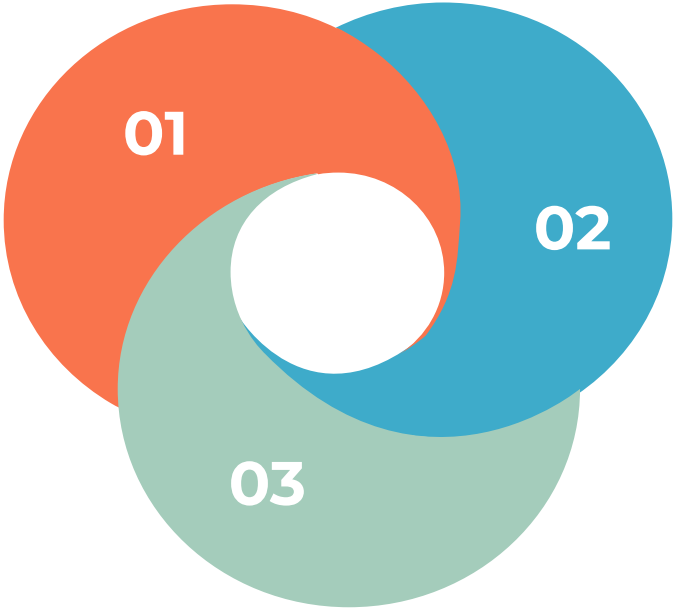
Universidad ICESI

Listas enlazadas dobles

Esta presentación proporciona una visión general de las listas enlazadas dobles en Java. Cubre los conceptos básicos y las operaciones involucradas en el trabajo con listas vinculadas.

Introducción a las listas enlazadas

La lista enlazada es una estructura de datos popular en programación

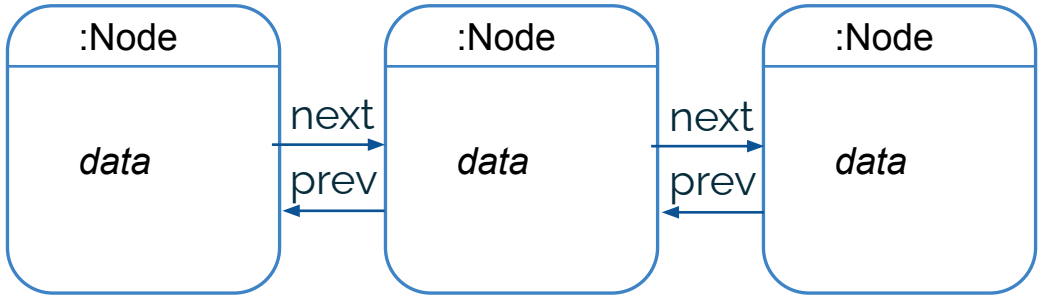


Consiste en una secuencia de nodos, donde cada nodo almacena una referencia al previo/siguiente nodo.

Permite la inserción y extracción eficiente de elementos

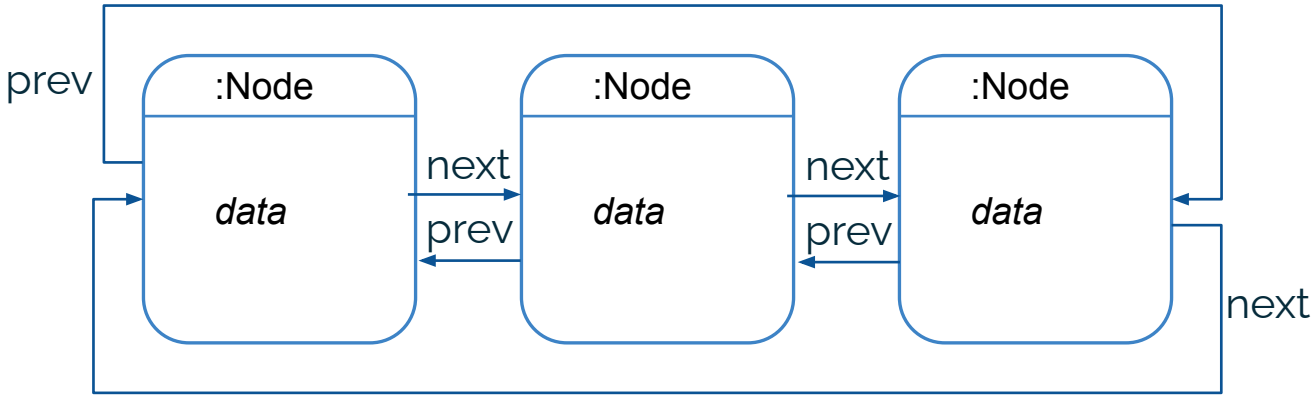
Clase ‘nodo’

- Cada nodo de una lista enlazada está representado por un objeto de una clase.
- La clase nodo tiene una información importante: data (atributos), y dos referencias importantes: prev (nodo anterior en la lista) next (siguiente nodo de la lista)



Caso especial: Lista circular

- El último elemento de la lista debe tener un enlace hacia el primer elemento y el primero debe tener un enlace hacia el último, formando un comportamiento de recorrido en bucle/circular.

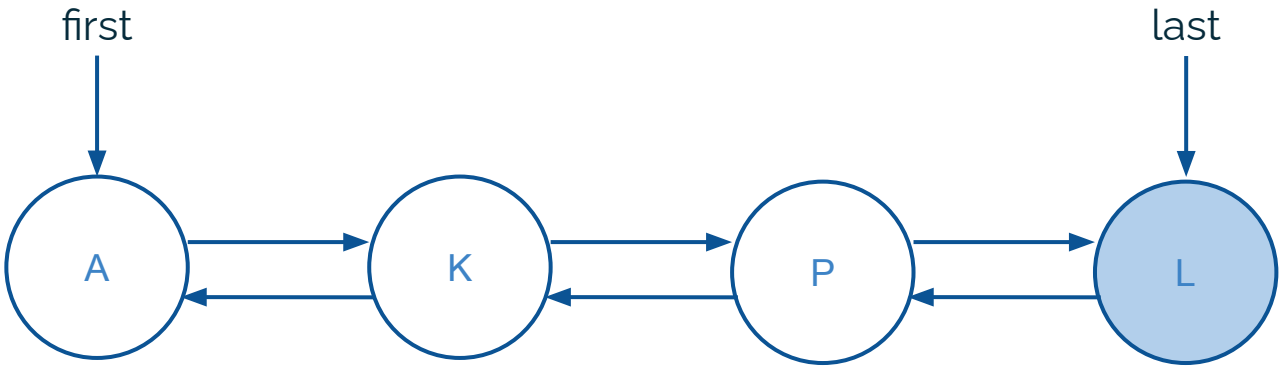


Ventajas de listas enlazadas- Complejidad temporal

Operation	Time Complexity	Auxiliary Space	Explanation
Insertion at Beginning	O(1)	O(1)	Constant-time pointer updates.
Insertion at End	O(n)	O(1)	Traversal required to find the last node.
Insertion at Position	O(n)	O(1)	Traversal to the desired position, then constant-time pointer updates.
Deletion at Beginning	O(1)	O(1)	Constant-time pointer update.
Deletion at End	O(n)	O(1)	Traversal required to find the second last node.
Deletion at Position	O(n)	O(1)	Traversal to the desired position, then constant-time pointer updates.
Searching in Linked list	O(n)	O(1)	Traversal through the list to find the desired value.

https://www.geeksforgeeks.org/time-and-space-complexity-of-linked-list/

Estructuras lineales: lista enlazada



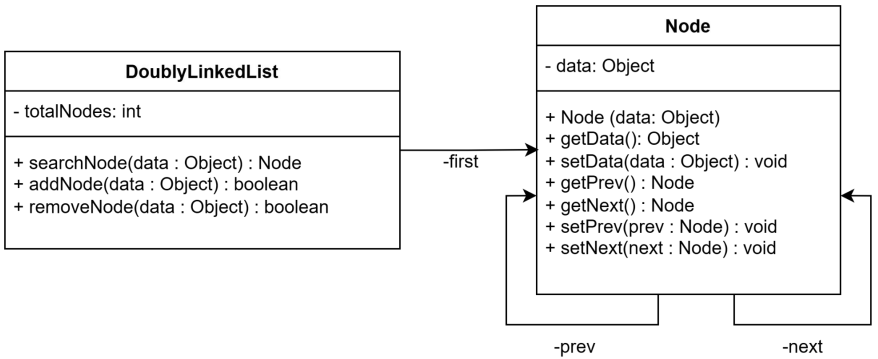
La lista enlazada debe tener un puntero que referencia al ***first/head***
También puede tener un enlace a la **last/*tail***.

Creación de una lista enlazada

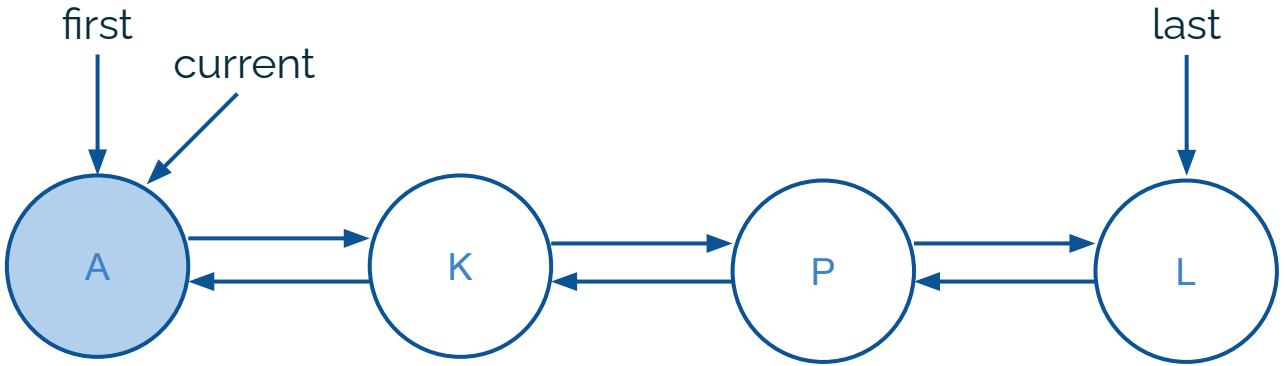
```
public class Node {
    private Object data; // Data stored in the node
    private Node prev; // Reference to the prev node
    private Node next; // Reference to the next node

    public Node(Object data) {
         this.data = data;
         this.prev = null;
         this.next = null;
    }

    public Object getData() {
        return data;
    }

    public void setData(Object data) {
        this.data = data;
    }

    public Node getPrevious() {
        return prev;
    }

    public void setPrevious(Node prev) {
        this.prev = prev;
    }

    public Node getNext() {
        return next;
    }

    public void setNext(Node next) {
        this.next = next;
    }
}
```

Creación de una lista enlazada

```
public class DoublyLinkedList {
    private Node first; // first of the list

    public DoublyLinkedList() {
        this.first = null;
    }

    public Node searchNode(Object data){}
    public boolean add(Object data){}
    public boolean remove(Object data){}
```



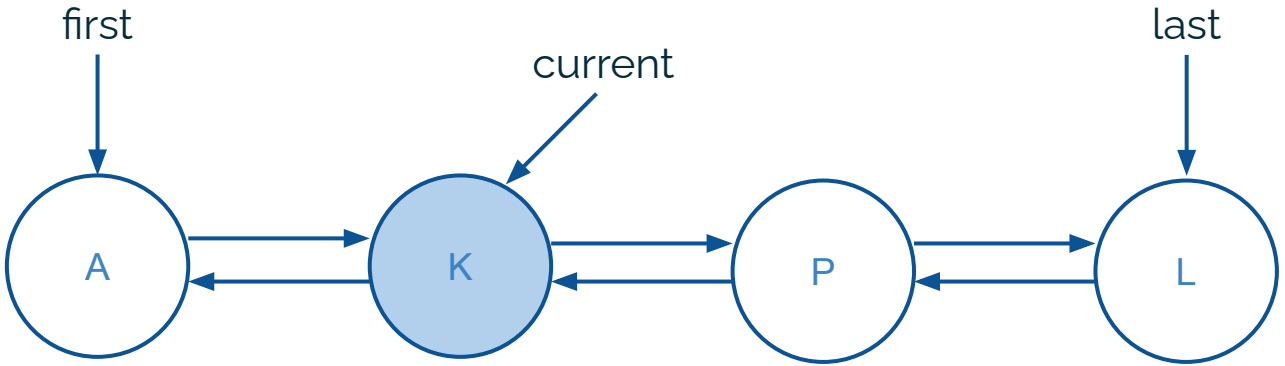
Listas enlazadas: buscar hacia adelante



El recorrido es simple y se puede usar un puntero para hacerlo.

No un puntero numérico, sino un objeto puntero con la referencia del nodo *current*

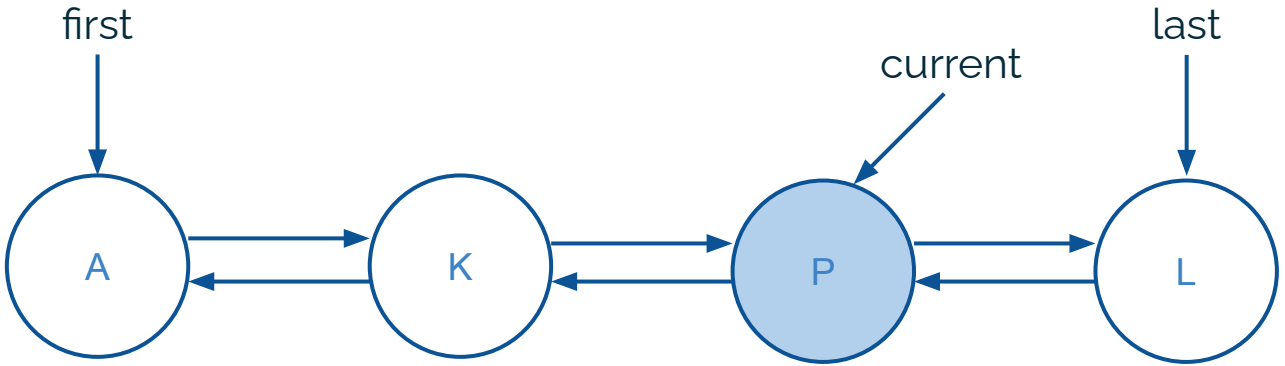
Listas enlazadas: buscar hacia adelante



El recorrido es simple y se puede usar un puntero para hacerlo.

No un puntero numérico, sino un objeto puntero con la referencia del nodo *current*

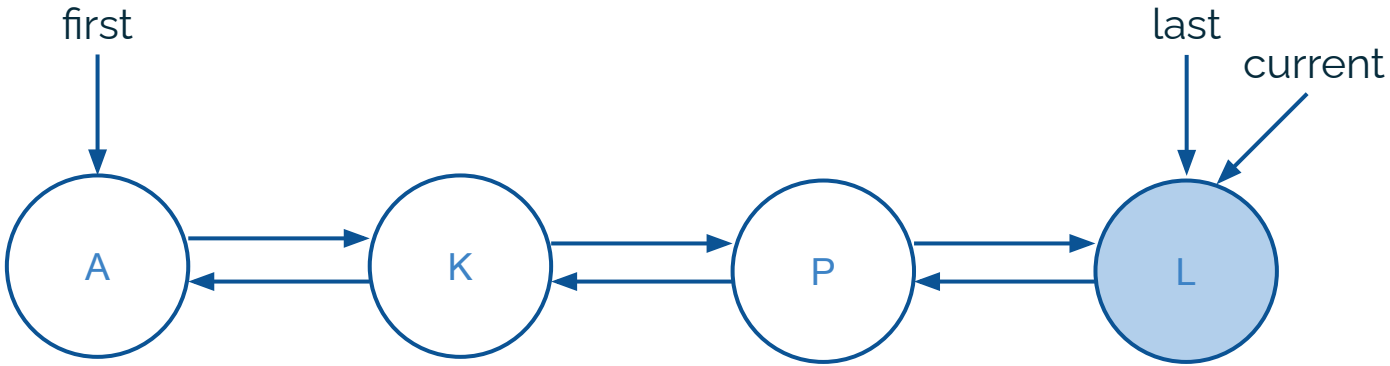
Listas enlazadas: buscar hacia adelante



El recorrido es simple y se puede usar un puntero para hacerlo.

No un puntero numérico, sino un objeto puntero con la referencia del nodo *current*

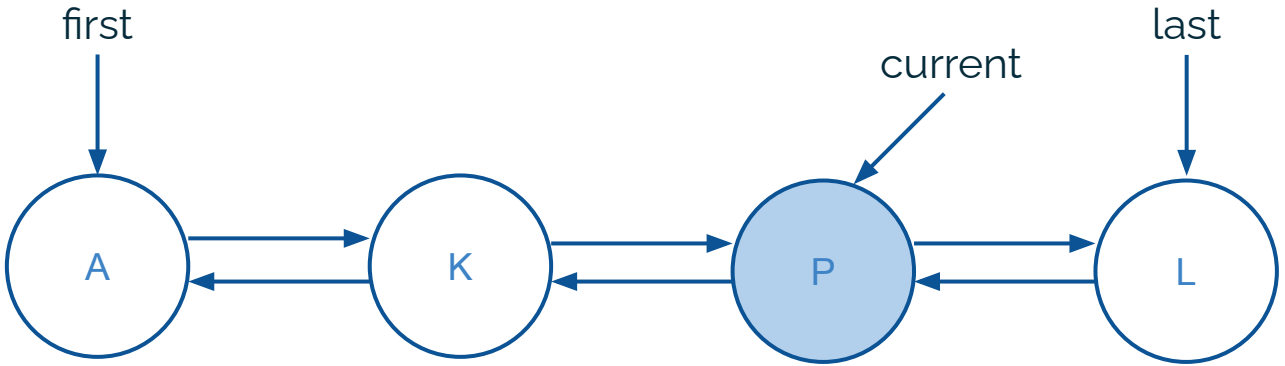
Listas enlazadas: buscar hacia adelante



El recorrido es simple y se puede usar un puntero para hacerlo.

No un puntero numérico, sino un objeto puntero con la referencia del nodo *current*

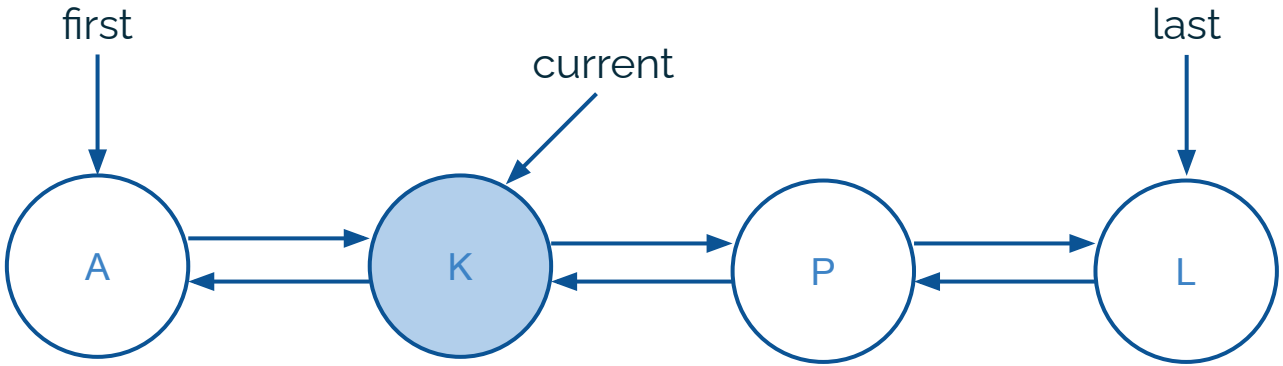
Listas enlazadas: buscar hacia atrás



El recorrido es simple y se puede usar un puntero para hacerlo.

No un puntero numérico, sino un objeto puntero con la referencia del nodo *current*

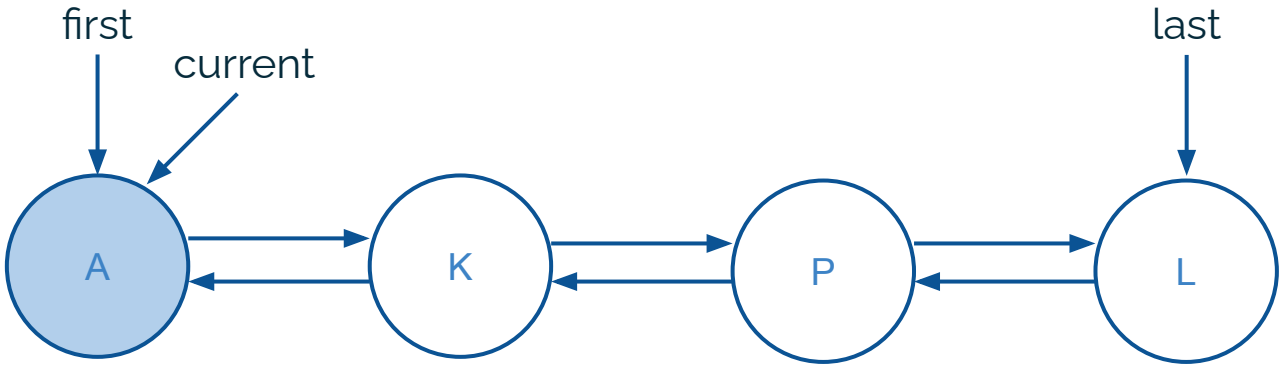
Listas enlazadas: buscar hacia atrás



El recorrido es simple y se puede usar un puntero para hacerlo.

No un puntero numérico, sino un objeto puntero con la referencia del nodo *current*

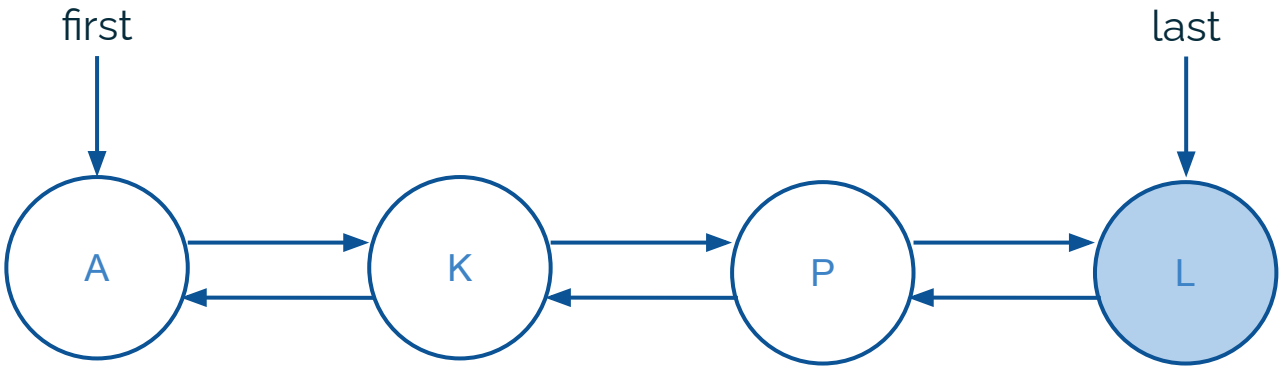
Listas enlazadas: buscar hacia atrás



El recorrido es simple y se puede usar un puntero para hacerlo.

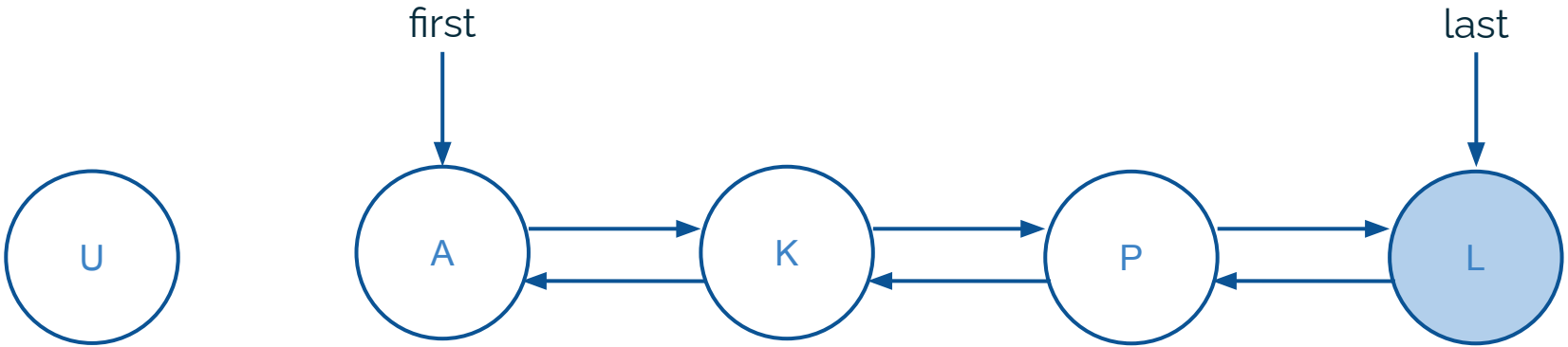
No un puntero numérico, sino un objeto puntero con la referencia del nodo *current*

Listas enlazadas: Agregar nuevo elemento



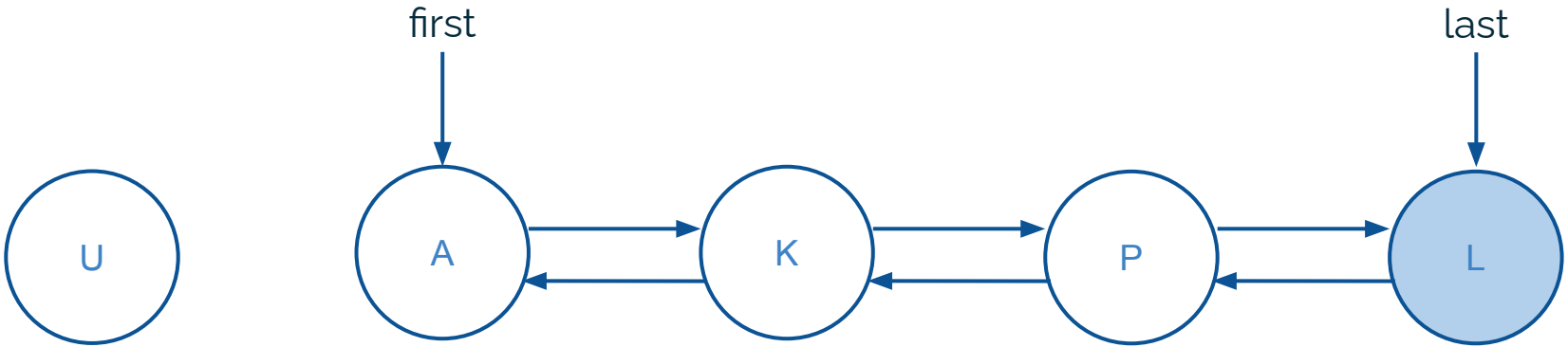
Para agregar debe tener en cuenta que debe mantener saludables los enlaces.
Por ejemplo si quiere insertar al principio una letra U

Listas enlazadas: Agregar nuevo elemento



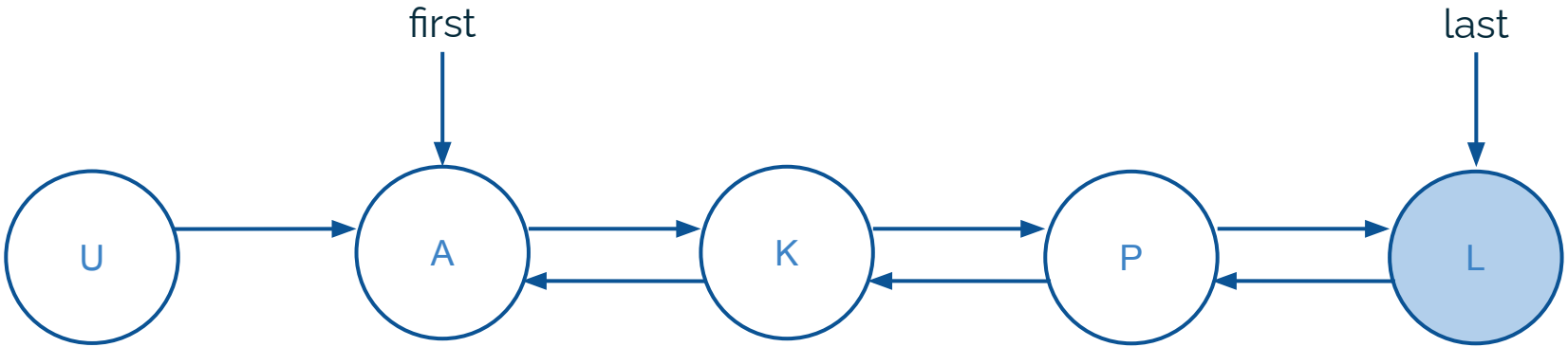
Para agregar debe tener en cuenta que debe mantener saludables los enlaces.
Por ejemplo si quiere insertar al principio una letra U

Listas enlazadas: Agregar nuevo elemento



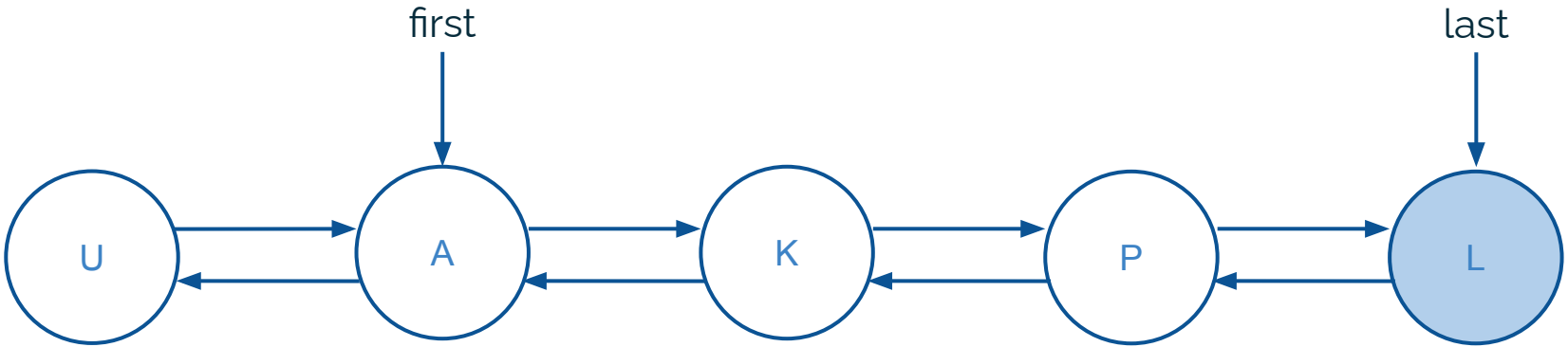
Se debe poner el enlace prev y *next* para la siguiente letra

Listas enlazadas: Agregar nuevo elemento



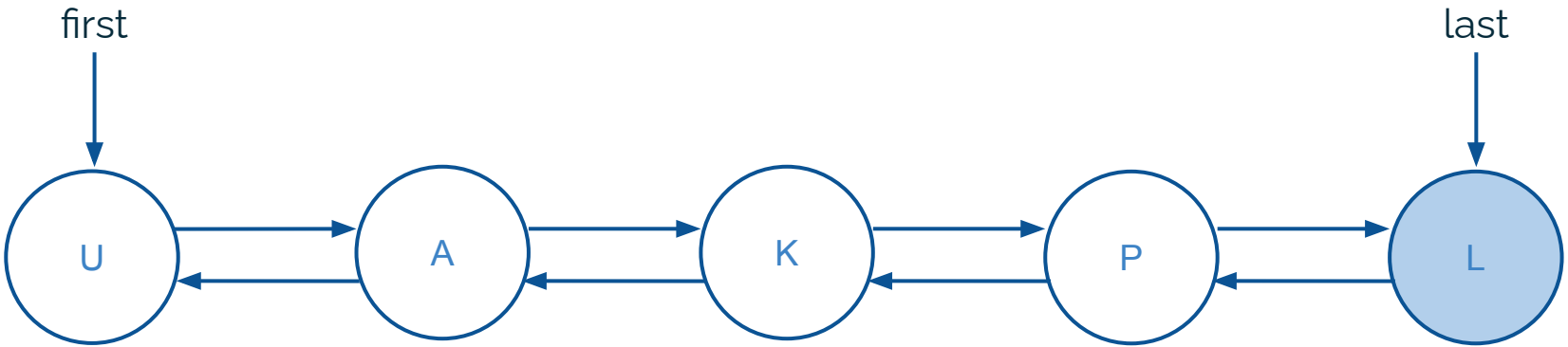
Se debe poner el enlace *next* para la siguiente letra

Listas enlazadas: Agregar nuevo elemento



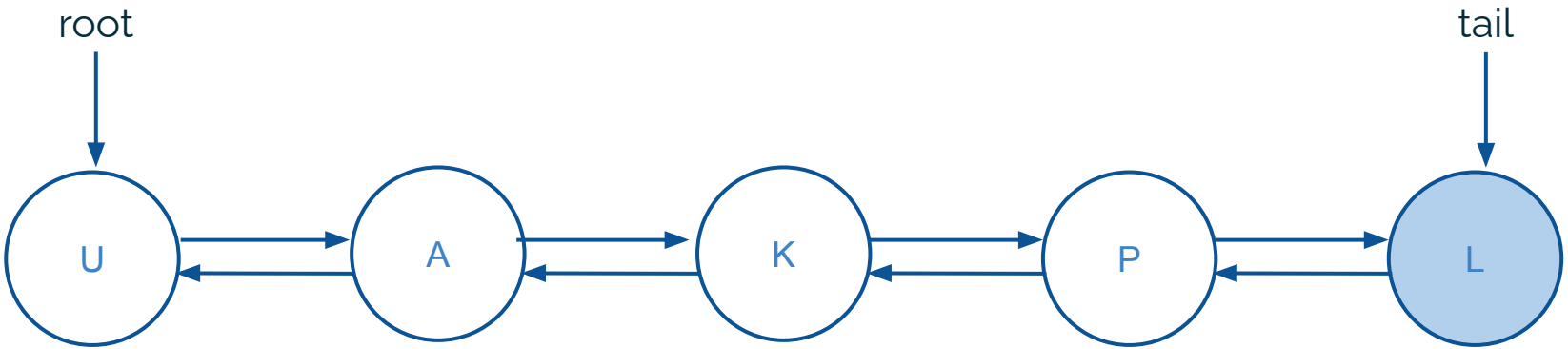
Se debe poner el enlace *prev* para la siguiente letra

Listas enlazadas: Agregar nuevo elemento



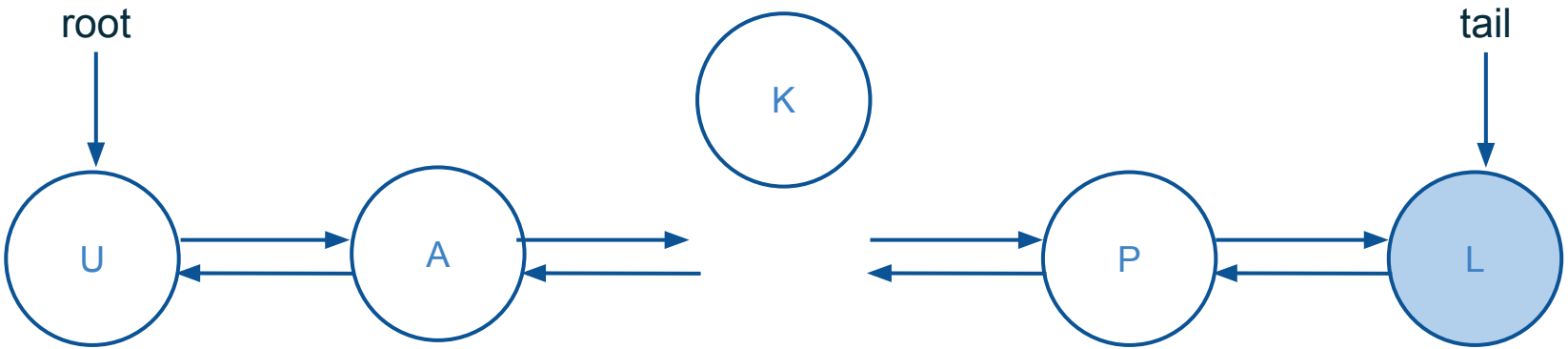
Se debe poner la nueva referencia al *first*

Listas enlazadas: eliminar



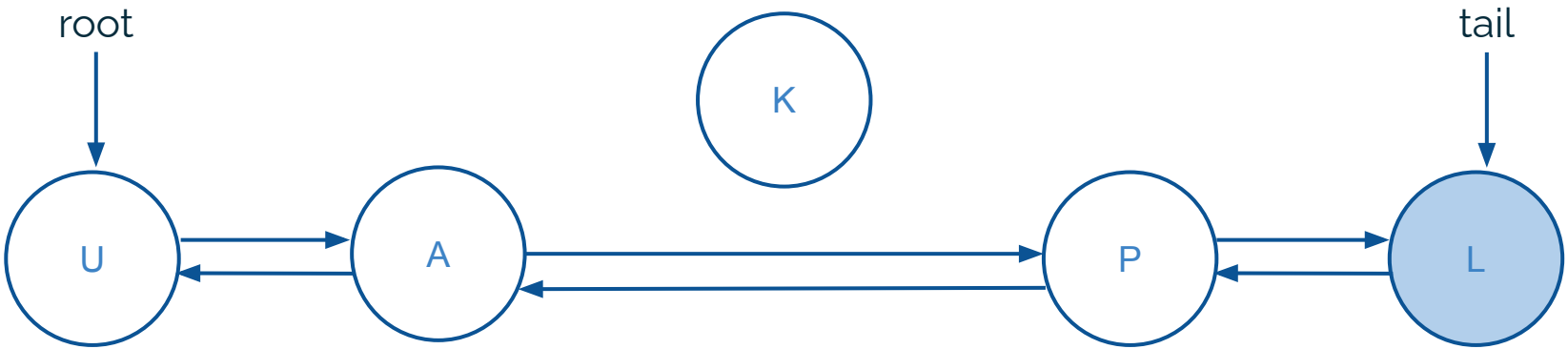
Si desea eliminar la letra K. Se debe modificar únicamente las referencias

Listas enlazadas: eliminar



Si desea eliminar la letra K. Se debe modificar únicamente las referencias

Listas enlazadas: eliminar



Si desea eliminar la letra K. Se debe modificar únicamente las referencias

Listas enlazadas: eliminar

Si desea eliminar la letra K. Se debe modificar únicamente las referencias.

Un objeto NO referenciado es eliminado por el Garbage Collector

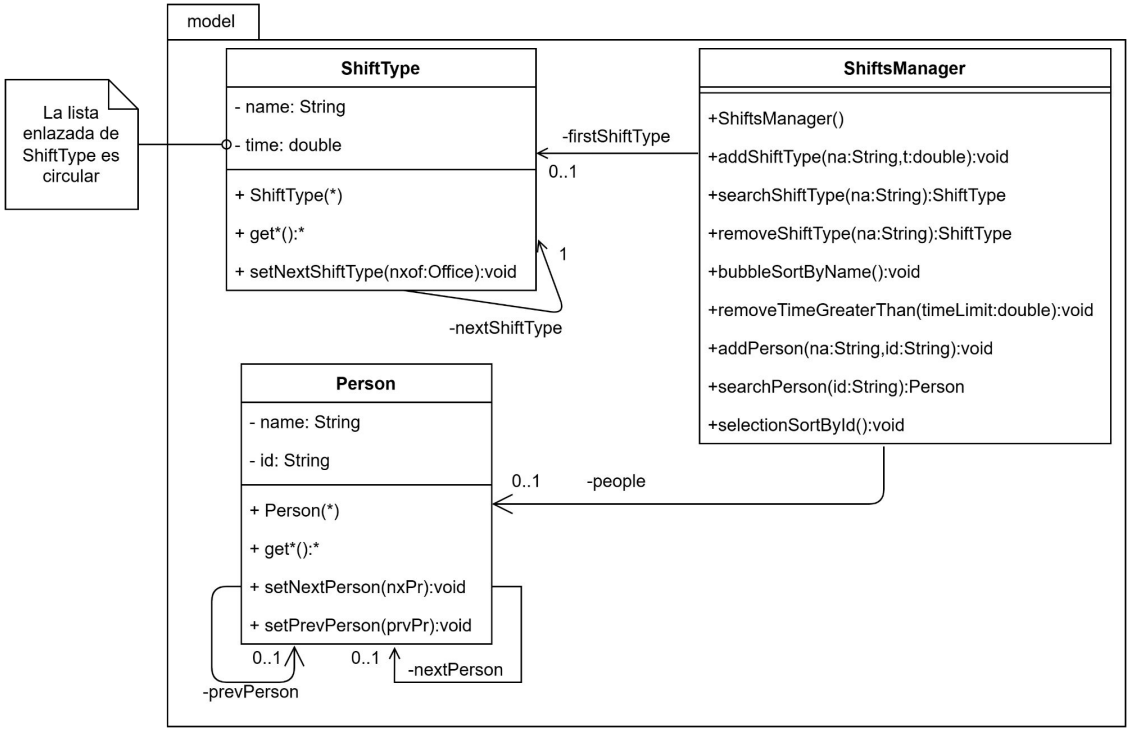
Ventajas de usar estructuras lineales

- Es más rápido agregar elementos al principio o al final de la lista.
- Tamaño de datos dinámicos, no es necesario definir el espacio inicial como un arreglo o matriz.
- Es bueno para insertar una gran cantidad de datos porque al agregar o quitar elementos no habrá desbordamiento de memoria.
- Flexibilidad, su comportamiento sirve como abstracción para múltiples estructuras de datos como colas, pilas y grafos.

Desventajas de usar estructuras lineales

- Acceso restringido, no es posible acceder a elementos por índice.
- Utiliza más memoria que una matriz. Esto se debe a que es necesario crear referencias a objetos entre nodos. Esto significa que necesitamos crear la siguiente referencia de objeto. Y para una lista doblemente enlazada, son dos punteros siguiente y anterior.

Caso de estudio



Universidad ICESI

Gracias.😄

refs: https://javachallengers.com/linked-list-data-structure-with-java/