**RESPONSI PRAKTIKUM**

**PEMROGRAMAN BERORIENTASI OBJEK**

**RABU, 27 Maret 2024**

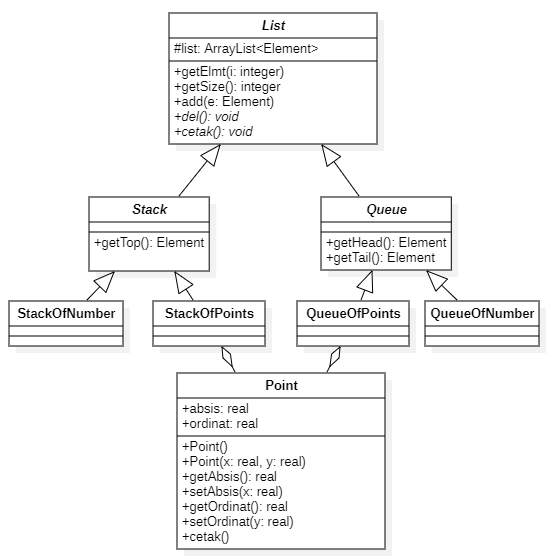
Sebuah struktur data linear terdiri atas 2 jenis, yaitu Stack dan Queue.

1. Stack (Last In First Out)
   1. Stack ditandai dengan TOP (elemen yang berada di tumpukan paling atas atau elemen terakhir).
   2. Penambahan elemen baru diletakkan setelah TOP.
   3. Penghapusan elemen dilakukan pada TOP.
   4. Cetak elemen dimulai dari TOP hingga elemen pertama.
2. Queue (First In First Out)
   1. Queue ditandai dengan HEAD (elemen pertama) dan TAIL (elemen terakhir).
   2. Penambahan elemen baru diletakkan setelah TAIL.
   3. Pengahpusan elemen dilakukan pada HEAD.
   4. Cetak elemen dimulai dari HEAD hingga TAIL.

Setiap jenis tersebut memiliki 2 jenis lagi sesuai dengan tipe elemennya. StackOfNumber dan QueueOfNumber memiliki elemen bertipe integer, sedangkan StackOfPoints dan QueueOfPoints memiliki elemen bertipe Point.

Buatlah program dalam Bahasa Java menggunakan paradigma berorientasi objek dengan menerapkan konsep enkapsulasi, pewarisan, *overloading*, dan *overriding* untuk kasus struktur data linear tersebut. Maksimalkan *reusability code*, sehingga method yang isinya sama tidak harus diimplementasi berulang kali, sebagai contoh method add(e: Elmt) memiliki implementasi yang sama pada Stack maupun Queue (sama-sama menambahkan elemen setelah elemen terakhir, sehingga hanya perlu diimplementasi satu kali). Ukuran Stack dan Queue dinamis. Untuk implementasi dapat menggunakan ArrayList.

Berikut adalah sketsa class diagram untuk kasus tersebu. Tambahkan konstruktor dan method lain yang diperlukan. Tambahkan asumsi lain jika diperlukan.



Berikut adalah contoh main program dan hasil outputnya

|  |  |
| --- | --- |
| public class MList {  public static void main(String[] args){  try {  System.out.println("-----StackOfNumber S1-----");  StackOfNumber S1 = new StackOfNumber();  S1.add(16);  S1.add(23);  S1.add(56);  S1.del();  S1.add(99);  S1.cetak();  System.out.println("TOP Element : " + S1.getTop());    System.out.println("-----StackOfPoints S2-----");  StackOfPoints S2 = new StackOfPoints();  S2.add(new Point(4,5));  S2.add(new Point(9,7));  S2.add(new Point(4,5));  S2.del();  S2.add(new Point(10,5));  S2.cetak();  System.out.print("TOP Element : "); S2.getTop().cetak();    System.out.println("-----QueueOfNumber Q1-----");  QueueOfNumber Q1 = new QueueOfNumber();  Q1.add(12);  Q1.add(23);  Q1.add(56);  Q1.del();  Q1.add(99);  Q1.cetak();  System.out.println("HEAD Element : " + Q1.getHead());  System.out.println("TAIL Element : " + Q1.getTail());    System.out.println("-----QueueOfPoints Q2-----");  QueueOfPoints Q2 = new QueueOfPoints();  Q2.add(new Point(34,89));  Q2.add(new Point(-76,90));  Q2.add(new Point(0,0));  Q2.del();  Q2.add(new Point(-10,-10));  Q2.cetak();  System.out.println("HEAD Element : ");  Q2.getHead().cetak();  System.out.println("TAIL Element : ");  Q2.getTail().cetak();  } catch (Exception ex) {  System.out.println(ex.getMessage());  }  }  } | -----StackOfNumber S1-----  99  23  16  TOP Element : 99  -----StackOfPoints S2-----  (10.0,5.0)  (9.0,7.0)  (4.0,5.0)  TOP Element : (10.0,5.0)  -----QueueOfNumber Q1-----  23  56  99  HEAD Element : 23  TAIL Element : 99  -----QueueOfPoints Q2-----  (-76.0,90.0)  (0.0,0.0)  (-10.0,-10.0)  HEAD Element :  (-76.0,90.0)  TAIL Element :  (-10.0,-10.0) |