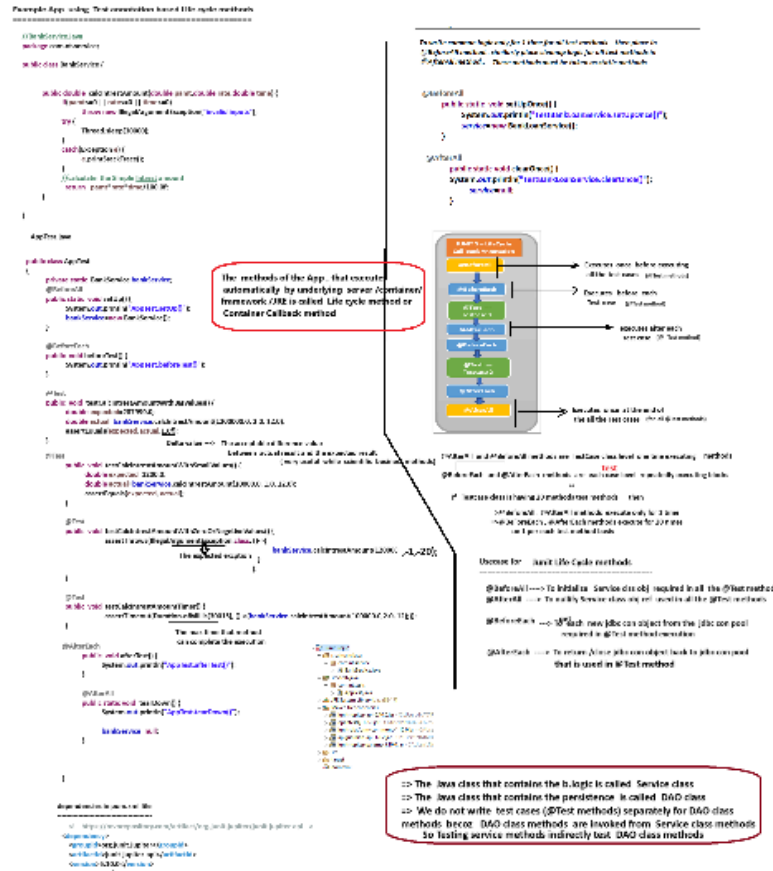

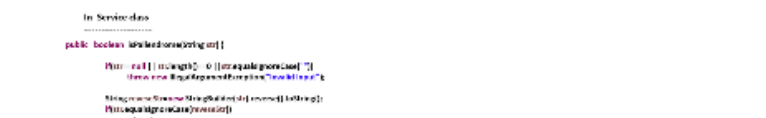

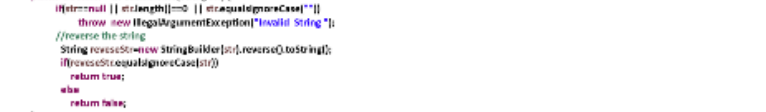

```
More example on assertXxx() methods
-----------------------------------
//Service class
-------------
package com.nt.service;

import java.util.Date;

import com.nt.model.Student;

public class MoreOperationsService {

    //B.method1
    public boolean   isPalindrome(String str) {
        if(str==null || str.length()==0  || str.equalsIgnoreCase(""))
            throw  new IllegalArgumentException("Invalid String ");
        //reverse the string
        String reveseStr=new StringBuilder(str).reverse().toString();
        if(reveseStr.equalsIgnoreCase(str))
          return true;
        else
          return false;
    }

    // B.method2
    public   Date     showDateByMonth(int month) {
        if(month>=1 && month<=12)
            return  new Date();
```

```
    public boolean  isPalindrome(String str) {

            if(str==null || str.length()==  0 || str.equalsIgnoreCase(""))
                throw new IllegalArgumentException("Invalid input");

            String reverseStr=new StringBuilder(str).reverse().toString();
            if(str.equalsIgnoreCase(reverseStr))
                return true;
            else
                return false;

        }


        In  Testcase class
        ------------------

    @Test
        public  void  testIsPalindromeWithValidInputs() {
            assertTrue(service.isPalindrome("madam"));
        }

        @Test
        public  void  testIsPalindromeWithInValidInputs() {
            assertFalse(service.isPalindrome("madam1"));
        }

        @Test
        public  void  testIsPalindromeWithNullInputs() {
            assertThrows(IllegalArgumentException.class,()->{service.isPalindrome("");});
        }
```

Assignment : Write Junit test cases for calculateCompoundInterestAmount(-,-,-) method

---

Example App using Test annotation based Life cycle methods

==========

```
//BankService.java
package com.nt.service;
public class BankService {
}
}
public double calcintrestAmount(double pamt,double rate,double time) {
if(pamt<=0 | | rate<=0 || time<=0)
try {
}
throw new IllegalArgumentException("invalid inputs");
Thread.sleep(30000);
catch(Exception e) {
}
e.printStackTrace();
//calculate the Simple Intrest amount
return pamt*rate*time/100.0f;
```

To write common logic only for 1 time for all test methods.. then place in @BeforeAll method.. similarly place cleanup logic for all test methods in @AfterAll method.. These methods must be taken as static methods

```
AppTest.java
public class AppTest
{
private static BankService bankService; @BeforeAll
public static void setUp() {
System.out.println("AppTest.setUp()");
bankService=new BankService();
```

The methods of the App, that executes automatically by underlying server /container/ framework/JRE is called Life cycle method or Container Callback method

```
}
@BeforeEach
public void beforeTest() {
System.out.println("AppTest.beforeTest()");
}
@Test
public void testCalcintrestAmountWithBigValues() {
double expected=287999.0;
```

```
double actual-bankService.calcintrestAmount (1200000.0, 2.0, 12.0); assertEquals(expected, actual, 1.0f);
}
@Test
@BeforeAll
public static void setUpOnce() {
System.out.println("TestBankLoanService.setUpOnce()");
service=new BankLoanService();
}
@AfterAll
public static void clearOnce() {
System.out.println("TestBankLoanService.clearOnce()");
service=null;
Delta value ---> The acceptable difference value
between actual result and the expected result
(very useful while scientific busines methods)
public void testCalcIntrestAmountWithSmallValues() {
double expected=1200.0;
double actual bankService.calcintrestAmount(10000.0, 1.0, 12.0); assertEquals(expected, actual);
}
}
JUNIT 5-LifeCycle
Call-back Annotation
@BeforeAll
@BeforeEach
@Test
Testcase 1
@AfterEach
@BeforeEach
@Test --- Testcase 2
Executes once before executing
all the test cases (@Test methods)
Executes before each
Test case
(@Test method)
executes after each
test case (@ Test method)
@AfterEach
```

@AfterAll

Executes once at the end of

the all the Test cases (for all @Test methods)

@AfterAll and @BeforeAll methods are TestCase class level one tine executing methods test @BeforeEach and @AfterEach methods are each case level repeatedly executing blocks

as

if Testcase class is having 20 methods test methods

then

=>@BeforeAll, @AfterAll methods execute only for 1 time =>@BeForeEach, @AfterEach methods execute for 20 times on 1 per each test method basis

```
@Test
public void testCalcIntrestAmountWithZeroOrNegetiveValues() {
The expected excption } );
assertThrows(IllegalArgumentException.class, ()->{
bankService.calcintrestAmount (120000,-1,-20);
Usecase for
Junit Life Cycle methods
}
@Test
public void testCalcintrestAmount Timer() {
assertTimeout(Duration.ofMillis(30015), ()->{bankService.calcIntrestAmount (100000.0,2.0, 12);});
}
```

The max time that method

```
@AfterEach
can complete the execution
public void afterTest() {
System.out.println("AppTest.afterTest()");
}
@AfterAll
public static void tearDown() {
System.out.println("AppTest.tearDown()");
bankService=null;
}
```

com.nt.service

>

✓

JunitProj02

src/main/java

> BankService.java

#src/test/java

com.nt.tests

> AppTest.java

JRE System Library [JavaSE-17]

Maven Dependencies

>junit-jupiter-api-5.10.0.jar - C:\Users\NATAR >opentest4j-1.3.0.jar - C:\Users\NATARAJ\.m?

> junit-platform-commons-1.10.0.jar - C:\Use >apiguardian-api-1.1.2.jar - C:\Users\NATARA
>junit-jupiter-params-5.10.0.jar - C:\Users\N/

>

src

› target

@BeforeAll ----> To initialize Service clss obj required in all the @Test methods @AfterAll ----> To nullify Service class obj ref used in all the @Test methods

--->

@BeforeEach Tgetach new jdbc con object from the jdbc con pool required in @Test method execution

@AfterEach----> To return/close jdbc con object back to jdbc con pool that is used in @Test method

pom.xml

}

dependencies in pom.xml file

<!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api --> <dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-api</artifactId>

<version>5.10.0</version>

=> The Java class that contains the b.logic is called Service class

=> The Java class that contains the persistence is called DAO class

=> We do not write test cases (@Test methods) separately for DAO class

methods becoz DAO class methods are invoked from Service class methods So Testing service methods indirectly test DAO class methods

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-params -->

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-params</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

How to test private method of the service class?

ans) we can not test private methods of service calss directly.. So call

them from public methods of the service class and write junit test cases on the public methods to perform the unit testing

Examples on assertTrue() and assertFalse() methods

======

=========

==========

In Service class

public boolean isPallendrome(String str) {

if(str==null || str.length()==0 || str.equalsIgnoreCase("")) throw new IllegalArgumentException("Invalid Input");

String reveseStr=new StringBuilder(str).reverse().toString(); if(str.equalsIgnoreCase(reveseStr))

}

return true;

else

return false;

In Testcase class

============

@Test

public void testIsPallendromeWithValidInputs() { assertTrue(service.isPallendrome("madam"));

}

@Test

public void testIs PallendromeWithInValidInputs() {

assertFalse(service.isPallendrome("madam1"));

}

@Test

public void testIsPallendromeWithNoInputs() {

assertThrows (IllegalArgumentException.class,()->{ service.isPallendrome(""); });

}

Assignment :: Write Junit test cases for calculateCompoundIntrestAmount(-,-,-) method

More example on assertXxx() methods

========

//Service class

==========

package com.nt.service;

```
import java.util.Date;
import com.nt.model.Student;
public class More OperationsService {
//=B.method1
=======
public boolean isPallendrome(String str) {
if(str==null || str.length()==0 || str.equalsIgnoreCase("")) throw new IllegalArgumentException("Invalid String "); //reverse the string
String reveseStr=new StringBuilder(str).reverse().toString(); if(reveseStr.equalsIgnoreCase(str))
return true;
else
return false;
}
}
// B.method2
public Date showDateByMonth(int month) {
if(month>=1 && month<=12)
else
}
return new Date();
return null;
// Test case Class
//MoreOpertionsService Test
package com.nt.service;
import static org.junit.jupiter.api.Assertions.assertFalse; import static org.junit.jupiter.api.Assertions.assertNotNull; import static org.junit.jupiter.api.Assertions.assertNull; import static org.junit.jupiter.api.Assertions.assertTrue; import static org.junit.jupiter.api.Assertions.assertSame; import static org.junit.jupiter.api.Assertions.assertNotSame;
import java.time.LocalTime;
import java.util.Date;
import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.Test;
public class More OperationsService Test {
private static MoreOperationsService service;
@BeforeAll
public static void setUpOnce() {
System.out.println("MoreOperationsServiceTest.setUpOnce()");
```

```
service=new More OperationsService();
}
}
@Test
public void testis PallendromeWithValidData() {
String msg="madam";
assertTrue(service.isPallendrome(msg));
}
@Test
public void testis PallendromeWithInValidData() {
String msg="nit";
assertFalse(service.isPallendrome(msg));
}
@Test
public void testShowDate ByValidMonth() {
Date actual-service.showDateByMonth(1);
assertNotNull(actual);
}
@Test
public void testShowDateByInValidMonth() {
Date actual-service.showDateByMonth(-1);
assertNull(actual);
}
@Test
public void testSingleton Runtime() {
Runtime time1=Runtime.getRuntime();
Runtime time2=Runtime.getRuntime();
assertSame(time1,time2);
}
@Test
public void testSingletonLocalTime() {
LocalTime time1=LocalTime.now();
LocalTime time2=LocalTime.of(10, 20);
assertNotSame(time1,time2);
}
@AfterAll
public static void tearDownOnce() {
```

```
System.out.println("MoreOperationsServiceTest.tearDownOnce()");
service=null;
}
```