MANEJO DE UNA ESTRUCTURA PILA

```java
import java.util.Stack;

public class Pila {
	public static void main(String[] args) {
		Stack<Character> pila = new Stack<Character>();

		/* ¿La pila está vacía? */
		System.out.println(pila.isEmpty());	// true

		pila.push('a');				// ['a']
		pila.push('b');				// ['a', 'b']
		pila.push('c');				// ['a', 'b', 'c']

		System.out.println(pila.peek());	// c
		pila.pop();				// ['a', 'b']
		System.out.println(pila.peek());	// b

		/* ¿La pila está vacía? */
		System.out.println(pila.isEmpty());	// false
	}
}
```

MANEJO DE UNA ESTRUCTURA COLA

```java
import java.util.LinkedList;
import java.util.Queue;

public class Cola {

	public static void main(String[] args) {
		Queue<Character> cola = new LinkedList<Character>();

		/* ¿La cola está vacía? */
		System.out.println(cola.isEmpty());	// true

		cola.offer('a');			// ['a']
		cola.offer('b');			// ['a', 'b']
		cola.offer('c');			// ['a', 'b', 'c']

		System.out.println(cola.peek());	// a
		cola.poll();				// ['b', 'c']
		System.out.println(cola.peek());	// b

		/* ¿La cola está vacía? */
		System.out.println(cola.isEmpty());	// false
	}
}
```

MANEJO DE UNA ESTRUCTURA BICOLA

```
import java.util.Deque;
import java.util.LinkedList;

public class Bicola {

	public static void main(String[] args) {
		Deque<Character> bicola = new LinkedList<Character>();

		/* ¿La bicola está vacía? */
		System.out.println(bicola.isEmpty());  // true

		bicola.addFirst('a');                  // ['a']
		bicola.addFirst('b');                  // ['b', 'a']
		bicola.addLast('x');                   // ['b', 'a', 'x']
		bicola.addLast('y');                   // ['b', 'a', 'x', 'y']

		System.out.printf("%c - %c\n", bicola.getFirst(),
				bicola.getLast());         // b - y
		bicola.pollFirst();                    // ['a', 'x', 'y']
		bicola.pollLast();                     // ['a', 'x']
		System.out.printf("%c - %c\n", bicola.getFirst(),
				bicola.getLast());         // a - x

		/* ¿La bicola está vacía? */
		System.out.println(bicola.isEmpty());  // false
	}
}
```

MANEJO DE UNA ESTRUCTURA COLA DE PRIORIDAD

```
class Clientes implements Comparable<Clientes> {
	private int id;
	private String nombre;
	public Clientes(int i, String n) {
		this.id = i;
		this.nombre = n;
	}
	@Override
	public String toString() {
		return "Clientes [id=" + id + ", nombre=" + nombre + "]";
	}

	@Override
	public int compareTo(Clientes o) {
		// return nombre.compareTo(o.nombre);
		return id - o.id;
	}
}

import java.util.PriorityQueue;
public class ColaDePrioridad {
	public static void main(String[] args) {
		PriorityQueue<Integer> cp = new PriorityQueue<Integer>();

		/* ¿La cola de prioridad está vacía? */
		System.out.println(cp.isEmpty());	// true

		cp.offer(100);				// [100]
		cp.offer(10);				// [100, 10]
		cp.offer(50);				// [100, 50, 10]

		System.out.println(cp.peek());		// 10
		cp.poll();				// [100, 50]
		System.out.println(cp.peek());		// 50

		/* ¿La cola de prioridad está vacía? */
		System.out.println(cp.isEmpty());	// false

		PriorityQueue<Clientes> clientes = new PriorityQueue<Clientes>();

		/* ¿La cola de prioridad está vacía? */
		System.out.println(clientes.isEmpty());	// true

		clientes.offer(new Clientes(100, "a"));	// [<100, a>]
		clientes.offer(new Clientes(10, "b"));	// [<100, a>, <10, b>]
		clientes.offer(new Clientes(50, "c"));	// [<100, a>, <50, c>, <10, b>]

		System.out.println(clientes.peek());	// Clientes [id=10, nombre=b]
		clientes.poll();			// [<100, a>, <50, c>]
		System.out.println(clientes.peek());	// Clientes [id=50, nombre=c]

		/* ¿La cola de prioridad esta vacía? */
		System.out.println(clientes.isEmpty());	// false

	}
}
```