Computer Science 228
Project 5
Splay Trees and Maps

Due: 11:59 pm, Friday, April 24

1 Project Overview

This project involves three important concepts covered in class: sets, splay trees, and maps.

- A ***set*** is a collection of distinct objects.
- A ***splay tree*** is a special kind of self-adjusting tree. It can represent a set of n elements with a natural ordering so that the amortized running time per operation (i.e., the time averaged over a worst-case sequence of operations) is $O(\log n)$[1].
- A ***map*** is an object that maps a finite set of ***keys*** to a collection of ***values***. Each key can map to at most one value, and a map cannot contain duplicate keys. Maps correspond to the mathematical concept of a ***function***. An example of a map is the function that maps the set of student ID numbers (integers) to student names (strings).

In this project, you will gain a deeper understanding of these notions by writing the following two classes.

`SplayTreeSet`: An implementation of sets based on splay trees.

`SplayTreeMap`: An implementation of maps, where the key-value pairs are stored as a `SplayTreeSet`.

We will provide you with a `Node` class and a `MapEntry` class — to represent tree nodes and key-value pairs in a map, respectively —, neither of which you are allowed to modify. We will also provide you with templates for `SplayTreeSet` and `SplayTreeMap`. You may add new instance variables and methods to these two classes, but you cannot rename or remove any existing variables or methods, or change any of these variables and methods from `public` to `protected` (or `private`), or vice versa.

This project is worth 200 points

[1] See the lecture notes `splay-tree.pptx` posted on Blackboard Learn under the April 6 folder.

2 SplayTreeSet

The SplayTreeSet class implements a set of elements with a natural ordering using a splay tree. We disallow null elements; any attempt to add a null element should result in a NullPointerException. The SplayTreeSet class has the following signature.

```
public class SplayTreeSet<E extends Comparable<? super E>>
   extends AbstractSet<E>
```

SplayTreeSet must override add(), contains(), remove(), successor(), size(), and iterator(). You must also override toString(), as explained next.

Overriding toString()

The toString() method should be overridden to display the current configuration of the underlying splay tree for the set. This should be done according to the following rules:

- Every node of the tree occupies a separate line with only its data on it.
- The data stored at a node at depth d is printed with indentation $4d$ (i.e., preceded by $4d$ blanks).
- Start at the root (at depth 0) and display the nodes in a preorder traversal. More specifically, suppose a node x is shown at line ℓ. Then, starting at line $\ell + 1$,
  - recursively print all nodes in the left subtree (if any) of x;
  - recursively print all nodes in the right subtree (if any) of x.
- If a node has a left child but no right child, print its right child as null.
- If a node has a right child but no left child, print its left child as null.
- If a node is a leaf, print it with no further recursion.

Figure 1 shows a splay tree with 12 nodes. Figure 2 shows the output that should be generated by calling the toString() and System.out.println().

3 SplayTreeMap

The SplayTreeMap class uses a splay tree to implement a mapping between a set of keys with a natural ordering and a collection of values. We disallow null keys or values; any attempt to add a null key or value should result in a NullPointerException. The SplayTreeMap class has the following signature.

```
public class SplayTreeMap<K extends Comparable<? super K>, V>
```

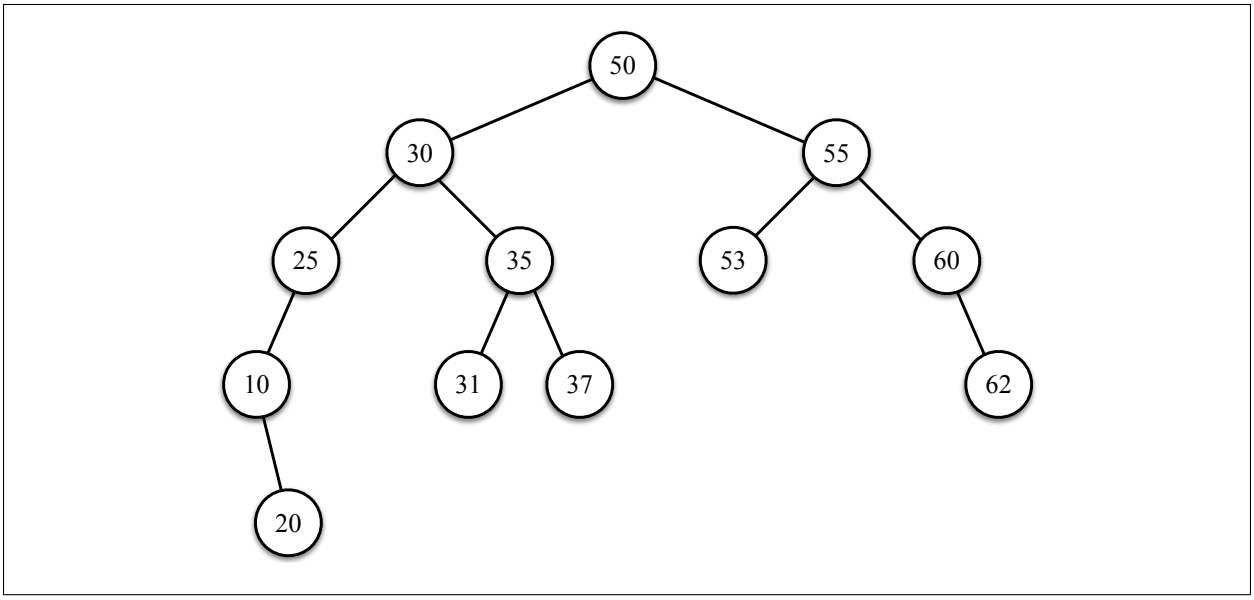

Figure 1: A splay tree.

```
50
    30
        25
            10
                null
                20
            null
        35
            31
            37
    55
        53
        60
            null
            62
```

Figure 2: The result of `toString()` for the tree of Figure 1.

SplayTreeMap has the following methods.

public V put(K key, V value)
Associates value with key in this map. Returns the previous value associated with key, or null if there was no mapping for key.

public V get(K key)
Returns the value to which key is mapped, or null if this map contains no mapping for key.

public V remove(K key)
Removes the mapping for key from this map if it is present. Returns the previous value associated with key, or null if there was no mapping for key.

public boolean containsKey(K key)
Returns true if this map contains a mapping for key; otherwise, it returns false.

public int size()
Returns the number of key-value mappings in this map.

public SplayTreeSet<K> keySet()
Returns a SplayTreeSet storing the keys (not the values) contained in this map.

Example. Suppose this map consists of the following (key, value) pairs: $(10, \texttt{Carol})$, $(21, \texttt{Bill})$, $(45, \texttt{Carol})$, $(81, \texttt{Alice})$, $(95, \texttt{Bill})$. Then, the SplayTreeSet returned should consist of $10, 21, 45, 81, 91$.

public ArrayList<V> values()
Returns an ArrayList storing the values contained in this map. Note that there may be duplicate values. The ArrayList should contain the values in ascending order of their corresponding keys.

Example. Suppose this map consists of the following (key, value) pairs: $(10, \texttt{Carol})$, $(21, \texttt{Bill})$, $(45, \texttt{Carol})$, $(81, \texttt{Alice})$, $(95, \texttt{Bill})$. Then, the ArrayList returned should consist of the strings Carol, Bill, Carol, Alice, Bill, in that order.

Note. Both keySet() and values() should be implemented by iterating through the SplayTreeSet that represents the map.

4 Submission

Write your classes in the edu.iastate.cs228.hw5 package. Turn in the zip file, not your class files. Please follow the guidelines posted under Documents & Links on Blackboard Learn. Also, follow project clarifications on Blackboard. Include the Javadoc tag @author in each class source file. Your zip file should be named Firstname_Lastname_HW5.zip.

Note. Although the official due date is 11:59 pm, Friday, April 24, you may submit the assignment without penalty until 11:59 pm, Wednesday, April 29, 2015.