

## B.Hanumanthu-ISTE60

### Multiple-choice questions (MCQs) without keys

Topic: Mockito and JUnit

Points:20

Time:20 mins

#### Instructions:

- 1.Download the document in the docx/word format
- 2.Highlight your response in yellow color
- 3.Share your complete screen while answering the test
- 4.Record your screen while answering the test
- 5.Upload your response in the pdf format in your folder in the drive
- 6.Evaluate your response on the uploaded copy and update the score-card sheet

1. What is Mockito used for in Java testing?
  - a) Code coverage analysis
  - b) Mocking objects for testing
  - c) Performance testing
  - d) Logging test results
2. In JUnit, what is the purpose of the `@Test` annotation?
  - a) Marking a method as a test method
  - b) Specifying the test class
  - c) Indicating the order of test execution
  - d) Ignoring a test method
3. Which of the following is a correct way to create a mock object using Mockito?
  - a) `MockObject mock = createMock(MockObject.class);`
  - b) `MockObject mock = new MockObject();`
  - c) `MockObject mock = Mockito.mock(MockObject.class);`
  - d) `MockObject mock = MockObject.createMock();`
4. What does the `@RunWith` annotation in JUnit allow you to do?
  - a) Run tests in parallel
  - b) Run tests with a specific test runner
  - c) Define test categories
  - d) Ignore test classes
5. In Mockito, what is the purpose of the `verify` method?
  - a) Verify the behavior of a mock object
  - b) Verify the code coverage of a method
  - c) Verify the execution order of test methods

- d) Verify the presence of a test case
6. How do you mock a method to throw a specific exception in Mockito?
- a) ``when(methodCall).thenThrow(Exception.class);``
  - b) ``when(methodCall).throwException(Exception.class);``
  - c) ``when(methodCall).thenReturn(Exception.class);``
  - d) ``when(methodCall).andThrow(Exception.class);``
7. What does the ``@Before`` annotation in JUnit indicate?
- a) It marks a method to run after each test case
  - b) It marks a method to run before each test case
  - c) It specifies the order of test execution
  - d) It ignores a test method
8. In Mockito, what is the purpose of the ``@Mock`` annotation?
- a) It marks a class as a mock class
  - b) It marks a method as a mock method
  - c) It injects mock objects into the test class
  - d) It defines the behavior of a mock object
9. How do you assert that an object is not null in JUnit?
- a) ``assertNull(obj);``
  - b) ``assertNotNull(obj);``
  - c) ``assertTrue(obj != null);``
  - d) ``assertFalse(obj == null);``
10. What is the purpose of the ``@Spy`` annotation in Mockito?
- a) It marks a method as a spy method
  - b) It marks a class as a spy class
  - c) It creates a partial mock of an object
  - d) It verifies the execution order of methods
11. In Mockito, what is the purpose of the ``@Captor`` annotation?
- a) It captures and verifies arguments for method calls on a mock
  - b) It marks a method as a capturing method
  - c) It captures exceptions thrown during tests
  - d) It verifies the order of method calls on a mock
12. Which JUnit annotation is used for parameterized testing?
- a) ``@ParameterizedTest``
  - b) ``@ParamTest``
  - c) ``@Parameterize``

d) `@Parameterized`

13. What is the purpose of the `@After` annotation in JUnit?

- a) It marks a method to run before each test case
- b) It marks a method to run after each test case
- c) It specifies the order of test execution
- d) It ignores a test method

14. How do you verify that a method was called a specific number of times in Mockito?

- a) `verify(methodCall).once();`
- b) `verify(methodCall).times(1);`
- c) `verify(methodCall).exact(1);`
- d) `verify(methodCall).count(1);`

15. What is the purpose of the `@Rule` annotation in JUnit?

- a) It marks a method as a rule method
- b) It specifies the order of test execution
- c) It defines a test rule that can be applied to test methods
- d) It ignores a test method

16. How do you mock a void method in Mockito?

- a) `when(methodCall).thenReturn();`
- b) `doNothing().when(methodCall);`
- c) `doReturn().when(methodCall);`
- d) `mockVoid(methodCall).execute();`

17. What is the purpose of the `@Ignore` annotation in JUnit?

- a) It marks a method as ignored, and it won't be executed
- b) It specifies the order of test execution
- c) It marks a method to run before each test case
- d) It marks a method to run after each test case

18. How do you reset a mock object in Mockito?

- a) `reset(mockObject);`
- b) `mockObject.reset();`
- c) `Mockito.reset(mockObject);`
- d) `resetMock(mockObject);`

19. Which JUnit assertion method is used to compare two objects for equality?

- a) `assertEquals()`
- b) `assertSame()`
- c) `assertEqual()`
- d) `assertMatch()`

20. What is the purpose of the `@ClassRule` annotation in JUnit?

- a) It marks a method as a class rule method
- b) It specifies the order of test execution
- c) It defines a class-level test rule that can be applied to test classes
- d) It ignores a test method