Nama = Amru Abid Zakly
NIM = 0706223139
Mata Kuliah = Implementasi Struktur Data
Kelas = D3 RPLA-1605
Tugas = Tracing logika pada modul 12 & 13

Modul 12
↳ Kelas Min Heap :

① -> Pada awal program, kelas MinHeap didefinisikan dengan menggunakan ArrayList sebagai wadah untuk menyimpan elemen2 heap.

② -> Konstruktor 'MinHeap()' digunakan untuk menginisialisasi 'elements' dengan menambahkan elemen null pertama ke dalam ArrayList.

③ -> Metode 'add(Comparable newElement)' digunakan untuk menambahkan elemen baru ke dalam heap. Elemen baru ditambahkan di akhir ArrayList kemudian dilakukan operasi memindahkan elemen ke posisi yang sesuai atau percolation up untuk memastikan struktur heap tetap terjaga

④ -> Metode 'peek()' digunakan untuk mengembalikan elemen terkecil (elemen pada indexs 1) dalam heap tanpa menghapusnya.

⑥ -> Metode 'FixHeap()' digunakan untuk memperbaiki struktur heap setelah elemen terkecil dihapus. Root (elemen pada indeks 1) dipromosikan ke posisi yg sesuai dgn membandingkan anak2nya

⑤ -> Metode 'remove()' digunakan untuk menghapus & mengembalikan elemen terkecil dalam heap. Elemen terkecil dihapus dari indeks 1 kemudian dilakukan operasi memindahkan elemen ke posisi yang sesuai atau percolation down untuk memastikan struktur heap tetap terjaga.

⑦ -> Metode 'size()' mengembalikan jumlah elemen dalam heap.

⑧ -> Terdapat beberapa method yaitu 'getLeftChild(int index)', 'getRightChild(int index)', dan 'getParent(int Index)' untuk mendapatkan elemen anak kiri, anak kanan, dan parent dari suatu indeks dalam heap.

Tracing logika program ini menjelaskan bagaimana elemen ditambahkan, dihapus, dan struktur heap dipertahankan melalui operasi percolation up dan percolation down. Hal ini memastikan bahwa elemen dengan nilai terkecil akan selalu berada pada posisi root dari heap yang diimplementasikan.

↳ Kelas WorkOrder:

-> Program dimulai dengan mendefinisikan kelas "WorkOrder" dengan dua variabel, yaitu "priority" bertipe int dan "description" berupa String.

-> Terdapat sebuah konstruktor "WorkOrder" yang menerima dua parameter, yaitu "priority" dan "description", yang digunakan untuk menginisialisasi nilai "priority" dan "description" dari objek "WorkOrder" yang dibuat.

-> Metode "toString()" digunakan untuk mengembalikan representasi String dari objek "WorkOrder". Metode ini menggabungkan nilai "priority" dan "description" dalam bentuk String yang sesuai.

-> Metode "compareTo(Object otherObject)" mengimplementasikan metode dari antarmuka "comparable" yang digunakan untuk membandingkan objek "WorkOrder" dengan objek lainnya. Metode ini mengambil objek lain sebagai parameter dan mengkonversinya menjadi objek "WorkOrder". Kemudian, perbandingan dilakukan berdasarkan nilai "priority". Jika nilai "prority" objek saat ini lebih kecil dari objek lain, akan dikembalikan nilai -1. Jika nilai "priority" objek saat ini lebih besar dari objek lain, akan dikembalikan nilai 1. Jika nilai keduanya sama, akan dikembalikan nilai 0.

Tracing logika program di atas menjelaskan bagaimana objek "WorkOrder" dibuat dengan nilai prioritas dan deskripsi, serta bagaimana objek tsb dapat dibandingkan dengan objek lainnya berdasarkan nilai prioritas mereka.

↳ Kelas HeapDemo (Main):

- Program dimulai dengan mendefinisikan kelas "HeapDemo" yang memiliki method "main" sebagai kode utama program.
- Dalam method "main", objek "MinHeap" dengan nama "q" dibuat menggunakan konstruktor "MinHeap()".
- Berbagai objek "WorkOrder" dibuat dengan prioritas dan deskripsi yang berbeda, lalu ditambahkan ke dalam heap "q" menggunakan method "add()" dari kelas "MinHeap".
- Setelah semua objek ditambahkan, dilakukan penghapusan elemen dari heap "q" dengan menggunakan method "remove()" dari kelas "MinHeap".
- selama ukuran heap "q" masih lebih dari 0, elemen terkecil dihapus dari heap menggunakan metode "remove()" dan ditampilkan menggunakan pernyataan "System.out.println()".
- Proses penghapusan elemen berulang hingga semua elemen di dalam heap "q" telah dihapus dan ditampilkan.

Tracing logika ini menjelaskan bagaimana objek "WorkOrder" dibuat dengan prioritas dan deskripsi yang berbeda, kemudian ditambahkan ke dalam heap "q" menggunakan method "add()". Setelah itu, elemen dihapus dari heap menggunakan metode "remove()" dan ditampilkan satu per satu hingga tidak ada elemen tersisa dalam heap.