Arrays versus Listas ligadas 1.

		Array (não ordenado)	Lista simplesmente ligada
Inserção	Princípio	$O(N)$ ①	$O(1)$ ④
	Meio	$O(N)$	$O(1)$ ④
	Fim	$O(1)$	$O(1)$ ④
Remoção	Princípio	$O(N)$ ②	$O(1)$ ④
	Fim	$O(1)$ ③	$O(1)$ ④
Indexação	Em qualquer ponto	$O(1)$	$O(N)$ ⑤

① Implica deslocar os elementos para a direita e depois é que insere

② Implica deslocamentos

③ Basta decrementar o contador de elementos e a última posição passa a não ser considerada.

④ Admitindo que se tem uma referência para o nó anterior/seguinte.

⑤ Indexação no <u>princípio</u> da lista é $O(1)$, e no <u>fim</u> também, caso tenha uma referência para "last". No resto, é $O(N)$.

Listas Simplesmente Ligadas

"Morada"

"REFERÊNCIA"

Lista sem sentinela:

dado que a referência "head" aponta para null ou diretamente para o 1º elemento

main:

```
var head: IntNode? = null
```

Stack memory

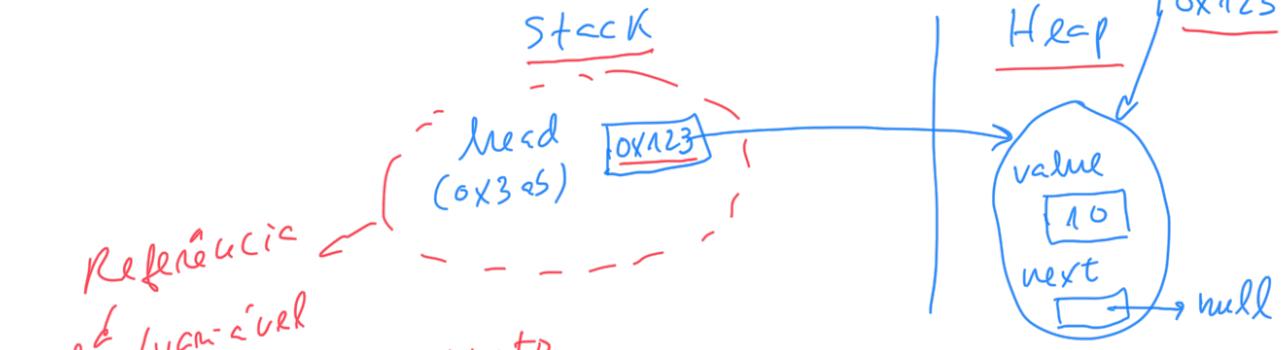
Heap memory

endereço 0x3a5 --> head | null

todas as variáveis/valores têm o seu endereço, ex: 0x3a5

```
head = IntNode(10)
```

Nó reside no endereço 0x123



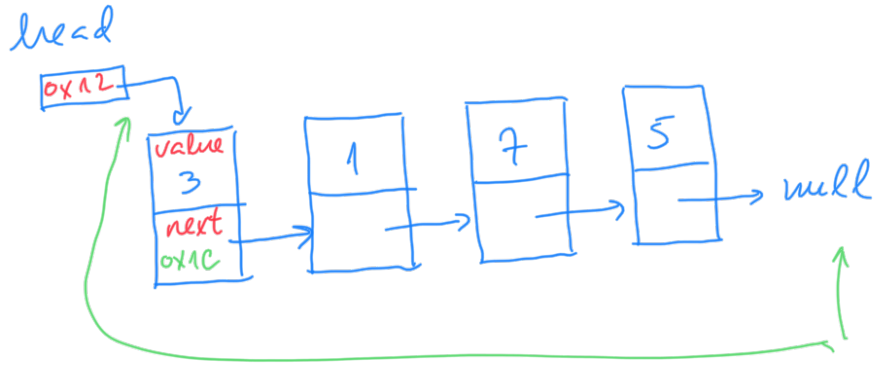
Referência: valor/variável que guarda um endereço dum objeto

Stack - zona de memória onde são alocadas variáveis locais / parâmetros, endereço de retorno de métodos / funções

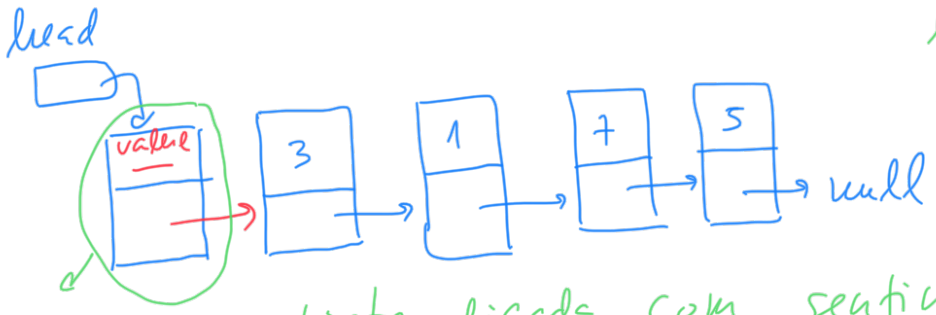
Heap - zona de memória onde são alojados os objetos

3.

Lista contendo elementos: 3, 1, 7, 5 4.



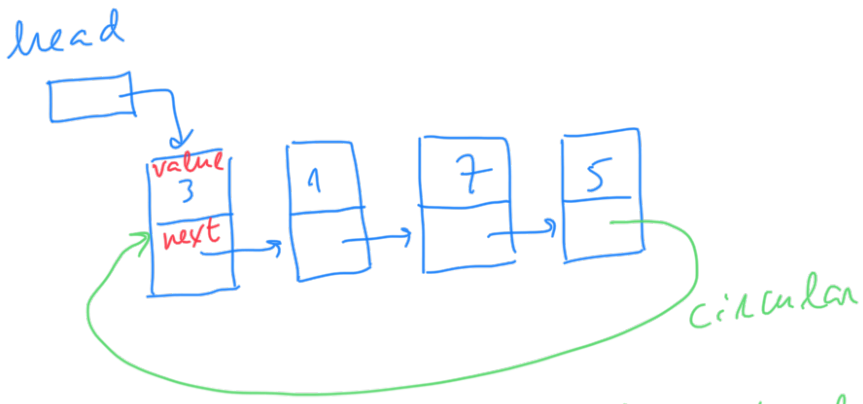
Lista simplesmente ligada, não circular, e sem sentinela



Nó sentinela

Lista ligada com sentinela, não circular

lista simplesmente ligada, sem sentinela, circular



Também existe lista simplesmente ligada, com sentinela, e circular.

Lista simplesmente ligada, sem sentinela, não circular

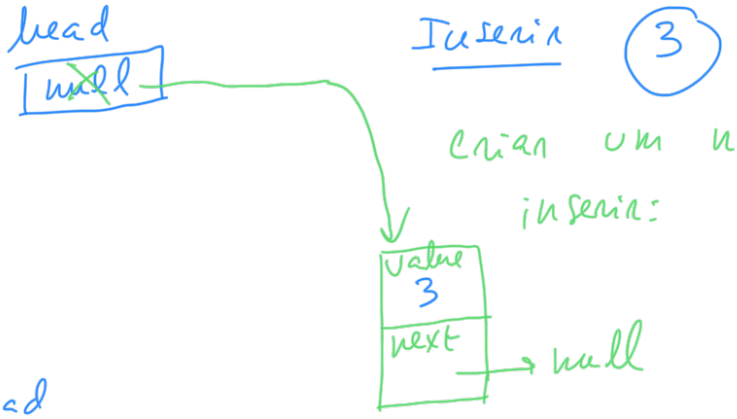
Inserção no princípio:

2 casos:

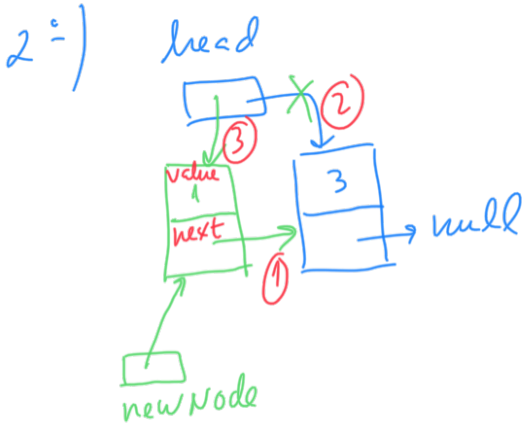
1º) lista está vazia

2º) " não " "

1º)



2º)



Inserir (1)

Criar nó e inserir no princípio

(7.)

```
class IntNode {
    var value: Int
    var next: IntNode?

    constructor() { }
    constructor(v: Int) {
        value = v
        next = null
    }
}
```

Na prática, é representado por um int que guarda o endereço de memória do nó seguinte

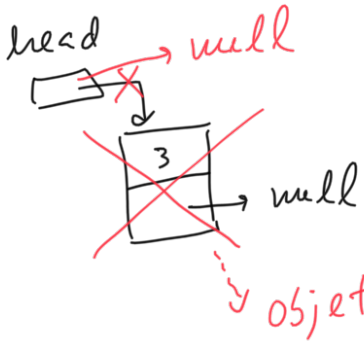
```
class SIntList {
    var head: IntNode?

    fun addFront(v: Int) {
        val newNode: IntNode =
                IntNode(v)
        if (head == null)
            head = newNode
        else {
            (1) newNode.next = head
            (2)
            (3)  } head = newNode
        }
    }
}
```

Remove front

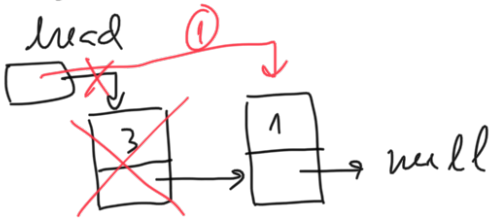
1º) lista vazia → não é possível

2º) lista com 1 elemento



objeto é limpo pelo GC (Garbage Collector)

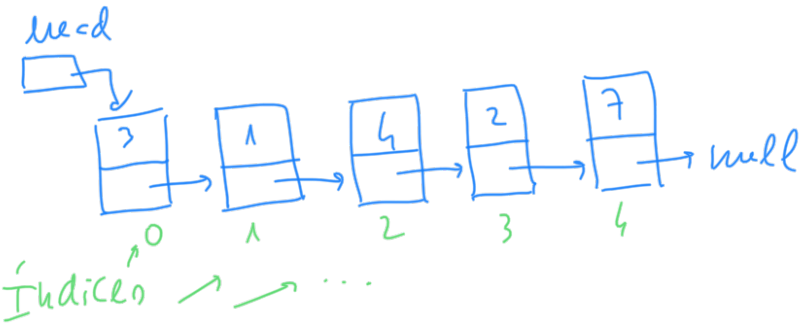
3º) lista com 2 ou mais elementos



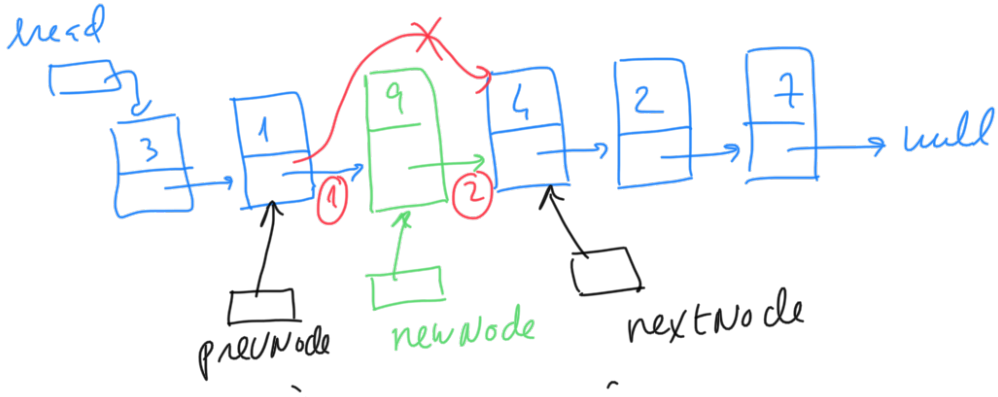
```
fun removeFront() : Int? {
    if (head == null)
        return null
    else {
        val elem = head.value
        head = head.next
        return elem
    }
}
```

→ elemento que está no 1º nó (nó a ser removido)

Inserção num dado índice



Ex: Inserir (9) no índice (2)



Referências auxiliares que permitem ligar por qualquer ordem

↓

vantagem!

```
fun insert(v: Int, index: Int): Boolean
    // sem sentinela => main difícil
    if (head == null)
        return false // não inseriu

    var prevNode: IntNode = head
    var i = 0
    while (i < index) {
```

T.P.C.