Jmockit

-Group 2-

Our Team

SE172607 - Nguyễn Nhật Phát

SE172565 - Bùi Thanh Tú

SE170526 - Trịnh Quốc Thái

SE170092 - Nguyễn Viết Nam Khánh

SE172605 - Hoàng Văn Anh Nghĩa

-Group 2-

Overview of JMockit

What

JMockit simplifies Java unit testing by providing tools to create and manage mock objects. Developers can use these mock objects to isolate and test specific components in their source code.

Why

- The primary purpose of jMockit is to simplify the process of writing effective and flexible tests for Java code.
- Enables developers to create mock objects, define expectations, and verify interactions, enhancing the overall testability of the codebase.

Who

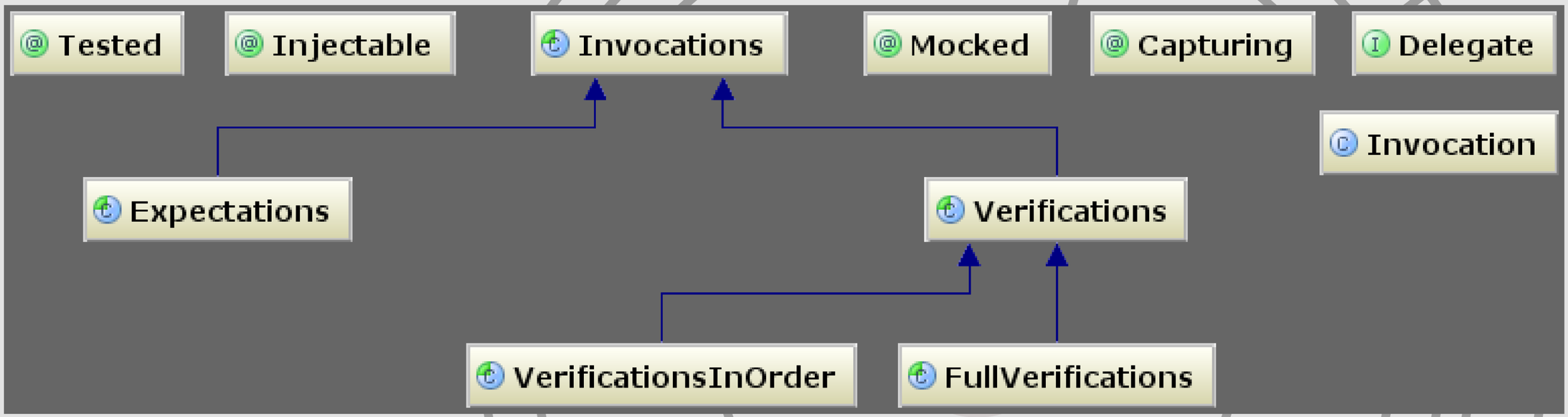
- Target Audience: Java developers and testing teams engaged in building and maintaining Java applications.
- Suitable for those who prioritize efficient and comprehensive testing practices in their software development workflow.

Customer Reference

- Software Development Enterprises
- Developers and Test Engineers

Architecture

Jmockit



```java
public class CalculatorTest {
    @Test
    public void testAdditionWithJMockit(@Mocked Calculator mockedCalculator) {
        // Setting expectations on the mocked method
        new Expectations() {{
            mockedCalculator.add(2, 3);
            result = 5;
        }};

        // Code under test
        Calculator calculator = new Calculator();
        int result1 = calculator.add(2, 3); // This should return the expected result (5)

        // Verifying invocations
        new Verifications() {{
            mockedCalculator.add(2, 3);
            times = 1; // Expecting this specific invocation exactly once
        }};

        // Assertions
        assertEquals(5, result1);
    }
```

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

Highlight of jmockit

- Dynamic Mocking Mechanism
- Annotation-Based Approach
- Rich Set of Annotations (@Mocked,@Tested,@Injectable,...)
- Flexible Mocking

Pros

- It allows mocking of static methods and final classes
- JMockit is far more complex, but more powerful than the other mocking frameworks

Cons

- Learning Curve for Beginners
- Less Popularity

Details of Features

1. Mocking Objects

Create mock objects for targeted testing.

2. Expectations

Define expected behaviors for mock objects.

3. Verification

Confirm method calls with specific parameters.

4. Capturing Arguments

Capture and inspect method arguments for validation.

5. Integration Testing

Support both unit and integration testing.

6. Code Coverage

Integrate with code coverage tools for comprehensive testing

DEMO 1

MOCK UP CLASS

DEMO 2

Expectations & Verifications


ANY QUESTION?

-GROUP 2-

THANK YOU !