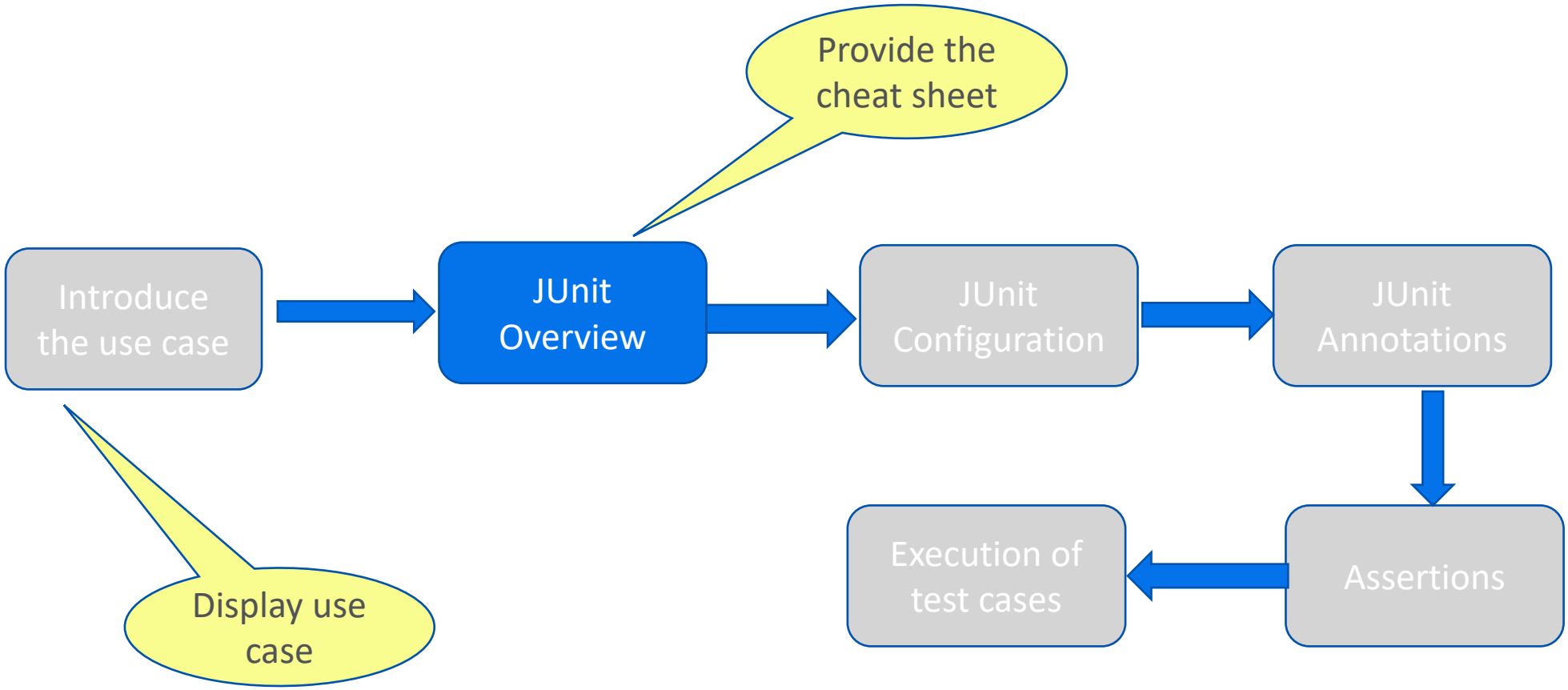
axess academy

Junit

standard chartered

axess academy

Module design



S/W Requirements
IDE :STS/ Intellij Jdk : 17

axess academy

Introduce the Use Case on JUnit

axess academy

Presenting the Use case on JUnit

- Write the JUnit test cases for Bank Customer Management System.
- Write the test case to make sure the Customer class object gets created successfully.
- Write the test cases to validate customer name, phone number, email id and account type.

axess academy

Junit Overview

axess academy

Junit Overview

- **JUnit** is a unit testing framework for the Java programming language.
- JUnit has been important in the development of test-driven development, and is one of a family of unit testing frameworks.
- JUnit 5 aims to adapt java 8 style of coding and several other features like lambda expression, that’s why java 8 is required to create and execute tests in JUnit 5.
- JUnit latest version is 5.4.2 released on April 2019.

JUnit

axess academy

JUnit Configuration and Test Case execution

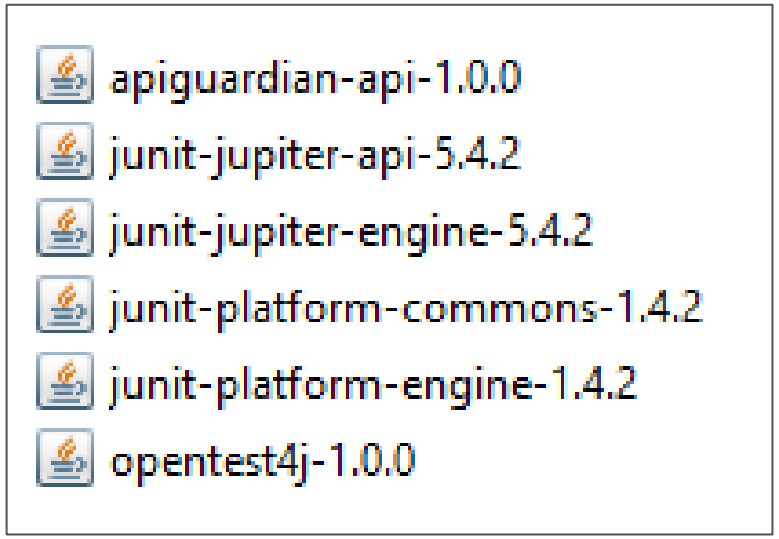
axess academy

How to configure JUnit

- Add these dependencies to POM file.

```
<dependencies>
  <dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-api</artifactId>
    <version>5.4.2</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-engine</artifactId>
    <version>5.4.2</version>
    <scope>test</scope>
  </dependency>
</dependencies>
```

- Add the required JUnit jar files in eclipse build path.



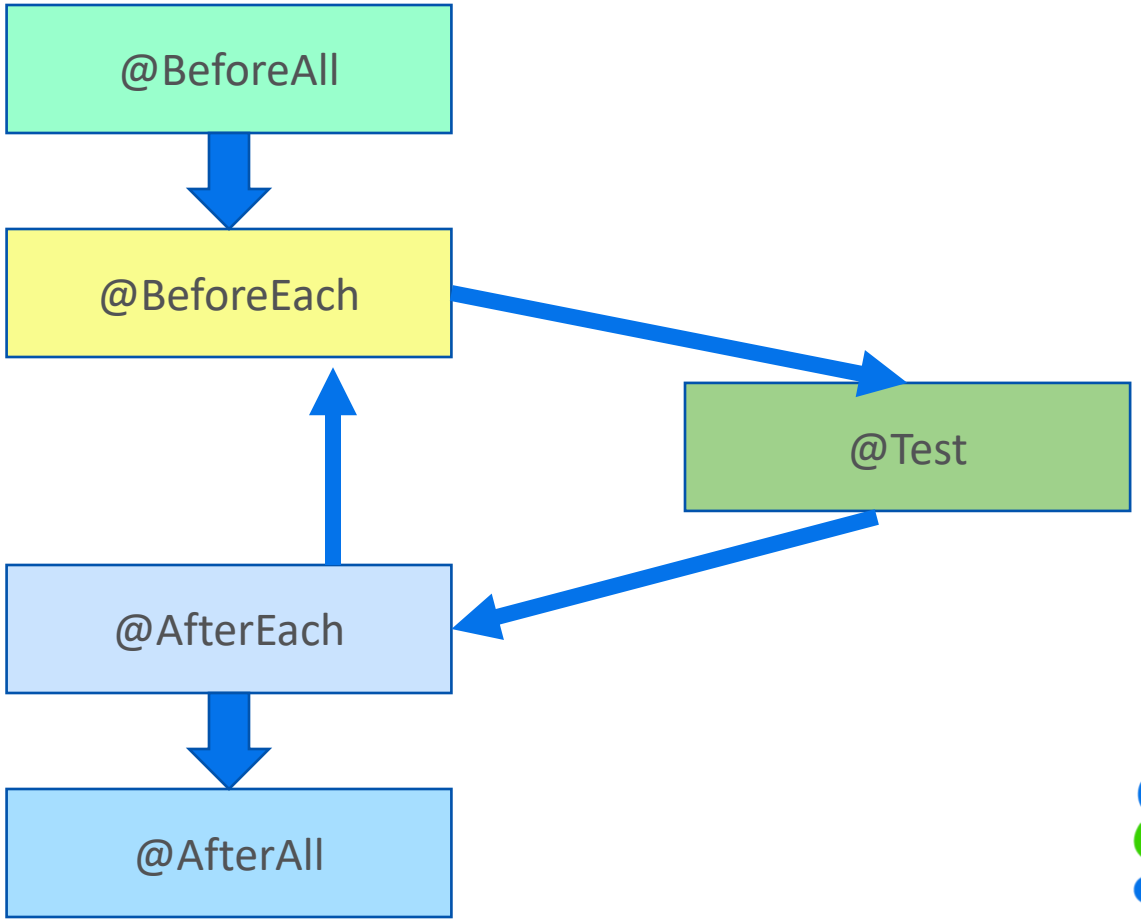
axess academy

Annotations

axess academy

Test LifeCycle

- In JUnit 5, test lifecycle is driven by 4 primary annotations i.e. @BeforeAll, @BeforeEach, @AfterEach and @AfterAll. Along with it, each test method must be marked with @Test annotation.



axess academy

Test LifeCycle Demo

```java
public class AppTest {
    @BeforeAll
    static void setup() {
        System.out.println("@BeforeAll executed");}
    @BeforeEach
    void setupThis() {
        System.out.println("@BeforeEach executed"); }
    @Test
    void testOne() {
        System.out.println("======TEST ONE EXECUTED=======");
        Assertions.assertEquals(4, 4);  }
    @Test
    void testTwo() {
        System.out.println("======TEST TWO EXECUTED=======");
        Assertions.assertEquals(6, 6);  }
    @AfterEach
    void tearThis() {
        System.out.println("@AfterEach executed");  }
    @AfterAll
    static void tear() {
        System.out.println("@AfterAll executed");   }
}
```

```
@BeforeAll executed
@BeforeEach executed
======TEST ONE EXECUTED=======
@AfterEach executed
@BeforeEach executed
======TEST TWO EXECUTED=======
@AfterEach executed
@AfterAll executed
```

axess academy

@Test

- @Test: Denotes that a method is a test method.

```
@Test
void testSum() {
    Calc calc= new Calc();
    int expected=20;
    int actual= calc.sum(10,10);
    assertEquals(expected,actual);
}
```

- @RepeatedTest: It enables to write repeatable test templates which could be run multiple times. The frequency can be configured as parameter to @RepeatedTest annotation.

```
@RepeatedTest(4)
void testAddition() {
    Calc calc= new Calc();
    int expected=20;
    int actual= calc.sum(10,10);
    assertEquals(expected,actual);
}
```

- .

axess academy

@BeforeEach and @BeforeAll

- @BeforeEach: Denotes that the annotated method should be executed *before* **each** @Test, @RepeatedTest, @ParameterizedTest,.
- @BeforeAll: Denotes that the annotated method should be executed *before* **all** @Test, @RepeatedTest, @ParameterizedTest,.

```
@BeforeAll
static void initAll() {
 System.out.println("before all test cases");
}

@BeforeEach
void init() {
    System.out.println("before each test cases");
}
```

axess academy

@AfterEach and @AfterAll

- @AfterEach: Denotes that the annotated method should be executed *after* **each** @Test, @RepeatedTest, @ParameterizedTest,.
- @AfterAll: Denotes that the annotated method should be executed *after* **all** @Test, @RepeatedTest, @ParameterizedTest,.

```
@AfterEach
void tearDown() {
    System.out.println("after each test cases");
}

@AfterAll
static void tearDownAll() {
    System.out.println("after all test cases");
}
```

axess academy

@ParametrizedTest

- @ParametrizedTest : Denotes that a method is a parameterized test. This makes it possible to run a test multiple times with different arguments.
- In addition, you must declare at least one source that will provide the arguments for each invocation and then consume the arguments in the test method.

- @ValueSource is one of the simplest possible sources. It lets you specify a single array of literal values and can only be used for providing a single argument per parameterized test invocation.
- The following types of literal values are supported by @ValueSource.
  - short
  - byte
  - int
  - long
  - float
  - double
  - char
  - java.lang.String
  - java.lang.Class

```
@ParameterizedTest
@ValueSource(strings = {"welcome", "", "bank" })
void palindromes(String input) {
    assertTrue(StringUtils.isNotBlank(input));
}
```

axess academy

Assertions

axess academy

Assertions

- Assertions help in validating the expected output with actual output of a testcase. All JUnit Jupiter assertions are static methods in the org.junit.jupiter.api.Assertions class.

- Few important methods are below:
  1. **void assertEquals(String expected, String actual):** Checks that two primitives/objects are equal.
  2. **void assertTrue(boolean condition):** Checks that a condition is true.
  3. **void assertFalse(boolean condition):** Checks that a condition is false.
  4. **void assertNotNull(Object object):** Checks that an object isn't null.
  5. **void assertNull(Object object):** Checks that an object is null.
  6. **void assertSame(object1, object2):** The assertSame() method tests if two object references point to the same object.
  7. **void assertNotSame(object1, object2):** The assertNotSame() method tests if two object references do not point to the same object.
  8. **void assertArrayEquals(expectedArray, resultArray);:**The assertArrayEquals() method will test whether two arrays are equal to each other.

axess academy

Calculator Test Cases

```
class Calc{
    public int sum(int a, int b) {
        return a+b;
    }
    public int sub(int a, int b) {
        return a-b;
    }
    public int multiply(int a, int b) {
        return a*b;
    }
    public int div(int a, int b) {
        int c=0;
        if(b!=0)
            c=a/b;
        return c;
    }
}
```

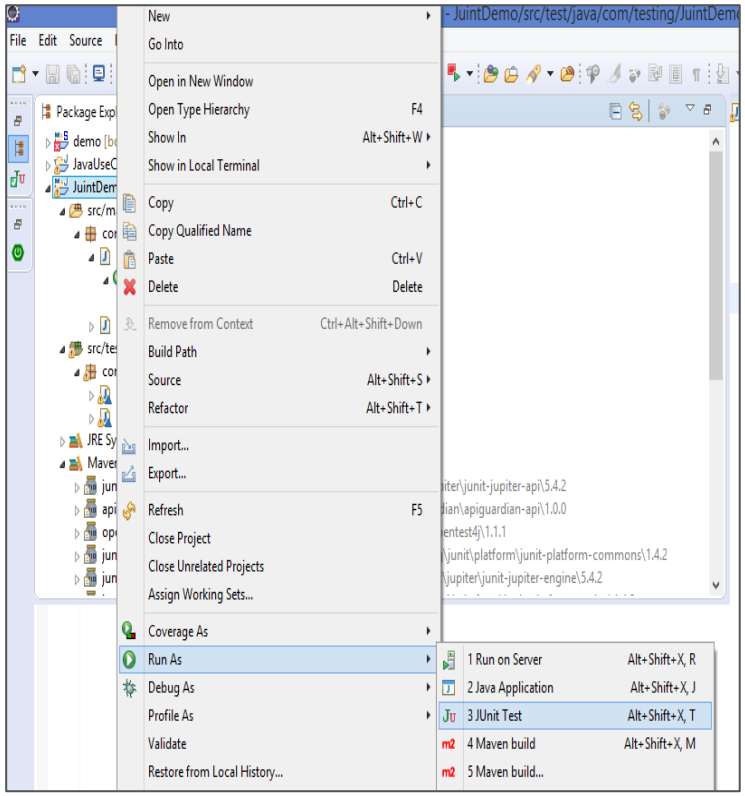
```
@Test
void testAddition() {
    Calc calc= new Calc();
    int expected=20;
    int actual= calc.sum(10,10);
    assertEquals(expected,actual);}
@Test
void testSubstraction() {
    Calc calc= new Calc();
    int expected=0;
    int actual= calc.sub(10,10);
    assertEquals(expected,actual);}
@Test
void testMultiply() {
    Calc calc= new Calc();
    int expected=100;
    int actual= calc.multiply(10,10);
    assertEquals(expected,actual);}
@Test
void testDivision() {
    Calc calc= new Calc();
    int expected=10;
    int actual= calc.div(100,10);
    assertEquals(expected,actual);}
```

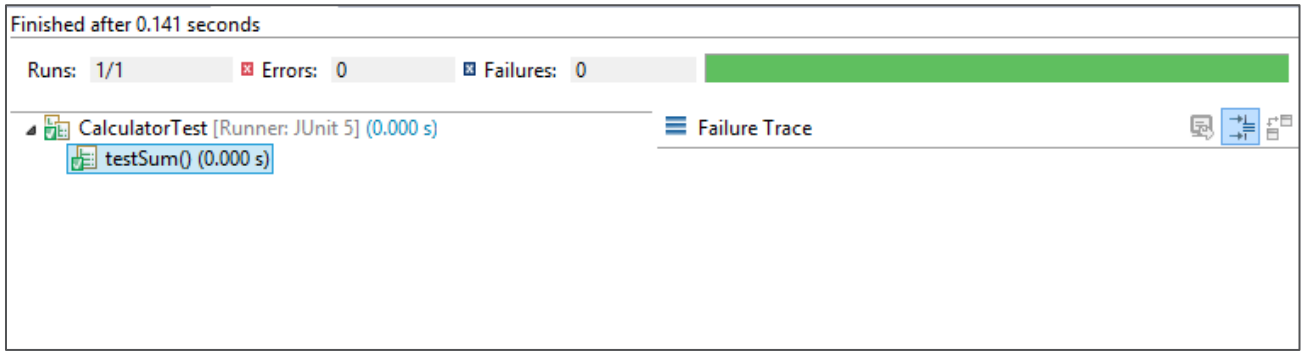
axess academy

Test Cases execution

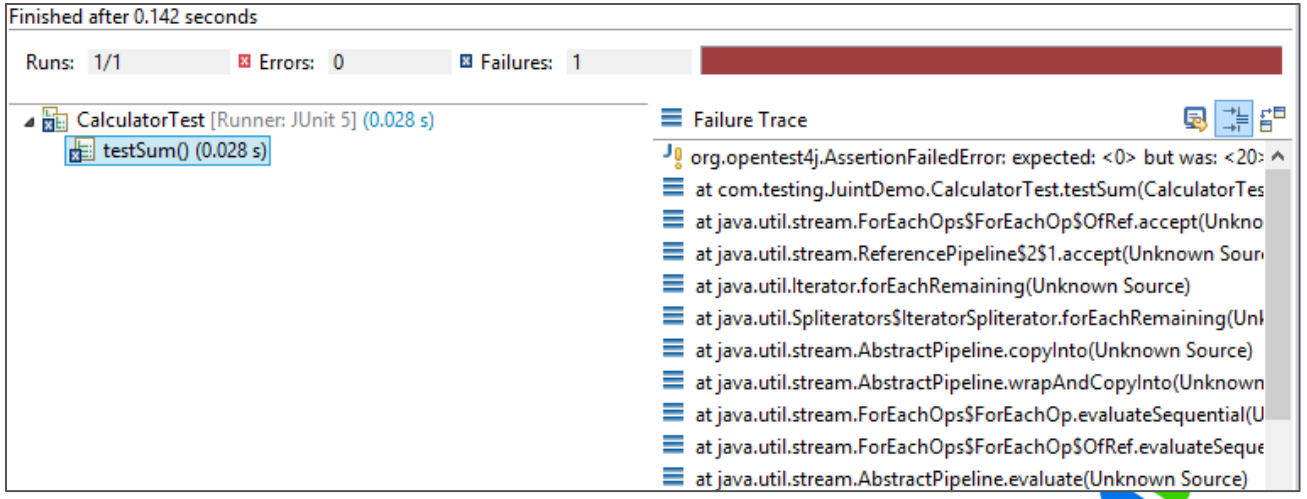
axess academy

How to execute the JUnit test cases





Successful test case



Failed test case

axess academy

Mockito for Rest API Testing

- Mockito is designed as an open-source testing framework for Java which is available under an MIT License.
- It allows programmers to create and test double objects (mock objects) in automated unit tests for the purpose of Test-driven Development (TDD).
- In simple words, we can say that Mockito is a framework that we specifically use to efficiently write certain kind of tests.
- The mock objects can be created in two ways –

1. with the static method mock(class)- MockitoAnnotations.initMocks(this) method call
2. with @Mock annotation- By annotating the class with @RunWith(MockitoJUnitRunner.class) annotation.

axess academy

Let us work on use case on JUnit

axess academy

Presenting the Use case on JUnit

- Write the JUnit test cases for Bank Customer Management System.
- Write the test case to make sure the Customer class object gets created successfully.
- Write the test cases to validate customer name, phone number, email id and account type.

axess academy