

Тесты

Профессия Java-разработчик на Hexlet

Преподаватель: Яковлев Егор

Вопросы к лекции

- Как проверить работоспособность конечного продукта?
- Какие бывают тесты?
- Как писать тесты в Java?

План

1. Тестирование
2. Виды тестов
3. Ручные тесты vs Автоматические
4. Автоматические тесты: `assert`, `assertJ`
5. JUnit

Тестирование

“ Тесты — единственный надёжный способ убедиться в работоспособности кода ”

Виды тестов

- модульные (проверяют работоспособность конкретных программных модулей функций)
- интеграционные (проверяют, что модули правильно работают вместе)
- системные (имитацию работы всей системы целиком)

Ручные тесты vs Автоматические

```
//sum(x, y) - метод, который возвращает сумму двух чисел  
System.out.println(sum(1, 3)); // 4  
System.out.println(sum(-1, 9)); // 8
```

Автоматические тесты: первый заход

Демо

Assert

Каждая проверка, которую мы написали для функции, в тестировании принято называть **утверждением** (assert)

Ключевое слово **assert**

```
assert 4 == 2 + 2 // короткий вариант
```

```
assert 4 == 2 + 2 : "Суммирование работает некорректно" // расширенный вариант с текстом ошибки
```

Демо

Assert

```
javac HelloWorld.java  
java -ea HelloWorld
```

Демо

AssertJ vs assert

Минусы assert:

- -ea - не всегда срабатывают
- после assert - либо true, либо false
- результат проверки неинформативен

AssertJ

“ Тесты превращаются в связанный текст на английском языке ”

Демо

Модульное тестирование: пример

Демо

JUnit vs AssertJ

- AssertJ - это только библиотека - соответственно часть программы
- JUnit - фреймворк - фактически "вторая" программа

JUnit

- @BeforeEach
- @BeforeAll

Демо

Правила хороших тестов

- Тесты не должны влиять друг на друга
- **Не использовать if**
- Тесты вне тестов - @BeforeAll и @BeforeEach

Домашнее задание

```
hexlet program download java tests  
hexlet program submit java tests
```


Вопросы?