PRAKTIKUM 24
PRIORITY QUEUE

A. TUJUAN

Mahasiswa diharapkan mampu :

1. Memahami konsep Priority Queue
2. Memahami implementasi dari Priority Queue
3. Memahami Representasi dan alternative dari model penyimpan antrian

B. DASAR TEORI

Dalam antrian yang telah dibahas di atas, semua elemen yang masuk dalam antrian dianggap mempunyai prioritas yang sama, sehingga elemen yang masuk lebih dahulu akan diproses lebih dahulu. Dalam praktek, elemen-elemen yang akan masuk dalam suatu antrian ada yang dikatakan mempunyai prioritas yang lebih tinggi dibanding yang lain. Antrian yang demikian ini disebut dengan antrian berprioritas (*priority queue*).

Dalam antrian berprioritas, setiap elemenn yang akan masuk dalam antrian sudah ditentukan lebih dahulu prioritasnya. Dalam hal ini berlaku dua ketentuan, yaitu:

1. Elemen-elemen yang mempunyai prioritas lebih tinggi akan diproses lebih dahulu.
2. Dua elemen yang mempunyai prioritas sama akan dikerjakan sesuai dengan urutan pada saat kedua elemen ini masuk dalam antrian.

Dengan memperhatikan kedua ketentuan di atas, akan berlaku ketentuan bahwa elemen yang mempunyai prioritas lebih tinggi akan dikerjakan lebih dahulu dibanding elemen yang mempunyai prioritas lebih rendah, meskipun elemen yang berprioritas tinggi masuknya sesudah elemen yang berprioritas rendah. Salah satu contoh antrian berprioritas ini adalah pada sistem berbagi waktu (time-sharing system) dimana program yang mempunyai prioritas tinggi akan dikerjakan lebih dahulu dan program-program yang berprioritas sama akan membentuk antrian biasa.

C. TUGAS PENDAHULUAN

Buatlah resume 1 halaman mengenai **Priority Queue** dan berikan penjelasannya.!

D. PERCOBAAN

D.1 Class PriorityQueue di Collection

Percobaan 1 : Memahami Penggunaan dari class PriorityQueue

```
import java.util.*;

public class PriorityQueueDemo {
    public static void main(String[] args) {
        PriorityQueue<String> stringQueue;
        stringQueue = new PriorityQueue<String>();
        stringQueue.add("ab");
        stringQueue.add("abcd");
        stringQueue.add("abc");
        stringQueue.add("a");

        while (stringQueue.size() > 0) {
            System.out.println(stringQueue.remove());
        }
    }
}
```

Percobaan 2 : Memahami Penggunaan dari class PriorityQueue dan data yang tersimpan dalam objek PriorityQueue mengimplementasikan interface Comparator.

```
import java.util.Comparator;
import java.util.PriorityQueue;

public class PQueueTest {
    public static void main(String[] args) {
        PriorityQueue<Integer> pQueue = new PriorityQueue<Integer>(10, new
Comparator<Integer>() {
            public int compare(Integer int1, Integer int2) {
                boolean flag1 = isPrime(int1);
                boolean flag2 = isPrime(int2);
                if (flag1 == flag2) {
                    return int1.compareTo(int2);
                } else if (flag1) {
                    return -1;
                } else if (flag2) {
                    return 1;
                }
                return 0;
            }
        });
        pQueue.add(1);
        pQueue.add(5);
        pQueue.add(6);
        pQueue.add(4);
        pQueue.add(2);
        pQueue.add(9);
        pQueue.add(7);
        pQueue.add(8);
        pQueue.add(10);
        pQueue.add(3);
```

```
        while (true) {
            Integer head = pQueue.poll();
            if (head == null) {
                break;
            }
            System.out.print(head + " <-- ");
        }
    }

    /**
     *
     * @param n
     * @return
     */
    public static boolean isPrime(int n) {
        if (n <= 1) {
            return false;
        }
        if (n == 2) {
            return true;
        }
        if (n % 2 == 0) {
            return false;
        }
        long m = (long) Math.sqrt(n);
        for (long i = 3; i <= m; i += 2) {
            if (n % i == 0) {
                return false;
            }
        }
        return true;
    }
}
```

Percobaan 3. Memahami Penggunaan dari class PriorityQueue dan data yang tersimpan adalah data Product.

```
import java.util.Comparator;
import java.util.PriorityQueue;

enum ProductQuality {
    High, Medium, Low
}

class Product implements Comparable<Product> {

    String name;
    ProductQuality priority;

    Product(String str, ProductQuality pri) {
        name = str;
        priority = pri;
    }

    public int compareTo(Product msg2) {
```

```
        return priority.compareTo(msg2.priority);
    }
}

class MessageComparator implements Comparator<Product> {

    public int compare(Product msg1, Product msg2) {
        return msg2.priority.compareTo(msg1.priority);
    }
}

public class Main {

    public static void main(String args[]) {
        PriorityQueue<Product> pq = new PriorityQueue<Product>(3);
        pq.add(new Product("A", ProductQuality.Low));
        pq.add(new Product("B", ProductQuality.High));
        pq.add(new Product("C", ProductQuality.Medium));
        Product m;
        while ((m = pq.poll()) != null) {
            System.out.println(m.name + " Priority: " + m.priority);
        }
        PriorityQueue<Product>  pqRev  =  new  PriorityQueue<Product>(3,  new
MessageComparator());
        pqRev.add(new Product("D", ProductQuality.Low));
        pqRev.add(new Product("E", ProductQuality.High));
        pqRev.add(new Product("F", ProductQuality.Medium));
        while ((m = pqRev.poll()) != null) {
            System.out.println(m.name + " Priority: " + m.priority);
        }
    }
}
```

D.2 Membuat Class PriorityQueue sendiri

Untuk membuat class PriorityQueue sendiri buatlah interface PQueue

Tabel 24.1 Interface PriorityQueue

Interface PQueue	
boolean isEmpty()	Mengembalikan nilai true jika queue kosong dan false jika queue terdapat minimal satu elemen
T peek()	Mengembalikan elemen dengan prioritas tertinggi. Jika queue kosong melempar exception yaitu: throws NoSuchElementException.
T pop()	Menghapus elemen di depan queue dan mengembalikan nilai berdasarkan prioritas tertinggi. Jika queue kosong maka melempar exception yaitu : NoSuchElementException.
void push(T item)	Menyisipkan item di queue
int size()	Mengembalikan jumlah elemen dari queue

Membuat Class HeapPQueue , dengan mengurutkan data menggunakan Comparator atau Comparable

```
import java.util.Collections;
import java.util.Comparator;
import java.util.LinkedList;

public class HeapPQueue<T extends Comparable<? super T>> implements PQueue<T> {

  private LinkedList<T> qlist = null;
  private Comparator<T> comparator = null ;

  public HeapPQueue()
  {
    qlist= new LinkedList<T>();
  }

  public HeapPQueue(Comparator comp)
  {
    qlist= new LinkedList<T>();
    comparator = comp ;
  }

   @Override
   public boolean isEmpty() {
          throw new UnsupportedOperationException("Not supported yet.");
   }

   @Override
   public T peek() {
          throw new UnsupportedOperationException("Not supported yet.");
   }

   @Override
   public T pop() {
          throw new UnsupportedOperationException("Not supported yet.");
   }

   @Override
   public void push(T item) {
          throw new UnsupportedOperationException("Not supported yet.");
   }

   @Override
   public int size() {
          throw new UnsupportedOperationException("Not supported yet.");
   }
}
```

Selanjutnya lakukan pengujian terhadap Class **HeapPQueue** menggunakan studi kasus 1-3 yang

Studi Kasus 1

```
public class HeapPQueueDemo1 {
   public static void main(String[] args) {
      Comparator comparator = Collections.reverseOrder();
      HeapPQueue<String> pq = new HeapPQueue<String>(comparator);
      pq.push("C");
      pq.push("A");
      pq.push("B");
      System.out.println(pq.pop());
      System.out.println(pq);
   }
}
```

Studi Kasus 2

```
public class HeapPQueueDemo2 {
   public static void main(String[] args) {

      HeapPQueue pq = new HeapPQueue(new PrimaComparator());
      pq.push(1);
      pq.push(5);
      pq.push(6);
      pq.push(4);
      pq.push(2);
      pq.push(9);
      pq.push(7);
      pq.push(8);
      pq.push(10);
      pq.push(3);

      while(!pq.isEmpty())
         System.out.print(pq.pop()+" ");
   }
}
```

Studi Kasus 3

```
public class HeapPQueueDemo3 {

   public static void main(String[] args) {

      HeapPQueue<Product> pq = new HeapPQueue<Product>();
      pq.push(new Product("A", ProductQuality.Low));
      pq.push(new Product("B", ProductQuality.High));
      pq.push(new Product("C", ProductQuality.Medium));

      while (!pq.isEmpty()){
         System.out.println(pq.pop().toString());
      }
   }
}
```

E. LATIHAN

Sebuah bank N, terdapat aturan antrian, nasabah dengan saldo besar akan mendapatkan prioritas tinggi. Terdapat 5 nasabah bank siap untuk memasuki antrian. Berikut adalah informasi nasabah dan tabungan dari nasabah tersebut. Bagaimana output dari antrian ? Implementasikan pada program yang sudah dibuat pada point D.2

No	Nama	Jumlah Tabungan
1	Andi	2.000.000
2	Budi	10.000.000
3	Cahya	15.000.000
4	Dinda	12.000.000
5	Rudi	3.000.000

F. LAPORAN RESMI

Kerjakan hasil percobaan(D) dan latihan(E) di atas dan tambahkan analisa.