ÇANKAYA UNIVERSITY

SENG 201 Data and Game Structures

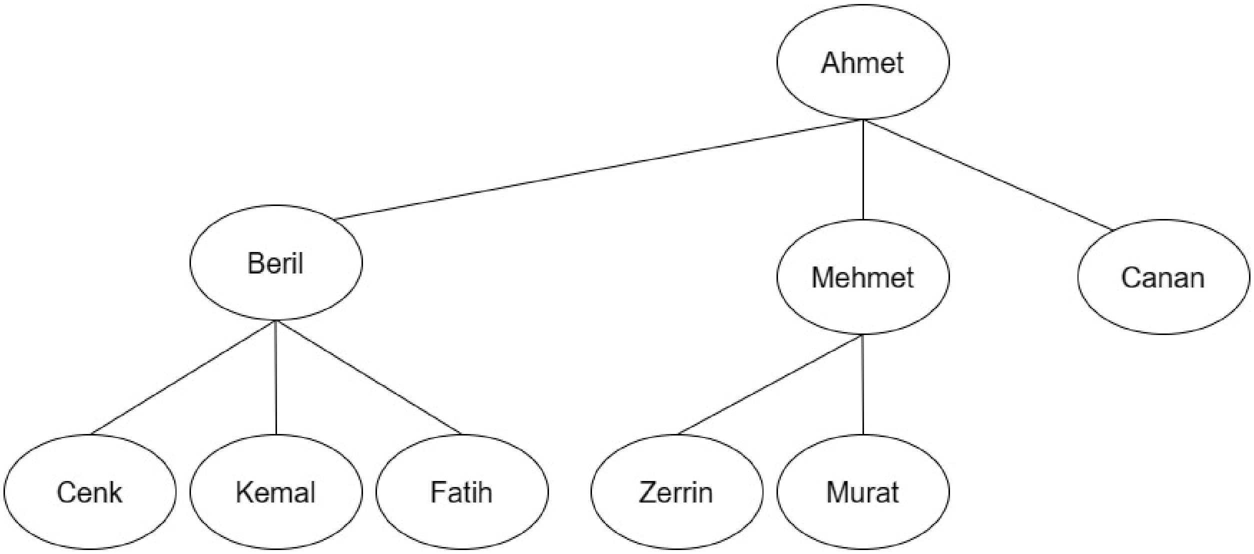
Lab Assignment 9

In this exercise you will practice **3-Heap** implementation.

PROBLEM

You will implement a **Minimum Oriented 3-Heap** data structure for **String** values (MinString3Heap).

3-Heap is a data structure where each node can have a maximum of 3 children. Other than that, the min-heap rules apply: Parent node is smaller than it's children. An example of a Min-3Heap is given below:



0	1	2	3	4	5	6	7	8	9
	Ahmet	Beril	Mehmet	Canan	Cenk	Kemal	Fatih	Zerrin	Murat

Figure 1 - An example of a Minimum Oriented 3-Heap for String values and how it's kept on array.

TASKS

Copy the following java code as the starting template for your *MinString3Heap* implementation.

```java
public class MinString3Heap {
  private String[] values;
  private int N = 0;

  public MinString3Heap() {
    this.values = new String[20];
  }

  public void insert( String value ) {
    // You will implement this method
    // Method takes a String and inserts into the 3-Heap
  }

  private void swimRecursive( int index ) {
    // You will implement this method RECURSIVELY!
    // Method takes an item index and swims the item up in the tree
  }

  public String removeMin() {
    // You will implement this method
    // Method removes and returns the minimum element from the tree
  }

  private void sinkRecursive( int index ) {
    // You will implement this method RECURSIVELY!
    // Method takes an item index and sinks the item down in the tree
  }

  public void delete( int index ) {
    // You will implement this method
    // Method takes an item index and removes the item from the tree
  }

  public void update( int index, String newValue ) {
    // You will implement this method
    // Method takes an item index, a String and updates the item’s value
  }

  public void print() {
    for (int i=1, _N=N; i<=_N; i++) {
      System.out.println( removeMin() );
    }
  }
}
```

The MinString3Heap holds at most 20 String values in an array and keeps the array as a “`Complete Tree`”. You will implement the methods explained in the code following the Heap rules and keeping **each method in `O(logN)` time complexity**.

TESTING

The “`print`” method for the heap is already given for you in the template. Test your implementation with the given Main method below in a Main.java file, where you can also add/remove/change strings to test with different inputs.

```java
public static void main(String[] args) {
  MinString3Heap heap = new MinString3Heap();

  heap.insert("Kemal");
  heap.insert("Zerrin");
  heap.insert("Ahmet");
  heap.insert("Beril");
  heap.insert("Canan");
  heap.insert("Hikmet");
  heap.insert("Okan");

  heap.update(2, "Mehmet");
  heap.update(4, "Fatih");

   heap.delete(2);

  heap.print(); // should print :
               // Ahmet
               // Fatih
               // Kemal
               // Mehmet
               // Okan
               // Zerrin
}
```

You should submit one zip file name as **“YourNameSurname_Lab9.zip”** and it should contain all the java files you created.