JUnit 5 & Mockito Annotations Cheat Sheet (Basic to Advanced)

JUnit 5 Annotations

----

1. @Test

Marks a method as a test case.

Example:

```
@Test
void testSum() { assertEquals(5, 2+3); }
```

2. @BeforeEach

Runs before each test; used for setup.

```
@BeforeEach
void init() { service = new UserService(); }
```

3. @AfterEach

Runs after each test; cleanup.

```
@AfterEach
void teardown() { service = null; }
```

4. @BeforeAll

Runs once before all tests (STATIC method).

```
@BeforeAll
static void setupAll() { System.out.println("Start tests"); }
```

5. @AfterAll

Runs once after all tests are completed.

```
@AfterAll
static void done() { System.out.println("End tests"); }
```

6. @DisplayName

Gives test a readable name.

```
@DisplayName("Check Sum Function")
@Test
void sumTest() {}
```

7. @Disabled

Skips the test case.

```
@Disabled("Pending implementation")
@Test
void testLogin() {}
```

8. @RepeatedTest

Runs the same test multiple times.

@RepeatedTest(5)

void repeatTest() {}

9. @Nested

Group related tests inside inner class.

@Nested

class AuthTests {

@Test void login() {}

}

10. @Tag

Used for grouping tests.

@Tag("fast")

@Test

void fastTest() {}

11. @ParameterizedTest

Runs test multiple times with different values.

@ParameterizedTest

@ValueSource(strings = {"admin", "user"})

void testUsers(String name) {}

12. @ValueSource

Provides values to parameterized test.

@ValueSource(ints = {1, 2, 3})

13. @CsvSource

Pass multiple arguments.

@ParameterizedTest

@CsvSource({"1,2,3", "2,3,5"})

void addTest(int a, int b, int expected) {}

14. @MethodSource

Provides complex objects.

static Stream provideData() {}

Mockito Annotations

----

1. @Mock

Creates a mock object.

@Mock

UserRepository repo;

2. @InjectMocks

Injects mocks into the class being tested.

@InjectMocks

UserService service;

3. @Spy

Partial mocking (calls real methods unless stubbed).

@Spy

UserService realService;

4. @Captor

Captures method arguments.

@Captor

ArgumentCaptor captor;

5. @ExtendWith(MockitoExtension.class)

Required for enabling Mockito annotations.

@ExtendWith(MockitoExtension.class)

6. when(...).thenReturn(...)

Stubbing method behaviour.

when(repo.findUser("Arjun")).thenReturn(user);

7. verify(...)

Check interaction calls.

verify(repo, times(1)).findUser("Arjun");

8. doReturn().when(...)

Used when spying real methods.

doReturn(5).when(service).getCount();

9. doThrow(...)

For throwing exceptions.

doThrow(new RuntimeException()).when(repo).deleteUser(1);

10. doAnswer(...)

Custom behavior.

doAnswer(inv -> "Mocked").when(repo).method();

Advanced Mockito

----

1. @Mock(answer = Answers.RETURNS_DEEP_STUBS)

Repo mock = mock(Repo.class, RETURNS_DEEP_STUBS);

2. InOrder verification

```
InOrder inOrder = inOrder(repo1, repo2);
inOrder.verify(repo1).step1();
inOrder.verify(repo2).step2();
```

3. Timeout verification

```
verify(repo, timeout(1000)).save();
```

4. Argument Matchers

```
when(repo.find(anyString())).thenReturn(user);
```

Practical Example

```
@Test
void testCreateUser() {
when(repo.save(any(User.class))).thenReturn(new User(1, "Arjun"));
User saved = service.createUser("Arjun");
assertEquals("Arjun", saved.getName());
verify(repo, times(1)).save(any(User.class));
}
```