Programming languages / Java (Bsc) Lab 8

Task 1

Open a Java Shell (`jshell`) and play around with the bitwise operators provided by the Java language (which are similar to those provided by the C language).

- Check the value of `0xBEE_FACED_BABE_ADEL`.
- Try the operators `&`, `|`, `~`, `^`, `<<`, `>>` and `>>>`.
- What are the prime factors of 1984? How many times can you divide it by 2? How do you express this with `>>`?

  https://www.geeksforgeeks.org/bitwise-operators-in-java/

  https://www.youtube.com/watch?v=mdafxtP4RZU

Task2

Define an `UnmodifiableStringArray` type, which is an immutable representation of an array of `Strings`.

The constructor of `UnmodifiableStringArray` has a `String[]` parameter. The class has the following methods:

- `size()`: the number of `String` elements
- `maxLength()`: the length of the longest element in the array
- `minLength()`: the length of the shortest element in the array
- `allLength()`: the sum of the lengths of the elements in the array
- `contains(String str)`: decides whether `str` is in the array
- `empty()`: a class-wide method, returning an `UnmodifiableStringArray`
- `get(int index)`: returns the element at the given index (if the element does not exist, it throws an `IllegalArgumentException`)
- `find(String str)`: returns `str` if it is in the array, otherwise `null`
- `getAllItems()`: returns an array of the stored `String` values (be careful, the internal state of our object should not escape!)

Prepare tests for the methods above!

http://tutorials.jenkov.com/java-collections/list.html#create-a-list

```
import java.util.Arrays;
import java.util.Collections;
import java.util.List;

public class UnmodifiableStringArray {
    private final String[] strArr;

    public UnmodifiableStringArray(String[] strArr) {
       this.strArr = Arrays.copyOf(strArr,strArr.length); //defensive copying
}

    public UnmodifiableStringArray empty() {
        return new UnmodifiableStringArray(new String[0]);
    }

    public String get(int index) {
        if(index < 0 || index >= strArr.length) {
            throw new IllegalArgumentException();
        }
        return strArr[index];
    }

    public String find(String str) {
        for (String string: strArr) {
            if(string.equals(str)) {
                return string;
            }
        }
        return null;
    }

    public boolean contains(String str) {
        for (String string: strArr) {
            if(string.equals(str)) {
                return true;
            }
        }
        return false;
    }

    public String[] getAllItems() {
        return Arrays.copyOf(strArr,strArr.length); //defensive copying
    }

    public int size() {
        return strArr.length;
    }

    public int maxLength() {
        int maxl = -1;
        for (String string:strArr) {
            if(string.length() > maxl) {
                maxl = string.length();
            }
        }
        return maxl;
    }

    public int minLength() {
        int minl = Integer.MAX_VALUE;
        for (String string:strArr) {
            if(string.length() < minl) {
                minl = string.length();
            }
        }
        return minl;
    }

    public int allLength() {
        int sum = 0;
        for (String string:strArr) {
            sum += string.length();
        }
        return sum;
    }
}


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

public class UnmodifiableStringArrayTest {
    private UnmodifiableStringArray usa;

    @Before
    public void init() {
        if(usa == null) {
            usa = new UnmodifiableStringArray(new
String[]{"abc","def","x","aaaaaaa","sw","Longest String"});
        }
    }

    @Test
    public void get_Test() {
        assertEquals("x",usa.get(2));
    }

    @Test(expected = IllegalArgumentException.class)
    public void illegal_Get_Test() {
        usa.get(-2);
    }

    @Test
    public void defensive_Copy_Test() {
        String[] str = {"a","b","c","d"};
        UnmodifiableStringArray usa = new UnmodifiableStringArray(str);
        str[0] = "f";
        assertEquals("a",usa.get(0));

        String[] usaItems = usa.getAllItems();
        usaItems[2] = "Nonsense";
        assertNotEquals("Nonsense", usa.get(2));
        assertEquals("c", usa.get(2));
    }


    @Test
    public void size_Adding_Test() {
        assertEquals(30, usa.allLength());
    }

    @Test
    public void test_Contains() {
        assertFalse(usa.contains("USA"));
        assertTrue(usa.contains("Longest String"));
        //assertTrue(usa.contains("LongestString"));
        //assertEquals(true,usa.contains("LongestString"));
    }//both could be tested w/ assertEq(true,...) maybe a better approach

}
```

Note: “Defensive copying is a technique which mitigates the negative effects caused by unintentional (or intentional) modifications of shared objects. As the title indicates, instead of sharing the original object, we share a copy of it and thus any modification made to the copy will not affect the original object”.

https://beginnersbook.com/2013/12/linkedlist-in-java-with-example/
https://www.javatpoint.com/java-list
https://www.w3schools.com/java/java_arraylist.asp