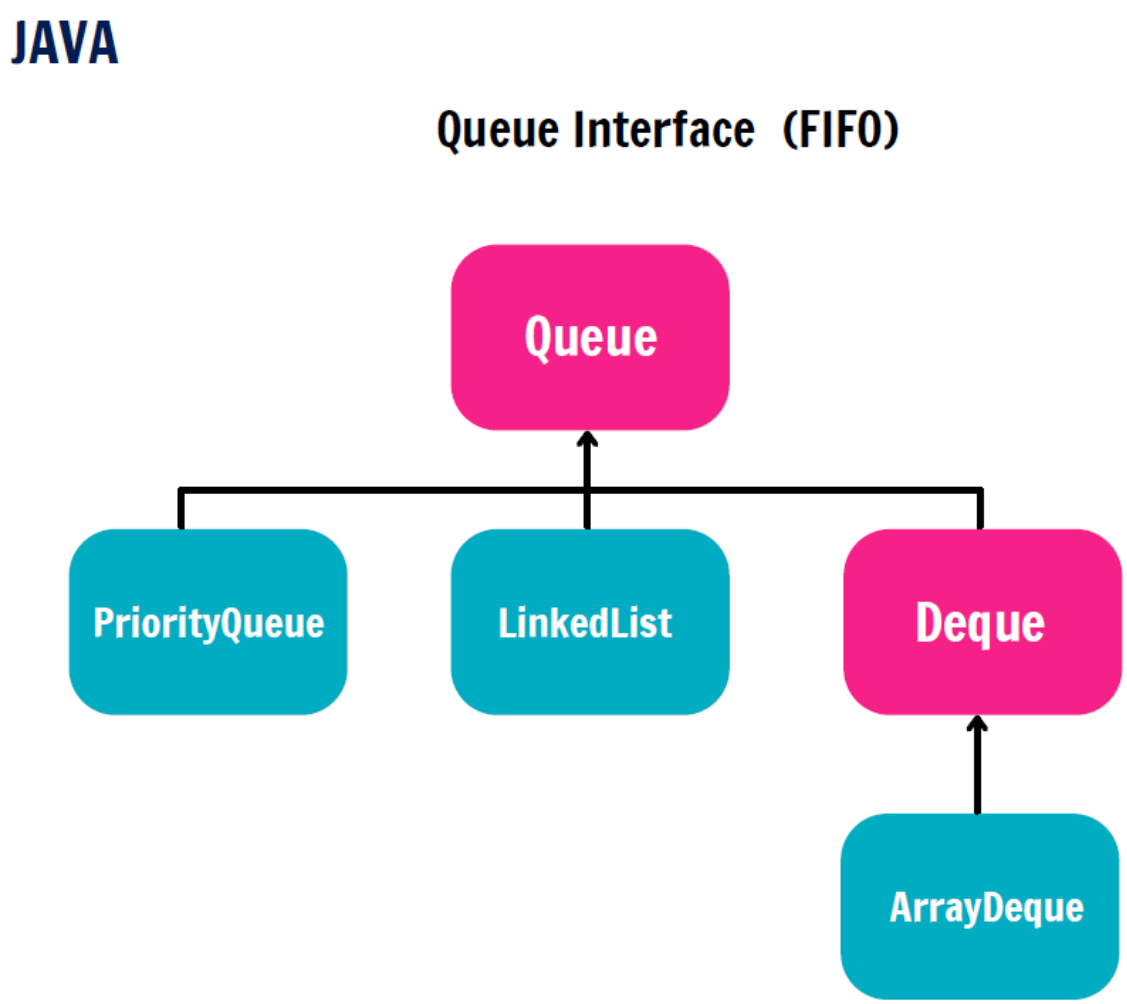
Java- Introduction to Queue
Lecture 15

Queue:



The Queue Interface is present in java.util package and extends the Collection interface. It stores and processes the data in FIFO(First In First Out) order. which means that the elements are retrieved in the order in which they were added to the queue.

It is an ordered list of objects limited to inserting elements at the end of the list and deleting elements from the start of the list.

Declaration of Java Queue Interface:

```
package myPackage;

import java.util.ArrayDeque;
import java.util.LinkedList;
import java.util.PriorityQueue;
```

```
import java.util.Queue;

public class Queues {

    public static void main(String[] args) {
        Queue <Integer> q = new LinkedList<Integer>();
        //or
        Queue<String>q2 = new ArrayDeque<String>();
        //or
        Queue<Character> q3 = new PriorityQueue<Character>();
    }
}
```

Queue is an interface; objects cannot be created of the type queue. We always need a class, which extends this list in order to create an object. LinkedList , PriorityQueue, ArrayDeque are the implementation classes of Queue.

** PriorityQueue do not allow null elements.*

Adding Elements: add()

```
package myPackage;
import java.util.LinkedList;
import java.util.Queue;

public class Queues {

    public static void main(String[] args) {
        Queue <Integer> q = new LinkedList<Integer>();
        q.add(2);
        q.add(3);
        q.add(4);
        q.add(5);
        System.out.println(q);
        }
}
```

Removing Elements: remove() and poll()

Remove():first occurrence of the object is removed.

Poll(): remove the head and return it.

```
package myPackage;
import java.util.LinkedList;
import java.util.Queue;

public class Queues {

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

```
        Queue <Integer> q = new LinkedList<Integer>();
        q.add(2);
        q.add(3);
        q.add(4);
        q.add(5);
        System.out.println(q);
        q.remove();
        System.out.println(q);
        System.out.println(q.poll());
        System.out.println(q);
        }
}
```

Iterating the Queue:

```
package myPackage;
import java.util.LinkedList;
import java.util.*;

public class Queues {

    public static void main(String[] args) {
        Queue <Integer> q = new LinkedList<Integer>();
        q.add(2);
        q.add(3);
        q.add(4);
        q.add(5);
        System.out.println(q);
        Iterator iterator = q.iterator();
        while (iterator.hasNext()) {
            System.out.println(iterator.next()+ " ");
        }
        }
}
```

Deque :

Deque Interface present in java.util package is a subtype of the queue interface. The Deque is related to the double-ended queue that supports adding or removing elements from either end of the data structure. It can either be used as a queue(first-in-first-out/FIFO) or as a stack(last-in-first-out/LIFO).

Declaration of Java Deque Interface:

```
package myPackage;

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

```
import java.util.LinkedList;

public class Queues {

    public static void main(String[] args) {
        Deque <Integer> d = new LinkedList<Integer>();
        //or
        Deque<String>q2 = new ArrayDeque<String>();
        }
}
```

Deque is an interface; objects cannot be created of the type deque. We always need a class that extends this list in order to create an object. LinkedList , ArrayDeque are the implementation classes of Deque.

Adding Elements: add(), addFirst(), addLast()

```
package myPackage;
import java.util.*;

public class Queues {

    public static void main(String[] args) {
        Deque <Integer> d = new ArrayDeque<Integer>();
        d.add(2);
        d.addFirst(3);
        d.addLast(4);
        System.out.println(d);

        }
}
```

Removing Elements: removeFirst(), removeLast(), poll(), pop(), pollFirst(), pollLast()

```
package myPackage;
import java.util.*;

public class Queues {

    public static void main(String[] args) {
        Deque <Integer> d = new ArrayDeque<Integer>();
        d.add(2);
        d.addFirst(3);
        d.addLast(4);
        System.out.println(d);
        System.out.println(d.pop());
```

```
        System.out.println(d.poll());

        System.out.println(d.pollFirst());

        System.out.println(d.pollLast());

        }
}
```

Iterating through the Deque

```
package myPackage;
import java.util.*;

public class Queues {

    public static void main(String[] args) {
        Deque <Integer> d = new ArrayDeque<Integer>();
        d.add(2);
        d.addFirst(3);
        d.addLast(4);
        d.add(6);
        d.add(5);
        System.out.println(d);
        // Iterator itr = d.iterator();
        // while(itr.hasNext()){
        //     System.out.println(itr.next()+" ");
        // }
        Iterator itr = d.descendingIterator();
        while(itr.hasNext()){
            System.out.println(itr.next()+" ");
        }
        }
}
```

PriorityQueue:

The PriorityQueue class in Java is part of the java.util package. It is known that a Queue follows the FIFO(First-In-First-Out) Algorithm, but the elements of the Queue are needed to be processed according to the priority, that's when the PriorityQueue comes into play.

No Null Elements are available in PriorityQueue it will throw NullPointerException in such a case.

Declaration of Java PriorityQueue Class:

```java
package myPackage;
import java.util.*;

public class Queues {

    public static void main(String[] args) {
        PriorityQueue <Integer> pq = new PriorityQueue<Integer>();

        }
}
```

Adding Elements: add()

```java
package myPackage;
import java.util.*;

public class Queues {

    public static void main(String[] args) {
        Queue <Integer> d = new PriorityQueue<Integer>();
        d.add(2);
        d.add(6);
        d.add(5);
        System.out.println(d);

        }
}
```

Removing Elements: remove(), poll()

```java
package myPackage;
import java.util.*;

public class Queues {

    public static void main(String[] args) {
        Queue <Integer> d = new PriorityQueue<Integer>();
        d.add(2);
        d.add(6);
        d.add(5);
        System.out.println(d);
        d.remove();
        System.out.println(d);
        System.out.println(d.poll());
        System.out.println(d);
```

```
        }
}
```

Accessing the elements: peek()

```
package myPackage;
import java.util.*;

public class Queues {

    public static void main(String[] args) {
        Queue <Integer> d = new PriorityQueue<Integer>();
        d.add(2);
        d.add(6);
        d.add(5);
        System.out.println(d);
        System.out.println(d.peek());

        }
}
```

Clear(),size(),contains()

```
package myPackage;
import java.util.*;

public class Queues {

    public static void main(String[] args) {
        Queue <Integer> d = new PriorityQueue<Integer>();
        d.add(2);
        d.add(6);
        d.add(5);
        System.out.println(d);
        //check if queue contains any specific element
        System.out.println(d.contains(5));
        //check size
        System.out.println(d.size());
        //clear the queue
        d.clear();
        System.out.println(d);
        }
}
```

Iterating the PriorityQueue:

```java
package myPackage;
import java.util.*;

public class Queues {

    public static void main(String[] args) {
        Queue <Integer> d = new PriorityQueue<Integer>();
        d.add(2);
        d.add(6);
        d.add(5);
        System.out.println(d);
        Iterator iterator = d.iterator();

        while (iterator.hasNext()) {
            System.out.print(iterator.next() + " ");
        }

        }
}
```

ArrayDeque: ArrayDeque is a resizable array implementation of the Deque interface, which stands for double-ended queue. It allows elements to be added or removed from both ends efficiently. It can be used as a stack (LIFO) or a queue (FIFO).

- ArrayDeque grows dynamically.
- It generally provides faster operations than LinkedList because it does not have the overhead of node management.

Declaration of Java ArrayDeque Class:

```java
package myPackage;
import java.util.*;

public class Queues {

    public static void main(String[] args) {
        ArrayDeque <Integer> pq = new ArrayDeque<Integer>();

        }
}
```

Adding Element: add(), addFirst(), addLast(), offer(), offerFirst(), offerLast()

```java
package myPackage;
import java.util.*;

public class Queues {

    public static void main(String[] args) {
        ArrayDeque <Integer> ad = new ArrayDeque<Integer>();
        ad.add(1);
        ad.addFirst(2);
        ad.addLast(3);
        System.out.println(ad);
        ad.offer(4);
        ad.offerFirst(5);
        ad.offerLast(6);
        System.out.println(ad);
        }
}
```

Accessing Elements: getFirst(), getLast(), peek(), peekFirst(), peekLast()

```java
package myPackage;
import java.util.*;

public class Queues {

    public static void main(String[] args) {
        ArrayDeque <Integer> ad = new ArrayDeque<Integer>();
        ad.add(1);
        ad.addFirst(2);
        ad.addLast(3);
        System.out.println(ad);
        ad.offer(4);
        ad.offerFirst(5);
        ad.offerLast(6);
        System.out.println(ad);
        System.out.println(ad.getFirst());
        System.out.println(ad.getLast());
        System.out.println(ad.peek());
        System.out.println(ad.peekLast());
        }
}
```

Removing Elements: remove(), removeFirst(), removeLast(), poll(), pollFirst(), pollLast(), pop()

```java
package myPackage;
import java.util.*;

public class Queues {

    public static void main(String[] args) {
        ArrayDeque <Integer> ad = new ArrayDeque<Integer>();
        ad.add(1);
        ad.addFirst(2);
        ad.addLast(3);
        System.out.println(ad);
        ad.offer(4);
        ad.offerFirst(5);
        ad.offerLast(6);
        System.out.println(ad);
        System.out.println(ad.pop());
        System.out.println(ad.poll());
        System.out.println(ad.pollLast());
        }
}
```

Iterating Elements:

```java
package myPackage;
import java.util.*;

public class Queues {

    public static void main(String[] args) {
        ArrayDeque <Integer> d = new ArrayDeque<Integer>();
        d.add(2);
        d.addFirst(3);
        d.addLast(4);
        d.add(6);
        d.add(5);
        System.out.println(d);
        // Iterator itr = d.iterator();
        // while(itr.hasNext()){
        //     System.out.println(itr.next()+" ");
        // }
        Iterator itr = d.descendingIterator();
        while(itr.hasNext()){
            System.out.println(itr.next()+" ");
        }
        }
}
```