# SWE619 Assignment-11

-Amish Papneja

-Avinash Arunachalam A Murugappan

- Rushil Nandan Dubey

***Goal:*** *Favoring composition over inheritance. Bloch, Item 18.*

*Consider the InstrumentedSet example from Bloch Item 18 (as well as in-class exercise #10A).*

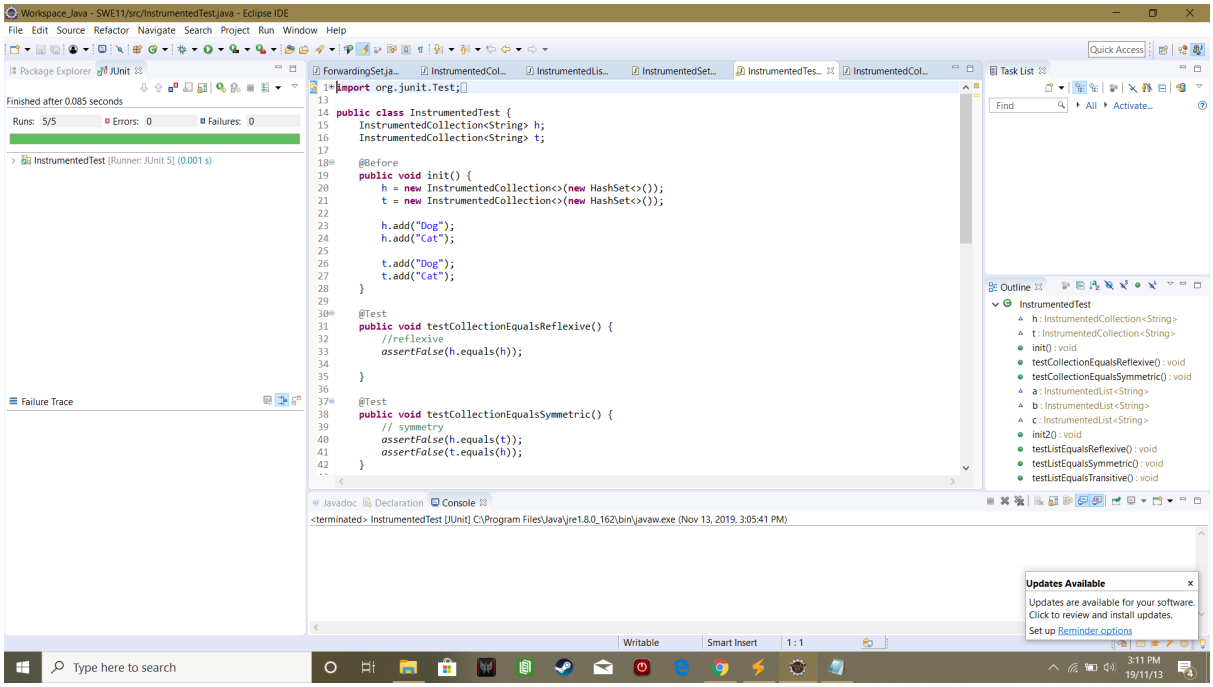
1. *Replace Set with List. There is no problem with equals(). Why not?*
2. *Replace Set with Collection. Now equals() does not satisfy its contract.*
    - *Explain why there is a problem.*
    - *Demonstrate the problem with a suitable JUnit test.*

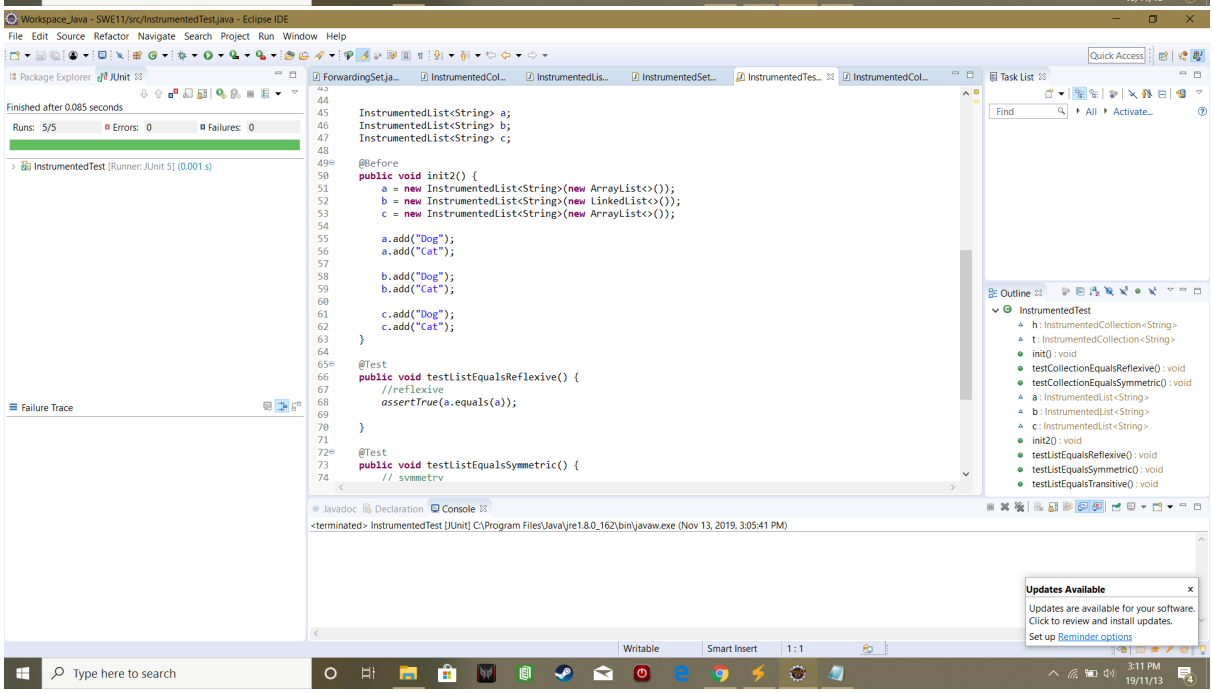
*The GTA will look for correct responses, appropriate JUnit tests, and plausible explanations when doing the grading.*

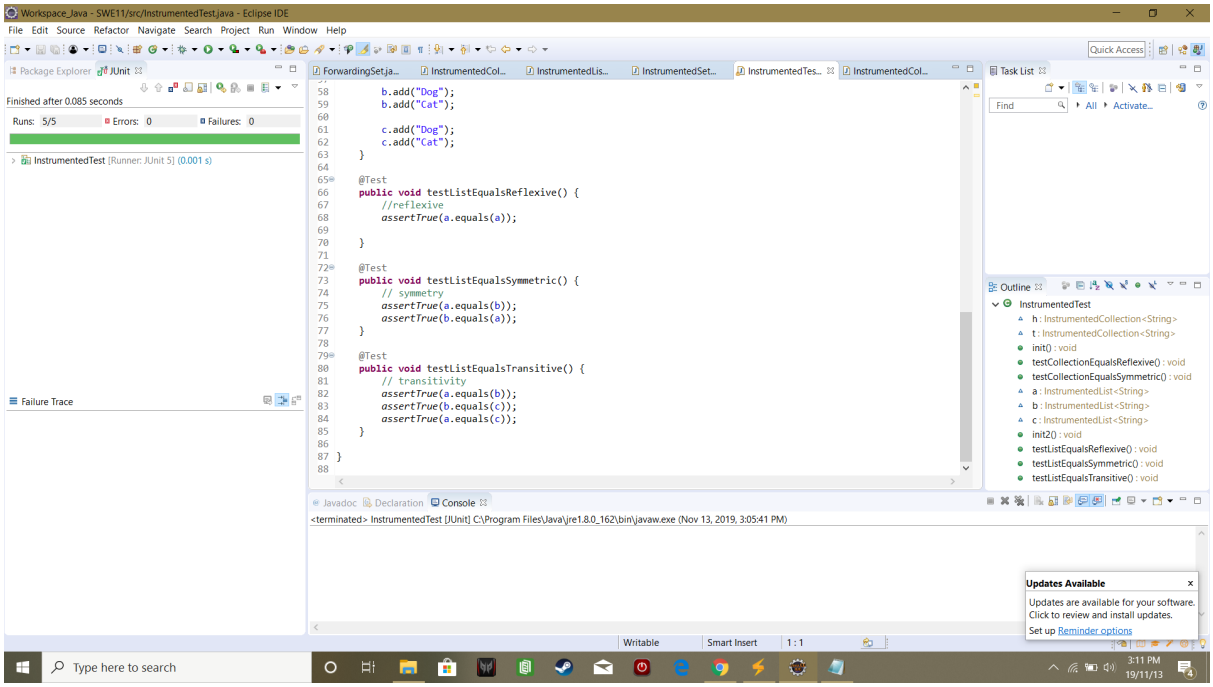
**Answer 1:**
There is no problem with the equals method in the list implementation. The equals method in the List interface calls the object's equals method. The List interface method defines two lists to be equal if they contain the same elements in the same order. It doesn't matter if the two lists being compared have different implementations, as long as they implement the List interface.

**Answer2:**
**a)**
Using the Collection interface instead of Set makes it such that equals() no longer satisfies the contract since, referring to the tests in InstrumentedTest.java, equals() compares the Collection s, which is a HashSet, in ForwardingCollection to an InstrumentedCollection object instance. The tests show that equals() fail reflexivity and symmetry when we replace the Set interface with Collection.

**b) JUnit Tests:**


Workspace_Java - SWE11/src/InstrumentedTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Finished after 0.085 seconds
Runs: 5/5 Errors: 0 Failures: 0
InstrumentedTest [Runner: JUnit 5] (0.001 s)
Failure Trace
1 import org.junit.Test;
13
14 public class InstrumentedTest {
15 InstrumentedCollection<String> h;
16 InstrumentedCollection<String> t;
17
18 @Before
19 public void init() {
20 h = new InstrumentedCollection<>(new HashSet<>());
21 t = new InstrumentedCollection<>(new HashSet<>());
22
23 h.add("Dog");
24 h.add("Cat");
25
26 t.add("Dog");
27 t.add("Cat");
28 }
29
30 @Test
31 public void testCollectionEqualsReflexive() {
32 //reflexive
33 assertFalse(h.equals(h));
34
35 }
36
37 @Test
38 public void testCollectionEqualsSymmetric() {
39 // symmetry
40 assertFalse(h.equals(t));
41 assertFalse(t.equals(h));
42 }
Outline
InstrumentedTest
h : InstrumentedCollection<String>
t : InstrumentedCollection<String>
init() : void
testCollectionEqualsReflexive() : void
testCollectionEqualsSymmetric() : void
a : InstrumentedList<String>
b : InstrumentedList<String>
c : InstrumentedList<String>
init2() : void
testListEqualsReflexive() : void
testListEqualsSymmetric() : void
testListEqualsTransitive() : void
Console
<terminated> InstrumentedTest [JUnit] C:\Program Files\Java\jre1.8.0_162\bin\javaw.exe (Nov 13, 2019, 3:05:41 PM)
Updates Available
Updates are available for your software.
Click to review and install updates.
Set up Reminder options
Writable
Smart Insert
1 : 1
Type here to search
3:11 PM
19/11/13


Workspace_Java - SWE11/src/InstrumentedTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Finished after 0.085 seconds
Runs: 5/5 Errors: 0 Failures: 0
InstrumentedTest [Runner: JUnit 5] (0.001 s)
Failure Trace
44
45 InstrumentedList<String> a;
46 InstrumentedList<String> b;
47 InstrumentedList<String> c;
48
49 @Before
50 public void init2() {
51 a = new InstrumentedList<String>(new ArrayList<>());
52 b = new InstrumentedList<String>(new LinkedList<>());
53 c = new InstrumentedList<String>(new ArrayList<>());
54
55 a.add("Dog");
56 a.add("Cat");
57
58 b.add("Dog");
59 b.add("Cat");
60
61 c.add("Dog");
62 c.add("Cat");
63 }
64
65 @Test
66 public void testListEqualsReflexive() {
67 //reflexive
68 assertTrue(a.equals(a));
69
70 }
71
72 @Test
73 public void testListEqualsSymmetric() {
74 // symmetry
Outline
InstrumentedTest
h : InstrumentedCollection<String>
t : InstrumentedCollection<String>
init() : void
testCollectionEqualsReflexive() : void
testCollectionEqualsSymmetric() : void
a : InstrumentedList<String>
b : InstrumentedList<String>
c : InstrumentedList<String>
init2() : void
testListEqualsReflexive() : void
testListEqualsSymmetric() : void
testListEqualsTransitive() : void
Console
<terminated> InstrumentedTest [JUnit] C:\Program Files\Java\jre1.8.0_162\bin\javaw.exe (Nov 13, 2019, 3:05:41 PM)
Updates Available
Updates are available for your software.
Click to review and install updates.
Set up Reminder options
Writable
Smart Insert
1 : 1
Type here to search
3:11 PM
19/11/13


Workspace_Java - SWE11/src/InstrumentedTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Package Explorer
JUnit
Finished after 0.085 seconds
Runs: 5/5
Errors: 0
Failures: 0
InstrumentedTest [Runner: JUnit 5] (0.001 s)
Failure Trace
ForwardingSet.ja...
InstrumentedCol...
InstrumentedLis...
InstrumentedSet...
InstrumentedTes...
InstrumentedCol...
58 b.add("Dog");
59 b.add("Cat");
60
61 c.add("Dog");
62 c.add("Cat");
63 }
64
65 @Test
66 public void testListEqualsReflexive() {
67 //reflexive
68 assertTrue(a.equals(a));
69
70 }
71
72 @Test
73 public void testListEqualsSymmetric() {
74 // symmetry
75 assertTrue(a.equals(b));
76 assertTrue(b.equals(a));
77 }
78
79 @Test
80 public void testListEqualsTransitive() {
81 // transitivity
82 assertTrue(a.equals(b));
83 assertTrue(b.equals(c));
84 assertTrue(a.equals(c));
85 }
86
87 }
88
Task List
Find
All
Activate...
Outline
InstrumentedTest
h : InstrumentedCollection<String>
t : InstrumentedCollection<String>
init() : void
testCollectionEqualsReflexive() : void
testCollectionEqualsSymmetric() : void
a : InstrumentedList<String>
b : InstrumentedList<String>
c : InstrumentedList<String>
init2() : void
testListEqualsReflexive() : void
testListEqualsSymmetric() : void
testListEqualsTransitive() : void
Javadoc
Declaration
Console
<terminated> InstrumentedTest [JUnit] C:\Program Files\Java\jre1.8.0_162\bin\javaw.exe (Nov 13, 2019, 3:05:41 PM)
Updates Available
Updates are available for your software.
Click to review and install updates.
Set up Reminder options
Writable
Smart Insert
1 : 1
Type here to search
3:11 PM
19/11/13

# Code Implementation

1. **ForwardingSet.java:**

```java
import java.util.Iterator;
import java.util.Set;
import java.util.Collection;

public class ForwardingSet<E> implements Set<E> {
    private final Set<E> s;
    public ForwardingSet(Set<E> s) {this.s = s; }

    public void clear() { s.clear(); }
    public boolean contains(Object o) { return s.contains(o); }
    public boolean isEmpty() { return s.isEmpty(); }
    public int size() { return s.size(); }
    public Iterator<E> iterator() {
        return s.iterator();
    }
    public boolean add(E e) { return s.add(e); }
    public boolean remove(Object o) { return s.remove(o); }
    public boolean containsAll(Collection<?> c) {
        return s.containsAll(c);
    }
    public boolean addAll(Collection<? extends E> c) {
        return s.addAll(c);
    }
    public boolean removeAll(Collection<?> c) {
        return s.removeAll(c);
    }
    public boolean retainAll(Collection<?> c) {
        return s.retainAll(c);
    }
    public Object[] toArray() {
        return s.toArray();
    }
    public <T> T[] toArray(T[] a) {
        return s.toArray(a);
    }
    @Override
    public boolean equals(Object o) {
        return s.equals(o);
    }
    @Override
    public int hashCode() {
        return s.hashCode();
    }
    @Override
    public String toString() {
        return s.toString();
    }
}
```

2. **InstrumentedCollection.java:**

```java
import java.util.Collection;
import java.util.Iterator;

public class InstrumentedCollection<E> extends ForwardingCollection<E>  {
    private int addCount = 0;

    public InstrumentedCollection(Collection<E> c) { super(c); }

    @Override public boolean add(E e) { addCount++; return super.add(e); }

    @Override public boolean addAll(Collection<? extends E> c) {
        addCount += c.size();
        return super.addAll(c);
    }

    public int getAddCount() {
        return addCount;
    }
}

class ForwardingCollection<E> implements Collection<E> {
    private final Collection<E> s;

    public ForwardingCollection(Collection<E> s) {this.s = s; }

    public void clear() { s.clear(); }
    public boolean contains(Object o) { return s.contains(o); }
    public boolean isEmpty() { return s.isEmpty(); }
    public int size() { return s.size(); }
    public Iterator<E> iterator() {
        return s.iterator();
    }
    public boolean add(E e) { return s.add(e); }
    public boolean remove(Object o) { return s.remove(o); }
    public boolean containsAll(Collection<?> c) {
        return s.containsAll(c);
    }
    public boolean addAll(Collection<? extends E> c) {
        return s.addAll(c);
    }
    public boolean removeAll(Collection<?> c) {
        return s.removeAll(c);
    }
    public boolean retainAll(Collection<?> c) {
        return s.retainAll(c);
    }
    public Object[] toArray() {
        return s.toArray();
    }
    public <T> T[] toArray(T[] a) {
        return s.toArray(a);
    }
    @Override
    public boolean equals(Object o) {
        return s.equals(o);
    }
    @Override
    public int hashCode() {
        return s.hashCode();
    }
    @Override
    public String toString() {
        return s.toString();
    }
}
```

### 3. InstrumentedList.java:

```
import java.util.Collection;
import java.util.Iterator;
import java.util.List;
import java.util.ListIterator;


public class InstrumentedList<E> extends ForwardingList<E>{
    private int addCount = 0;

    public InstrumentedList(List<E> l){ super(l); }

    @Override public boolean add(E e){
        addCount++;
        return super.add(e);
    }

    @Override public void add(int index, E e){
        addCount++;
        super.add(index, e);
    }

    @Override public boolean addAll(Collection<? extends E> c) {
        addCount += c.size();
        return super.addAll(c);
    }

    public int getAddCount(){ return addCount; }
}

class ForwardingList<E> implements List<E> {
    private final List<E> l;

    public ForwardingList(List<E> l)  { this.l = l; }
    public boolean add(E e)           { return l.add(e); }
    public void add(int index, E e)   { l.add(index, e); }
    public boolean addAll(Collection<? extends E> c)            { return
l.addAll(c); }
    public boolean addAll(int index, Collection<? extends E> c) { return
l.addAll(index, c); }
    public void clear()               { l.clear(); }
    public boolean contains(Object o) { return l.contains(o); }
    public boolean containsAll(Collection<?> c)                 { return
l.containsAll(c); }
    public boolean equals(Object o)   { return l.equals(o); }
    public E get(int index)           { return l.get(index); }
    public int hashCode()             { return l.hashCode(); }
    public int indexOf(Object o)      { return l.indexOf(o); }
    public boolean isEmpty()          { return l.isEmpty(); }
    public Iterator<E> iterator()     { return l.iterator(); }
    public int lastIndexOf(Object o)  { return l.lastIndexOf(o); }
    public ListIterator<E> listIterator()                       { return
l.listIterator(); }
    public ListIterator<E> listIterator(int index)              { return
l.listIterator(index); }
    public E remove(int index)        { return l.remove(index); }
    public boolean remove(Object o)   { return l.remove(o); }
    public boolean removeAll(Collection<?> c)                   { return
l.removeAll(c); }
    public boolean retainAll(Collection<?> c)                   { return
l.retainAll(c); }
    public E set(int index, E element)                          { return
l.set(index, element); }
    public int size()                 { return l.size(); }
    public List<E> subList(int fromIndex, int toIndex)          { return
l.subList(fromIndex, toIndex); }
```

```
    public Object[] toArray()          { return l.toArray(); }
    public <T> T[] toArray(T[] a)      { return l.toArray(a); }
    public String  toString()          { return l.toString(); }

}
```

### 4. InstrumentedSet.java:

```
import java.util.Set;
import java.util.Collection;

public class InstrumentedSet<E> extends ForwardingSet<E> {
    private int addCount = 0;

    public InstrumentedSet(Set<E> s) {
        super(s);
    }

    @Override
    public boolean add(E e) {
        addCount++;
        return super.add(e);
    }

    @Override
    public boolean addAll(Collection<? extends E> c) {
        addCount += c.size();
        return super.addAll(c);
    }

    public int getAddCount() {
        return addCount;
    }
}
```

### 5. InstrumentedTest.java

```
import org.junit.Test;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertFalse;
import static org.junit.Assert.assertTrue;

import org.junit.Before;

import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;
import java.util.HashSet;

public class InstrumentedTest {
    InstrumentedCollection<String> h;
    InstrumentedCollection<String> t;

    @Before
    public void init() {
        h = new InstrumentedCollection<>(new HashSet<>());
        t = new InstrumentedCollection<>(new HashSet<>());

        h.add("Dog");
```

```
        h.add("Cat");

        t.add("Dog");
        t.add("Cat");
    }

    @Test
    public void testCollectionEqualsReflexive() {
        //reflexive
        assertFalse(h.equals(h));

    }

    @Test
    public void testCollectionEqualsSymmetric() {
        // symmetry
        assertFalse(h.equals(t));
        assertFalse(t.equals(h));
    }


    InstrumentedList<String> a;
    InstrumentedList<String> b;
    InstrumentedList<String> c;

    @Before
    public void init2() {
        a = new InstrumentedList<String>(new ArrayList<>());
        b = new InstrumentedList<String>(new LinkedList<>());
        c = new InstrumentedList<String>(new ArrayList<>());

        a.add("Dog");
        a.add("Cat");

        b.add("Dog");
        b.add("Cat");

        c.add("Dog");
        c.add("Cat");
    }

    @Test
    public void testListEqualsReflexive() {
        //reflexive
        assertTrue(a.equals(a));

    }

    @Test
    public void testListEqualsSymmetric() {
        // symmetry
        assertTrue(a.equals(b));
        assertTrue(b.equals(a));
    }

    @Test
    public void testListEqualsTransitive() {
        // transitivity
        assertTrue(a.equals(b));
        assertTrue(b.equals(c));
        assertTrue(a.equals(c));
    }

}
```

**Contributions:**

**-Amish Papneja**

- Implemented InstrumentedCollection portion of the assignment

-Contributed to discussion regarding the assignment

**-Avinash Arunachalam A Murugappan**

- Implemented InstrumentedTest portion of the assignment

-Contributed to discussion regarding the assignment

**- Rushil Nandan Dubey**

- Implemented InstrumentedList portion of the assignment

-Contributed to discussion regarding the assignment