Санкт-Петербургский государственный университет

Кафедра системного программирования

Тестирование и Kotlin

Санкт-Петербург
2024

Аксиомы Шуры-Буры

1. Любая программа содержит ошибки
2. Если программа не содержит ошибок, то их содержит спецификация, по которой программа разрабатывалась
3. Если ни программа, ни спецификация ошибок не содержат, то такая программа никому не нужна

Много ли ошибок в коде?

- Согласно C. Kaner «Testing computer software» (1988)
  - после передачи в тестирование 1-3 ошибки на 100 строк
  - в процессе разработки 1.5 ошибки на 1 строку
  - исправление 10 операторов происходит верно в 50% случаев
- Числительные не столь принципиальны, важна картина в целом...

Тестирование

- Система — белый ящик
- Система — чёрный ящик

Ручное тестирование

- Наиболее дешёвое
- Требует несколько меньшей квалификации
- Тестирование UI сложно автоматизируется
- Можно быстро произвести в ходе разработки

Автоматизированное тестирование

- Требует навыков программирования
- Не любой код можно написать автотестируемым
- Тесты требуется поддерживать и сопровождать
- Чем чаще релизы, тем оно нужнее

Модульные (Unit) тесты

- Тестирование отдельного класса
- Проверяют внешнее поведение класса
- Полностью автоматические
- Направлены на поиск ошибок в конкретном методе
- Не влияют на функциональность системы и не поставляются пользователю

Три A

- Arrange
- Act
- Assert

Пример 1

```
public fun max(x: Int, y: Int): Int {
    if (x > y) {
        return x;
    } else {
        return y;
    }
}
```

Пример 2

```
public fun simpleCondition(c: SimpleDataClass): Int {
    if (c.a < 5 && c.b > 10) {
        return 0;
    } else {
        c.a = 3;
        return c.a;
    }
}
```

Пример 3

```
public fun byteArray(a: Array<Byte>, x: Byte): Byte {
    if (a.length != 2) {
        return -1;
    }
    a[0] = 5;
    a[1] = x;
    if (a[0] + a[1] > 20) {
        return 1;
    }
    return 0;
}
```

Kotlin

- JUnit 4, JUnit 5, TestNg — наследие Java
- AssertJ, AssertK
- Mocking: Mockito (Mockito-Kotlin), Powermock
- BDD Tests: kotest, spek

Best practices (Для Unit-тестов)

- Независимость тестов
  - Желательно, чтобы поломка одного куска функциональности ломала один тест
- Тесты должны работать быстро
  - И запускаться после каждой сборки (CI)
- Тестов должно быть много
  - Следить за Code coverage
- Каждый тест должен проверять конкретный тестовый сценарий
- Test-driven development

Test driven development (TDD)

Разработка непрерывно проходит следующие этапы:

- идея новой функциональности
- написание теста: как оно должно работать
- проверка, что тесты падают
- первичная реализация
- отладка до прохождения теста
- проверка прохождения предыдущих тестов
- рефакторинг
- проверка прохождения теста