**✅ MOCKITO Interview Questions and Answers**

**1. What is Mockito? How is it different from a mock object manually written?**

**Answer:**  
Mockito is a mocking framework for Java that allows you to create and configure mock objects for unit testing. It simplifies the creation of test doubles and verification of interactions.

**Difference:**

* Manual mocks require writing boilerplate code.
* Mockito generates mocks at runtime using proxying, reducing manual effort and boilerplate.

**Cross-question:**  
**Q:** Can you mock final classes or static methods in Mockito?  
**A:** Yes, from Mockito 3.4+ and above with inline mocking and additional configuration, final classes and static methods can be mocked.

**2. Explain the difference between @Mock, @InjectMocks, and @Spy.**

**Answer:**

* @Mock: Creates a mock instance of the class.
* @InjectMocks: Injects mock dependencies into the class.
* @Spy: Wraps a real object and allows partial mocking.

**Example:**

java

CopyEdit

@Mock private UserService userService;

@InjectMocks private UserController userController;

@Spy private User realUser = new User();

**Cross-question:**  
**Q:** What happens if you use @InjectMocks without @Mock?  
**A:** It tries to inject real objects unless mocks are provided. If dependencies are missing, it may result in NullPointerException.

**3. How do you verify a method is called a certain number of times?**

**Answer:**

java

CopyEdit

verify(service, times(2)).callMethod();

verify(service, never()).someOtherMethod();

verify(service, atLeastOnce()).log();

**Cross-question:**  
**Q:** What’s the difference between times(0) and never()?  
**A:** Both verify no interaction, but never() is more expressive for intent.

**4. How can you mock a void method using Mockito?**

**Answer:**

java

CopyEdit

doNothing().when(service).sendEmail(any());

doThrow(new RuntimeException()).when(service).notifyUser(any());

**Cross-question:**  
**Q:** Can you use when().thenReturn() on a void method?  
**A:** No. Use doXXX().when() syntax for void methods.

**5. Can you mock private methods with Mockito?**

**Answer:**  
Mockito by default does not support mocking private methods. However, with **PowerMockito** or **Mockito-inline** (from version 3.4+), it’s possible with configuration.

**6. What is ArgumentCaptor and when should it be used?**

**Answer:**  
ArgumentCaptor is used to capture method arguments passed during invocation to verify them.

**Example:**

java

CopyEdit

ArgumentCaptor<User> captor = ArgumentCaptor.forClass(User.class);

verify(userService).saveUser(captor.capture());

assertEquals("John", captor.getValue().getName());

**7. What is the purpose of MockitoAnnotations.openMocks(this)?**

**Answer:**  
It initializes the annotated mocks (@Mock, @Spy, @InjectMocks) in a test class. Typically used in @Before method.

**✅ MOCKSERVER Interview Questions and Answers**

**1. What is MockServer and when do you use it?**

**Answer:**  
MockServer is an API mocking framework used to simulate HTTP(s) APIs for testing and integration scenarios. It is used when:

* External services are not available.
* You want consistent, predictable responses.
* You want to simulate failure conditions (timeouts, error codes).

**2. How do you create an expectation in MockServer?**

**Answer:**

java

CopyEdit

mockServerClient.when(

request()

.withMethod("GET")

.withPath("/users")

).respond(

response()

.withStatusCode(200)

.withBody("{ \"name\": \"John\" }")

);

**3. How do you simulate different HTTP statuses like 404, 500 using MockServer?**

**Answer:**  
Use .withStatusCode(XXX) in the response setup.

java

CopyEdit

response().withStatusCode(404).withBody("Not Found");

response().withStatusCode(500).withBody("Internal Error");

**4. How can you simulate delay or timeout in MockServer?**

**Answer:**

java

CopyEdit

response()

.withDelay(TimeUnit.SECONDS, 5)

.withBody("Delayed response");

Or for socket timeout:

java

CopyEdit

error()

.withDropConnection(true)

.withDelay(TimeUnit.SECONDS, 10);

**5. Can MockServer match requests based on query parameters, headers, or body content?**

**Answer:**  
Yes. You can match based on:

* Headers: withHeader("Authorization", "Bearer token")
* Query parameters: withQueryStringParameter("type", "admin")
* Body content: withBody(JsonBody.json("{ \"id\": 1 }"))

**6. How do you verify that a request was received in MockServer?**

**Answer:**

java

CopyEdit

mockServerClient.verify(

request()

.withMethod("GET")

.withPath("/status"),

VerificationTimes.exactly(1)

);

**7. How do you reset expectations in MockServer?**

**Answer:**

java

CopyEdit

mockServerClient.reset();

**8. Can you run MockServer as a Docker container or standalone?**

**Answer:**  
Yes. You can run it:

* As a JAR (mockserver-netty)
* As a Docker container:

bash

CopyEdit

docker run -d -p 1080:1080 mockserver/mockserver

**✅ Real-World Scenario-Based Question**

**Q: Suppose your microservice depends on 3 downstream services. How would you test it using Mockito and MockServer?**

**Answer:**

* Use **Mockito** for unit testing individual components with mocked dependencies.
* Use **MockServer** for **integration testing** to simulate all 3 downstream services with:
  + Different responses (success, failure)
  + Delay and timeouts
  + Header and query validation
  + API contract testing

**✅ MOCKITO – Senior-Level Interview Q&A**

**1. How do you architect tests in a layered microservice using Mockito?**

**Answer:**  
I follow a tiered approach:

* **Unit Test**: Service and DAO classes are tested using @Mock and @InjectMocks.
* **Integration Test**: With actual Spring context using @SpringBootTest, real DB or H2, and sometimes Mockito for mocking only specific beans.
* **Contract Test**: I mock external services with WireMock or MockServer.

**Cross-question:**  
**Q:** When should you not use Mockito?  
**A:** When verifying real HTTP interaction, persistence behavior, threading, or scheduler jobs — Mockito isn’t suitable for system or black-box testing.

**2. Mockito vs Spy vs Partial Mocks – Real-time Use Case?**

**Answer:**

* I use **@Spy** when I need to verify only certain behavior in an otherwise real object — like tracking audit log invocations in a real LoggerService.
* I use **partial mocking** if the method being tested depends on a specific method but mocking all behavior would defeat the test purpose.

java

CopyEdit

UserService realService = new UserService();

UserService spyService = spy(realService);

doReturn("mocked").when(spyService).getUserName();

**3. Can Mockito mock chained/fluent interfaces or builder patterns?**

**Answer:**  
Yes, but it requires deeper configuration. Mockito needs method stubbing for each level of chaining.

java

CopyEdit

when(builder.setX(any()).setY(any()).build()).thenReturn(obj);

But for complex chains, I prefer creating a TestBuilder utility instead of mocking.

**Cross-question:**  
**Q:** How do you refactor code that’s hard to mock due to chaining?  
**A:** Use wrapper classes (adapter pattern), or inject builders as beans to make them mockable.

**4. How do you test concurrency and threading with Mockito?**

**Answer:**  
Mockito itself doesn’t test concurrency. But I:

* Use **CountDownLatch** to synchronize thread behavior.
* Use **Mockito verify(timeout())** to test async methods:

java

CopyEdit

verify(service, timeout(1000)).updateUserStatus();

For real concurrency validation, I prefer tools like Awaitility or real stress/load tests.

**5. How do you handle mocking for static, final, or private methods in legacy systems?**

**Answer:**

* For **static methods**, use Mockito.mockStatic() (Mockito 3.4+).
* For **private methods**, refactor if possible. Else, use PowerMockito.
* For **final classes**, enable mock-maker-inline.

java

CopyEdit

try (MockedStatic<Utility> mocked = mockStatic(Utility.class)) {

mocked.when(() -> Utility.doSomething()).thenReturn("Mocked");

}

**Cross-question:**  
**Q:** Is PowerMockito still recommended?  
**A:** No. Mockito-inline is preferred due to better support, fewer hacks, and better community backing.

**6. How do you integrate Mockito in CI/CD pipelines?**

**Answer:**

* Ensure unit tests run with **JUnit + Mockito** in a CI pipeline.
* Use **Surefire reports**, **JaCoCo** for code coverage.
* Use **SonarQube** to enforce mock coverage and test quality.

**✅ MOCKSERVER – Senior-Level Interview Q&A**

**1. Explain how you use MockServer for contract testing in microservices.**

**Answer:**

* I use **MockServer to mock downstream services** in contract tests.
* Expectations are created based on API contracts (OpenAPI/Swagger).
* I verify the interaction and structure of requests (headers, body, auth tokens).
* It allows testing consumer-provider boundaries without the real service being up.

**2. How do you inject MockServer in Spring Boot integration tests?**

**Answer:**

* I start a MockServer on a random port in @BeforeAll.
* I override RestTemplateBuilder or WebClient base URL with the MockServer port using Spring profiles or dynamic property injection.
* I shutdown server in @AfterAll.

java

CopyEdit

@DynamicPropertySource

static void configureMockServer(DynamicPropertyRegistry registry) {

registry.add("external.api.url", () -> "http://localhost:" + mockServer.getPort());

}

**3. How do you simulate flaky APIs, connection resets, or throttling with MockServer?**

**Answer:**

java

CopyEdit

mockServerClient

.when(request().withPath("/payment"))

.error()

.withDropConnection(true)

.withDelay(TimeUnit.SECONDS, 3);

You can also simulate:

* **Rate limit**: 429 response
* **Socket timeout**: drop connection
* **Delay**: .withDelay(...)
* **Response time throttling**: .withConnectionOptions(...)

**4. What’s the difference between MockServer and WireMock? Which do you prefer?**

**Answer:**

| **Feature** | **MockServer** | **WireMock** |
| --- | --- | --- |
| Java DSL | Yes | Yes |
| HTTPS Support | Yes | Yes |
| Request Verification | Yes | Yes |
| Docker Support | Yes | Yes |
| Admin UI | No | Yes |
| Best For | System/Contract Tests | Consumer Driven Tests |

**I prefer MockServer** for:

* Programmatic Java DSL
* Real-time dynamic behavior
* Complex scenarios like delay, drop, etc.

**5. How do you test retry logic using MockServer?**

**Answer:**  
Simulate first few failures and then success:

java

CopyEdit

mockServerClient

.when(request().withPath("/retries"), Times.exactly(2))

.respond(response().withStatusCode(500));

mockServerClient

.when(request().withPath("/retries"))

.respond(response().withStatusCode(200).withBody("Success"));

**6. How do you use MockServer in a multi-service CI/CD pipeline?**

**Answer:**

* Run it in Docker container in the CI environment.
* Each service test connects to it as a mock of dependent APIs.
* Helps test integrations without deploying full stack.
* Assertions are made on request verification and response matching.

**✅ Summary**

| **Tool** | **Senior Concepts You Should Highlight** |
| --- | --- |
| **Mockito** | Partial mocking, concurrency, static mocking, test architecture, pipeline integration |
| **MockServer** | Contract testing, request verification, failure simulation, dynamic expectations, CI integration |