**Question: 1** Select one correct answer about spring bean life cycle.  
  
**Your Answer:** None  
  
**Correct Answer:** The method annotated with @PostConstruct is called before before the afterPropertiesSet callback method of the InitializingBean interface  
  
**Description:** 1. In the bean lifecycle, method annotated with @PostConstruct is called after the properties set step and the BeanPostProcessors#postProcessBeforeInitialization step   
2. Destroy methods of prototype beans are never called   
3. In the bean lifecycle, the afterPropertiesSet callback method of the InitializingBean is called after the method annotated with the @PostConstruct annotation and before the init-method declared in the XML configuration file.   
4. In the bean lifecycle, the method annotated with the @PreDestroy annotation is called before the destroy callback of the DisposableBean interface and before the destroy-method declared in the XML configuration file.

**Question: 2** What is/are typically case(s) where you usually need to manually instanciated an ApplicationContext?  
  
**Your Answer:** None  
  
**Correct Answer:** In a standalone application started with a main method  
  
**Description:** None

**Question: 3** Which one is not correct about the advantages for using Spring when writing integration tests?  
  
**Your Answer:** None  
  
**Correct Answer:** Create mock or stub  
  
**Description:** Mocking or stubbing is more frequent in unit tests than in integration tests. And Spring does not provide any implementation or abstraction of mock framework.

**Question: 4** What are the main advantage(s) for using Spring when writing unit tests?  
  
**Your Answer:** None  
  
**Correct Answer:** Provide some mocks for servlet classes  
  
**Description:** What are the main advantage(s) for using Spring when writing unit tests?   
1. You don´t need Spring container to writer unit test   
2. Refer to the answer number 1.   
3. The org.springframework.mock package provides mock classes like MockHttpSession or MockHttpContext. They could be helpful for unit test in the presentation layer and when you don´t use any mock framework such as Mockity or EasyMock.

**Question: 5** What are the features of the XML <context: namespace? Select correct option  
  
**Your Answer:** None  
  
**Correct Answer:** @Autowired annotation enabling  
  
**Description:** 1. Use <tx:annotation-driven /> to enable @Transactional annotation scanning   
2. Use <aop:aspectj-autoproxy /> to enable detection of @Aspect bean   
3. Turns on <context:annotation-config /> or <context:component-scan /> to enable @Autowiring annotation

**Question: 6** Select correct statement about transactional support of the spring test module.  
  
**Your Answer:** None  
  
**Correct Answer:** Transaction manager could be set within the @TransactionConfiguration annotation  
  
**Description:** None

**Question: 7** Select correct statement about developing integration test with Spring support.  
  
**Your Answer:** None  
  
**Correct Answer:** Spring context configuration could be inherits from the super class  
  
**Description:** 1. The Spring context is cached across tests unless you use @DirtiesContext annotation   
2. With the Spring test module, dependency injection is available in test case. So you may autowired the bean to test   
3. By default, a @ContextConfiguration annoted class inherits the spring context configuration file locations defined by an annotated superclass. The inheritLocations of this attribute allows to change this default behavior.   
4. If no context configuration file is provided to the @ContextConfiguration annotation, Spring use a file convention naming. It try to load a file named with the test class name and suffices by the "-context.xml" suffice (i.e. MyDaoTest-context.xml)

**Question: 8** What statement is not correct in live environment? Select a unique answer.  
  
**Your Answer:** None  
  
**Correct Answer:** All of the above  
  
**Description:** None

**Question: 9** What is right about the spring test module?  
  
**Your Answer:** None  
  
**Correct Answer:** None of the above  
  
**Description:** None

**Question: 10** Given the following configuration class, what are correct affirmations? Select one or more answers.   
public class ApplicationConfig {   
private DataSource dataSource;   
@Autowired   
public ApplicationConfig(DataSource dataSource) {   
this.dataSource = dataSource;   
}   
@Bean(name="clientRepository")   
ClientRepository jpaClientRepository() {   
return new JpaClientRepository();   
}   
}  
  
**Your Answer:** None  
  
**Correct Answer:** @Configuration annotation is missing  
  
**Description:** 1. In order to be taken into account by Spring, the ApplicationConfig class has to be annotated with the @Configuration annotation