CS 7140 Advanced Software Engineering Homework 2 Question 1

Prof: Dr. Mateti
Submitted by: Pranav Pranav

26 July 2016

1 Introduction

This class implements a set of integers. It is limited by the size sz chosen at compile time. This is expected to be a good example of software engineering in the tiny. Possibly contains bugs. We naively assume all objects are non-null. It is also intended as an example for use with learning JUnit/TestNG, JML and FindBugs.In this document we will go through the implementation of a small set class implemented in java by Dr. Mateti. For further details, see http://cecs.wright.edu/ pmateti/Courses/7140/Lectures [1] /Design/smallSet-DesignDoc.html; long names have been shortened. [2]

1.1 Small Set Class

Apparantly javadoc has no tags for pre- and post-conditions; so I am placing assert()s at the top of the methods for pre- and at the bottom for post. As is, we are not using JML syntax. The old objects, setOf() and classInv() are needed only because of asserts. The obj1 == obj2 used below is intended to be deep equality.

The class SMALLSET contains three properties (ear, ux, sz). The elements are stored in (ear) array while (ux) is an index that is always referring to the position after the last element in the set. The property (sz) is the maximam size of the set. The set is initialized using the constructors as shown in the code below. To maintain sets the class SMALLSET exploit these functions:

- SmallSet setOf(SmallSet s)
- SmallSet setOf(int [] a, int m, int n)
- indexOf(int e)

[1]http://cecs.wright.edu/ pmateti/Courses/7140/Lectures
[2]/Design/smallSet-DesignDoc.html

- isin(int e)
- SmallSet compact()
- SmallSet insert(int e)
- delete(int a, int b, int e)
- delete(int e)
- int nitems()
- SmallSet union(SmallSet tba)
- SmallSet diff(SmallSet tbs)
- String toString()
- main(String[] args)

The upcoming sections explain these functions in more details.

</SmallSet.java 1>

```
public class SmallSet {
    private int[] ear;
    private int ux = 0;
    private int sz = 1000;
    public SmallSet() {
        ear = new int[sz];
    }

    public SmallSet(int [] a, int m, int n) {
        assert m < n;
        ear = new int[sz];
        for (int i = m; i < n; i++) {
            insert(a[i]);
        }
    }
    <<classinvariant 2>>
    <<set Of 3>>
    <<indexOf 4>>
    <<isin 5>>
    <<Compact 6>>
    <<Insert 7>>
    <<Delete 8>>
  <<nitems 9>>
  <<Union and Diff 10>>
  <<ToString 11>>
  <<main 12>>
}
```

INCLUDED BLOCKS: 2 on page **??**, 3 on page **??**, 4 on page **??**, 5 on page **??**, 6 on page **??**, 7 on page **??**, 8 on page **??**, 9 on page **??**, 10 on page **??**, 11 on page **??**, 12 on page **??**

1.1.1 Class Invariant

As to make sure of the correctness of the object, the objects must always holds true. This can be achieved by making sure the index (ux) is not less than 0 and not more than size (sz). This function is needed only for assert. @param s any SmallSet Class invariant, expressed as a boolean function. @return truthhood of the boolen exp of the invariant

<classinvariant 2>

```java
public boolean classInv(SmallSet s) {
        return
            0 <= s.ux && s.ux < s.sz
            && setOf(s) == setOf(s.ear, 0, ux-1)
            ;
    }
```

USED IN: /SmallSet.java on page **??**

1.1.2 (Set Of)

The setOf() function is needed only for assert.

<set Of 3>

```java
private SmallSet setOf(SmallSet s) {
        return s.compact();
    }

    private SmallSet setOf(int [] a, int m, int n) {
        return setOf(new SmallSet(a, m, n));
    }
```

USED IN: /SmallSet.java on page **??**

1.1.3 Returining the index of an element

Returning the index of an element (e).

<indexOf 4>

```java

    private int indexOf(int e) {
        assert classInv(this);
        int i = 0;
        ear[ux] = e;
        while (ear[i] != e)
            i++;
        assert 0 <= i && i <= ux && e == ear[i] ;
        return i;
    }
```

USED IN: /SmallSet.java on page **??**

1.1.4 Checking if the element exist

This function return true if element (e) exist otherwise it returns false. This function invoking indexOf(e) to return the index of an element. @param e element to search for @return any i such that ear[i] == e

<isin 5>

```
 public boolean isin(int e) {
        int x = indexOf(e);
        assert x >= ux || (x == ear[x] && 0 <= x && x < ux);
        return (x < ux);
    }
```

USED IN: /SmallSet.java on page **??**

1.1.5 Compact

Compacting the elements to avoid reputation of an element by deleting elements one by one and copying it to the new object (newset) and returning it. While keeping the abstraction intact, compactify the ear[] so that ux can be the lowest possible.Rewrite it without using newset. @return the new SmallSet equal to setOf(this)

<Compact 6>

```
  public SmallSet compact() {
        SmallSet newset = new SmallSet();
        while (ux > 0) {
            int e = ear[ux-1];
            delete(e);
            if (! newset.isin(e))
                newset.insert(e);
        }
        assert setOf(this) == setOf(newset) && newset.ux <=
            this.ux;
        return newset;
    }
```

USED IN: /SmallSet.java on page **??**

1.1.6 Insert

Inserting an element is done by copying the first element to where (ux) is pointing and then replace the first element with the element that need to be inserted.

<Insert 7>

```
    public SmallSet insert(int e) {
        if (ux < sz - 1) {
            if (ux > 0)
                ear[ux] = ear[0];
            ear[0] = e;
            ux++;
        } else {

        }
        assert ux == sz - 1 || isin(e);
        return this;
    }
```

USED IN: /SmallSet.java on page **??**

1.1.7 Delete

To delete element (e) the function with one parameter is first invoked and then in turn invoking the delete function with the three parameters. where (a) is 0, (b) is the index. This function is returning the number of required deletion (nd). Delete all occurrences of e in ear[a to b-1], and compact ear[]. A casual reader comments: This is tricky! Do write a loop invariant. @return the count of deletions.

<Delete 8>

```
    private int delete(int a, int b, int e) {
        assert 0 <= a && a < b;
        int nd = 0;

        for (int i = a, j = a; i < b; i++) {
            if (ear[i] == e)
                nd ++;
            else {
                if (j < i)
                    ear[j] = ear[i];
                j++;
            }
        }
        assert 0 <= nd && nd < b - a && ! setOf(ear, a, b).isin
            (e);
        return nd;
    }


    public SmallSet delete(int e) {
        assert ux < sz;
        ux -= delete(0, ux, e);
        assert ux < sz && !isin(e);
        return this;
    }
```

USED IN: /SmallSet.java on page **??**

1.1.8 Returning number of items

Returning the number of items in the set that is represented by the index (ux). Invoking the compact() is necessary to compact the set and avoid reputation.

<nitems 9>

```
    public int nitems() {
        SmallSet s = compact();
        ux = s.ux;
        ear = s.ear;
        assert ux == setOf(s).nitems();
        return ux;
    }
```

USED IN: /SmallSet.java on page **??**

1.1.9 Adding and Subtracting Sets

Concatenating two sets is done by the union() function where (tba) represent the set to be added. Subtracting two sets is doen by the diff() function where (tbs) represent the set that need to be subtracted.

<Union and Diff 10>

```
   public SmallSet union(SmallSet tba) {
        SmallSet old = this;
        for (int i = 0; i < tba.ux; i++)
            insert(tba.ear[i]);
        assert setOf(this) == setOf(old).union(setOf(tba));
        return this;
    }

    public SmallSet diff(SmallSet tbs) {
        SmallSet old = this;
        SmallSet newset = new SmallSet();
        for (int i = 0; i < tbs.ux; i++)
            if (! this.isin(tbs.ear[i])) newset.insert(tbs.ear[
                i]);
        for (int i = 0; i < this.ux; i++) {
            if (! tbs.isin(this.ear[i])) newset.insert(this.ear
                [i]);
        }
        ear = newset.ear;
        ux = newset.ux;
        assert setOf(this) == setOf(old).diff(setOf(tbs));
        return this;
    }
```

USED IN: /SmallSet.java on page **??**

1.1.10 Printing nicely organized set

This function override the original ToString() to provide a customized printing pattern for the set.

<ToString 11>

```
    public String toString() {
        String s = "el: ";
        for (int i = 0; i < ux; i++) {
            s += ", " + ear[i];
        }
        s += ", ux=" + ux;
        return s;
    }
```

USED IN: /SmallSet.java on page **??**

1.1.11 Testing some data

Some test data to test the implemented class SMALLSET.

<main 12>

```
    public static void main(String[] args) {
        SmallSet s = new SmallSet();
        SmallSet t = new SmallSet();
        int [] a = {1,2,3,4,5,6};
        for (int j = 0; j < 3; j++)
            for (int i = 0; i < a.length; i++) { // or use
                setOf()

                s.insert(a[i]);
                t.insert(a[i] - 1);
            }
        // some simple tests

        System.out.println("set s " + s + "; nitems=" + s.
            nitems());
        s.delete(1);
        s.delete(3);
        System.out.println("set s " + s + "; nitems=" + s.
            nitems());
        s.union(t);
        System.out.println("set s " + s + "; nitems=" + s.
            nitems());
        t.insert(7);
        t.diff(s);
        System.out.println("set t " + t + "; nitems=" + t.
            nitems());
    }
```

USED IN: /SmallSet.java on page **??**

1.1.12 The output

set s el: , 6, 1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 5, ux=18; nitems=6 set s el: , 6, 5, 4, 2, ux=4; nitems=4 set s el: , 4, 2, 4, 5, 6, 5, 0, 1, 2, 3, 4, 5, 0, 1, 2, 3, 4, 5, 0, 1, 2, 3, ux=22; nitems=7 set t el: , 7, 6, ux=2; nitems=2

2 Discussion

This program is designed in a way that meet the Design by Contract. The class invariant, post-conditions, and preconditions are used rigorously in this class. One issue and as shown in the output section the set when printed for the first time it is not compacted, thus, there is a need to invoke nitems() before printing the set