큐 (Queue)

강 지 훈
jhkang@cnu.ac.kr

충남대학교 컴퓨터공학과

큐 (Queue)

큐 (Queue)

- 서비스를 받기 위해 서는 줄
  - 줄에 먼저 서는 것이 먼저 서비스 받는다.
  - 버스를 타기 위해서 서는 줄
  - 식당에서 배식을 위해 서는 줄
  - 컴퓨터 안에서 일(job)들이 처리 서비스를 받기 위해 서는 줄
- 선입선출 (先入先出)
  - FIFO: First-In-First-Out

큐

- 순서 리스트 (ordered list)
- 삽입과 삭제는 큐의 양 끝에서
  - 새로운 원소의 삽입은 항상 큐의 뒤쪽에서
  - 서비스 받기 위한 원소의 삭제는 큐의 앞쪽에서

$$Q = (a_0, \cdots, a_{n-1})$$

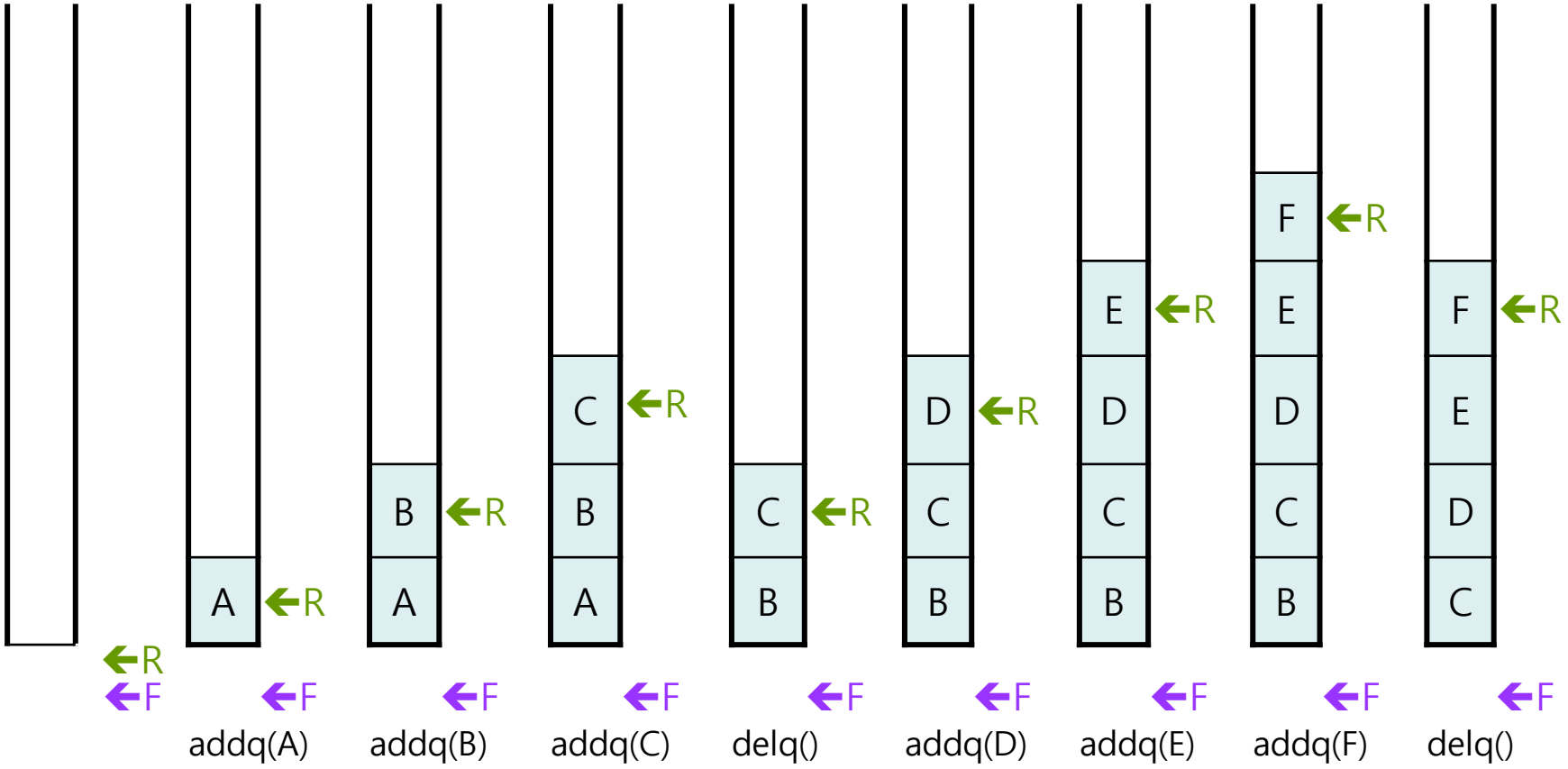
front 원소 rear 원소

a_{i+1} 은 a_i 뒤에 있다 $(0 \le i < n-1)$

예: 큐의 작동 원리

"←R" 큐에서 rear 원소를 가리킨다

"←F" 큐에서 front 원소를 가리킨다



Class “ArrayQueue”

❑ ArrayQueue의 공개함수

- ArrayQueue 객체 사용법
  - public ArrayQueue () ;
  - public ArrayQueue (int initialCapacity) ;

  - public boolean isEmpty () ;
  - public boolean isFull () ;
  - public int size () ;

  - public T frontElement () ;

  - public boolean enQueue (T anElement) ;
  - public T deQueue () ;
  - public void clear () ;

Class “ArrayQueue”의 구현

❑ ArrayQueue: 비공개 인스턴스 변수

```
public class ArrayQueue<T>
{
    // 비공개 인스턴스 변수
    private static final int DEFAULT_INITIAL_CAPACITY = 50 ;
    private int              _maxSize ;
    private int              _front ;
    private int              _rear ;
    private T[]              _elements ;
```

❑ ArrayQueue의 생성자

```
public class ArrayQueue<T>
{
    // 생성자
    public ArrayQueue ()
    {
        this  ArrayQueue.DEFAULT_INITIAL_CAPACITY ) ;
    }

    public ArrayQueue (int initialCapacity)
    {
        @SuppressWarnings("Unchecked") ;
        this._elements = (T[ ]) new Object[initialCapacity] ;
        this._maxSize = initialCapacity ;
        this._front = -1 ;
        this._rear = -1  ;
    }
```

- static으로 선언된 상수/변수/함수를 언급할 때에는 class의 이름으로.
- (static으로 선언되지 않은 것을 언급할 때에는 this)

❑ ArrayQueue : 상태 알아보기

```
public class ArrayQueue<T>
{

    ......

    // 상태 알아보기
    public boolean  isEmpty ()
    {
        return  (this._front == this._rear) ;
    }

    public boolean  isFull ()
    {
        return  (this._rear == (this._maxSize-1) ) ;
    }

    public int  size()
    {
        return ( this._rear – this._front) ;
    }
```

❑ ArrayQueue : frontElement()

```
public T  frontElement()
{
    T  frontElement = null ;
    if ( ! isEmpty() ) {
        frontElement = this._elements[this._front+1] ;
    }
    return  frontElement ;
}
```

❑ ArrayQueue : enQueue()

```
// 원소 추가 함수
public boolean  enQueue (T anElement)
{
    if ( this.isFull() ) {
        return false ;
    }
    else {
        this._rear++ ;
        this._elements[this._rear] = anElement ;
        return  true ;
    }
}
```

❑ ArrayQueue : deQueue()

```
public T  deQueue()
{
    T  frontElement = null ;
    if ( ! this.isEmpty() ) {
        this._front++ ;
        frontElement = this._elements[this._front] ;
        this._elements[this._front] = null ;
    }
    return frontElement ;
}
```

❑ ArrayQueue : clear()

```
public void  clear ()
{
    for ( i = 1 ; i <= this.size() ; i++ ) {
        this._elements[this._front+i] = null ;
    }
    this._front = -1 ;
    this._rear = -1 ;
}
```

CircularArrayQueue로의 구현

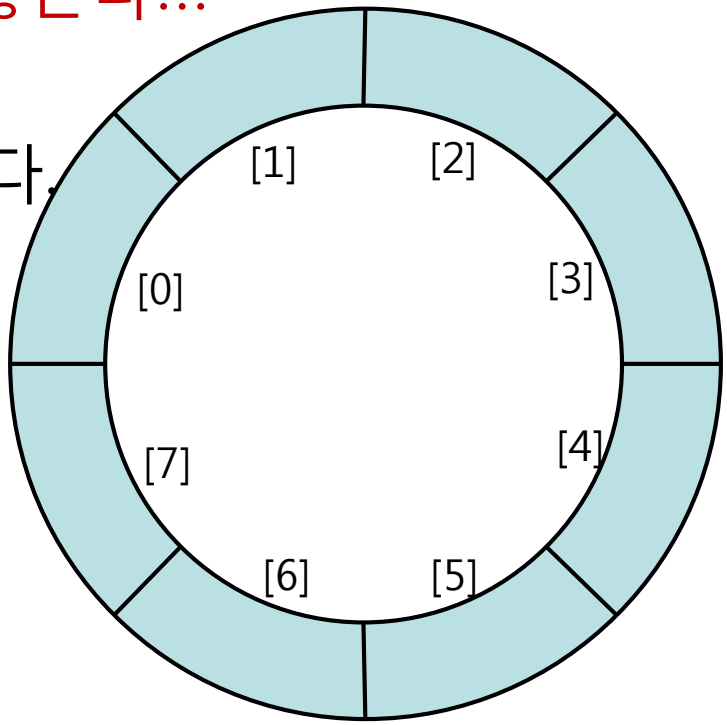
❑ ArrayQueue 구현의 문제점

■ 큐는 점진적으로 왼쪽에서 오른쪽으로 이동!!

front	rear	Q[0]	Q[1]	Q[2]	Q[3]	설명
-1	-1					Empty queue
-1	0	J_1				enQ J_1
-1	1	J_1	J_2			enQ J_2
-1	2	J_1	J_2	J_3		enQ J_3
0	2		J_2	J_3		deQ J_1
1	2			J_3		deQ J_2
1	3			J_3	J_4	enQ J_4
2	3				J_4	deQ J_3

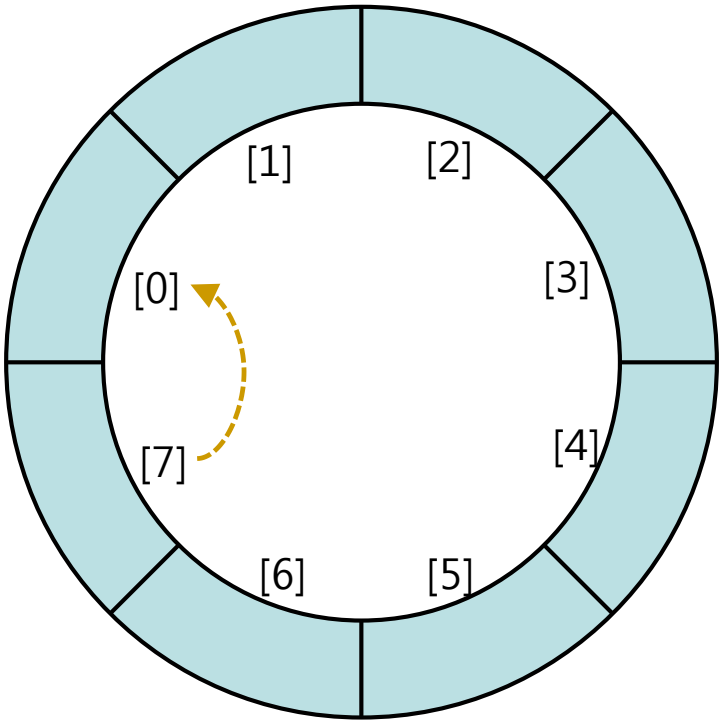
Circular Array Queue로 구현

- 반지 형태의 환형 배열:
  - 배열의 끝이 배열의 맨 처음과 붙어 있는 것으로 본다.
- Public 함수의 사용법은 바뀌지 않는다!!!
- 함수의 구현만 약간 변경될 뿐이다.
  - 함수 코드를 일부 수정



다음 위치 계산은?

- 크기가 8인 Circular Queue 에서,
  - [7]번째 다음은 [0]번째



- 일반적으로 다음 위치 계산은 어떻게?

```
this._rear++;
if (this._rear == this._maxSize)
    this._rear = 0 ;
```

  - 이렇게도...

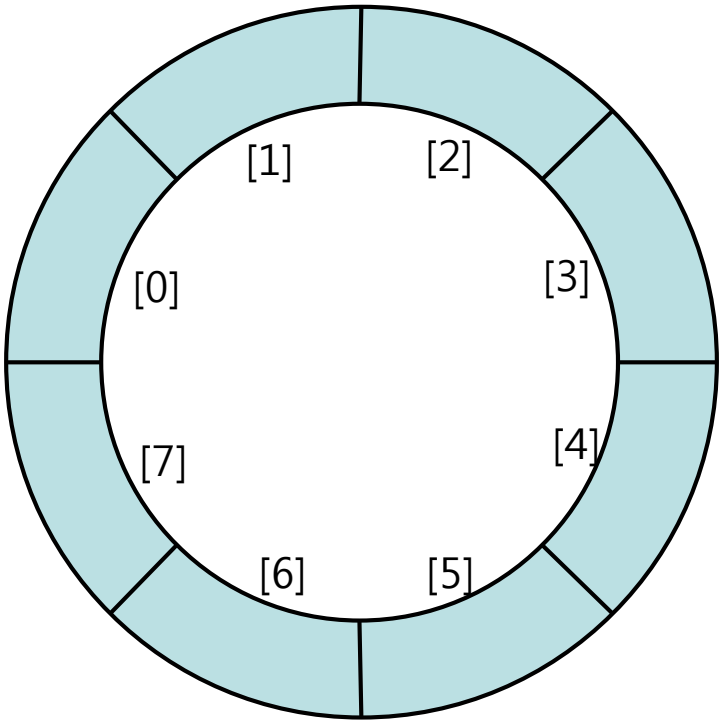
```
this._rear = (this._rear+1) % this._maxSize ;
```

초기화 / Empty 조건

- 초기화는?
  - this._front = 0 ;
  - this._rear = 0 ;

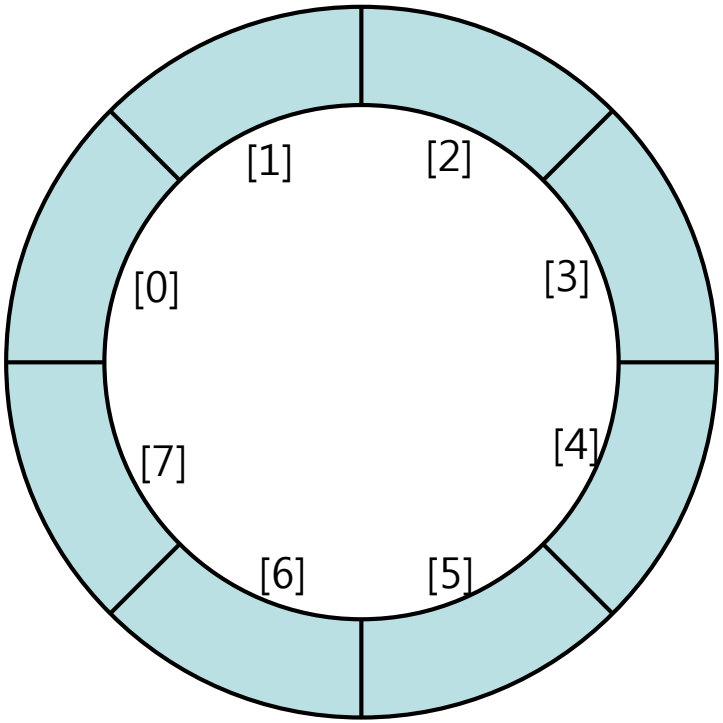
- 큐 Empty 조건은?
  - (this._front == this._rear)
    - 초기 상태도 empty임

- 큐 Full 조건은?



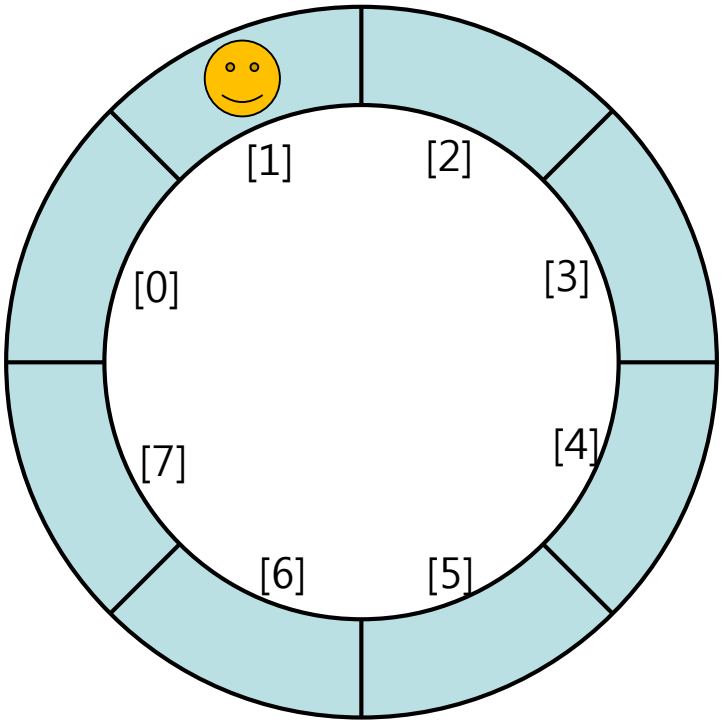
Full 조건 [1]

- 큐 Full 조건은?
  - 초기 시작은 (큐 empty 상태),
    - this._front == 0
    - this._rear == 0



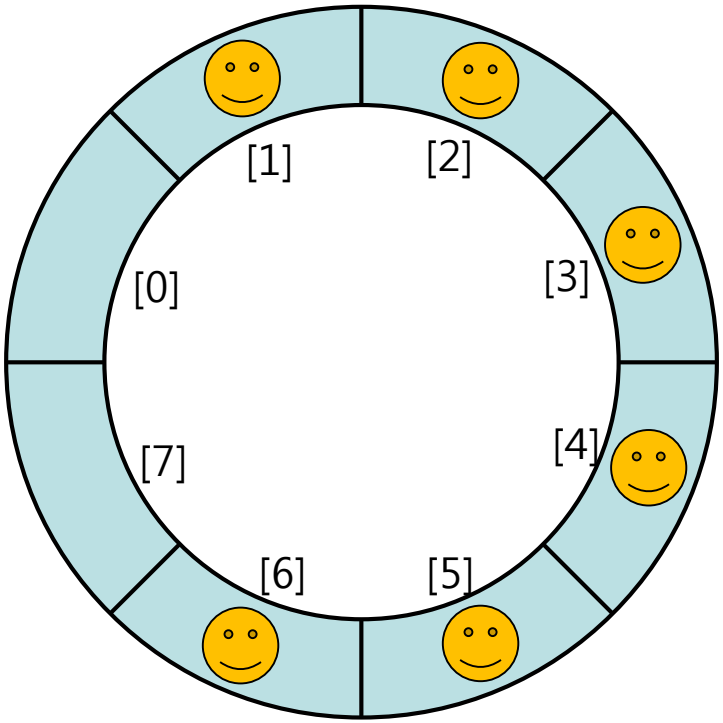
Full 조건 [2]

- 큐 Full 조건은?
  - 초기 시작은,
    - this._front == 0
    - this._rear == 0
  - 1개가 차면,
    - this._front == 0
    - this._rear == 1



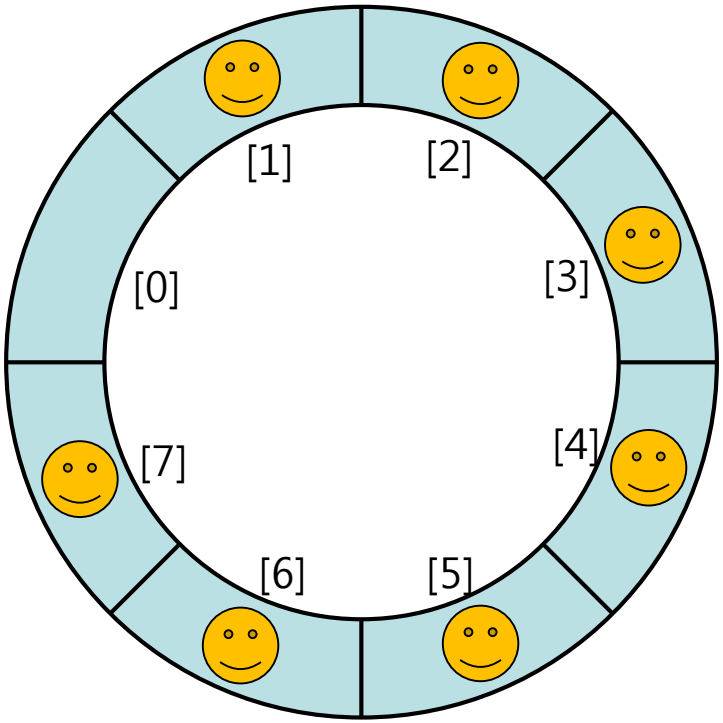
Full 조건 [2]

- 큐 Full 조건은?
  - 초기 시작은,
    - this._front == 0
    - this._rear == 0
  - 6개가 차면,
    - this._front == 0
    - this._rear == 6



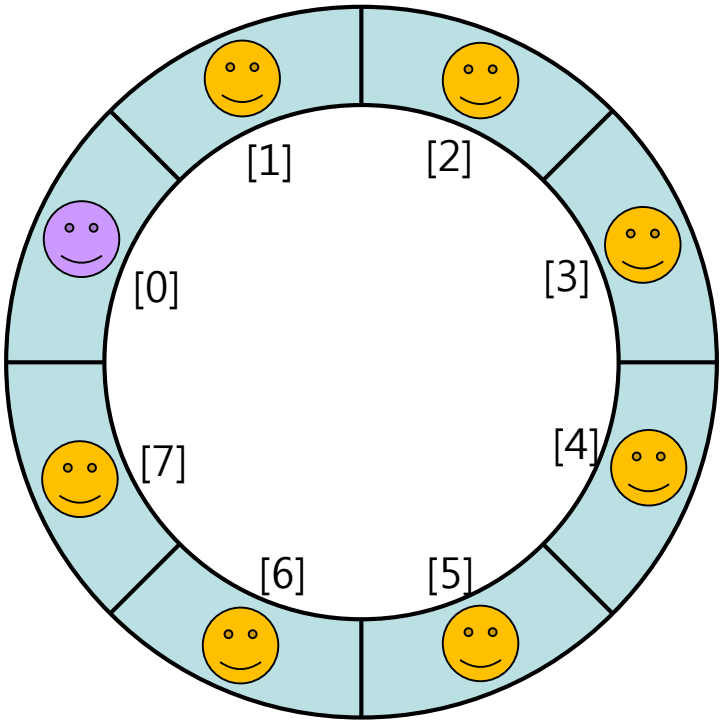
Full 조건 [2]

- 큐 Full 조건은?
  - 초기 시작은,
    - this._front == 0
    - this._rear == 0
  - 6개가 차면,
    - this._front == 0
    - this._rear == 6
  - 7개가 차면,
    - this._front == 0
    - this._rear == 7



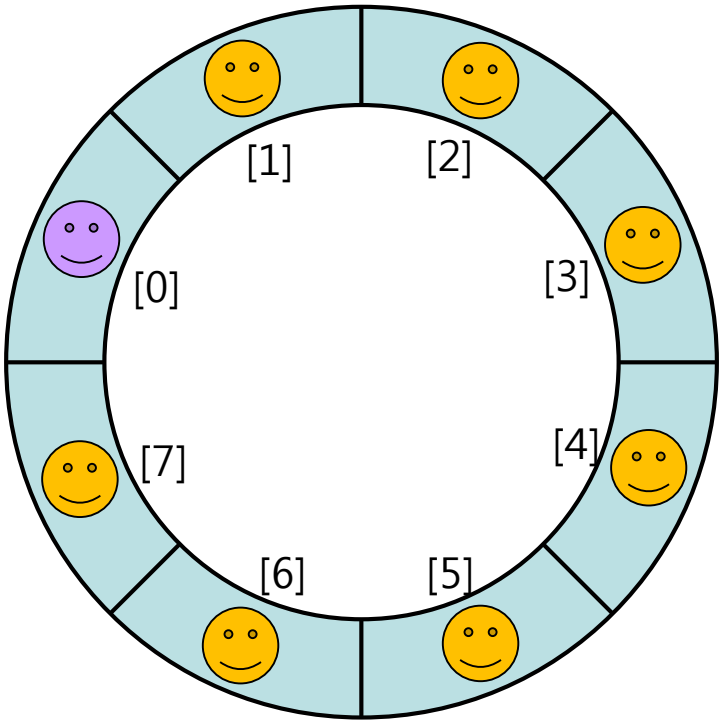
Full 조건 [3]

- 큐 Full 조건은?
  - 초기 시작은,
    - this._front == 0
    - this._rear == 0
  - 6개가 차면,
    - this._front == 0
    - this._rear == 6
  - 7개가 차면,
    - this._front == 0
    - this._rear == 7
  - 8개가 차면,
    - this._front == 0
    - this._rear == 0



Full 조건 [4]

- 큐 Full 조건은?
  - 초기 시작은,
    - this._front == 0
    - this._rear == 0
  - 6개가 차면,
    - this._front == 0
    - this._rear == 6
  - 7개가 차면,
    - this._front == 0
    - this._rear == 7
  - 8개가 차면,
    - this._front == 0
    - this._rear == 0



- 모두 채우면 큐가 empty인지 full인지 구분 불가!

CircularArrayQueue의 공개함수

- CircularArrayQueue 객체 사용법
  - public CircularArrayQueue () ;
  - public CircularArrayQueue (int initialCapacity) ;

  - public boolean isEmpty () ;
  - public boolean isFull () ;
  - public int size () ;

  - public T frontElement () ;

  - public boolean enQueue (T anElement) ;
  - public T deQueue () ;
  - public void clear () ;

Class “CircularArrayQueue” 의 구현

CircularArrayQueue: 비공개 인스턴스 변수

```
public class CircularArrayQueue<T>
{
    // 비공개 인스턴스 변수
    private static final int DEFAULT_INITIAL_CAPACITY = 50 ;
    private int              _maxSize ;
    private int              _front ;
    private int              _rear ;
    private T[]              _elements ;
```

CircularArrayQueue의 생성자

```
public class CircularArrayQueue<T>
{
    // 생성자
    public CircularArrayQueue ()
    {
        this (CircularArrayQueue.DEFAULT_INITIAL_CAPACITY) ;
    }

    public CircularArrayQueue (int initialCapacity)
    {
        @SuppressWarnings("Unchecked") ;
        this._elements = (T[ ]) new Object[initialCapacity] ;
        this._maxSize = initialCapacity ;
        this._front = 0 ;
        this._rear = 0 ;
    }
```

CircularArrayQueue : 상태 알아보기

```
public class CircularArrayQueue<T>
{
    ......

    // 상태 알아보기
    public boolean  isEmpty ()
    {
        return  (this._front == this._rear) ;
    }

    public boolean  isFull ()
    {
        int  nextRear = (this._rear+1) % this._maxSize ;
        return  (nextRear == this._front) ;
    }

    public int  size()
    {
        if ( this._front <= this. rear ) {
            return  (this._rear – this.front)
        }
        else {
            return ( (this._rear+this._maxSize) – this._front ) ;
        }
    }
```

CircularArrayQueue : frontElement()

```
public T  frontElement()
{
    T  frontElement = null ;
    if ( ! this.isEmpty() ) {
        frontElement = this._elements[this._front] ;
    }
    return  frontElement ;
}
```

CircularArrayQueue : enQueue()

```
// 원소 추가 함수
public boolean  enQueue (T anElement)
{
    if ( this.isFull() ) {
        return false ;
    }
    else {
        this._rear = (this._rear+1) % this._maxSize ;
        this._elements[this._rear] = anElement ;
        return  true ;
    }
}
```

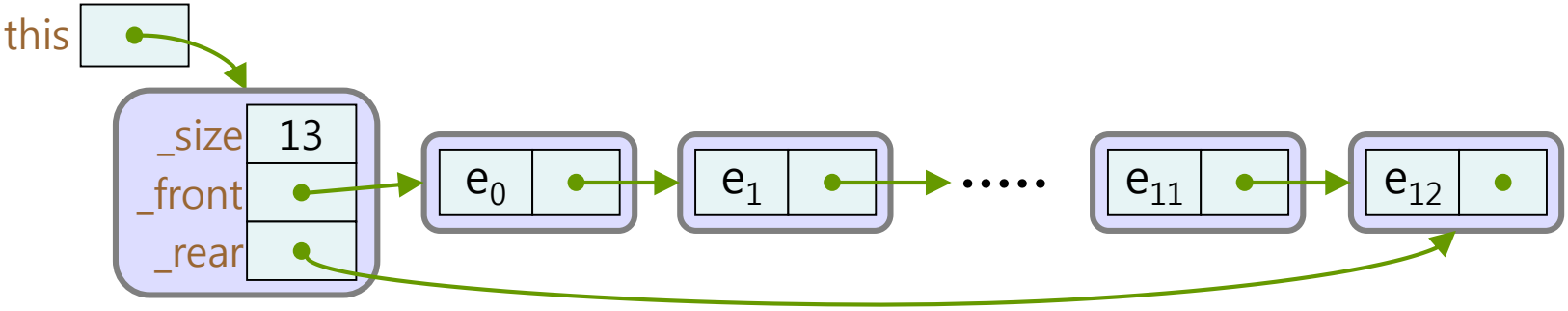
CircularArrayQueue : deQueue()

```
public T  deQueue()
{
    T  frontElement = null ;
    if ( ! this.isEmpty() ) {
        frontElement = _elements[this._front] ;
        this._elements[this._front] = null ;
        this._front = (this._front+1) % this._maxSize ;
    }
    return frontElement ;
}
```

CircularArrayQueue : clear()

```
public void  clear ()
{
    for ( i = 1 ; i <= this.size() ; i++ ) {
        this._elements[(this._front+i) % this._maxSize] = null ;
    }
    this._front = 0 ;
    this._rear = 0 ;
}
```

Class “LinkedQueue”

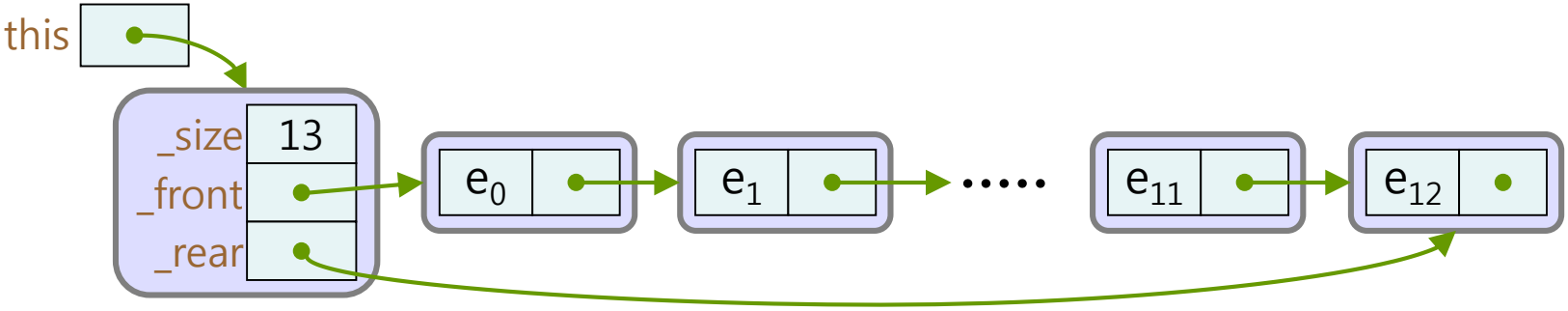


LinkedQueue 의 공개함수

- LinkedQueue 객체 사용법
  - public LinkedQueue() ;
  - public boolean isEmpty() ;
  - public boolean isFull () ;
  - public int size();
  - public T frontElement ();
  - public boolean enQueue (T anElement) ;
  - public T deQueue ();
  - public void clear () ;

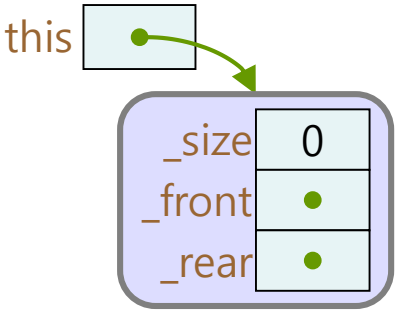
LinkedQueue : 멤버변수

```
public class LinkedQueue<T>
{
    // 비공개 멤버 변수
    private   int      _size ;
    private   Node     _front ;
    private   Node     _rear ;
```



Class "LinkedQueue"의 구현: 생성자

```
public class LinkedQueue<T>
{
    // 비공개 멤버 변수
    ......

    // 생성자
    public LinkedQueue ( )
    {
        this._size = 0 ;
        this._front = null ;
        this._rear = null ;
    }
```

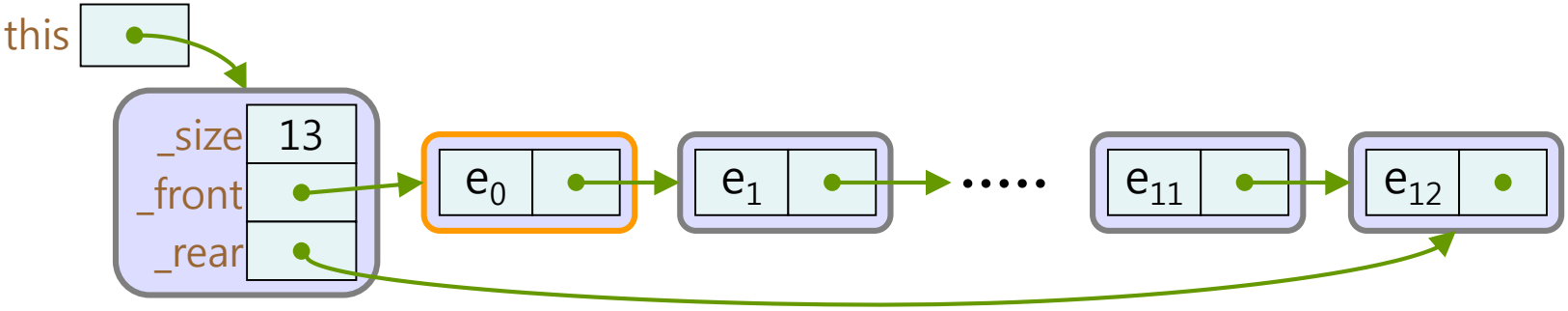


LinkedQueue : 상태 알아보기

```
public class LinkedQueue<T>
{
    // 비공개 멤버 변수
    ......

    // Queue가 비어있는지 확인
    public boolean  isEmpty ()
    {
        return  (this._front== null) && (this._rear == null);
        // 또는 간단히:  return (this._front== null) ;
    }

    public boolean  isFull ()
    {
        return  false ;
    }

    public int  size ()
    {
        return  this._size ;
    }
```

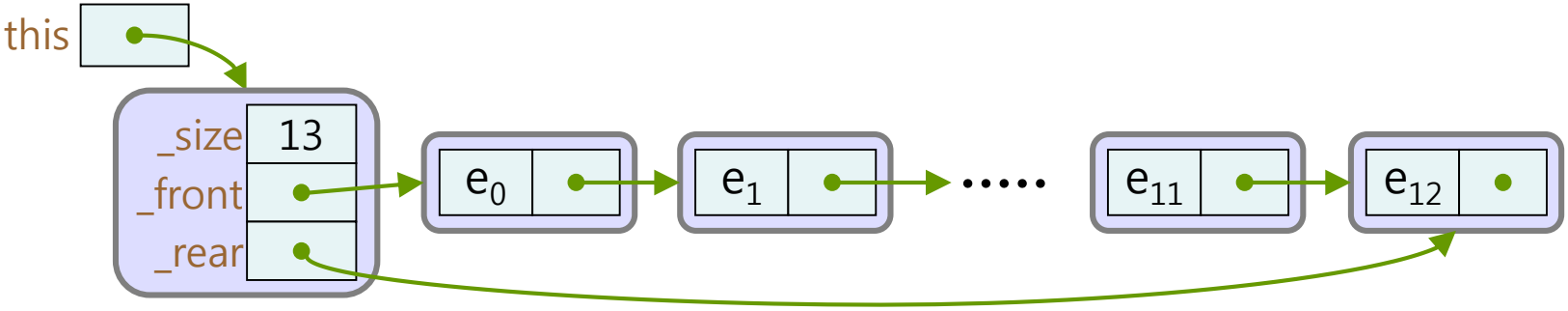
LinkedQueue : frontElement ()

```
public T  frontElement()
{
    T  frontElement = null ;
    if ( ! this.isEmpty() ) {
        frontElement = this._front.element() ;
    }
    return frontElement ;
}
```



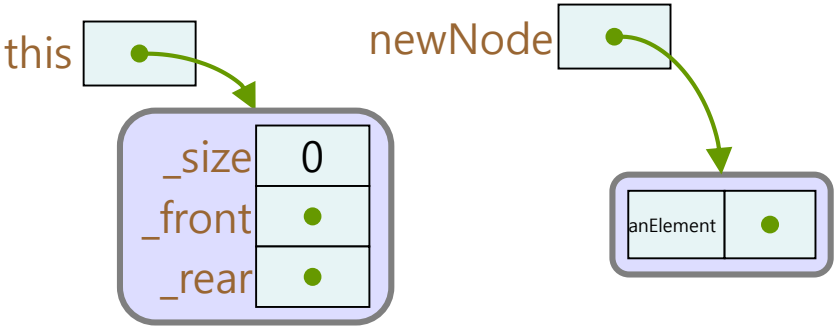
LinkedQueue: enQueue()

```
public void enQueue (T anElement)
{
    Node newNode = new Node (anElement, null) ;
    if ( this.isEmpty() ) {
        this._front = newNode ;
    }
    else {
        this._rear.setNext(newNode) ;
    }
    this._rear = newNode ;
    this._size++ ;
}
```



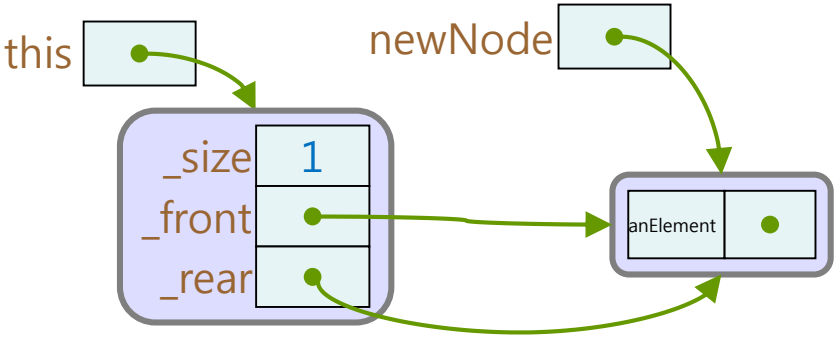
enQueue() [1]

```
public void enQueue (T anElement)
{
    Node newNode = new Node(anElement, null) ;
    if ( this.isEmpty() ) {
        this._front = newNode ;
    }
    else {
        this._rear.setNext(newNode) ;
    }
    this._rear = newNode ;
    this._size++ ;
}
```



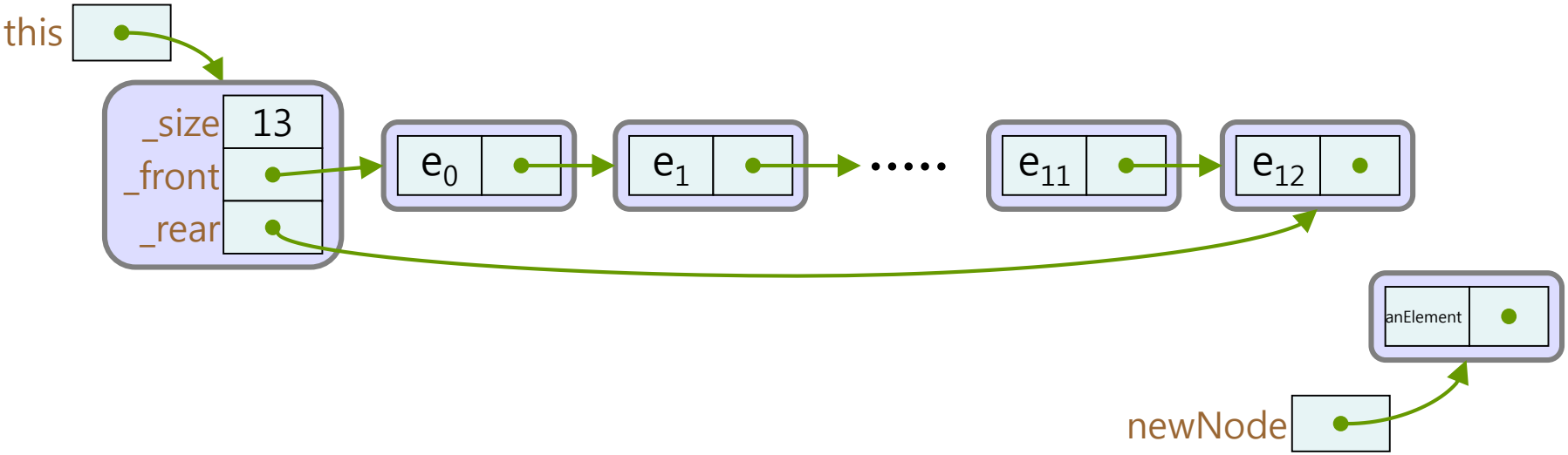
enQueue() [2]

```
public void enQueue (T anElement)
{
    Node newNode = new Node(anElement, null) ;
    if ( this.isEmpty() ) {
        this._front = newNode ;
    }
    else {
        this._rear.setNext(newNode) ;
    }
    this._rear = newNode ;
    this._size++ ;
}
```



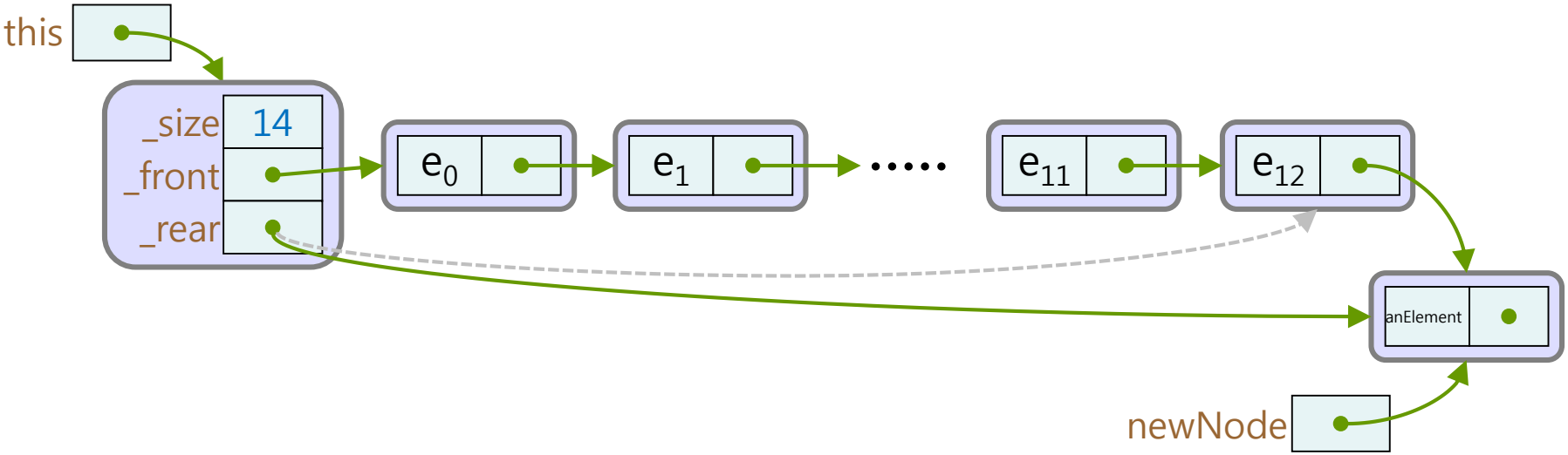
enQueue() [3]

```
public void enQueue (T anElement)
{
    Node newNode = new Node(anElement, null) ;
    if ( this.isEmpty() ) {
        this._front = newNode ;
    }
    else {
        this._rear.setNext(newNode) ;
    }
    this._rear = newNode ;
    this._size++ ;
}
```



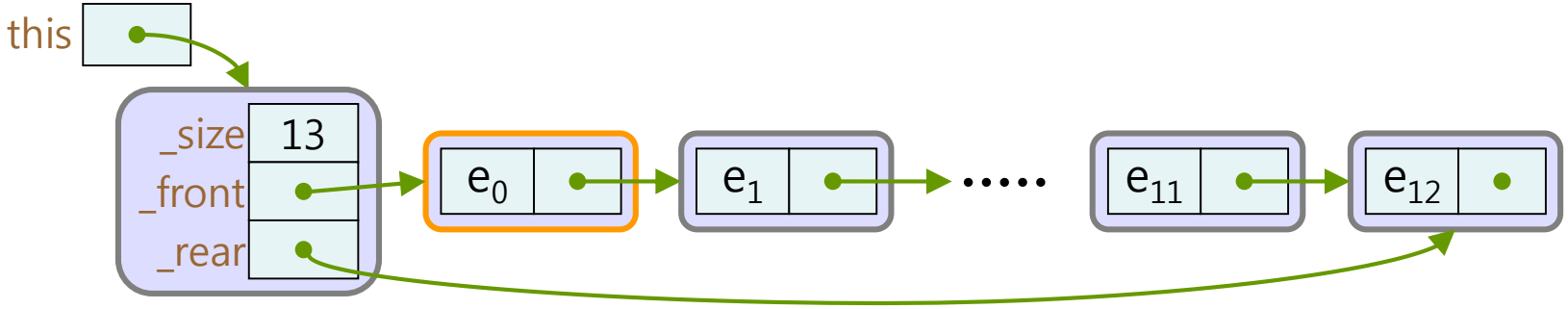
enQueue() [4]

```
public void enQueue (T anElement)
{
    Node newNode = new Node(anElement, null) ;
    if ( this.isEmpty() ) {
        this._front = newNode ;
    }
    else {
        this._rear.setNext(newNode) ;
    }
    this._rear = newNode ;
    this._size++ ;
}
```



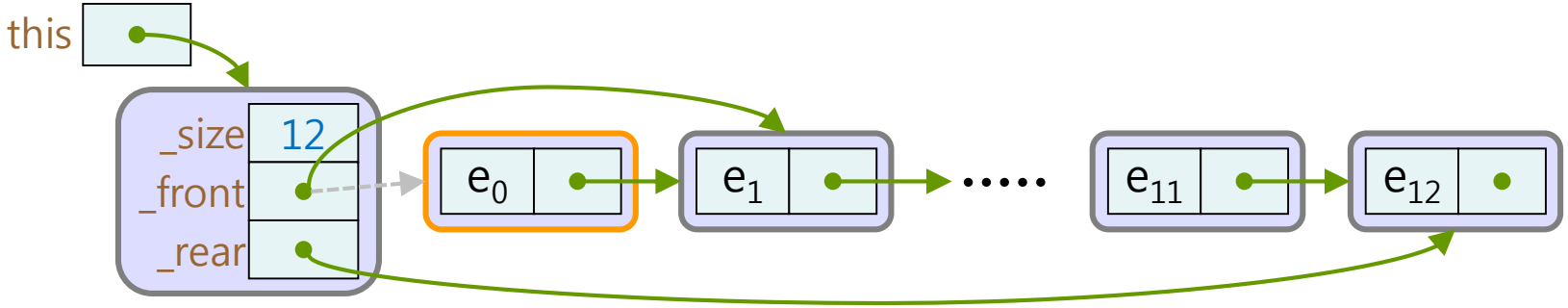
LinkedQueue : deQueue()

```
public T  deQueue()
{
    T  frontElement = null ;
    if ( ! this.isEmpty() )
    {
        frontElement = this._front.element() ;
        this._front = this._front.next() ;
        if ( this._front == null ) {
            this._rear = null;
        }
        this._size--;
    }
    return frontElement ;
}
```



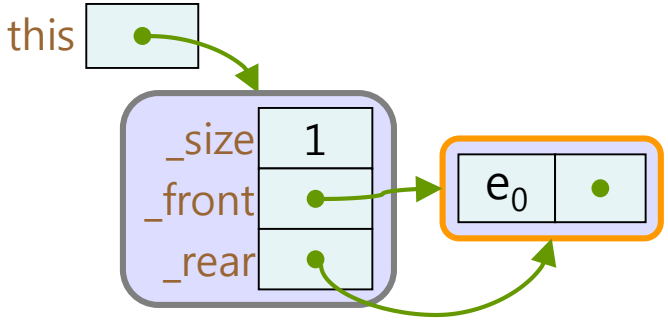
deQueue() [1]

```
public T  deQueue()
{
    T  frontElement = null ;
    if ( ! this.isEmpty() )
    {
        frontElement = this._front.element() ;
        this._front = this._front.next() ;
        if ( this._front == null ) {
            this._rear = null;
        }
        this._size--;
    }
    return frontElement ;
}
```



deQueue() [2]

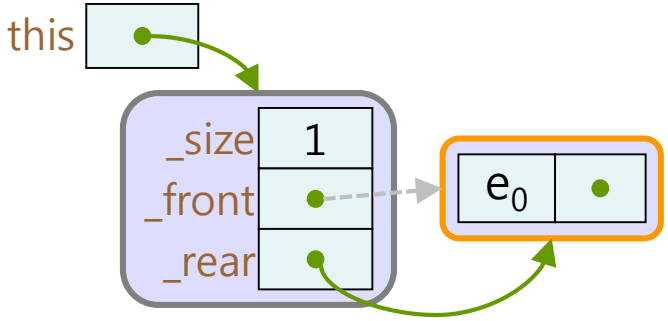
```
public T  deQueue()
{
    T  frontElement = null ;
    if ( ! this.isEmpty() )
    {
        frontElement = this._front.element() ;
        this._front = this._front.next() ;
        if ( this._front == null ) {
            this._rear = null;
        }
        this._size--;
    }
    return frontElement ;
}
```



deQueue() [3]

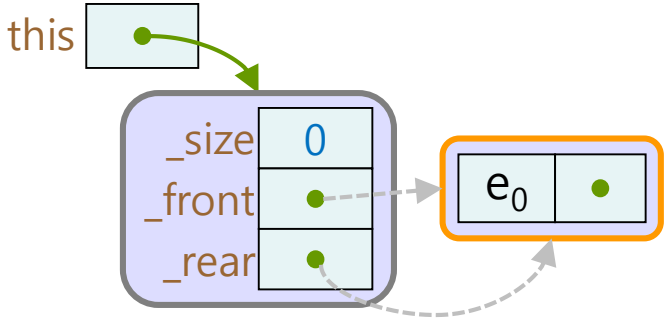
```
public T  deQueue()
{
    T  frontElement = null ;
    if ( ! this.isEmpty() )
    {
        frontElement = this._front.element() ;
        this._front = this._front.next() ;
        if ( this._front == null ) {
            this._rear = null;
        }
        this._size--;
    }
    return frontElement ;
}
```

_front의 값은 null이 된다



deQueue() [4]

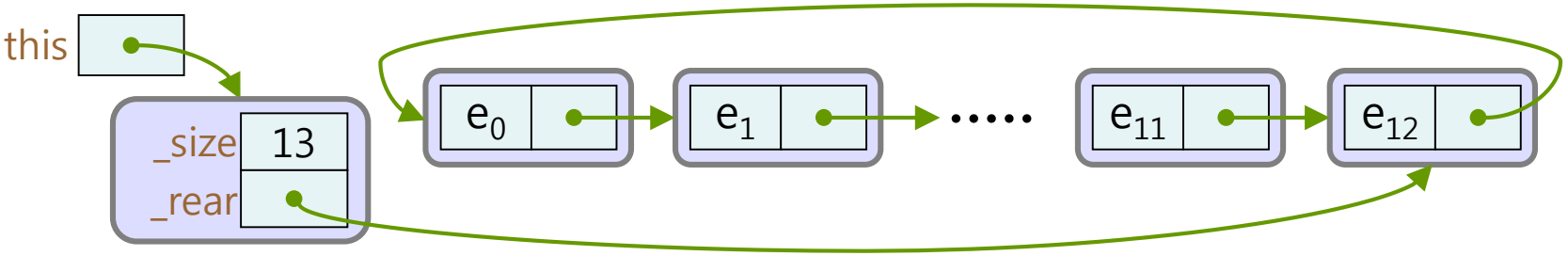
```
public T  deQueue()
{
    T  frontElement = null ;
    if ( ! this.isEmpty() )
    {
        frontElement = this._front.element() ;
        this._front = this._front.next() ;
        if ( this._front == null ) {
            this._rear = null;
        }
        this._size--;
    }
    return frontElement ;
}
```



LinkedQueue : clear()

```
public void clear()
{
    this._front = null ;
    this._rear = null ;
    _size = 0 ;
}
```

Class "CircularlyLinkedQueue"

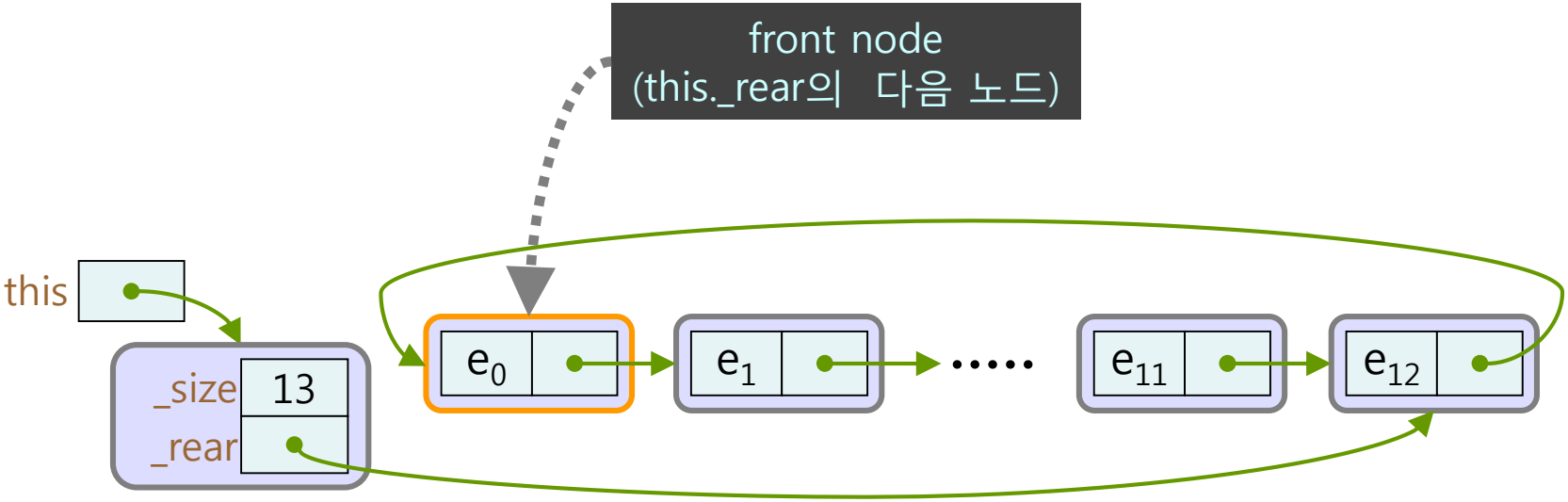


CircularlyLinkedQueue의 공개함수

- CircularlyLinkedQueue 객체 사용법
  - public CircularlyLinkedQueue() ;
  - public boolean isEmpty () ;
  - public boolean isFull () ;
  - public int size () ;
  - public T frontElement ();
  - public boolean enQueue (T anElement) ;
  - public T deQueue ();

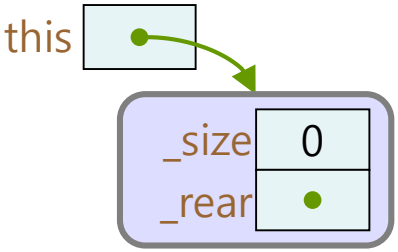
CircularlyLinkedQueue: 비공개 인스턴스 변수

```
public class CircularlyLinkedQueue<T>
{
    // 비공개 인스턴스 변수
    private int        _size ;
    private Node       _rear ;
```



CircularlyLinkedQueue: 생성자

```
public class circularlyLinkedQueue<T>
{
    // 생성자
    public circularlyLinkedQueue( )
    {
        this._size = 0 ;
        this._rear = null ;
    }
```

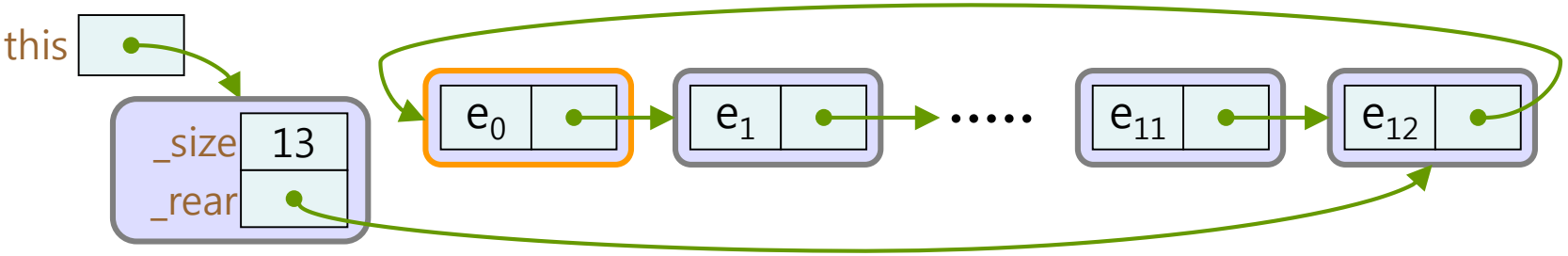


CircularlyLinkedQueue: 상태 알아보기

```
public class circularlyLinkedQueue<T>
{
    ......

    // 상태 알아보기
    public boolean  isEmpty ()
    {
        return ( this._rear ==null ) ;
    }

    public boolean  isFull ()
    {
        return  false ;
    }

    public int  size ()
    {
        return  this._size ;
    }
```

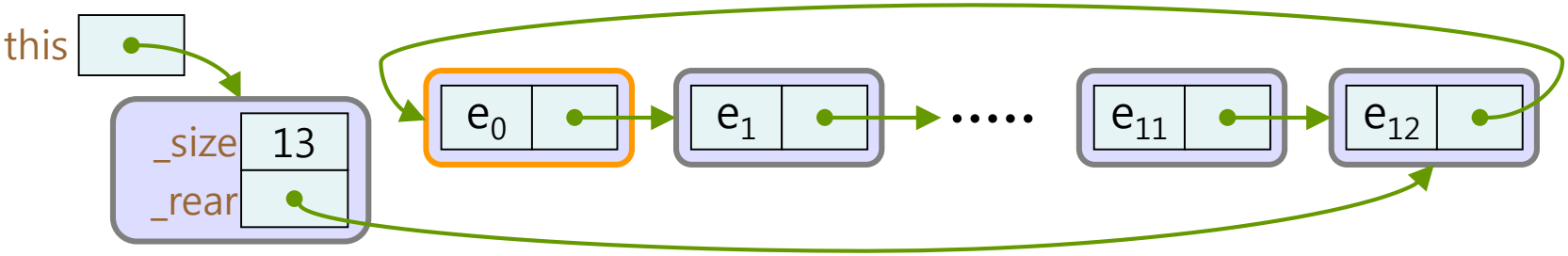
CircularlyLinkedQueue: frontElement()

```
public T  frontElement ()
{
    T  frontElement = null ;
    if ( ! this.isEmpty() ) {
        frontElement = this._rear.next().element() ;
    }
    return frontElement ;
}
```



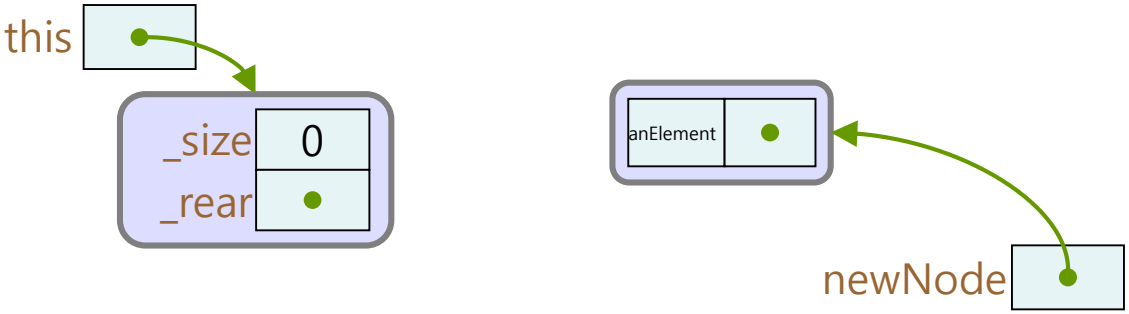
CircularlyLinkedQueue : enQueue()

```
public void enQueue (T anElement)
{
    Node newNode = new Node (anElement, null) ;
    if ( this.isEmpty() ) {
        newNode.setNext (newNode) ;
    }
    else {
        newNode.setNext (this._rear.next()) ;
        this._rear.setNext (newNode) ;
    }
    this._rear= newNode ;
    this._size++ ;
}
```



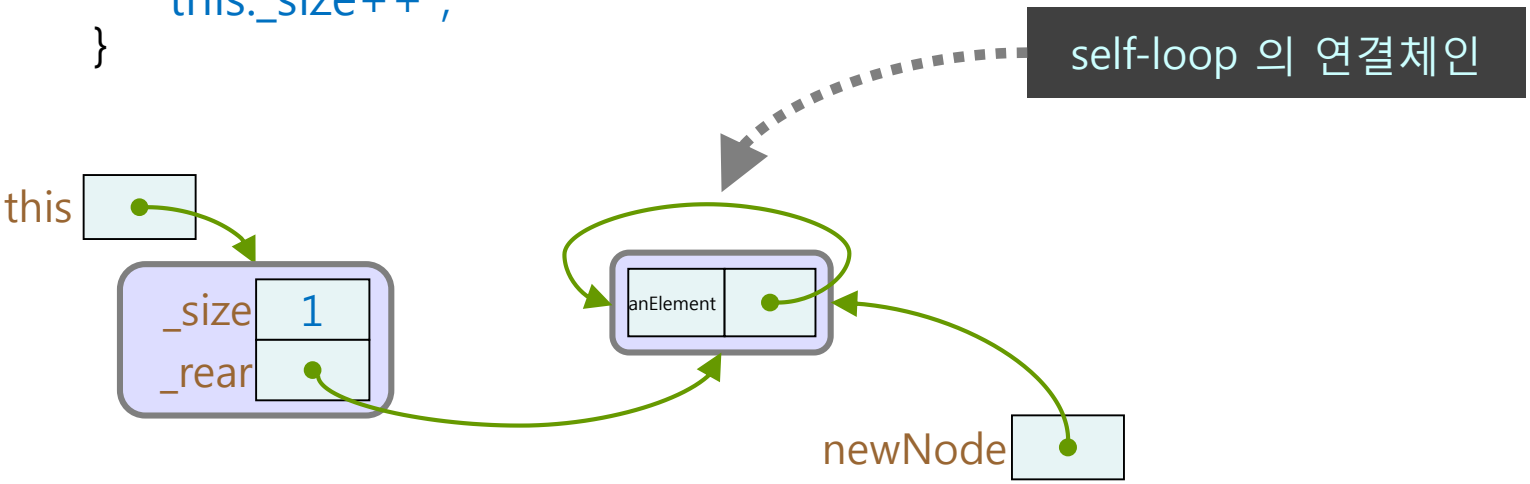
enQueue() [0]

```
public void enQueue (T anElement)
{
    Node newNode = new Node (anElement, null) ;
    if ( this.isEmpty() ) {
        newNode.setNext (newNode) ;
    }
    else {
        newNode.setNext (this._rear.next()) ;
        this._rear.setNext (newNode) ;
    }
    this._rear= newNode ;
    this._size++ ;
}
```



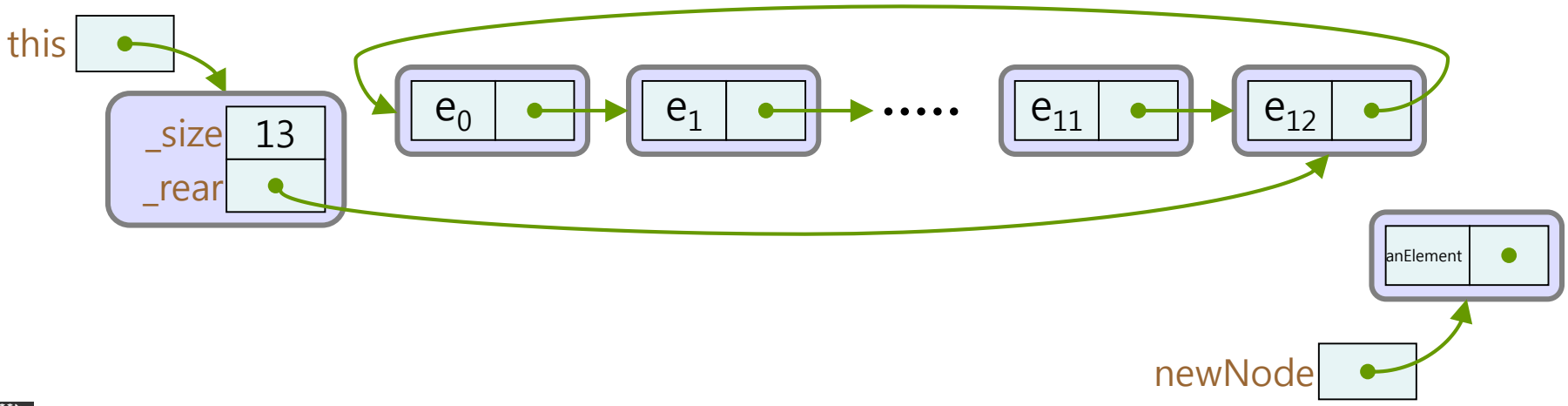
enQueue() [1]

```
public void enQueue (T anElement)
{
    Node newNode = new Node (anElement, null) ;
    if ( this.isEmpty() ) {
        newNode.setNext (newNode) ;
    }
    else {
        newNode.setNext (this._rear.next()) ;
        this._rear.setNext (newNode) ;
    }
    this._rear= newNode ;
    this._size++ ;
}
```



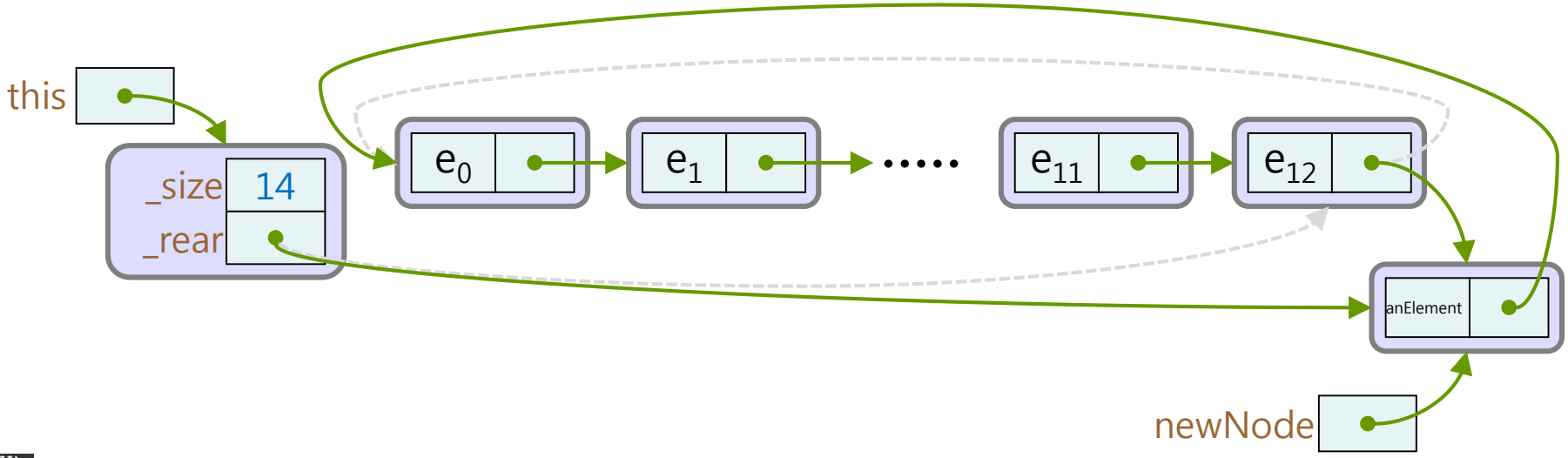
enQueue() [2]

```
public void enQueue (T anElement)
{
    Node newNode = new Node (anElement, null) ;
    if ( this.isEmpty() ) {
        newNode.setNext (newNode) ;
    }
    else {
        newNode.setNext (this._rear.next()) ;
        this._rear.setNext (newNode) ;
    }
    this._rear= newNode ;
    this._size++ ;
}
```



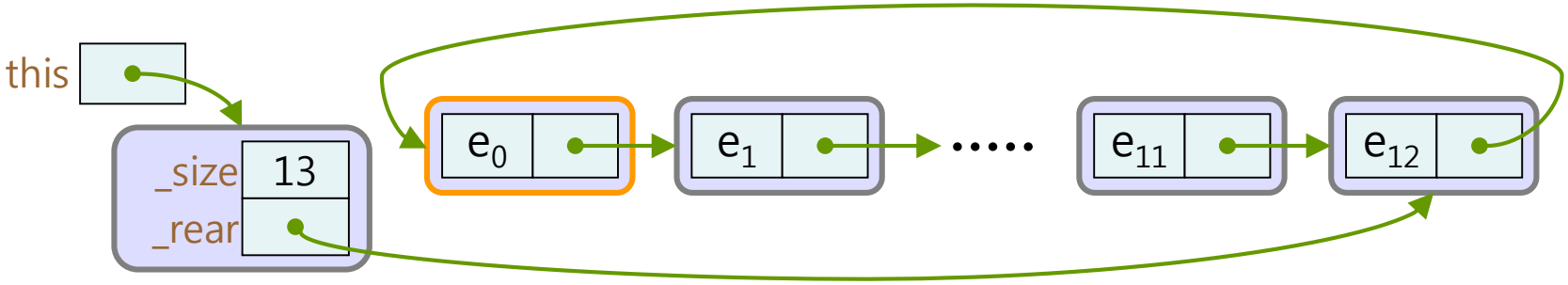
enQueue() [3]

```
public void enQueue (T anElement)
{
    Node newNode = new Node (anElement, null) ;
    if ( this.isEmpty() ) {
        newNode.setNext (newNode) ;
    }
    else {
        newNode.setNext (this._rear.next()) ;
        this._rear.setNext (newNode) ;
    }
    this._rear= newNode ;
    this._size++ ;
}
```



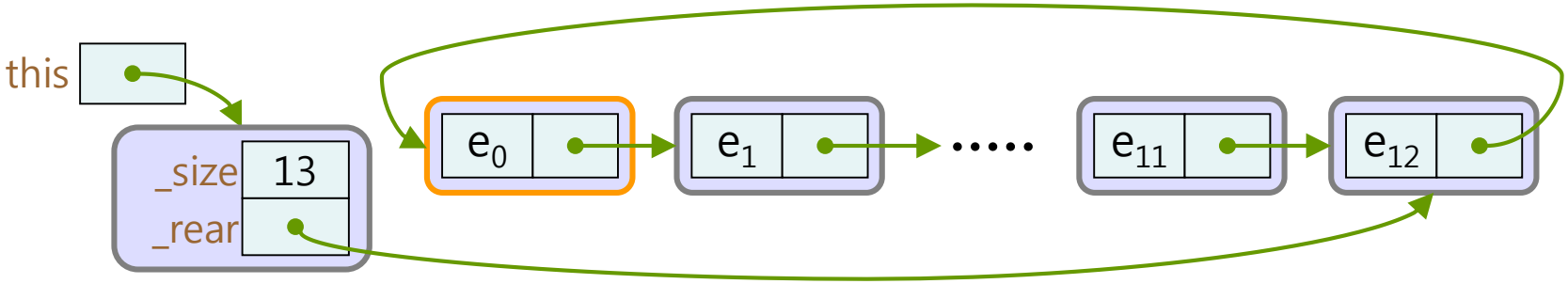
CircularlyLinkedQueue: deQueue()

```
public T  deQueue()
{
    T  frontElement = null ;
    if ( ! this.isEmpty() )
    {
        frontElement = this._rear.next().element() ;
        if ( this._rear == this._rear.next() ) { 노드가 한 개인 self-loop의 경우
            this._rear = null ;
        }
        else { // 노드가 2 개 이상
            this._rear.setNext() = this._rear.next().next() ;
        }
        this._size-- ;
    }
    return frontElement ;
}
```



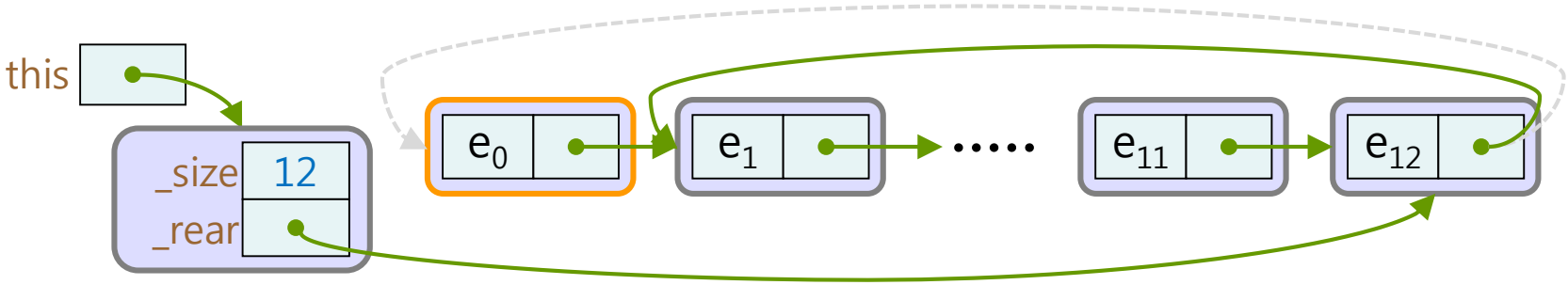
deQueue() [1]

```
public T  deQueue()
{
    T  frontElement = null ;
    if ( ! this.isEmpty() )
    {
        frontElement = this._rear.next().element() ;
        if ( this._rear == this._rear.next() ) { 노드가 한 개인 self-loop의 경우
            this._rear = null ;
        }
        else { // 노드가 2 개 이상
            this._rear.setNext() = this._rear.next().next() ;
        }
        this._size--;
    }
    return frontElement ;
}
```



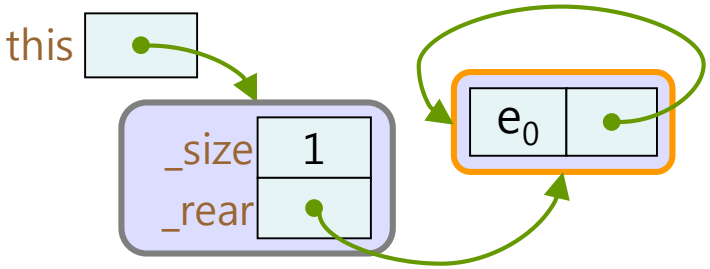
deQueue() [2]

```
public T  deQueue()
{
    T  frontElement = null ;
    if ( ! this.isEmpty() )
    {
        frontElement = this._rear.next().element() ;
        if ( this._rear == this._rear.next() ) { 노드가 한 개인 self-loop의 경우
            this._rear = null ;
        }
        else { // 노드가 2 개 이상
            this._rear.setNext() = this._rear.next().next() ;
        }
        this._size--;
    }
    return  frontElement ;
}
```



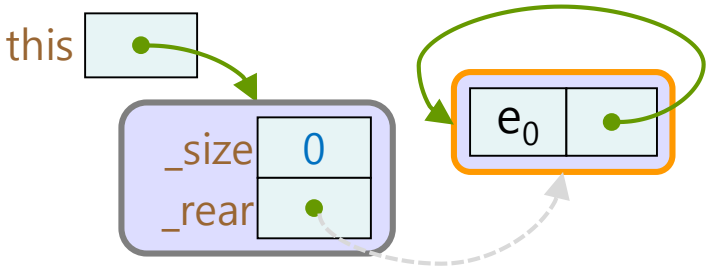
deQueue() [3]

```
public T  deQueue()
{
    T  frontElement = null ;
    if ( ! this.isEmpty() )
    {
        frontElement = this._rear.next().element() ;
        if ( this._rear == this._rear.next() ) { 노드가 한 개인 self-loop의 경우
            this._rear = null ;
        }
        else { // 노드가 2 개 이상
            this._rear.setNext() = this._rear.next().next() ;
        }
        this._size--;
    }
    return  frontElement ;
}
```



deQueue() [4]

```
public T  deQueue()
{
    T  frontElement = null ;
    if ( ! this.isEmpty() )
    {
        frontElement = this._rear.next().element() ;
        if ( this._rear == this._rear.next() ) { 노드가 한 개인 self-loop의 경우
            this._rear = null ;
        }
        else { // 노드가 2 개 이상
            this._rear.setNext() = this._rear.next().next() ;
        }
        this._size--;
    }
    return  frontElement ;
}
```



LinkedQueue : clear()

```
public void clear()
{
    this._front = null ;
    this._rear = null ;
    _size = 0 ;
}
```

실습:
Simulating Waiting Line

실습: Simulating Waiting Line

- Queue를 이용하여 대기열 프로그램을 작성한다.
- 각각 손님마다 일이 처리 되는데 걸리는 시간은 다르다.
- 손님이 왔을 때 대기시간과 현재 대기 인원을 출력한다.

“큐” [끝]