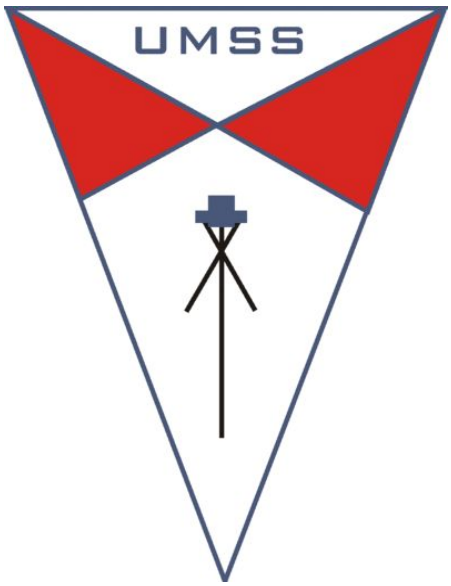
UNIVERSIDAD MAYOR DE SAN SIMÓN

FACULTAD DE CIENCIAS Y TECNOLOGÍA



PRACTICA : ASSERTS

MATERIA: PROGRAMACIÓN

NOMBRES:

GARCES HUAPALLA PAMELA MARIEL

MORUNO RODRIGUEZ TATIANA ANDREA

CARRERA:INGENIERÍA INFORMÁTICA

DOCENTE: LIC. ROSEMARY TORRICO BASCOPE

FECHA: 06-JULIO-2016

COCHABAMBA - BOLIVIA

I. PLANTEAMIENTO DEL PROBLEMA

Se desea conocer el fundamento teórico de Test unitarios y tipos de asserts disponibles en la biblioteca Java.

II. DESCRIPCIÓN (ESTRATEGIA DE SOLUCIÓN)

Se revisará la biblioteca Assert de Java, correspondiente al paquete `org.junit` para explicar e ilustrar el uso de los métodos asserts más usuales.

III. CÓDIGO

`assertArrayEquals` Este método compara si 2 arreglos son iguales, es un método sobrecargado, ya que recibe diferentes tipos de arreglos como `byte, char, int, long` u `Object`. También recibe un parámetro `message` el cual permite personalizar un mensaje en caso de que el assert falle, es decir, retorne un tipo `AssertionError`.

```
assertArrayEquals(byte[] expecteds, byte[] actuals)
assertArrayEquals(char[] expecteds, char[] actuals)
assertArrayEquals(int[] expecteds, int[] actuals)
assertArrayEquals(long[] expecteds, long[] actuals)
assertArrayEquals(java.lang.Object[] expecteds, java.lang.Object[] actuals)
assertArrayEquals(short[] expecteds, short[] actuals)
assertArrayEquals(java.lang.String message, byte[] expecteds, byte[] actuals)
assertArrayEquals(java.lang.String message, char[] expecteds, char[] actuals)
assertArrayEquals(java.lang.String message, int[] expecteds, int[] actuals)
assertArrayEquals(java.lang.String message, long[] expecteds, long[] actuals)
assertArrayEquals(java.lang.String message, java.lang.Object[] expecteds,
java.lang.Object[] actuals)
assertArrayEquals(java.lang.String message, short[] expecteds, short[] actuals)
```

Ejemplo:

```
 ...
@Test
public void testAssertArrayEquals() {
      assertArrayEquals(new int[]{1,2,3},new int[]{1,2,3});
      assertArrayEquals(new Double[]{1.1,2.2,3.3},new Double[]{1.1,2.2,3.3});
      }
```

...

assertEquals Este método compara si 2 valores son iguales, es un método sobrecargado, ya que recibe diferentes tipos de valores como `double, float, long, Object[]`. También recibe un parámetro `message` el cual permite personalizar un mensaje en caso de que el assert falle, es decir, retorne un tipo `AssertionError`.

```
assertEquals(double expected, double actual)
assertEquals(double expected, double actual, double delta)
assertEquals(float expected, float actual, float delta)
assertEquals(long expected, long actual)
assertEquals(Object[] expecteds, Object[] actuals)
assertEquals(Object expected, Object actual)
assertEquals(String message, double expected, double actual)
assertEquals(String message, double expected, double actual, double delta)
assertEquals(String message, float expected, float actual, float delta)
assertEquals(String message, long expected, long actual)
assertEquals(String message, Object[] expecteds, Object[] actuals)
assertEquals(String message, Object expected, Object actual)
```

Ejemplo:

```
...
@Test
public void testAssertEquals(){

      String expectedName = "Ana";
     assertEquals(expectedName, "Ana");

      int expectedNumber = 4;
      assertEquals(expectedNumber, 4);
}
...
```

assertFalse Este método verifica si dado un valor Booleano, es Falso. También recibe un parámetro `message` el cual permite personalizar un mensaje en caso de que el assert falle, es decir, retorne un tipo `AssertionError`.

```
assertFalse(boolean condition)
assertFalse(String message, boolean condition)
```

Ejemplo:

```
…
@Test
public void testAssertFalse(){

    Boolean expectedFalse = false;
    Boolean expectedFalseComparison = 3 < 1;

    assertFalse(expectedFalse);
    assertFalse(expectedFalseComparison);
}
...
```

assertNotEquals Es un método sobrecargado y recibe 3 parámetros: unexpected que es el valor no esperado, actual que representa el valor que se desea verificar y delta que es el valor máximo entre unexpected y actual para el que ambos números todavia son considerados iguales. También recibe un parámetro `message` el cual permite personalizar un mensaje en caso de que el assert falle, es decir, retorne un tipo `AssertionError`.

```
assertNotEquals(double unexpected, double actual, double delta)
assertNotEquals(float unexpected, float actual, float delta)
assertNotEquals(long unexpected, long actual)
assertNotEquals(Object unexpected, Object actual)
assertNotEquals(String message, double unexpected, double actual, double delta)
assertNotEquals(String message, float unexpected, float actual, float delta)
assertNotEquals(String message, long unexpected, long actual)
assertNotEquals(String message, Object unexpected, Object actual)
```

Ejemplo:

…

```
@Test
public void testAssertNoEquals(){
      int expected = 2;
      int actual = 4;
      int delta = 0;

     assertNotEquals(expected, actual,delta);

     Double expected2 = 2.6;
      Double actual2 = 2.4;
      Double delta2 = 0.1;

      assertNotEquals("not Equals",expected2, actual2,delta2);
}
```

assertNotNull Es un método sobrecargado que verifica si dado un objeto NO es nulo. También recibe un parámetro `message` el cual permite personalizar un mensaje en caso de que el assert falle, es decir, retorne un tipo `AssertionError`.

```
assertNotNull(Object object)
assertNotNull(String message, Object object)
```

```
Ejemplo
...
@Test
public void testAssertNoNull(){
      Double notNull = 2.4;

    assertNotNull(notNull);
}
...
```

assertNotSame Es un método sobrecargado que verifica que dados 2 objetos, no hagan referencia al mismo objeto. También recibe un parámetro `message` el cual permite personalizar un mensaje en caso de que el assert falle, es decir, retorne un tipo `AssertionError`.

```
assertNotSame(Object unexpected, Object actual)
assertNotSame(String message, Object unexpected, Object actual)
```

```
Ejemplo:
…
@Test
public void testAssertNotSame(){
    String str1 = new String ("abc");
    String str2 = null;

    assertNotSame(str1, str2);
}
….
```

assertNull Es un método sobrecargado que verifica si dado un objeto es nulo. También recibe un parámetro `message` el cual permite personalizar un mensaje en caso de que el assert falle, es decir, retorne un tipo `AssertionError`.

```
assertNull(Object object)
assertNull(String message, Object object)
```

```
Ejemplo:

@Test
public void testAssertNull(){
     Double nullVal = null;

    assertNull(nullVal);
}
```

assertSame Es un método sobrecargado que verifica que dados 2 objetos, hagan referencia al mismo objeto. También recibe un parámetro `message` el cual permite personalizar un mensaje en caso de que el assert falle, es decir, retorne un tipo `AssertionError`.

```
assertSame(Object expected, Object actual)
assertSame(String message, Object expected, Object actual)
```

```
Ejemplo
...
@Test
public void testAssertSame(){
      String str1 = new String ("abc");
      String str2 = str1;

    assertSame(str1, str1);
    assertSame(str1, str2);
}
…
```

assertThat Es un método sobrecargado que busca satisfacer un valor actual, a partir de una condición. Los parametros que recibe son:

`reason`, que es un mensaje explicativo de lo que se desea comparar.

`actual`, representa un tipo abstracto que representa el valor actual que esta siendo comparado.

`matcher`, es una expresión construida a partir de la biblioteca Matcher y que especifica los valores permitidos, para que se cumpla la verificación.

```
assertThat(String reason, T actual, Matcher<? super T> matcher)
assertThat(T actual, Matcher<? super T> matcher)
```

Ejemplo.

```
@Test
public void testAssertThat(){
      int number = 3;
      ArrayList<String> list = new ArrayList<String>();
      list.add("1");
      list.add("2");
      list.add("3");
      list.add("4");

      assertThat("Cero no es uno", 0, is(not(1)));
      assertThat(number, is(3));
      assertThat(number, is(not(4)));
      assertThat(list, hasItem("3"));
}
```

`assertTrue` Es un método sobrecargado que verifica si dado un valor, este es Verdadero. . También recibe un parámetro `message` el cual permite personalizar un mensaje en caso de que el assert falle, es decir, retorne un tipo `AssertionError`.

```
assertTrue(boolean condition)
assertTrue(String message, boolean condition)
```

Ejemplo:

```
…
@Test
public void testAssertTrue(){

    Boolean expectedTrue = true;
    Boolean expectedTrueComparison = (true && expectedTrue);

    assertTrue(expectedTrue);
    assertTrue(expectedTrueComparison);
}
...
```