# Práctica 8
# Cola en arreglo

Estructuras de datos

## 1. Meta

Que el alumno domine el manejo de información almacenada en una **Cola**.

## 2. Objetivos

Al finalizar la práctica el alumno será capaz de:

- Implementar el tipo de dato abstracto **Cola** utilizando un arrelgo.
- Diferenciar entre distintos tipos de implementación para el tipo de dato abstracto **Cola**.

## 3. Antecedentes

Una `Cola` es una estructura de datos caracterizada por:

1. Ser una estructura de tipo `FIFO`, esto es que el primer elemento que entra es el primero que sale.
2. Tener un tamaño dinámico.
3. Ser lineal.

La definición del tipo de dato abstracto `Cola` utilizada en la práctica anterior es igualmente válida para la nueva implementación. La interfaz `Cola` que se debe implementar, también es la misma. La diferencia radica en la forma en que serán almacenados los datos. Para programar una cola en un arreglo, es necesario utilizar algunos trucos:

1. Se debe tener dos enteros indicando las posiciones del primer elemento (cabeza) y último elemento en la cola.
2. Los elementos se colocan a la derecha de la cabeza, módulo la longitud del arreglo, siempre que haya espacios disponibles. Si no hay espacios, se debe cambiar el arreglo por uno más grande.
3. Los elementos se remueven de la posición de la cabeza y el inidicador de esta se recorre a la casilla siguiente, a la derecha, módulo la longitud del arreglo. Un ejemplo se muestra en la Figura `1`.

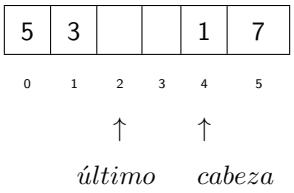


Figura 1: Cola en un arreglo, cuando ya se han eliminado elementos de la cabeza y se han formado elementos más allá de la longitud del buffer.

## 4. Desarrollo

En esta práctica se implementará la $Cola$ utilizando arreglos. De nuevo se hará una extención de la clase `ColeccionAbstracta`. Por razones didácticas, no se permite el uso de ninguna clase que se encuentre en el API de Java, por ejemplo clases como `Vector`, `ArrayList` o cualquiera que haga el manejo de arreglos dinámicos.

1. Programa el método para agregar un elemento, en forma semejante a como lo hiciste con la pila. Ojo: en este caso las dimensiones del arreglo no cambian si se llega al final, es posible que haya espacios vacíos al inicio del arreglo y deberás reutilizarlos antes que cambiar el tamaño del arreglo. Así puedes ahorrar tiempo, al no copiar a todos al nuevo arreglo. Verifica que funcione.
2. Programa el método para sacar un elemento. Deberás ir recorriendo la cabeza según sea necesario, dejando y hueco a su izquierda. Verifica que funcione.
3. Continúa con los otros métodos.

## 5. Preguntas

1. ¿Qué fórmula utilizas para detectar cuando el buffer de la cola está lleno?
2. Explica cómo funcionan los métodos `encolar` y `desencolar`.
3. Explica qué hiciste para que el arreglo en la implementación `ColaArreglo` se ajuste al tamaño de la cola.
4. ¿Cuál es la complejidad, en el peor caso, de los métodos `mira`, `encolar` y `desencolar`? Justifica.

## 6. Forma de entrega

- Asegúrense de borrar todos los archivos generados, incluyendo archivos de respaldo usando `ant clean`.
- Copien el directorio `Practica8` dentro de un directorio `<apellido1_apellido2>` donde `apellido1` es el primer apellido de un miembro del equipo y `apellido2` es el primer apellido del otro miembro. Por ejemplo: `marquez_vazquez`.
- Borren el directorio libs en la copia.
- Agreguen un archivo `reporte.pdf` dentro del directorio `Practica8` de la copia, con el nombre completo de los integrantes del equipo, las repuestas a las preguntas de la sección anterior y cualquier comentario que quieran hacer sobre la práctica.

- Compriman el directorio <apellido1_apellido2> en un archivo <apellido1_apellido2>.tar.gz. Por ejemplo: marquez_vazquez.tar.gz.
- Suban este archivo en la sección correspondiente del aula virtual. Basta una entrega por equipo.

## 7. Forma de evaluación

Para su calificación final se tomarán en cuenta los aspectos siguientes:

| | |
|---|---|
| 70 % | Calificación generada por las pruebas automáticas. Usaremos nuestros archivos, por lo que si realizan modificaciones a sus copias, éstas no tendrán efecto al momento de calificar. |
| 15 % | Documentación completa y adecuada. Entrega en el formato requerido, sin archivos .class, respaldos, bibliotecas de JUnit u otros no requeridos. |
| 15 % | Revisión manual del código, para verificar que se cumpla con las especificaciones, que no se haya copiado, etcétera. |