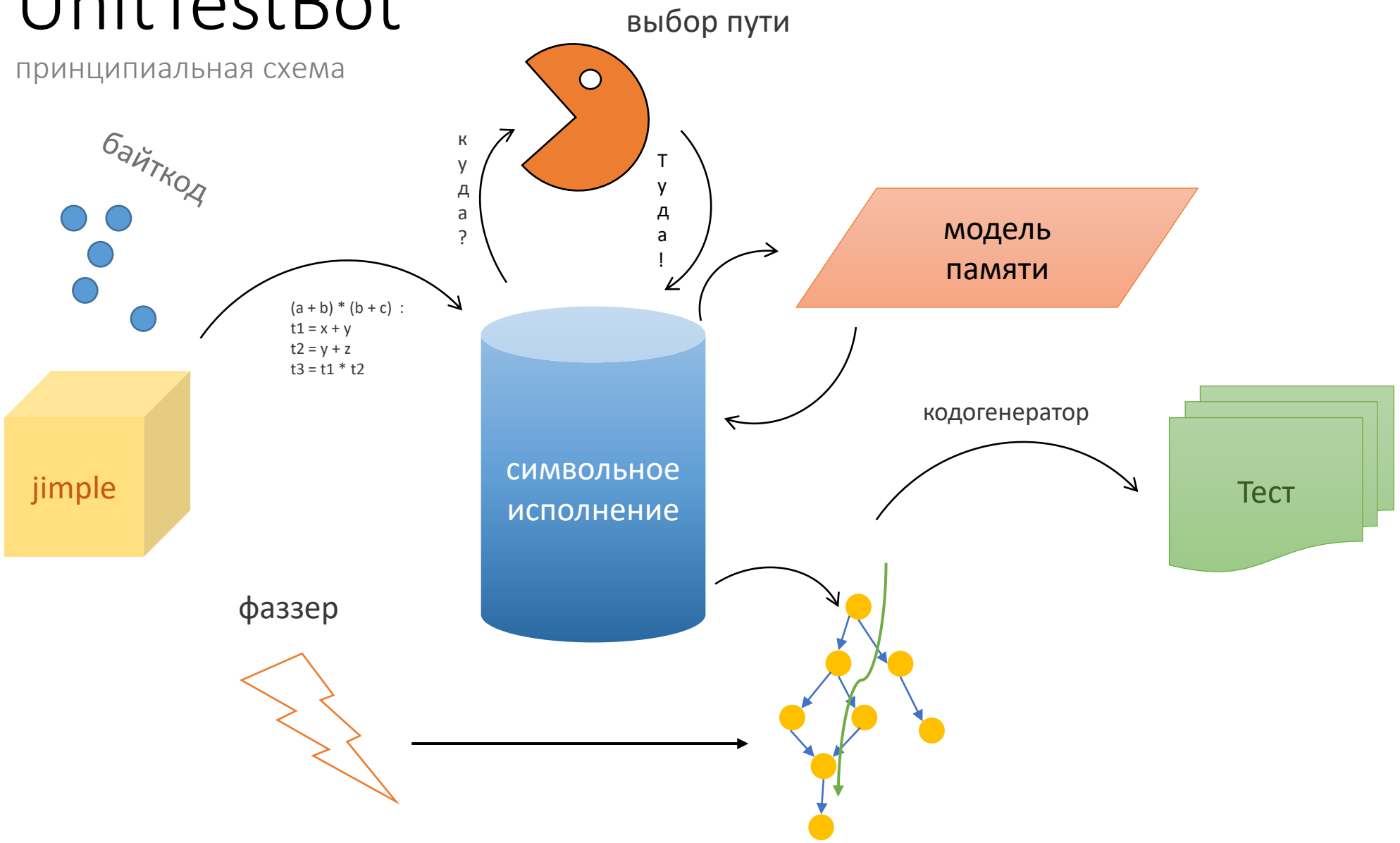
UnitTestBot

принципиальная схема

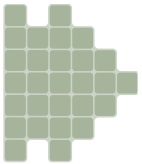


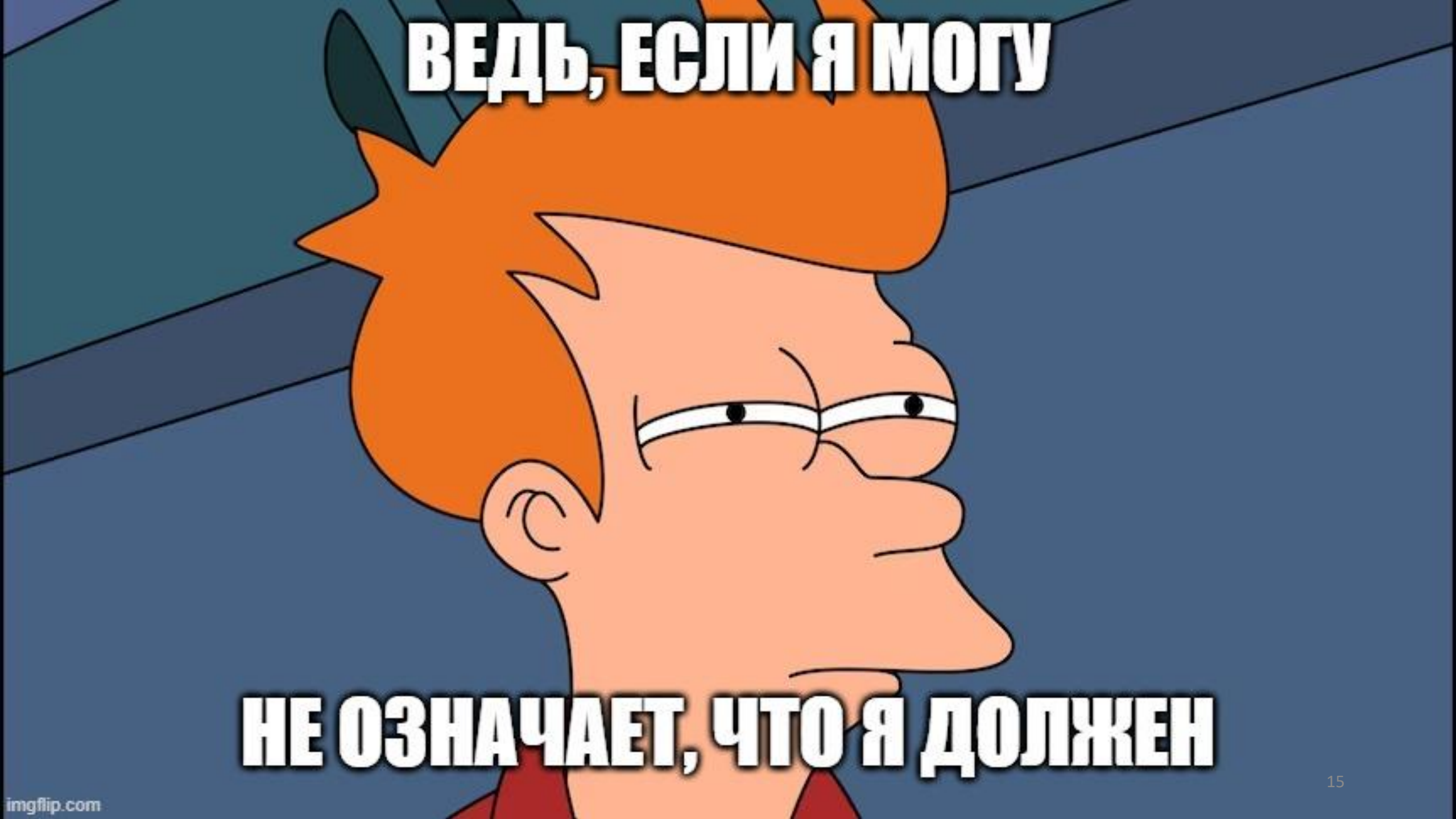
UnitTestBot

принципиальная схема

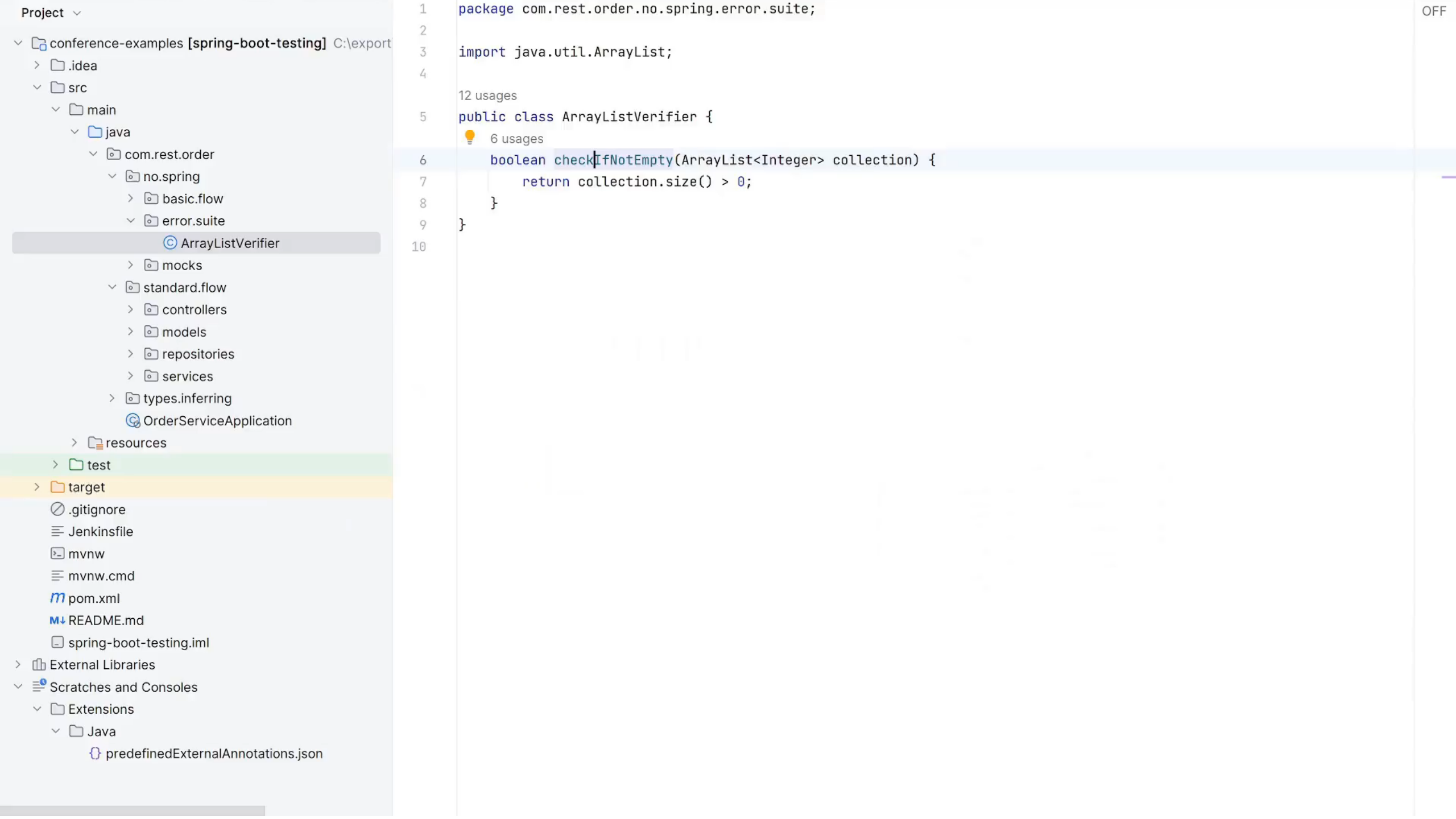


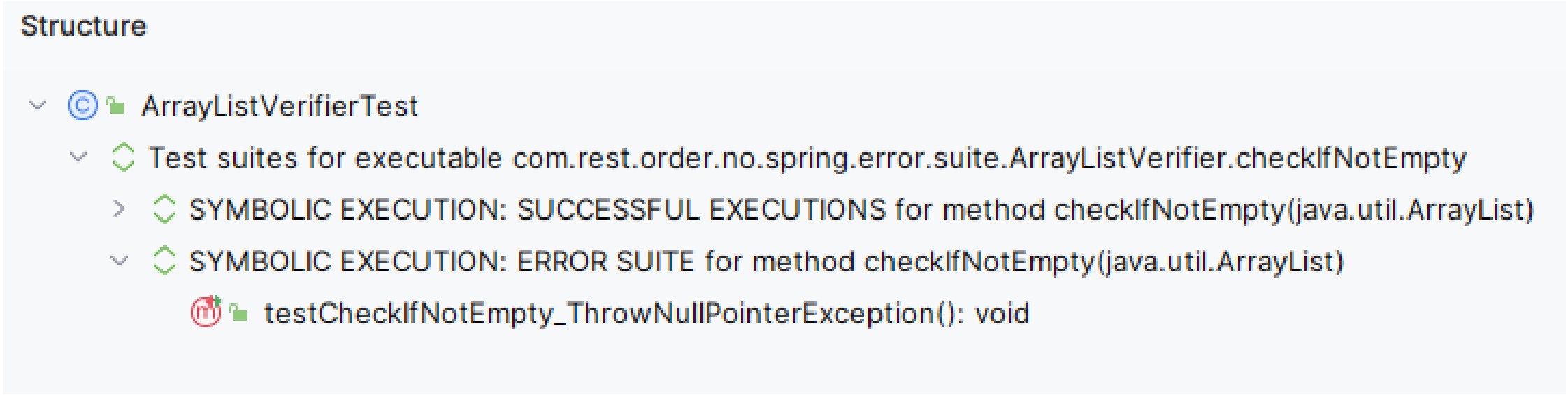
```java
public final class IntComparatorTest {
    ///region Test suites for executable com.rest.order.no.spring.basic.flow.IntComparator.max

    ///region SYMBOLIC EXECUTION: SUCCESSFUL EXECUTIONS for method max(int, int)

    /**
     * @utbot.classUnderTest {@link IntComparator}
     * @utbot.methodUnderTest {@link IntComparator#max(int, int)}
     * @utbot.executesCondition {@code (x > y): True}
     * @utbot.returnsFrom {@code return x;}
     */
    @Test
    @DisplayName("max: x > y : True -> return x")
    public void testMax_XGreaterThanY() {
        IntComparator intComparator = new IntComparator();

        int actual = intComparator.max(-1, -2);

        assertEquals(-1, actual);
    }
```




ВЕДЬ, ЕСЛИ Я МОГУ
НЕ ОЗНАЧАЕТ, ЧТО Я ДОЛЖЕН
imgflip.com

```
public class ArrayListVerifier {

    boolean checkIfNotEmpty(ArrayList<Integer> collection) {
        return collection.size() > 0;
    }

}
```


Project
conference-examples [spring-boot-testing] C:\export
.idea
src
main
java
com.rest.order
no.spring
basic.flow
error.suite
ArrayListVerifier
mocks
standard.flow
controllers
models
repositories
services
types.inferring
OrderServiceApplication
resources
test
target
.gitignore
Jenkinsfile
mvnw
mvnw.cmd
pom.xml
README.md
spring-boot-testing.iml
External Libraries
Scratches and Consoles
Extensions
Java
predefinedExternalAnnotations.json
1 package com.rest.order.no.spring.error.suite;
2
3 import java.util.ArrayList;
4
12 usages
5 public class ArrayListVerifier {
6 boolean checkIfNotEmpty(ArrayList<Integer> collection) {
7 return collection.size() > 0;
8 }
9 }
10
OFF


Structure
ArrayListVerifierTest
Test suites for executable com.rest.order.no.spring.error.suite.ArrayListVerifier.checkIfNotEmpty
SYMBOLIC EXECUTION: SUCCESSFUL EXECUTIONS for method checkIfNotEmpty(java.util.ArrayList)
SYMBOLIC EXECUTION: ERROR SUITE for method checkIfNotEmpty(java.util.ArrayList)
testCheckIfNotEmpty_ThrowNullPointerException(): void

UnitTestBot

TABLE III Result for Test Case Understandability Study

Tool	LMap	Months	DUtils	Average
EvoSuite	2.38	2.08	2.23	2.23
UTBot-concolic	1.92	2.23	2.23	2.13
UTBot-fuzzer	3.54	2.77	2.69	3.00
Kex-symbolic	3.62	4.00	4.23	3.95
Kex-concolic	3.54	3.92	3.62	3.69

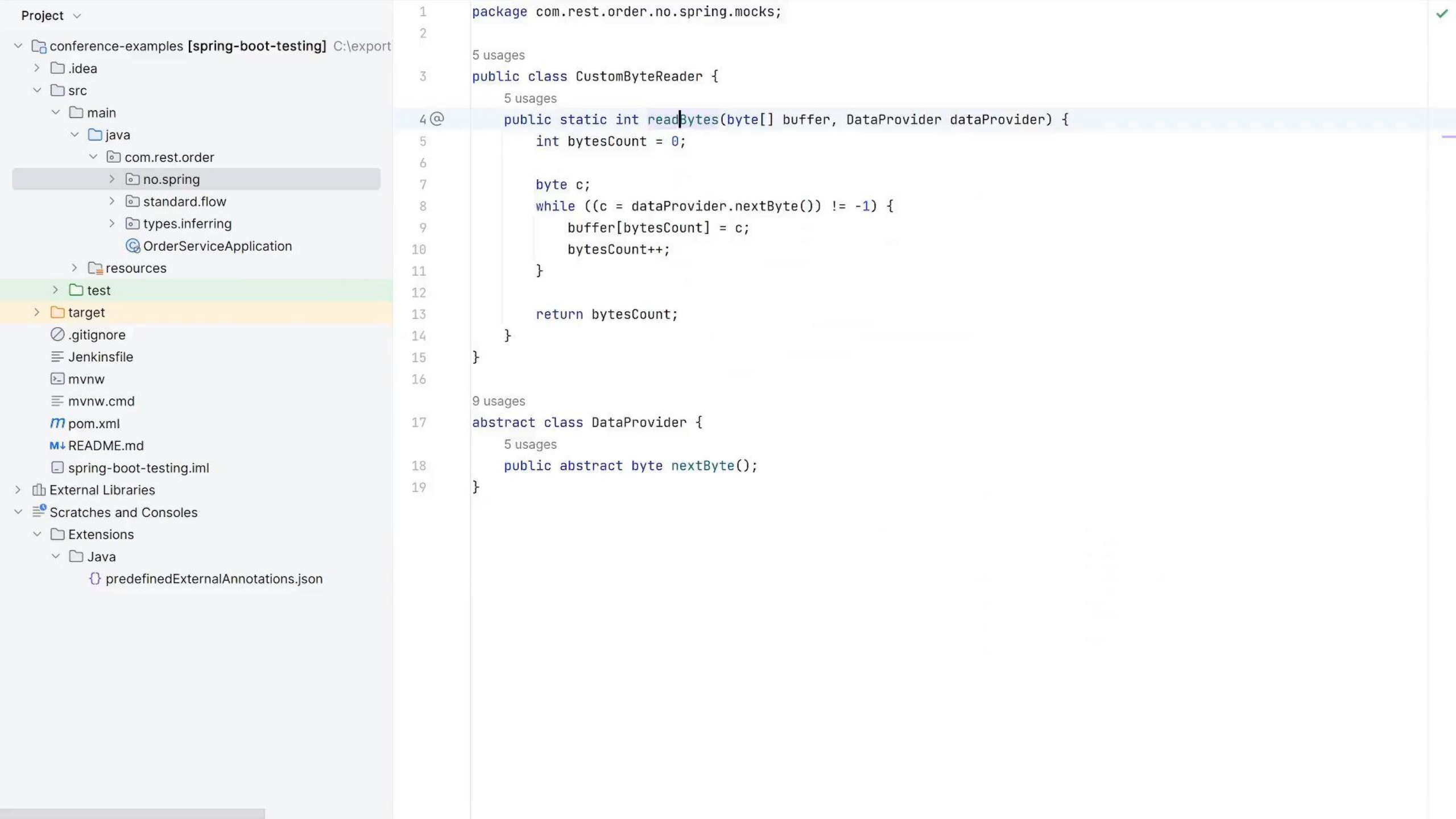
Table III reports the results of the study. Columns *LMap, Months, DUtils* report the average rankings the human participants assigned for each tool to the test cases for LinkedMap, Months and DistanceUtils classes accordingly. Column *Average* reports the average across the test cases for the three classes. As the results show, the tool with the highest understandability of the selected test cases is UTBOT-CONCOLIC, followed by EVOSUITE.

Scores and Rankings. The formula for the score [8] has been created and improved during the previous editions of the tool competition and takes into account the line and branch coverage, the mutation score, and the time budget used by the generator. Moreover, it applies a penalty for flaky and non-compiling tests. We observed a final score of 678.12 for EVOSUITE, 530.71 for UTBOT-CONCOLIC, 426.19 for RANDOOP, 381.48 for UTBOT-FUZZER, 195.09 for KEX-CONCOLIC and 128.80 for KEX-SYMBOLIC. The rankings of the tools are reported in the column *CoverageR* of Table IV.

Суровая реальность

- Внешние зависимости
- Сервисы
- Базы данных
- Файловая система

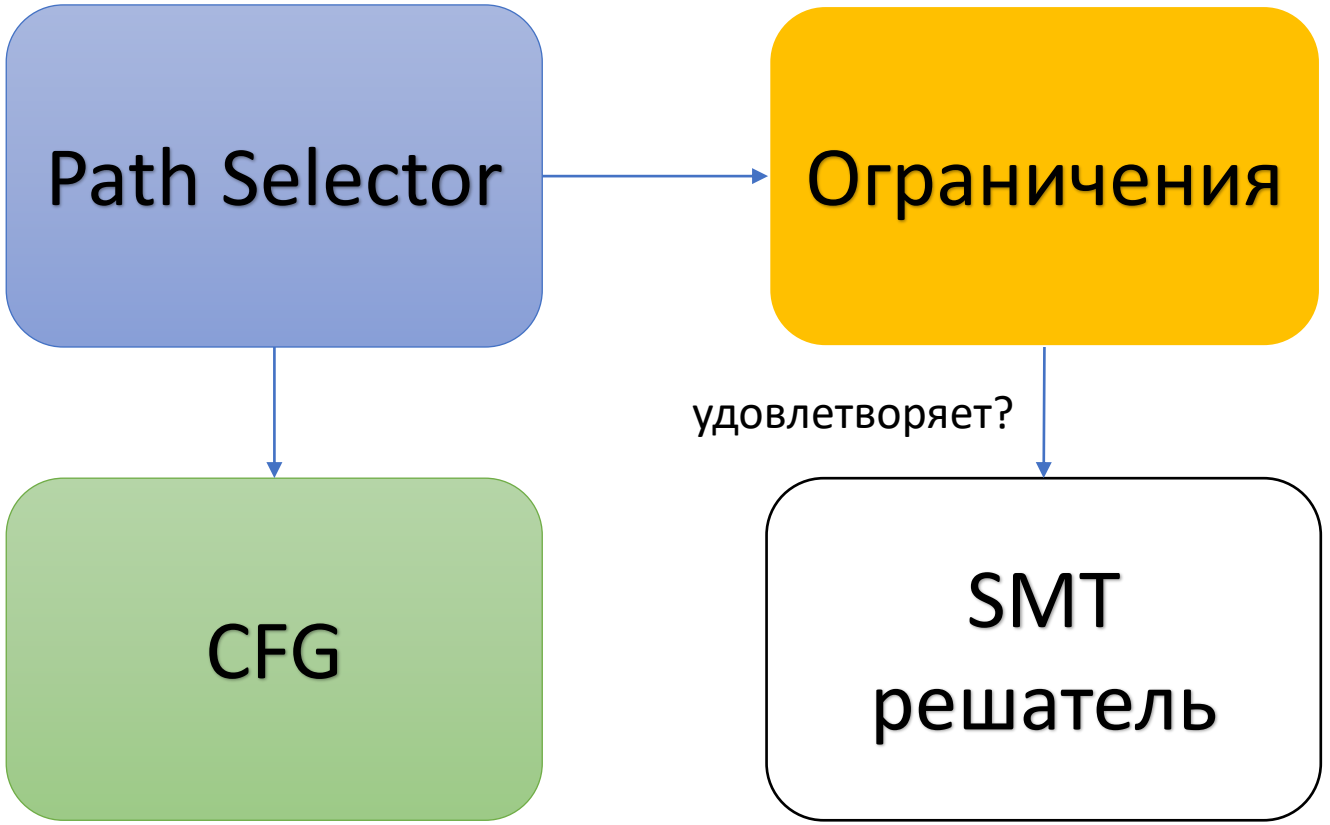
```
public class CustomByteReader {
    public static int readBytes(byte[] buffer, DataProvider dataProvider) {
        int bytesCount = 0;

        byte c;
        while ((c = dataProvider.nextByte()) != -1) {
            buffer[bytesCount] = c;
            bytesCount++;
        }

        return bytesCount;
    }
}

abstract class DataProvider {
    public abstract byte nextByte();
}
```


Project
conference-examples [spring-boot-testing] C:\export
.idea
src
main
java
com.rest.order
no.spring
standard.flow
types.inferring
OrderServiceApplication
resources
test
target
.gitignore
Jenkinsfile
mvnw
mvnw.cmd
pom.xml
README.md
spring-boot-testing.iml
External Libraries
Scratches and Consoles
Extensions
Java
predefinedExternalAnnotations.json
package com.rest.order.no.spring.mocks;
5 usages
public class CustomByteReader {
5 usages
public static int readBytes(byte[] buffer, DataProvider dataProvider) {
int bytesCount = 0;
byte c;
while ((c = dataProvider.nextByte()) != -1) {
buffer[bytesCount] = c;
bytesCount++;
}
return bytesCount;
}
}
9 usages
abstract class DataProvider {
5 usages
public abstract byte nextByte();
}

```java
@Test
public void testReadBytes1() {
    byte[] buffer = {Byte.MIN_VALUE};
    DataProvider dataProviderMock = mock(DataProvider.class);
    (when(dataProviderMock.nextByte())).thenReturn((byte) 0, (byte) -1);

    int actual = CustomByteReader.readBytes(buffer, dataProviderMock);

    assertEquals( expected: 1, actual);

    byte finalBuffer0 = buffer[0];

    assertEquals((byte) 0, finalBuffer0);
}
```

Символьное исполнение

принципиальная схема



$$\neg\alpha \wedge (\beta < 5) \wedge \neg\gamma$$

A Survey of Symbolic Execution Techniques

ROBERTO BALDONI, EMILIO COPPA, DANIELE CONO D'ELIA, CAMIL DEMETRESCU, and IRENE FINOCCHI, Sapienza University of Rome

Many security and software testing applications require checking whether certain properties of a program hold for any possible usage scenario. For instance, a tool for identifying software vulnerabilities may need to rule out the existence of any backdoor to bypass a program's authentication. One approach would be to test the program using different, possibly random inputs. As the backdoor may only be hit for very specific program workloads, automated exploration of the space of possible inputs is of the essence. Symbolic execution provides an elegant solution to the problem, by systematically exploring many possible execution paths at the same time without necessarily requiring concrete inputs. Rather than taking on fully specified input values, the technique abstractly represents them as symbols, resorting to constraint solvers to construct actual instances that would cause property violations. Symbolic execution has been incubated in dozens of tools developed over the last four decades, leading to major practical breakthroughs in a number of prominent software reliability applications. The goal of this survey is to provide an overview of the main ideas, challenges, and solutions developed in the area, distilling them for a broad audience.

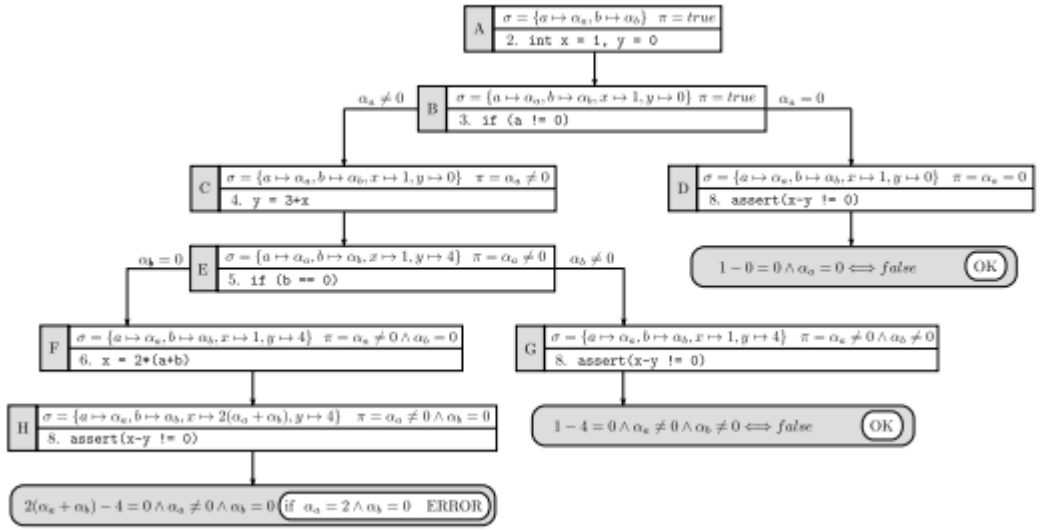
CCS Concepts: • **Software and its engineering** → **Software verification**; *Software testing and debugging*; • **Security and privacy** → Software and application security;

Additional Key Words and Phrases: Symbolic execution, static analysis, concolic execution, software testing

ACM Reference Format:
Roberto Baldoni, Emilio Coppa, Daniele Cono D'Elia, Camil Demetrescu, and Irene Finocchi. 2018. A Survey of Symbolic Execution Techniques. *ACM Comput. Surv.* 51, 3, Article 0 (2018), 37 pages. https://doi.org/0000001.0000001

"Sometimes you can't see how important something is in its moment, even if it seems kind of important. This is probably one of those times."

(Cyber Grand Challenge highlights from DEF CON 24, August 6, 2016)



TALK

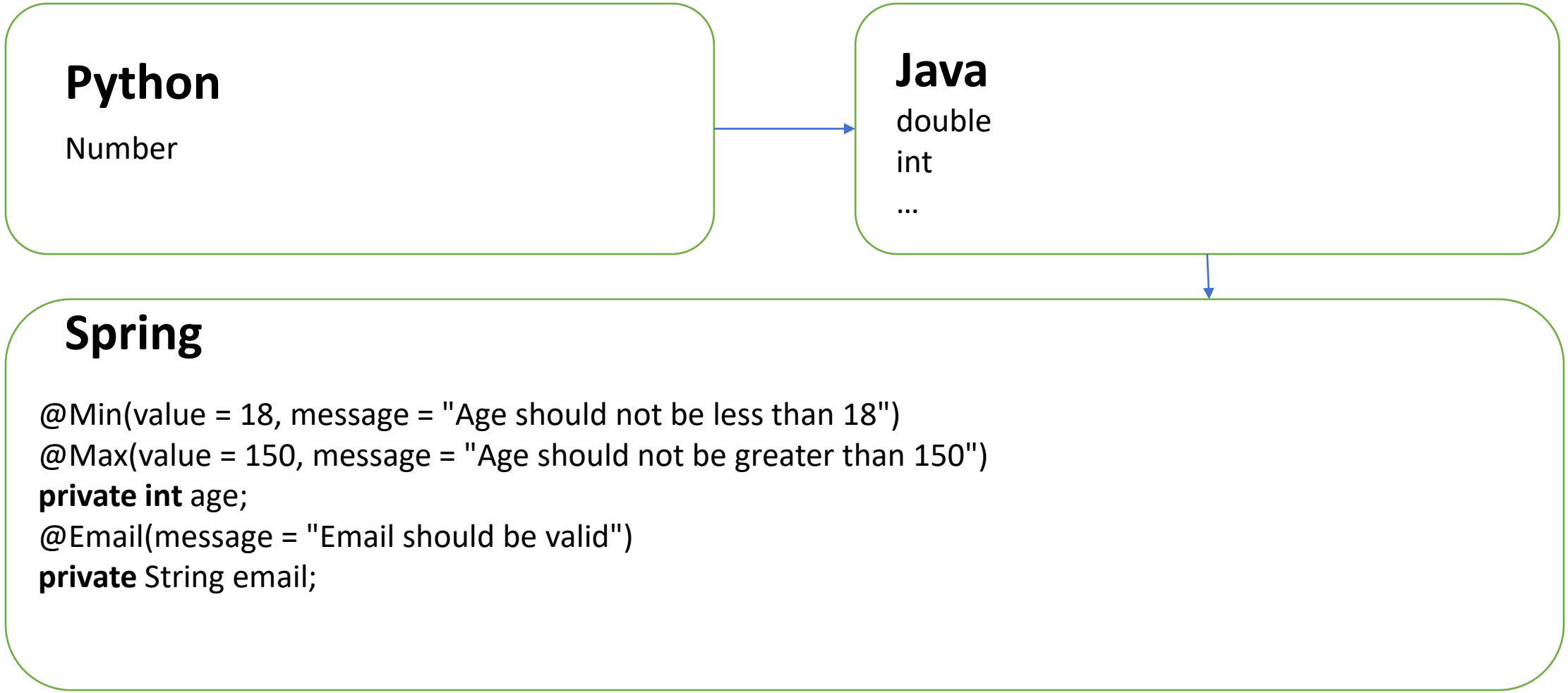
Zero false positive code analysis in Java with dynamic symbolic execution

Dmitry Ivanov
Huawei

Dmitry Mordvinov

https://arxiv.org/pdf/1610.00502.pdf

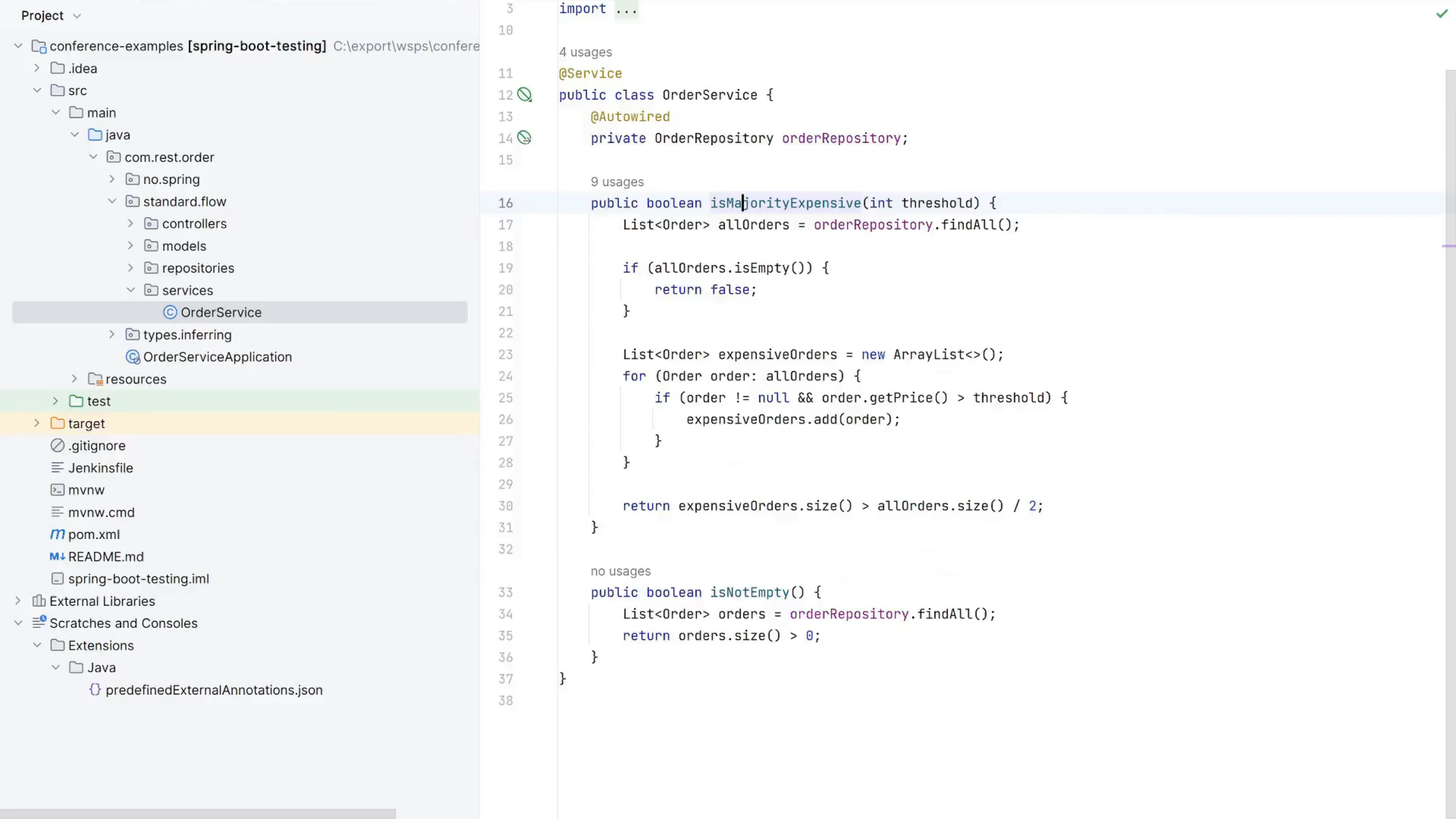
Сужение домена анализа



```java
@Service
public class OrderService {
    @Autowired
    private OrderRepository orderRepository;

    public boolean isMajorityExpensive(int threshold) {
        List<Order> allOrders = orderRepository.findAll();

        if (allOrders.isEmpty()) {
            return false;
        }

        List<Order> expensiveOrders = new ArrayList<>();
        for (Order order: allOrders) {
            if (order != null && order.getPrice() > threshold) {
                expensiveOrders.add(order);
            }
        }
        return expensiveOrders.size() > allOrders.size() / 2;
    }
```

1 — Зависимости

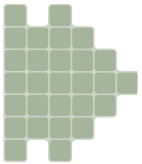
2 — Логика

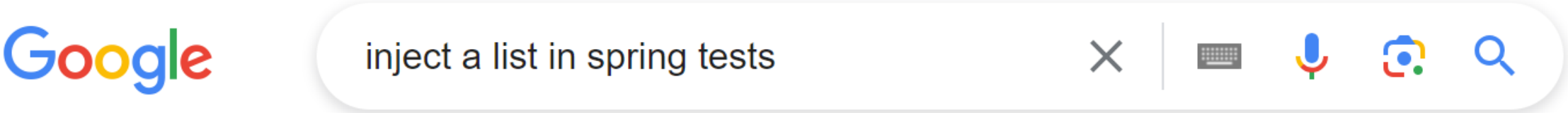

Project
conference-examples [spring-boot-testing] C:\export\wsps\confere
.idea
src
main
java
com.rest.order
no.spring
standard.flow
controllers
models
repositories
services
OrderService
types.inferring
OrderServiceApplication
resources
test
target
.gitignore
Jenkinsfile
mvnw
mvnw.cmd
pom.xml
README.md
spring-boot-testing.iml
External Libraries
Scratches and Consoles
Extensions
Java
predefinedExternalAnnotations.json
import ...
4 usages
@Service
public class OrderService {
@Autowired
private OrderRepository orderRepository;
9 usages
public boolean isMajorityExpensive(int threshold) {
List<Order> allOrders = orderRepository.findAll();
if (allOrders.isEmpty()) {
return false;
}
List<Order> expensiveOrders = new ArrayList<>();
for (Order order: allOrders) {
if (order != null && order.getPrice() > threshold) {
expensiveOrders.add(order);
}
}
return expensiveOrders.size() > allOrders.size() / 2;
}
no usages
public boolean isNotEmpty() {
List<Order> orders = orderRepository.findAll();
return orders.size() > 0;
}
}

```
@Test
public void testIsMajorityExpensive3() {
    LinkedList linkedList = new LinkedList();
    Order orderMock = mock(Order.class);
    (when(orderMock.getPrice())).thenReturn(Integer.MAX_VALUE);
    linkedList.add(orderMock);
    Order orderMock1 = mock(Order.class);
    (when(orderMock1.getPrice())).thenReturn(1);
    linkedList.add(orderMock1);
    Order orderMock2 = mock(Order.class);
    (when(orderMock2.getPrice())).thenReturn(3);
    linkedList.add(orderMock2);
    Order orderMock3 = mock(Order.class);
    (when(orderMock3.getPrice())).thenReturn(-1);
    linkedList.add(orderMock3);
    Order orderMock4 = mock(Order.class);
    (when(orderMock4.getPrice())).thenReturn(0);
    linkedList.add(orderMock4);
    (when(orderRepositoryMock.findAll())).thenReturn(linkedList);

    boolean actual = orderService.isMajorityExpensive(17);

    assertFalse(actual);
}
```

Сложные случаи

- У вас есть два кандидата-компонента на вставку
- Коллекция компонентов





Все Картинки Видео Книги Покупки Ещё Инструменты

Результатов: примерно 78 000 000 (0,38 сек.)

Stack Overflow
https://stackoverflow.com › h... · Перевести эту страницу

How to mock a autowired list of Spring beans? - java

23 окт. 2015 г. — I would make the field final and simply create a constructor. That way you can really make it a unit **test** and construct the object yourself and ...

3 ответа · Лучший ответ: I finally figured it out... Sometimes, asking a question can give you a…

Mockito - **Injecting a List** of mocks - Stack Overflow	20 февр. 2017 г.
How to mock **injected list** of implementations? - Stack Overflow	9 авг. 2021 г.
Mockito: How to properly mock a **List** of **Spring** Services	26 мар. 2021 г.
How to use Dependency **Injection** with **Spring** Boot Unit **Testing**?	11 дек. 2020 г.

Другие результаты с сайта stackoverflow.com

I finally figured it out...

46

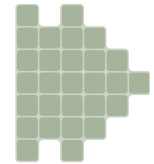
Sometimes, asking a question can give you a better approach to your problems :p

The problem is I was linking the validators to the list before they were mocked. The validators was then null and no reference could be updated when the `MockitAnnotations.initMocks(this)` was called.

Moreover, to avoid iterator problems on `List`, I had to use `@Spy` instead of `@Mock`.

Here is the final solution:

```
@Mock
private EsmRegexValidator regexValidator;

@Mock
private EsmFormNotNullValidator notNullValidator;

@Mock
private EsmFormDataTypeValidator dataValidator;

@InjectMocks
private EsmFormValidatorManager validatorManager;

@Spy
private List<IEsmFormValidator> validators = new ArrayList<IEsmFormValidator>();

@Mock
```



```java
@Service
public class OrderService {
    @Autowired
    private OrderRepository orderRepository;

    public boolean isMajorityExpensive(int threshold) {
       {…}
    }

    public boolean isNotEmpty() {
        List<Order> orders = orderRepository.findAll();
        return orders.size() > 0;
    }
}
```

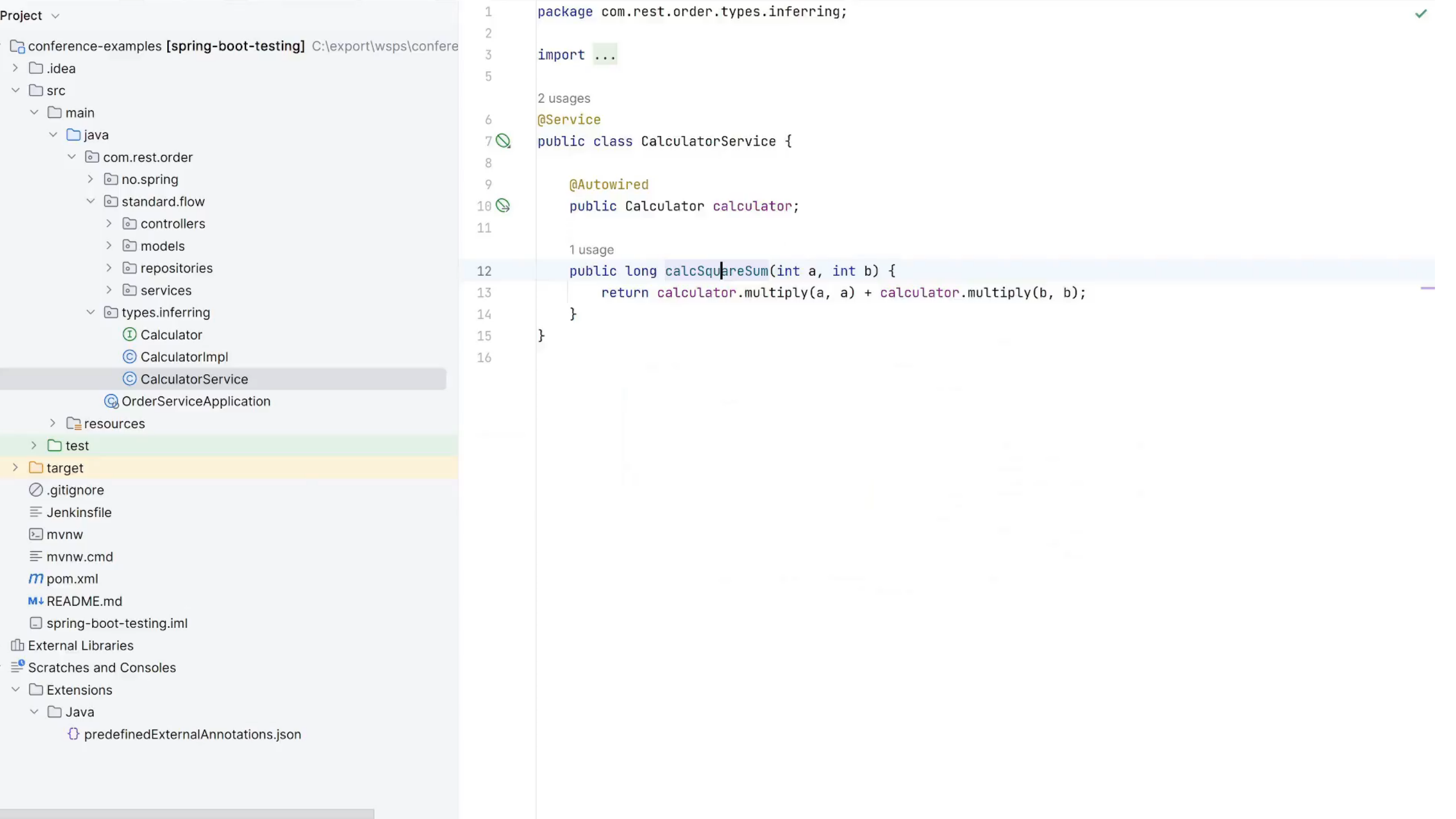
```
@Test
public void testIsNotEmpty1() {

    List listMock = mock(List.class);
    (when(listMock.size())).thenReturn(0);
    (when(orderRepositoryMock.findAll())).thenReturn(listMock);

    boolean actual = orderService.isNotEmpty();

    assertFalse(actual);
}
```


НЕ КОЛИЧЕСТВО
КАЧЕСТВО
imgflip.com

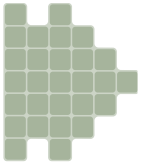
```java
@Service
public class OrderService {
    @Autowired
    private OrderRepository orderRepository;

    public boolean isMajorityExpensive(int threshold) {
       {…}
    }

    public boolean isNotEmpty() {
        List<Order> orders = orderRepository.findAll();
        return orders.size() > 0;
    }
}
```

```java
@Service
public class CalculatorService {

    @Autowired
    public Calculator calculator;

    public long calcSquareSum(int a, int b) {
        return calculator.multiply(a, a) + calculator.multiply(b, b);
    }
}
```

```
public void testCalcSquareSum1() {
    CalculatorService calculatorService = new CalculatorService();
    Calculator calculatorMock = mock(Calculator.class);
    (when(calculatorMock.multiply(anyInt(), anyInt()))).thenReturn(0L, 0L);
    calculatorService.calculator = calculatorMock;

    long actual = calculatorService.calcSquareSum(1, -255);

    assertEquals(0L, actual);
}
```


Project
conference-examples [spring-boot-testing] C:\export\wsps\confere
.idea
src
main
java
com.rest.order
no.spring
standard.flow
controllers
models
repositories
services
types.inferring
Calculator
CalculatorImpl
CalculatorService
OrderServiceApplication
resources
test
target
.gitignore
Jenkinsfile
mvnw
mvnw.cmd
pom.xml
README.md
spring-boot-testing.iml
External Libraries
Scratches and Consoles
Extensions
Java
predefinedExternalAnnotations.json
package com.rest.order.types.inferring;
import ...
2 usages
@Service
public class CalculatorService {
@Autowired
public Calculator calculator;
1 usage
public long calcSquareSum(int a, int b) {
return calculator.multiply(a, a) + calculator.multiply(b, b);
}
}

```java
@Test
public void testCalcSquareSum1() {
    CalculatorService calculatorService = new CalculatorService();
    CalculatorImpl calculator = new CalculatorImpl();
    calculatorService.calculator = calculator;

    long actual = calculatorService.calcSquareSum(1, -255);

    assertEquals(65026L, actual);
}
```



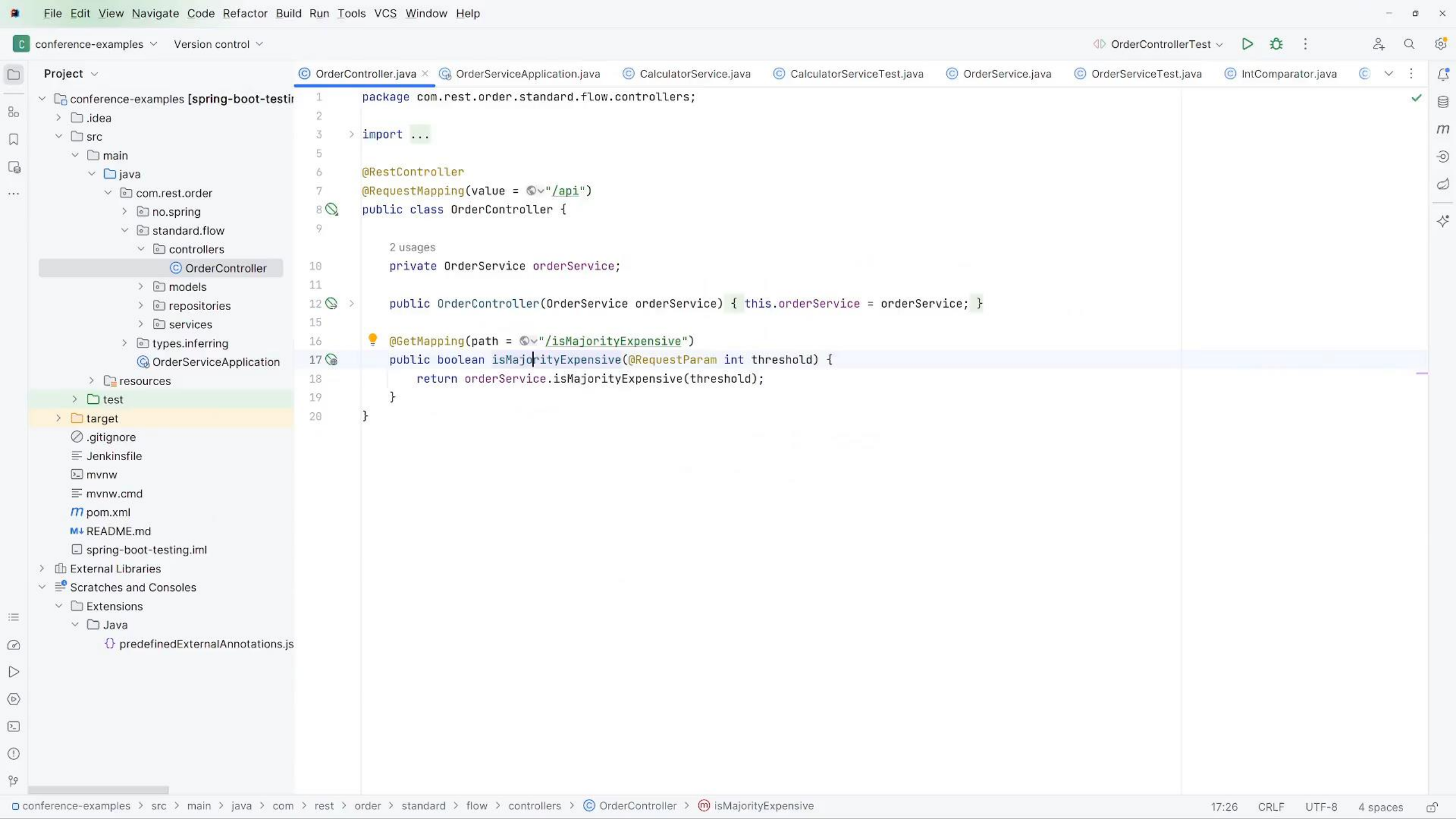
Безопасно?

Что делает плагин?

- Запуск приложения Spring
- Пост-обработка компонентов (bean)
- Получение определений компонентов (bean)
- Остановка приложений

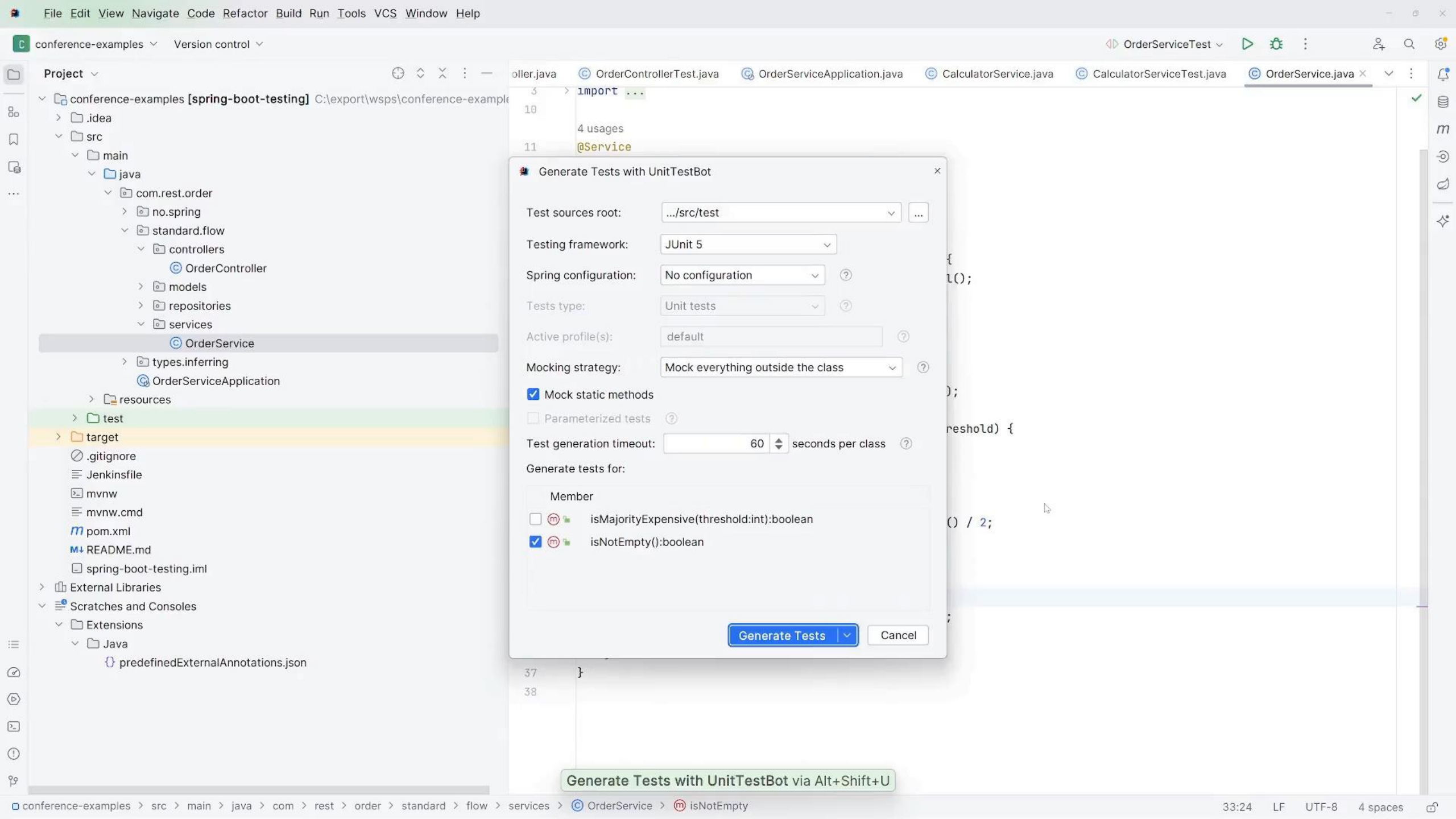
Конструирование bean'ов не началось

```java
@RestController
@RequestMapping(value = "/api")
public class OrderController {

    private OrderService orderService;

    public OrderController(OrderService orderService) {
        this.orderService = orderService;
    }

    @GetMapping(path = "/isMajorityExpensive")
    public boolean isMajorityExpensive(@RequestParam int threshold) {
        return orderService.isMajorityExpensive(threshold);
    }

}
```


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
conference-examples
Version control
OrderControllerTest
Project
conference-examples [spring-boot-testi
.idea
src
main
java
com.rest.order
no.spring
standard.flow
controllers
OrderController
models
repositories
services
types.inferring
OrderServiceApplication
resources
test
target
.gitignore
Jenkinsfile
mvnw
mvnw.cmd
pom.xml
README.md
spring-boot-testing.iml
External Libraries
Scratches and Consoles
Extensions
Java
predefinedExternalAnnotations.js
OrderController.java
OrderServiceApplication.java
CalculatorService.java
CalculatorServiceTest.java
OrderService.java
OrderServiceTest.java
IntComparator.java
1 package com.rest.order.standard.flow.controllers;
2
3 import ...
5
6 @RestController
7 @RequestMapping(value = "/api")
8 public class OrderController {
9
2 usages
10 private OrderService orderService;
11
12 public OrderController(OrderService orderService) { this.orderService = orderService; }
15
16 @GetMapping(path = "/isMajorityExpensive")
17 public boolean isMajorityExpensive(@RequestParam int threshold) {
18 return orderService.isMajorityExpensive(threshold);
19 }
20 }
conference-examples > src > main > java > com > rest > order > standard > flow > controllers > OrderController > isMajorityExpensive
17:26
CRLF
UTF-8
4 spaces

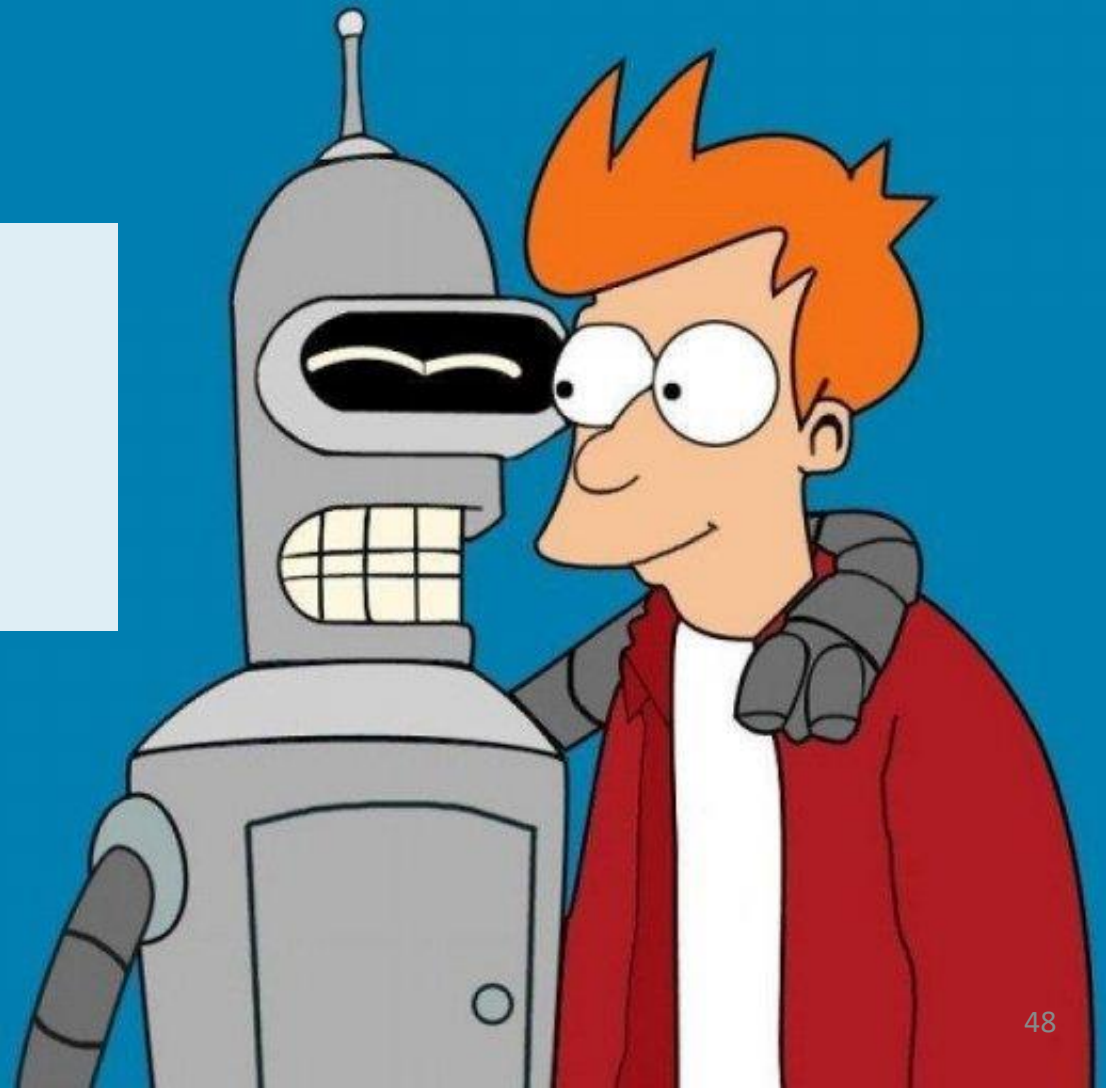
```
@Test
public void testIsMajorityExpensive1() throws Exception {
    UriComponentsBuilder uriComponentsBuilder = fromPath("/api/isMajorityExpensive");
    Object[] values = {4097};
    UriComponentsBuilder uriComponentsBuilder1 = uriComponentsBuilder.queryParam("threshold", values);
    String urlTemplate = uriComponentsBuilder1.toUriString();
    Object[] uriVariables = {};
    MockHttpServletRequestBuilder mockHttpServletRequestBuilder = get(urlTemplate, uriVariables);

    ResultActions actual = mockMvc.perform(mockHttpServletRequestBuilder);

    actual.andDo(print());
    actual.andExpect((status()).is(200));
    actual.andExpect((content()).string("false"));
}
```

```java
@Service
public class OrderService {

    @Autowired
    private OrderRepository orderRepository;

    public boolean isMajorityExpensive(int threshold) {…}

    public boolean isNotEmpty() {
        List<Order> orders = orderRepository.findAll();
        return orders.size() > 0;
    }

}
```


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
conference-examples
Version control
OrderServiceTest
Project
conference-examples [spring-boot-testing] C:\export\wsps\conference-example
.idea
src
main
java
com.rest.order
no.spring
standard.flow
controllers
OrderController
models
repositories
services
OrderService
types.inferring
OrderServiceApplication
resources
test
target
.gitignore
Jenkinsfile
mvnw
mvnw.cmd
pom.xml
README.md
spring-boot-testing.iml
External Libraries
Scratches and Consoles
Extensions
Java
predefinedExternalAnnotations.json
oller.java
OrderControllerTest.java
OrderServiceApplication.java
CalculatorService.java
CalculatorServiceTest.java
OrderService.java
import ...
4 usages
@Service
Generate Tests with UnitTestBot
Test sources root:
.../src/test
Testing framework:
JUnit 5
Spring configuration:
No configuration
Tests type:
Unit tests
Active profile(s):
default
Mocking strategy:
Mock everything outside the class
Mock static methods
Parameterized tests
Test generation timeout:
60
seconds per class
Generate tests for:
Member
isMajorityExpensive(threshold:int):boolean
isNotEmpty():boolean
Generate Tests
Cancel
{
l();
);
reshold) {
() / 2;
;
}
Generate Tests with UnitTestBot via Alt+Shift+U
conference-examples > src > main > java > com > rest > order > standard > flow > services > OrderService > isNotEmpty
33:24
LF
UTF-8
4 spaces

Integrity test

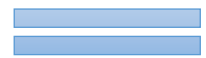
```
/**
 * This sanity check test fails if the
application context cannot start.
 */
@Test
public void contextLoads() {}
```



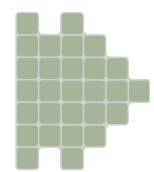
```
@SpringBootTest
@BootstrapWith(SpringBootTestContextBootstrapper.class)
@DirtiesContext(classMode = DirtiesContext.ClassMode.BEFORE_EACH_TEST_METHOD)
@Transactional
@AutoConfigureTestDatabase
@AutoConfigureMockMvc
public class OrderControllerTest {
```

Символьный анализ (**Symbolic**)

Конкретное исполнение (**Concrete**)



Concolic

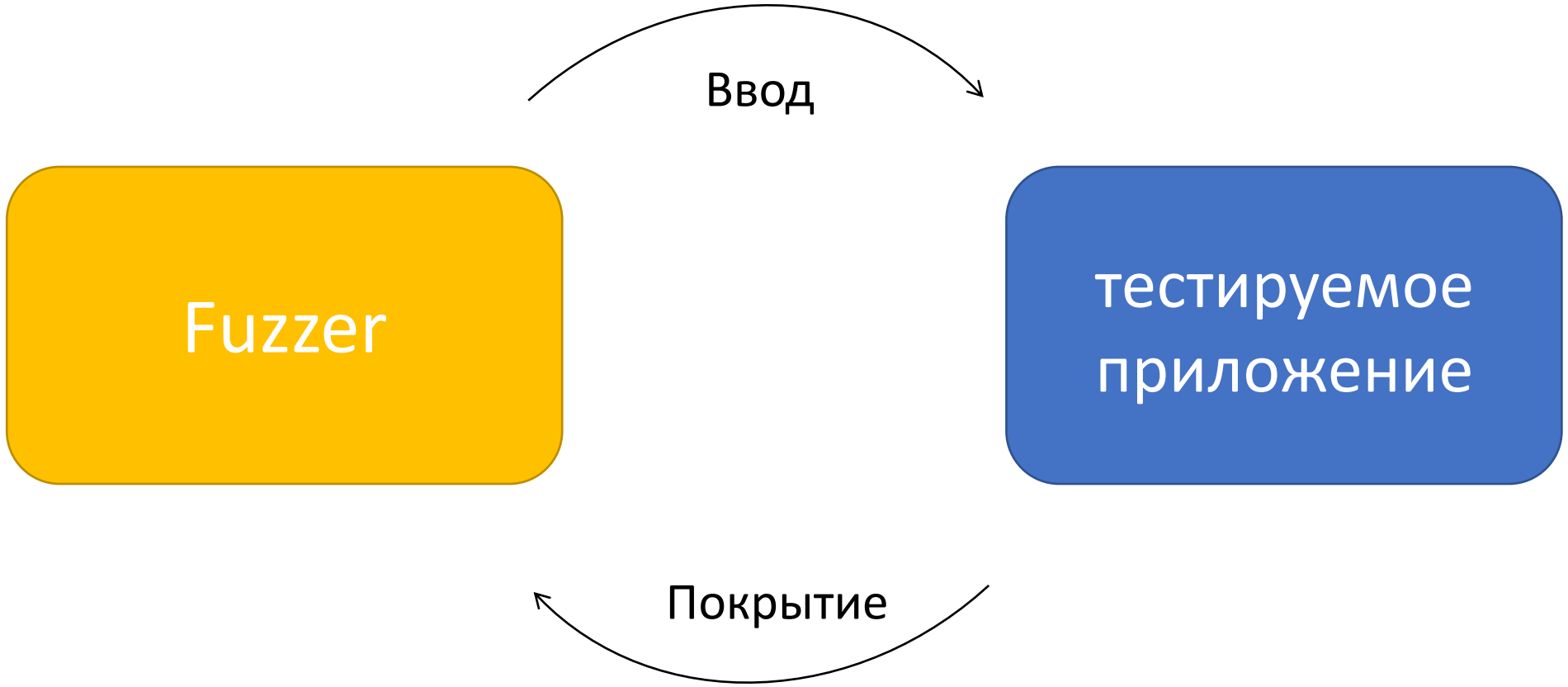


“Ай эм ёр фаззер”

Fuzze

естируемое
риложение

“Ай эм ёр фаззер”



“Ай эм ёр фаззер”

Fuzz

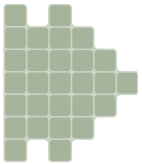
ДОКЛАД

Знакомство с фаззерами

Максим Пелевин

Huawei

тируемое
иложение

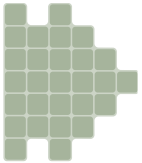


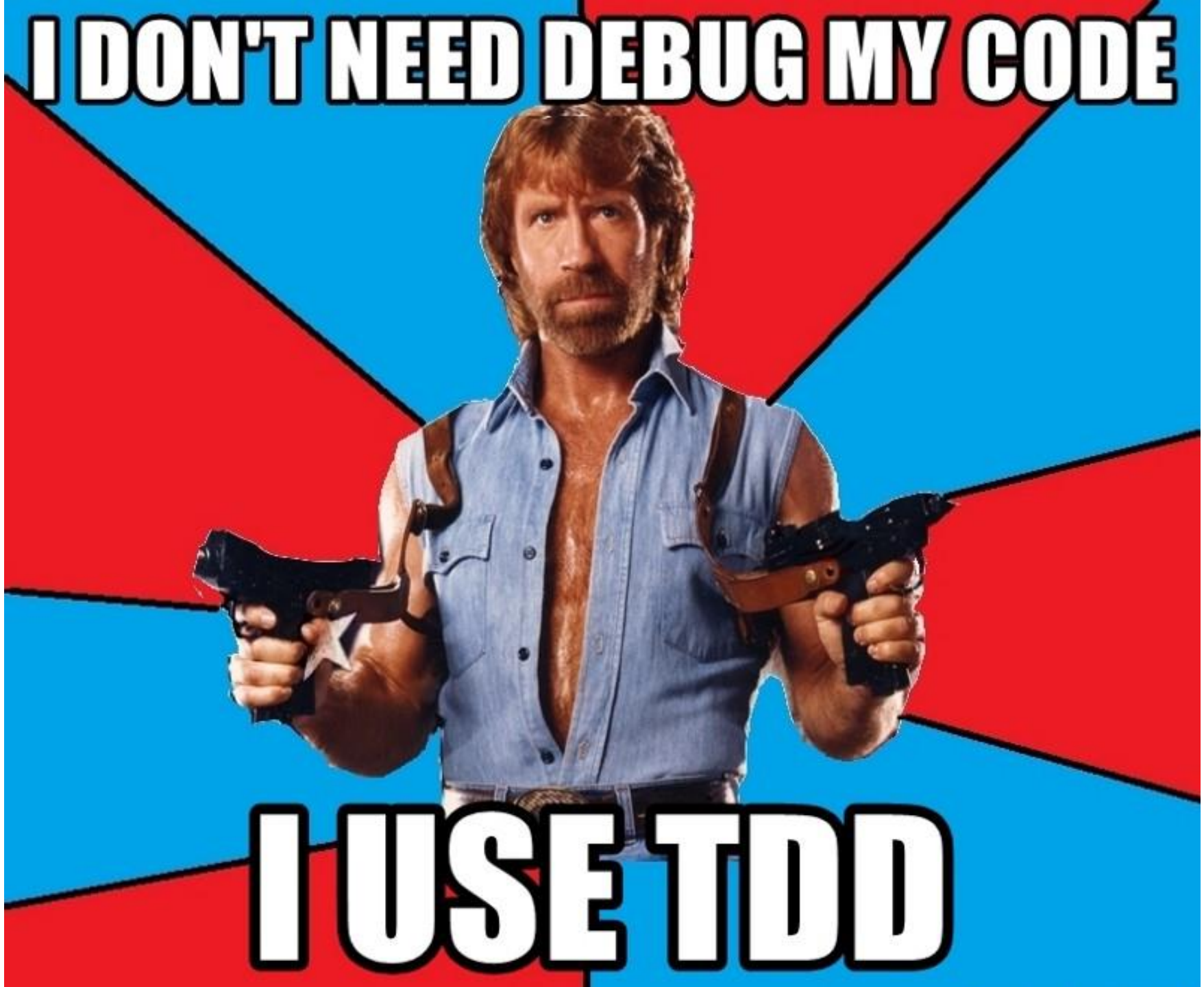
Безопасно?

Зоны риска

- Spring контекст
- Вызовы служб
- Обновления баз данных
- Файловая система

Не генерируйте тестов на продуктовой конфигурации




I DON'T NEED DEBUG MY CODE
I USE TDD

KLEE: Unassisted and Automatic Generation of High-Coverage Tests for Complex Systems Programs

Cristian Cadar, Daniel Dunbar, Dawson Engler*
Stanford University

Abstract

We present a new symbolic execution
pable of automatically generating
high coverage on a diverse set
environmentally-intensive program
to all 90 programs in the GNU Core

uts with symbolic values and replace cor-
te program operations with ones that
lues. When program execution
ic value, the system (con-
once, maintaining on
path condition
hen a path

grams in existence. For 84% of these utilities, KLEE's automatically generated tests covered 80–100% of executable statements and, in aggregate, significantly beat the coverage of the developers' own hand-written test suites. KLEE also found nine serious bugs (including

Плюсы генераторов тестов

- Фиксация поведения
- Фиксация ошибок (Дополнительные инспекции)
- Автоматические тесты лучше, чем их отсутствие
- Можно переписать, если будет время

Before

Element	Class, %	Method, %	Line, %
com	79% (117/147)	70% (955/1358)	31% (2087/6697)
h	79% (117/147)	70% (955/1358)	31% (2087/6697)
	79% (117/147)	70% (955/1358)	31% (2087/6697)
	57% (4/7)	69% (50/72)	71% (54/76)
	100% (0/0)	100% (0/0)	100% (0/0)
	100% (18/18)	95% (369/388)	83% (523/625)
	73% (30/41)	55% (170/308)	20% (559/2680)
	33% (1/3)	21% (3/14)	4% (3/68)
	100% (1/1)	100% (1/1)	100% (2/2)
	80% (25/31)	61% (176/285)	25% (495/1937)
	84% (16/19)	69% (104/149)	33% (176/530)
	100% (1/1)	75% (3/4)	91% (22/24)
	84% (21/25)	59% (79/132)	34% (253/737)
	0% (0/1)	0% (0/5)	0% (0/18)

After

Element	Class, %	Method, %	Line, %
com	86% (133/153)	81% (1194/1463)	64% (3754/5824)
h	86% (133/153)	81% (1194/1463)	64% (3754/5824)
	86% (133/153)	81% (1194/1463)	64% (3754/5824)
	57% (4/7)	69% (50/72)	71% (54/76)
	90% (18/20)	78% (197/252)	75% (339/450)
	100% (0/0)	100% (0/0)	100% (0/0)
	94% (16/17)	85% (306/356)	77% (456/592)
	81% (26/32)	90% (209/232)	63% (1221/1927)
	100% (3/3)	50% (7/14)	77% (44/57)
	100% (1/1)	100% (2/2)	100% (3/3)
	88% (22/25)	76% (172/226)	61% (823/1332)
	100% (19/19)	83% (126/151)	55% (291/525)
	100% (1/1)	100% (4/4)	95% (22/23)
	85% (23/27)	81% (121/149)	61% (501/821)
	0% (0/1)	0% (0/5)	0% (0/18)

Q&A