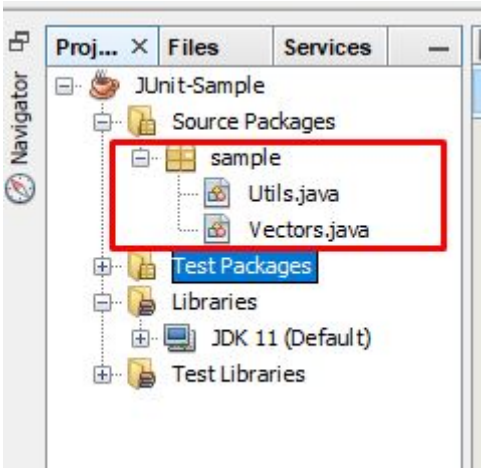
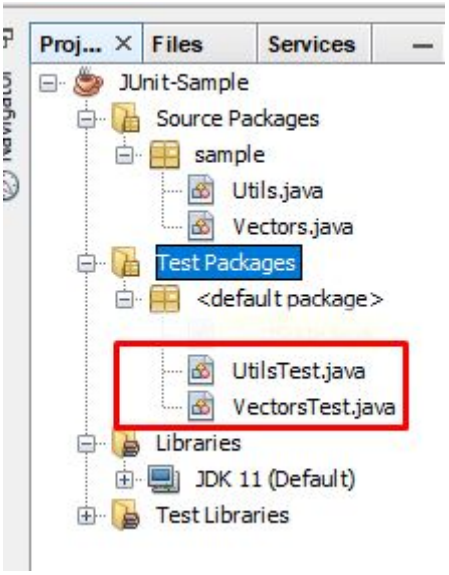
Joaquin Montero. 2DAM. DI. 5/03/2020

Generación de pruebas JUnit en el IDE NetBeans

1. **Cada alumno creará el proyecto e incorporará las clases Utils.java y Vectors.java procedentes del proyecto de ejemplo que NetBeans proporciona en su servidor de versiones tal como se especifica en el apartado correspondiente. Cada alumno analizará el código de estas clases. Y cerrará el proyecto de ejemplo de NetBeans.**



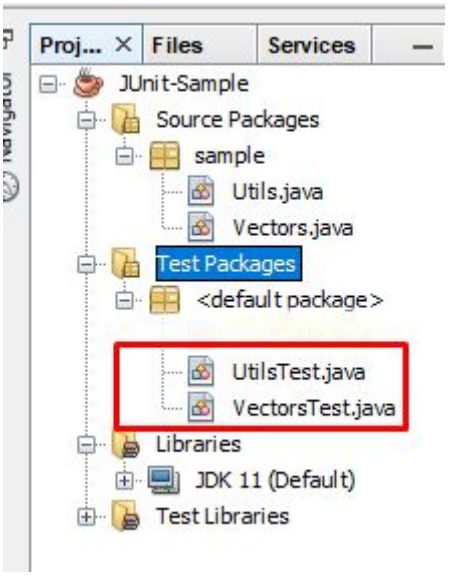
2. **Cada alumno generará la clase VectorsTest.java, y analizará el código tal como se especifica en los pasos detallados del tema. A su vez se replicarán los métodos ScalarMultiplicationCheck() y equalsCheck() de la clase de prueba VectorsTest.java tal como se especifica en el tema.**



```java
@Test
public void ScalarMultiplicationCheck()
{
    /*
    System.out.println("scalarMultiplication");
    int[] a = null;
    int[] b = null;
    int expResult = 0;
    int result = Vectors.scalarMultiplication(a, b);
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.
    fail("The test case is a prototype.");

    System.out.println("* VectorsTest: ScalarMultiplicationCheck()");

    assertEquals( 0, Vectors.scalarMultiplication(new int[] { 0, 0}, new int[] { 0, 0}));
    assertEquals( 39, Vectors.scalarMultiplication(new int[] { 3, 4}, new int[] { 5, 6}));
    assertEquals(-39, Vectors.scalarMultiplication(new int[] {-3, 4}, new int[] { 5,-6}));
    assertEquals( 0, Vectors.scalarMultiplication(new int[] { 5, 9}, new int[] {-9, 5}));
    assertEquals(100, Vectors.scalarMultiplication(new int[] { 6, 8}, new int[] { 6, 8}));
}
```

```java
    /**
     * Test of equal method, of class Vectors.
     */
    @Test
    public void equalsCheck()
    {
        System.out.println("* VectorsTest: equalsCheck()");
        assertTrue(Vectors.equal(new int[] {}, new int[] {}));
        assertTrue(Vectors.equal(new int[] {0}, new int[] {0}));
        assertTrue(Vectors.equal(new int[] {0, 0}, new int[] {0, 0}));
        assertTrue(Vectors.equal(new int[] {0, 0, 0}, new int[] {0, 0, 0}));
        assertTrue(Vectors.equal(new int[] {5, 6, 7}, new int[] {5, 6, 7}));

        assertFalse(Vectors.equal(new int[] {}, new int[] {0}));
        assertFalse(Vectors.equal(new int[] {0}, new int[] {0, 0}));
        assertFalse(Vectors.equal(new int[] {0, 0}, new int[] {0, 0, 0}));
        assertFalse(Vectors.equal(new int[] {0, 0, 0}, new int[] {0, 0}));
        assertFalse(Vectors.equal(new int[] {0, 0}, new int[] {0}));
        assertFalse(Vectors.equal(new int[] {0}, new int[] {}));

        assertFalse(Vectors.equal(new int[] {0, 0, 0}, new int[] {0, 0, 1}));
        assertFalse(Vectors.equal(new int[] {0, 0, 0}, new int[] {0, 1, 0}));
        assertFalse(Vectors.equal(new int[] {0, 0, 0}, new int[] {1, 0, 0}));
        assertFalse(Vectors.equal(new int[] {0, 0, 1}, new int[] {0, 0, 3}));
    }
```

3. **Cada alumno generará la clase UtilsTest.java, y analizará el código tal como se especifica en los pasos detallados del tema. A su vez replicará los cambios a los métodos setUpClass(), tearDownClass(), setUp(), y tearDown() de la clase de prueba UtilsTest.java tal como se especifica en los pasos detallados en el epígrafe anterior.**




```java
@BeforeClass
public static void setUpClass()
{
    System.out.println("* UtilsTest: @BeforeClass method");
}

@AfterClass
public static void tearDownClass()
{
    System.out.println("* UtilsTest: @AfterClass method");
}

@Before
public void setUp()
{
    System.out.println("* UtilsTest: @Before method");
}

@After
public void tearDown()
{
    System.out.println("* UtilsTest: @After method");
}
```

4. **Cada alumno renombrará el método concatWords() por helloWorldCheck() de la clase de prueba UtilsTest.java y añadirá los cambios tal como se especifica en el tema.**

```java
@Test
public void helloWorldCheck()
{
    System.out.println("* UtilsTest: test method 1 - helloWorldCheck()");
    assertEquals("Hello, world!", Utils.concatWords("Hello", ", ", "world", "!"));
}
/**
```

5. **Cada alumno eliminará el método TestComputeFactorial e incluirá testWithTimeout en la clase de prueba UtilsTest.java tal como se especifica en el tema.**

```java
// Esta prueba demuestra cómo verificar si un método tarda demasiado en completarse. Si el método
// tarda demasiado, el hilo de prueba se interrumpe y la prueba falla. Se puede especificar el límite de
// tiempo en la prueba.
@Test(timeout=1000)
public void testWithTimeout(){
    System.out.println("* UtilsTest: test method 2 - testWithTimeout()");
    final int factorialOf = 1 + (int) (30000 * Math.random());
    System.out.println("computing " + factorialOf + '!');
    System.out.println(factorialOf + "! = " + Utils.computeFactorial(factorialOf));
}

/**
```

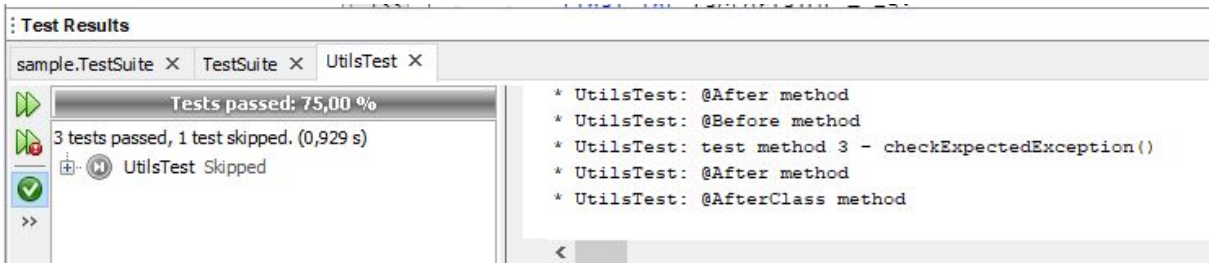
6. **Cada alumno incluirá el método checkExpectedException en la clase de prueba UtilsTest.java tal como se especifica en el tema.**

```java
// Esta prueba demuestra cómo probar una excepción esperada. El método falla si no genera la
// excepción esperada especificada. En este caso, se está probando que el método computeFactorial
// genera una IllegalArgumentException si la variable de entrada es un número negativo (-5).
@Test (expected=IllegalArgumentException.class)
public void checkExpectedException()
{
    System.out.println("* UtilsTest: test method 3 - checkExpectedException()");
    final int factorialOf = -5;
    System.out.println(factorialOf + "! = " + Utils.computeFactorial(factorialOf));
}
```

7. **Cada alumno eliminará el método testNormalizeWord e incluirá el método temporarilyDisabledTest de la clase de prueba UtilsTest.java tal como se especifica en el tema.**

```java
// Esta prueba demuestra cómo deshabilitar temporalmente un método de prueba. Simplemente
// agregando la anotación @Ignore para desactivar la prueba.
@Ignore
@Test
public void temporarilyDisabledTest() throws Exception
{
    System.out.println("* UtilsTest: test method 4 - checkExpectedException()");
    assertEquals("Malm\u00f6", Utils.normalizeWord("Malmo\u0308"));
}
```

8. **Cada alumno ejecutará e interpretará los resultados de las pruebas. Para ello ejecutará la clase de prueba UnitsTest y analizará la salida.**



9. **Cada alumno creará el conjunto de pruebas, en la clase TestSuite tal como se especifica en el punto del tema. A su vez ejecutará el conjunto de pruebas y revisará la salida.**