**Lab8:**

* **Queue interface**
* **Queue implementation using array**
* **Queue implementation using Singly Linked List**
* **CircularlyQueue implementation using Circularly Linked List**

**Code:**

public interface Queue<E> {  
  
 boolean isEmpty();  
 int size();  
 E first();  
 void enqueue(E element);  
 E dequeue();  
  
}

public class ArrayQueue<E> implements Queue<E> {  
 E data[];  
 static int *capacity*=100;  
 int f=0;//مؤشر لاول عنصر  
 int sz=0;// عدد العناصر في المصفوفة  
  
 public ArrayQueue(int c) {  
 data= (E[])new Object[c];  
 }  
  
 public ArrayQueue() {  
 this(*capacity*);  
 //data= (E[])new Object[capacity];  
 }  
  
 @Override  
 public boolean isEmpty() {  
 return sz==0;  
 }  
  
 @Override  
 public int size() {  
 return sz;  
 }  
  
 @Override  
 public E first() {  
 if (isEmpty())  
 return null;  
 return data[f];  
 }  
  
 @Override  
 public void enqueue(E element) throws IllegalStateException { //Enter method  
 if(sz==data.length) throw new IllegalStateException("Queue is full!");  
 int x=(f+sz)%data.length;//علشان مايخرج ضمن حدود المصفوفة نقسم على طولها  
 data[x]=element;  
 sz++;  
 }  
  
 @Override  
 public E dequeue() { //deleted method  
 if (isEmpty())  
 return null;  
 E del =data[f];  
 data[f]=null;  
 f=(f+1)%data.length;  
 sz--;  
 return del;  
 }  
}

public class LinkedQueue<E> implements Queue<E> {  
 SinglyLinkedList<E> list=new SinglyLinkedList<E>();  
  
 @Override  
 public boolean isEmpty() {  
 return list.isEmpty();  
 }  
  
 @Override  
 public int size() {  
 return list.Size();  
 }  
  
 @Override  
 public E first() {  
 return list.first();  
 }  
  
 @Override  
 public void enqueue(E element) {  
 list.addLast(element);  
 }  
  
 @Override  
 public E dequeue() {  
 return list.removeFirst();  
 }  
}

public interface CircularyQueue<E> extends Queue<E>{  
 void rotate();  
}

public class CircularQueue<E> implements CircularyQueue<E> {  
 CircularyLinkedList<E> list =new CircularyLinkedList<E>();  
  
  
 @Override  
 public boolean isEmpty() {  
 return list.isEmpty();  
 }  
  
 @Override  
 public int size() {  
 return list.getSize();  
 }  
  
 @Override  
 public E first() {  
 return list.first();  
 }  
  
 @Override  
 public void enqueue(E element) {  
 list.addLast(element);  
 }  
  
 @Override  
 public E dequeue() {  
 return list.removeFirst();  
 }  
  
 @Override  
 public void rotate() {  
 list.rotate();  
 }  
}