```java
package proj5;

import org.junit.Rule;
import org.junit.Test;
import org.junit.rules.Timeout;
import static org.junit.Assert.*;

/**
 * test the functionality of GenericContainableBag.java
 * author: Son Nguyen (Kyrie)
 * version: 6/4/2020
 */
public class GenericBagTest {

    @Rule // a test will fail if it takes longer than 1/10 of a second to run
    public Timeout timeout = Timeout.millis(100);

    private GenericContainableBag<Synonym> makeDefaultBag() {
        GenericContainableBag<Synonym> sample = new GenericContainableBag<>(10);
        return sample;
    }

    private GenericContainableBag<Synonym> makeSpecificBag(String text) {
        GenericContainableBag<Synonym> sample = new GenericContainableBag<>(10);
        String[] parts = text.split(" ");
        for (String part: parts) {
            sample.add(new Synonym(part));
        }
        return sample;
    }

    @Test // string representation
    public void testToString() {
        GenericContainableBag<Synonym> empty = makeDefaultBag();
        assertEquals("{}", empty.toString());
        assertEquals(0, empty.size());

        GenericContainableBag<Synonym> nonEmpty = makeSpecificBag("Hello CSC 151");
        assertEquals("{Hello, CSC, 151}", nonEmpty.toString());
        assertEquals(3, nonEmpty.size());
    }

    @Test // capacity of a bag
    public void testCapacity() {
        GenericContainableBag<Synonym> empty = makeDefaultBag();
        assertEquals(10, empty.capacity());

        GenericContainableBag<Synonym> nonEmpty = makeSpecificBag("Hello CSC 151");
        assertEquals(10, nonEmpty.capacity());
    }

    @Test // size of a bag
    public void testSize() {
        GenericContainableBag<Synonym> empty = makeDefaultBag();
        assertEquals(0, empty.size());

        GenericContainableBag<Synonym> nonEmpty = makeSpecificBag("Hello CSC 151");
        assertEquals(3, nonEmpty.size());
    }

    @Test // empty state of a bag
    public void testEmpty() {
        GenericContainableBag<Synonym> empty = makeDefaultBag();
        assertTrue(empty.isEmpty());

        GenericContainableBag<Synonym> nonEmpty = makeSpecificBag("Hello CSC 151");
        assertFalse(nonEmpty.isEmpty());
    }

    @Test // adding an item to the bag
    public void testAdd() {
        GenericContainableBag<Synonym> empty = makeDefaultBag();
        empty.add(new Synonym("Hi"));
        assertFalse(empty.isEmpty());
        assertEquals(1, empty.size());

        GenericContainableBag<Synonym> nonEmpty = makeSpecificBag("Hello CSC");
        nonEmpty.add(new Synonym("151"));
        assertEquals(3, nonEmpty.size());
    }

    @Test // check if a bag contains an item
    public void testContains() {
        GenericContainableBag<Synonym> empty = makeDefaultBag();
        assertFalse(empty.contains(new Synonym("Anything")));

        GenericContainableBag<Synonym> nonEmpty = makeSpecificBag("Hello CSC 151");
        assertTrue(nonEmpty.contains(new Synonym("Hello")));
        assertFalse(nonEmpty.contains(new Synonym("Something")));
    }

    @Test // remove an item from the bag
    public void testRemove() {
        GenericContainableBag<Synonym> empty = makeDefaultBag();
        empty.remove(new Synonym("Anything"));
        assertTrue(empty.isEmpty());

        GenericContainableBag<Synonym> nonEmpty = makeSpecificBag("Hello CSC 151 151");
        nonEmpty.remove(new Synonym("151"));
        assertEquals(3, nonEmpty.size());
    }

    @Test // remove a random item from the bag
    public void testRemoveRandom() {
        GenericContainableBag<Synonym> empty = makeDefaultBag();
        assertNull(empty.removeRandom());
        assertTrue(empty.isEmpty());

        GenericContainableBag<Synonym> nonEmpty = makeSpecificBag("Hello CSC 151");
        assertNotNull(nonEmpty.removeRandom());
        assertEquals(2, nonEmpty.size());
    }

    @Test // grab a random item from the bag
    public void testGrabRandom() {
        GenericContainableBag<Synonym> empty = makeDefaultBag();
        assertNull(empty.grabRandom());
        assertTrue(empty.isEmpty());

        GenericContainableBag<Synonym> nonEmpty = makeSpecificBag("Hello CSC 151");
        assertNotNull(nonEmpty.grabRandom());
        assertEquals(3, nonEmpty.size());
    }

    @Test // trim a bag to its size
    public void testTrim() {
        GenericContainableBag<Synonym> empty = makeDefaultBag();
        empty.trimToSize();
        assertEquals(empty.size(), empty.capacity());

        GenericContainableBag<Synonym> nonEmpty = makeSpecificBag("Hello CSC 151");
        nonEmpty.trimToSize();
        assertEquals(nonEmpty.size(), nonEmpty.capacity());
    }

    @Test // clear the bag
    public void testClear() {
        GenericContainableBag<Synonym> empty = makeDefaultBag();
        empty.clear();
        assertEquals(0, empty.size());

        GenericContainableBag<Synonym> nonEmpty = makeSpecificBag("Hello CSC 151");
        nonEmpty.clear();
        assertEquals(0, nonEmpty.size());
    }

    @Test // clone a bag
    public void testClone() {
        GenericContainableBag<Synonym> original = makeSpecificBag("Hello CSC 151");
        GenericContainableBag<Synonym> copy = makeSpecificBag("Hello CSC 151");
        assertTrue(original.equals(copy));
    }

    @Test // check if two bags is equal
    public void testEquals() {
        GenericContainableBag<Synonym> original = makeSpecificBag("Hello CSC 151");
        GenericContainableBag<Synonym> copy = makeSpecificBag("Hello CSC 151");
        assertTrue(original.equals(copy));
        copy.add(new Synonym("Something"));
        assertFalse(original.equals(copy));
    }

    @Test // merge two bag
    public void testUnion() {
        GenericContainableBag<Synonym> original = makeSpecificBag("Hello 20SP CSC 151.");
        GenericContainableBag<Synonym> toAdd = makeSpecificBag("This is Kyrie Nguyen from Hanoi, Vietnam.");
        original = original.union(toAdd);
        int expectedSize = 11;
        int expectedCapacity = 20;
        assertEquals(expectedSize, original.size());
        assertEquals(expectedCapacity, original.capacity());
    }
}

```