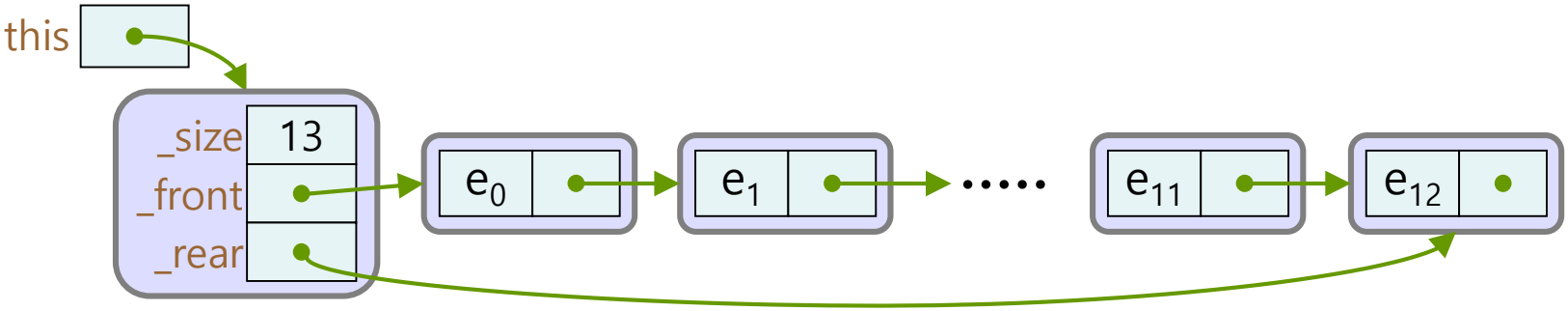
자료구조: 2022년 1학기 [강의]

Linked Queue

강지훈
jhkang@cnu.ac.kr
충남대학교 컴퓨터융합학부

Class "LinkedQueue"

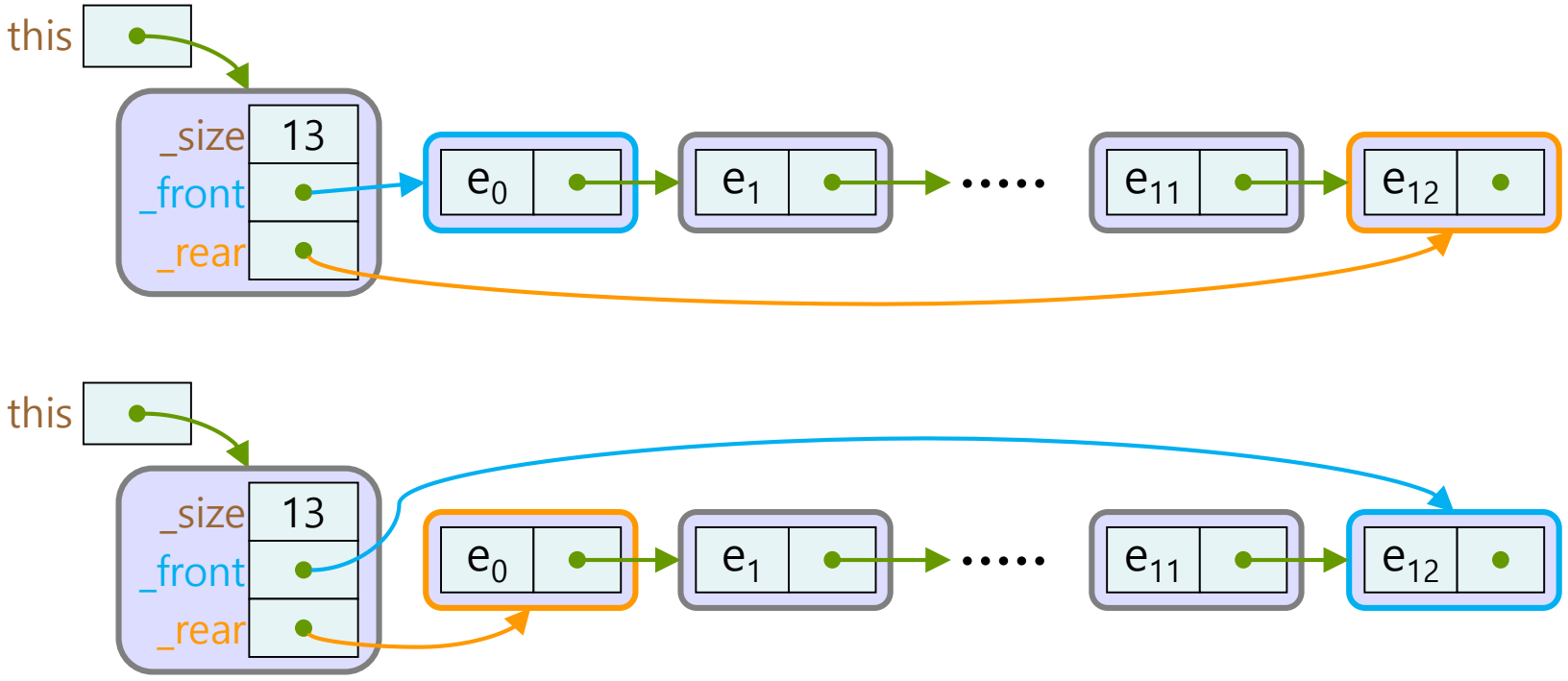


LinkedQueue<T> 의 공개함수

- LinkedQueue<T> 객체 사용법
  - public LinkedQueue() ;
  - public boolean isEmpty() ;
  - public boolean isFull () ;
  - public int size();
  - public T front () ;
  - public T rear () ;
  - public boolean enQueue (T anElement) ;
  - public T deQueue () ;
  - public void clear () ;

연결체인에서의 front 와 rear 는?

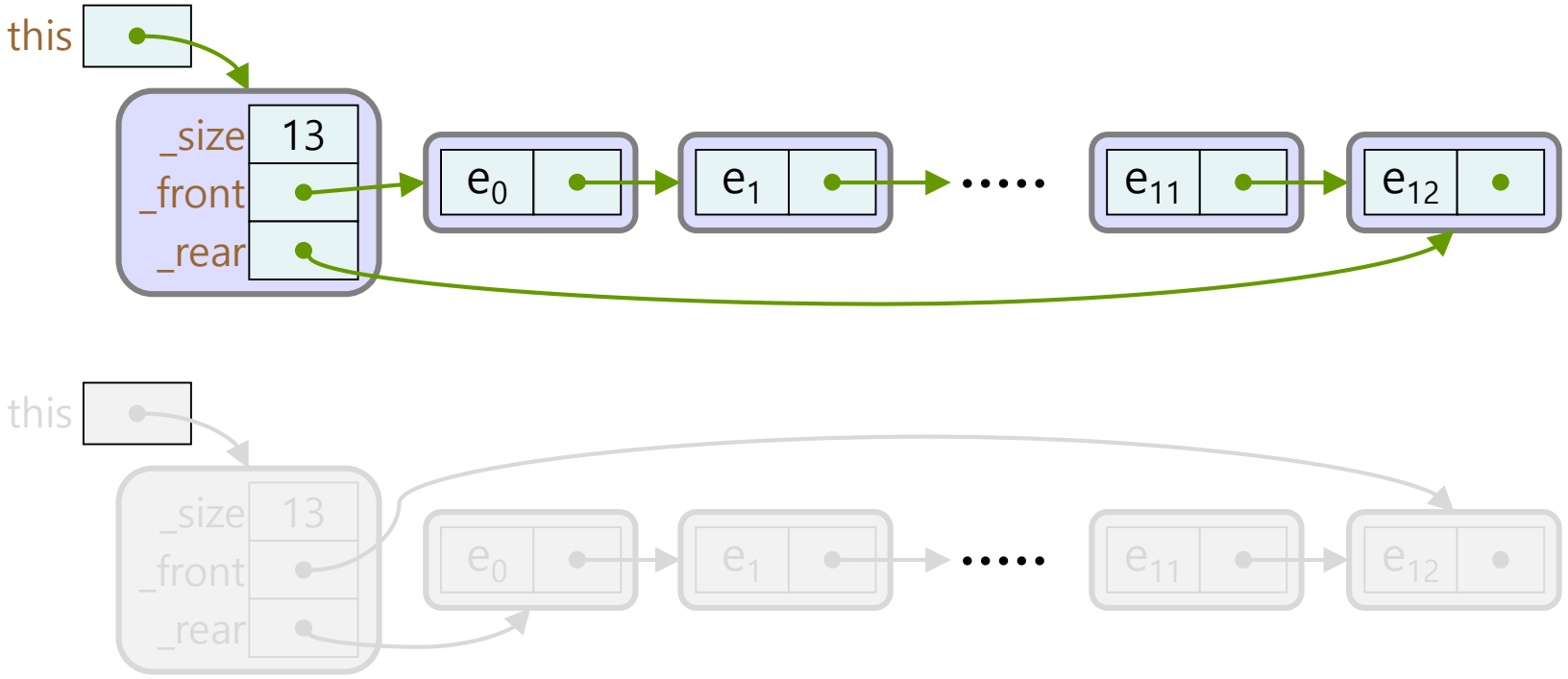
- front 와 rear 는 사용자 관점에서는 추상적.
- Class 내부에서 구체적인 위치를 결정하여 구현해야 함



- 어느 편이 더 효율적인 구현이 될까?

연결체인의 맨 앞을 front 로

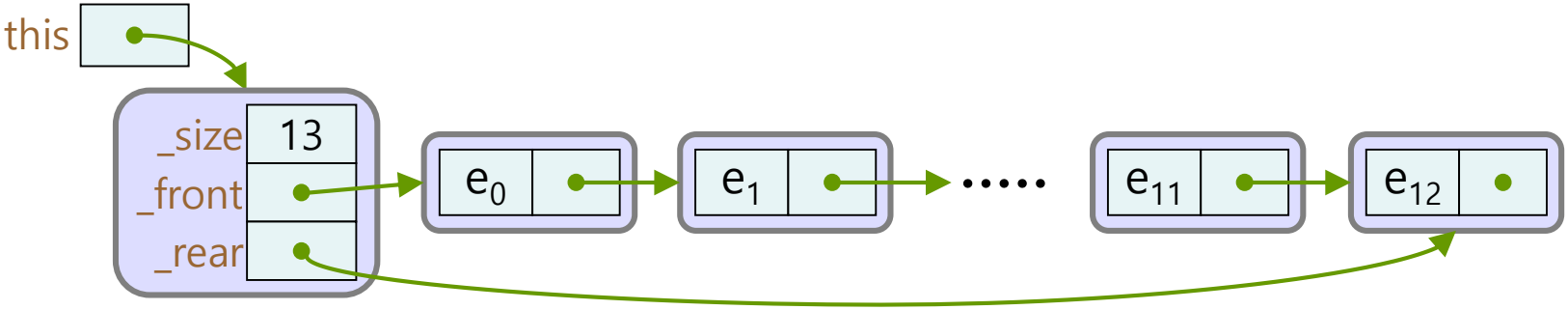
- front 와 rear 는 사용자 관점에서는 추상적.
- Class 내부에서 구체적인 위치를 결정하여 구현해야 함



- 결론:
  - 연결체인의 맨 앞을 front 로 한다.

LinkedQueue<T>: 인스턴스 변수

```
public class  LinkedQueue<T>
{
    // 비공개 인스턴스 변수
    private   int                 _size ;
    private   LinkedNode<T>   _frontNode ;
    private   LinkedNode<T>   _rearNode ;
```



LinkedQueue<T>: Getter/Setter [1]

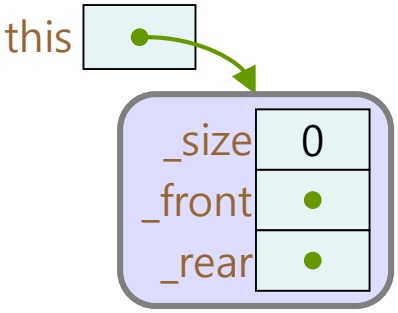
```
public class  LinkedQueue<T>
{
    // 비공개 인스턴스 변수
    private   int                 _size ;
    private   LinkedNode<T>   _frontNode ;
    private   LinkedNode<T>   _rearNode ;

    // Getter/Setter
    public int   size() {
        return   this._size ;
    }
    private void   setSize (int newSize) {
        this._size = newSize ;
    }
    private T   frontNode() {
        return   this._frontNode ;
    }
    private void   setFrontNode (T newFrontNode) {
        this._frontNode = newFrontNode ;
    }
    private T   rearNode() {
        return   this._rearNode ;
    }
    private void   setRearNode (T newRearNode) {
        this._rearNode = newRearNode ;
    }
```

LinkedQueue<T>: Getter/Setter [2]

```
public class  LinkedQueue<T>
{
    // 비공개 인스턴스 변수
    private   int                 _size ;
    private   LinkedNode<T>   _frontNode ;
    private   LinkedNode<T>   _rearNode ;

    // Getter/Setter
    public int   size() {
        return   this._size ;
    }
    private void   setSize (int newSize) {
        this._size = newSize ;
    }
    private T   frontNode() {
        return   this._frontNode ;
    }
    private void   setFrontNode (T newFrontNode) {
        this._frontNode = newFrontNode ;
    }
    private T   rearNode() {
        return   this._rearNode ;
    }
    private void   setRearNode (T newRearNode) {
        this._rearNode = newRearNode ;
    }
```

LinkedQueue<T>: 생성자

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

    // 생성자
    public  LinkedQueue ( )
    {
        this.setSize (0) ;
        this.setFrontNode (null) ;
        this.setRearNode  (null) ;
    }
```

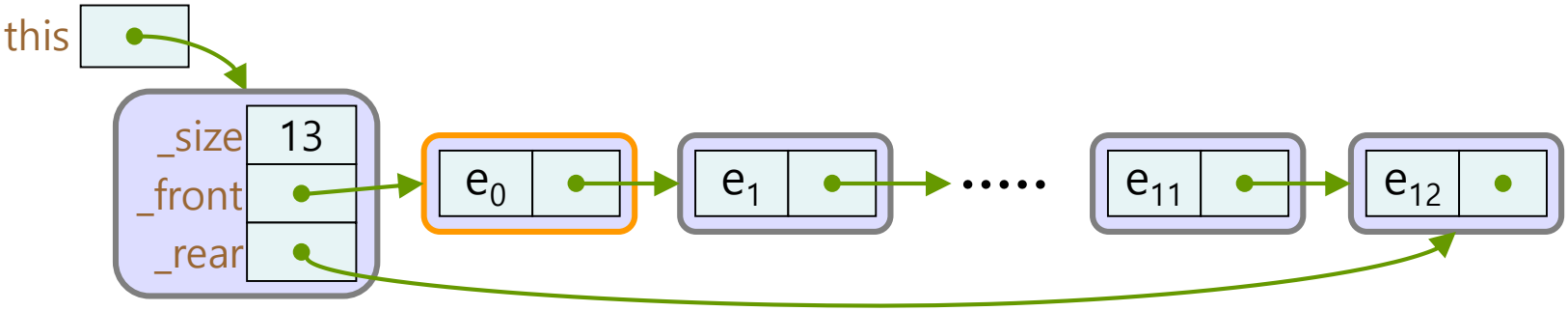


LinkedQueue<T>: 상태 알아보기

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

    // Queue가 비어있는지 확인
    public boolean  isEmpty () {
        return  (this.frontNode() == null) && (this.rearNode() == null) ;
        // 또는 간단히:  return (this.frontNode() == null) ;
        // 또는: return (this.size() == 0) ;
    }

    public boolean  isFull () {
        return  false ;
    }

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

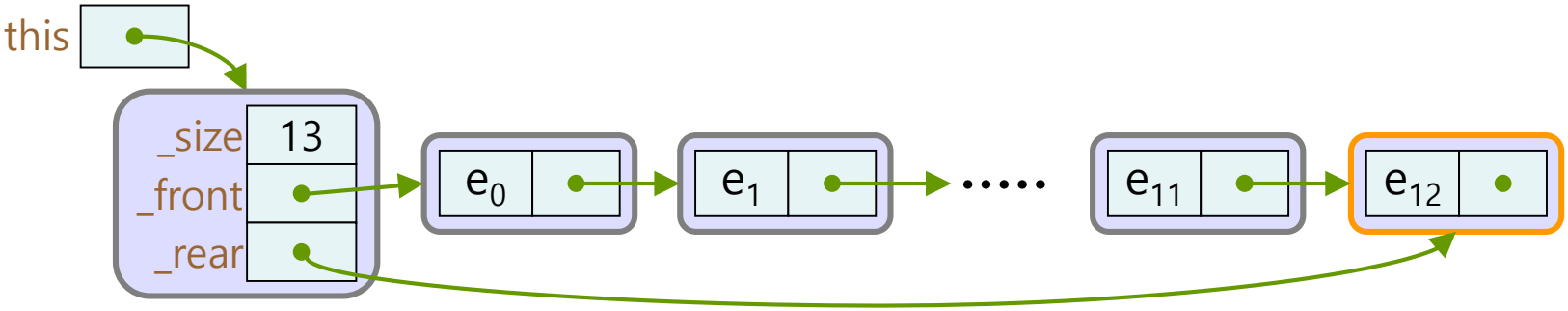
LinkedQueue<T>: front ()

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



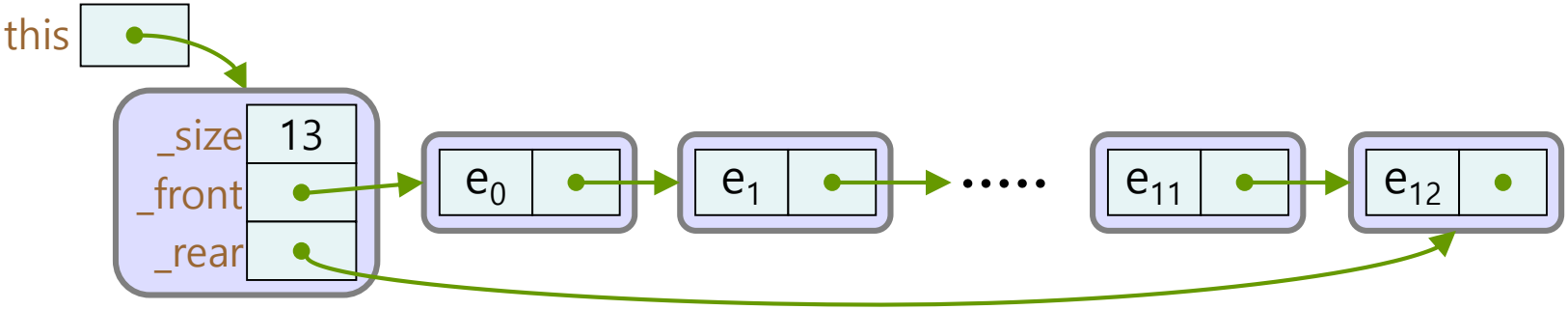
LinkedQueue<T>: rear ()

```
public T  rear()
{
    T  rearElement = null ;
    if ( ! this.isEmpty() ) {
        rearElement = this.rearNode().element() ;
    }
    return  rearElement ;
}
```



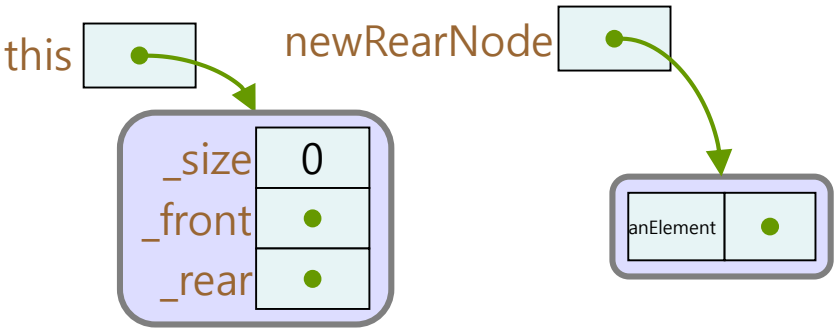
LinkedQueue<T>: enQueue()

```
public void  enQueue (T anElement)
{
    LinkedNode<T> newRearNode = new LinkedNode<T>(anElement, null)
;
    if ( this.isEmpty() ) {
        this.setFrontNode (newRearNode) ;
    }
    else {
        this.setRearNode.setNext (newRearNode) ;
    }
    this.setRearNode (newRearNode) ;
    this.setSize (this.size()+1) ;
}
```



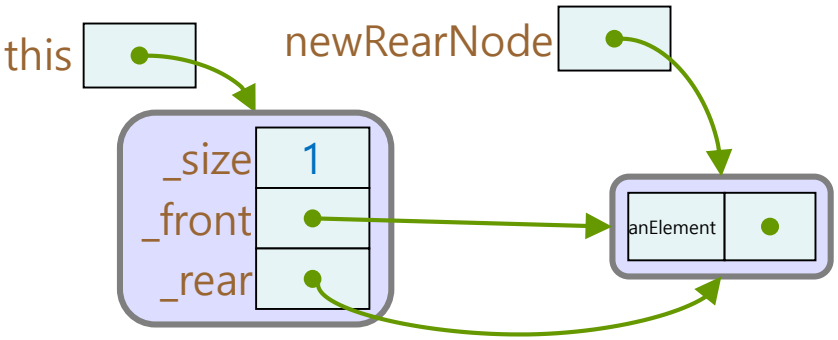
enQueue() [1]

```
public void enQueue (T anElement)
{
    LinkedNode<T> newRearNode = new LinkedNode<T>(anElement, null)
;
    if ( this.isEmpty() ) {
        this.setFrontNode (newRearNode) ;
    }
    else {
        this.rearNode().setNext (newRearNode) ;
    }
    this.setRearNode (newRearNode) ;
    this.setSize (this.size()+1) ;
}
```



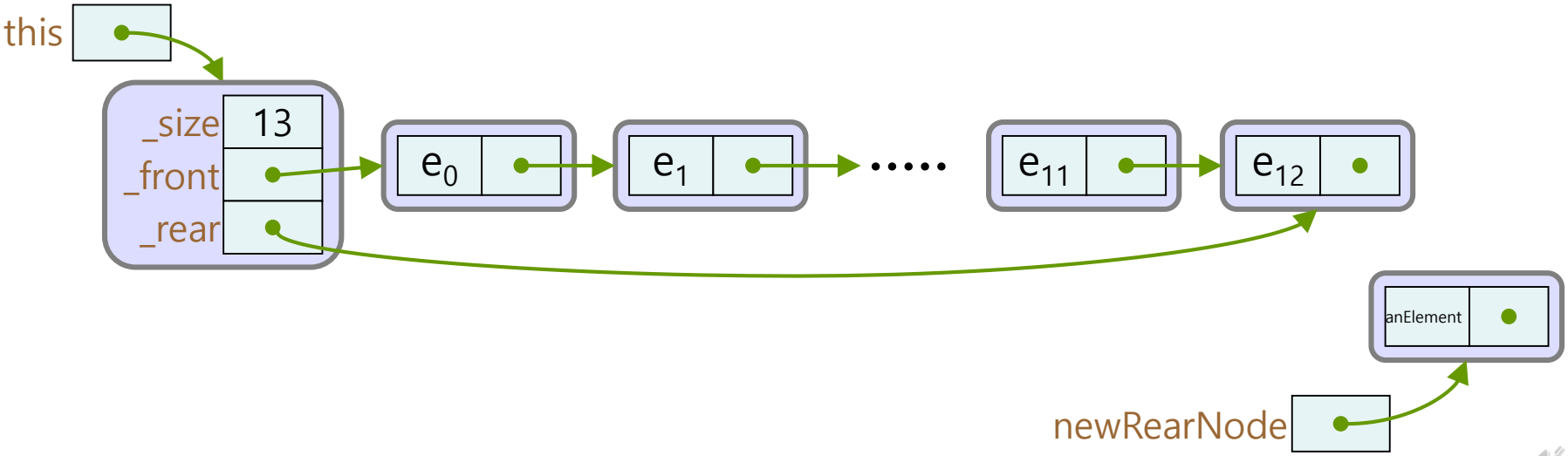
enQueue() [2]

```
public void  enQueue (T anElement)
{
    LinkedNode<T> newRearNode = new LinkedNode<T>(anElement, null) ;
    if ( this.isEmpty() ) {
        this.setFrontNode (newRearNode) ;
    }
    else {
        this.rearNode().setNext (newRearNode) ;
    }
    this.setRearNode (newRearNode) ;
    this.setSize (this.size()+1) ;
}
```



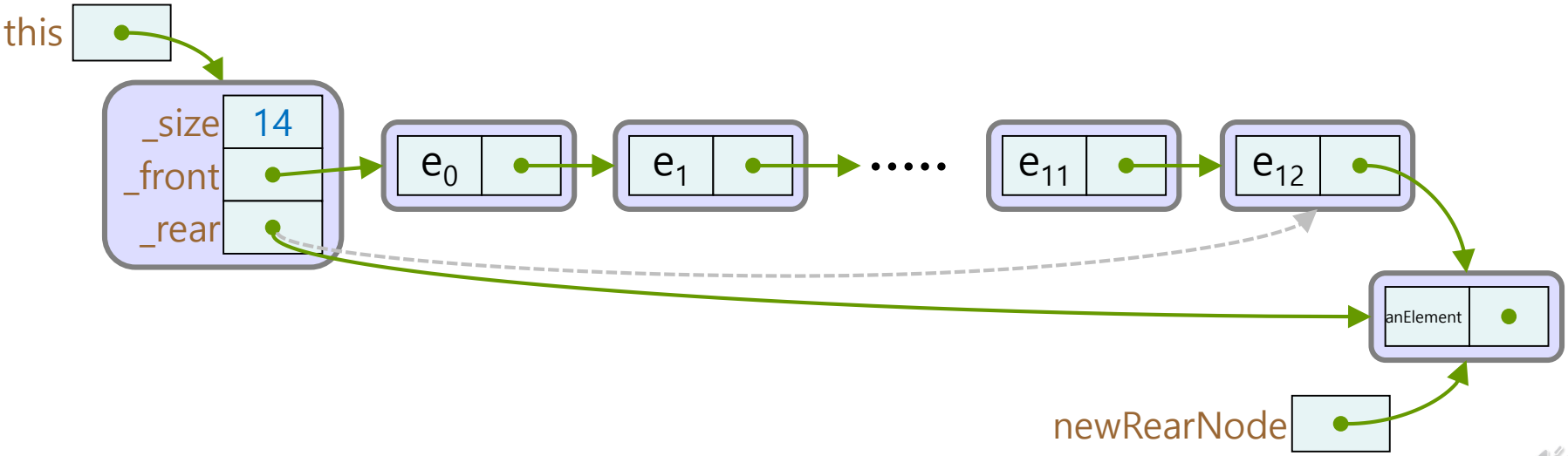
enQueue() [3]

```
public void  enQueue (T anElement)
{
    LinkedNode<T> newRearNode = new LinkedNode<T>(anElement, null) ;
    if ( this.isEmpty() ) {
        this.setFrontNode (newRearNode) ;
    }
    else {
        this.rearNode().setNext (newRearNode) ;
    }
    this.setRearNode (newRearNode) ;
    this.setSize (this.size()+1) ;
}
```



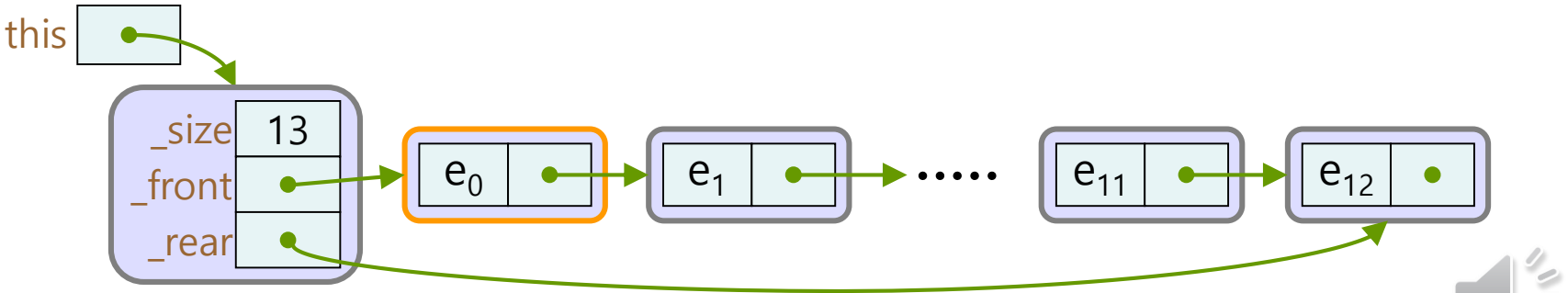
enQueue() [4]

```
public void  enQueue (T anElement)
{
    LinkedNode<T> newRearNode = new LinkedNode<T>(anElement, null) ;
    if ( this.isEmpty() ) {
        this.setFrontNode (newRearNode) ;
    }
    else {
        this.rearNode().setNext (newRearNode) ;
    }
    this.setRearNode (newRearNode) ;
    this.setSize (this.size()+1) ;
}
```



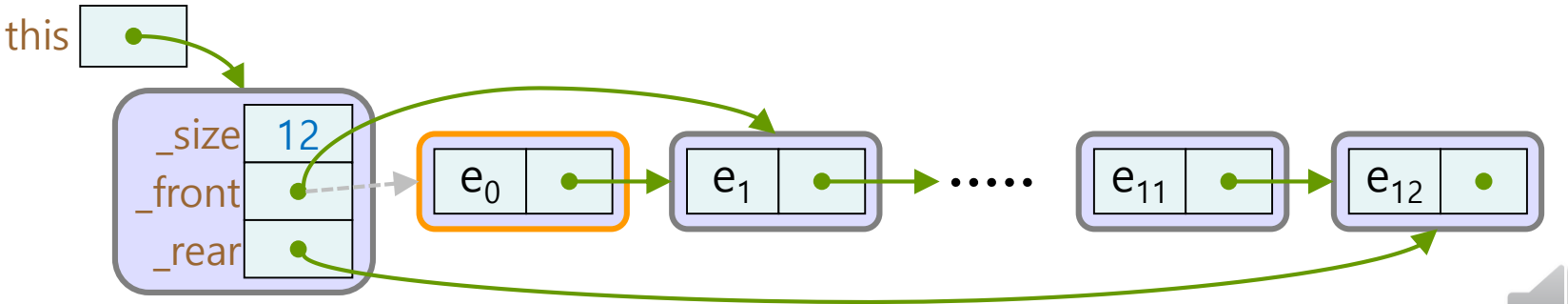
LinkedQueue: deQueue()

```
public T  deQueue()
{
    T  frontElement = null ;
    if ( ! this.isEmpty() ) {
        frontElement = this.frontNode().element() ;
        this.setFrontNode (this.frontNode().next()) ;
        if ( this.frontNode() == null ) {
            this.setRearNode (null) ;
        }
        this.setSize (this.size()-1) ;
    }
    return  frontElement ;
}
```



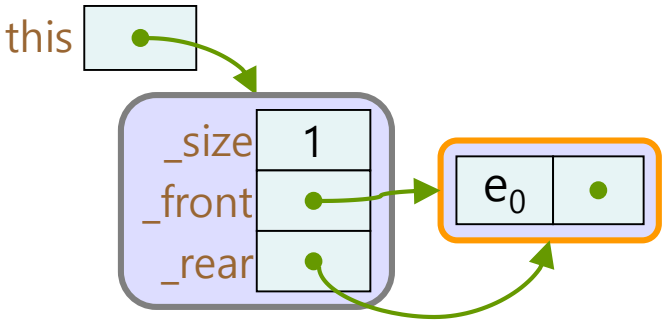
deQueue() [1]

```
public T  deQueue()
{
    T  frontElement = null ;
    if ( ! this.isEmpty() ) {
        frontElement = this.frontNode().element() ;
        this.setFrontNode (this.frontNode().next()) ;
        if ( this.frontNode() == null ) {
            this.setRearNode (null) ;
        }
        this.setSize (this.size()-1) ;
    }
    return  frontElement ;
}
```



deQueue() [2]

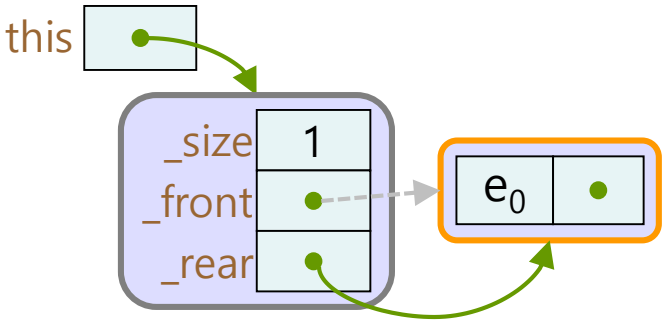
```
public T  deQueue()
{
    T  frontElement = null ;
    if ( ! this.isEmpty() ) {
        frontElement = this.frontNode().element() ;
        this.setFrontNode (this.frontNode().next()) ;
        if ( this.frontNode() == null ) {
            this.setRearNode (null) ;
        }
        this.setSize (this.size()-1) ;
    }
    return  frontElement ;
}
```



deQueue() [3]

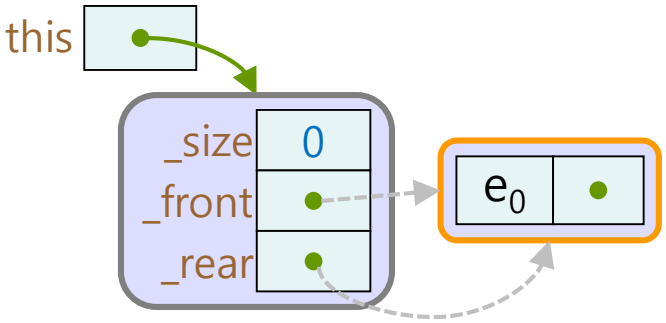
```
public T  deQueue()
{
    T  frontElement = null ;
    if ( ! this.isEmpty() ) {
        frontElement = this.frontNode().element() ;
        this.setFrontNode (this.frontNode().next()) ;
        if ( this.frontNode() == null ) {
            this.setRearNode (null) ;
        }
        this.setSize (this.size()-1) ;
    }
    return  frontElement ;
}
```

현재 큐의 _size 가 1 인데 삭제가 되면 _frontNode 의 값은 null 이 된다



deQueue() [4]

```
public T  deQueue()
{
    T  frontElement = null ;
    if ( ! this.isEmpty() ) {
        frontElement = this.frontNode().element() ;
        this.setFrontNode (this.frontNode().next()) ;
        if ( this.frontNode() == null ) {
            this.setRearNode (null) ;
        }
        this.setSize (this.size()-1) ;
    }
    return  frontElement ;
}
```



LinkedQueue : clear()

```
public void  clear()
{
    this.setFrontNode (null) ;
    this.setRearNode  (null) ;
    this.setSize (0) ;
}
```

Class "LinkedNode<T>"

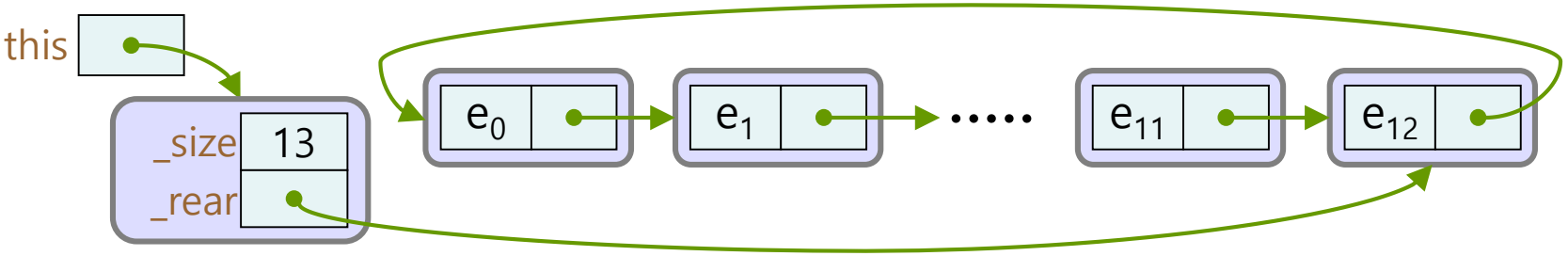
LinkedNode: Instance variables, Getters/Setters

```
public class LinkedNode<T> {
    // Private instance variables
    private T                   _element ;
    private LinkedNode<T>       _next ;

    // Getters/Setters
    public T element() {
        return this._element ;
    }
    public void setElement (T newElement) {
        this._element = newElement ;
    }

    public LinkedNode<T> next() {
        return  this._next ;
    }
    public void setNext (LinkedNode<T> newNext) {
        this._next = newNext ;
    }
```

LinkedNode: Constructors

```
public class LinkedNode<T> {
    // Private instance variables
    ......
    // Getters/Setters
    ......

    // Constructors
    public LinkedNode() {
        this.setElement (null) ;
        this.setNext (null) ;
    }

    public LinkedNode (T givenElement, LinkedNode<T> givenNext) {
        this.setElement (givenElement) ;
        this.setNext (givenNext) ;
    }
}
```

Class "CircularlyLinkedQueue"

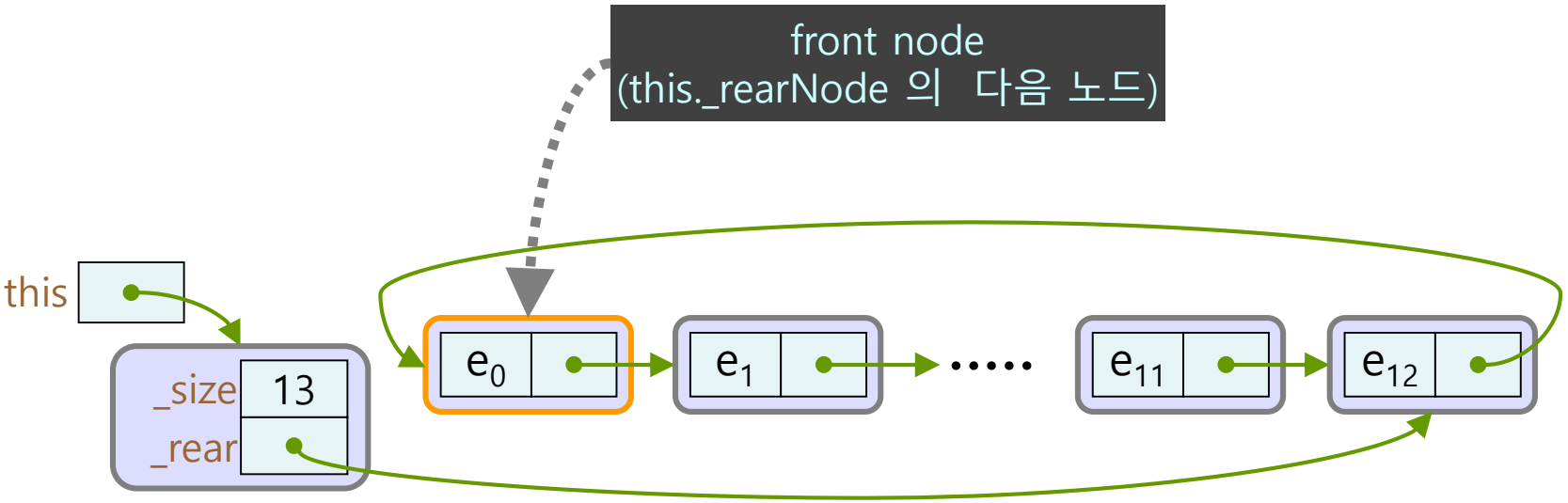


CircularlyLinkedQueue 의 공개함수

- CircularlyLinkedQueue<T> 객체 사용법
  - public CircularlyLinkedQueue() ;
  - public boolean isEmpty () ;
  - public boolean isFull () ;
  - public int size () ;
  - public T front () ;
  - public T rear () ;
  - public boolean enQueue (T anElement) ;
  - public T deQueue () ;
  - public void clear () ;

CircularlyLinkedQueue: 비공개 인스턴스 변수

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

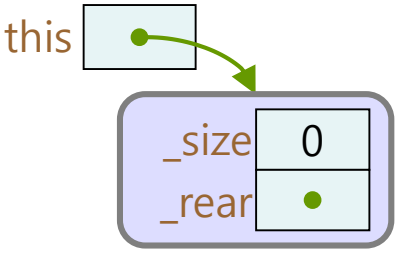


CircularlyLinkedQueue: Getter/Setter

```
public class  LinkedQueue<T>
{
    // 비공개 인스턴스 변수
    private int                 _size ;
    private LinkedNode<T>  _rearNode ;

    // Getter/Setter
    public int  size() {
        return  this._size ;
    }
    private void  setSize (int newSize) {
        this._size = newSize ;
    }
    private T  rearNode() {
        return  this._rearNode ;
    }
    private void  setRearNode (T newRearNode) {
        this._rearNode = newRearNode ;
    }
```

CircularlyLinkedQueue: 생성자

```
public class  CircularlyLinkedQueue<T>
{
    // 생성자
    public  circularlyLinkedQueue( )
    {
        this.setSize (0) ;
        this.setRearNode (null) ;
    }
```

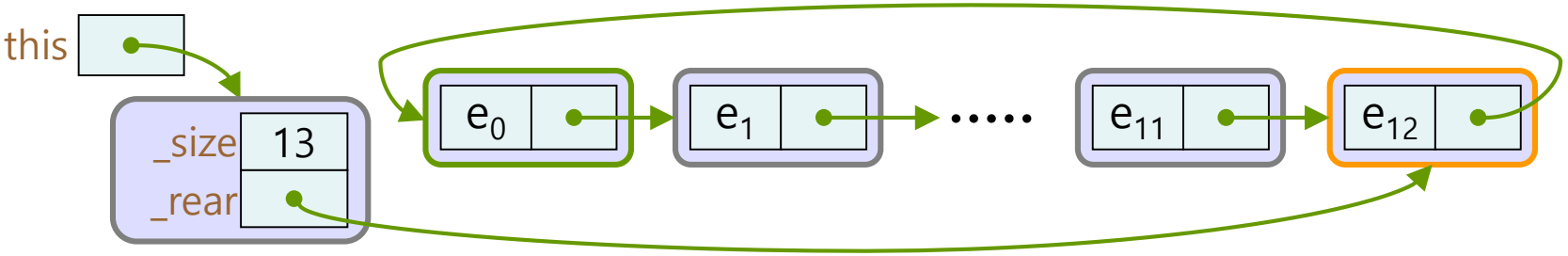


CircularlyLinkedQueue: 상태 알아보기

```
public class  CircularlyLinkedQueue<T>
{
    ……

    // 상태 알아보기
    public boolean  isEmpty ()
    {
        return ( this.rearNode() == null ) ; // 또는 return (this.size == 0) ;
    }

    public boolean  isFull ()
    {
        return  false ;
    }

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

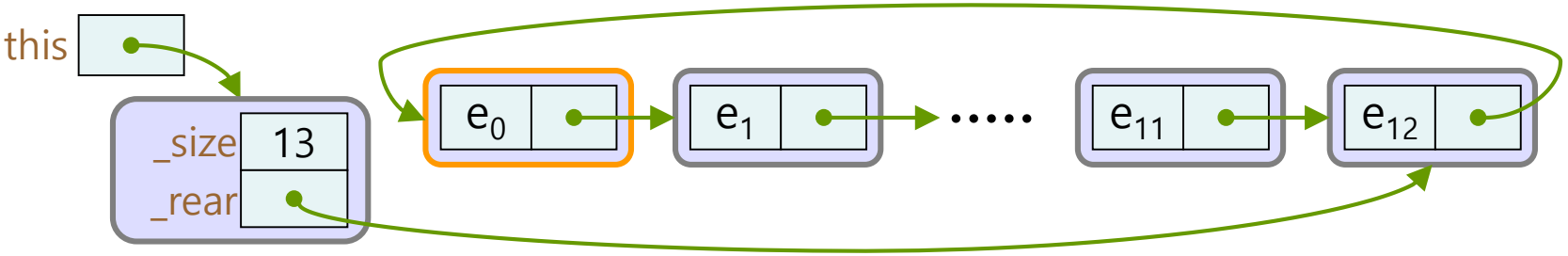
CircularlyLinkedQueue: rear()

```
public T  rear ()
{
    T  rearElement = null ;
    if ( ! this.isEmpty() ) {
        rearElement = this.rearNode().element() ;
    }
    return  rearElement ;
}
```



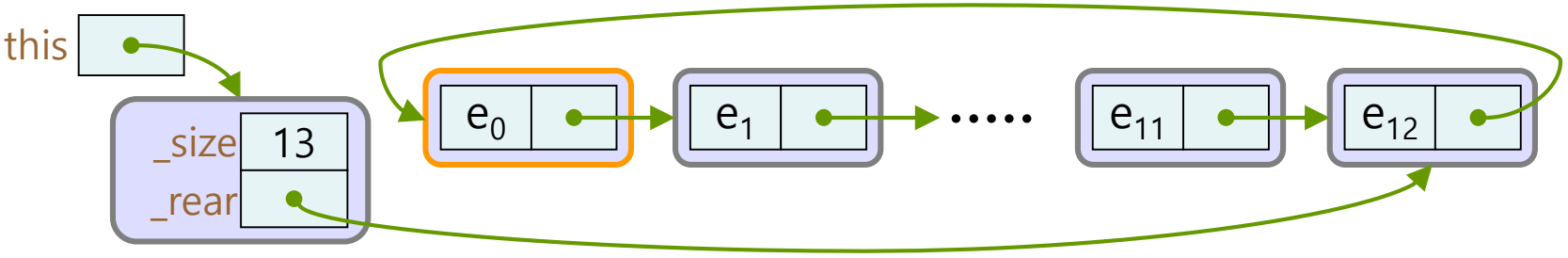
CircularlyLinkedQueue: front()

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



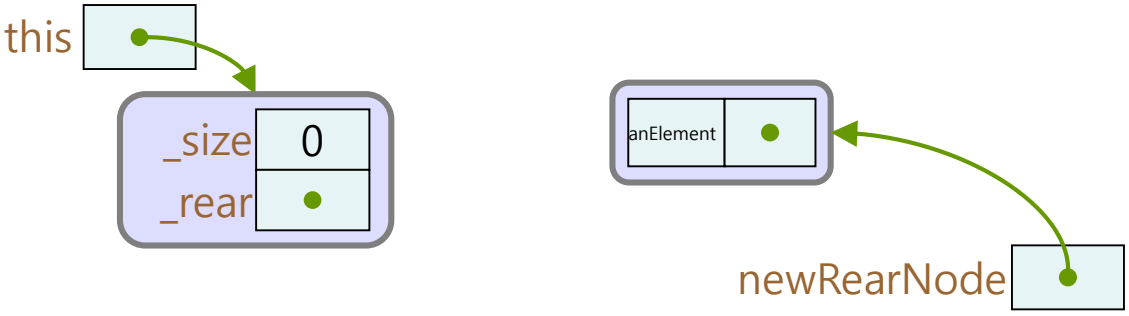
CircularlyLinkedQueue: enQueue()

```
public void  enQueue (T anElement)
{
    LinkedNode<T> newRearNode = new LinkedNode (anElement, null) ;
    if ( this.isEmpty() ) {
        newRearNode.setNext (newRearNode) ;
    }
    else {
        newRearNode.setNext (this.rearNode().next()) ;
        this.rearNode().setNext (newRearNode) ;
    }
    this.setRearNode (newRearNode) ;
    this.setSize (this.size()+1) ;
}
```



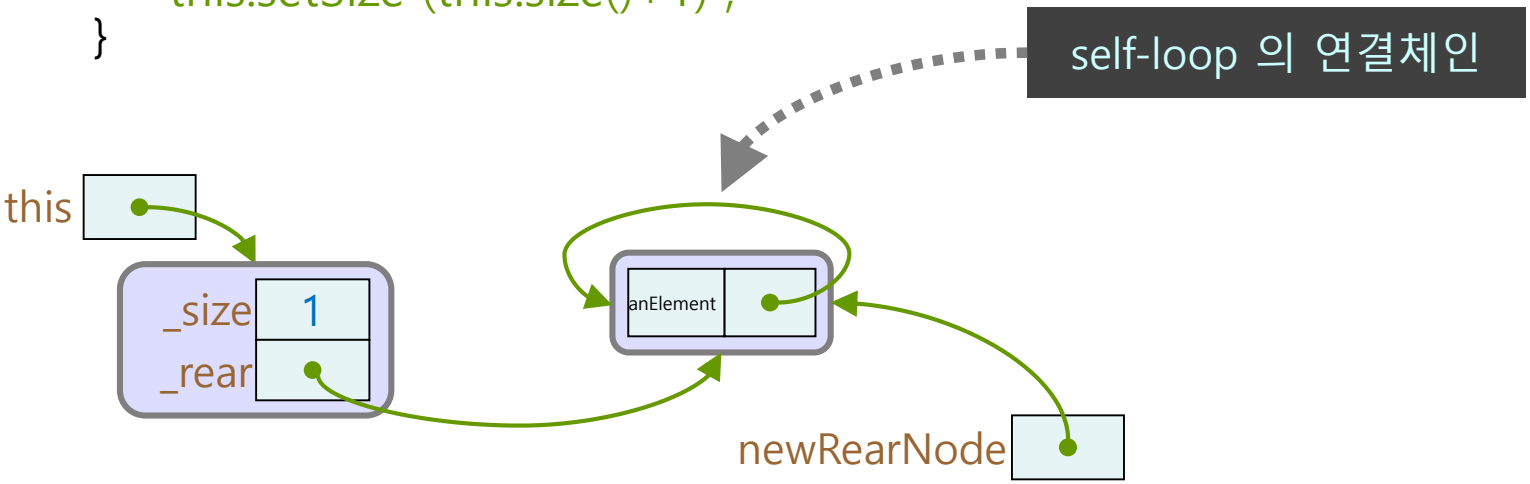
CircularlyLinkedQueue: enQueue() [0]

```
public void  enQueue (T anElement)
{
    LinkedNode<T> newRearNode = new LinkedNode (anElement, null) ;
    if ( this.isEmpty() ) {
        newRearNode.setNext (newRearNode) ;
    }
    else {
        newRearNode.setNext (this.rearNode().next()) ;
        this.rearNode().setNext (newRearNode) ;
    }
    this.setRearNode (newRearNode) ;
    this.setSize (this.size()+1) ;
}
```



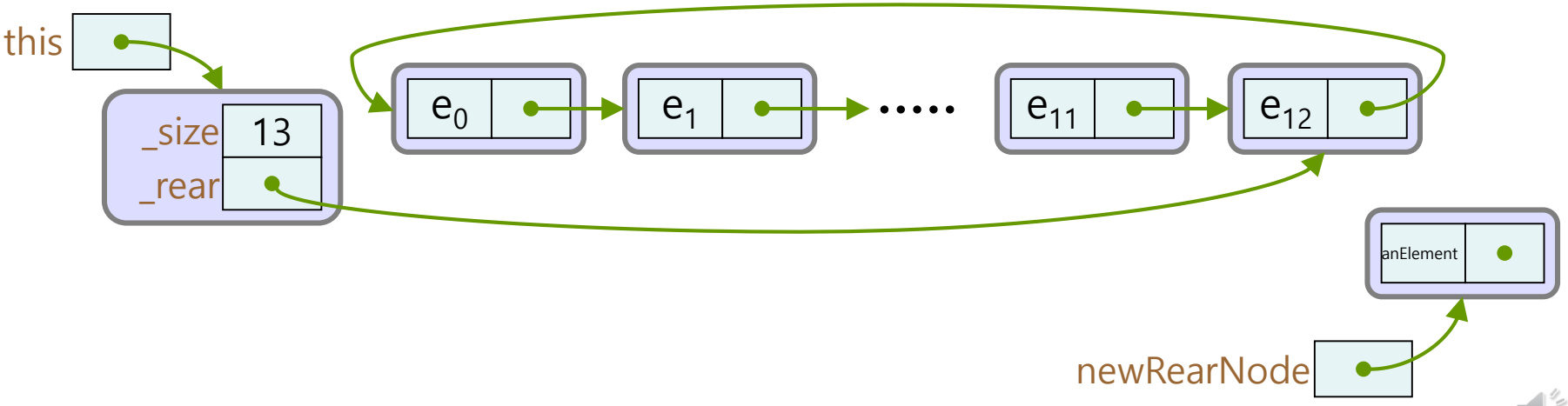
CircularlyLinkedQueue: enQueue() [1]

```
public void  enQueue (T anElement)
{
    LinkedNode<T> newRearNode = new LinkedNode (anElement, null) ;
    if ( this.isEmpty() ) {
        newRearNode.setNext (newRearNode) ;
    }
    else {
        newRearNode.setNext (this.rearNode().next()) ;
        this.rearNode().setNext (newRearNode) ;
    }
    this.setRearNode (newRearNode) ;
    this.setSize (this.size()+1) ;
}
```



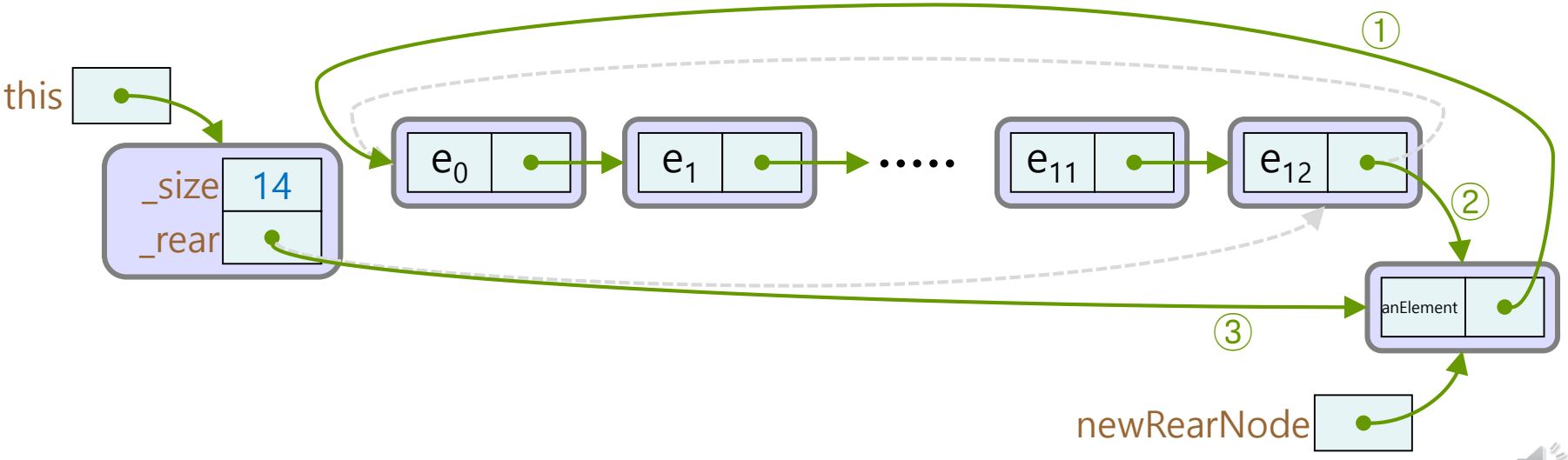
CircularlyLinkedQueue: enQueue() [2]

```
public void  enQueue (T anElement)
{
    LinkedNode<T> newRearNode = new LinkedNode (anElement, null) ;
    if ( this.isEmpty() ) {
        newRearNode.setNext (newRearNode) ;
    }
    else {
        newRearNode.setNext (this.rearNode().next()) ;
        this.rearNode().setNext (newRearNode) ;
    }
    this.setRearNode (newRearNode) ;
    this.setSize (this.size()+1) ;
}
```



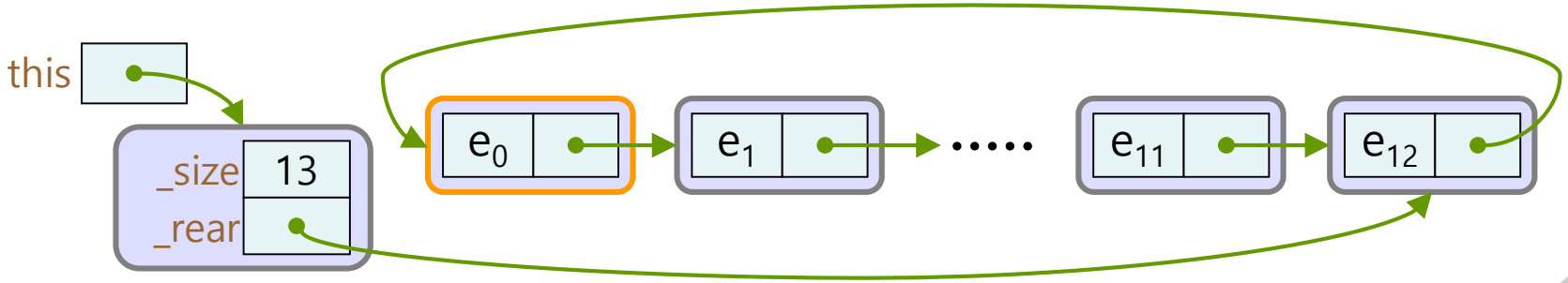
CircularlyLinkedQueue: enQueue() [3]

```
public void  enQueue (T anElement)
{
    LinkedNode<T> newRearNode = new LinkedNode (anElement, null) ;
    if ( this.isEmpty() ) {
        newRearNode.setNext (newRearNode) ;
    }
    else {
        newRearNode.setNext (this.rearNode().next()) ; // ①
        this.rearNode().setNext (newRearNode) ; // ②
    }
    this.setRearNode (newRearNode) ; // ③
    this.setSize (this.size()+1) ;
}
```



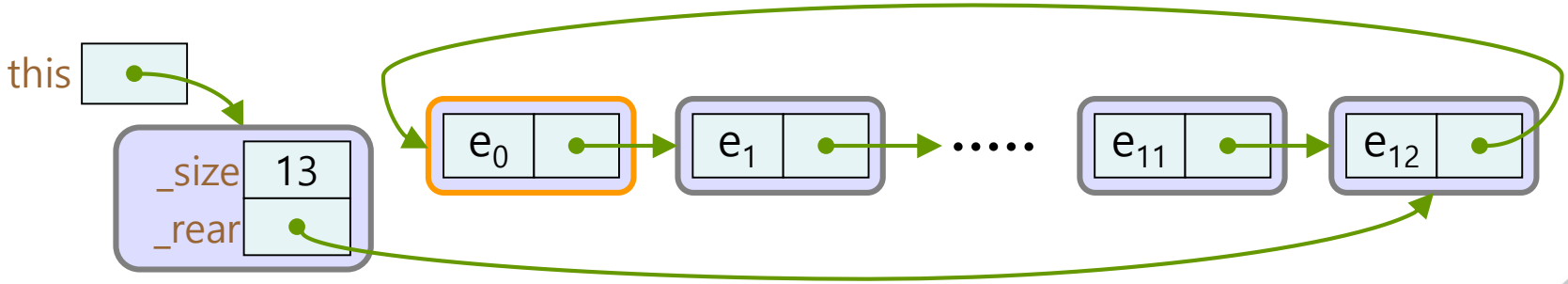
CircularlyLinkedQueue: deQueue()

```
public T  deQueue()
{
    T  frontElement = null ;
    if ( ! this.isEmpty() ) {
        frontElement = this.rearNode().next().element() ;
        if ( this.rearNode() == this.rearNode().next() ) {
            // 노드가 한 개: self-loop 의 경우
            this.setRearNode (null) ;
        }
        else { // 노드가 2 개 이상
            this.rearNode().setNext (this.rearNode().next().next()) ;
        }
        this.setSize (this.size()-1) ;
    }
    return  frontElement ;
}
```



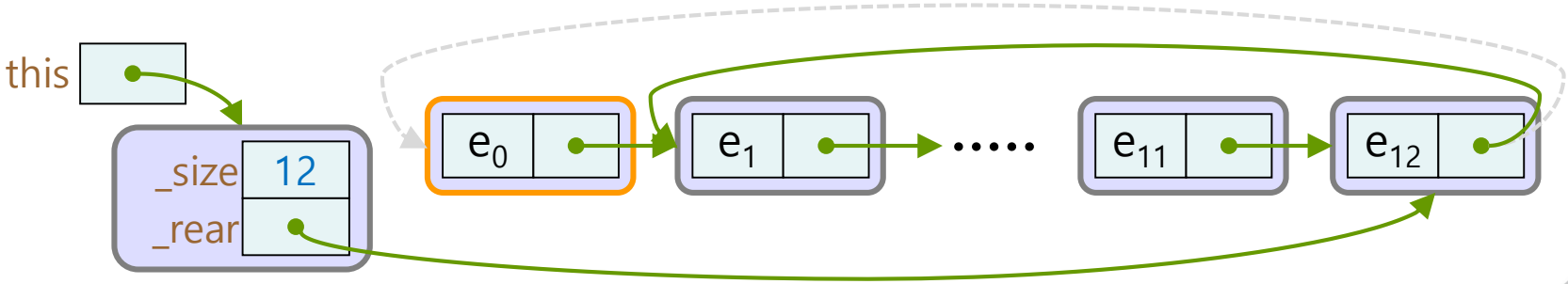
deQueue() [1]

```
public T  deQueue()
{
    T  frontElement = null ;
    if ( ! this.isEmpty() ) {
        frontElement = this.rearNode().next().element() ;
        if ( this.rearNode() == this.rearNode().next() ) {
            // 노드가 한 개: self-loop 의 경우
            this.setRearNode (null) ;
        }
        else { // 노드가 2 개 이상
            this.rearNode().setNext (this.rearNode().next().next()) ;
        }
        this.setSize (this.size()-1) ;
    }
    return  frontElement ;
}
```



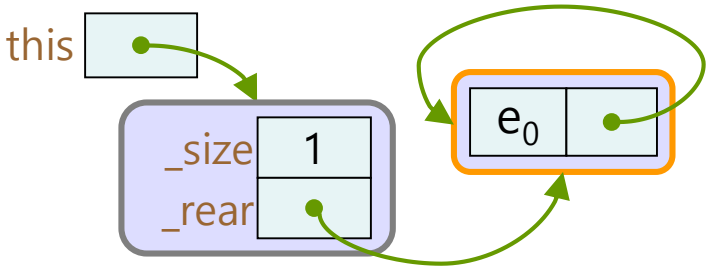
deQueue() [2]

```
public T  deQueue()
{
    T  frontElement = null ;
    if ( ! this.isEmpty() ) {
        frontElement = this.rearNode().next().element() ;
        if ( this.rearNode() == this.rearNode().next() ) {
            // 노드가 한 개: self-loop 의 경우
            this.setRearNode (null) ;
        }
        else { // 노드가 2 개 이상
            this.rearNode().setNext (this.rearNode().next().next()) ;
        }
        this.setSize (this.size()-1) ;
    }
    return  frontElement ;
}
```



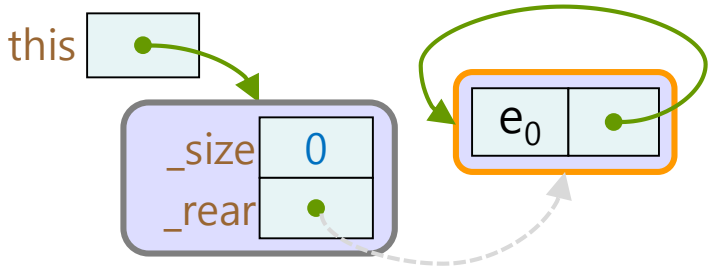
deQueue() [3]

```
public T  deQueue()
{
    T  frontElement = null ;
    if ( ! this.isEmpty() ) {
        frontElement = this.rearNode().next().element() ;
        if ( this.rearNode() == this.rearNode().next() ) {
            // 노드가 한 개: self-loop 의 경우
            this.setRearNode (null) ;
        }
        else { // 노드가 2 개 이상
            this.rearNode().setNext (this.rearNode().next().next()) ;
        }
        this.setSize (this.size()-1) ;
    }
    return  frontElement ;
}
```



deQueue() [4]

```
public T  deQueue()
{
    T  frontElement = null ;
    if ( ! this.isEmpty() ) {
        frontElement = this.rearNode().next().element() ;
        if ( this.rearNode() == this.rearNode().next() ) {
            // 노드가 한 개: self-loop 의 경우
            this.setRearNode (null) ;
        }
        else { // 노드가 2 개 이상
            this.rearNode().setNext (this.rearNode().next().next()) ;
        }
        this.setSize (this.size()-1) ;
    }
    return  frontElement ;
}
```



CircularlyLinkedQueue: clear()

```
public void  clear()
{
    this.setRearNode (null) ;
    this.setSize (0) ;
}
```

End of

"Linked Queue"