Core Java Day-8

**Q.** Which of the given statements represents business layer in layered architecture?

**1)**. **Layer in which business methods are implemented**

**2)**. **Business layer implementation is hidden from user only interface is given to user**

**3)**. Business layer implementation is accessible to user

**4)**. Business layer contains implementation for accessing and getting connected with database.

**Q.** Which of the given statement is true about org.junit.runners.Suite class?

**1)**. **It runs group of test cases**

**2)**. It specifies runner class to run the annotated class

**3)**. both 0 and 1

**4)**. none of the above

**Q.** Which of the below annotation is used to test with different parameters?

**1)**. **@Parameterized.parameter**

**2)**. **@RunWith(Parameterized.class)**

**3)**. @Test(Parameterized.class)

**4)**. @Test.parameter

**Q.** @RunWith(Suite.class)   
@Suite.SuiteClasses({ TestPerson.class, TestPerson2.class,TestPersonFixture.class})   
public class TestPersonSuite {  
  
 @BeforeClass  
 public static void setUpBeforeClass() throws Exception {  
  System.out.println("Now running the Test Suite");  
 }  
 @AfterClass  
 public static void tearDownAfterClass() throws Exception {  
  System.out.println("The Test Suite is completed");  
 }  
  
}What is true about above code?

**1)**. **It will run three test classes in a group**

**2)**. **@BeforeClass causes  the method to be run once before any of the test methods in the class.**

**3)**. **Annotating a public static void method with @AfterClass causes that method to be run after all the tests in the class have been run.**

**4)**. @BeforeClass method will run only once at before testing starts.

**Q.** Choose the correct statements about parameterized test.

**1)**. **It allows to run the same test with different parameters**

**2)**. **It can be specified as given below  
@RunWith(Parameterized.class)**

**3)**. **To use it, we need to add static method which returns collection of data**

**4)**. It can be done with @RunWith(Suite.class)

**Q.** Which of the given options are mock frameworks?

**1)**. **DynaMock**

**2)**. **Jmock**

**3)**. **EasyMock**

**4)**. StaticMock

**Q.**Rima wants to validate user's credential before showing the account details back to user, in online banking application, in which of the given layer she can validate the user's credential?

**1)**. Presentation Layer

**2)**. **Business Layer**

**3)**. DataAccessLayer

**4)**. ServiceLayer

**Q.** Which of the given is the correct implementation of Test suites?

**1)**. import org.junit.runner.RunWith;   
import org.junit.runners.Suite;   
@RunWith   
@Suite.SuiteClasses({ TestCalAdd.class, TestCalSubtract.class,   
TestCalMultiply.class, TestCalDivide.class })   
public class CalSuite {   
  
} 

**2)**. **import org.junit.runner.RunWith;   
import org.junit.runners.Suite;   
@RunWith(Suite.class)   
@Suite.SuiteClasses({ TestCalAdd.class, TestCalSubtract.class,   
TestCalMultiply.class, TestCalDivide.class })   
public class CalSuite {   
  
}**

**3)**. import org.junit.runner.RunWith;   
import org.junit.runners.Suite;   
@Run(Suite.class)   
@Suite.SuiteClasses({ TestCalAdd.class, TestCalSubtract.class,   
TestCalMultiply.class, TestCalDivide.class })   
public class CalSuite {   
// the class remains completely empty,   
// being used only as a holder for the above annotations   
} 

**4)**. import org.junit.runner.RunWith;   
import org.junit.runners.Suite;   
  
@Suite.SuiteClasses({ TestCalAdd.class, TestCalSubtract.class,   
TestCalMultiply.class, TestCalDivide.class })   
public class CalSuite {   
// the class remains completely empty,   
// being used only as a holder for the above annotations   
}

**Q.** Which of the given statement is true about DTO?

**1)**. **It is used to encapsulate business data**

**2)**. It abstracts the logic to access underlying data store

**3)**. It is concerned about retrieval,management,processing of application data

**4)**. It is defined to accept user inputs

**Q.** Which of the given method is used to create EasyMock?

**1)**. EasyMock.createMock();

**2)**. **EasyMock.createMock(Classname.class);**

**3)**. EasyMock mock = new EasyMock();

**4)**. EasyMock mock = new EasyMock(Classname.class);