JUnit

фреймворк для автоматического юнит-тестирования приложений

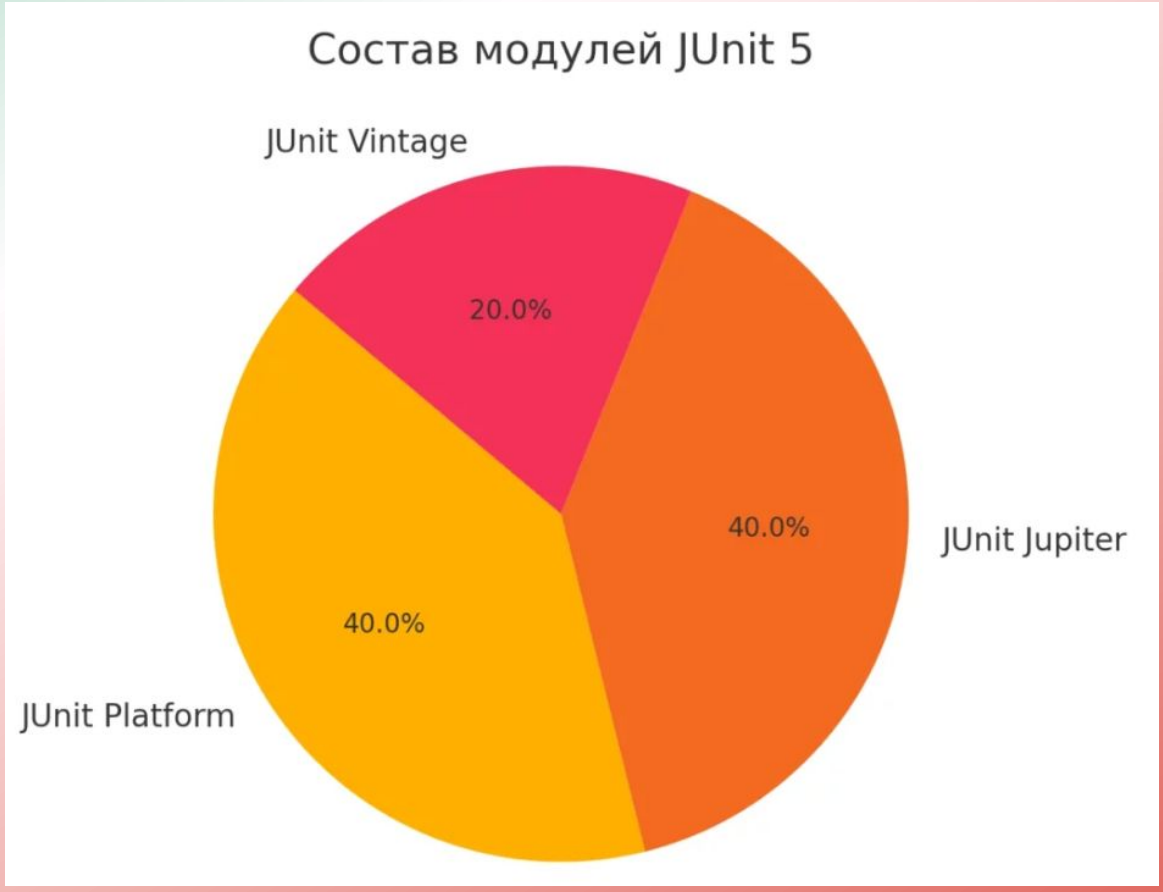
Выполнили:
Шутова Ю.
Быстров А.

JUnit 5

JUnit Platform — фундамент, на котором все держится

JUnit Jupiter — новый API для написания тестов.

JUnit Vintage — для поддержки старых тестов.



Основные аннотации в JUnit

@Test сообщает JUnit, что данный метод является тестовым.

@Before запускается **один раз перед всеми тест-кейсами.**

@After запускается **один раз после всех тест-кейсов**.

Другие: @BeforeClass, @AfterClass, @BeforeAll, @AfterAll, @Ignore

Calculator.java

```java
package proj;

public class Calculator {
    public int add(int a, int b) {
        return a+b;
    }
}
```

CalculatorTest.java

```java
package proj;

import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.Test;

class calculatorTest {

    @Test
    public void testAdd() {
        //Arrange
        Calculator calculator = new Calculator();
        int a = 2;
        int b = 2;

        //Act
        int result = calculator.add(a,b);

        //Assert
        assertEquals(4, result);
    }
}
```

CalculatorTest.java class CalculatorTest

```java
@Before
private void setup() {
    System.out.println("@Before - исполняет метод один раз перед всеми unit-тестами");
}

@After
private void done() {
    System.out.println("@After - исполняет метод один раз после всех unit-тестов");
}
```

```java
@Test
private void asserts() {
        assertEquals(1,1);
        assertTrue(1 == 1);
        assertNotNull(1);
}
```

Calculator.java class Calculator

```java
public int divide(int a, int b) {
    if (b == 0) {
        throw new IllegalArgumentException("Деление на ноль невозможно");
    }
    return a/b;
}
```

CalculatorTest.java class CalculatorTest

```java
@Test
void testDivide() {
    Calculator calculator = new Calculator();
    int a = 2;
    int b = 2;

    int result = calculator.divide(a,b);

    assertEquals(1, result);
}

@Test
void testDivideException() {
    Calculator calculator = new Calculator();
    int a = 10;
    int b = 0;

    Exception exception = assertThrows(IllegalArgumentException.class, () -> {
        calculator.divide(a,b);
    });
    assertEquals("Деление на ноль невозможно", exception.getMessage());
}
```

```java
@Ignore("not done yet")
    @Test
    public void testAdd() {
        Calculator calculator = new Calculator();
        int a = 2;
        int b = 2;

        int result = calculator.add(a,b);

        assertEquals(4, result);
    }
```

Параметризованные тесты

@ParameterizedTest — основная аннотация для параметризованных тестов

@CsvSource — данные в формате CSV
@ValueSource — простые значения
@EnumSource — для перечислений
@MethodSource — передача данных из метода

Если у вас много тестовых данных, удобнее хранить их в CSV-файле **@CsvFileSource**.

Calculator.java class Calculator

```
public int square(int a) {
        return a*a;
}
```

CalculatorTest.java class CalculatorTest

```java
@ParameterizedTest
@ValueSource(ints = {1, 2, 4, 10})
public void testAdd(int a) {
    Calculator calculator = new Calculator();

    assertNotNull(calculator.square(a));
}
```

```java
@ParameterizedTest
@CsvSource({
        "1, 1, 2",
        "100, 1, 101",
        "-5, 5, 0",
        "-3, 0, -3"
})
public void testAdd(int a, int b, int result) {
    Calculator calculator = new Calculator();

    assertEquals(result, calculator.add(a, b));
}
```

```java
@ParameterizedTest
@EnumSource(Month.class)
void testMonthValueBetween1and12(Month month) {
    int monthNumber = month.getValue();
    assertTrue(monthNumber >= 1 && monthNumber <= 12);
}
```

```java
static Stream<Arguments> numbersForAddition() {
    return Stream.of(
            Arguments.of(1,1,2),
            Arguments.of(2,3,5),
            Arguments.of(-1,-1,-2),
            Arguments.of(-10,1,-9)
            );

}

@ParameterizedTest
@MethodSource("numbersForAddition")
public void testAdd(int a, int b, int result) {
    Calculator calculator = new Calculator();

    assertEquals(result, calculator.add(a, b));
}
```

Параметризованные тесты

@ArgumentsSource — для сложных кастомных источников данных, когда встроенных аннотаций недостаточно. (Если нужны особые данные)

CustomArgumentsProvider.java

```java
package proj;

import java.util.stream.Stream;

import org.junit.jupiter.api.extension.ExtensionContext;
import org.junit.jupiter.params.provider.Arguments;
import org.junit.jupiter.params.provider.ArgumentsProvider;

public class CustomArgumentsProvider implements ArgumentsProvider{
    @Override
    public Stream<? extends Arguments> provideArguments(ExtensionContext context) throws Exception {
        return Stream.of(
                Arguments.of(1,1,2),
                Arguments.of(2,3,5),
                Arguments.of(-1,-1,-2),
                Arguments.of(-10,1,-9)
                );
    }

}
```

CalculatorTest.java class CalculatorTest

```java
@ParameterizedTest
@ArgumentsSource(CustomArgumentsProvider.class)
public void testAdd(int a, int b, int result) {
    Calculator calculator = new Calculator();

    assertEquals(result, calculator.add(a, b));
}
```

Источники

https://proselyte.net/tutorials/junit/introduction/

https://javascopes.com/junit5-tutorial-47c6aa6b/

https://coderlessons.com/articles/java/uchebnik-junit-po-modulnomu-testirovaniiu-rukovodstvo-ultimate-pdf-download

Спасибо за внимание!