SWE619 Assignment-6

-*Amish Papneja*
-Avinash Arunachalam A Murugappan
- Rushil Nandan Dubey

Table of Contents

Implementation of Bag.java

Insert method:

The isIn() will Create a matcher that matches when the examined object is found within the specified collection, if so the element is given a key inside the map.put method . Else the map.put method is used to add one occurrence of e to the current object.

```
// add 1 occurrence of e to this
public void insert(E e) {
   if (isIn(e))
      map.put(e, map.get(e) + 1);
   else
      map.put(e, 1);
}
```

Remove method:

Map.get(e) will get the element's key for the current object and checks if it is greater than one and if it is greater, then the map.put method (the element and the key) is executed else if the count is equal to equal to one, then map.remove method is executed which will remove the element.

```
// remove 1 occurrence of e from this
public void remove(E e) {
   Integer count = map.get(e);
   if (count > 1)
      map.put(e, map.get(e) - 1);
   else if (count == 1)
      map.remove(e);
}
```

IsIn method:

The isIn() will Create a matcher that matches when the examined object is found within the specified collection. if and only if the object is found in the collection, then true will be returned.

```
// return true iff e is in this
public boolean isIn(E e) { return map.containsKey(e); }
```

Size method:

This method returns the size of the map which also means the number of key-value pairs present in the map.

```
// return cardinality of this
public int size() { return map.size(); }
```

Choose method:

This choose() method checks if the map.size is zero, if it is zero then it will throw ISE else it will utilize the nextInt(radix) method of java.util.Scanner class scans the next token of the input as a Int. If the translation is successful, the scanner advances past the input that matched. If the parameter radix is not passed, then it behaves similarly as nextInt(radix) where the radix is assumed to be the default radix.

```
// if this is empty throw ISE
// else return arbitrary element of this
public E choose() throws IllegalStateException {
   if (map.size() == 0)
      throw new IllegalStateException("Bag is empty");
   List<E> keysAsArray = new ArrayList<E>(map.keySet());
```

```
    Random r = new Random();
    return keysAsArray.get(r.nextInt(keysAsArray.size()));
}


// conveniently, the <E,Integer> map is exactly the abstract state
public String toString() { return map.toString(); }
```

repOk method:
Given the rep: map each object to the count of that object in this; rep-inv: range of map contains only positive integers.
In the repOk method, if the value is less than zero it will return false else the method will return true.

```
public boolean repOK() {
    for (Integer value : map.values()) {
        if (value < 0) return false;
    }
    return true;
}
```

Bag.java vs inset: LiskovGenericSet.java**:**

The insets are modeled as mathematical sets, they must have a size greater than or equal to zero, and they also must not contain duplicate elements.
Whereas in bag, there can be duplicate elements in which the count will be kept in track, and they might contain negative or zero elements also.

Implementation of BagTest.java

Liskov's Properties rule:
Bag cannot be a subtype of LiskovGenericSet because the set cannot have any duplicates. So to implement this rule, Junit is implemented which checks if cat is in the bag, in which there are 2 cats, and asserted the count.

```
@Test
public void PropertiesRuleTest(){

    Bag<String> b = new Bag<String>();
    b.insert("cat");
    b.insert("Dog");
    b.insert("cat");
    int count=0;
    for(int i=0;i<b.size();i++) {
        if(b.isIn("cat")) {
            count++;
        }
    }
    assertEquals(2,count);
}
```

Test for IsIn method:
This test will insert cat first, then assertEqual true if the element in isIn is cat, and it will assertEqual false if the bag contains dog.

```
@Test
public void testIsIn(){
    // test isIn() method
    Bag<String> b = new Bag<String>();
    b.insert("cat");
    assertEquals(true, b.isIn("cat"));
}

@Test
public void testIsIn1(){
    // test isIn() method
    Bag<String> b = new Bag<String>();
    b.insert("cat");
    assertEquals(false, b.isIn("dog"));
}
```

Test for remove method:

This testRemove method will test the remove method. Firstly, cat, dog is inserted into the bag, then cat is removed from the bag. To make sure this method works, we asserted Equals to false if there is still cat in the bag.

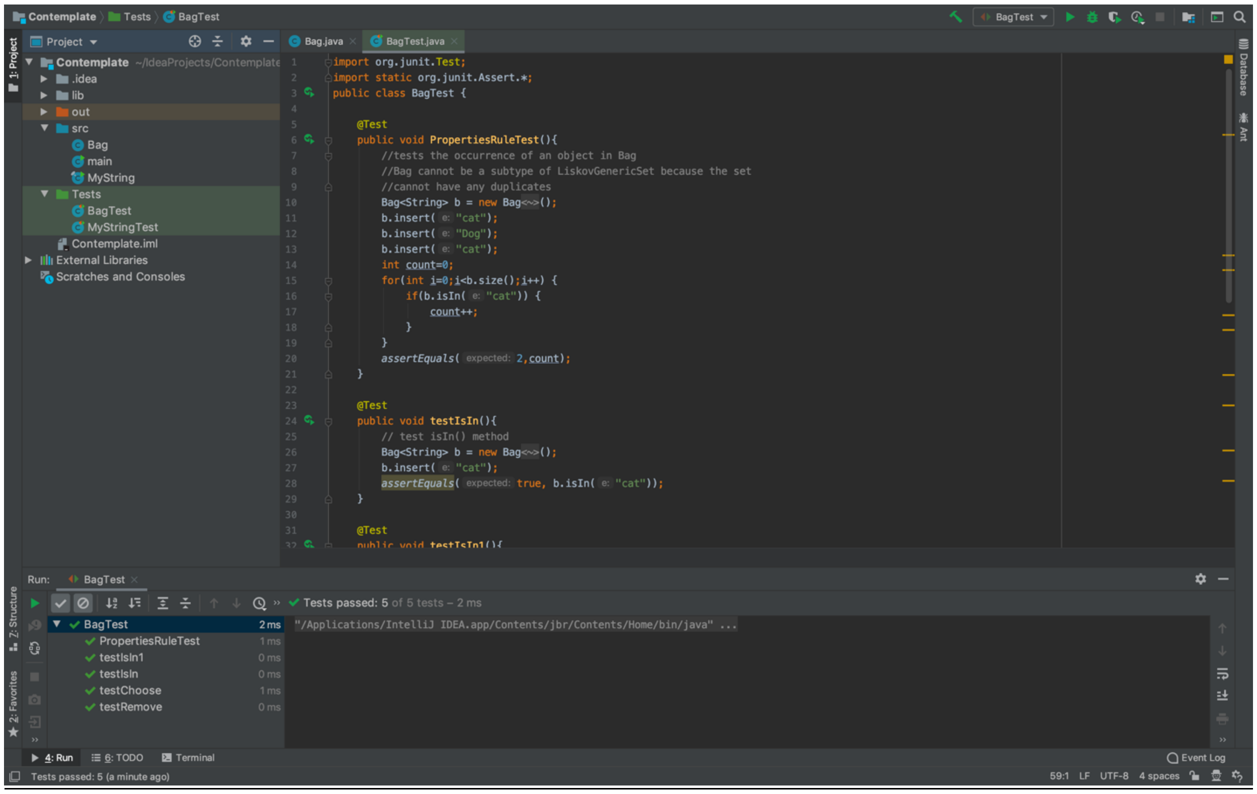
```
@Test
public void testRemove(){
    // test remove() method
    Bag<String> b = new Bag<String>();
    b.insert("cat");
    b.insert("dog");
    b.remove("cat");
    assertEquals(false, b.isIn("cat"));
}
```

Test for Choose method:

This testChoose will test the choose method, which will return arbitrary element of the collection. To make this as a test, we first insert elements into the bag, the we asserted equal true if the isIn method has the choose method. Then an element can be returned.

```
@Test
public void testChoose(){
    // test choose() method
    Bag<String> b = new Bag<String>();
    b.insert("cat");
    b.insert("dog");
    b.insert("bat");
    assertEquals(true, b.isIn(b.choose()));
}
```

Junit test passed:



Contemplate 〉 Tests 〉 BagTest

Bag.java | BagTest.java

```java
import org.junit.Test;
import static org.junit.Assert.*;
public class BagTest {

    @Test
    public void PropertiesRuleTest(){
        //tests the occurrence of an object in Bag
        //Bag cannot be a subtype of LiskovGenericSet because the set
        //cannot have any duplicates
        Bag<String> b = new Bag<~>();
        b.insert( e: "cat");
        b.insert( e: "Dog");
        b.insert( e: "cat");
        int count=0;
        for(int i=0;i<b.size();i++) {
            if(b.isIn( e: "cat")) {
                count++;
            }
        }
        assertEquals( expected: 2,count);
    }

    @Test
    public void testIsIn(){
        // test isIn() method
        Bag<String> b = new Bag<~>();
        b.insert( e: "cat");
        assertEquals( expected: true, b.isIn( e: "cat"));
    }

    @Test
    public void testIsIn1(){
```

Run: BagTest

```
✓ Tests passed: 5 of 5 tests – 2 ms

✓ BagTest                 2 ms    "/Applications/IntelliJ IDEA.app/Contents/jbr/Contents/Home/bin/java" ...
    ✓ PropertiesRuleTest  1 ms
    ✓ testIsIn1           0 ms
    ✓ testIsIn            0 ms
    ✓ testChoose          1 ms
    ✓ testRemove          0 ms
```

Tests passed: 5 (a minute ago)

Code

Bag.java

```java
import java.util.*;

public class Bag<E> {

    // rep: map each object to the count of that object in this
    // rep-inv:  range of map contains only positive integers
    // Example:  A bag of 2 cats and a dog is map = { cat=2, dog=1 }

    private Map<E, Integer> map;
    public Bag() {
        map = new HashMap<E, Integer>();
    }

    // add 1 occurrence of e to this
    public void insert(E e) {
        if (isIn(e))
            map.put(e, map.get(e) + 1);
        else
            map.put(e, 1);
    }

    // remove 1 occurrence of e from this
    public void remove(E e) {
        Integer count = map.get(e);
        if (count > 1)
            map.put(e, map.get(e) - 1);
        else if (count == 1)
            map.remove(e);
    }

    // return true iff e is in this
    public boolean isIn(E e) { return map.containsKey(e); }

    // return cardinality of this
    public int size() { return map.size(); }

    // if this is empty throw ISE
    // else return arbitrary element of this
    public E choose() throws IllegalStateException {
        if (map.size() == 0)
            throw new IllegalStateException("Bag is empty");
        List<E> keysAsArray = new ArrayList<E>(map.keySet());
        Random r = new Random();
        return keysAsArray.get(r.nextInt(keysAsArray.size()));
    }

    // conveniently, the <E,Integer> map is exactly the abstract state
    public String toString() { return map.toString(); }

    public boolean repOK() {
        for (Integer value : map.values()) {
            if (value < 0) return false;
        }
        return true;
    }

}
```

BagTest.java

```java
import org.junit.Test;
import static org.junit.Assert.*;
public class BagTest {

    @Test
    public void PropertiesRuleTest(){
        //tests the occurrence of an object in Bag
        //Bag cannot be a subtype of LiskovGenericSet because the set
        //cannot have any duplicates
        Bag<String> b = new Bag<String>();
        b.insert("cat");
        b.insert("Dog");
        b.insert("cat");
        int count=0;
        for(int i=0;i<b.size();i++) {
            if(b.isIn("cat")) {
                count++;
            }
        }
        assertEquals(2,count);
    }

    @Test
    public void testIsIn(){
        // test isIn() method
        Bag<String> b = new Bag<String>();
        b.insert("cat");
        assertEquals(true, b.isIn("cat"));
    }

    @Test
    public void testIsIn1(){
        // test isIn() method
        Bag<String> b = new Bag<String>();
        b.insert("cat");
        assertEquals(false, b.isIn("dog"));
    }

    @Test
    public void testRemove(){
        // test remove() method
        Bag<String> b = new Bag<String>();
        b.insert("cat");
        b.insert("dog");
        b.remove("cat");
        assertEquals(false, b.isIn("cat"));
    }

    @Test
    public void testChoose(){
        // test choose() method
        Bag<String> b = new Bag<String>();
        b.insert("cat");
        b.insert("dog");
        b.insert("bat");
        assertEquals(true, b.isIn(b.choose()));
    }
}
```

Contributions:

Amish Papneja (G01211503)

- *Implemented insert(), remove() for the generic type of bag.java*
- *Wrote the story for the above methods*

Avinash Arunachalam A Murugappan (G01163980)

- *Wrote Junit Tests for the methods developed from the generic type bag.java*
- *Wrote story for the junit tests*

Rushil Nandan Dubey (G01203932)

- *Implemented Choose(), repOk() for the generic type bag.java*
- *Wrote story for the above methods*