Bài 8
ArrayList - LinkedList

Module: **ADVANCED PROGRAMMING WITH PHP 2.0**

Mục tiêu

- Trình bày được PHP data structures
- Phân biệt được các trường hợp sử dụng của ArrayList, LinkedList
- Sử dụng được cấu trúc dữ liệu FixedList
- Sử dụng được cấu trúc dữ liệu LinkedList

Cấu trúc dữ liệu

Cấu trúc dữ liệu

- Cấu trúc dữ liệu là hình thức tổ chức một nhóm dữ liệu:
  - Lưu trữ dữ liệu
  - Cung cấp các phương thức để thao tác với dữ liệu
- Các khái niệm:
  - Container: Lớp chứa dữ liệu
  - Elements: Các phần tử dữ liệu
- Ví dụ:
  - Lớp ArrayList là cấu trúc danh sách, lưu trữ nhiều giá trị
  - Các phương thức được cung cấp để thực hiện các thao tác: Thêm phần tử, xoá phần tử, duyệt phần tử, tìm kiếm…
- Việc lựa chọn cấu trúc dữ liệu và thuật toán phù hợp là rất quan trọng đối với hiệu năng của ứng dụng

Các cấu trúc dữ liệu thông dụng

- Set (Tập hợp): Nhóm các phần tử không trùng nhau
- Array
- List (Danh sách): Nhóm ác phần tử có thể trùng nhau
- Stack: Nhóm các phần tử theo trật tự first-in/last-out (vào trước/ra sau)
- Queue: Nhóm các phần tử theo trật tự first-in/first-out (vào trước/ra trước)
- Map (Bản đồ): Lưu trữ các cặp key/value
- Tree (Cây): Lưu trữ các phần tử theo mối quan hệ cha-con
- Graph (Đồ thị): Lưu trữ các phần tử theo mối quan hệ mạng lưới

ArrayList

ArrayList

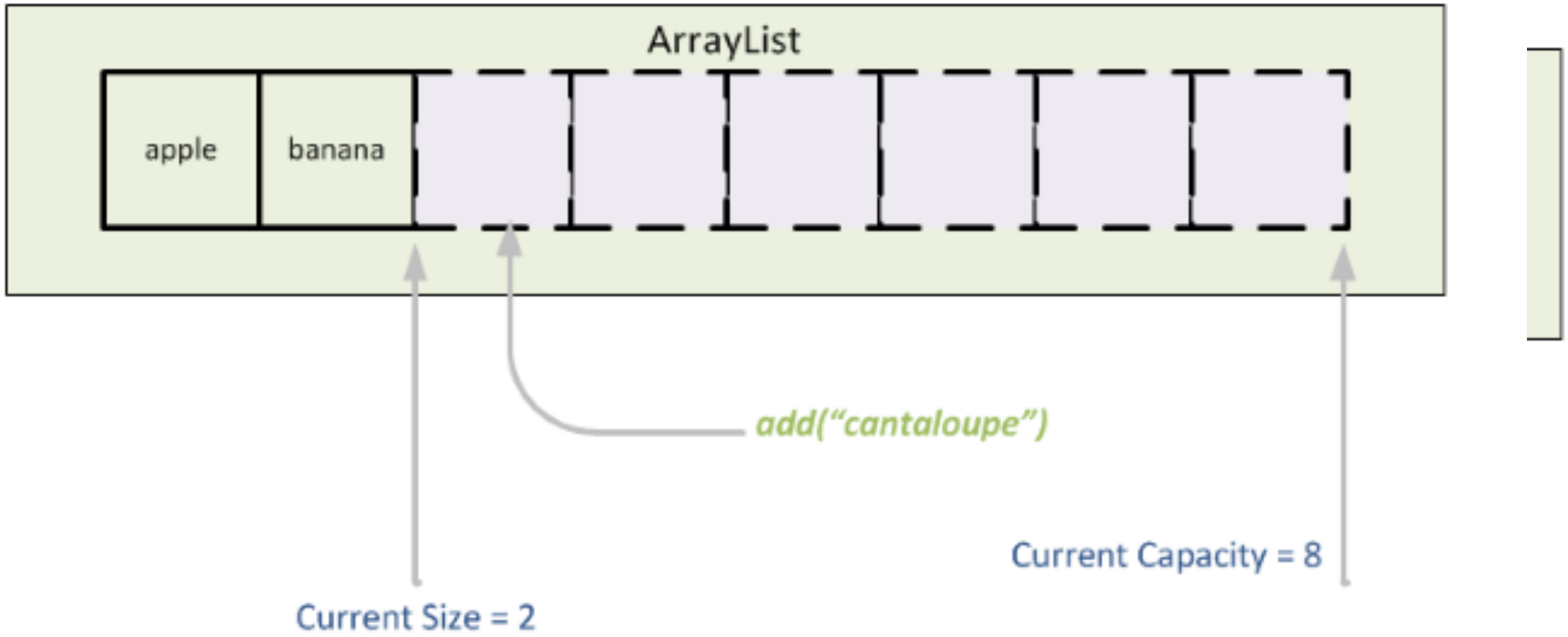
- Là một danh sách
- Lưu trữ dữ liệu trong mảng
- Kích thước thay đổi được
- Truy xuất các phần tử nhanh
- Thêm hoặc xoá các phần tử chậm

Các thao tác cơ bản của ArrayList

- get(): Lấy về một phần tử
- add(): Thêm một phần tử
- remove(): Xoá một phần tử
- size(): Lấy về số lượng phần tử
- find(): Tìm kiếm phần tử
- isEmpty(): Kiểm tra rỗng

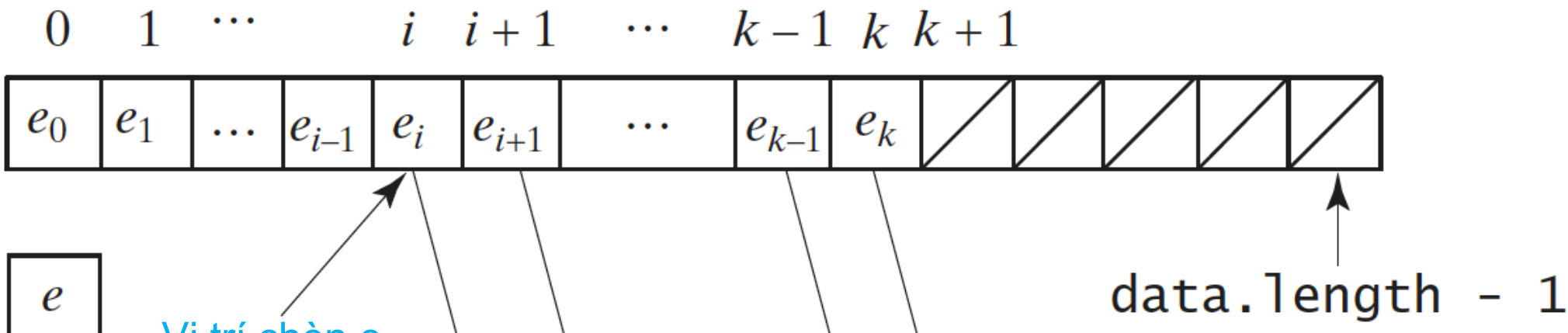
- Sử dụng mảng để lưu trữ các phần tử

Cấu trúc của ArrayList

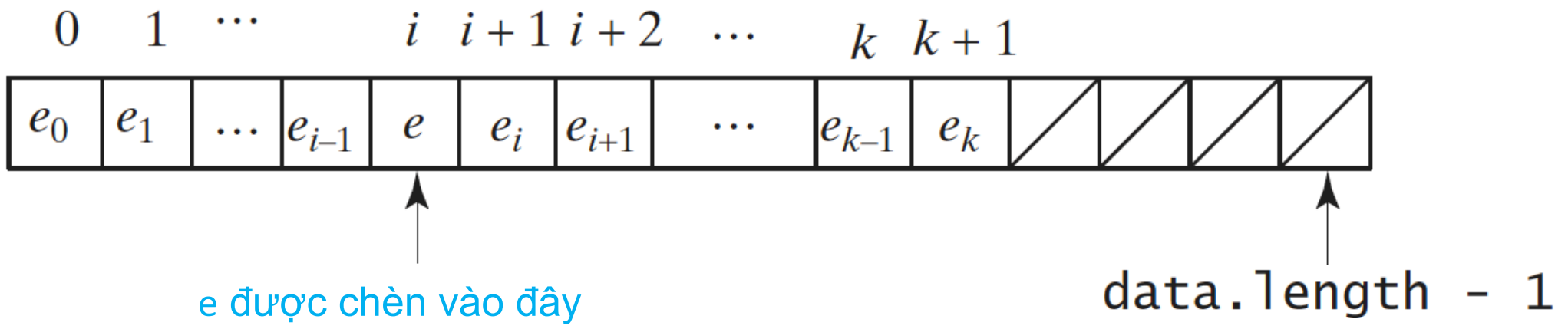


Thêm phần tử vào ArrayList

Trước khi chèn *e* vào vị trí *i*

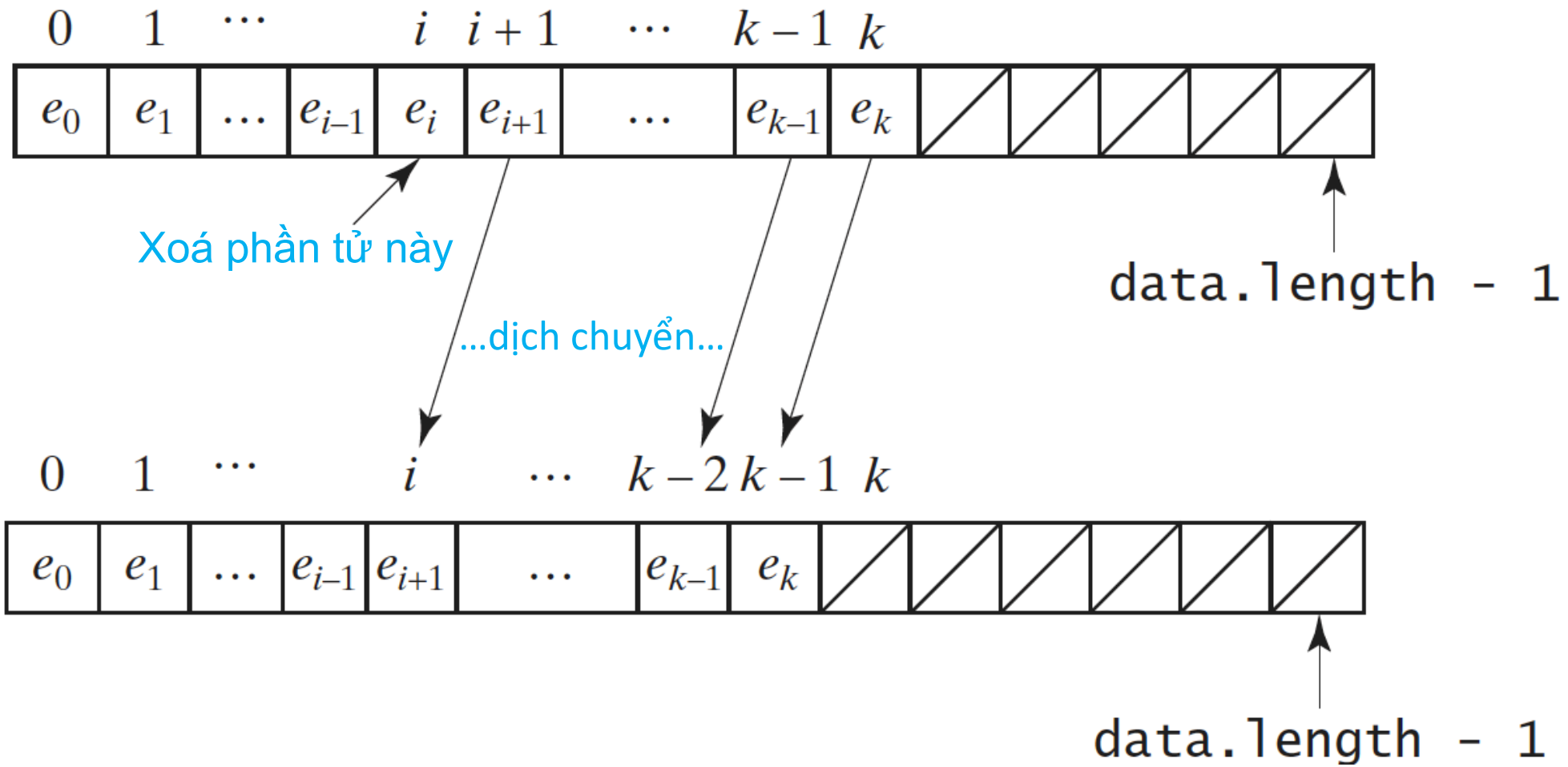


Sau khi chèn *e* vào vị trí *i, kích thước ArrayList tăng thêm 1*



Xoá phần tử khỏi ArrayList

Trước khi xoá phần tử tại vị trí *i*



Sau khi xoá phần tử tại vị trí *i*, kích thước của ArrayList giảm xuống 1

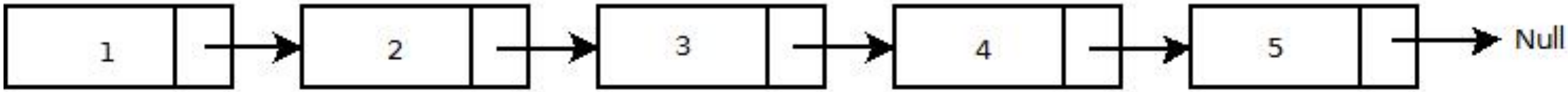
Cài đặt lớp ArrayList

```php
class ArrayList
{
    private $arrayList = NULL;

    function __construct(){...}

    function add($item){...}

    function addAtPos($item, $index){...}

    function removeByIndex($index){...}

    function contains($item){...}

    function get($index){...}

    function toArray(){...}

    function size(){...}

    function isEmpty(){...}

    function shiftItemsUp($startIndex, $endIndex){...}

    function shiftItemsDown($startIndex, $endIndex){...}
}
```

LinkedList

LinkedList

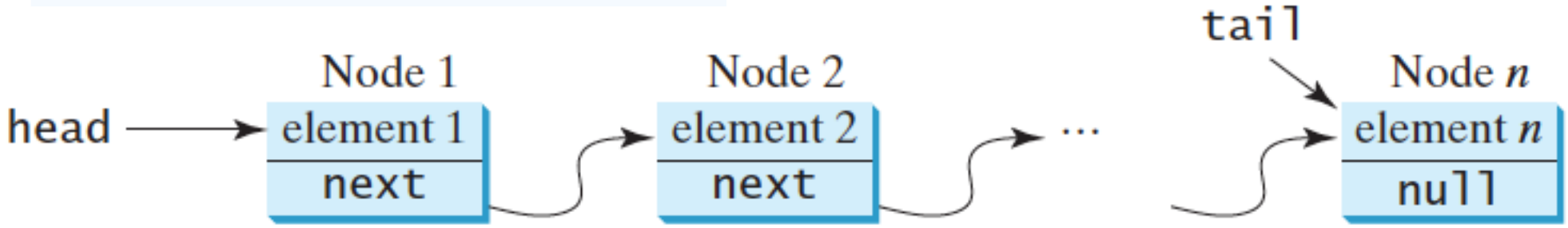
- Các phần tử liên kết với nhau thông qua tham chiếu
- Truy xuất ngẫu nhiên chậm
- Thêm và xoá phần tử nhanh



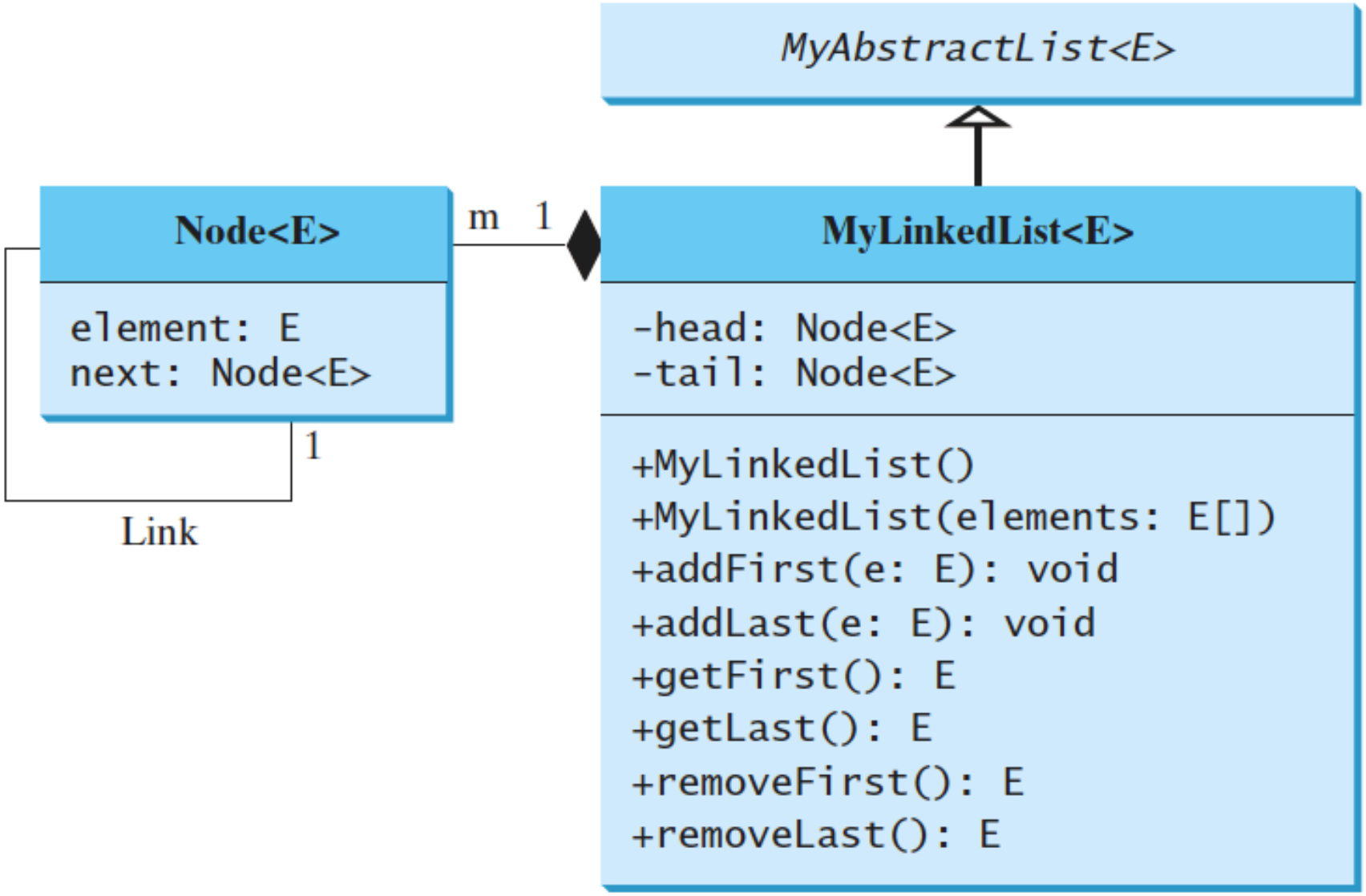
Cấu trúc của LinkedList

- LinkedList hoạt động dựa trên cơ chế liên kết giữa các Node
- Mỗi node chứa dữ liệu của node đó và liên kết đến node khác

```php
class Node
{
    // stores added item
    public $element;

    // stores address to next node
    public $next;

    // Node constructor
    function __construct($item = FALSE){...}

    // returns item
    public function getData(){...}
}
```

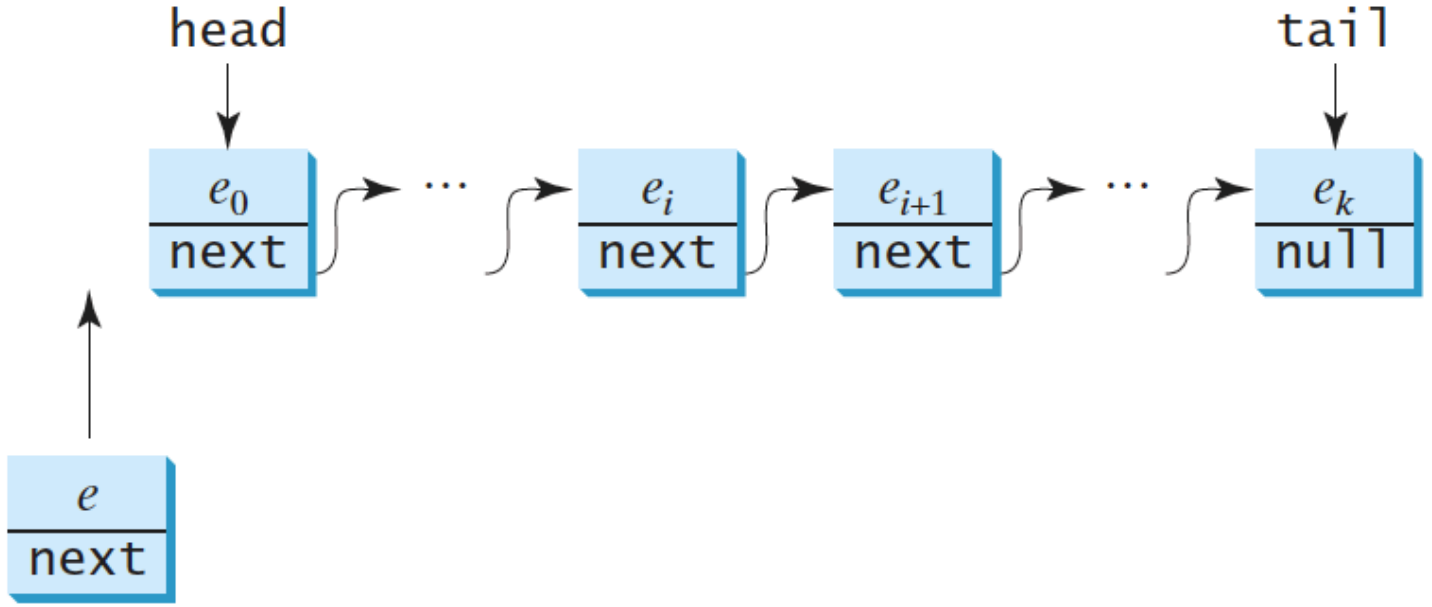


Lớp MyLinkedList

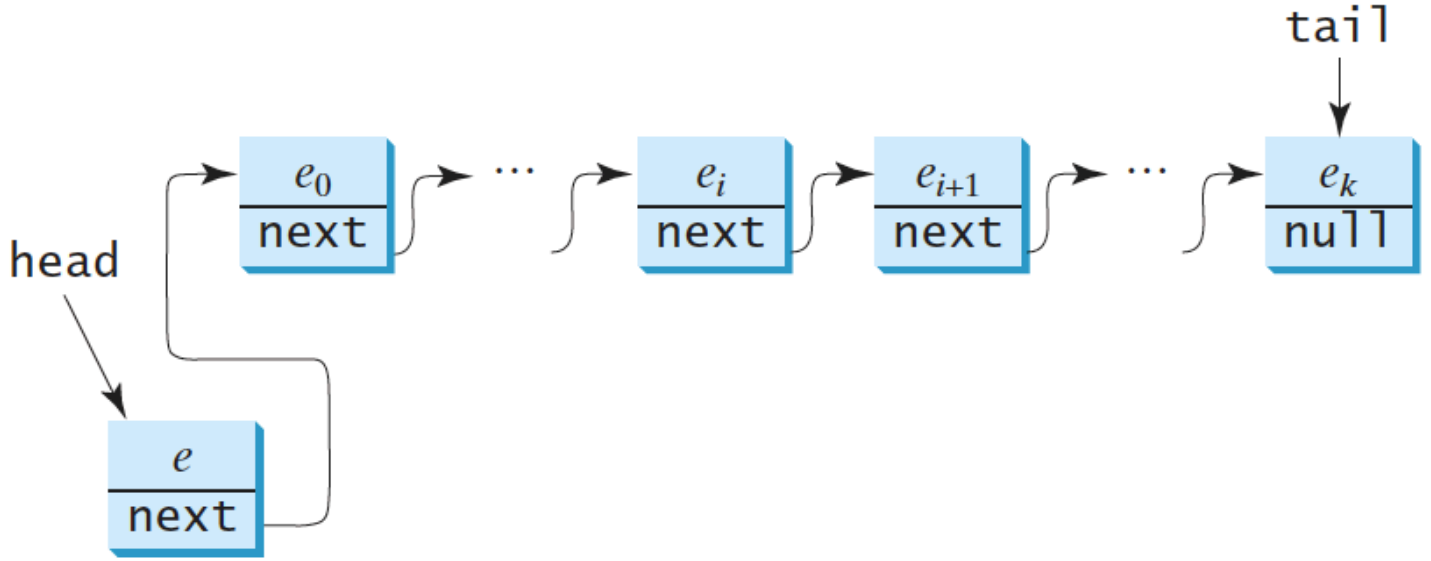


Phương thức addFirst()

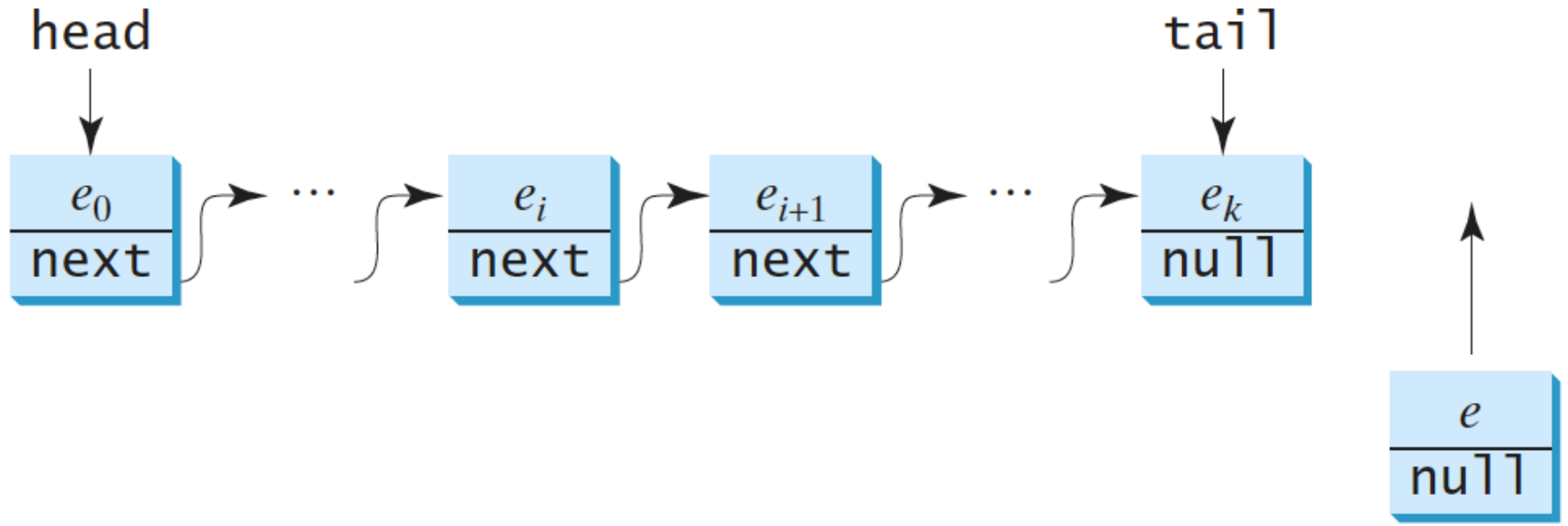
Trước khi chèn phần tử vào đầu



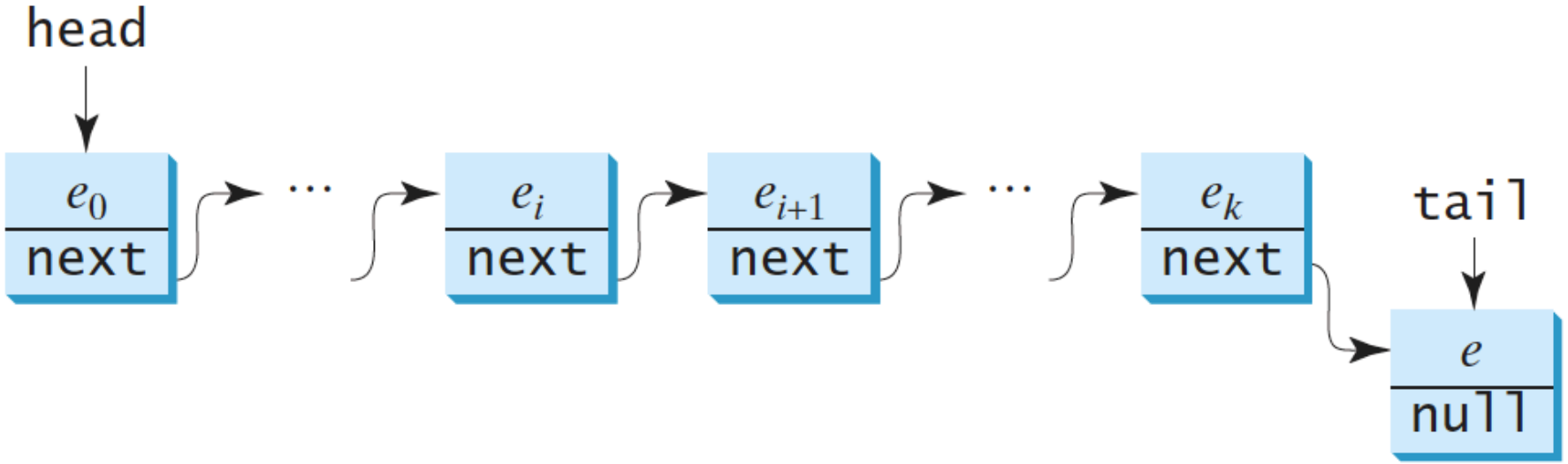
Sau khi được chèn vào đầu



Phương thức addLast()

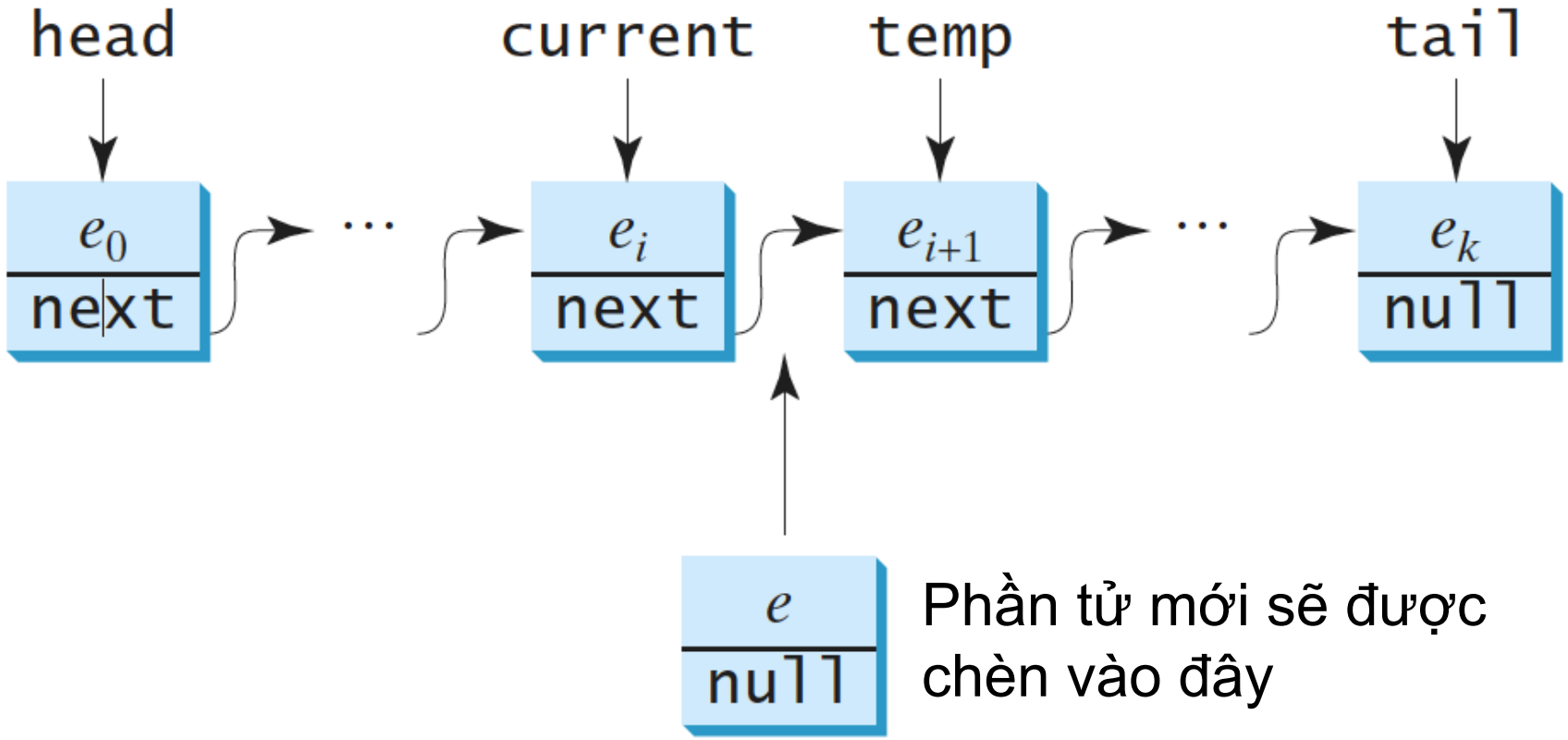


Trước khi chèn phần tử vào đuôi

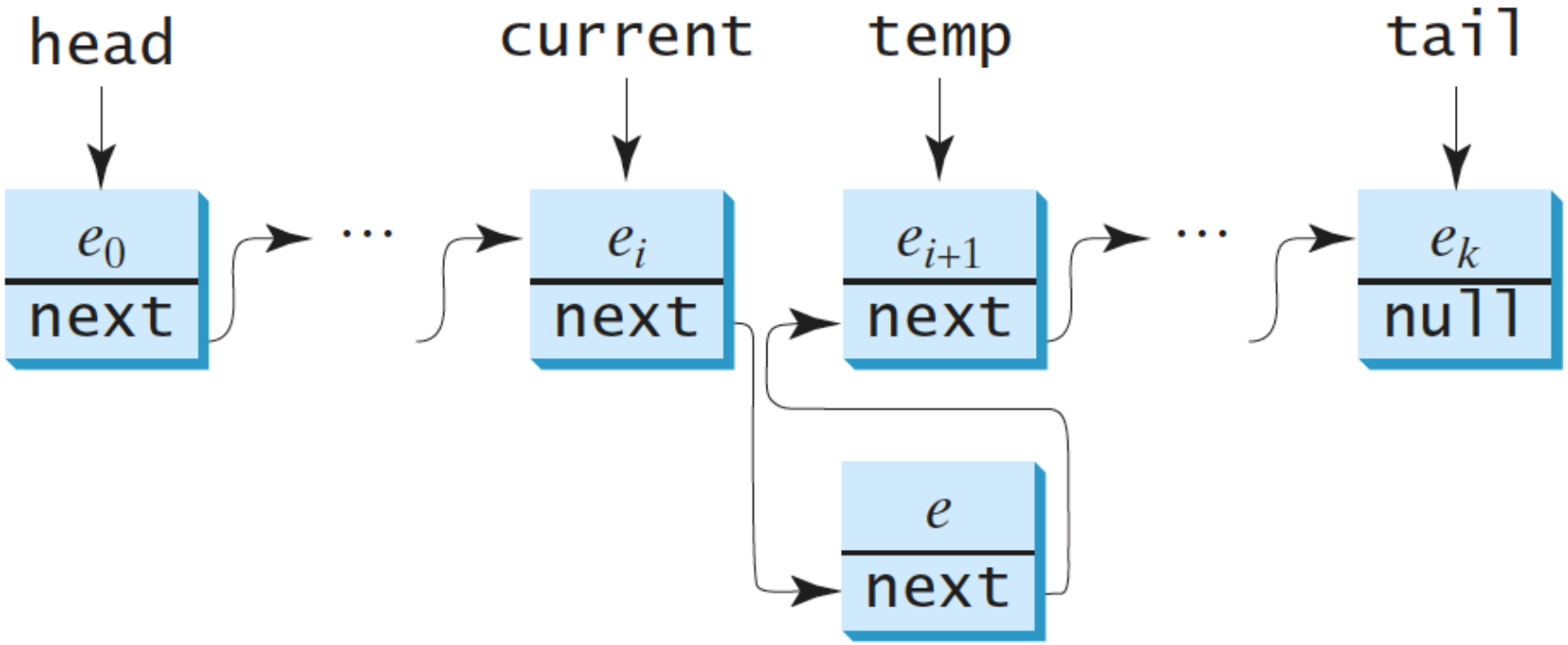


Sau khi chèn phần tử vào đuôi

Phương thức add(index: int, e: Object)

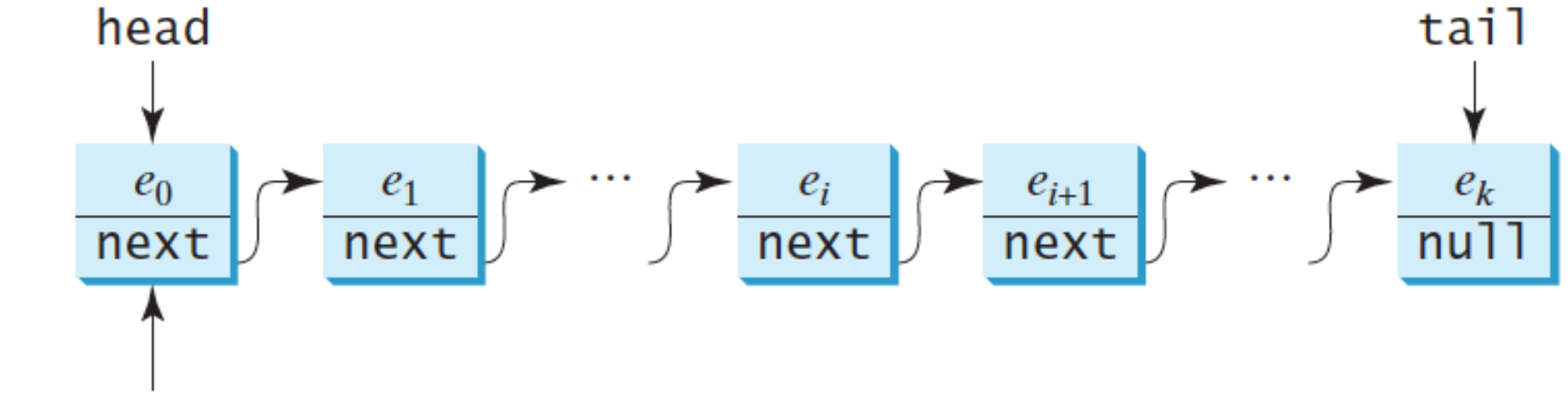


Phương thức add()

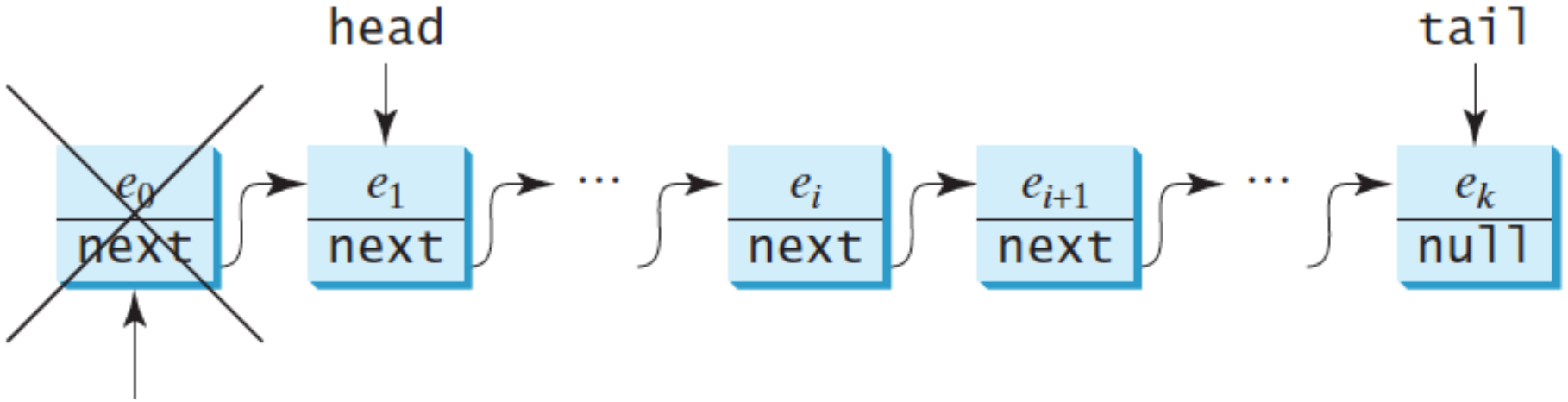


Phần tử mới đã được chèn vào LinkedList

Phương thức removeFirst()

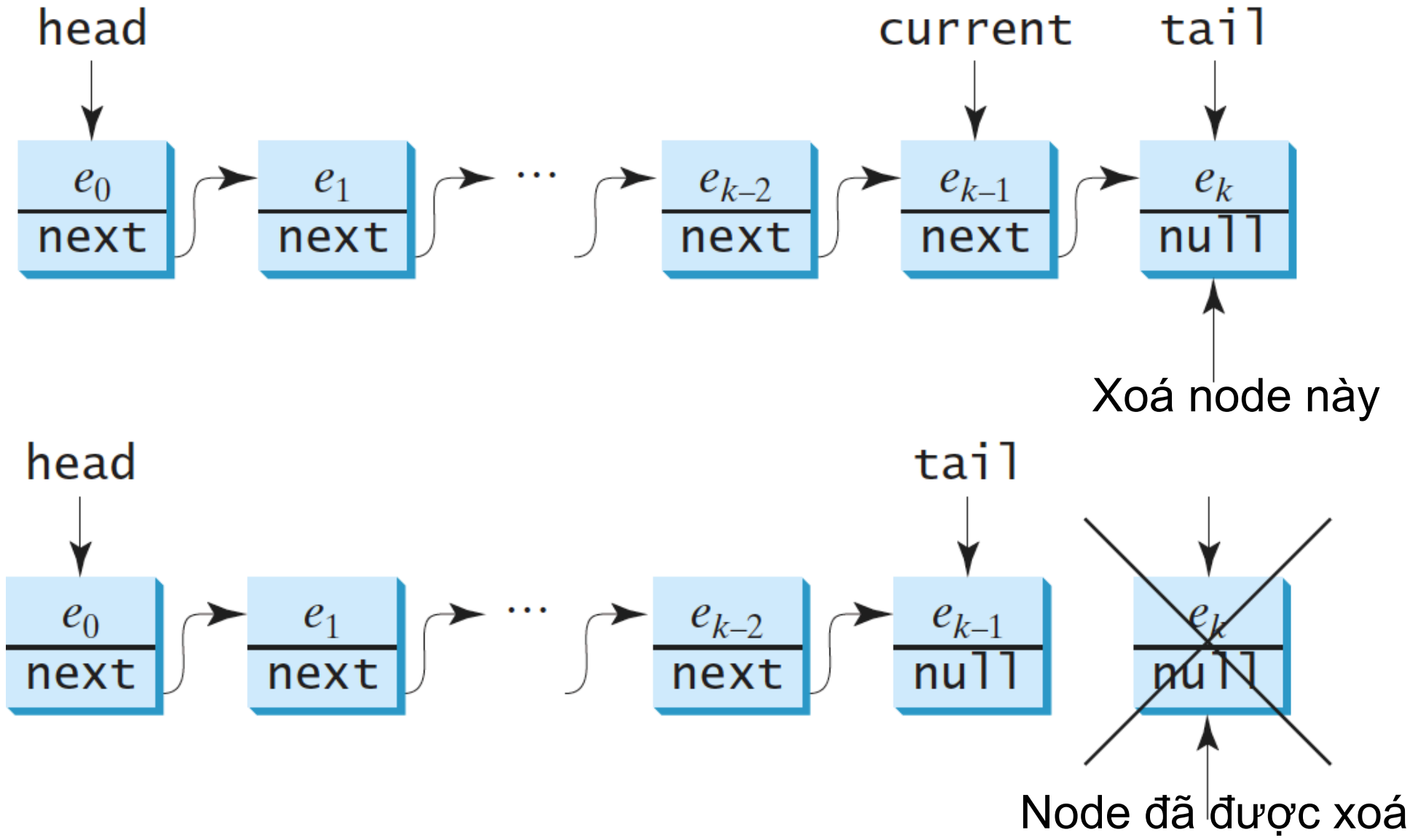


Xoá node này

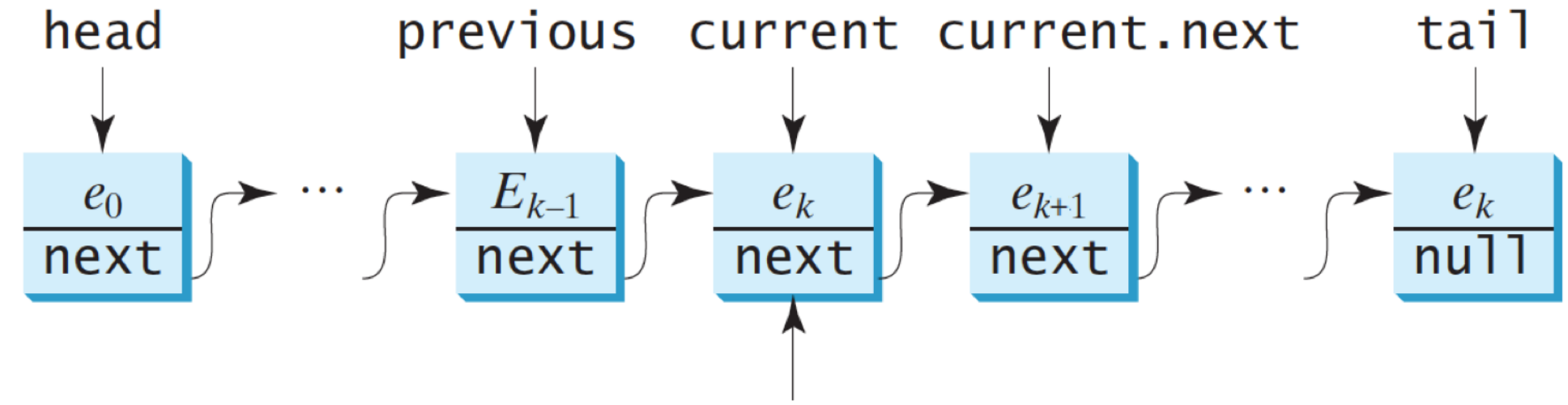


Node đã được xoá

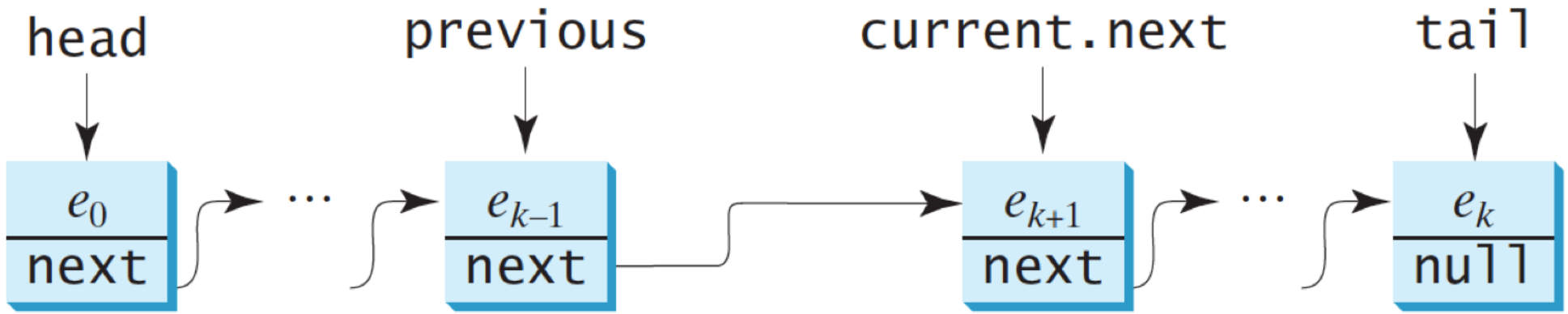
Phương thức removeLast()



Phương thức remove()



Xoá node này



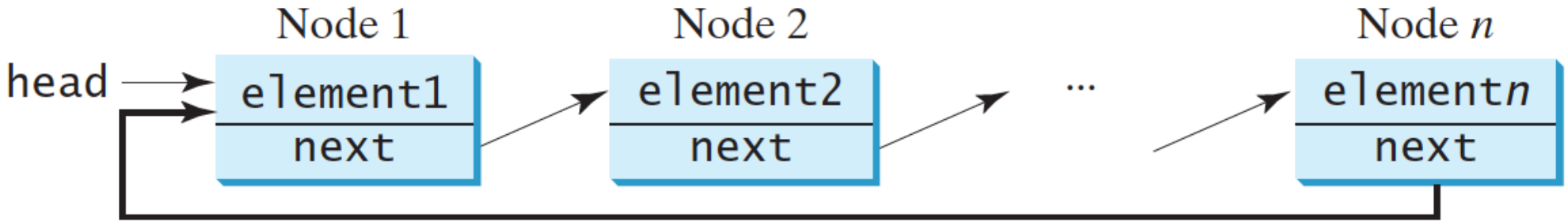
Node đã được xoá

Cài đặt lớp LinkedList

```php
class LinkedList
{
    //link to first node of linked list
    private $head;

    // link to last node of linked list
    private $tail;

    // count of total item in linked list
    private $count;

    function __construct(){...}

    public function insertAtLast($item){...}

    public function insertAtFirst($item){...}

    public function insertAtPos($item, $position){...}

    public function toArray(){...}

    public function removeByIndex($index){...}

    public function contains($item){...}

    public function get($index){...}

    public function size(){...}

    public function isEmpty(){...}

    function removeFirst(){...}

    private function remove($index){...}

    private function insert($item, $position){...}

}
```

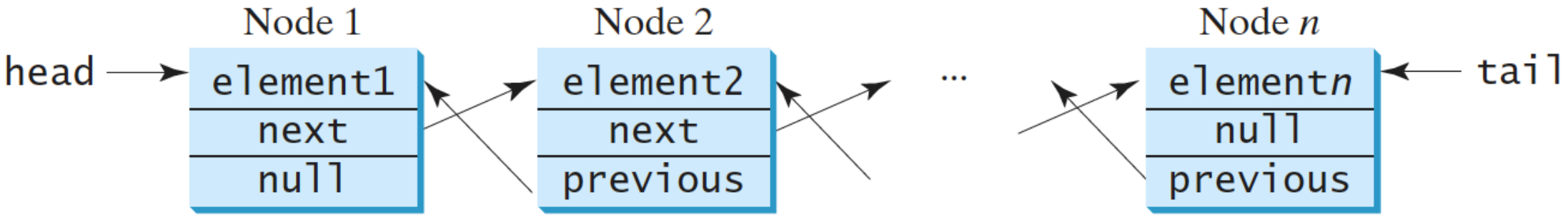
Singly Linked List

- Singly Linked List (Danh sách liên kết đơn):
  - Một node chỉ có một liên kết đến node phía sau nó
  - Node cuối cùng trỏ đến null
- Circular Singly LinkedList (Danh sách liên kết đơn vòng):
  - Node cuối vùng trỏ đến node đầu tiên



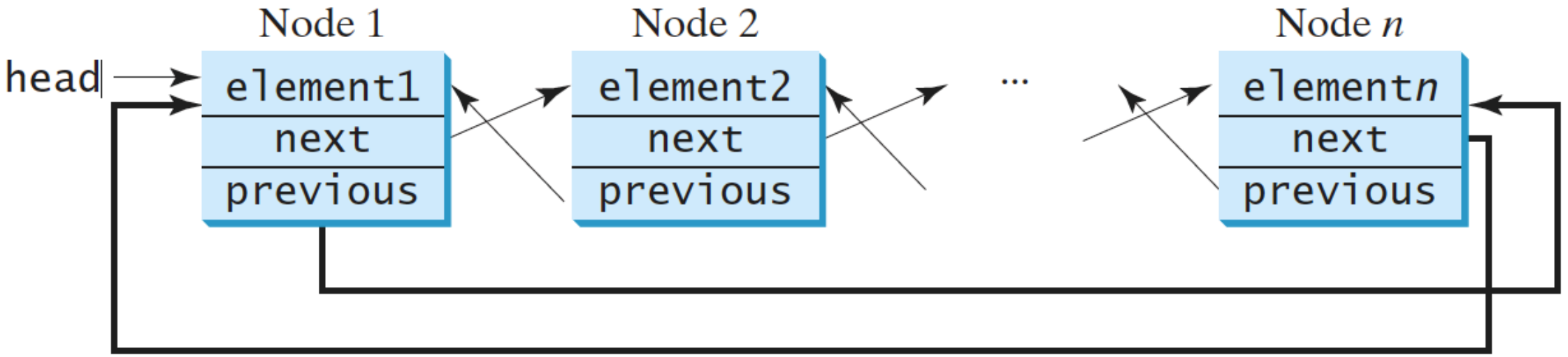
Doubly Linked List

- Doubly Linked List (Danh sách liên kết đôi):
  - Một node chứa hai liên kết trỏ đến phần tử đứng trước và sau nó
  - Phần tử trước của phần tử đầu tiên là null
  - Phần tử sau của phần tử sau là null



Circular Doubly Linked List

- Circular Doubly Linked List (Danh sách liên kết đôi vòng):
  - Node đầu tiên và node cuối cùng có liên kết trỏ đến nhau



Lựa chọn ArrayList hay LinkedList

ArrayList	LinkedList
Truy xuất ngẫu nhiên nhanh	Truy xuất ngẫu nhiên chậm
Thêm, xoá chậm	Thêm, xoá nhanh

- ArrayList phù hợp với các bài toán cần thực hiện nhiều thao tác truy xuất ngẫu nhiên và ít thêm, xoá ở đầu danh sách
- LinkedList phù hợp với các bài toán cần thêm, xoá nhiều ở đầu danh sách và ít truy xuất ngẫu nhiên

Tóm tắt bài học

- Cấu trúc dữ liệu là hình thức tổ chức dữ liệu và cung cấp cácphương thức để thao tác với các phần tử
- Cấu trúc List chứa các phần tử cho phép trùng lặp
- ArrayList lưu trữ các phần tử trong mảng
- LinkedList lưu trữ các phần tử theo cơ chế liên kết giữa các phần tử

Hướng dẫn

- Hướng dẫn làm bài thực hành và bài tập
- Chuẩn bị bài tiếp: ***Stack-Queue-Tree***