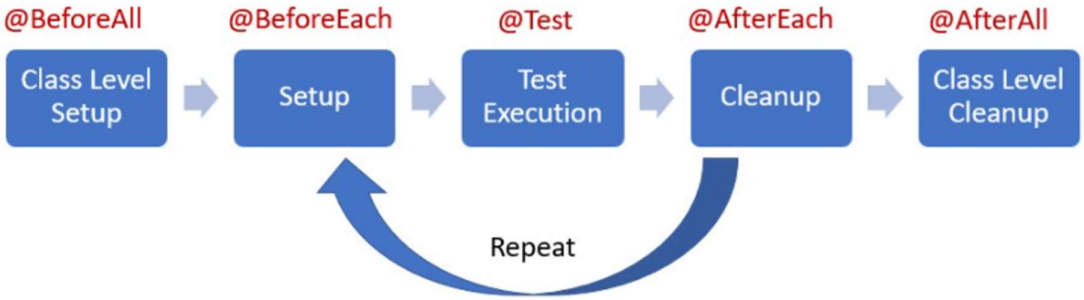
JUNIT 5 GUIDE

I. Setup Environment

- Java 8 (or higher)
- Add dependency (Using Maven)

```
<!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter -->
<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter</artifactId>
    <version>5.8.2</version>
    <scope>test</scope>
</dependency>
```

II. Lifecyle of Junit5



- **@BeforeAll:** called only once in the entire test execution cycle. And must be declared **static.** Use when you want to setup some environment for the whole test cases at once.

- **@BeforeEach:** executed ***before each*** test case. Use when you want to setup the environment just right before each test case.

- **@Test:** Denotes that a method is a test method. Put above the test method (you could see in some Test methods)

- **@AfterEach:** invoked for ***after each*** test case.

- **@AfterAll:** called only once in the entire test execution cycle. And must be declared **static**

Ex: Before each test case will create new Instance of ContactManager.

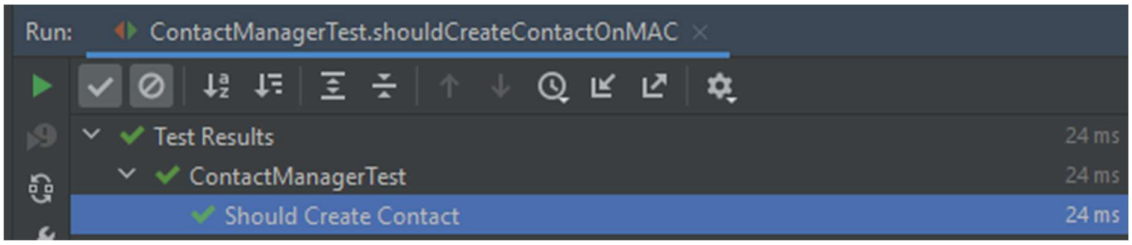
```java
class ContactManagerTest {

    private ContactManager contactManager;

    @BeforeAll
    public static void setupAll() {
        System.out.println("Should Print Before All Tests");
    }

    @BeforeEach
    public void setup() {
        System.out.println("Instantiating Contact Manager");
        contactManager = new ContactManager();
    }
```

III. Annotation

Note: The demo code below is not mention about naming convention for the test method. it should be defined and follow by each member of team.

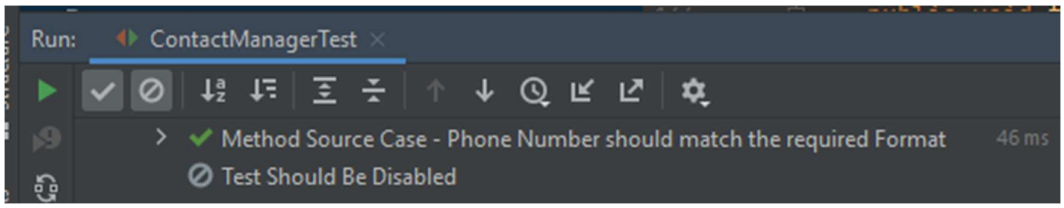
1. **@DisplayName**: Declares a custom display name for the test class or test method.

```java
@Test
@DisplayName("Should Create Contact")
public void shouldCreateContact() {
    contactManager.addContact( firstName: "John", lastName: "Doe", phoneNumber: "0123456789");
    assertFalse(contactManager.getAllContacts().isEmpty());
    assertEquals( expected: 1, contactManager.getAllContacts().size());
}
```



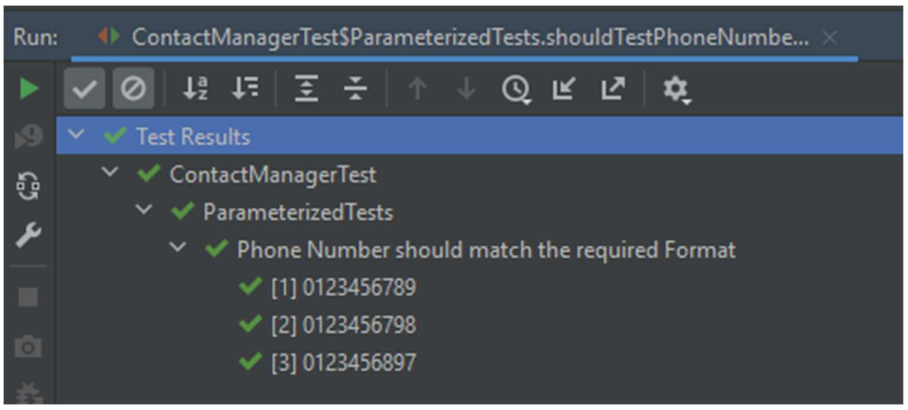
2. **@Disabled:** will **disable** the test case when run the test class. And the test case will be ignored (will not passed or failed)

```java
@Test
@DisplayName("Test Should Be Disabled")
@Disabled
public void shouldBeDisabled() {
    throw new RuntimeException("Test Should Not be executed");
}
```



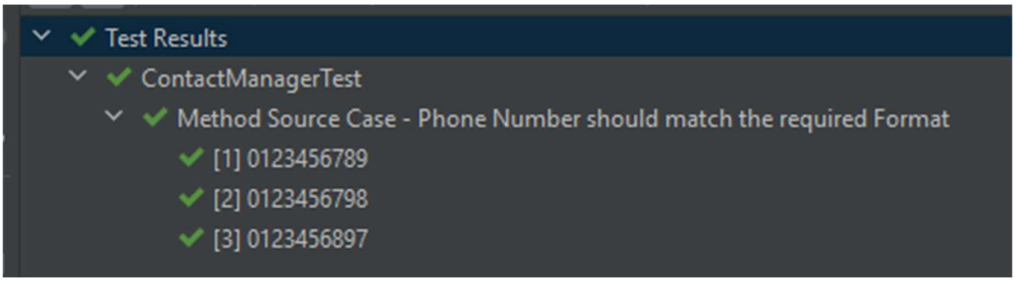
3. **@ValueSource:** Use with Annotation **@ParameterizedTest** to test value from a source (strings, ints,…).

```java
@DisplayName("Phone Number should match the required Format")
@ParameterizedTest
@ValueSource(strings = {"0123456789", "0123456798", "0123456897"})
public void shouldTestPhoneNumberFormatUsingValueSource(String phoneNumber) {
    contactManager.addContact( firstName: "John", lastName: "Doe", phoneNumber);
    assertFalse(contactManager.getAllContacts().isEmpty());
    assertEquals( expected: 1, contactManager.getAllContacts().size());

}
```



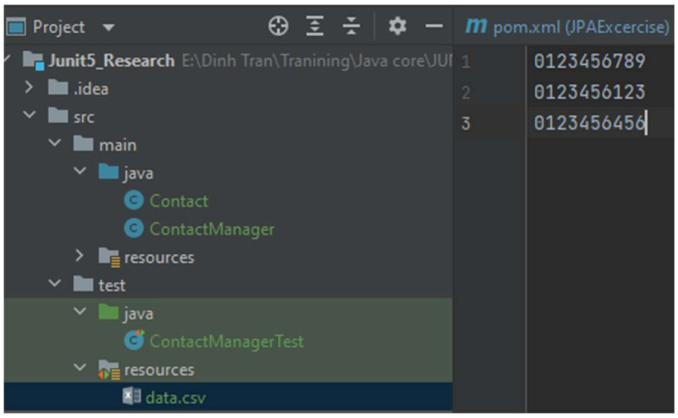
4. **@MethodSource:** will use source from a method

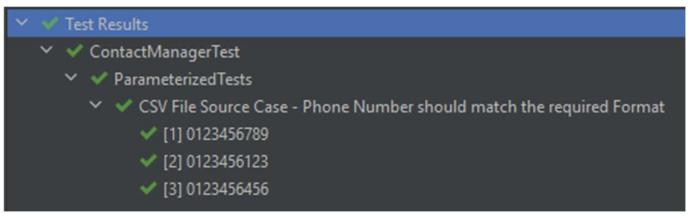
```
@DisplayName("Method Source Case - Phone Number should match the required Format")
@ParameterizedTest
@MethodSource("phoneNumberList")
public void shouldTestPhoneNumberFormatUsingMethodSource(String phoneNumber) {
    contactManager.addContact( firstName: "John", lastName: "Doe", phoneNumber);
    assertFalse(contactManager.getAllContacts().isEmpty());
    assertEquals( expected: 1, contactManager.getAllContacts().size());
}

private static List<String> phoneNumberList() {
    return Arrays.asList("0123456789", "0123456798", "0123456897");
}
```



5. **@CSVFileSource:** use data from a csv file

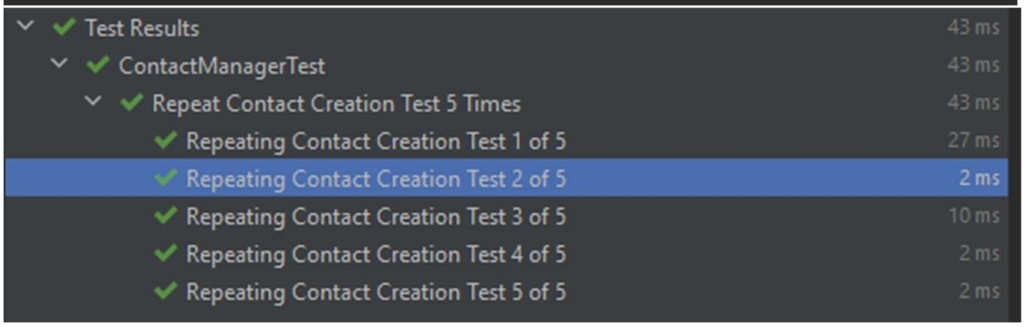
```
@DisplayName("CSV File Source Case - Phone Number should match the required Format")
@ParameterizedTest
@CsvFileSource(resources = "/data.csv")
public void shouldTestPhoneNumberFormatUsingCSVFileSource(String phoneNumber) {
    contactManager.addContact( firstName: "John", lastName: "Doe", phoneNumber);
    assertFalse(contactManager.getAllContacts().isEmpty());
    assertEquals( expected: 1, contactManager.getAllContacts().size());
}
```





6. **@RepeatedTest:** use for some case that you want to repeat the test many time (the attribute “name” could be remove if you don’t need it)

```
@DisplayName("Repeat Contact Creation Test 5 Times")
@RepeatedTest(value = 5,
        name = "Repeating Contact Creation Test {currentRepetition} of {totalRepetitions}")
public void shouldTestContactCreationRepeatedly() {
    contactManager.addContact( firstName: "John",  lastName: "Doe",  phoneNumber: "0123456789");
    assertFalse(contactManager.getAllContacts().isEmpty());
    assertEquals( expected: 1, contactManager.getAllContacts().size());
}
```



IV. Assertions

1. **assertEqual:** Check if two primitives/objects are equal

```
@Test
@DisplayName("Should Create Contact")
public void shouldCreateContact() {
    contactManager.addContact( firstName: "John",  lastName: "Doe",  phoneNumber: "0123456789");
    assertEquals( expected: 1, contactManager.getAllContacts().size());

}
```

The test case will be passed if the ContactManager have size is 1. Otherwise the test case will be failed.

2. **assertTrue / assertFalse:** assert that the specified boolean condition is true / False

```
@Test
@DisplayName("Should Create Contact")
public void shouldCreateContact() {
    contactManager.addContact( firstName: "John",  lastName: "Doe",  phoneNumber: "0123456789");
    assertFalse(contactManager.getAllContacts().isEmpty());

}
```

The result of condition “contactManger.getAllcontacts().isEmpty()” is false (as we added 1 contact into contactManage). Then the assertFalse will be passed.

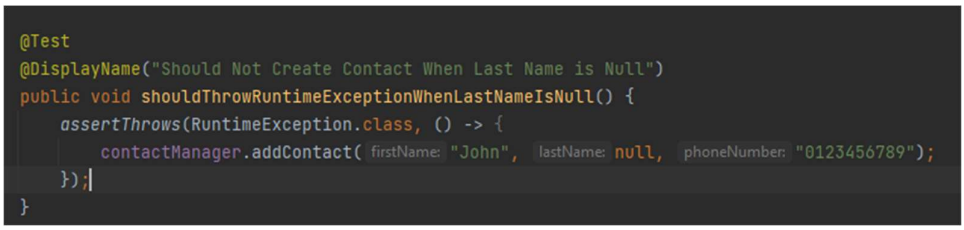
3. **assertNull:** It assert that the specified object is null.

```
@Test
public void correctInputPhoneNumber_NotPhoneNumber9Digits_ShouldReturnNull(){
    assertNull(PhoneHelpers.correctInputPhoneNumber( inputValue: "03x712345"));
}
```

The method correctInputPhoneNumber will return **null** if could not correct the inputValue to phone number format. The inputValue 03x712345 could not be corrected into phone number, then return null → Test case **passed.**

4. **assertNotNull:** assert that the specified object is not null.

5. **assertThrows:** Check if the executable throw exception.



The method addContact will throw a RuntimeException if one of firstName, lastName, phoneNumber being null. → the test case will be passed

Reference:

https://junit.org/junit5/docs/current/user-guide/#writing-tests-annotations

https://howtodoinjava.com/junit5/junit-5-test-lifecycle/#:~:text=In%20JUnit%205%2C%20the%20test,Test%20annotation%20from%20package%20org.

https://github.com/SaiUpadhyayula/contact-manager