ackage com.simplilearn.JUnitDemo;

import org.junit.jupiter.api.AfterAll;

import org.junit.jupiter.api.BeforeAll;

import org.junit.jupiter.api.Test;

public class Demo1 {

@BeforeAll

public static void setup() {

System.out.println("Hello");

}

@Test

public void test1() {

System.out.println("Test 1");

}

@AfterAll

public static void tearDown() {

System.out.println("Bye Bye");

}

}

package com.simplilearn.JUnitDemo;

import static org.junit.jupiter.api.Assertions.assertEquals;

import static org.junit.jupiter.api.Assertions.assertTrue;

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.assertSame;

import static org.junit.jupiter.api.Assertions.assertNotSame;

import static org.junit.jupiter.api.Assertions.assertArrayEquals;

import org.junit.jupiter.api.Test;

public class Demo2 {

@Test

public void testAssertions() {

String str1 = new String("abc");

String str2 = new String("abc");

String str3 = null;

String str4 = "abc";

String str5 = "abc";

int val1 = 5;

int val2 = 6;

String[] expectedArray = {"one","two","three"};

String[] resultArray = {"one","two","three"};

//Check that two objects are equal

assertEquals(str1, str2);

//Check that a condition is true

assertTrue(val1 < val2);

//check that a condition is false

assertFalse(val1 >val2);

//Check that an object is not null

assertNotNull(str1);

//Check that an object is null

assertNull(str3);

//Check of two object references point to the same object

assertSame(str4, str5);

//Check if two object references do not point to the same object

assertNotSame(str1,str3);

//Check if 2 arrays are equal

assertArrayEquals(expectedArray,resultArray);

}

}

package com.simplilearn.JUnitDemo;

import org.junit.jupiter.api.AfterAll;

import org.junit.jupiter.api.BeforeAll;

import org.junit.jupiter.api.Disabled;

import org.junit.jupiter.api.Test;

public class Demo3 {

@BeforeAll

public static void setUp() {

System.out.println("Hello");

}

@Test

public void test1() {

System.out.println("Today");

}

@Disabled

@Test

public void test2() {

System.out.println("is");

}

@Test

public void test3() {

System.out.println("Friday");

}

@AfterAll

public static void tearDown() {

System.out.println("Bye Bye");

}

}

package com.simplilearn.JUnitDemo;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertTrue;

import org.junit.jupiter.api.Assumptions;

public class Demo4 {

@Test

public void test1() {

//assertTrue("abc".contains("a"));

Assumptions.assumeTrue("abc".contains("z"));

System.out.println("Test1");

}

@Test

public void test2() {

Assumptions.assumingThat("abc".contains("z"),()-> {

System.out.println("Friday");

});

}

}

**package** com.simplilearn.JUnitDemo;

**public** **interface** Demo5 {

**public** **void** day();

**public** **default** **void** Month() {

System.***out***.println("It is July");

}

}

package com.simplilearn.JUnitDemo;

import org.junit.jupiter.api.DisplayName;

import org.junit.jupiter.api.RepeatedTest;

import org.junit.jupiter.api.RepetitionInfo;

import org.junit.jupiter.api.TestInfo;

public class Demo6 {

@RepeatedTest(5)

public void test1() {

System.out.println("Hello");

}

@RepeatedTest(value = 5, name = "{displayName} {currentRepetition}/{totalRepetition}")

@DisplayName("Execution")

public void test2(TestInfo testinfo) {

System.out.println(testinfo.getDisplayName());

}

@RepeatedTest(5)

public void test3(RepetitionInfo repetitionInfo) {

System.out.println("Current Test Count" +repetitionInfo.getCurrentRepetition());

}

}

package com.simplilearn.JUnitDemo;

import java.util.Arrays;

import java.util.Collection;

import static org.junit.jupiter.api.Assertions.assertTrue;

import static org.junit.jupiter.api.DynamicTest.dynamicTest;

import org.junit.jupiter.api.DynamicTest;

import org.junit.jupiter.api.TestFactory;

import org.junit.jupiter.api.function.Executable;

public class Demo7 {

@TestFactory

public Collection<DynamicTest> dynamicTests(){

return Arrays.asList(

dynamicTest("Simple Test", ()-> assertTrue(true)),

dynamicTest("Executable Class", new MyExecutable()),

dynamicTest("Exception Executable",()->{throw new Exception ("Exception Example");}),

dynamicTest("Simple Test 2",()-> assertTrue(true)));

}

}

class MyExecutable implements Executable{

@Override

public void execute() {

System.out.println("Dynamic Test");

}

}

package com.simplilearn.JUnitDemo;

import static org.junit.Assert.assertTrue;

import org.junit.jupiter.params.ParameterizedTest;

import org.junit.jupiter.params.provider.ValueSource;

public class Demo8 {

@ParameterizedTest

@ValueSource(ints = {1,2,3})

public void test1(int i) {

System.out.println(i);

}

@ParameterizedTest

@ValueSource(strings = {"4", "5", "6"})

public void test2(String s) {

assertTrue(Integer.parseInt(s)< 6);

}

}

package com.simplilearn.JUnitDemo;

import org.junit.jupiter.params.ParameterizedTest;

import org.junit.jupiter.params.provider.CsvSource;

import static org.junit.Assert.assertEquals;

public class Demo9 {

@ParameterizedTest

@CsvSource({"test, TEST","monDAy, MONday","July,july"})

public void test1(String actual, String expected) {

String actualValue = actual.toLowerCase();

String expectedValue= expected.toLowerCase();

assertEquals(actualValue, expectedValue);

}

}

package com.simplilearn.JUnitDemo;

import java.lang.annotation.ElementType;

import java.lang.annotation.Retention;

import java.lang.annotation.RetentionPolicy;

import java.lang.annotation.Target;

import org.junit.jupiter.api.Tag;

import org.junit.jupiter.api.Test;

@Target({ElementType.TYPE, ElementType.METHOD})

@Retention(RetentionPolicy.RUNTIME)

@Tag("fast")

@Test

public @interface Fast {

}

package com.simplilearn.JUnitDemo;

import static org.junit.Assert.\*;

import java.util.concurrent.TimeUnit;

import org.junit.jupiter.params.ParameterizedTest;

import org.junit.jupiter.params.provider.ValueSource;

public class Demo10 {

@ParameterizedTest

@ValueSource(strings = "SECONDS")

public void test1(TimeUnit arg) {

System.out.println(arg.name());

assertNotNull(arg.name());

}

}

package com.simplilearn.JUnitDemo;

import static org.junit.Assert.assertNotNull;

import java.util.concurrent.TimeUnit;

import org.junit.jupiter.params.ParameterizedTest;

import org.junit.jupiter.params.converter.ArgumentConverter;

import org.junit.jupiter.params.converter.ConvertWith;

import org.junit.jupiter.params.provider.EnumSource;

public class Demo11 {

@ParameterizedTest

@EnumSource(TimeUnit.class)

public void test2(@ConvertWith(ArgumentConverter.class) String arg) {

assertNotNull(TimeUnit.valueOf(arg));

}

}

package com.simplilearn.JUnitDemo;

import static org.junit.Assert.assertTrue;

import java.lang.annotation.ElementType;

import java.util.EnumSet;

import org.junit.jupiter.params.ParameterizedTest;

import org.junit.jupiter.params.provider.EnumSource;

public class Demo12 {

@ParameterizedTest

@EnumSource(value = ElementType.class, names = {"TYPE","METHOD", "FIELD"})

public void test1(ElementType et) {

assertTrue(EnumSet.of(ElementType.FIELD, ElementType.TYPE,ElementType.METHOD).contains(et));

}

}

**package** com.simplilearn.JUnitDemo;

**public** **class** Demo13 {

@Fast

**public** **void** myFastTest() {

}

}

package com.simplilearn.JUnitDemo;

import org.junit.jupiter.api.Tag;

import org.junit.jupiter.api.Test;

import org.junit.platform.runner.JUnitPlatform;

import org.junit.platform.suite.api.IncludeTags;

import org.junit.runner.RunWith;

@RunWith(JUnitPlatform.class)

@IncludeTags("production")

public class Demo14 {

@Test

@Tag("development")

@Tag("production")

public void test1() {

}

@Test

@Tag("production")

public void test2() {

}

@Test

@Tag("development")

public void test3() {

}

}

package com.simplilearn.JUnitDemo;

import org.junit.jupiter.api.Tag;

import org.junit.jupiter.api.Test;

import org.junit.platform.runner.JUnitPlatform;

import org.junit.platform.suite.api.ExcludeTags;

import org.junit.runner.RunWith;

@RunWith(JUnitPlatform.class)

@ExcludeTags("production")

public class Demo15 {

@Test

@Tag("development")

@Tag("production")

public void test1() {

}

@Test

@Tag("production")

public void test2() {

}

@Test

@Tag("development")

public void test3() {

}

}