In junit 4 hamcrest is already added as a transitive dependency

From junit 5 you have to manually add Hamcrest, but in junit 4 it was automatically added

Main jars are ()

// https://mvnrepository.com/artifact/org.hamcrest/hamcrest

testImplementation 'org.hamcrest:hamcrest:2.2'

Mockito-core

Mockito-junit-jupiter

Mockito-inline (this is to mock static and final methods)

## Run Gradle Test from command prompt intellij

* In windows use backslash only
* Dot( .) means current directory
* **Start** command in git bash Is to open the file

In intellij type below commands

1. Run “**test** “command with **gradlew** file

Command is “**.\gradlew test”**

1. To see all the test reports use the below command

**“Start** **./build/reports/tests/test/index.html**”

Note: - in git hub terminology start means open whereas dot means current directory

Hamcrest

## Run gradle tests

## **Hamcrest**

Hamcrest-library

### Documentation –

Home page: <http://hamcrest.org>

Hamcrest 2.2 is stable for several years

Java implementation: <http://hamcrest.org/JavaHamcrest/>

JavaDocs: <http://hamcrest.org/JavaHamcrest/javadoc/>

Tutorial: <https://code.google.com/archive/p/hamcrest/wikis/Tutorial.wiki>

Mailing list: <https://groups.google.com/forum/?fromgroups#!forum/hamcrest-java>

Source code: <https://github.com/hamcrest/JavaHamcrest>

## Main classes in Hamcrest

Org.hamcrest.Matcher is an interface

Org.hamcrest.Matchers is a class – all methods is , null, notnull, all these utility methods are present in this class

Org.hamcrest.MatcherAssert.

In this class we have 2 important methods 🡪 assertThat() it is a class contain only assertThat methods

#### Methods in Matchers Class

The below are the methods present in org.hamcrest.Matchers class

Famous one is

**assertThat(“abc”, is(“abc”));**

For all below assertions example refer **simpleAssertions.java,StringTests.java**

* anyOf(matcher1,matcher2); true if any matchers match
* allOf → true if all matchers match
* ContainsString(String str)

Ex:- assertThat(“abcd”,containsString(“abcdef”));

* anyOf → not → flips boolean
* isA,
* instanceOf → type test
* not
* nullValue, ex:- assertThat(null,nullValue());// to assert left hand side var is null
* notNullValue
* sameInstance
* greaterThan,
* greaterThanOrEqualTo
* lessThan,

for time comparisions use this

assertThat(LocalDate.now(),is(lessThan(localDate.now().minusDays(2))));

* lessThanOrEqualTo
* closeTo → used for floating point and BigDecimals is,
* equalTo
* equalToIgnoreCase,
* equalToCompressingWhiteSpace white space trimmed, then internal collapsed to single space
* containsString,
* endsWith,
* startsWith
* emptyString()
* emptyOrNullString()

In collections we can use these methods

hasSize(int size)

contains

containsInAnyOrder

hasItem()

In Arrays we can use these methods

## Suggestions in junit-Mockito

1. Avoid usin g Mockito.anyString()
2. Try using short hand operator Mockito.thenReturn(“abcs”, “def”); instead of using thenReturn 2 times
3. Verify() is mandatory
4. Assertion is mandatory- Use captor while writing assert statements
5. Try reading the documentation for each and every method and class

### Injections in Mockito

1. Either use Runner like annotate the class with @RunWith(MockitoJUnitRunner.class)
2. Or use @Rule
3. Or write “**MockitoAnnotations.openMocks(this);”** in @Before method

Follow any of above approaches then only all these annotations will work

@RunWith(MockitoJUnitRunner.class)

public class AddingMachineAnnotationTest {

@Mock

private List<Integer> mockList;

@InjectMocks

private AddingMachine machine;

// @Rule

// public MockitoRule rule = MockitoJUnit.rule().strictness(Strictness.STRICT\_STUBS);

// @Before

// public void setUp() {

// MockitoAnnotations.openMocks(this);

// }

# Spring related testing

Once spring container is started, you can happily autowire the application context and u can autowire Environment class

### **Load only specific configuration classes**

This will load only those specific @configuration classes this is not like mocking it will start the spring container

@SpringJUnitConfig(classes (RestTemplateConfigMock.class, MongoClientConfig.class, LogExecutionTimeConfig.class, RestTemplateConfig.class))

### **Load property files or provide properties at runtime for juint**

@TestPropertySource(locations="classpath: config/application-test.yml", properties = {

"ccs.mongo.connection.enabled-true", "spring.data.mongodb.authMechanism-PLAIN",

"ccs.mtls.resttemplate.enabled-true", "ccs.mtls.resttemplate. interceptors-com.wellsfargo.ccs.filter.LogInterceptor"})

Important points about loading property files

@TestPropertySource(locations="classpath: config/application-test.yml",

1. For the above as it is loading application-test.yml from classpath first it will check if that is already present in src/**test**/resources or not If it is present it will load else it will search from src/**main**/resources so make sure from which location you are loading

Activate profile for that test class

@ActiveProfiles("test")

Spring mock annotations

Mock related

@MockBean – means it will create a mock and it will keep that object in spring container