1. Which of the following annotations can be used to register Mockito with a JUnit 5 test class?
   1. @RunWith(MockitoJUnitRunner.class)
   2. @ExtendWith(MockitoExtension.class)
   3. @Mock
   4. @InjectMocks
2. Which of the following is the correct way to register Mockito with a JUnit 5 test class?
   1. @RunWith(MockitoJUnitRunner.class)
   2. @ExtendWith(MockitoExtension.class)
   3. @Mock
   4. @InjectMocks
3. What is the difference between @RunWith and @ExtendWith?
   1. @RunWith is used to select a specific runner for a JUnit test class. @ExtendWith is used to register an extension with a JUnit test class.
   2. @RunWith is deprecated in JUnit 5. @ExtendWith is the preferred way to register an extension with a JUnit 5 test class.
   3. @RunWith is used to register Mockito with a JUnit 5 test class. @ExtendWith is used to register Spring dependency injection with a JUnit 5 test class.
   4. @RunWith and @ExtendWith are two different annotations that can be used to register extensions with a JUnit test class.
4. How can you use Mockito to test a class that depends on another class?
   1. You can use the @InjectMocks annotation to inject a mock object into the test class.
   2. You can use the @Mock annotation to create a mock object for the class that the test class depends on.
   3. You can use the when() method to mock the methods of the mock object.
   4. You can use the verify() method to verify that the methods of the mock object were called in the expected way.
5. What is the difference between @Mock and @InjectMocks?
   1. @Mock is used to create a mock object. @InjectMocks is used to inject a mock object into a test class.
   2. @Mock is deprecated in JUnit 5. @InjectMocks is the preferred way to inject a mock object into a JUnit 5 test class.
   3. @Mock is used to mock a class that the test class depends on. @InjectMocks is used to mock a class that is used by the test class.
   4. @Mock and @Inject
   5. Mocks are two different annotations that can be used to create and inject mock objects into JUnit 5 test classes.
6. What is the difference between when() and doReturn()?
   1. when() is used to mock a method. doReturn() is used to stub a method.
   2. when() takes a lambda expression that defines the behavior of the mocked method. doReturn() takes a value that will be returned when the mocked method is called.
   3. when() is used to verify that a method was called. doReturn() is used to stub a method.
   4. when() and doReturn() are two different methods that can be used to mock and stub methods in Mockito.
7. What is a common mistake to avoid when using Mockito?
   1. Forgetting to mock a method that is called by the class that you are testing.
   2. Forgetting to verify that a method was called in the expected way.
   3. Using Mockito to mock methods that are not testable.
   4. All of the above.
8. Which of the following annotations can be used to create a parameterized test?
   1. @ParameterizedTest
   2. @Parameter
   3. @ValueSource
   4. @CsvSource
9. What is the difference between @ParameterizedTest and @Parameter?
   1. @ParameterizedTest is used to create a parameterized test. @Parameter is used to define a single parameter for a parameterized test.
   2. @ParameterizedTest is deprecated in JUnit 5. @Parameter is the preferred way to define a parameter for a parameterized test.
   3. @ParameterizedTest is used to create a parameterized test that can be used with multiple data sources. @Parameter is used to define a single parameter for a parameterized test that can only be used with a single data source.
   4. @ParameterizedTest and @Parameter are two different annotations that can be used to create and define parameters for parameterized tests.
10. Which of the following data sources can be used with @ParameterizedTest?
    1. ValueSource
    2. CSVSource
    3. CSVFileSource
    4. All of the above
11. What is the difference between ValueSource and CSVSource?
    1. ValueSource is used to provide a list of values for a single parameter. CSVSource is used to provide a CSV file that contains a list of values for a single parameter.
    2. ValueSource is deprecated in JUnit 5. CSVSource is the preferred way to provide a list of values for a single parameter in JUnit 5.
    3. ValueSource is used to provide a list of values for a single parameter that can be used with multiple data sources. CSVSource is used to provide a CSV file that contains a list of values for a single parameter that can only be used with a single data source.
    4. ValueSource and CSVSource are two different data sources that can be used with @ParameterizedTest.
12. What is the difference between CSVSource and CSVFileSource?
    1. CSVSource is used to provide a list of values from a CSV string. CSVFileSource is used to provide a list of values from a CSV file.
    2. CSVSource is deprecated in JUnit 5. CSVFileSource is the preferred way to provide a list of values from a CSV file in JUnit 5.
    3. CSVSource is used to provide a list of values from a CSV string that can be used with multiple data sources. CSVFileSource is used to provide a list of values from a CSV file that can only be used with a single data source.
    4. CSVSource and CSVFileSource are two different data sources that can be used with @ParameterizedTest.
13. How can you organize code with @Nested?
    1. You can use the @Nested annotation to group related tests together.
    2. You can use the @Nested annotation to create a nested test class.
    3. You can use the @Nested annotation to create a test class that depends on another test class.
    4. All of the above
14. What is the difference between a nested test class and a test class that depends on another test class?
    1. A nested test class is a test class that is contained within another test class. A test class that depends on another test class is a test class that uses the other test class.
    2. A nested test class is deprecated in JUnit 5. A test class that depends on another test class is the preferred way to organize related tests in JUnit 5.
    3. A nested test class can only be used with @ParameterizedTest. A test class that depends on another test class can be used with any type of test.
    4. A nested test class and a test class that depends on another test class are two different ways to organize related tests in JUnit.
15. What is a JUnit rule?
    1. A JUnit rule is an object that can be used to add functionality to a JUnit test class.
    2. A JUnit rule is a class that implements the org.junit.rules.TestRule interface.
    3. A JUnit rule can be used to execute code before or after each test method.
    4. All the above
16. The ExpectedException rule can be used to assert that an exception is thrown by a test method.
    1. True
    2. False
17. The Timeout rule can be used to assert that a test method does not take too long to execute.
    1. True
    2. False
18. The ExternalResource rule can be used to manage external resources, such as files or databases.
    1. True
    2. False
19. The TestWatcher rule can be used to watch a test method and execute code before, after, or during the execution of the method.
    1. True
    2. False
20. What is the difference between @Rule and @ClassRule?
    1. @Rule is used to declare a JUnit rule that will be applied to each test method in a test class. @ClassRule is used to declare a JUnit rule that will be applied to all test methods in a test class.
    2. @Rule is deprecated in JUnit 5. @ClassRule is the preferred way to declare a JUnit rule in JUnit 5.
    3. @Rule can only be used with nested test classes. @ClassRule can be used with any type of test class.
    4. @Rule and @ClassRule are two different annotations that can be used to declare JUnit rules.
21. What are some benefits of using JUnit rules?
    1. JUnit rules can be used to add functionality to a JUnit test class.
    2. JUnit rules can be used to make tests more readable and maintainable.
    3. JUnit rules can be used to improve the performance of tests.
    4. All of the above
22. What are some limitations of using JUnit rules?
    1. JUnit rules can make tests more complex.
    2. JUnit rules can make tests less portable.
    3. JUnit rules can make tests less efficient.
    4. All of the above
23. What is the best way to learn about JUnit rules?
    1. Read the documentation for JUnit rules.
    2. Look for examples of JUnit rules online.
    3. Experiment with different JUnit rules in your own code.
    4. All of the above
24. What are some best practices for using JUnit rules?
    1. Use only the JUnit rules that you need.
    2. Keep JUnit rules simple and easy to understand.
    3. Test JUnit rules in isolation.
    4. Document JUnit rules so that other developers can understand how to use them.
    5. All the above
25. What are some common mistakes to avoid when using JUnit rules?
    1. Using too many JUnit rules.
    2. Making JUnit rules too complex.
    3. Not testing JUnit rules in isolation.
    4. Not documenting JUnit rules.
26. What is the future of JUnit rules?
    1. JUnit rules are likely to become more popular in the future.
    2. JUnit rules are likely to be replaced by other technologies in the future.
    3. It is too early to say what the future holds for JUnit rules.
    4. The future of JUnit rules is uncertain.
27. What are some other advanced concepts of JUnit?
    1. JUnit Jupiter
    2. JUnit Vintage
    3. JUnit Platform
    4. All of the above
28. What is the difference between JUnit Jupiter and JUnit Vintage?
    1. JUnit Jupiter is the latest version of JUnit. It is a rewrite of JUnit that is designed to be more modern and flexible.
    2. JUnit Vintage is the previous version of JUnit. It is still supported, but it is not recommended for new development.
    3. JUnit Jupiter is a subset of JUnit Vintage.
    4. JUnit Vintage is a subset of JUnit Jupiter.
29. What is the JUnit Platform?
    1. The JUnit Platform is a framework that provides common infrastructure for JUnit testing frameworks. It includes the JUnit Runner, which is responsible for running tests, and the JUnit Assertions, which are used to assert the results of tests.
    2. The JUnit Platform is a test runner that can be used to run tests written in different programming languages.
    3. The JUnit Platform is a set of assertions that can be used to assert the results of tests.
    4. The JUnit Platform is a framework that provides common infrastructure for unit testing.
30. What is the difference between a test class and a test suite?
    1. A test class is a Java class that contains one or more test methods. A test suite is a collection of test classes.
    2. A test class is a Java class that contains one or more assertions. A test suite is a collection of assertions.
    3. A test class is a Java class that contains one or more methods. A test suite is a collection of methods.
    4. A test class is a Java class that contains one or more annotations. A test suite is a collection of annotations.
31. What is the difference between a static and a dynamic test?
    1. A static test is a test that is run without the need to start the application under test. A dynamic test is a test that requires the application under test to be started.
    2. A static test is a test that is run by a human. A dynamic test is a test that is run by a computer.
    3. A static test is a test that is run on the source code. A dynamic test is a test that is run on the compiled code.
    4. A static test is a test that is run on the binary code. A dynamic test is a test that is run on the running application.
32. What is the difference between a unit test and an integration test?
    1. A unit test tests a single unit of code, such as a class or method. An integration test tests how multiple units of code interact with each other.
    2. A unit test tests the entire application. An integration test tests a single unit of code.
    3. A unit test tests the functionality of the application. An integration test tests the performance of the application.
    4. A unit test tests the security of the application. An integration test tests the reliability of the application.
33. What is the difference between a functional test and a system test?
    1. A functional test tests the functionality of the application under test. A system test tests the entire system, including the application under test and its environment.
    2. A functional test tests the performance of the application under test. A system test tests the reliability of the application under test.
    3. A functional test tests the security of the application under test. A system test tests the scalability of the application under test.
    4. A functional test tests the maintainability of the application under test. A system test tests the usability of the application under test.
34. What is the difference between a black box test and a white box test?
    1. A black box test is a manual test. A white box test is an automated test.
    2. A black box test is a functional test. A white box test is a system test.
    3. A black box test tests the application under test without knowing its internal implementation. A white box test tests the application under test by knowing its internal implementation.
    4. A black box test is a unit test. A white box test is an integration test.
35. What is the difference between a smoke test and a regression test?
    1. A smoke test is a manual test. A regression test is an automated test.
    2. A smoke test is a unit test. A regression test is a system test.
    3. A smoke test is a functional test. A regression test is a performance test
    4. A smoke test is a quick test that is used to verify that the application under test is working as expected. A regression test is a test that is used to verify that changes to the application under test have not introduced any new defects.
36. What is the @Test annotation?
    1. The @Test annotation is used to mark a class as a test class.
    2. The @Test annotation is used to mark a method as a test method.
    3. The @Test annotation is used to mark a field as a test field.
    4. The @Test annotation is used to mark a constructor as a test constructor.
37. What is the assertThrows() method?
    1. The assertThrows() method is used to assert that a method throws an exception.
    2. The assertThrows() method is used to assert that a method does not throw an exception.
    3. The assertThrows() method is used to assert that a method returns a specific value.
    4. The assertThrows() method is used to assert that a method takes a specific number of arguments.
38. What is the @ParameterizedTest annotation?
    1. The @ParameterizedTest annotation is used to run the same test method with different data sets.
    2. The @ParameterizedTest annotation is used to run the same test method with different parameters.
    3. The @ParameterizedTest annotation is used to run the same test method with different assertions.
    4. The @ParameterizedTest annotation is used to run the same test method with different expected results.
39. What is the @Nested annotation?
    1. The @Nested annotation is used to create a nested test class.
    2. The @Nested annotation is used to group related tests together.
    3. The @Nested annotation is used to create a test class that depends on another test class.
    4. The @Nested annotation is used to create a test class that uses the same data set as another test class.
40. What is the @Rule annotation?
    1. The @Rule annotation is used to declare a JUnit listener.
    2. The @Rule annotation is used to declare a JUnit rule.
    3. The @Rule annotation is used to declare a JUnit fixture.
    4. The @Rule annotation is used to declare a JUnit assertion.