



UNIVERSIDAD NACIONAL DE CAJAMARCA  
FACULTAD DE INGENIERÍA  
PROGRAMA DE INGENIERÍA DE SISTEMAS



# **ALGORITMOS Y ESTRUCTURA DE DATOS II**

## **11: TALLER DE EJERCICIOS I**

## INTRODUCCIÓN

Semana 11 - Taller de ejercicios I: bicolos, colas con prioridad y notación postfija



# ¿QUÉ OBSERVA?



**Cada vez que embarazadas o ancianos suben a un autobús  
las personas sentadas duermen.**

*Reflexiones para Tí y para Mí. com*



# RECORDEMOS



- ¿Cuál es la característica principal de una cola?
- ¿Qué operaciones encontramos en una cola?
- ¿Recuerda que significa PEPS? ¿y qué significa FIFO?
- ¿Podría mencionar ejemplos de colas que no estén relacionados con las colas típicas, por ejemplo en una computadora podría identificar una cola?
- ¿Qué sabemos de las pilas?



## ¿QUÉ PASARÍA SI ...

- ... me piden crear una cola en donde considere prioridad, es decir que puede llegar alguien que tenga mayor prioridad y debe de ser atendido antes de alguien que ya estuvo en la cola?



# LOGRO ESPERADO



- Al término de la sesión, el estudiante elabora un programa java en donde a través de un menú gestione una cola con prioridad, verificando el buen funcionamiento de cada acción de su menú de opciones.

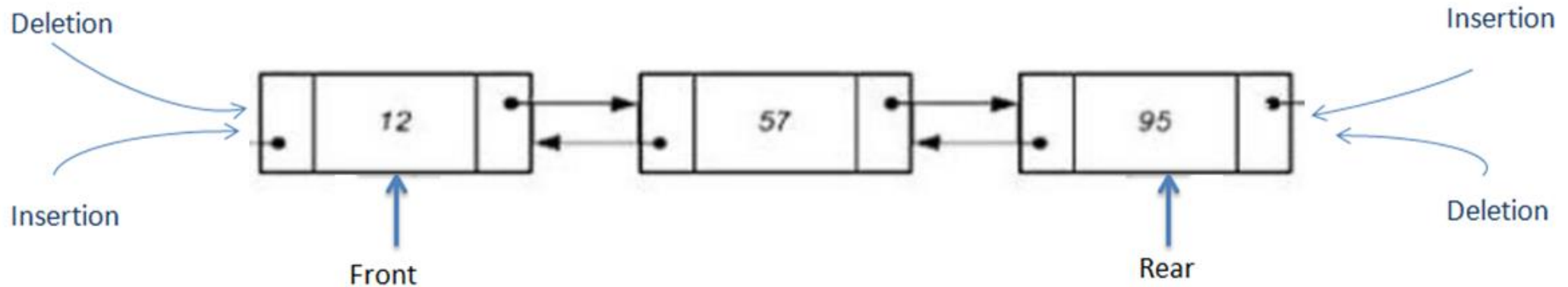
## DESARROLLO DEL TEMA

### Estructura de Datos



# BICOLAS O COLAS DOBLES














- Es un tipo de cola especial que permiten la inserción y eliminación de elementos de ambos extremos de la cola.
- Se les llama DEQUE (Double Ended QUEue)















# ESTRUCTURAS BÁSICAS PARA UNA BICOLA



c  Nodo		
f 	dato	int
f 	ant	Nodo
f 	sgt	Nodo
m 	Nodo(int, Nodo, Nodo)	
m 	Nodo(int)	
m 	getDato()	int
m 	setDato(int)	void
m 	getAnt()	Nodo
m 	setAnt(Nodo)	void
m 	getSgt()	Nodo
m 	setSgt(Nodo)	void
m 	toString()	String

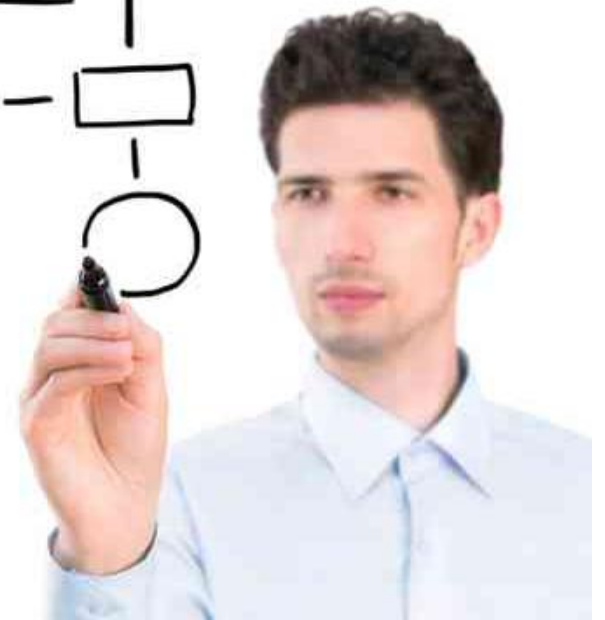
c  Bicola		
f	lisBicola	ListaDoblementeEnlazada
m	Bicola()	
m	encolaFin(int)	void
m	encolaIni(int)	void
m	desencolaIni()	int
m	desencolaFin()	int
m	estaVacia()	boolean
m	toString()	String

c  ListaDoblementeEnlazada		
f 	inicio	Nodo
f 	fin	Nodo
m 	ListaDoblementeEnlazada()	
m 	estaVacia()	boolean
m 	insertaIni(int)	void
m 	insertaFin(int)	void
m 	eliminaIni()	int
m 	eliminaFin()	int
m 	toString()	String

# BICOLAS



- Tal como indicamos una bicola se caracteriza porque se puede insertar y eliminar por ambos extremos de la cola.
- Para su construcción necesita trabajar tomando como base una lista doblemente enlazada.
- Recuerde como construir las funciones listadas en la diapositiva anterior haciendo uso diagramas



# COLAS CON PRIORIDAD



- En muchos casos de la vida real encontramos que hay colas de muchos de tipos. Actualmente en una cola se da preferencia a las personas adultas, embarazadas o personas con niños (atención preferencial), por otro lado las entidades bancarias dan preferencia a sus clientes, dejando en espera y generando molestias a los que no son clientes.



Mujeres  
Embarazadas



Personas con  
bebés en brazos,  
niños y niñas



Personas  
en condición de  
discapacidad

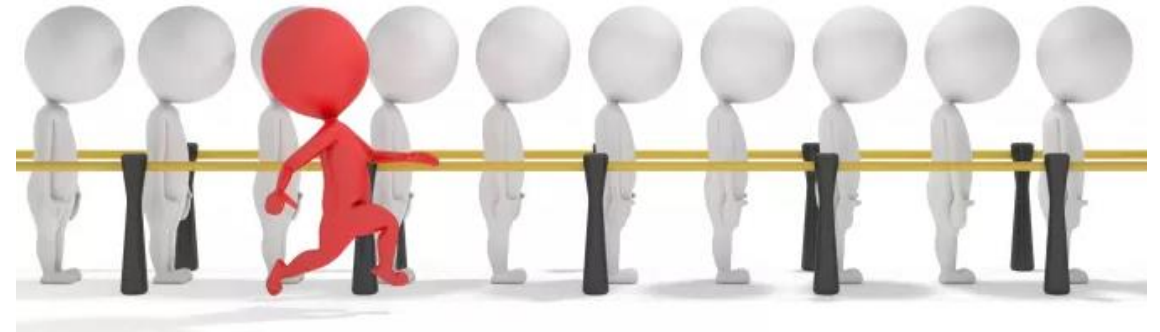


Adultos  
Mayores

# COLAS CON PRIORIDAD



- Partimos de una cola tradicional, donde se inserta al final (push) y se elimina al inicio (pop)
- La diferencia, es que no se inserta siempre en la posición inicial, la posición a insertar ahora depende de la prioridad
- Esta prioridad se registra como parte del nodo





# COLAS CON PRIORIDAD



- Partimos de una cola en la cual cada elemento tiene una prioridad, de ellas, 1 es la prioridad más alta y la que se debe atender. Cuando un cliente llega a la cola, debe colocarse de tal manera que todos los que estén delante de él tengan mayor o igual prioridad.
- Cada nodo tendrá un atributo adicional que viene a ser la prioridad. Cuando encolemos un elemento debemos buscar en qué posición encolarlo de acuerdo a la prioridad
- Habrá que considerar los siguientes casos:
  - Si la cola está vacía
  - Si tiene mayor prioridad que el primer elemento de la cola
  - De lo contrario buscar su posición

# COLAS CON PRIORIDAD



## DEFINICIONES INICIALES

```
public class Nodo<T> {  
    private T dato;  
    private Nodo<T> sgt;  
    private int prioridad;  
  
    public Nodo(T dato, Nodo<T> sgt, int prioridad) {  
        this.dato = dato;  
        this.sgt = sgt;  
        this.prioridad = prioridad;  
    }  
}
```

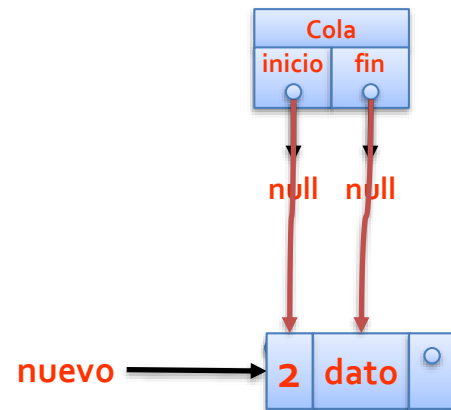
## CAMBIAR EL MÉTODO ENCOLAR

```
public class Cola<T> {  
    private Nodo<T> inicio;  
    private Nodo<T> fin;  
    public int length;  
  
    public Cola(){...}  
  
    public T desencolar(){...}  
    public void encolar(T dato){...}  
  
    public void encolaPrioridad(T dato, int prioridad){...}  
  
    private boolean estaVacia() { return fin == null; }  
  
    @Override  
    public String toString() {...}  
}
```

El método ahora debe de encolar de acuerdo a la prioridad

# ENCOLANDO CON PRIORIDAD

- Si la cola está vacía
  - Partimos con una lista vacía y el nodo a insertar
  - Hacemos de que inicio y fin referencien a nuevo



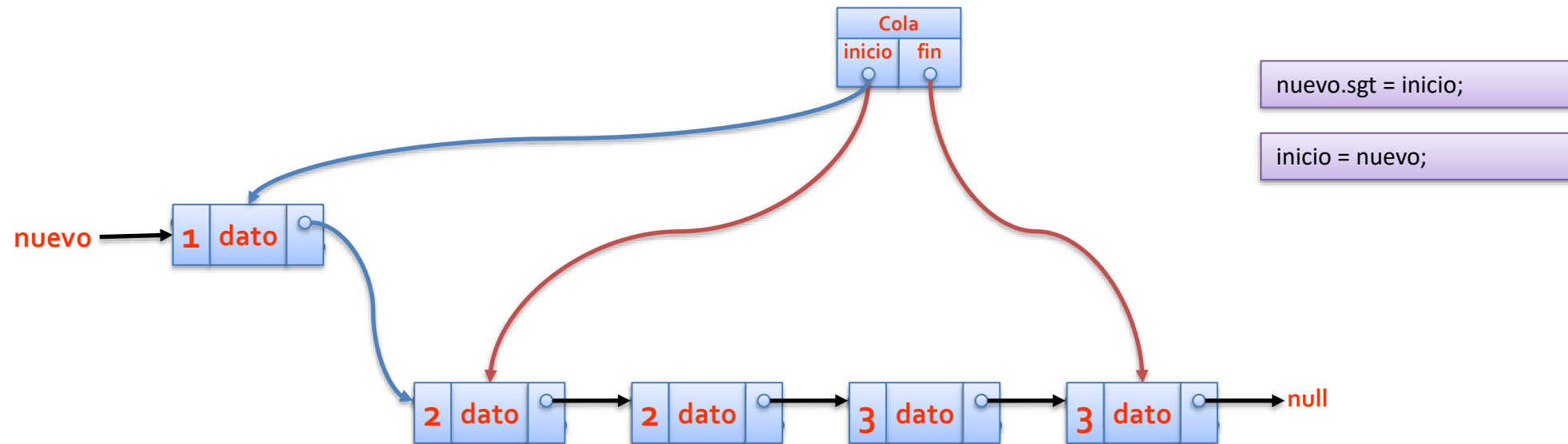
```
Nodo nuevo = new Nodo(dato, prioridad);
```

```
inicio = nuevo;
```

```
fin = nuevo;
```

# ENCOLANDO CON PRIORIDAD

- Insertando en una cola no vacía
  - Partimos con una cola con elementos y el nodo a insertar con prioridad menor al primer elemento de la cola
  - Hacemos que el siguiente de nuevo referencie al inicio de cola
  - Luego que el inicio de la cola referencie a nuevo





# ENCOLANDO CON PRIORIDAD



- Insertando en una cola no vacía
  - Partimos con una cola con elementos y el nodo a insertar nuevo. Se definen los nodos auxiliares anterior y siguiente
  - Mientras  $\text{sig} \neq \text{null}$  y la prioridad de nuevo es mayor que la de siguiente hacer anterior sea igual que siguiente y que siguiente referencie al siguiente
  - Luego  $\text{anterior.siguiente}$  referencia a nuevo y  $\text{nuevo.siguiente}$  referencia a siguiente
  - **Si** se inserta al final se actualiza último

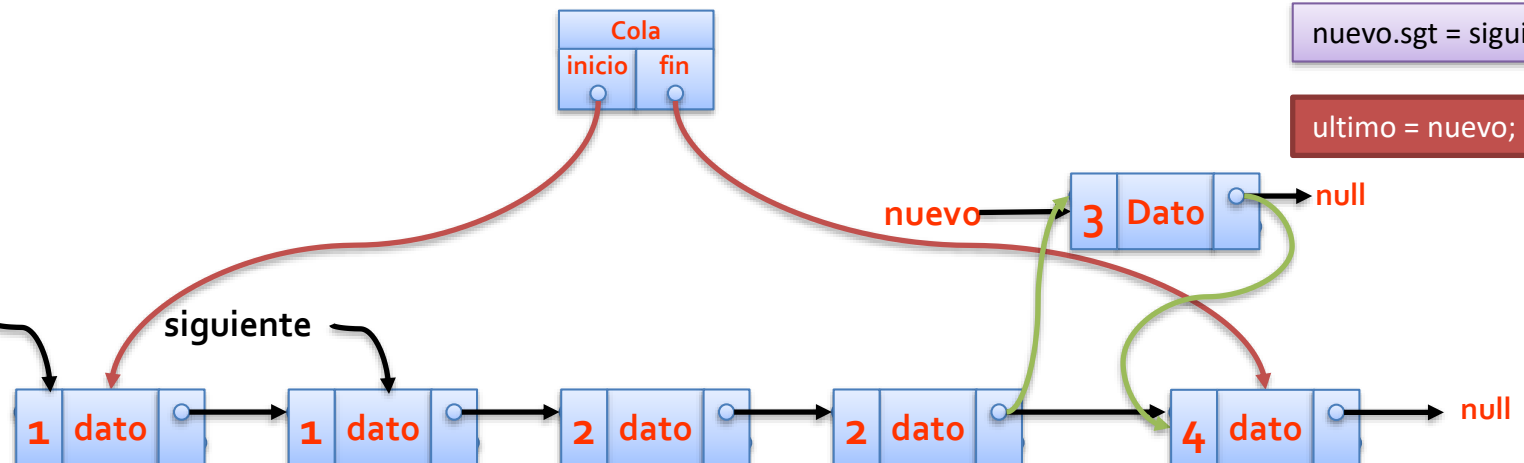
Nodo anterior = inicio

Nodo siguiente = anterior.sgt;

```
while(siguiente != null &&
      nuevo.prioridad >= siguiente.prioridad){
    anterior = siguiente;
    siguiente = siguiente.sgt;
}
```

anterior

siguiente



$\text{anterior.sgt} = \text{nuevo};$

$\text{nuevo.sgt} = \text{siguiente};$

$\text{ultimo} = \text{nuevo};$



# EVALUACIÓN DE EXPRESIONES ARITMÉTICAS CON PILAS

- Una expresión aritmética está formada por operandos y operadores. La expresión  $x*y - (a+b)$  consta de los operadores  $*$ ,  $-$ ,  $+$  y de los operandos  $x$ ,  $y$ ,  $a$ ,  $b$ .
- Los operadores, como es sabido, tienen distintos niveles de precedencia o prioridad a la hora de su evaluación.
- Existen otras formas de escribir expresiones aritméticas, que se diferencian por la ubicación del operador respecto de los operandos.

---

$a*b/(a+c)$	(infija) $\rightarrow$	$a*b/+ac$	$\rightarrow$	$*ab/+ac$	$\rightarrow$	$/*ab+ac$	(polaca)
$a*b/a+c$	(infija) $\rightarrow$	$*ab/a+c$	$\rightarrow$	$/*aba+c$	$\rightarrow$	$+/*abac$	(polaca)
$(a-b)^c+d$	(infija) $\rightarrow$	$-ab^c+d$	$\rightarrow$	$^-abc+d$	$\rightarrow$	$+^-abcd$	(polaca)

---

$a*b/(a+c)$	(infija) $\rightarrow$	$a*b/ac+$	$\rightarrow$	$ab*/ac+$	$\rightarrow$	$ab*ac+/*$	(polaca inversa)
$a*b/a+c$	(infija) $\rightarrow$	$ab*/a+c$	$\rightarrow$	$ab*a/+c$	$\rightarrow$	$ab*a/c+/*$	(polaca inversa)
$(a-b)^c+d$	(infija) $\rightarrow$	$ab-^c+d$	$\rightarrow$	$ab-c^+d$	$\rightarrow$	$ab-c^d+/*$	(polaca inversa)

---



# NOTACIÓN POLACA INVERSA O POSFIJA

- Se parte de una expresión en notación infija que tiene operandos, operadores y puede tener paréntesis.
- La transformación se realiza utilizando una pila en la que se almacenan los operadores y los paréntesis izquierdos.
- Los operandos pasan directamente a formar parte de la expresión en postfija.
- Un operador se mete en la pila si se cumple que:
  - ✓ La pila está vacía.
  - ✓ El operador tiene mayor prioridad que el operador cima de la pila.
  - ✓ El operador tiene igual prioridad que el operador cima de la pila y se trata de la máxima prioridad.
- Si la prioridad es menor o igual, se saca el elemento cima de la pila, se pone la expresión en postfija y se vuelve a hacer la comparación con el nuevo elemento cima.



# NOTACIÓN POLACA O POSFIJA

- El paréntesis izquierdo siempre se mete en la pila
- Cuando se lee un paréntesis derecho, hay que sacar todos los operadores de la pila y ponerlos en la expresión postfija, hasta llegar a un paréntesis izquierdo, el cual se elimina, ya que los paréntesis no forman parte de la expresión postfija.
- El algoritmo termina cuando no hay más ítems de la expresión origen y la pila está vacía.
- Escribir un programa java que permita transformar una expresión infija en postfija



# FCFM



---

FACULTAD DE CIENCIAS FÍSICO MATEMÁTICAS

**EVALUACIÓN DEL TEMA DESARROLLADO**

Reflexionemos!



# RECORDANDO LO APRENDIDO



- ¿Qué características tiene una bicola?
- Considera que existe una aplicación práctica para las colas de prioridad. Mencione algunos ejemplos en donde pueda aplicar las colas de prioridad.
- ¿Qué es una expresión aritmética?
  - ¿Qué otras notaciones existen?
  - ¿Tienen alguna utilidad estas notaciones?
- Transforme a prefija y postfija la siguiente expresión:  $(a+b)^2 * (c/d)^3$

## EJERCICIOS DE APLICACIÓN





# PONIENDO EN PRÁCTICA LO APRENDIDO

- Escriba un programa que permita decidir a una Caja de ahorro cuál es la mejor opción al momento de formar las colas en su oficina de la ciudad. Simule para ambas el mismo número de clientes a atender, el tiempo de pasar a la cola a la caja es de 5s y el tiempo de atención puede variar entre 30s y 180s.
- Tiene que decir entre formar una sola cola para los tres cajeros o que conforme vayan llegando formen 3 colas independientes. Vea la figura.

