Estructuras: Pilas y Colas

OBJETIVOS:

- Trabajar con la estructura de datos Pila.
- Trabajar con la estructura de datos Cola.

a) Cite ejemplos de la vida real que puedan ser modelados con una Cola.
b) Cite ejemplos de la vida real que puedan ser modelados con una Pila.
c) Asegúrese de comprender las diferencias y utilidades de una y otra estructura.

Ejercicio 1

Implemente las siguientes funciones básicas para la estructura Pila (Stack) genérica.

a) Pila ();
Pos: Constructor. Crea el stack vacío.

b) boolean estaVacia();
Pos: Retorna true si el stack es vacío.

c) Void apilar (T dato);
Pos: Inserta el dato en el stack.

d) void desapilar();
Pre: El stack no es vacío.
Pos: Elimina el elemento ubicado en el tope del stack.

e) T top ();
Pre: El stack no es vacío.
Pos: Retorna el elemento ubicado en el tope del stack.

f) void vaciar ();
Pos: Elimina todos los elementos del stack.

g) int cantidadNodos()
Pos: Indica la cantidad de nodos que tiene el stack.

Ejercicio 2

Implemente las siguientes funciones básicas para la estructura Cola (Queue) genérica.

a) Cola ();
Pos: Constructor. Crea la cola vacía.

b) void encolar (T dato);
Pos: Inserta el elemento en la cola.

c) void desencolar ();
Pre: La cola no es vacía.
Pos: Elimina el elemento ubicado al frente de la cola.

d) T front ();
Pre: La cola no es vacía.
Pos: Retorna el elemento ubicado en el frente de la cola.

e) boolean isEmpty ();
Pos: Retorna true si la cola es vacía.

h) void vaciar ();
Pos: Elimina todos los elementos de la cola.

i) int cantidadNodos()
Pos: Indica la cantidad de nodos que tiene la cola.

Ejercicio resumen
Implementar las siguientes interfaces

```
public interface IPila<T> {
      public boolean estaVacia();
      public void apilar (T dato);
      public void desapilar();
      public T top ();
      public void vaciar ();
      public int cantidadNodos();
}
```

```
public interface ICola<T> {

      public void encolar (T dato);
      public void desencolar ();
      public T front ();
      public boolean isEmpty ();
      public void vaciar ();
      public int cantidadNodos();

}
```

Ejercicio 3

Construya una función similar a la que utiliza el compilador para verificar si el uso de paréntesis es correcto. Los posibles paréntesis que podrá encontrar en el código son los siguientes: '(', ')', '{', '}', '[', ']'. La función debe retornar true si el balanceo de paréntesis es correcto y false en caso contrario.

Ejercicio 4

Modele el sistema de reservas de asientos en un vuelo. Suponga que los asientos están numerados del 1 al N. Si un pasajero desea adquirir un asiento en el avión y hay lugar, se le asigna un asiento. En caso de que no haya más lugar, el pasajero deberá quedar en espera a que algún asiento asignado sea devuelto.

La rutina implementada deberá tomar una acción en función del evento capturado por el sistema. Suponga que cuenta con el siguiente esqueleto para la rutina:

```
CAPTURAR EVENTO
        CASE EVENTO = NUEVO PASAJERO QUIERE ADQUIRIR PASAJE
                // Código para adquirir pasaje.
        CASE EVENTO =  PASAJERO CONFIRMADO DEVUELVE SU ASIENTO (CANCELACIÓN)
                // Código para cancelar pasaje.
END CASE
```

Nota: Cuando un pasajero obtiene un asiento en el avión no es necesario registrar nada, sólo se debe actualizar la cantidad de asientos disponibles.