Dt : 14/10/2023

Ex-1:(Demonstrating Queue<E> operations)

Program : DemoQueue.java

```
package p2;
import java.util.*;
public class DemoQueue {
     @SuppressWarnings("removal")
     public static void main(String[] args) {
        PriorityQueue<Integer> pq=
             new PriorityQueue<Integer>();
        for(int i=21;i<=25;i++) {
        pq.add(new ~~Integer~~(i));
        }
        System.out.println("===Queue<E> elements====");
        System.out.println(pq.toString());
        System.out.println("====offer(E)====");
        pq.offer(new ~~Integer~~(500));
        System.out.println(pq.toString());
        System.out.println("====remove()=====");
        pq.remove();
        System.out.println(pq.toString());
        System.out.println("====poll()=====");
        pq.poll();
        System.out.println(pq.toString());
        System.out.println("====element()=====");
        System.out.println("element : "+pq.element());
        System.out.println("====peek()=====");
        System.out.println("peek : "+pq.peek());
     }
}
```

o/p:

```
===Queue<E> elements====
[21, 22, 23, 24, 25]

====offer(E)====

[21, 22, 23, 24, 25, 500]

====remove()=====

[22, 24, 23, 500, 25]

====poll()=====

[23, 24, 25, 500]

====element()=====

element : 23

====peek()=====

peek : 23
```

----

add(E) : method is used to add the element to Queue<E> object

offer(E) : method also used to add the element to Queue<E> object

remove() : method is used to remove the element from the Queue<E> Object

poll()   : method also used to remove the element from the Queue<E> object

element()   : method is used to display the top-of-queue

peek()   : method also used to display the top-of-queue

==

Ex-2:(Demonstraing Deque<E> Operations)

Program : DemoDeque.java

```
package p2;
import java.util.*;
public class DemoDeque {
    @SuppressWarnings("removal")
    public static void main(String[] args) {
        ArrayDeque<Integer> ad =
            new ArrayDeque<Integer>();
        for(int i=21;i<=25;i++)
        {
        ad.add(new ~~Integer~~(i));
        }
        System.out.println("====Deque<E> elements====");
        System.out.println(ad.toString());
        ad.addFirst(new ~~Integer~~(600));
        ad.addLast(new ~~Integer~~(800));
        System.out.println(ad.toString());
        ad.removeFirst();
        ad.removeLast();
        System.out.println(ad.toString());
        System.out.println("FirstEle:"+ad.getFirst());
        System.out.println("LastEle:"+ad.getLast());
        System.out.println("****iterator()****");
        Iterator<Integer> ob1 = ad.iterator();
        ob1.forEachRemaining((k)->
        {
        System.out.print(k.toString()+" ");
        });
System.out.println("\n***descendingIterator()****");
        Iterator<Integer> ob2 = ad.descendingIterator();
        ob2.forEachRemaining((k)->
        {
        System.out.print(k.toString()+" ");
        });
    }
}
```

o/p:

```
====Deque<E> elements====
```

[21, 22, 23, 24, 25]

[600, 21, 22, 23, 24, 25, 800]

[21, 22, 23, 24, 25]

FirstEle:21

LastEle:25

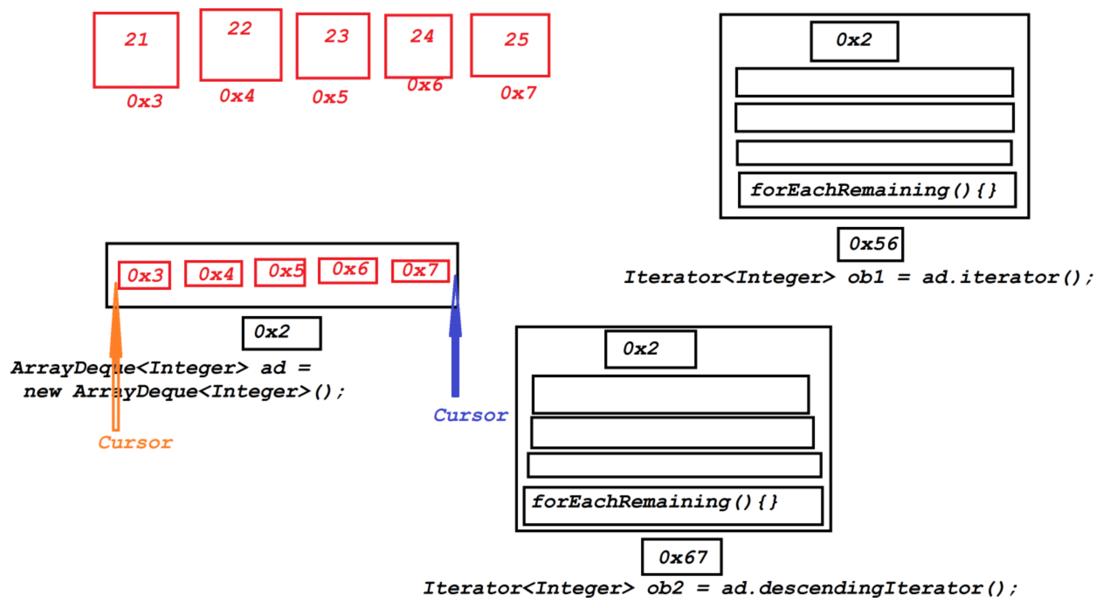
*******iterator()*******

21 22 23 24 25

******descendingIterator()*******

25 24 23 22 21

Diagram:



===

faq:

wt is the diff b/w

(i)iterator()

(ii)descendingIterator()

(iii)asIterator()

(i)iterator():

=>iterator() is a method from java.lang.Iterable<E> interface and which is available to Set<E>,List<E> and Queue<E>,which creates the implementation object for Iterator<E> interface and the object will hold the reference of Collection<E> object,and the iterator() method also generate cursor pointing before the first element of Collection<E> object.

(ii)descendingIterator():

=>descendingIterator() is a method from Deque<E> and which is used to create implementation object for Iterator<E> interface and the object will hold the reference of Deque<E> object,and the method also generate cursor pointing after the last element of Deque<E> object.

imp

(iii)asIterator():

=>asIterator() is a method from Enumeration<E> and which is used to create implementation object for Iterator<E> and the object will hold the reference of Enumeration<E> object,the the method also generate cursor pointing before the first element of Vector<E> Object.

==

imp

Limitation of Collection<E>:

=>Collection<E> cannot differentiate Primary-key while holding database table data.

Note:

=>Limitation of Collection<E> can be Overcomed using Map<K,V>

==

imp

Map<K,V> in Java:

=>Map<K,V> is an interface from java.util package and which organizes

elements in the for of key-value pairs.

K - Key

V - Values

=>The following are some important methods from Map<K,V>:

```
public abstract int size();
public abstract boolean isEmpty();
public abstract boolean containsKey(java.lang.Object);
public abstract boolean containsValue(java.lang.Object);
public abstract V get(java.lang.Object);
public abstract V put(K, V);
public abstract V remove(java.lang.Object);
public abstract void putAll(java.util.Map<? extends K, ? extends V>);
public abstract void clear();
public abstract java.util.Set<K> keySet();
public abstract java.util.Collection<V> values();
public abstract java.util.Set<java.util.Map$Entry<K, V>> entrySet();
public default void forEach
    (java.util.function.BiConsumer<? super K, ? super V>);
public default void replaceAll
  (java.util.function.BiFunction<? super K, ? super V, ? extends V>);
public static <K, V> java.util.Map<K, V> of();
```

=>The following are the implementation classes of Map<K,V>:

(a)HashMap<K,V>

(b)LinkedHashMap<K,V>

(c)TreeMap<K,V>

(d)Hashtable<K,V>

===