Java

Junit5

Testing

Básicamente existen dos métodos para crear pruebas automatizadas:

- Caja Negra: las pruebas se crean para comprobar que los resultados son los esperados, sin examinar el código. Probar casos válidos e inválidos
- Caja Blanca: en base al código se crean pruebas de cobertura.
    - Las pruebas tienen que "cubrir" el 100% del código.
    - Por cada if: pruebas que "entren" al if, y pruebas que no
    - Por cada ciclo: pruebas que no entren al ciclo (si es posible), que entren una vez, más de una vez

Unit test

- Un "*software test*" es software que ejecuta otro software, verificando si el resultado es el esperado.
- Un "*unit test*" es código que ejecuta una funcionalidad específica a ser "testeada" y afirma (asserts) que se obtiene el resultado esperado.
- Cobertura de test (*Test coverage*): el porcentaje de código analizado
- El *target* de un testeo unitario es una porción pequeña de código: un método, una clase.
- Las dependencias externas (archivos de parámetros, bases de datos, etc.) no deben ser incluídas en el test. Se reemplazan con una implementación u objeto "*mock*" creado para las pruebas
- Los *unit tests* no están pensados para interfaces de usuario o interacción de components. Para ello se debe desarrollar "*integration tests*"
- En general no verifican performance

JUnit

- Normalmente los tests son creados en un proyecto aparte o separados de los códigos fuente del código "real". La convención es:
  - Clases de Java: src/main/java
  - Clases de testing: src/test/java
- JUnit 5 se encuentra en https://github.com/junit-team/junit5
- Otro framework para testeos en java es TestNG (http://testng.org/doc/)

```
import static org.junit.jupiter.api.Assertions.assertEquals;
import org.junit.jupiter.api.*;
class FirstJUnit5Tests {
    @Test
    void myFirstTest() {
        assertEquals(2, 1 + 1);
    }
}
```

Ejemplo

```
public interface BST<T> {
    Comparator<? super T> comparator();
    void add(T value);
    int size();
    boolean contains(T value);
    void remove(T value);
    Iterator<T> inorder();
    Iterator<T> preorder();
    Iterator<T> postorder();
}
```

Annotations más comunes

```
import org.junit.jupiter.api.*
```

- `@Test:` indica que es un método de testeo.
- `@BeforeAll`: se ejecuta una vez antes de comenzar los testeos. Debe ser static.
- `@AfterAll:` se ejecuta una vez al finalizar los testeos. Debe ser static.
- `@BeforeEach:` se ejecuta antes de cada método a testear.
- `@AfterEach:` se ejecuta luego de cada método testeado.

Ejemplo

```
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import java.util.Iterator;

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

public class BSTTest {

   private BST<Integer> myTree;
   private static final BST<Integer> empty =
             new BSTImpl<>((o1, o2) -> o1.compareTo(o2));
   private int expectedSize;
```

Ejemplo

```
@BeforeEach
public void setUp() {
    myTree = new BSTImpl<>((o1, o2) -> o1.compareTo(o2));
    myTree.add(8);
    myTree.add(4);
    myTree.add(3);
    myTree.add(6);
    myTree.add(1);
    // una rama "degenerada"
    for (int i = 12; i < 30; i += 2)
        myTree.add(i);
    expectedSize = 14;
}
```

Ejemplo

```
@Test
public void testSize()  {
   assertEquals(myTree.size(), expectedSize);
   assertEquals(empty.size(), 0);
}
```

```
@Test
public void testInorder()  {
   Iterator<Integer> it = myTree.inorder();
   Integer[] v = new Integer[]{1, 3, 4, 6, 8, 12, 14, 16, 18, 20, 22,
24, 26, 28};
   testorder(it, v);

   assertFalse(empty.inorder().hasNext());
}
```

Ejemplo

```
@Test
public void testPostorder()  {
   Iterator<Integer> it = myTree.postorder();
   Integer[] v = new Integer[]{1, 3, 6, 4, 28, 26, 24, 22, 20, 18, 16,
14, 12, 8};
   testorder(it, v);
}

private void testorder(Iterator<Integer> it, Integer[] v) {
   int count = 0;
   while (it.hasNext()) {
       assertEquals(it.next(), v[count++]);
   }
   assertTrue(count == v.length);
}
```

Test de cobertura

Coverage Summary for Class: BSTImpl

Class	Method, %	Line, %
BSTImpl	53.8% (7/ 13)	26% (19/ 73)
BSTImpl$BSTInorderIterator	100% (3/ 3)	94.1% (16/ 17)
BSTImpl$BSTPostorderIterator	100% (3/ 3)	90.5% (19/ 21)
BSTImpl$BSTPreorderIterator	100% (3/ 3)	93.3% (14/ 15)
BSTImpl$Node	100% (2/ 2)	100% (7/ 7)
total	75% (18/ 24)	56.4% (75/ 133)

Test de cobertura

```
57  public int size() {
58          return size;
59      }
60
61      public boolean contains(T value) {
62          return contains(root, value);
63      }
64
65      public void remove(T value) {
66          root = remove(root, value);
67      }
```

Test de cobertura

```
132  private class BSTInorderIterator<T> implements Iterator<T> {
133      Stack<Node<T>> stack;
134
135      public BSTInorderIterator(Node<T> root) {
136          stack = new Stack<Node<T>>();
137          while (root != null) {
138              stack.push(root);
139              root = root.left;
140          }
141      }
142
143      public boolean hasNext() {
144          return !stack.isEmpty();
145      }
146
147      public T next() {
148          if (stack.isEmpty())
149              throw new NoSuchElementException();
150          Node<T> node = stack.pop();
151          T value = node.value;
152          if (node.right != null) {
153              node = node.right;
154              while (node != null) {
...
```

Assertions

- `void assertTrue(boolean condition)`
- `void assertTrue(boolean condition, String message)`
- `assertFalse`
- `assertNull`
- `assertEquals`
- `assertArrayEquals`
- `assertSame`
- `asserThrows`
- `assertDoesNotThrow`
- `assertTimeout`
- `...`

https://junit.org/junit5/docs/5.0.1/api/org/junit/jupiter/api/Assertions.html

Otras annotations

- `@Disabled`
- `@DisplayName("<Name>")`
- `@RepeatedTest(<Number>)`
- `@TestFactory`
- `@Nested`
- `@Tag("<TagName>")`
- `@ExtendWith`

Ejemplo

```
@DisplayName("Single test successful")
@Test
void testSingleSuccessTest() {
    System.out.println("Success");
}

@Test
@Disabled("Not implemented yet")
void testShowSomething() {
}
```

Etc.

Para más detalles ver

https://junit.org/junit5/docs/current/user-guide/