give

We can "Displayname for test methods and TestCase class using @DisplayName(-) annotation

```
@DisplayName("Testing the BankService ")
public class AppTest
{
private static BankService bankService;
@BeforeAll
public static void setUp() {
System.out.println("AppTest.setUp()");
bankService=new BankService();
}
@BeforeEach
public void beforeTest() {
}
@Test
System.out.println("AppTest.before Test()");
@DisplayName("testWithBigValues")
public void testCalcIntrestAmountWithBigValues() {
double expected=287999.0;
double actual-bankService.calcIntrestAmount(1200000.0, 2.0, 12.0); assertEquals(expected, actual, 1.0f);
@Test
@DisplayName("testWithSmallValues")
public void testCalcIntrestAmountWithSmallValues() {
}
@Test
double expected=1200.0;
double actual-bankService.calcintrestAmount (10000.0, 1.0, 12.0); assertEquals(expected, actual);
@DisplayName("testWithZeros")
public void testCalcIntrestAmountWithZeroOrNegetiveValues() {
assertThrows(IllegalArgumentException.class, ()->{
}
@Test
}
);
bankService.calcintrestAmount(120000.0, 0.0,12.0);
//@Disabled
@DisplayName("testWithTimer")
public void testCalcIntrestAmount Timer() {
assertTimeout(Duration.ofMillis(30015), ()->{bankService.calcIntrest Amount (100000.0,2.0, 12);});
}
@AfterEach
public void afterTest() {
System.out.println("AppTest.afterTest()");
}
@AfterAll
public static void tearDown() {
System.out.println("AppTest.tearDown()");
bankService=null;
}
}
```

```
Finished after 0.125 seconds
Runs: 4/4
* Errors: 0
Testing the BankService [Runner: JUnit 5] (0.011 s)
testWithSmallValues (0.004 s)
testWithZeros (0.002 s)
testWith Timer (0.004 s)
testWithBigValues (0.001 s)
Failures: 0
Failure Trace
```

Different Project Development and Releasing environments

"dev"

coding

software company

"test"

"uat"

Q & A

release

Q&A

Client orgnization

"prod" Project Live

if we place test methods in test case class for different env of project development to production using @Tag annotation then run the test methods by specifying the env.. name as the tag name

In Test case class

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

@Test

@DisplayName("testWithBigValues")

@Tag("test")

public void testCalcIntrestAmountWithBigValues() {

}

@Test

double expected=287999.0;

of

note:: At the time project "dev","test" env the integration with third party services like UPI Payment

card payment and etc.. may not be ready, so we can not test those integrations here means in "dev","test" envs.. Mostly they will be ready during the release or after the release of the Project, So we can write test cases representing third party services in the UAT env.. So to tag different test cases for different env.. take the support

double actual bankService.calcIntrestAmount(1200000.0, 2.0, 12.0); of @Tag annotation
assertEquals(expected, actual, 1.0f);

@DisplayName("testWithSmallValues")

@Tag("test")

public void testCalcIntrestAmountWithSmallValues() {

}

double expected=1200.0;

double actual-bankService.calcintrestAmount(10000.0, 1.0, 12.0); assertEquals(expected, actual);

To specify the tag/env.. name while running the test cases

run configurations

Right click on Test Case class ---> run as -->

In "dev","test" env.. we will not get real data for testing.. but we get that real data at env.. So better tag few test cases as "dev","test" env test cases and few other test cases as "uat" env test cases

note:: if the documentation of the ant annotation is having @Repetable then that annotation can be used for multiple times at same levelgava code

-----> include /exclude tag----> configure ---->

✓ Include Tags

Newline separated tags to be included in the test run: test

Exclude Tags

--> apply ---> run

The multiple test methods of test case class can be executed by specifying the order of execution

using different params

=> Through @Order(-) priority value

=> Through Method name

=> Through Display name

=> Through Randomly (default)

@Order(-) contains the priroty value (numeric value)

--> High value indicates low priority

---> Low value indicates high priority

Example

Decides the test methods order based on the method names

```
@DisplayName("Testing the BankService ")
@TestMethodOrder(MethodName.class)
//@TestMethodOrder(org.junit.jupiter.api.MethodOrderer.OrderAnnotation.class) //best test methods order is
decided by Order(-) priority order
//@TestMethodOrder(org.junit.jupiter.api.Method Orderer.DisplayName.class)
//@TestMethodOrder(org.junit.jupiter.api.Method Orderer.Random.class) //default
order is decided by display name
random order will be chosen
public class AppTest
{
private static BankService bankService;
@BeforeAll
public static void setUp() {
System.out.println("AppTest.setUp()");
}
bankService=new BankService();
@BeforeEach
public void beforeTest() {
System.out.println("AppTest.beforeTest()");
}
@Test
@DisplayName("testWithBigValues")
//@Tag("test")
@Order(1)
public void testCalcIntrestAmountWithBigValues() {
double expected=287999.0;
double actual bankService.calcintrestAmount (1200000.0, 2.0, 12.0);
assertEquals(expected, actual, 1.0f);
}
@Test
@DisplayName("testWithSmallValues")
```

```
@Order(10)
//@Tag("test")
public void testCalcIntrestAmountWithSmallValues() {
double expected=1200.0;
double actual bankService.calcintrestAmount(10000.0, 1.0, 12.0); assertEquals(expected, actual);
}
@Test
@DisplayName("testWithZeros")
//@Tag("uat")
//@Tag("uat")
//@Tag("test")
@Order(15)
public void testCalcIntrestAmountWithZeroOrNegetiveValues() {
assertThrows(IllegalArgumentException.class, ()->{
bankService.calcintrestAmount(120000.0, 0.0,12.0);
}
);
}
@Test
//@Disabled
@DisplayName("testWithTimer")
//@Tag("uat")
@Order(7)
public void testCalcIntrestAmountWithTimer() {
assertTimeout(Duration.ofMillis(30015), ()->{bankService.calcIntrestAmount (100000.0,2.0, 12);});
}
@AfterEach
public void afterTest() {
@AfterEach
public void afterTest() {
}
System.out.println("AppTest.afterTest()");
}
@AfterAll
public static void tearDown() {
System.out.println("AppTest.tearDown()");
bankService=null;
```

}

=>In Java Project, we can take any no.of test case classes with any no.of test methods

to test any no.of methods belonging to any no.of service classes

LogicalOperationsService.java

package com.nt.service;

import java.util.Date;

public class Logical OperationService {

note:: It is always recomanded to take 1 Test case class for 1 Service class

public boolean isPositive(int x) {

if(x<0)

return false;

else

return true;

}

public boolean isNull(Date d) {

if(d==null)

return true;

else

return false;

}

}

More methods in service class for working with assertXxx() methods

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

=====

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

public LocalDate showSysDate(LocalDate ldt) {

// get week day number

int weekNo=ldt.getDayOfWeek().getValue();

System.out.println("weekday number:"+weekNo); if(weekNo==6 || weekNo==7)

return null;

else

return ldt;

}

AppTest1.java

public class AppTest1 {

private static LogicalOperationService service;

@BeforeAll

```
public static void setUp() {
service=new Logical OperationService();
}
@Test
public void testIsPositive With PositiveNumber() { assertTrue(service.isPositive(10));
}
@Test
public void testIsPositiveWithNegetiveNumber() {
assertFalse(service.isPositive(-10));
}
@Test
public void testIsNullWithObject() {
assertFalse(service.isNull(new Date()));
}
@Test
public void testIsNullWith Reference() {
Date d=null;
assertTrue(service.isNull(d));
}
@AfterAll
public static void clerDown() {
service=null;
}
}
```

More @Test methods in Test Case class

```
@Test
public void testShowSysDateOnWeekDays() { LocalDate ldt=LocalDate.now();
LocalDate ldt1=service.showSysDate(ldt); assertNotNull(ldt1);
}
@Test
public void testShowSysDateOnWeekEndDays() {
LocalDate ldt=LocalDate.now(); ldt=ldt.plusDays(5);
LocalDate ldt1=service.showSysDate(ldt); assertNull(ldt1);
```

=> we can link multiple test cases classes with single TestSuite class to execute the

test methods of all the Test case classes togather (TestSuite is a class where we can combine multiple test case

step1)

classes into single unit for execution)

make sure that following additional jar file is added to pom.xml dependencies

```
<!-- https://mvnrepository.com/artifact/org.junit.platform/junit-platform-suite-engine -->
<dependency>
<groupId>org.junit.platform</groupId>
<artifactId>junit-platform-suite-engine</artifactId>
<version>1.10.0</version>
</dependency>
```

step Develop the TestSuite class as shown below in the same package "com.nt.tests" package where the Test case classes are present

```
//AllTestSuite.java
package com.nt.tests;
import org.junit.platform.suite.api.SelectClasses;
import org.junit.platform.suite.api.SelectPackages;
import org.junit.platform.suite.api.Suite;
@SelectPackages("com.nt.tests") | package name
@SelectClasses({AppTest.class, AppTest1.class}) | Test case class name
@Suite
public class AllTestsSuite {
✓ JunitProj02
#src/main/java
}
```

step3) Run The Tests using junit

✓

right click on AllTestSuite class

--> run as --> Junit Test

FAQS on Junit

==========

What is Unit Testing?

What is Peer Testing?

List out various unit testing tools?

What is the latest version and name of junit?

What is the difference test error and test failure?

Explain different assertXxx() methods ?

What is static import in java and explain it?

Explain different annotations of Junit?

How

to display custom names for the @Test methods ?

>com.nt.service (target class)

✓ src/test/java

com.nt.tests

>AllTestsSuite.java (test suite class)

› App Test.java

testcases class

> JRE System Library [JavaSE-17]

Maven Dependencies >junit-jupiter-api-5.10.0.jar - C:\Users\NATARAJ\.m2\repository\orc > opentest4j-1.3.0.jar - C:\Users\NATARAJ\.m2\repository\org\open > junit-platform-commons-1.10.0.jar - C:\Users\NATARAJ\.m2\repo >apiguardian-api-1.1.2.jar - C:\Users\NATARAJ\.m2\repository\org\ >junit-jupiter-params-5.10.0.jar - C:\Users\NATARAJ\.m2\repository >junit-platform-suite-engine-1.10.0.jar - C:\Users\NATARAJ\.m2\re >junit-platform-engine-1.10.0.jar - C:\Users\NATARAJ\.m2\repositc > junit-platform-suite-api-1.10.0.jar - C:\Users\NATARAJ\.m2\reposi junit-platform-suite-commons-1.10.0.jar - C:\Users\NATARAJ\.mz

>

> junit-platform-launcher-1.10.0.jar - C:\Users\NATARAJ\.m2\reposi

>

src

>

target

pom.xml

How to check whether service method is throwing expected exception or not?

How to check whether service method is completing executing in the given time period?

What is TestCase class?

What is TestSuite class?

What is the purpose of tagging the @Test methods using @Tag annotation?

How can we decide @Test methods execution order in Test case class?

How can we decide Test case classes execution order when they are linked with TestSuite class?

What are testings that will be done by Developer?

What are testings that will be done by Tester or QA Team?

How can we control the execution order of Test case classes that are linked with TestSuite class ?

Ans)

We can @TestClassOrder annotation on top of TestSuite class to specify the

Execution order .we can control the order by using @Order value or

Example

=======

@Suite

class name or

Display name or Random Order

```
@SelectPackages("com.nt.tests")
@SelectClasses({ BankOperationServiceTest.class, MoreOperationsServiceTest.class })
//@TestClassOrder(ClassOrderer.DisplayName.class)
//@TestClassOrder(ClassOrderer.OrderAnnotation.class) //For this we need to place @Order annotation
//@TestClassOrder(ClassOrderer.ClassName.class)
@TestClassOrder(ClassOrderer.Random.class) (default)
public class AllTests {
}
```

on the top of the TestCase classes