



Estructura de Datos

Maria C. Torres

Maria C. Torres

Ing. Electrónica (UNAL)

M.E. Ing. Eléctrica (UPRM)

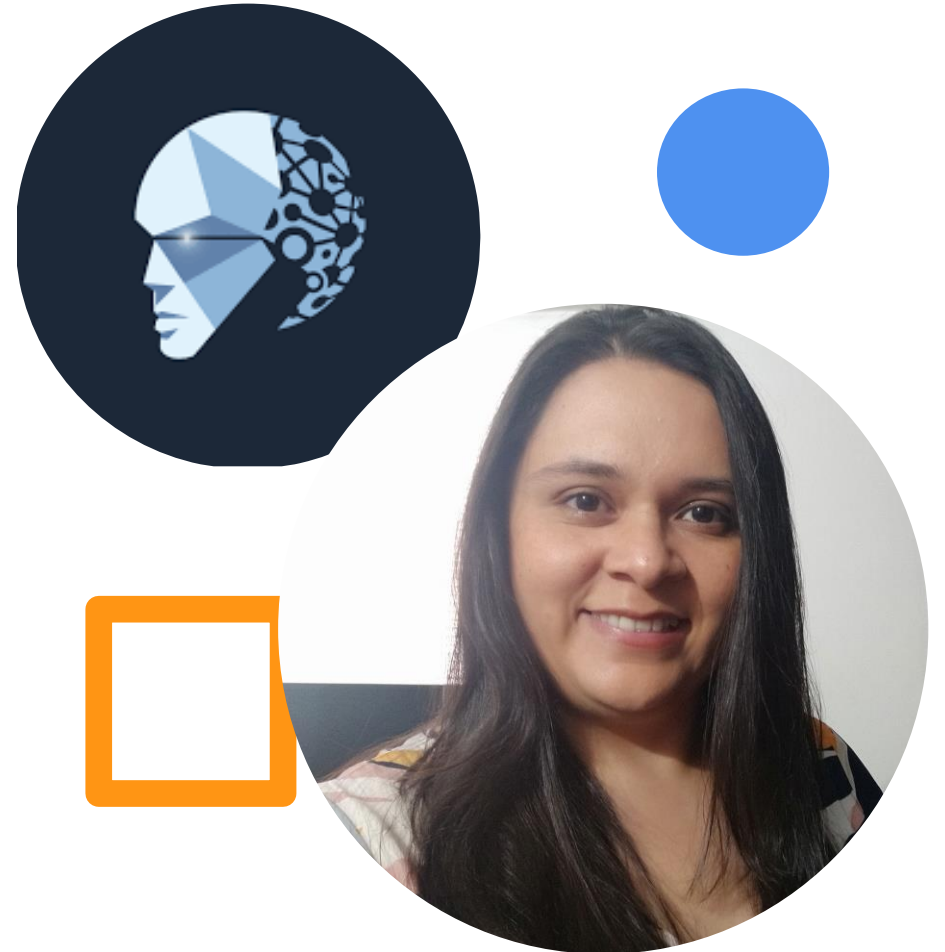
Ph.D. Ciencias e Ingeniería de la Computación y la Información (UPRM)


Profesora asociada

Dpto. Ciencias de la Computación y la Decisión


mctorresm@unal.edu.co

HORARIO DE ATENCIÓN: Martes 10:00 am a
12:00 m – Oficina 313 M8A





Contenido del Curso

- 
- ☐ Introducción: revisión fundamentos y POO
 - ☐ Análisis de complejidad
 - ☐ Arreglos
 - ☐ Listas enlazadas
 - ☐ **Pilas y colas**
 - ☐ Heap
 - ☐ Árboles binarios
 - ☐ Tablas hash
 - ☐ Grafos



Pilas y colas

- ❑ Pilas – Stacks
 - ❑ Implementación usando arreglos y lista simple
- ❑ **Colas – Queue**
 - ❑ Implementación usando arreglos y lista simple
- ❑ Análisis de complejidad operaciones
- ❑ Aplicaciones y algoritmos

COLA - QUEUE

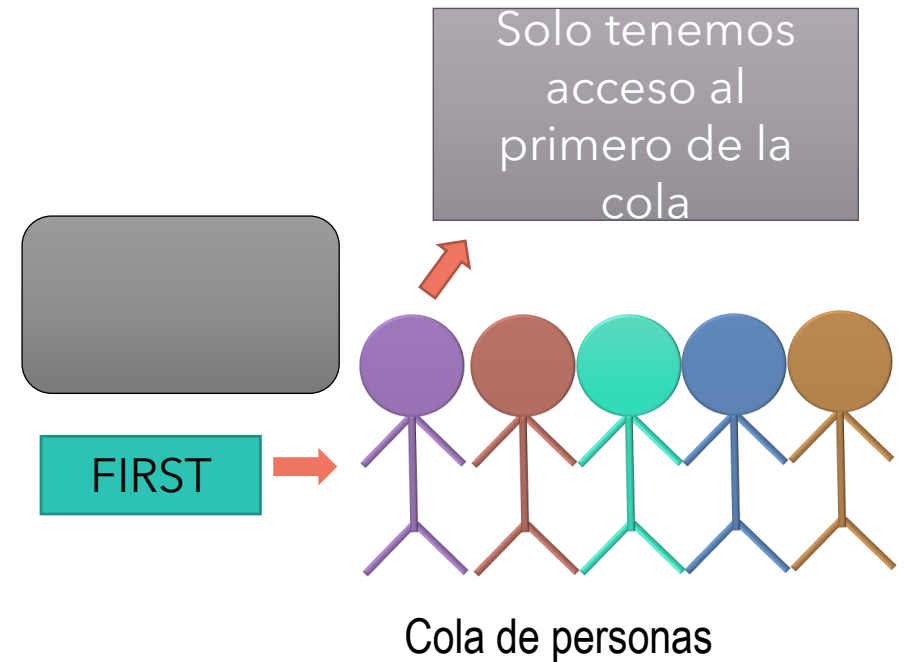
- ❑ **Acceso secuencial - limitado:** solo podemos acceder a un elemento de la estructura
- ❑ Una cola o QUEUE es una colección de objetos que son insertados o eliminados de acuerdo con el principio "**primero en entrar - primero en salir**"
- ❑ En ingles este principio se denomina **FIFO** (first-in first-out), es decir, el objeto que lleva más tiempo en la cola es el único elemento al cual se tiene acceso



COLA - QUEUE

Una cola o QUEUE tiene dos atributos:

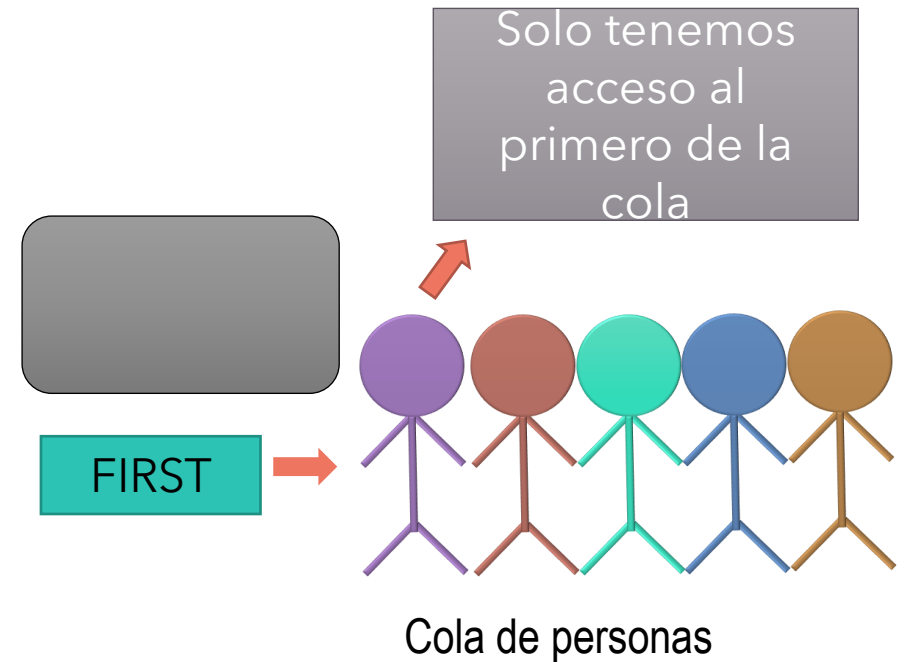
- ❑ FIRST: primer objeto de la cola - es el único elemento al que se tiene acceso y se puede eliminar
- ❑ SIZE: número de objetos en la cola



COLA - QUEUE

Una cola o QUEUE solo permite dos operaciones

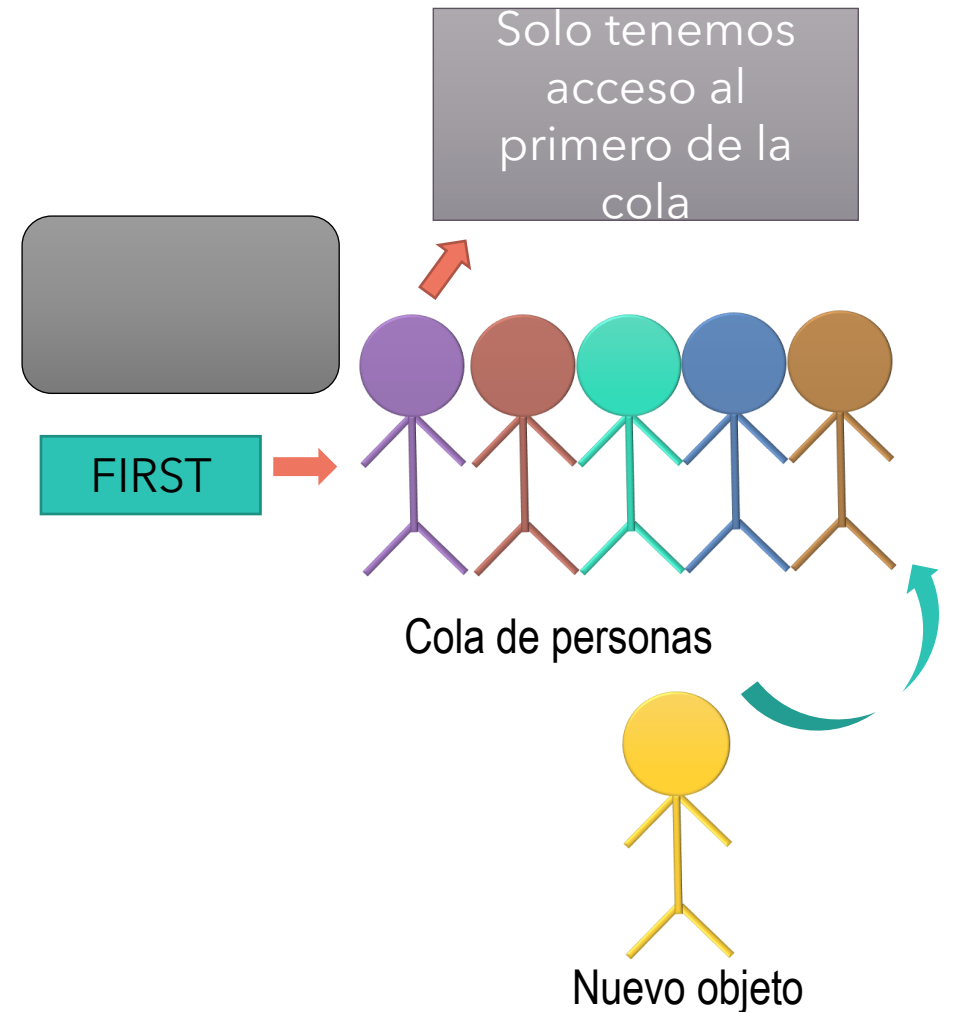
- ❑ ENQUEUE: operación de ingresar un nuevo objeto a la cola, este objeto siempre se agrega al final de la cola
- ❑ DEQUEUE: operación de sacar o remover el objeto al principio de la cola



COLA - QUEUE

Una cola o QUEUE solo permite dos operaciones

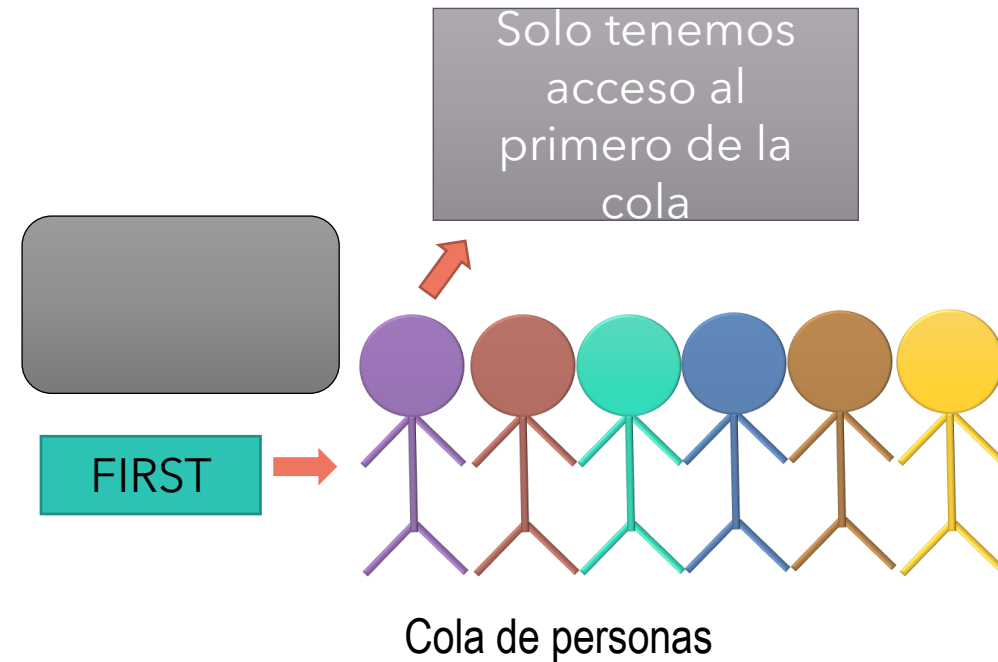
- ❑ **ENQUEUE:** operación de ingresar un nuevo objeto a la cola, este objeto siempre se agrega al final de la cola
- ❑ **DEQUEUE:** operación de sacar o remover el objeto al principio de la cola



COLA - QUEUE

Una cola o QUEUE solo permite dos operaciones

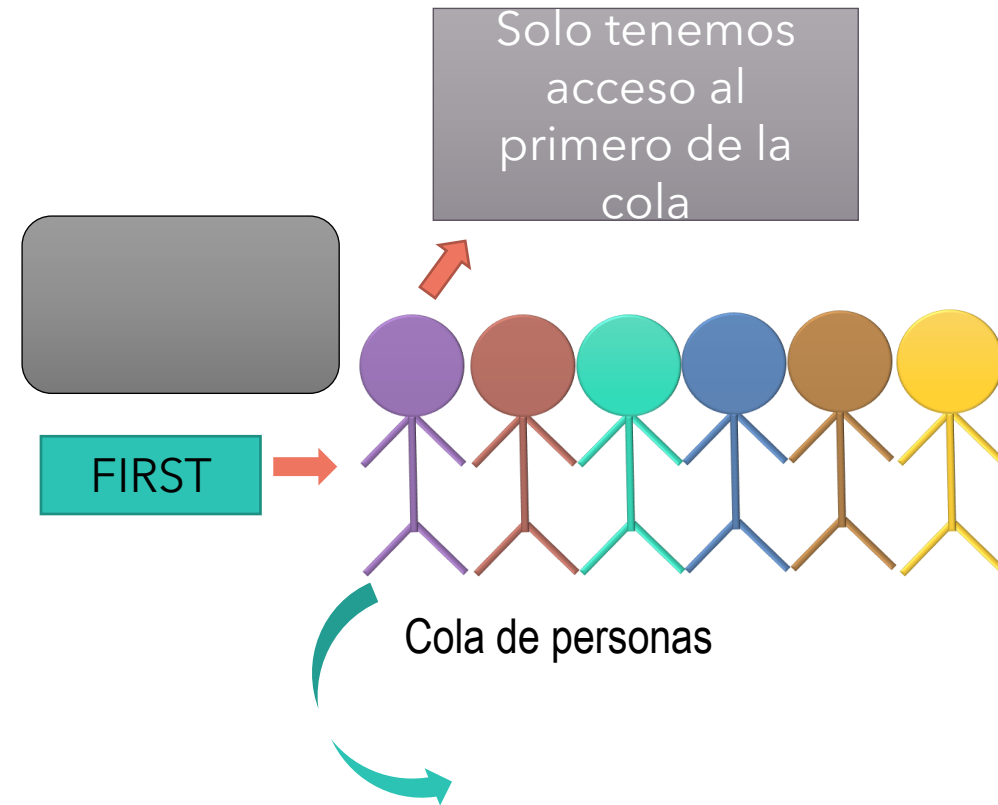
- ❑ **ENQUEUE:** operación de ingresar un nuevo objeto a la cola, este objeto siempre se agrega al final de la cola
- ❑ **DEQUEUE:** operación de sacar o remover el objeto al principio de la cola



COLA - QUEUE

Una cola o QUEUE solo permite dos operaciones

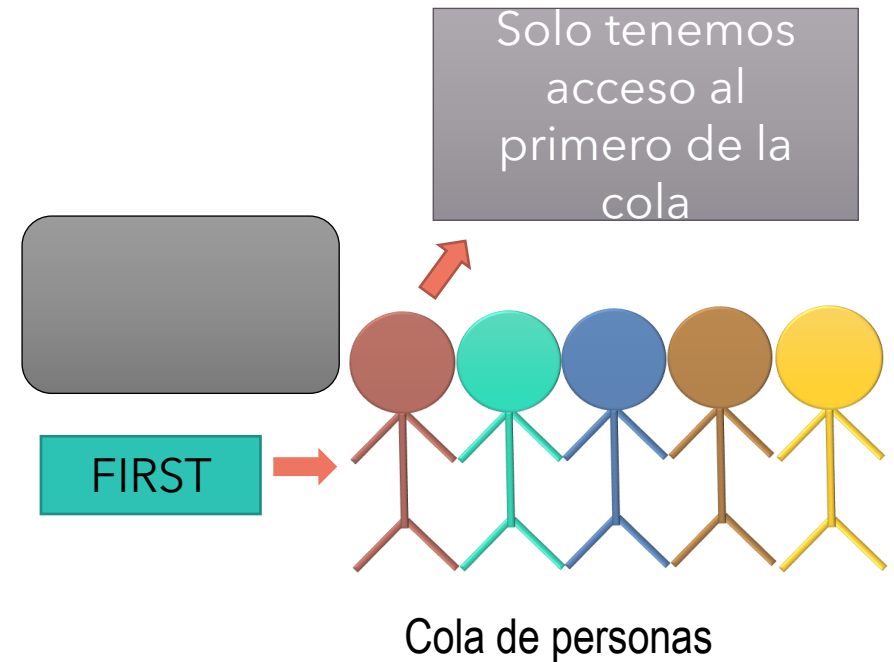
- ❑ **ENQUEUE**: operación de ingresar un nuevo objeto a la cola, este objeto siempre se agrega al final de la cola
- ❑ **DEQUEUE**: operación de sacar o remover el objeto al principio de la cola



COLA - QUEUE

Una cola o QUEUE solo permite dos operaciones

- ❑ **ENQUEUE**: operación de ingresar un nuevo objeto a la cola, este objeto siempre se agrega al final de la cola
- ❑ **DEQUEUE**: operación de sacar o remover el objeto al principio de la cola



Aplicaciones

- Sistemas de espera para atención de usuarios



COLA - QUEUE

- Documentos en espera para impresión

Impresora	Documento	Ver					
Nombre del documento	Estado	Propietario	Páginas	Tamaño	Enviado	Puerto	
Página de prueba	Imprimiendo	aulaClic	1	192 KB	15:07:26 29/08/2007	LPT1:	
Microsoft Word - Temario aulaClic...	Pausado	aulaClic	1	2,49 KB	15:30:34 29/08/2007		
proyecto.txt - Bloc de notas		Bruno	7	226 KB	20:28:39 29/08/2007		

3 documentos en la cola

- Asignación de puesto en vuelo sobrevendido





Pilas y colas

- ❑ Pilas – Stacks
 - ❑ Implementación usando arreglos y lista simple
- ❑ Colas – Queue
 - ❑ **Implementación usando arreglos y lista simple**
- ❑ Análisis de complejidad operaciones
- ❑ Aplicaciones y algoritmos

- ❑ Estudiaremos una primera implementación QUEUE usando arreglos
- ❑ Esta implementación se recomienda usar cuando requerimos un QUEUE con un número limitado y pequeño de datos

Clase ArrayQueue

Se mantienen los siguientes atributos

- Un arreglo con un tamaño por defecto que almacena los datos de la cola
- FIRST: corresponde al índice donde se encuentra el primer elemento de la cola
- REAR: corresponde al índice donde se encuentra el último elemento de la cola
- FIRST y REAR toman valores entre 0 y el tamaño del arreglo

ArrayQueue

- data[]: Object
- first: int
- rear: int

Clase ArrayQueue

Los métodos que se incluyen son:

- El constructor que recibe la capacidad inicial del arreglo
- Los métodos para obtener el número de datos en la pila y saber si está vacía

ArrayQueue

- data[]: Object
- first: int
- rear: int

+ArrayQueue(int capacity)
+size():int
+isEmpty(): boolean

Clase ArrayQueue

Los métodos que se incluyen son:

- Los métodos de enqueue (insertar) y dequeue (eliminar)
- Un método para acceder al primer elemento sin eliminarlo (operación first)

ArrayQueue

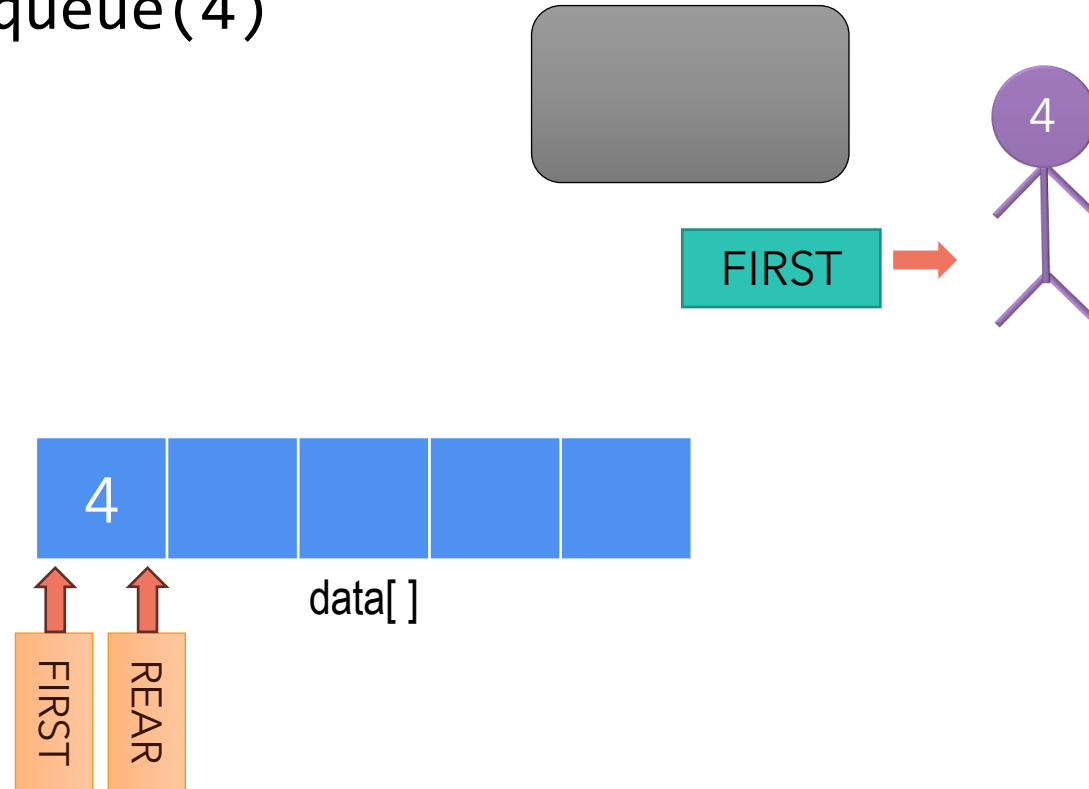
- data[]: Object
- first: int
- rear: int

+ArrayQueue(int capacity)
+size():int
+isEmpty(): boolean
+enqueue(Object e)
+dequeue():Object e
+first():Object e

COLA - QUEUE

Ejemplo de ArrayQueue:
>> enqueue(4)

ArrayQueue
- data[]: Object - first: int - rear: int
+ArrayQueue(int capacity) +size():int +isEmpty(): boolean +enqueue(Object e) +dequeue():Object e +first():Object e



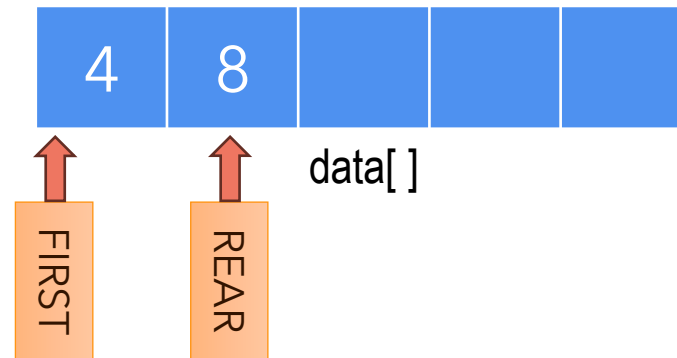
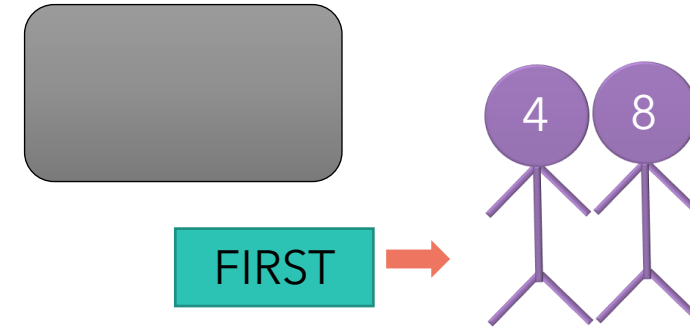
COLA - QUEUE

Ejemplo de ArrayQueue:

```
>> enqueue(4)
```

```
>> enqueue(8)
```

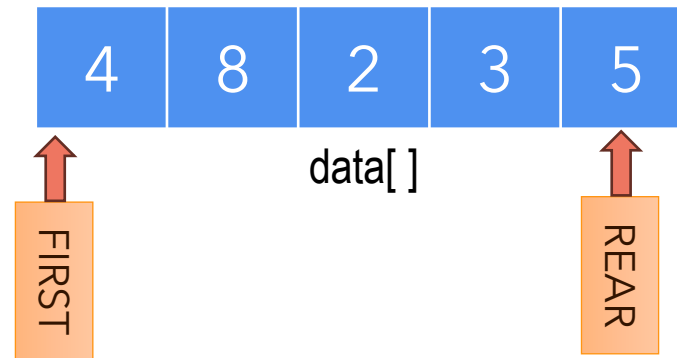
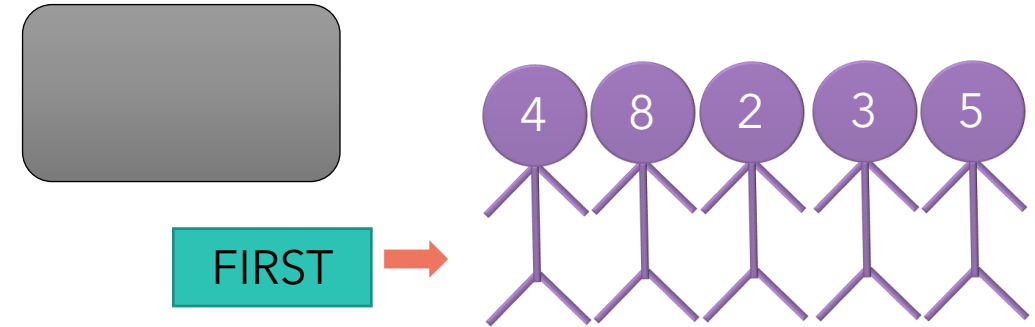
ArrayQueue
<ul style="list-style-type: none">- data[]: Object- first: int- rear: int
<ul style="list-style-type: none">+ArrayQueue(int capacity)+size():int+isEmpty(): boolean+enqueue(Object e)+dequeue():Object e+first():Object e



COLA - QUEUE

Ejemplo de ArrayQueue:

```
>> enqueue(4)
>> enqueue(8)
>> enqueue(2)
>> enqueue(3)
>> enqueue(5)
```

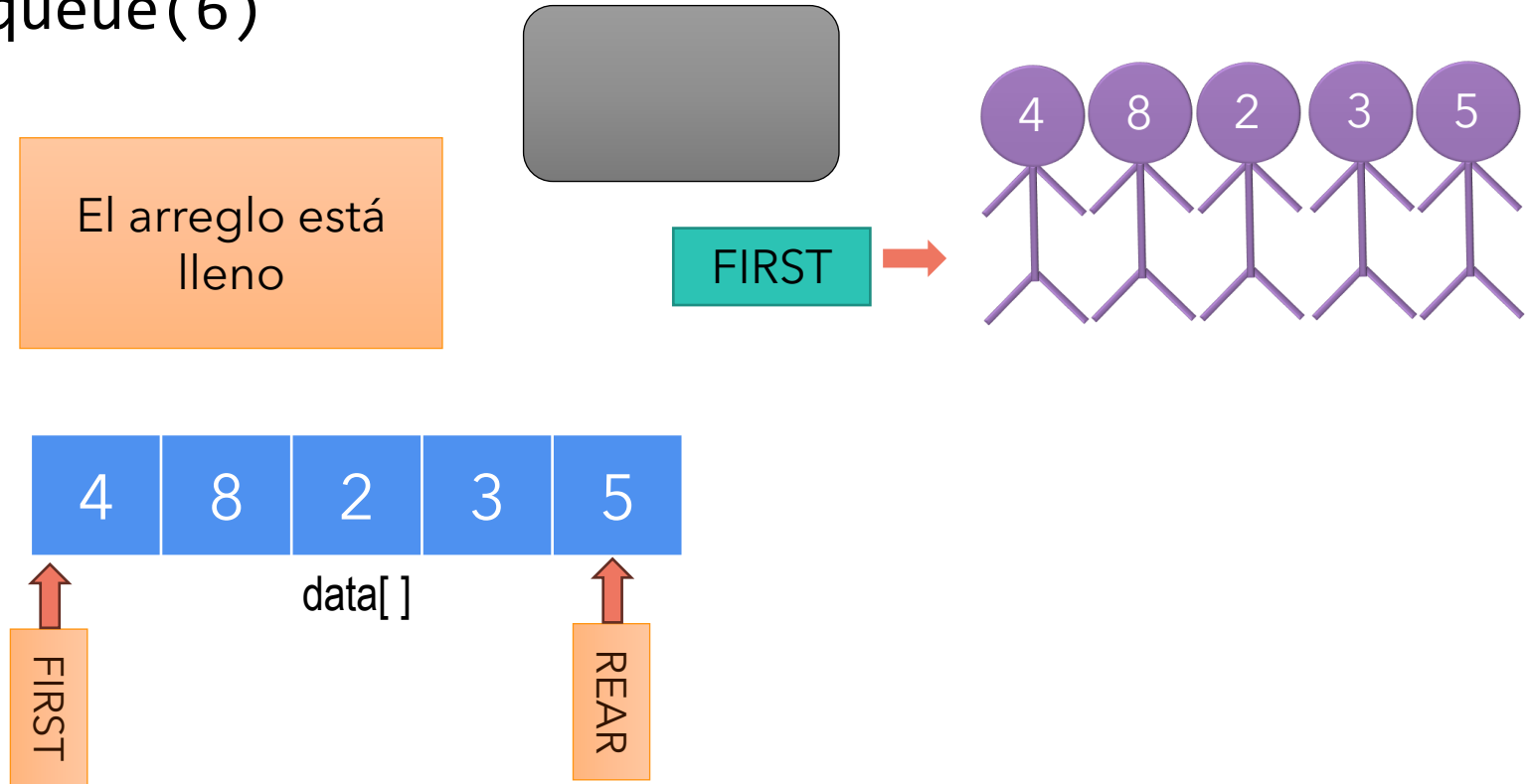


ArrayQueue
- data[]: Object - first: int - rear: int
+ArrayQueue(int capacity) +size():int +isEmpty(): boolean +enqueue(Object e) +dequeue():Object e +first():Object e

COLA - QUEUE

Ejemplo de ArrayQueue:
>> enqueue(6)

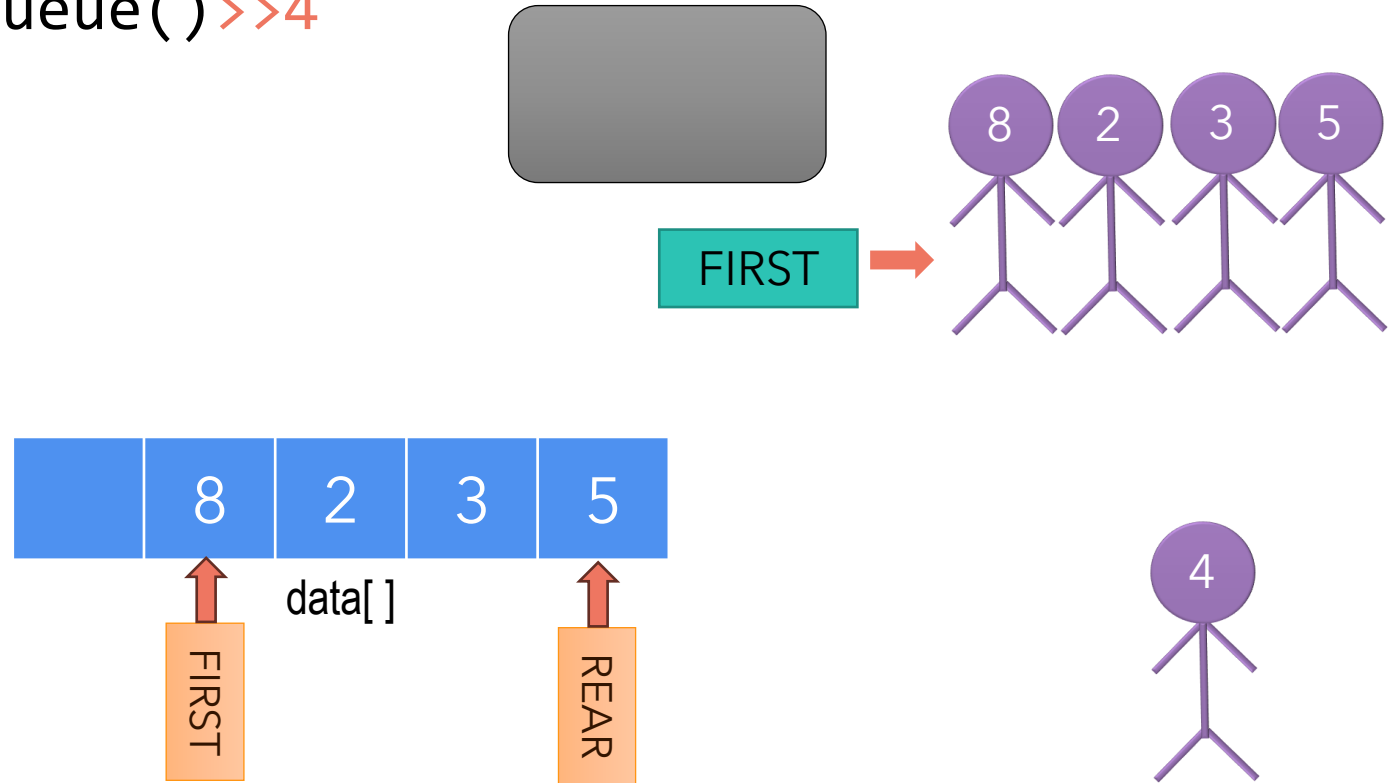
ArrayQueue
<ul style="list-style-type: none">- data[]: Object- first: int- rear: int
<ul style="list-style-type: none">+ArrayQueue(int capacity)+size():int+isEmpty(): boolean+enqueue(Object e)+dequeue():Object e+first():Object e



COLA - QUEUE

Ejemplo de ArrayQueue:
>> dequeue() >> 4

ArrayQueue
<ul style="list-style-type: none">- data[]: Object- first: int- rear: int
<ul style="list-style-type: none">+ArrayQueue(int capacity)+size():int+isEmpty(): boolean+enqueue(Object e)+dequeue():Object e+first():Object e



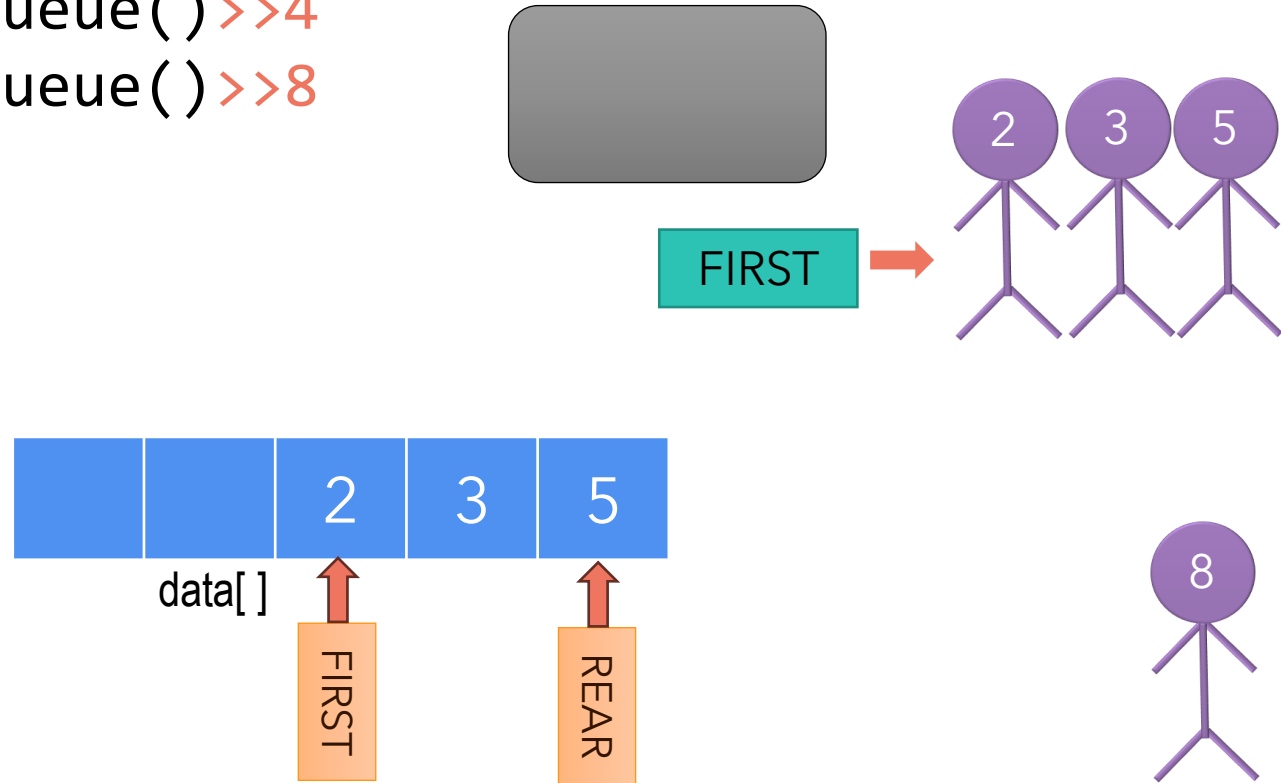
COLA - QUEUE

Ejemplo de ArrayQueue:

>> dequeue() >> 4

>> dequeue() >> 8

ArrayQueue
- data[]: Object - first: int - rear: int
+ArrayQueue(int capacity) +size():int +isEmpty(): boolean +enqueue(Object e) +dequeue():Object e +first():Object e

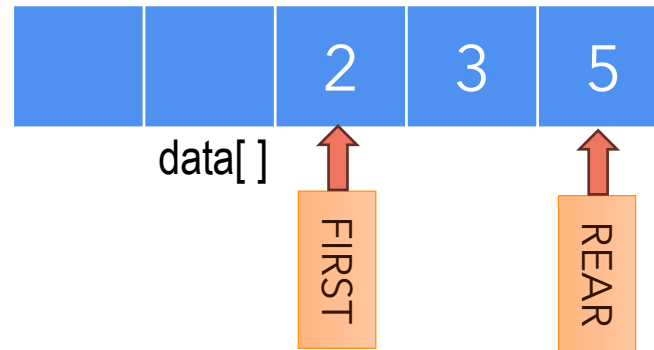
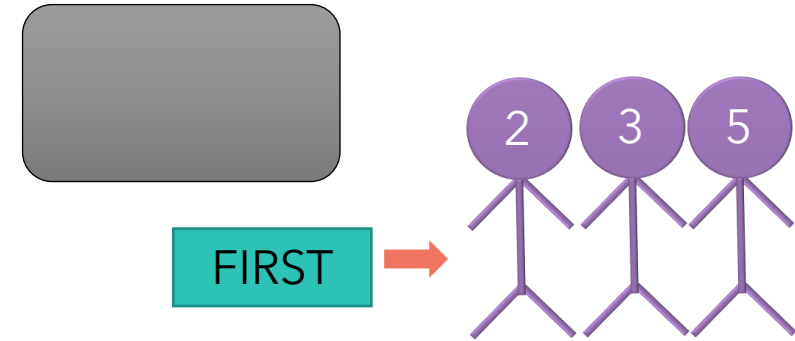


COLA - QUEUE

Ejemplo de ArrayQueue:

```
>> dequeue() >> 4  
>> dequeue() >> 8  
>> first() >> 2
```

ArrayQueue
<ul style="list-style-type: none">- data[]: Object- first: int- rear: int
<ul style="list-style-type: none">+ArrayQueue(int capacity)+size():int+isEmpty(): boolean+enqueue(Object e)+dequeue():Object e+first():Object e



COLA - QUEUE

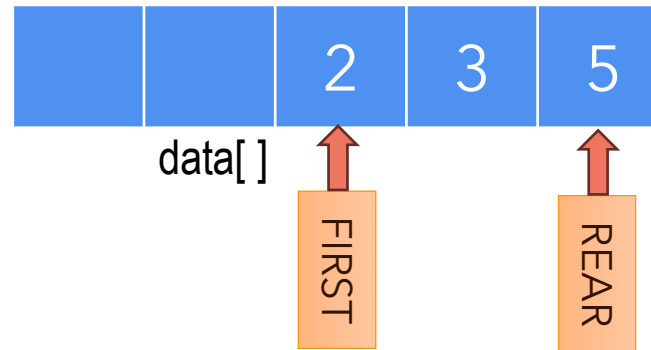
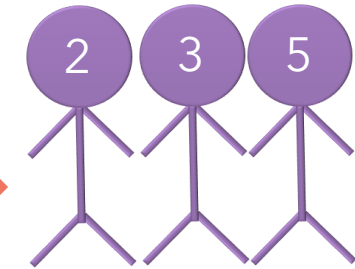
Ejemplo de ArrayQueue:
>> enqueue(1)

ArrayQueue
<ul style="list-style-type: none">- data[]: Object- first: int- rear: int
<ul style="list-style-type: none">+ArrayQueue(int capacity)+size():int+isEmpty(): boolean+enqueue(Object e)+dequeue():Object e+first():Object e

Si hay espacio en el arreglo!!



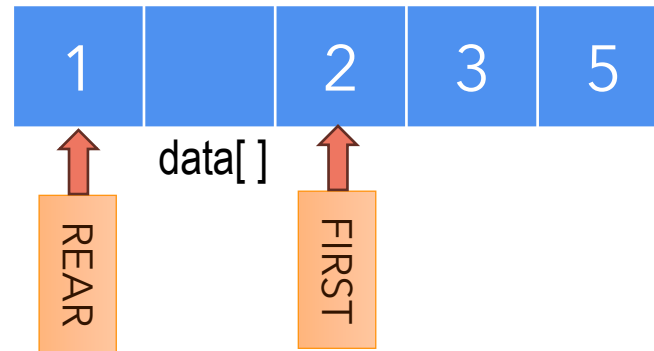
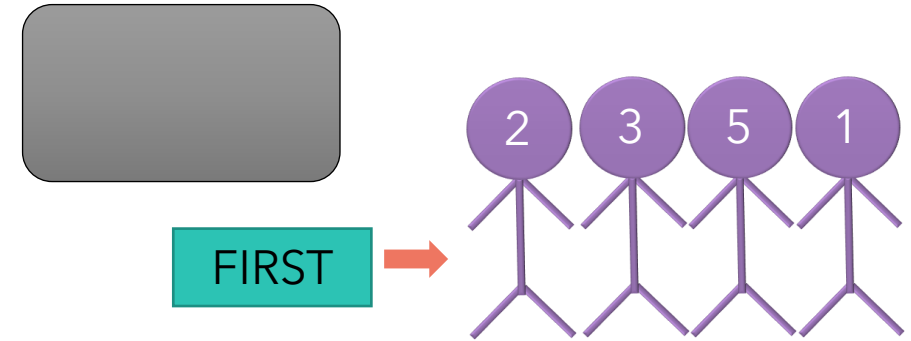
FIRST



COLA - QUEUE

Ejemplo de ArrayQueue:
>> enqueue(1)

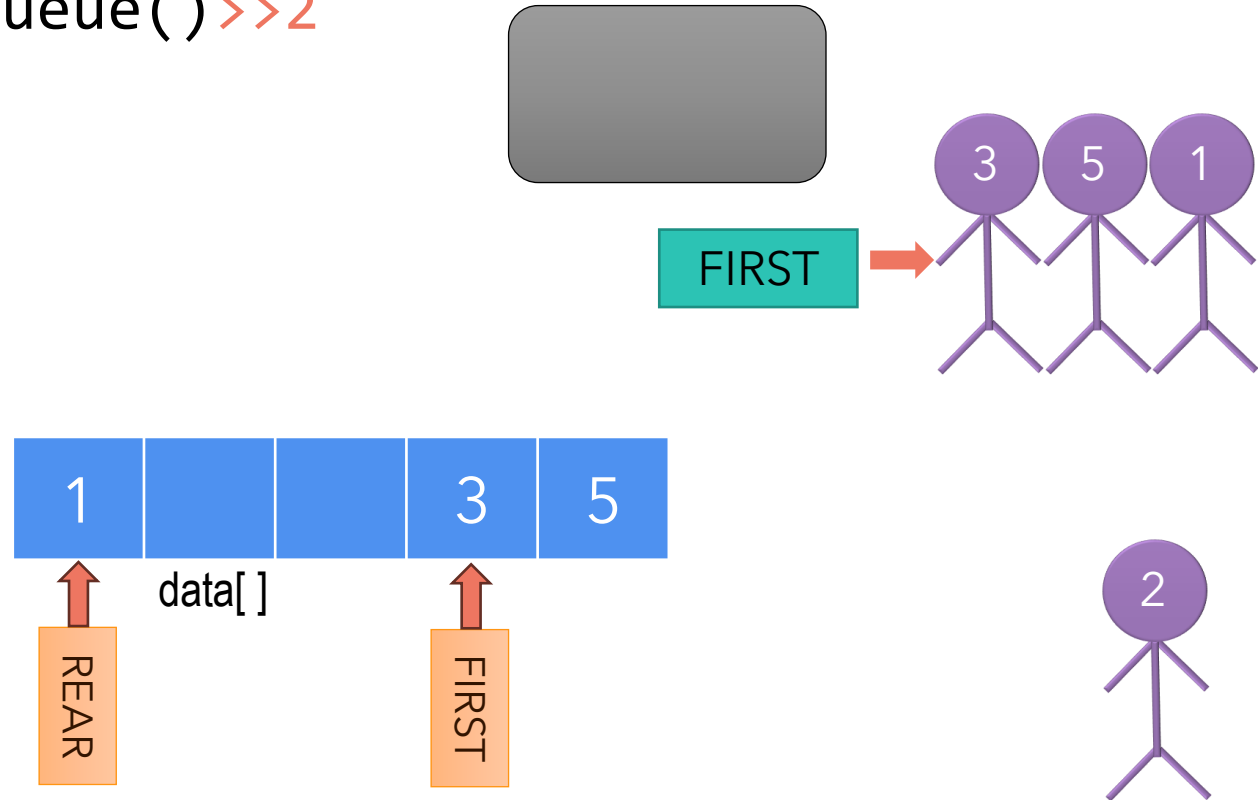
ArrayQueue
<ul style="list-style-type: none">- data[]: Object- first: int- rear: int
<ul style="list-style-type: none">+ArrayQueue(int capacity)+size():int+isEmpty(): boolean+enqueue(Object e)+dequeue():Object e+first():Object e



COLA - QUEUE

Ejemplo de ArrayQueue:
>> dequeue() >> 2

ArrayQueue
<ul style="list-style-type: none">- data[]: Object- first: int- rear: int
<ul style="list-style-type: none">+ArrayQueue(int capacity)+size():int+isEmpty(): boolean+enqueue(Object e)+dequeue():Object e+first():Object e



Seudocódigo implementación ArrayQueue:
Constructor

```
ArrayQueue(int capacity)
    data = new Object[capacity]
    first = -1
    rear = -1
```

Recuerda: la capacidad (capacity)
determina el tamaño máximo de la cola

ArrayQueue

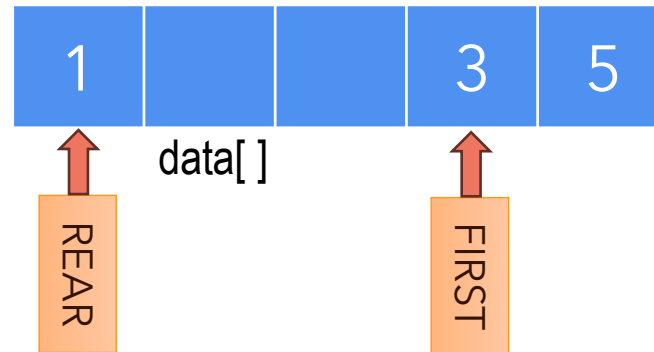
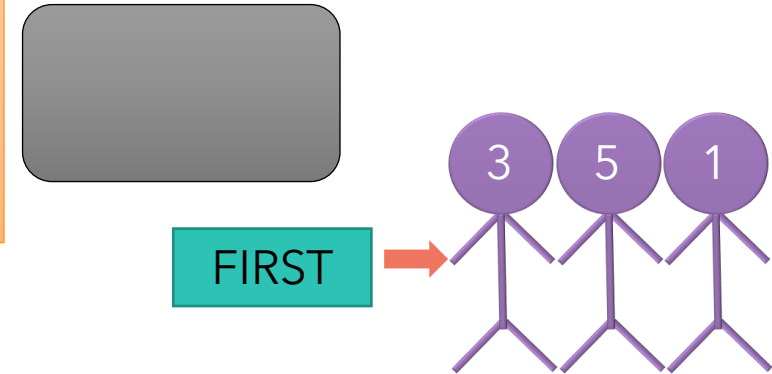
- data[]: Object
- first: int
- rear: int

- +ArrayQueue(int capacity)
- +size():int
- +isEmpty(): boolean
- +enqueue(Object e)
- +dequeue():Object e
- +first():Object e

Ejemplo de ArrayQueue:

ArrayQueue
<ul style="list-style-type: none">- data[]: Object- first: int- rear: int
<ul style="list-style-type: none">+ArrayQueue(int capacity)+size():int+isEmpty(): boolean+enqueue(Object e)+dequeue():Object e+first():Object e

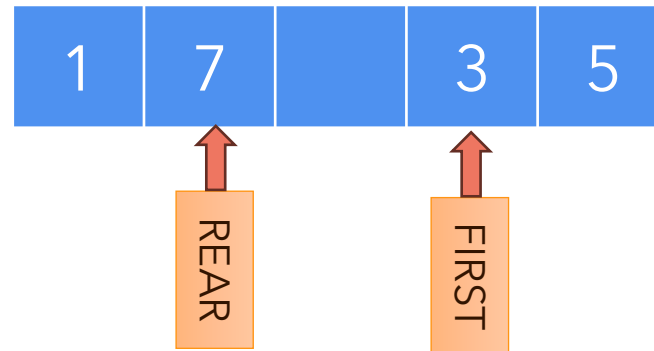
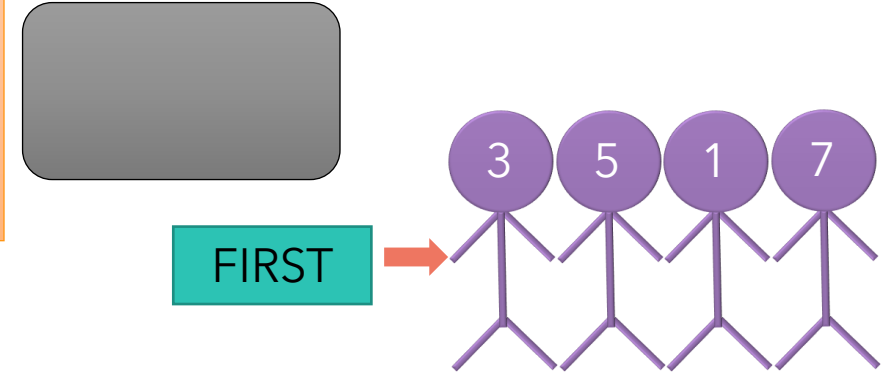
El número de datos en la cola se calcula a partir de la ecuación
$$[(\text{capacity} - \text{first} + \text{rear}) \bmod \text{capacity}] + 1$$
$$(5 - 3 + 0) \% 5 + 1 = 2 \% 5 + 1 = 3$$



Ejemplo de ArrayQueue:

ArrayQueue
<ul style="list-style-type: none">- data[]: Object- first: int- rear: int
<ul style="list-style-type: none">+ArrayQueue(int capacity)+size():int+isEmpty(): boolean+enqueue(Object e)+dequeue():Object e+first():Object e

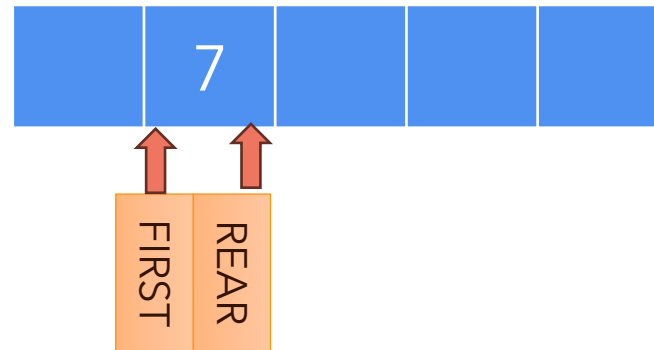
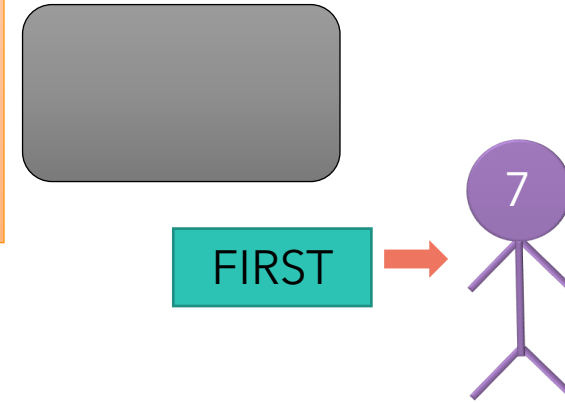
El número de datos en la cola se calcula a partir de la ecuación
$$[(\text{capacity} - \text{first} + \text{rear}) \bmod \text{capacity}] + 1$$
$$(5 - 3 + 1) \% 5 + 1 = 3 \% 5 + 1 = 4$$



Ejemplo de ArrayQueue:

ArrayQueue
<ul style="list-style-type: none">- data[]: Object- first: int- rear: int
<ul style="list-style-type: none">+ArrayQueue(int capacity)+size():int+isEmpty(): boolean+enqueue(Object e)+dequeue():Object e+first():Object e

El número de datos en la cola se calcula a partir de la ecuación
$$[(\text{capacity} - \text{first} + \text{rear}) \bmod \text{capacity}] + 1$$
$$(5 - 1 + 1) \% 5 + 1 = 5 \% 5 + 1 = 1$$



Seudocódigo implementación ArrayQueue:

```
size()  
    int temp  
    temp = data.length - first + rear  
    temp = temp%data.length + 1  
    return temp  
  
isEmpty()  
    return size()==0
```

ArrayQueue

- data[]: Object
- first: int
- rear: int

- +ArrayQueue(int capacity)
- +size():int
- +isEmpty(): boolean
- +enqueue(Object e)
- +dequeue():Object e
- +first():Object e

Seudocódigo implementación ArrayQueue:

Método para agregar un elemento a la cola

Si la cola no esta llena

1.1 Incrementamos en uno la posición de rear.

Como rear puede llegar a ser mayor que la capacidad del arreglo usamos la operación modulo para calcular la posición correcta

1.2 Insertamos el dato en la posición rear

ArrayQueue

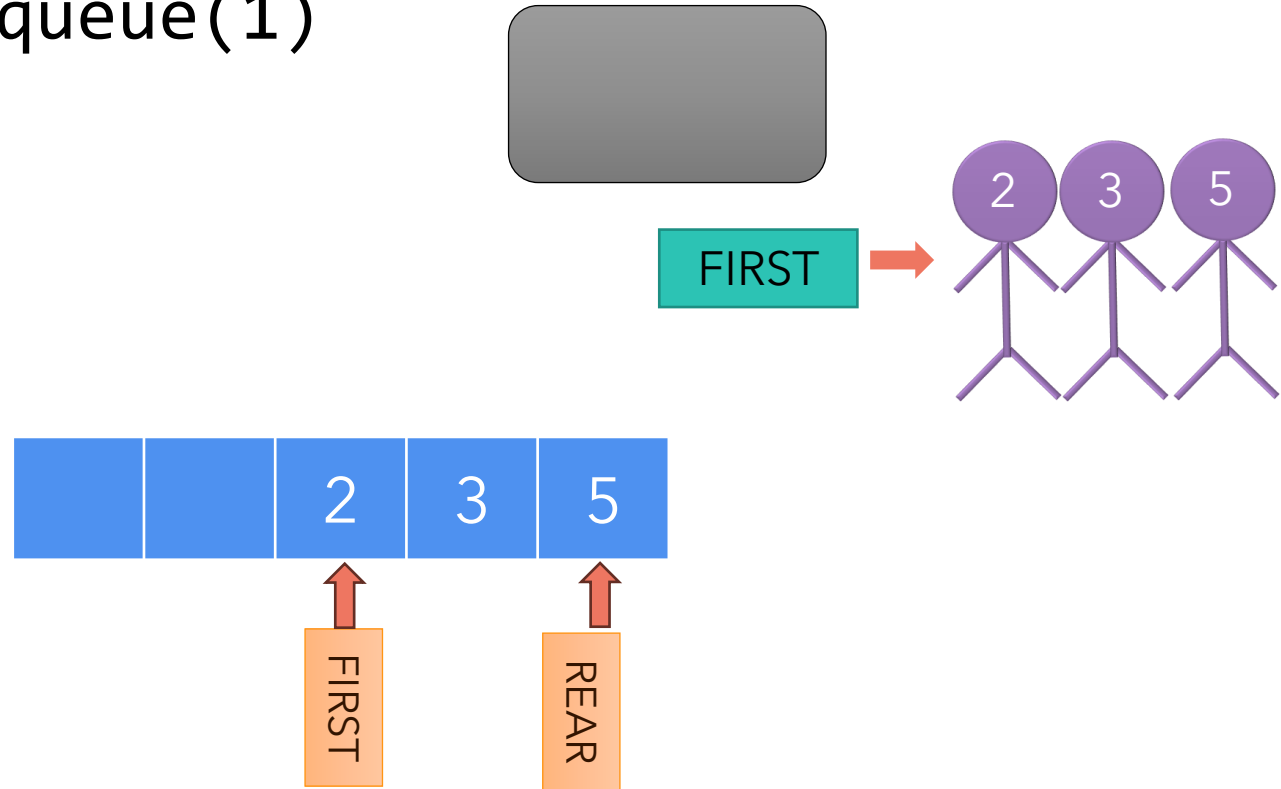
- data[]: Object
- first: int
- rear: int

- +ArrayQueue(int capacity)
- +size():int
- +isEmpty(): boolean
- +enqueue(Object e)
- +dequeue():Object e
- +first():Object e

COLA - QUEUE

Ejemplo de ArrayQueue:
>> enqueue(1)

ArrayQueue
- data[]: Object - first: int - rear: int
+ArrayQueue(int capacity) +size():int +isEmpty(): boolean +enqueue(Object e) +dequeue():Object e +first():Object e



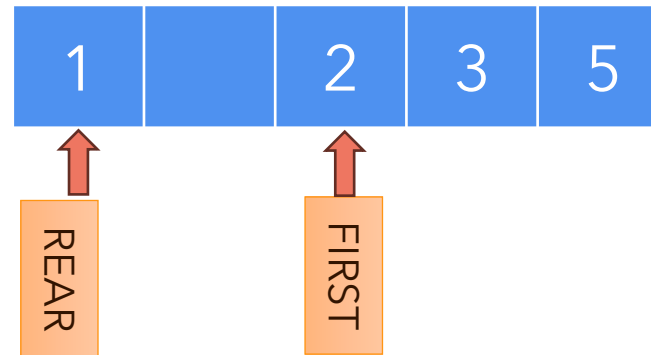
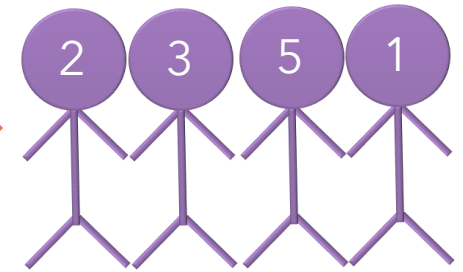
COLA - QUEUE

Ejemplo de ArrayQueue:
>> enqueue(1)

$\text{rear} = (\text{rear} + 1) \bmod \text{capacity}$
 $\text{rear} = (4 + 1) \% 5 = 5 \% 5 = 0$



FIRST



ArrayQueue

- data[]: Object
- first: int
- rear: int

- +ArrayQueue(int capacity)
- +size():int
- +isEmpty(): boolean
- +enqueue(Object e)
- +dequeue():Object e
- +first():Object e

Seudocódigo implementación ArrayQueue:

Método para agregar un elemento a la cola

```
enqueue(Object e)
//verificamos disponibilidad
if size() < data.length()
    //calculamos la posición con mod (%)
    rear = (rear + 1) % data.length
    //insertamos el dato en la cola
    data[rear] = e
```

ArrayQueue

- data[]: Object
- first: int
- rear: int

- +ArrayQueue(int capacity)
- +size():int
- +isEmpty(): boolean
- +enqueue(Object e)
- +dequeue():Object e
- +first():Object e

Seudocódigo implementación ArrayQueue:

Método para elimina un elemento a la cola

Si la cola no está vacía

1.1 Guardamos en una variable temporal el dato en la posición first

1.2 Colocamos la posición first en nulo

1.3 Calculamos la nueva posición de first como $(first + 1) \bmod capacity$

ArrayQueue

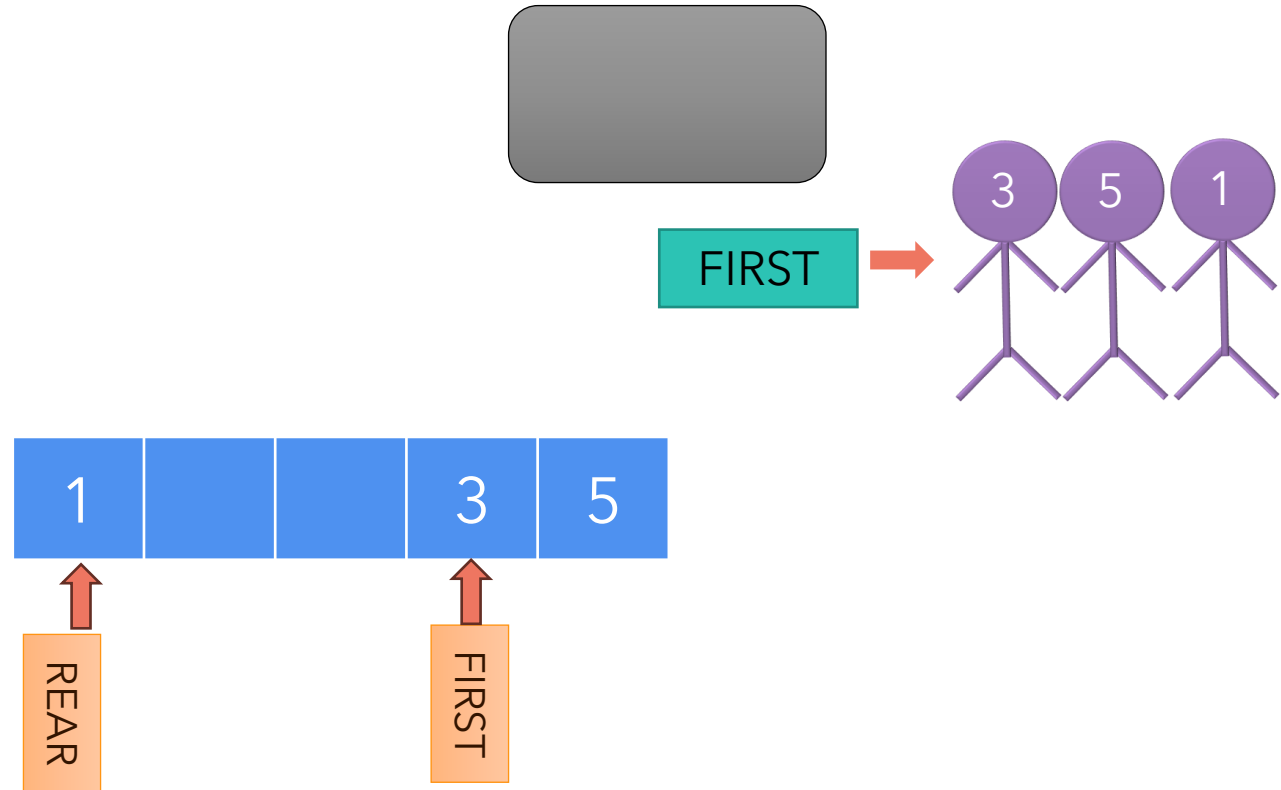
- data[]: Object
- first: int
- rear: int

- +ArrayQueue(int capacity)
- +size():int
- +isEmpty(): boolean
- +enqueue(Object e)
- +dequeue():Object e
- +first():Object e

COLA - QUEUE

Ejemplo de ArrayQueue:

ArrayQueue
<ul style="list-style-type: none">- data[]: Object- first: int- rear: int
<ul style="list-style-type: none">+ArrayQueue(int capacity)+size():int+isEmpty(): boolean+enqueue(Object e)+dequeue():Object e+first():Object e



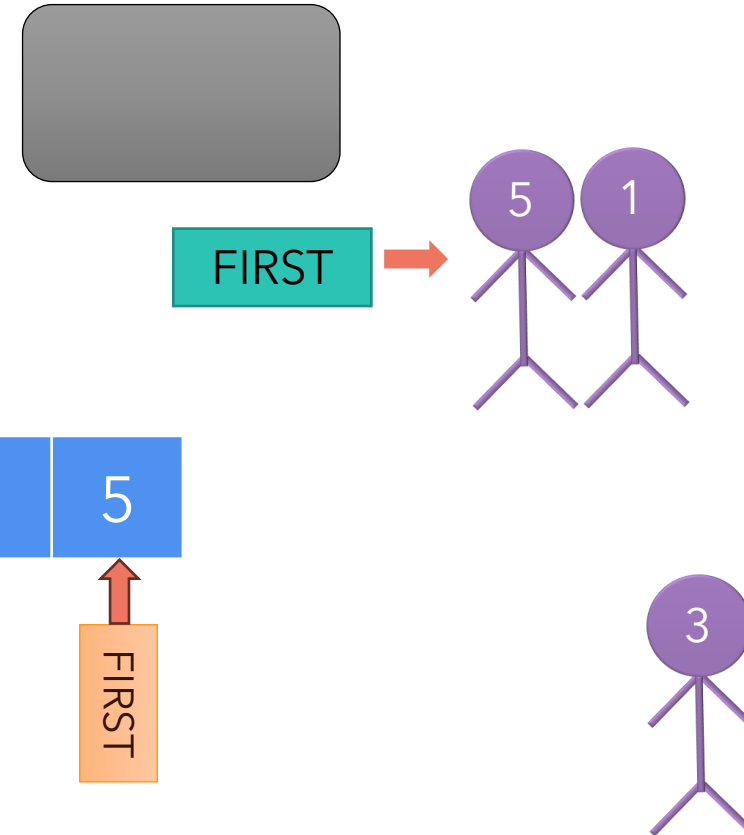
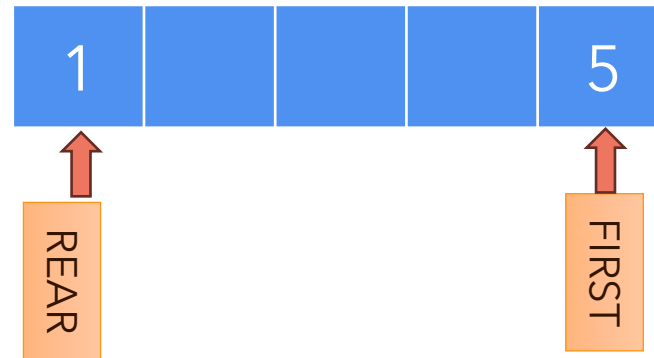
COLA - QUEUE

Ejemplo de ArrayQueue:

>> dequeue() >>3

$\text{first} = (\text{first} + 1) \bmod \text{capacity}$
 $\text{first} = (3 + 1) \% 5 = 4$

ArrayQueue
<ul style="list-style-type: none">- data[]: Object- first: int- rear: int
<ul style="list-style-type: none">+ArrayQueue(int capacity)+size():int+isEmpty(): boolean+enqueue(Object e)+dequeue():Object e+first():Object e

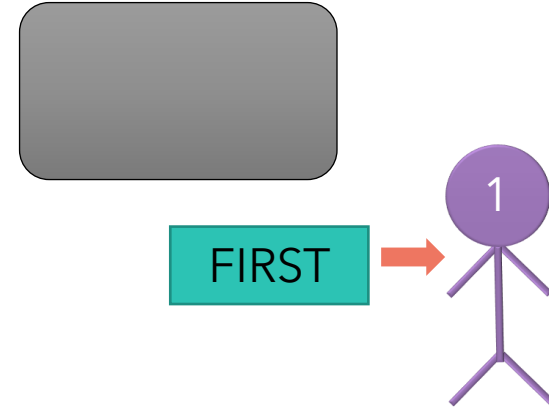
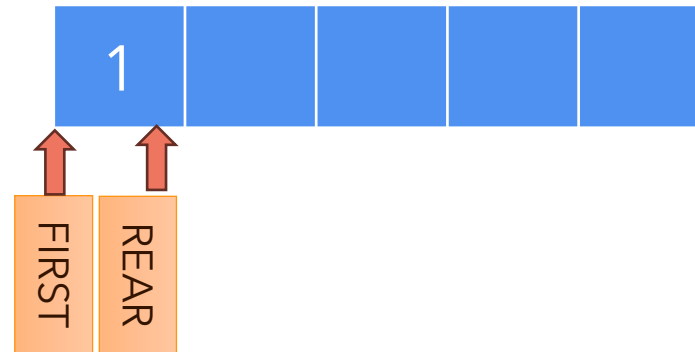


COLA - QUEUE

Ejemplo de ArrayQueue:

>> dequeue() >>5

$\text{first} = (\text{first} + 1) \bmod \text{capacity}$
 $\text{first} = (4 + 1) \% 5 = 0$



ArrayQueue
<ul style="list-style-type: none">- data[]: Object- first: int- rear: int
<ul style="list-style-type: none">+ArrayQueue(int capacity)+size():int+isEmpty(): boolean+enqueue(Object e)+dequeue():Object e+first():Object e

Seudocódigo implementación ArrayQueue:
Método para eliminar un elemento

```
dequeue()  
    if isEmpty()  
        //la cola esta vacia retornamos nulo  
        return null  
    else  
        //almacenamos el dato a retornar  
        Object temp = data[first]  
        //eliminamos el dato de la cola  
        data[first] = null  
        //calculamos la nueva posición de first  
        first = (first+1)%data.length()  
        return temp //retornamos el dato
```

ArrayQueue

- data[]: Object
- first: int
- rear: int

- +ArrayQueue(int capacity)
- +size():int
- +isEmpty(): boolean
- +enqueue(Object e)
- +dequeue():Object e
- +first():Object e

- ❑ La segunda implementación QUEUE usa la lista simple
- ❑ Esta implementación se recomienda usar cuando requerimos un QUEUE con un número no conocido de datos

Clase Queue

Mantendremos los siguientes atributos

- ❑ Una lista simple con los datos de la cola
- ❑ No se requiere el atributo first y rear, ya que estos corresponden a head y tail de la lista simple, y los podemos acceder a través de First() y Last()

Queue
- data: List

Clase Queue

Mantendremos los siguientes métodos

- ❑ Constructor vacío
- ❑ Size: retorna el número de elementos en la cola
- ❑ isEmpty: indica si la cola esta vacía
- ❑ Las operaciones enqueue(), dequeue(), y first() similar a la implementación con arreglos

Queue

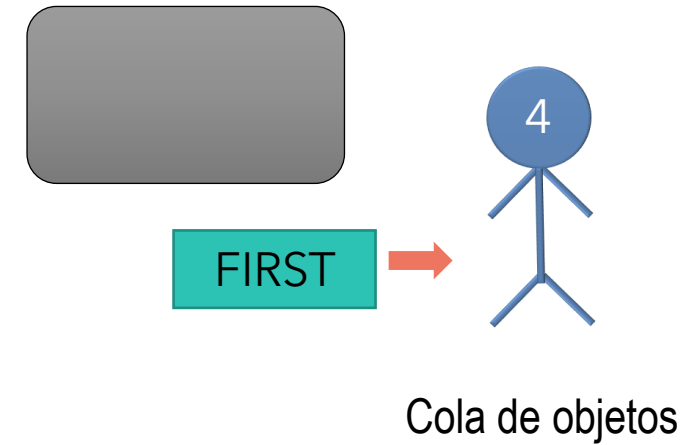
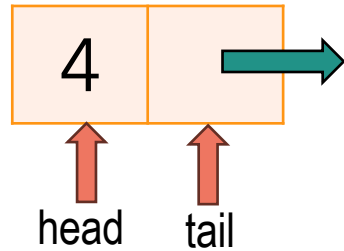
- data: List

+ Queue()
+ size(): int
+ isEmpty(): Boolean
+ enqueue(Object e)
+ dequeue(): Object
+ first(): Object

COLA - QUEUE

Ejemplo de Queue:
>> enqueue(4)

Queue
- data: List
+ Queue() + size(): int + isEmpty(): Boolean + enqueue(Object e) + dequeue(): Object + first(): Object



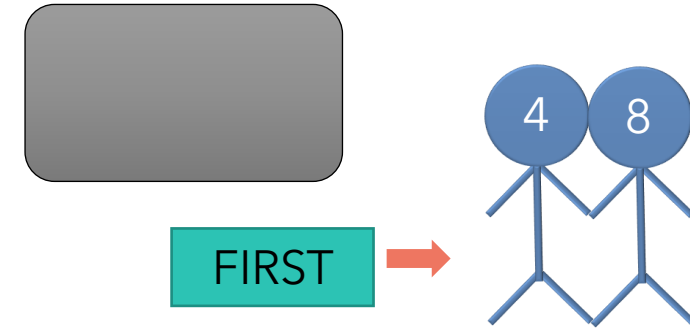
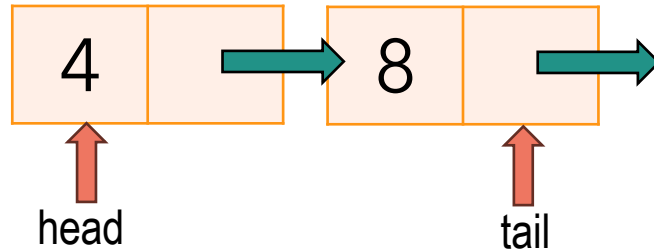
COLA - QUEUE

Ejemplo de Queue:

>> enqueue(4)

>> enqueue(8)

Queue
- data: List
+ Queue() + size(): int + isEmpty(): Boolean + enqueue(Object e) + dequeue(): Object + first(): Object



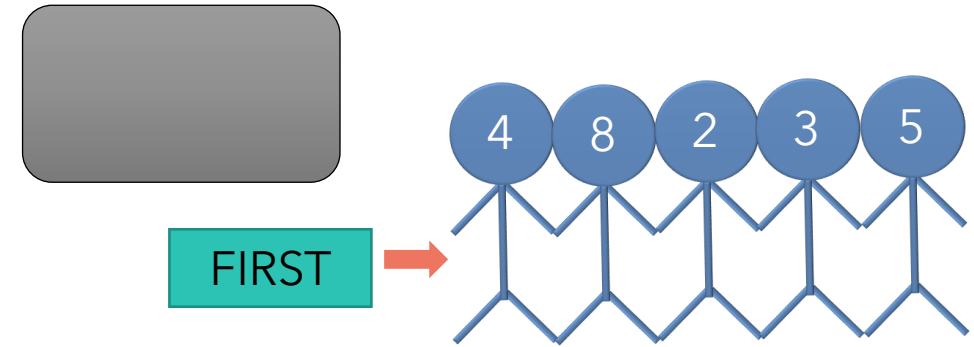
Cola de objetos

COLA - QUEUE

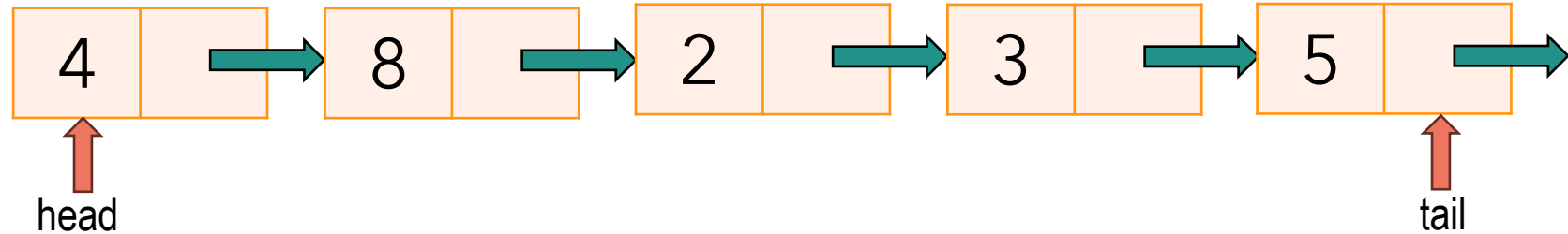
Ejemplo de Queue:

```
>> enqueue(4)
>> enqueue(8)
>> enqueue(2)
>> enqueue(3)
>> enqueue(5)
```

Queue
- data: List
+ Queue() + size(): int + isEmpty(): Boolean + enqueue(Object e) + dequeue(): Object + first(): Object



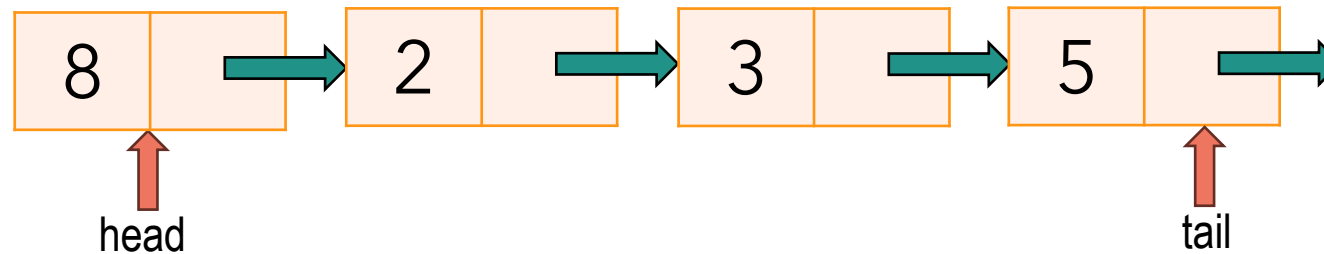
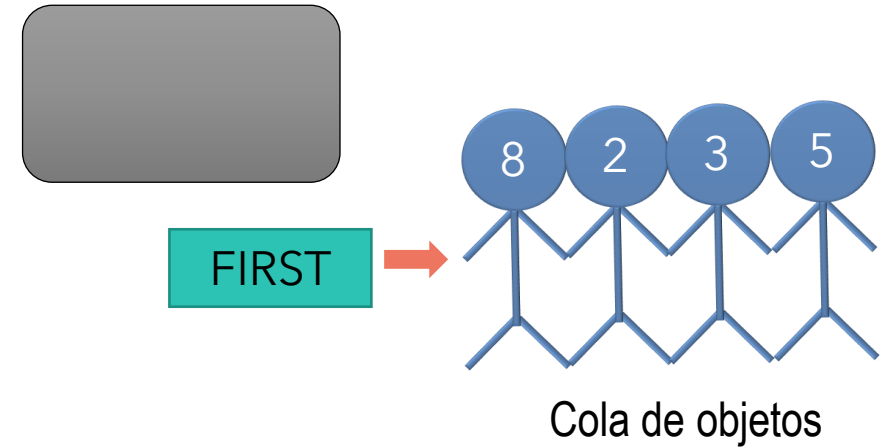
Cola de objetos



COLA - QUEUE

Ejemplo de Queue:
>> dequeue() >> 4

Queue
- data: List
+ Queue() + size(): int + isEmpty(): Boolean + enqueue(Object e) + dequeue(): Object + first(): Object



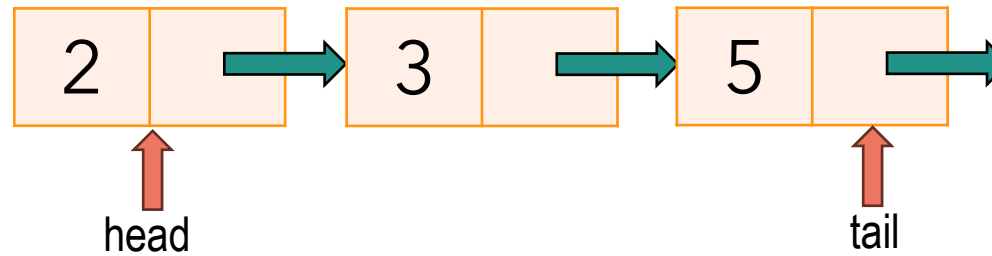
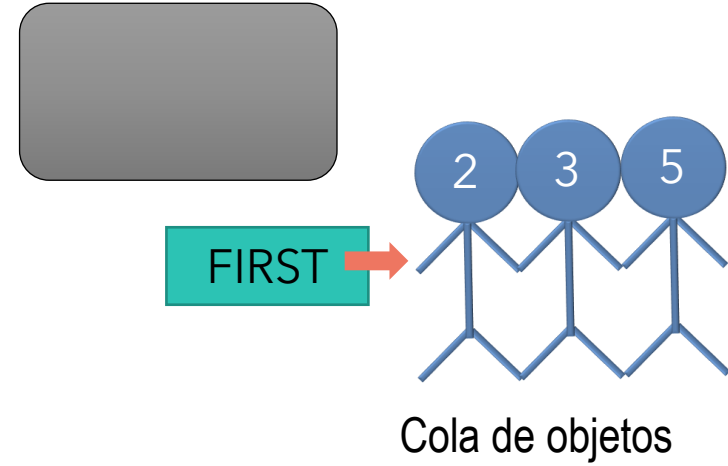
COLA - QUEUE

Ejemplo de Queue:

>> dequeue() >> 4

>> dequeue() >> 8

Queue
- data: List
+ Queue() + size(): int + isEmpty(): Boolean + enqueue(Object e) + dequeue(): Object + first(): Object



COLA - QUEUE

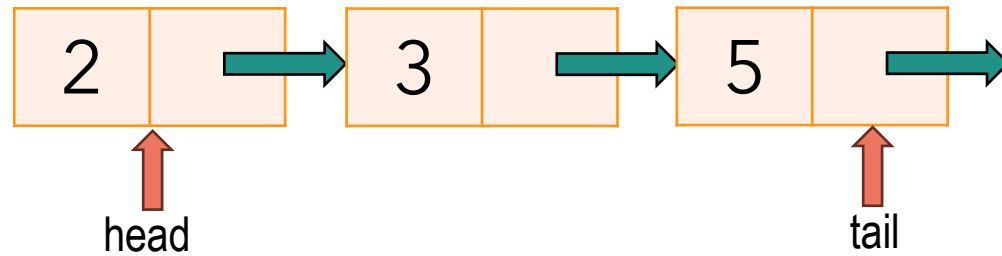
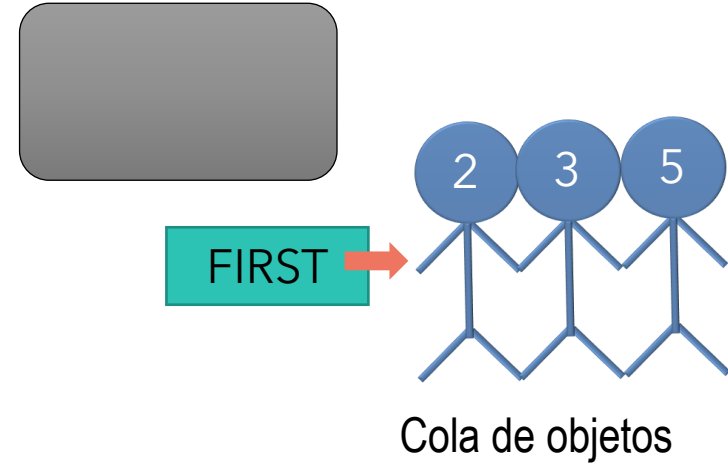
Ejemplo de Queue:

```
>> dequeue()>>4
```

```
>> dequeue()>>8
```

```
>> first()>>2
```

Queue
- data: List
+ Queue() + size(): int + isEmpty(): Boolean + enqueue(Object e) + dequeue(): Object + first(): Object



Clase Queue

```
Queue( )  
    data = new List()
```

Recuerda: la cola implementada con una lista no tiene una capacidad limitada, usa la memoria de forma dinámica

Queue

- data: List

+ Queue()
+ size(): int
+ isEmpty(): Boolean
+ enqueue(Object e)
+ dequeue(): Object
+ first(): Object

Clase Queue

```
size( )  
    return data.size()
```

```
isEmpty()  
    return size==0
```

Queue
- data: List
+ Queue() + size(): int + isEmpty(): Boolean + enqueue(Object e) + dequeue(): Object + first(): Object

Clase Queue

Usamos el método de agregar al final de la lista definidos en nuestra clase lista simple

```
enqueue(Object e )  
    data.addLast(e)
```

Usamos el método removeFirst() de nuestra lista simple, que se encarga de eliminar el primer nodo de la lista y retornarlo

```
dequeue( )  
    return data.removeFirst()
```

Usamos el método First() de nuestra lista simple, que se encarga de retornar la cabecera de nuestra lista

```
first( )  
    return data.First().getData()
```

COLA - QUEUE

Queue

- data: List

+ Queue()

+ size(): int

+ isEmpty(): Boolean

+ enqueue(Object e)

+ dequeue(): Object

+ first(): Object

