Csc241 Program 1
Spring 2018

Some points about writing this program

1. To say it succinctly: **you should be the sole author of this program**. You must draw a line when helping or getting help from another student. **Do not** give other students your code. **Do not** ask other students for their code. **Do not** use code of students who have taken this course previously.
2. Your name should be within the documentation of all source files.
3. Organize your code well.
4. When the assignment is returned I will present my solution, but will not make it available to the class as source code.

Archive (in any format) your **entire** project directory and submit it through D2L. The instructions are on the **D2L Submission** page.

This programming assignment involves adding functionality to the `SearchTreeSet` implementation found in the `SetMap` project. Specifically you will edit the file:

`util.SearchTreeSet`

and write Java-correct implementations of these methods:

- `last`
- `pollLast`
- `tailSet`
- `ceiling`

The goal is to make these methods **behave exactly** like they would for a Java `TreeSet`. Additional programming requirements:

- the `last`, `pollLast`, and `ceiling` methods should just take one or two passes down the tree. For a "well-balanced" tree, the time should be O(log(n)).
- the `tailSet` method, although O(n), should be done as efficiently as possible.

Skeleton Program

Add the following starter code **at the end of the class**, fix the imports (for SortedSet), and set **YOUR NAME**:

util.SearchTreeSet

```
//...
public class SearchTreeSet<E> extends NavSetAdapter<E> {
  // ...
  // Please add the following starter code AT THE END of this class
  // and fix imports

  //---------------------- added by YOUR NAME (please set this)
  @Override
  public E last() {
    return null;
  }
```

```
  @Override
  public E pollLast() {
    return null;
  }

  @Override
  public SortedSet<E> tailSet(E fromElement) {
    SortedSet<E> set = new SearchTreeSet<>();
    tailSet(root, fromElement, set);
    return set;
  }

  private void tailSet(Node<E> n, E elt, SortedSet<E> set) {
  }

  @Override
  public E ceiling(E elt) {
    return null;
  }
}
```

Write Comparison/Test Programs

Create a new package in `SetMap`, like `prog1`, and for each member create a main class something like this:

prog1.TestPollLast

```
package prog1;

import java.util.Arrays;
import java.util.Random;
import java.util.TreeSet;

import util.SearchTreeSet;

public class TestPollLast {

  public static void main(String[] args) {
    TreeSet<Integer> java_tree = new TreeSet<>();
    SearchTreeSet<Integer> my_tree = new SearchTreeSet<>();

    // create a fixed array of entries
    Integer entries_fixed[] = {
      2, 3, 1
    };

    // or make an array of random entries
    int rand_size = 10;
    Random rand = new Random();
    Integer entries_random[] = new Integer[rand_size];
    for(int i = 0; i < rand_size; ++i) {
      entries_random[i] = rand.nextInt(rand_size);
    }

    Integer[] entries = entries_random;  // or = entries_fixed
```

```
    System.out.println("------------ java:");
    java_tree.addAll(Arrays.asList(entries));
    System.out.println(java_tree);
    System.out.println("size before = " + java_tree.size());
    Integer last = java_tree.pollLast();
    System.out.println("size after =  " + java_tree.size());
    System.out.println("last = " + last);
    System.out.println(java_tree);

    System.out.println("------------ me:");
    my_tree.addAll(Arrays.asList(entries));
    my_tree.reverseInorderOutput();
  }
}
```

Look at JavaDoc Documentation

For the Java operations, select its usage, right-click and run **Show JavaDoc** to see what Java says about the behavior. For example to see the JavaDoc of `pollLast`, work with the selection:

```
... java_tree.pollLast();
```
