Objective(s):

a. To understand heap insertion and removal mechanism.

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

Task 1: MyMinHeap

Implement MyMinHeap.Java, inside package \Lab09\pack.

implement insert(int d) and int remove() methods. (MAX_SIZE = 100)

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

  public static void main(String[] args) {
    // println("-task1---");
    // task_1();
    // println("-task2---");
    // task_2();
  }
  static void task_1 () {
    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 strucutre is " + heap);
    least3.add(heap.remove());
    least3.add(heap.remove());
    least3.add(heap.remove());
    println("least 3 value is " + least3);
  }
  static void task_2() {
    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);
  }
}
```

Implement boolean isFull(), boolean isEmpty() and int peek() for task_2()

Task 2: Given an abstract class MyQueueInterface.java, implement MyPriorityQueue.java from MyMinHeap's capabilities.

```
package code;

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

Task 3: draw / write the heap snapshot during each dequeue() was performed.

```
-task2---
pq structure is [11,13,16,15,17,18,19,]
heap snapshot (cur_size = 6) [13,19,16,15,17,18,]
heap snapshot (cur_size = 6) [13,15,16,19,17,18,]
heap snapshot (cur_size = 5) [15,18,16,19,17,]
heap snapshot (cur_size = 5) [15,17,16,19,18,]
heap snapshot (cur_size = 4) [16,17,18,19,]
least 3 value is [11, 13, 15]
```

Task 3: Comparator for Priority Queue

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 task_3() {

  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)]
}
```

Task 4: Challenge Example

1046. Last Stone Weight Solved

Easy Topics Companies Hint

You are given an array of integers `stones` where `stones[i]` is the weight of the i^{th} stone.

We are playing a game with the stones. On each turn, we choose the **heaviest two stones** and smash them together. Suppose the heaviest two stones have weights `x` and `y` with `x <= y`. The result of this smash is:

- If `x == y`, both stones are destroyed, and
- If `x != y`, the stone of weight `x` is destroyed, and the stone of weight `y` has new weight `y - x`.

At the end of the game, there is **at most one** stone left.

Return *the weight of the last remaining stone*. If there are no stones left, return `0`.

Example 1:

```
Input: stones = [2,7,4,1,8,1]
Output: 1
Explanation:
We combine 7 and 8 to get 1 so the array converts to [2,4,1,1,1] then,
we combine 2 and 4 to get 2 so the array converts to [2,1,1,1] then,
we combine 2 and 1 to get 1 so the array converts to [1,1,1] then,
we combine 1 and 1 to get 0 so the array converts to [1] then that's the
value of the last stone.
```

```
static void task_4() {

    int lastStoneWeight = 0;
    int[] stones = {2,7,4,1,8,1};
    /* your code */
    System.out.println(lastStoneWeight);
}
```

Submission: MyMinHeap_XXYYYY, MyPriorityQueue_XXYYYY.java and this pdf.

Due date: TBA

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