

Modul Pemrograman Berbasis Objek (PBO)

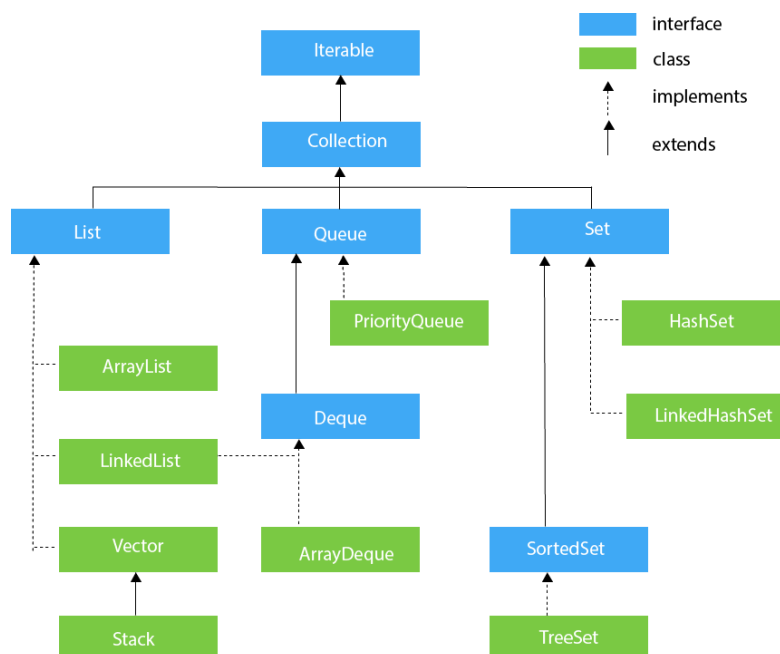
Java Collection

Java Collection Framework adalah library yang berisi berbagai interface dan class yang berfungsi untuk melakukan manipulasi data.

Jika teman-teman ingat di modul array, kita sempat membahas mengenai ArrayList. ArrayList adalah bagian dari java collection yang berfungsi agar kita bisa menggunakan berbagai fungsi array yang tidak dimiliki oleh array tradisional.

Pada java collection, disediakan berbagai macam class yang dapat langsung kita gunakan untuk melakukan berbagai manipulasi data.

Mari kita lihat diagram berikut :



(Sumber : <https://www.javatpoint.com/collections-in-java>)

Pada diagram diatas, terdapat interface collection yang memiliki berbagai method di interface nya. Lalu interface tersebut menurunkan sifat sifat yang dimiliki nya ke anak nya yaitu interface list, queue, dan

set. Pada masing masing interface, mereka kembali menurunkan sifat nya kepada anak anak nya.

Pertama kita akan membahas lebih detail interface list terlebih dahulu

List adalah interface yang mempunyai sifat bahwa data yang berada pada list akan memiliki urutan/posisi

```
data = ["fikri", "hanif", "yoga"]
```

Interface list akan mengenal data diatas secara berurutan. Fikri ada di posisi 0, hanif ada di posisi 1, yoga ada di posisi 2

Interface list mewariskan sifat nya ke class ArrayList, LinkedList, Vector, Stack.

Yang pertama kita akan bahas adalah ArrayList.

ArrayList adalah sebuah collection framework yang digunakan untuk penyimpanan data dan bersifat dynamic Array. Dynamic array sendiri memiliki arti bahwa array yang akan kita buat tidak dibatasi jumlah nya, kita dapat menambahkan atau mengurangi array sebebas yang kita mau.

Mari kita lihat implementasi pada kodingan. Buatlah folder bernama ArrayList dan buat file baru bernama ArrayListPertama.java dan masukan kode berikut

```
import java.util.ArrayList;

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

        ArrayList item = new ArrayList();

        item.add("Fikri");
        item.add(20);
        item.add("Nonton anime");
    }
}
```

```

    int umur = (Integer) item.get(1);

    System.out.println("Item = " + item);
    System.out.println("Nama saya adalah = " + item.get(0));
    System.out.println("Umur saya adalah = " + umur);
    System.out.println("Hobi saya adalah = " + item.get(2));

    item.set(2, "Main game");
    System.out.println("Item terbaru = " + item);

    item.remove(1);
    System.out.println("Item terbaru = " + item);

    ArrayList <Integer> umur2 = new ArrayList <Integer> ();
    umur2.add(17);
    umur2.add(20);
    int angkaUmur = umur2.get(0);
    System.out.println("umur nya adalah = " + angkaUmur);

}
}

```

Mari kita bedah 1 persatu kode diatas

```

import java.util.ArrayList;

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

        ArrayList item = new ArrayList();
    }
}

```

Pertama, kita mengimport library arraylist lalu membuat objek baru dari arraylist bernama item

```

item.add("Fikri");
item.add(20);
item.add("Nonton anime");

```

add adalah method arraylist yang digunakan untuk menambahkan data kedalam variabel item.

```
int umur = (Integer) item.get(1);
```

Secara default, jika kita tidak menentukan tipe data dari arraylist, maka tipe data yang berada dalam arraylist adalah obyek. Pada contoh diatas, kita ingin mengcopy angka 20 pada arraylist item ke dalam variabel umur yang bertipe data integer. Secara default hal ini tidak dapat dilakukan, karena angka 20 bertipe data objek. Penambahan kata (Integer) dilakukan untuk mengubah tipe data objek menjadi integer.

```
System.out.println("Item = " + item);  
System.out.println("Nama saya adalah = " + item.get(0));  
System.out.println("Umur saya adalah = " + umur);  
System.out.println("Hobi saya adalah = " + item.get(2));
```

Pada perintah diatas, kita menampilkan apa saja yang berada pada variabel item. Jika kita hanya menuliskan item saja, maka yang muncul semua data pada variabel item. Sedangkan jika menggunakan method get maka yang muncul data sesuai posisi yang ditentukan. Pada contoh diatas, item.get(0) berarti merujuk item pada posisi 0 yaitu "fikri", maka ketika dipanggil akan menampilkan tulisan "fikri".

```
item.set(2, "Main game");  
System.out.println("Item terbaru = " + item);  
  
item.remove(1);  
System.out.println("Item terbaru = " + item);
```

Method set kita gunakan untuk mengubah data sesuai posisi yang ditentukan. Pada contoh diatas, data yang di ubah adalah data pada posisi 2 yaitu "Nonton anime" menjadi "Main game". Sedangkan, untuk method remove digunakan untuk menghapus data sesuai posisi yang ditentukan, Contoh diatas menghapus data pada posisi 1 yaitu 20

```
ArrayList <Integer> umur2 = new ArrayList <Integer>();
```

Perintah diatas, kita gunakan untuk membuat objek arraylist umur2, tetapi bertipe data integer. Berbeda dengan objek item yang secara default tipe data nya adalah objek, pada objek umur2 sudah ditentukan bahwa tipe data yang akan kita gunakan adalah bertipe data integer. Sehingga data yang dapat dimasukan kedalam umur2 adalah yang bertipe data integer.

```
umur2.add(17);  
umur2.add(20);  
int angkaUmur = umur2.get(0);  
System.out.println("umur nya adalah = " + angkaUmur);
```

Pada contoh diatas, kita dapat langsung mengcopy nilai umur2.get(0) tanpa perlu merubah lagi tipe data kedalam integer. Hal ini disebabkan karena umur2 secara default bertipe data integer.

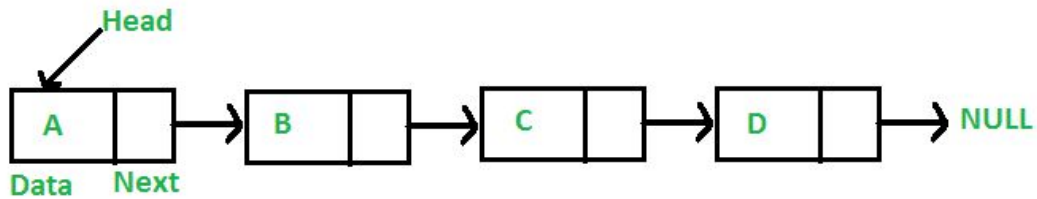
Mari kita coba jalankan kode diatas didalam terminal



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: bash  
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/java/Praktikum 8/List/ArrayList$ java ArrayListPertama  
Item = [Fikri, 20, Nonton anime]  
Nama saya adalah = Fikri  
Umur saya adalah = 20  
Hobi saya adalah = Nonton anime  
Item terbaru = [Fikri, 20, Main game]  
Item terbaru = [Fikri, Main game]  
umur nya adalah = 17  
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/java/Praktikum 8/List/ArrayList$
```

Dapat kita lihat, pertama kita menampilkan semua item, lalu tampil item nya satu persatu. Setelah itu kita menampilkan hasil perubahan dari nonton anime menjadi main game. Dan diatas, kita juga melihat perubahan angka 20 yang dihapus. Terakhir kita melihat hasil dari variabel umur2.

Mari kita beralih ke turunan interface list berikutnya yaitu LinkedList



Sumber gambar :

<https://www.geeksforgeeks.org/implementing-a-linked-list-in-java-using-class/>

Pada gambar diatas adalah contoh penerapan pada linked list. Linked list sendiri adalah algoritma dimana setiap data disimpan pada setiap node dan saling terhubung melalui pointer. Contoh mudah nya, setiap data disimpan pada 1 gerbong kereta, dan tiap data tersebut saling terhubung melalui rantai.

Kita tidak akan membahas lebih detail mengenai algoritma ini dan tidak akan membuat kodingan nya secara manual. Kita akan menggunakan collection class linked list.

Pertama buatlah folder baru bernama linkedlist, dan buatlah file baru bernama LinkedListPertama.java lalu masukan kode dibawah ini

```
import java.util.LinkedList;
class LinkedListPertama {
    public static void main(String [] args) {
        LinkedList <Integer> umur = new LinkedList <Integer>();
        umur.add(17);
        umur.add(20);
        int angkaUmur = umur.get(0);
        System.out.println("umur nya adalah = " + angkaUmur);

        umur.addFirst(19);
        umur.addLast(34);
        System.out.println("data umur = " + umur);

        umur.removeFirst();
        umur.removeLast();
        System.out.println("data umur = " + umur);
    }
}
```

Mari kita bedah kode tersebut satu persatu

```
import java.util.LinkedList;
class LinkedListPertama {
    public static void main(String [] args) {
        LinkedList <Integer> umur = new LinkedList <Integer>();
        umur.add(17);
        umur.add(20);
        int angkaUmur = umur.get(0);
        System.out.println("umur nya adalah = " + angkaUmur);
    }
}
```

Pada contoh kode diatas, kita telah mengimport library linkedlist. Kita membuat objek umur dari librari linked list yang bertipe data integer. Lalu menambahkan data pada objek umur dan menampilkan datanya.

Kita lihat pada contoh kode diatas, ada kemiripan method dengan ArrayList. Kenyataan nya kita bisa menggunakan method add, get, remove, dan set di LinkedList. Alasannya, karena LinkedList adalah turunan dari interface List, dimana method add, get, remove, set, adalah method yang disediakan interface list. Sehingga baik arraylist dan linkedlist dapat menggunakan method yang disediakan interface list.

```
umur.addFirst(19);
umur.addLast(34);
System.out.println("data umur = " + umur);
```

Nah, pada kode diatas, kita menggunakan method yang hanya ada pada LinkedList. Method addFirst digunakan untuk menambahkan data pada posisi ke pertama, sedangkan addLast untuk menambahkan data pada posisi terakhir

umur = [17, 20]

umur.addFirst(19) -> [19,17,20]

umur.addLast(34) -> [19,17,20,34]

```
umur.removeFirst();  
umur.removeLast();  
System.out.println("data umur = " + umur);
```

Begitupun pada method diatas, removeFirst berarti akan menghapus data pada posisi pertama, dan removeLast akan menghapus data pada posisi terakhir.

Kita telah membahas arraylist dan linkedlist, masih terdapat beberapa method lain pada list interface begitupun arraylist dan linkedlist, kita hanya membahas beberapa method saja

Berikut referensi lain yang berkaitan dengan arraylist dan linkedlist :

<https://www.javatpoint.com/java-arraylist>

<https://www.javatpoint.com/java-linkedlist>

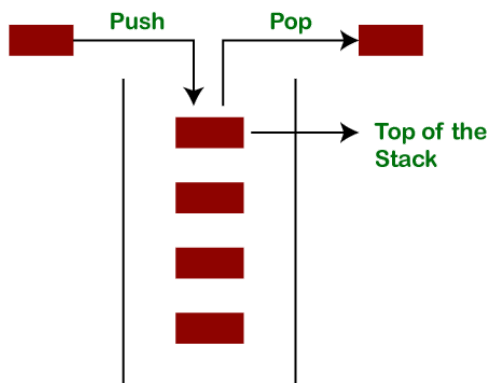
Perbedaan arraylist dan linkedlist :

<https://www.javatpoint.com/difference-between-arraylist-and-linkedlist>

Pembuatan linkedlist secara manual tidak menggunakan collection :

<https://www.geeksforgeeks.org/implementing-a-linked-list-in-java-using-class/>

Kita telah membahas arraylist dan linkedlist, mari kita bahas turunan interface list berikutnya yaitu stack



Sumber : <https://www.javatpoint.com/java-stack>

Seperti yang teman teman lihat diatas, algoritma stack memiliki arti bahwa data diinput secara menumpuk. Contoh mudah nya, saat temen temen menumpuk buku, maka buku yang pertama kali ditumpuk adalah buku yang berada pada urutan paling bawah. Sedangkan buku yang berada pada posisi atas adalah buku yang terakhir kali ditumpuk.

Sedangkan, kaitan nya dengan data adalah data yang pertama kali di input akan menempati posisi bawah, data selanjutnya akan ditaruh di atas data pertama, dan begitupun setelah nya. Saat ingin mengeluarkan data, maka data yang akan dikeluarkan terlebih dahulu adalah data paling atas, dimana data paling atas adalah data yang terakhir di input

Mari kita lihat langsung implementasi pada kodingan, buatlah folder baru bernama stack, lalu buat file bernama StackPertama.java, masukan kode dibawah ini pada file tersebut

```

import java.util.Stack;

public class StackPertama {

    public static void main(String[] args) {

        //creating an instance of Stack class
        Stack<Integer> stk= new Stack<>();
        // checking stack is empty or not
        boolean result = stk.empty();
        System.out.println("Is the stack empty? " + result);
        // pushing elements into stack
        stk.push(78);
        stk.push(113);
        stk.push(90);
        stk.push(120);
        //prints elements of the stack
        System.out.println("Elements in Stack: " + stk);
        result = stk.empty();
        System.out.println("Is the stack empty? " + result);

        stk.pop();
        System.out.println("Elements in Stack: " + stk);
    }
}

```

Mari kita bedah 1 persatu kode diatas

```

import java.util.Stack;

public class StackPertama {

    public static void main(String[] args) {

        //creating an instance of Stack class
        Stack<Integer> stk= new Stack<>();

```

Pertama kita mengimport stack untuk kita gunakan. Lalu kita membuat objek stk yang menampung class Stack

```
// checking stack is empty or not
boolean result = stk.empty();
System.out.println("Is the stack empty? " + result);
// pushing elements into stack
```

Method empty dapat kita gunakan untuk mengecek apakah data pada stack kosong atau tidak

```
// pushing elements into stack
stk.push(78);
stk.push(113);
stk.push(90);
stk.push(120);
```

Lalu, method push kita gunakan untuk menambahkan data pada stack

```
//prints elements of the stack
System.out.println("Elements in Stack: " + stk);
result = stk.empty();
System.out.println("Is the stack empty? " + result);
```

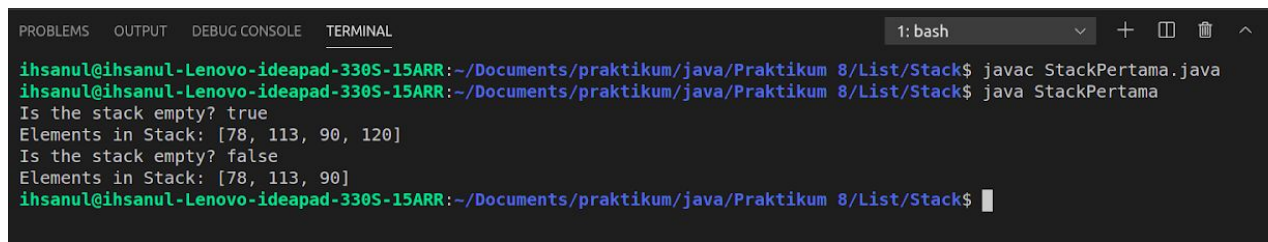
Setelah menambahkan data, kita mengecek kembali apakah data pada stack masih kosong atau tidak

```
stk.pop();
System.out.println("Elements in Stack: " + stk);
```

Method pop kita gunakan untuk menghapus data pada stack. Data yang dihapus adalah data yang terakhir di input pada stack. Sesuai prinsip stack, data akan ditaruh secara bertumpuk, maka data yang terakhir kali di input akan menempati posisi atas, sehingga ketika data dihapus yang akan terhapus adalah data yang diinput terakhir kali.

Notes : jika ingin melihat data paling terakhir di input pada stack, dapat menggunakan method peek(). Data yang akan tampil adalah data yang terakhir kali di input

Mari kita jalankan kode diatas, pada terminal

A screenshot of a terminal window with a dark background. The terminal shows the execution of Java code. The prompt is 'ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/java/Praktikum 8/List/Stack\$'. The first command is 'javac StackPertama.java'. The second command is 'java StackPertama'. The output shows 'Is the stack empty? true' and 'Elements in Stack: [78, 113, 90, 120]'. The third command is 'java StackPertama' again, and the output shows 'Is the stack empty? false' and 'Elements in Stack: [78, 113, 90]'. The terminal window has tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL'. The 'TERMINAL' tab is active. There are icons for adding, removing, and refreshing tabs in the top right corner.

```
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/java/Praktikum 8/List/Stack$ javac StackPertama.java
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/java/Praktikum 8/List/Stack$ java StackPertama
Is the stack empty? true
Elements in Stack: [78, 113, 90, 120]
Is the stack empty? false
Elements in Stack: [78, 113, 90]
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/java/Praktikum 8/List/Stack$
```

Kita telah berhasil menjalankan kode diatas, Pertama kita mengecek apakah stack kosong? Jawaban nya true, karena kita belum mengisi data apapun pada stack. Lalu kita menambahkan data pada stack, dan menampilkan semua data yang kita input pada stack.

Setelah itu kita mengecek lagi apakah data pada stack kosong, jawaban nya false karena kita telah mengisi data pada stack. Setelah itu kita menghapus data pada stack, dan data yang terhapus adalah data yang terakhir kali di input.

Kita check lagi data yang berada pada stack, dan dapat kita lihat data terakhir telah dihapus

Referensi tentang stack dan method lain nya yang ada pada stack:

<https://www.javatpoint.com/java-stack>

Kita telah membahas interface list, sebelum kita beralih ke interface berikutnya yaitu queue, kita akan membahas terlebih dahulu mengenai iterator

Sesuai namanya, iterator memiliki fungsi yang sedikit mirip dengan looping. Pada iterator, setiap collection class dapat melakukan iterasi datanya secara langsung.

Mari kita lihat penerapannya secara langsung, buat folder baru bernama iterator, lalu buatlah file bernama Iterator.java

```
import java.util.Iterator;

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

        ArrayList <Integer> umur = new ArrayList <Integer> ();
        umur.add(17);
        umur.add(8);
        umur.add(16);

        System.out.println(umur);

        Iterator <Integer> it = umur.iterator();

        System.out.println(it.next());
        System.out.println(it.next());
    }
}
```

Kita tidak akan membahas mengenai kode arrayList, tetapi kita akan membahas langsung pada kode iterator

```
import java.util.Iterator;
```

Pertama kita mengimport library iterator

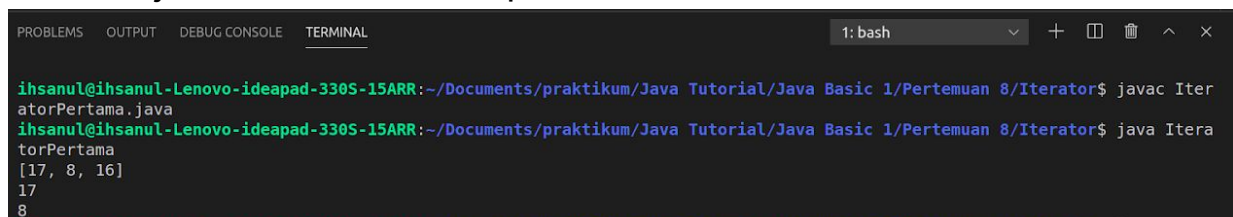
```
Iterator <Integer> it = umur.iterator();
```

Setelah itu kita membuat objek it, dengan tipe data interface Iterator, dan objek it tersebut menyimpan arraylist umur dengan method iterator pada arraylist tersebut.

```
System.out.println(it.next());  
System.out.println(it.next());
```

Kita menjalankan perintah it.next(). Method next kita gunakan untuk menampilkan data sesuai iterasi nya. Pada pemanggilan it.next() yang pertama, yang akan tampil adalah angka 17. Lalu pemanggilan it.next() yang kedua, adalah angka 8.

Mari kita jalankan kode diatas pada terminal



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: bash  
ihسانul@ihسانul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 8/Iterator$ javac Iter  
atorPertama.java  
ihسانul@ihسانul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 8/Iterator$ java Itera  
torPertama  
[17, 8, 16]  
17  
8
```

Pertama, kita menampilkan keseluruhan arraylist nya, lalu kita tampilkan satu persatu menggunakan iterator dengan method next.

Referensi penggunaan iterator :

https://www.w3schools.com/java/java_iterator.asp

Kita telah membahas interface list, mari beralih ke interface berikutnya yaitu queue

Queue adalah interface yang digunakan untuk mengatur bagaimana data di input dan dikeluarkan

Pada queue, data yang pertama kali diinput, maka data tersebut akan di keluarkan terlebih dahulu.

Terdapat 2 class yang mewarisi interface queue , yaitu priority queue dan dequeue

Mari kita bahas priority queue terlebih dahulu

Priority queue menyediakan method method yang dapat kita gunakan layaknya queue.

Kita lihat langsung penerapan nya pada kodingan, buatlah folder baru bernama Queue dan buat file baru bernama PriorityQueuePertama.java, dan masukan kode dibawah ini

```
import java.util.PriorityQueue;
import java.util.Iterator;

class PriorityQueuePertama{
    public static void main(String args[]){
        PriorityQueue<String> queue=new PriorityQueue<String>();
        queue.add("Fatih");
        queue.add("Hardi");
        queue.add("Karan");
        System.out.println("head:"+queue.element());
        System.out.println("head:"+queue.peek());
        System.out.println("iterating the queue elements:");

        Iterator itr=queue.iterator();
        while(itr.hasNext()){
            System.out.println(itr.next());
        }

        queue.remove();
    }
}
```

```

        queue.poll();
        System.out.println("after removing two elements:");
        Iterator<String> itr2=queue.iterator();
        while(itr2.hasNext()){
            System.out.println(itr2.next());
        }
    }
}

```

Mari kita bedah 1 persatu kode diatas

```

import java.util.PriorityQueue;
import java.util.Iterator;

class PriorityQueuePertama{
    public static void main(String args[]){
        PriorityQueue<String> queue=new PriorityQueue<String>();
        queue.add("Fatih");
        queue.add("Hardi");
        queue.add("Karan");
    }
}

```

Kita mengimport library Priority queue dan iterator. Lalu membuat object queue dengan tipe data Strong. Objek queue kita gunakan sebagai instansiasi dari PriorityQueue. Lalu method add kita gunakan untuk menambahkan data kedalam objek queue.

```

System.out.println("head:"+queue.element());
System.out.println("head:"+queue.peek());

```

Method element dan peek kita gunakan untuk melihat data pertama pada queue. Sesuai konsep dari queue, data yang pertama kali diinput maka data tersebut lah yang dapat pertama kali dikeluarkan (Atau data yang menempati posisi pertama, maka data tersebut yang akan ditampilkan)

Perbedaan antara element dan peek, jika tidak terdapat data pada queue, maka peek akan menghasilkan null.


```
System.out.println("iterating the queue elements:");

Iterator itr=queue.iterator();
while(itr.hasNext()){
    System.out.println(itr.next());
}
```

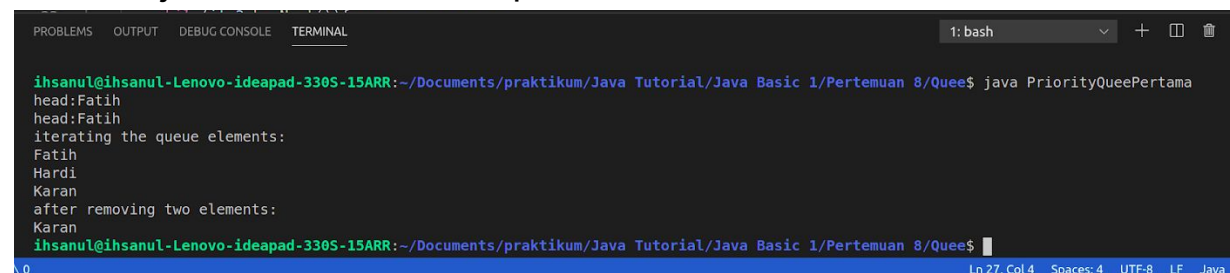
Lalu, kita menggunakan iterator untuk melihat keseluruhan data pada queue 1 persatu. Perulangan while kita gunakan untuk menampilkan keseluruhan data queue. Perulangan dilakukan jika queue memiliki data, contoh nya jika data terakhir sudah ditampilkan, maka akan dicek apakah terdapat data lagi, jika tidak, maka perulangan berhenti

Notes : method hasNext digunakan untuk mengecek apakah terdapat data setelah data yang terakhir kali ditampilkan.

```
queue.remove();
queue.poll();
System.out.println("after removing two elements:");
Iterator<String> itr2=queue.iterator();
while(itr2.hasNext()){
    System.out.println(itr2.next());
}
```

Lalu, kita menghapus method remove dan poll, fungsi kedua method tersebut adalah menghapus data yang menempati posisi awal. Perbedaan nya, jika tidak terdapat data pada queue, maka poll akan menghasilkan nilai null. Setelah menghapus data, Kita membuat objek itr2 sebagai iterator baru yang bertugas untuk menampilkan queue

Mari kita jalankan kode diatas pada terminal



```
ihسانul@ihسانul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 8/Queue$ java PriorityQueuePertama
head:Fatih
head:Fatih
iterating the queue elements:
Fatih
Hardi
Karan
after removing two elements:
Karan
ihسانul@ihسانul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 8/Queue$
```

Kita telah menjalankan kode priority queue

Mari kita bahas turunan interface queue berikut nya yaitu dequeue

Jika pada queue biasa data yang menempati posisi pertama maka akan dikeluarkan terlebih dahulu, maka pada dequeue data yang menempati posisi pertama atau terakhir dapat dikeluarkan terlebih dahulu.

Kita lihat langsung penerapan nya pada kodingan, Masih pada folder Queue, buat file baru bernama DequeuePertama.java, dan masukan kode dibawah ini

```
import java.util.ArrayDeque;
import java.util.Deque;

public class DequeuePertama {
    public static void main(String[] args) {
        Deque<String> deque=new ArrayDeque<String>();
        deque.offer("arvind");
        deque.offer("vimal");
        deque.add("mukul");
        deque.offerFirst("jai");
        System.out.println("After offerFirst Traversal...");
        for(String s:deque){
            System.out.println(s);
        }
        //deque.poll();
        //deque.pollFirst();//it is same as poll()
        deque.pollLast();
        System.out.println("After pollLast() Traversal...");
        for(String s:deque){
            System.out.println(s);
        }
    }
}
```

Mari kita bedah method pada kode diatas, method offer dan add digunakan untuk menambah data pada dequeue dan data akan ditambahkan pada posisi terakhir. Sedangkan method offerFirst digunakan untuk menambahkan data pada posisi pertama.

Sedangkan untuk method poll dan pollFirst digunakan untuk menghapus data pada posisi pertama, sedangkan method pollLast digunakan untuk menghapus data pada posisi terakhir.

Notes : Kalian bisa membuat perulangan khusus data collection dengan cara seperti diatas

```
for(String s:deque) {  
    System.out.println(s);  
}
```

Kode diatas memiliki arti bahwa kita akan melakukan pengulangan pada data deque, dimana deque akan diwakili dengan string s. Setiap perulangan akan memunculkan data deque sesuai posisi dan iterasi perulangan nya. Jika iterasi ke 0 maka posisi nya akan 0, dan data yg ke 0 yang akan ditampilkan. Perulangan ini sama seperti foreach pada bahasa pemrograman lain

Mari kita jalankan kode diatas pada terminal

```
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 8/Quee$ javac DequeuePertama.java  
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 8/Quee$ java DequeuePertama  
After offerFirst Traversal...  
jai  
arvind  
vimal  
mukul  
After pollLast() Traversal...  
jai  
arvind  
vimal
```

Berikut referensi untuk priority queue dan dequeue

<https://www.javatpoint.com/java-priorityqueue>

<https://www.javatpoint.com/java-dequeue-arraydeque>

Kita telah selesai membahas interface queue, mari kita beralih ke interface berikut nya yaitu set

Pada interface set data yang dibuat haruslah berbentuk unique. Jadi tidak boleh ada data yang sama. Ketika ada 2 data yang sama dimasukan, maka data tersebut tidak akan dianggap sebagai 2 data, melainkan hanya 1 data saja. Set tidak mementingkan urutan datanya, yang penting hanya apakah data nya unique atau tidak

Terdapat 3 class yang mewarisi set yaitu hashset, linkedhashset, dan treeset.

HashSet = Penyimpanan data menggunakan algoritma hash table untuk penyimpanan data

LinkedHashSet = Menambahkan implementasi algoritma linkedlist pada hashset

TreeHashSet = Penyimpanan data menggunakan algoritma tree

Kita tidak akan membahas terlalu detail berkaitan dengan algoritma diatas, dan kita hanya akan melihat implementasi kodingan pada hashset

Buatlah sebuah folder baru bernama hashset, dan buat file baru bernama HashSetPertama.java . Masukan kode dibawah ini pada file tersebut

```
import java.util.HashSet;

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

        HashSet <String> item = new HashSet <String> ();

        item.add("Pedang");
        item.add("1192");
        item.add("api");
        item.add("1192");

        System.out.println("Item nya adalah = " + item);
    }
}
```

```

        System.out.println("\nHapus element api");
        item.remove("api");
        System.out.println("tambah element air");
        item.add("air");

        System.out.println("\nItem nya sekarang adalah = " + item);

        System.out.println("Ukuran Item nya = " + item.size());
        System.out.println("Apakah ada element air didalam item ? " +
            item.contains("air"));

        for (String i : item) {
            System.out.println(i);
        }
    }
}

```

Method method pada kode diatas memiliki fungsi yang sama dengan method yang sudah dijelaskan pada interface lain nya. Perbedaan nya hanya pada data nya unique

```

item.add("Pedang");
item.add("1192");
item.add("api");
item.add("1192");

```

Seperti yang teman teman lihat diatas, kita memasukan 2x data 1192. Mari kita coba jalankan diterminal

```

ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 8/HashSet$ javac HashSetPertama.java
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 8/HashSet$ java HashSetPertama
Item nya adalah = [Pedang, api, 1192]

Hapus element api
tambah element air

Item nya sekarang adalah = [Pedang, air, 1192]
Ukuran Item nya = 3
Apakah ada element air didalam item ? true
Pedang
air
1192
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 8/HashSet$

```

Seperti yang temen temen lihat diatas, data 1192 yang di input 2x, hanya terbaca 1x saja ketika dijalankan. Hal ini terjadi karena set tidak mengizinkan adanya data double, semua data bersifat unique.

Berikut referensi mengenai hashset :

<https://www.javatpoint.com/java-hashset>

Untuk teman teman yang penasaran dengan implementasi linkedhashset dan tree hashset dapat mengunjungi link dibawah

<https://www.javatpoint.com/java-linkedhashset>

<https://www.javatpoint.com/java-treeset>

Kita telah membahas mengenai interface set, mari beralih ke interface berikut nya yaitu map

Pada interface map, data disimpan dalam bentuk key dan value. Pada interface interface sebelum nya, data mempunyai posisi masing masing

["fikri", "hanif", "azhar"]

Pada data diatas, fikri menempati posisi 0, hanif menempati posisi 1, azhar menempati posisi 2.

Nah pada map, posisi tersebut akan digantikan dengan key

{"pertama": "fikri", "kedua": "hanif", "ketiga": "azhar"}

Pada contoh diatas, fikri, hanif, dan azhar tidak menggunakan posisi, melainkan menggunakan key. Fikri menempati key pertama, hanif menempati key kedua, azhar menempati key ketiga.

Terdapat 3 class yang mewarisi map yaitu hashmap, linkedhashmap, dan treemap.

HashMap = Penyimpanan data menggunakan algoritma hash table untuk penyimpanan data

LinkedHashMap = Menambahkan implementasi algoritma linkedlist pada hashmap

TreeHashMap = Penyimpanan data menggunakan algoritma tree

Kita tidak akan membahas terlalu detail berkaitan dengan algoritma diatas, dan kita hanya akan melihat implementasi kodingan pada hashmap

Buatlah sebuah folder baru bernama hashmap, dan buat file baru bernama HashMapPertama.java . Masukan kode dibawah ini pada file tersebut

```
import java.util.HashMap;

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

        HashMap <String, String> item = new HashMap <String, String> ();

        item.put("Senjata", "Pedang");
        item.put("Kekuatan", "1192");
        item.put("Element", "api");

        System.out.println("Senjata = "+item.get("Senjata") +
                           "\nmempunyai kekuatan sebesar = " +
                           item.get("Kekuatan") + "\nElement pedang
                           adalah = " + item.get("Element"));

        item.put("Element", "Air");

        item.remove("Kekuatan");
        System.out.println("\n" + "Item nya adalah = " + item);
        System.out.println("\n" + "Ukuran Item nya = " + item.size());

        for (String i : item.keySet()) {
            System.out.println("\nItem nya adalah = " + item.get(i));
        }
    }
}
```

Mari kita sedikit bedah kode diatas

```
item.put("Senjata", "Pedang"); // Senjata = key, pedang = value
item.put("Kekuatan", "1192"); // Kekuatan = key, 1192 = value
item.put("Element", "api"); // Element = key, api = value

System.out.println("Senjata = "+item.get("Senjata") +
    "\nmempunyai kekuatan sebesar = " +
    item.get("Kekuatan") + "\nElement pedang
    adalah = " + item.get("Element"));
```

Untuk menambahkan data, kita menggunakan method put. Dimana pada parameter pertama adalah key nya, sedangkan parameter kedua adalah valuenya. Untuk memunculkan value nya, kita dapat menggunakan method get, lalu memasukkan key data yang akan kita munculkan.

Contoh nya item.get("Senjata") , maka yang akan muncul adalah pedang. item.get(Kekuatan) maka yang akan muncul adalah 1192

```
item.put("Element", "Air");

item.remove("Kekuatan");
```

Nah, method put selain untuk menambahkan data, juga dapat merubah data. Kita tahu bahwa key Element menyimpan value api, pada kode diatas, kita merubah value dari key Element menjadi air

Sedangkan untuk method remove digunakan untuk menghapus data. Parameter yang digunakan pada method remove adalah key dari data yang ingin dihapus

Notes : Untuk pembuatan hasmap, kita tidak dapat membuat 2 key yang sama. Ketika kita membuat key yang sama, maka akan dianggap sebagai 1 key.

Mari kita coba jalankan di terminal

```
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 8/HashMap$ javac HashMapPertama.java
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 8/HashMap$ java HashMapPertama
Senjata = Pedang
mempunyai kekuatan sebesar = 1192
Element pedang adalah = api

Item nya adalah = {Element=Air, Senjata=Pedang}

Ukuran Item nya = 2

Item nya adalah = Air

Item nya adalah = Pedang
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 8/HashMap$
```

Kita telah menjalankan hashmap

Berikut referensi dari hashmap :

<https://www.javatpoint.com/java-hashmap>

<https://www.javatpoint.com/working-of-hashmap-in-java>

Untuk teman teman yang penasaran dengan implementasi linkedhashmap dan treehashmap dapat mengunjungi link dibawah

<https://www.javatpoint.com/java-linkedhashmap>

<https://www.javatpoint.com/java-treemap>

Tugas

1. Buatlah laporan terkait praktikum modul Java Collection
2. Screenshot kode dan hasil kode yang dijalankan dari penjelasan praktikum ini. Masukkan hasil screenshot tersebut dan jelaskan kode nya

Tugas dikirim pada elen, dengan format sebagai berikut

1. Pada penamaan file gunakan format berikut
Praktikum_keberapa_Judul_Praktikum_Nama_Nim
Contoh :
Praktikum01_Pengenalan-Java_IhsanulFikriAbiyyu_0220318021
2. Format laporan bertipe pdf
3. Pada cover praktikum, masukan beberapa hal dibawah ini

- Praktikum ke berapa
- Judul Praktikum
- Nama
- Nim

4. Tenggat waktu mengerjakan adalah 1 pekan semenjak tugas diberikan

5. Diperbolehkan berkelompok tetapi hanya antara yang tidak punya laptop/komputer dengan yang punya laptop/komputer. Selain itu tidak boleh