```
package test.java;

import static org.junit.Assert.*;

import org.junit.Before;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.junit.runners.JUnit4;

import main.java.Article;

/**
 * Test unitaire de la classe Article.
 * Vous trouverez une description pour les tests les moins explicites
 *
 * @author MESTRALLET Alexis
 * @author RISCH Philippe
 */

@RunWith(JUnit4.class)

public class ArticleTest {

	private Article articleValid;
	private Article articleValid2;
	@SuppressWarnings("unused")
	private Article articleInvalid;

	@Before
	public void setUp() throws Exception {
		articleValid = new Article("Article1", 50, 1);
		articleValid2 = new Article("Article2", 50, 2);
	}

	/**
	 * Test l'assertion du cas name.isEmpty()
	 */
	@Test(expected=AssertionError.class)
	public void ArticleEmptyNameTest() {
		articleInvalid = new Article("", 50.99, 3);
	}

	/**
	 * Test de l'assertion du cas prix < 0
	 */
	@Test(expected=AssertionError.class)
	public void ArticleNegativePriceTest() {
		articleInvalid = new Article("Article1", -50.99, 3);
	}

	/**
	 * Test de l'assertion du cas number < 0
	 */
	@Test(expected=AssertionError.class)
	public void ArticleNegativeNumberTest() {
		articleInvalid = new Article("Article1", 50.99, -3);
	}

	/**
	 * Test de l'assertion du cas le plus dÃ©favorable
	 */
	@Test(expected=AssertionError.class)
	public void ArticleInvalidTest() {
		articleInvalid = new Article("", -25.99, -3);
	}

	/**
	 * Test de l'article gratuit valide
```

```
	 */
	@Test
	public void ArticleValideTest() {
		articleInvalid = new Article("Article1", 0.00, 0);
	}

	@Test
	public void getNameTest() {
		assertEquals("Article1", articleValid.getName());
	}

	@Test
	public void getPriceTest() {
		assertEquals(50, articleValid.getPrice(), 0);
	}

	@Test
	public void getNumberTest() {
		assertEquals(1, articleValid.getNumber(), 0);
	}

	@Test
	public void isEqualTest() {
		assertTrue(articleValid.isEqual(articleValid));
		assertFalse(articleValid.isEqual(articleValid2));
		assertFalse(articleValid.isEqual(new Article("Art", 50, 1)));
		assertFalse(articleValid.isEqual(new Article("Article1", 45, 1)));
		assertFalse(articleValid.isEqual(new Article("Article1", 50, 2)));
	}

	@Test
	public void smallerThanTest() {
		assertTrue(articleValid.smallerThan(articleValid2));
		assertFalse(articleValid2.smallerThan(articleValid));
		assertFalse(articleValid.smallerThan(new Article("Article1", 45, 1)));
		assertFalse(articleValid.smallerThan(new Article("Article1", 45, 2)));
		assertTrue(articleValid.smallerThan(new Article("Article1", 55, 1)));
	}
}
```