CLASE 5

Un Tipo Abstracto de Datos (TAD) es un modelo que define un tipo a través de su **comportamiento** (qué hace) y no de su **estructura interna** (cómo lo hace).

Características Principales de un TAD

- **Se define por una interfaz**: Un conjunto de operaciones que determinan cómo se puede interactuar con los datos. Esta interfaz es un "contrato" que especifica el nombre, tipo y comportamiento de las operaciones.
- **Ocultamiento de la información**: La representación interna de los datos y la implementación de las operaciones están ocultas para el usuario. Solo se puede acceder a los datos a través de la interfaz provista.
- **Implementaciones intercambiables**: Un mismo TAD puede tener múltiples implementaciones. Mientras la interfaz no cambie, se puede sustituir una implementación por otra sin que el código del usuario se vea afectado.

¿Por qué usar TADs en lugar de Tipos Algebraicos?

Los tipos algebraicos presentan limitaciones que los TADs solucionan:

- El código que usa tipos algebraicos depende fuertemente de su estructura interna. Un cambio en la estructura (por ejemplo, en un constructor) obliga a modificar todo el código que lo utiliza.
- No permiten restringir la creación de datos inválidos, ya que los constructores son de acceso libre.

Los Tres Roles en el Ciclo de Vida de un TAD

Para entender un TAD, se distinguen tres roles fundamentales:

1. **Diseñador**: Define la interfaz del TAD, pensando en todos los usos que se le darán.
2. **Implementador**: Elige la representación interna, codifica las operaciones de la interfaz y se asegura de que se cumplan las propiedades del tipo.
3. **Usuario**: Utiliza el TAD a través de su interfaz, sin necesidad de conocer los detalles de la implementación.

Concepto Clave: Invariante de Representación

- Es una propiedad o conjunto de reglas que la representación interna de un TAD debe cumplir **siempre**.
- El implementador es responsable de definirlo y garantizar que todas las operaciones lo mantengan. Por ejemplo, garantizar que la edad en el TAD Persona sea siempre >= 0 o que el máximo almacenado en el TAD Termómetro sea realmente el máximo de la lista de valores.
- Gracias al invariante y al ocultamiento de la información, el usuario tiene la certeza de que los datos son siempre válidos.

Análisis de Eficiencia

La eficiencia mide cómo se comporta una función en el **peor caso posible** y en función de la cantidad de datos de la estructura (denotado como "n"). Se utiliza la notación **Big O** para clasificarla.

- **Costo Constante - O(1)**: El tiempo de ejecución es siempre el mismo, sin importar el tamaño de los datos. Ejemplos: head o tail de una lista.
- **Costo Lineal - O(n)**: El tiempo de ejecución crece de forma proporcional al tamaño de los datos. Por cada elemento, se realiza una operación de costo constante. Ejemplo: length o maximum de una lista.
- **Costo Cuadrático - O(n2)**: Por cada elemento de la estructura, se realiza una operación de costo lineal. El tiempo de ejecución crece de forma exponencial.

TADs Clásicos

Existen estructuras de datos clásicas que se modelan como TADs:

1. **Stack (Pila)**: Su propiedad es **LIFO (Last In, First Out)**. El último elemento que se ingresa es el primero que se puede sacar.
2. **Queue (Cola)**: Su propiedad es **FIFO (First In, First Out)**. El primer elemento que se ingresa es el primero que sale.
3. **Set (Conjunto)**: Su propiedad principal es que **no contiene elementos repetidos**.

Criterio para Elegir una Implementación

La elección de una implementación sobre otra depende del **contexto de uso** y de la **frecuencia** con que se utilizará cada operación. Generalmente, existe un balance: mejorar la eficiencia de una operación puede empeorar la de otra. Por ejemplo, una implementación puede hacer que maxT sea O(1) pero quitarUltimaT sea O(n), mientras que otra podría tener los costos invertidos.