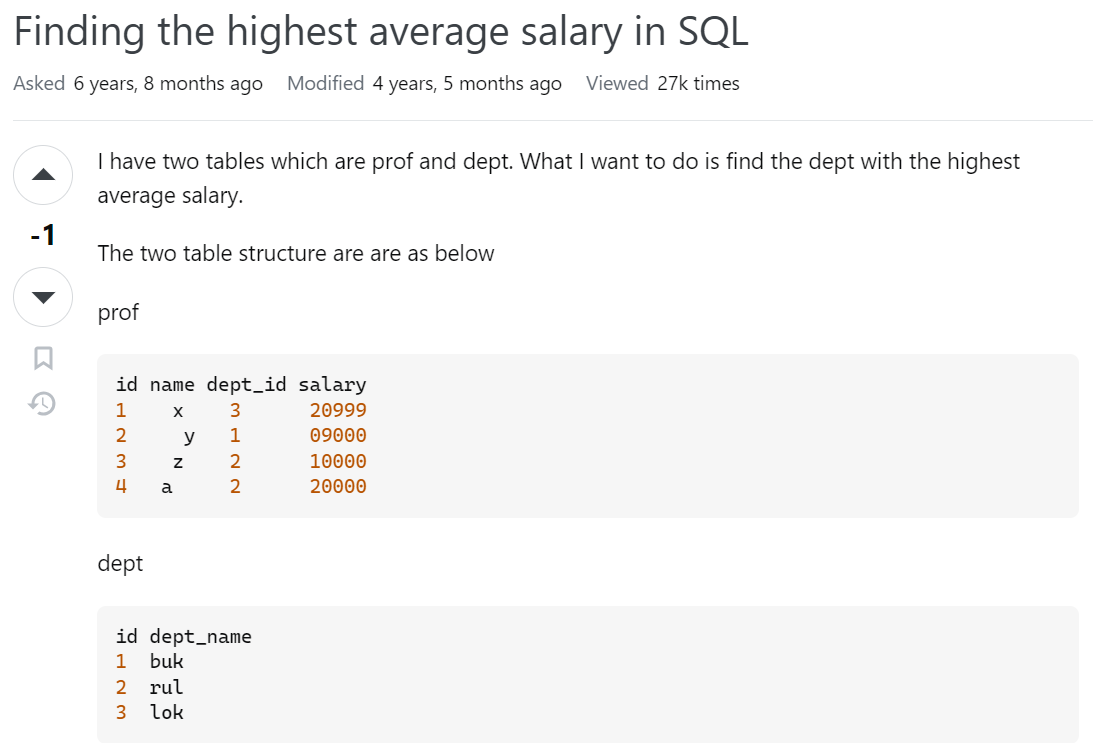
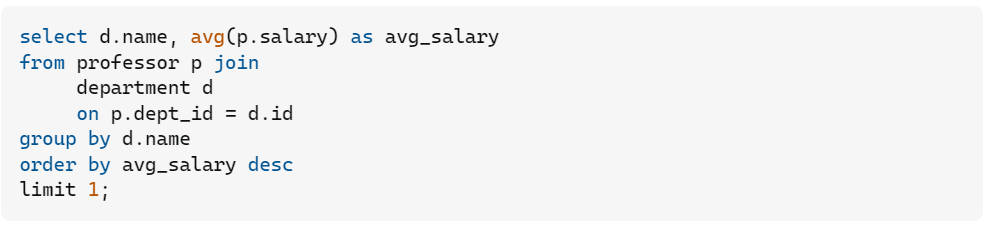
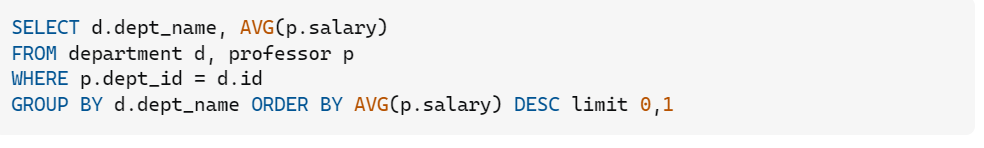
**Question 1st :- **

**Answer: -**

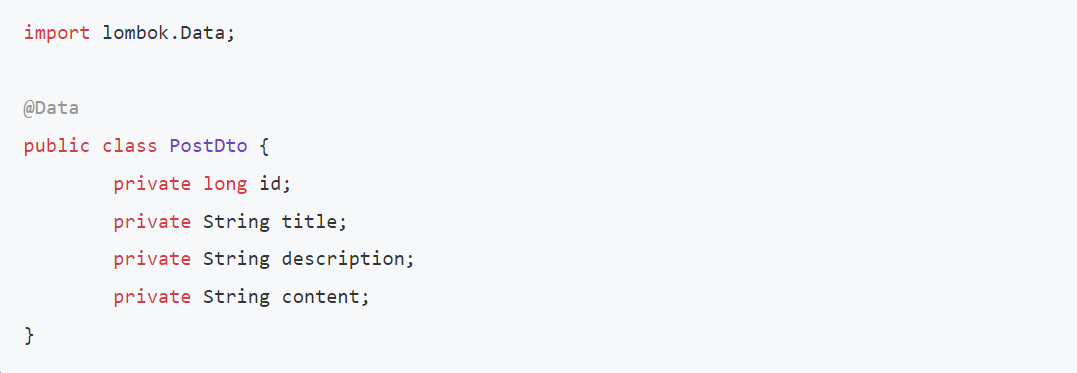
****

**OR**

****

**Question 2nd:- How to write Response DAO Class .**

**Ans:-**

****

**OR**

****

**Question 3ed: - write the unit test cases with all layer (service Layer) .**

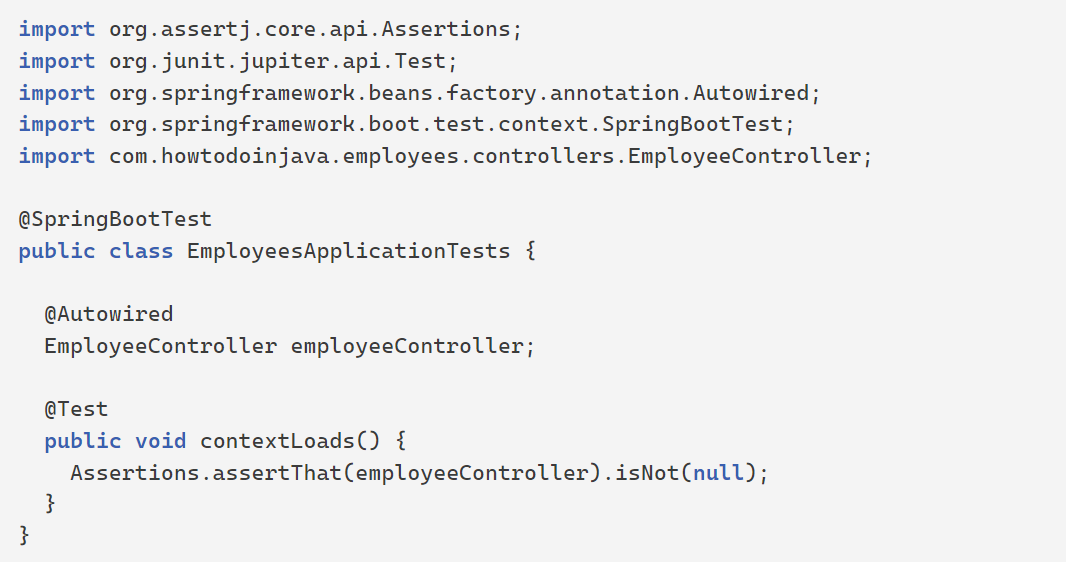
**Ans: -**

* Unit testing REST Controller with [@WebMvcTest](https://howtodoinjava.com/spring-boot2/testing/spring-boot-mockmvc-example/)
* Unit testing Service Layer with [Mockito](https://howtodoinjava.com/mockito/junit-mockito-example/)
* Unit testing DAO Layer with [@DataJpaTest](https://howtodoinjava.com/spring-boot2/testing/datajpatest-annotation/) and @AutoConfigureTestDatabase
* Integration testing using [@SpringBootTest](https://howtodoinjava.com/spring-boot2/testing/springboottest-annotation/)
* System testing using [RestTemplate](https://howtodoinjava.com/spring-boot2/resttemplate/spring-restful-client-resttemplate-example/)

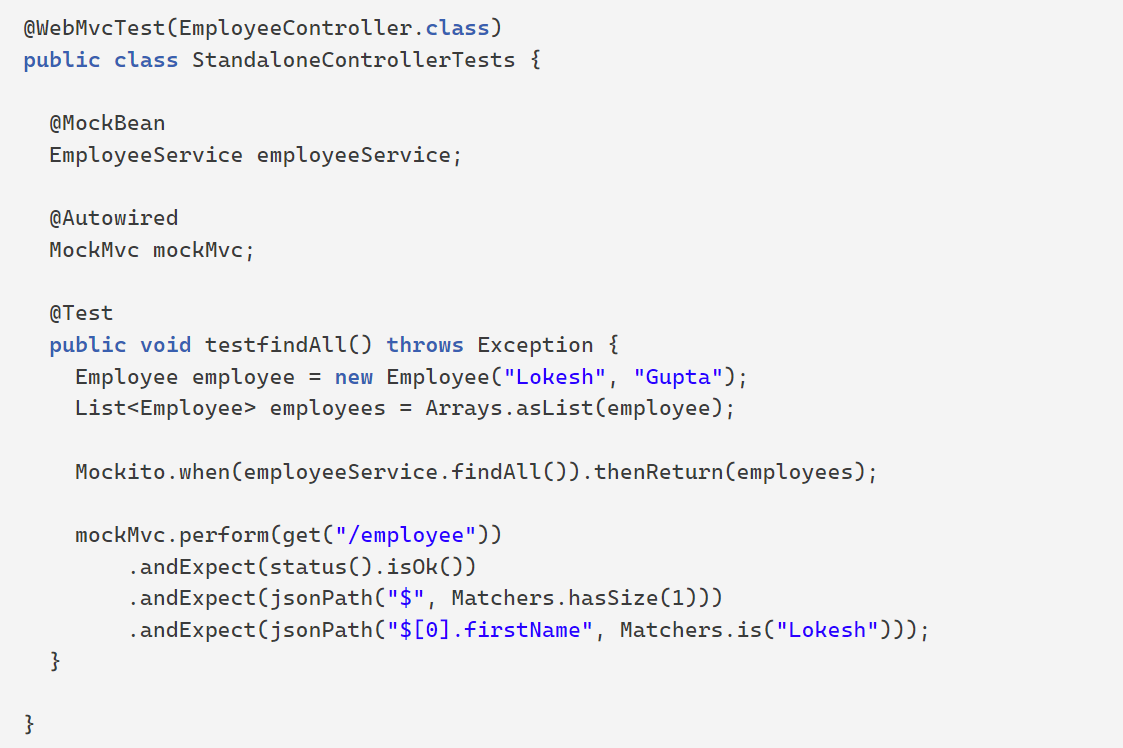
**Pom.xml dependency: -**

* spring-boot-starter-web
* spring-boot-starter-validation
* spring-boot-starter-test with Junit 5
* spring-boot-starter-data-jpa

****

****

****

****

****

@ExtendWith(MockitoExtension.**class**)

**public** **class** ServiceTests {

@InjectMocks

EmployeeService service;

@Mock

EmployeeRepository dao;

@BeforeEach

**public** **void** init() {

MockitoAnnotations.openMocks(**this**);

}

@Test

**void** testFindAllEmployees() {

List<Employee> list = **new** ArrayList<Employee>();

Employee empOne = **new** Employee("John", "John");

Employee empTwo = **new** Employee("Alex", "kolenchiski");

Employee empThree = **new** Employee("Steve", "Waugh");

list.add(empOne);

list.add(empTwo);

list.add(empThree);

when(dao.findAll()).thenReturn(list);

*//test*

List<Employee> empList = service.findAll();

assertEquals(3, empList.size());

verify(dao, times(1)).findAll();

}

@Test

**void** testCreateOrSaveEmployee() {

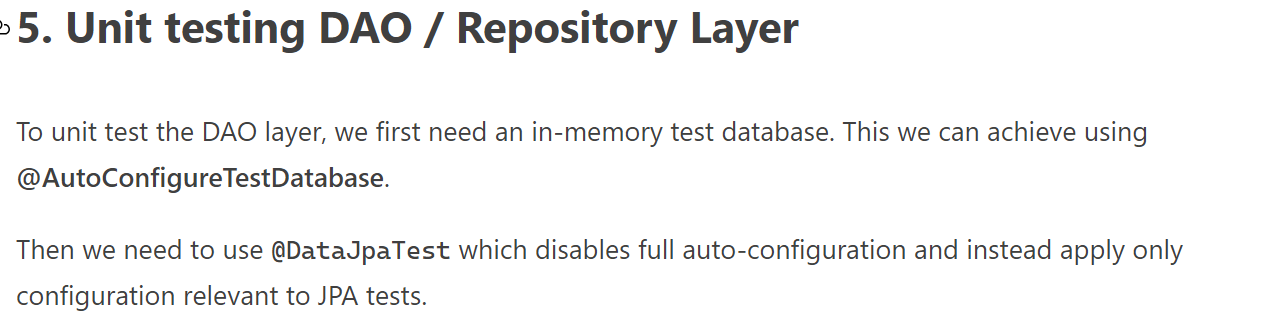
Employee employee = **new** Employee("Lokesh", "Gupta");

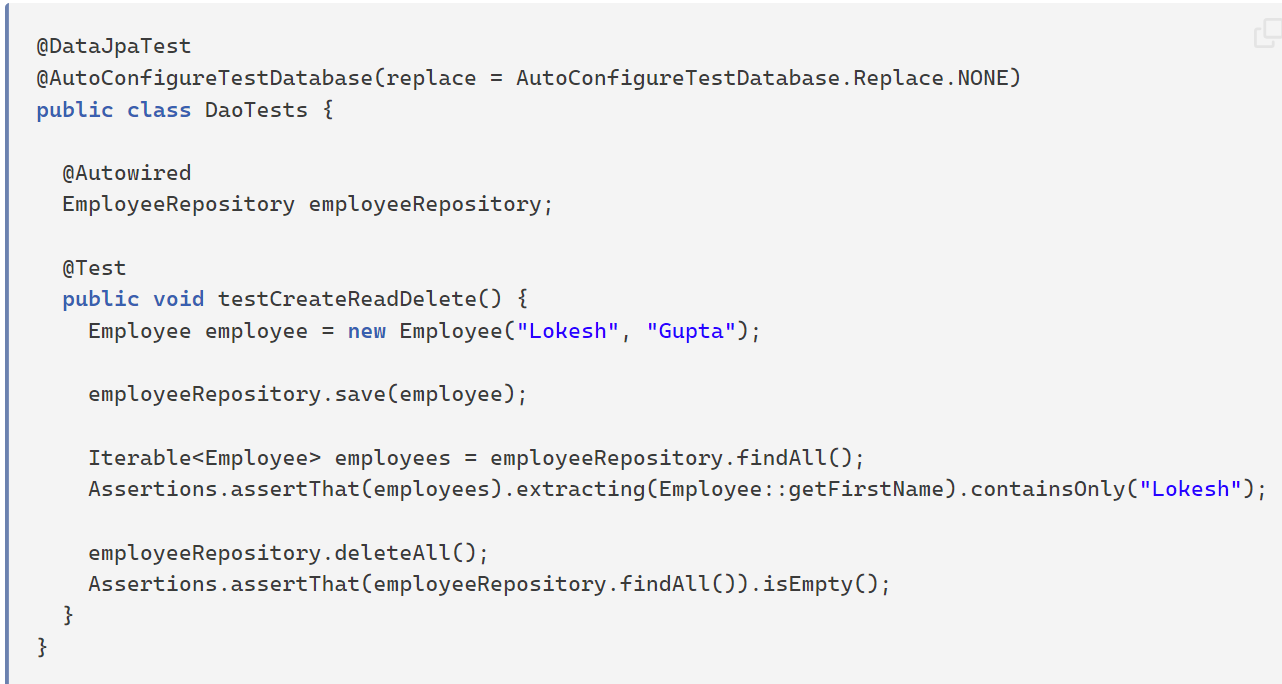
service.save(employee);

verify(dao, times(1)).save(employee);

}

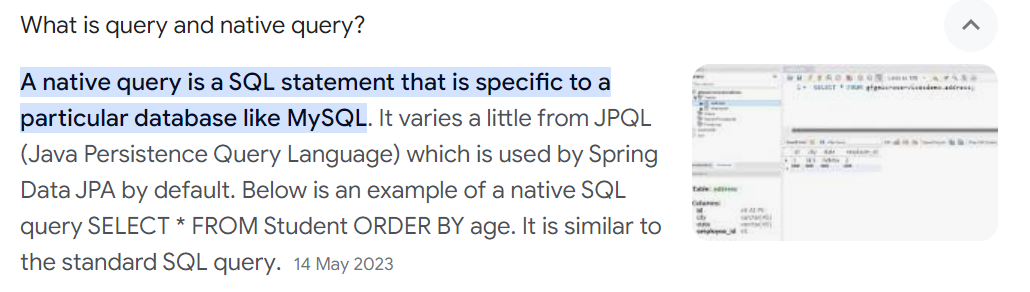
}



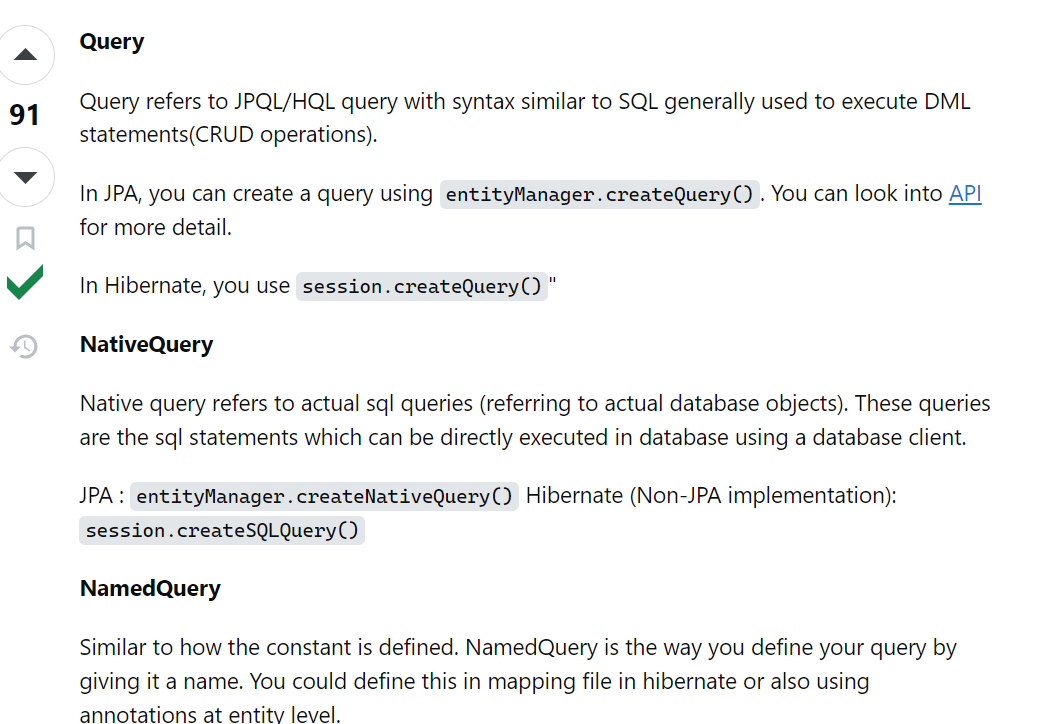


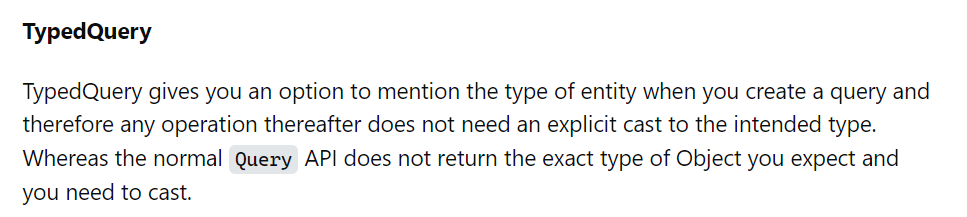
**Question 4th: - What is different b/w query and Native query.**

**Ans: -**

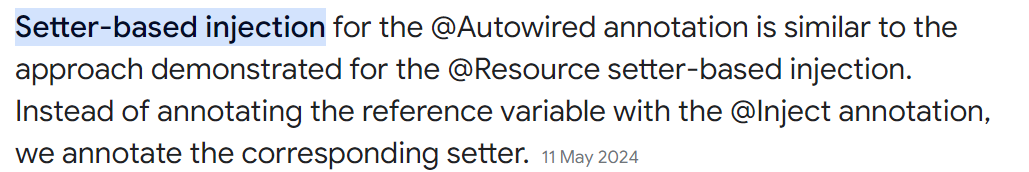
****

**OR**

****

****

**Question 5th: - @Autowired annotation used setter injection or construction Injection.**

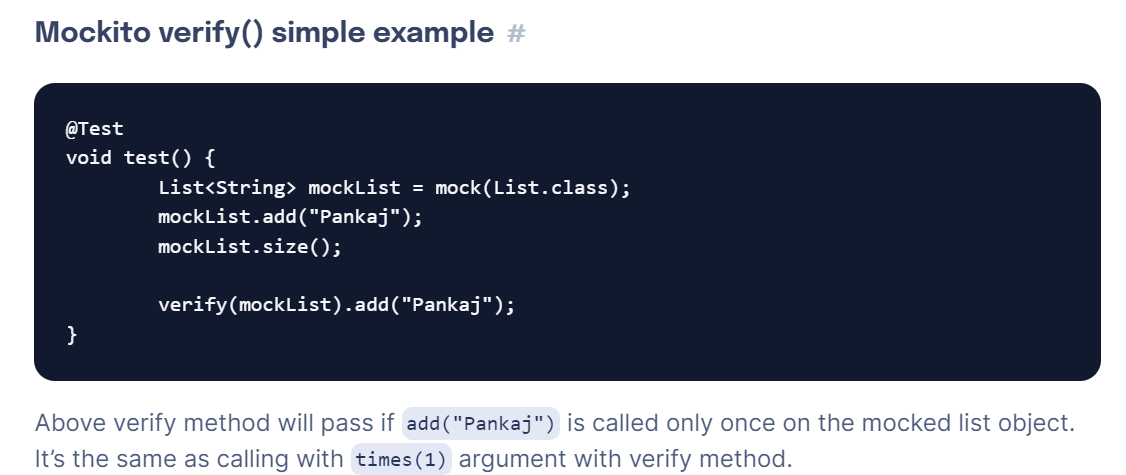
**Ans: - **

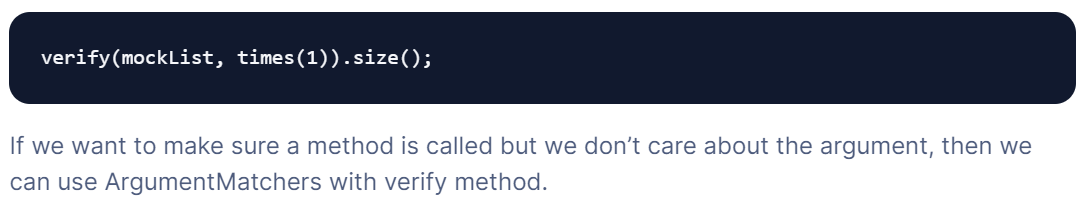
**Question 6th: - Why we used @responseentity annotation.**

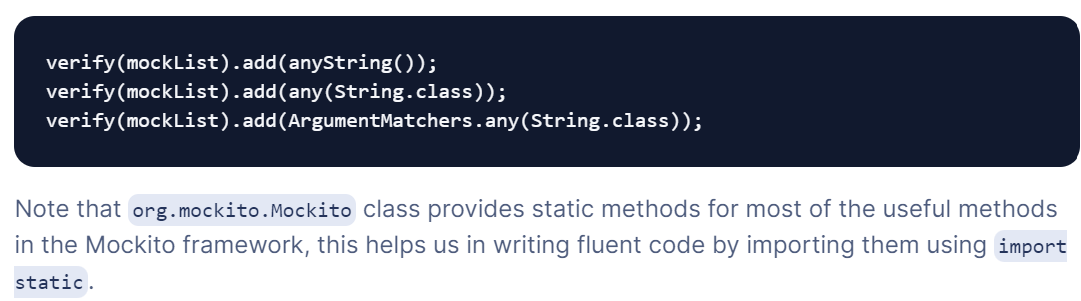
**Ans: -** to provide response error code.****

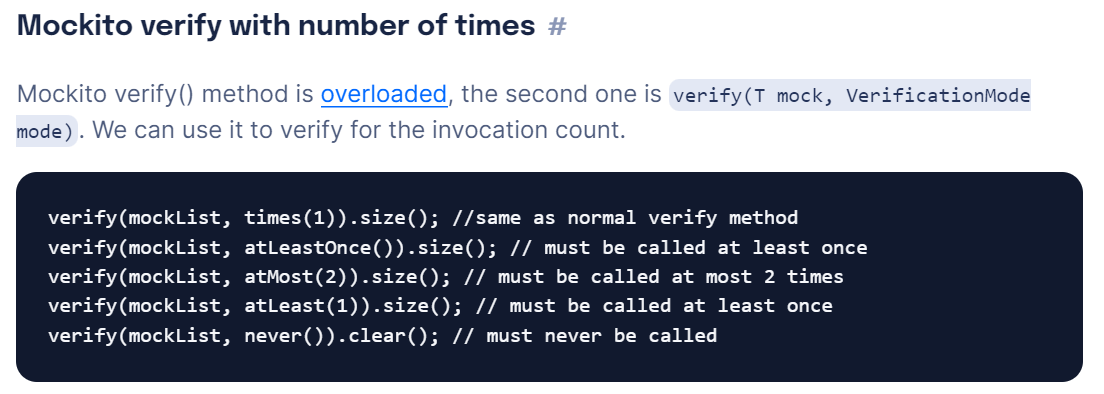
**Question 6th: - What is ‘verify’ in Junit.**

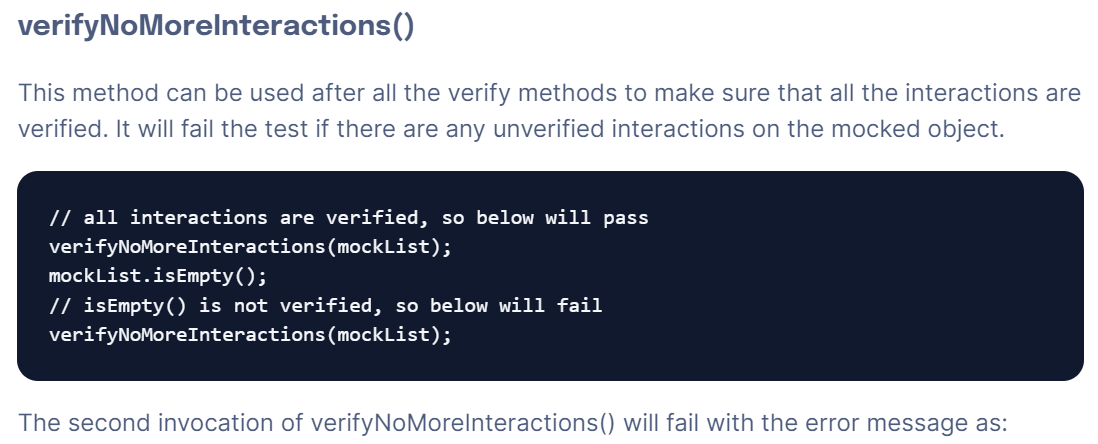
**Ans: -** generally varify method we know- for how many time after call we get DB resorce .

****

****

****

****

****

[**verifyZeroInteractions()**](https://www.digitalocean.com/community/tutorials/mockito-verify#verifyzerointeractions)

verifyZeroInteractions() method behavior is same as verifyNoMoreInteractions() method.

Map mockMap = mock(Map.class);

Set mockSet = mock(Set.class);

verify(mockList).isEmpty();

verifyZeroInteractions(mockList, mockMap, mockSet);

[**Mockito verify only method call**](https://www.digitalocean.com/community/tutorials/mockito-verify#mockito-verify-only-method-call)

If we want to verify that only one method is being called, then we can use only() with verify method.

Map mockMap = mock(Map.class);

mockMap.isEmpty();

verify(mockMap, only()).isEmpty();

[**Mockito Verify Order of Invocation**](https://www.digitalocean.com/community/tutorials/mockito-verify#mockito-verify-order-of-invocation)

We can use InOrder to verify the order of invocation. We can skip any method to verify, but the methods being verified must be invoked in the same order.

InOrder inOrder = inOrder(mockList, mockMap);

inOrder.verify(mockList).add("Pankaj");

inOrder.verify(mockList, calls(1)).size();

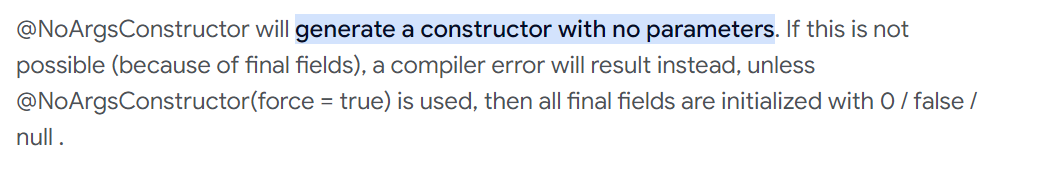
inOrder.verify(mockList).isEmpty();

inOrder.verify(mockMap).isEmpty();

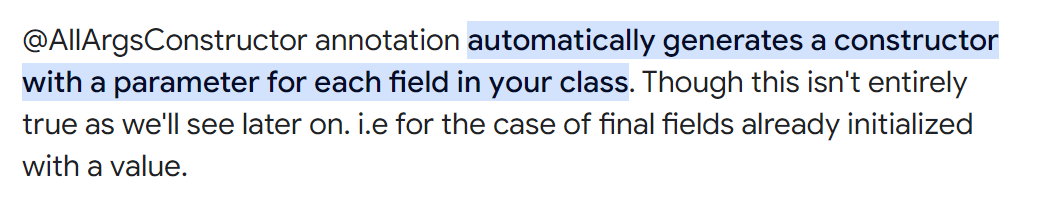
[**Summary**](https://www.digitalocean.com/community/tutorials/mockito-verify#summary)

Mockito verify() methods can be used to make sure the mock object methods are being called. If any method call is deleted by mistake, then verify method will throw an error.

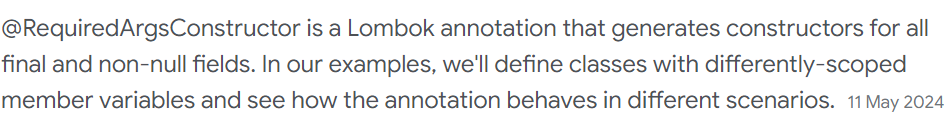
**Question 7th: - What is @NoArgsConstructor in springBoot.**

**Ans: - **

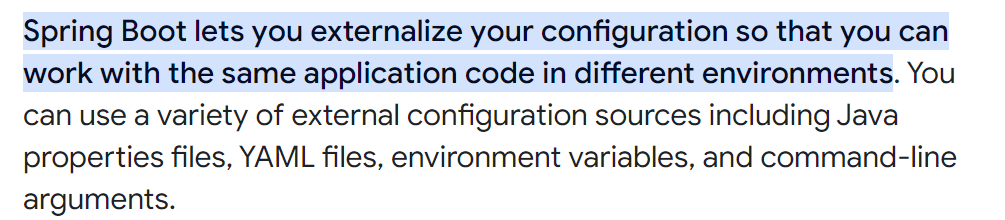
**Question 8th: - What is @AllArgsConstructor in springBoot.**

**Ans: - **

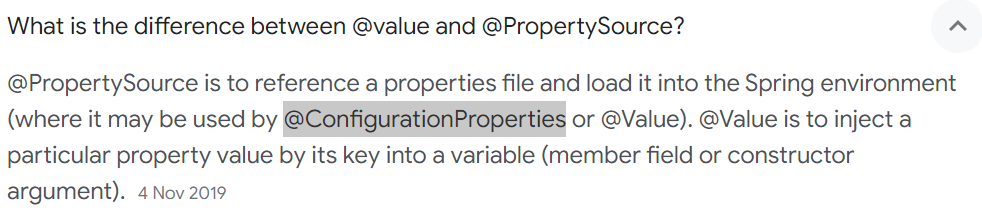
**AND What is @RequiredArgsConstructor in springBoot.?**

****

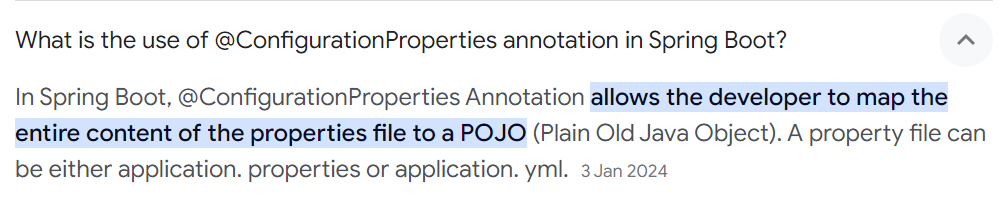
**Question 9th: - What is externalization in springBoot.**

**Ans: - **

**AND**

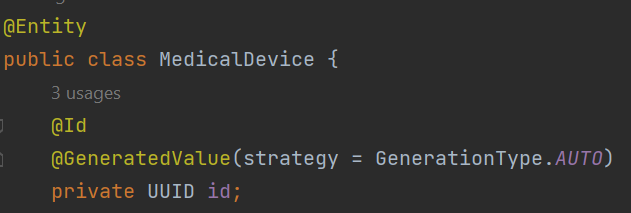
****

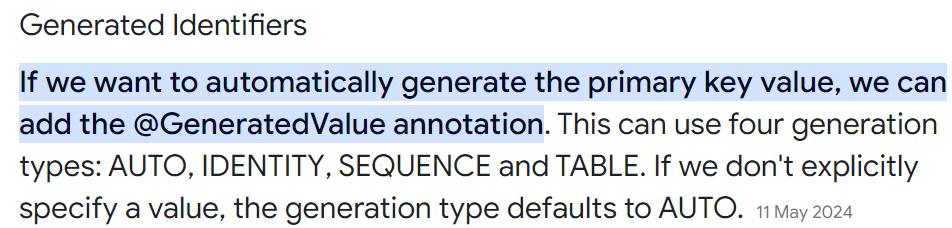
**AND**

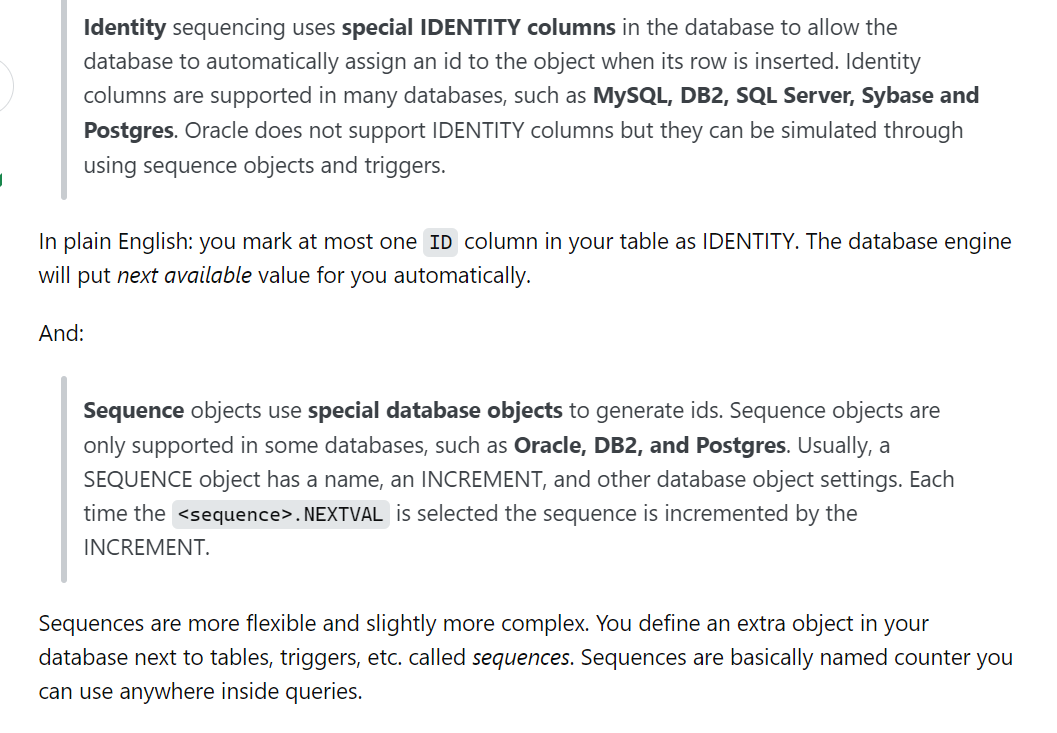
****

**Question 10th: - for Id Generation in Hibernate ,Explain GeneratedValue AUTO ,Sequence and Identity.**

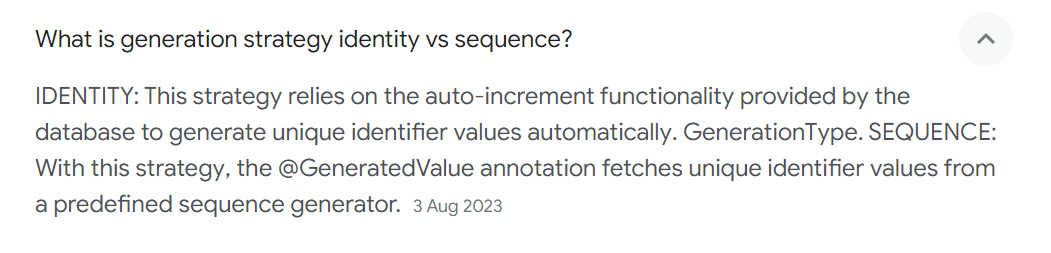
**Ans: -**

****

****

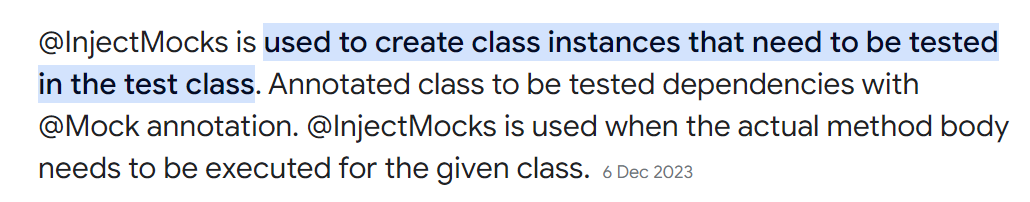
****

**OR**

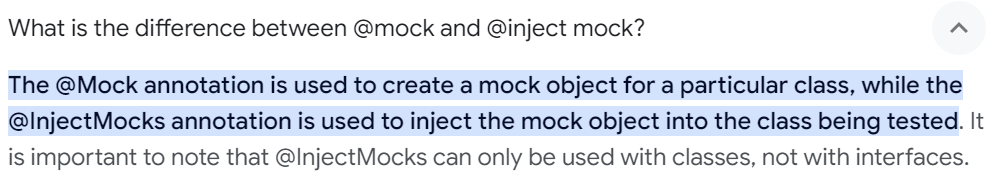
****

**Question 11th: - What is @InjectMocks annotation in Junit5.**

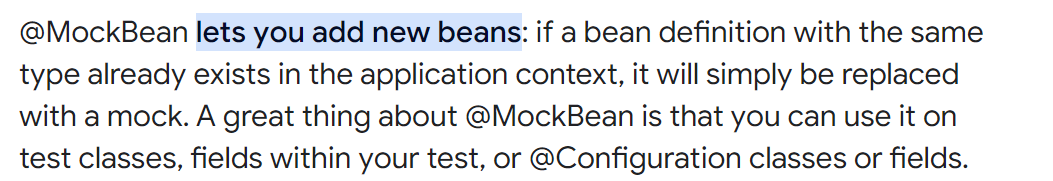
**Ans: -**

****

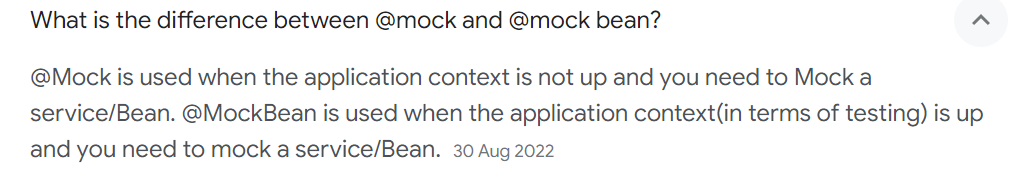
**OR**

****

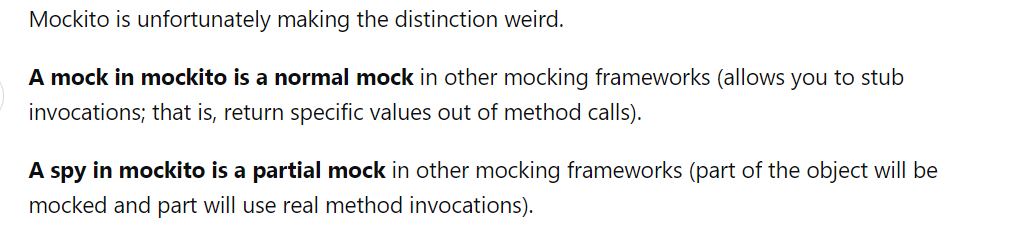
**Question 12th: - What is @MockBean annotation in Junit5.**

**Ans: - **

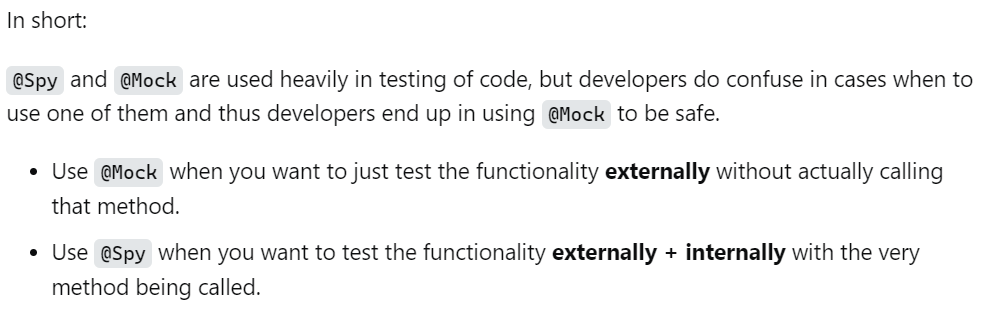
**OR**

****

**Question 13th: - What is @MockBean annotation in Junit5.**

**Ans: - **

**OR**



**Question 14th: - Write the Unit Test case and run ,the code was simple String concate ?**

**Ans: -**

