Objective(s):

a. To understand heap insertion and removal mechanism.

b. To understand the relationship between heap and priority queue.

Task 1: Create package named solutions.pack9_Heap with package code inside. Study MyMinHeap's insert(int d) and remove() methods. Test L9_PQ's task1().

กำหนด **MAX_SIZE = 6**;

```
import code.*;
public class L9_PQ_Main {
  static ArrayList<Integer> least3;

  public static void main(String[] args) {
    println("-task1---");
    task1();
    // println("-task2---");
    // task2();
  }
  static void task1() {
    least3 = new ArrayList<>();
    MyMinHeap heap = new MyMinHeap();
    heap.insert(11);    heap.insert(15);
    heap.insert(16);    heap.insert(13);
    heap.insert(17);    heap.insert(18);
    println("heap structure is " + heap);
    least3.add(heap.remove());
    least3.add(heap.remove());
    least3.add(heap.remove());
    println("least 3 value is " + least3);
  }
  static void task2() {
    least3 = new ArrayList<>();
    MyPriorityQueue pq = new MyPriorityQueue();
    pq.enqueue(11);    pq.enqueue(15);
    pq.enqueue(16);    pq.enqueue(13);
    pq.enqueue(17);    pq.enqueue(18);
    pq.enqueue(19);   // <-- isFull() is true … discard
    println("pq structure is " + pq);
    least3.add(pq.dequeue());
    least3.add(pq.dequeue());
    least3.add(pq.dequeue());
    println("least 3 value is " + least3);
  }
}
```

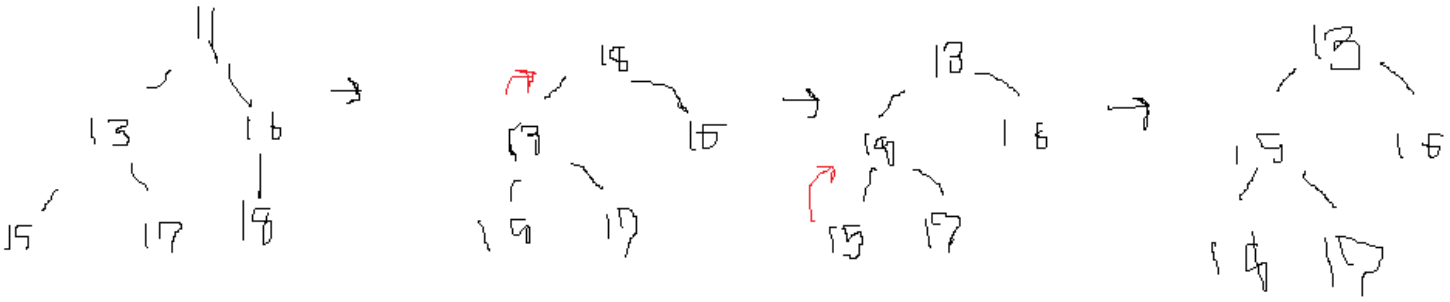
Task 2:

2.1 Given (create one yourself) an abstract class MyQueueInterface.java, implement MyPriorityQueue.java from MyMinHeap's capabilities.

```
public interface MyQueueInterface {
    public void enqueue(int d);
    public int dequeue();
    public int front();
    public boolean isFull();
    public boolean isEmpty();
}
```

Note that MyMinHeap's rudimentary implementation was that it does not check whether the heap is full or empty when insert or remove. Make sure that your enqueue() and dequeue() do not cause such mistakes.

2.2 draw / write the heap snapshot (up to the original sized i.e. including the deleted data) during each dequeue() was performed.



Submission: MyMinHeap_XXYYYY.java and MyPriorityQueue_XXYYYY.java and this pdf.

Bonus Material (next page)

Java collections provide a PriorityQueue class. It is an unbounded priority queue based on a priority heap. The elements of the priority queue are ordered according to their natural ordering, or by a Comparator provided at queue construction time, depending on which constructor is used. [1]

An example of working with a priority queue is as follows.

```
static void demo_priorityQueue() {

  PriorityQueue<Employee> pq = new PriorityQueue<>(
                               (e1,e2) -> Integer.compare(e1.salary,e2.salary));

  List<Employee> list = Arrays.asList(new Employee("Yindee", 2000),
                                      new Employee("Preeda",1500),
                                      new Employee("Pramote", 3000));
  pq.addAll(list);
  System.out.println(pq);
  // [Emp Preeda(1500), Emp Yindee(2000), Emp Pramote(3000)]
}
```

Due date: TBA

[1] https://docs.oracle.com/javase/8/docs/api/java/util/PriorityQueue.html