Introducción a Estructuras de Datos y Java

Torres Guerra Christian Axel

17/Mayo/2025

—

Estructura de Datos

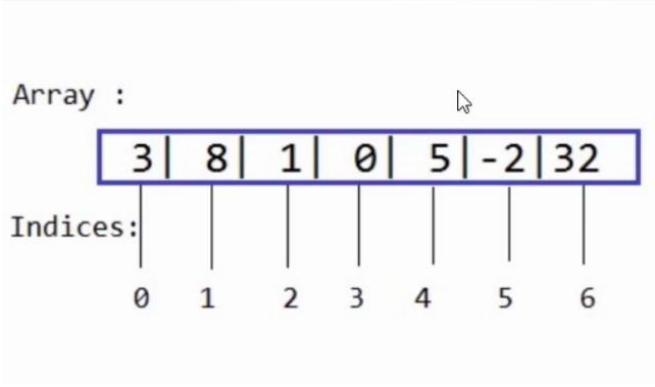
—

Flor Denisse Arenas Cortes

¿Qué es una Estructura de Datos?

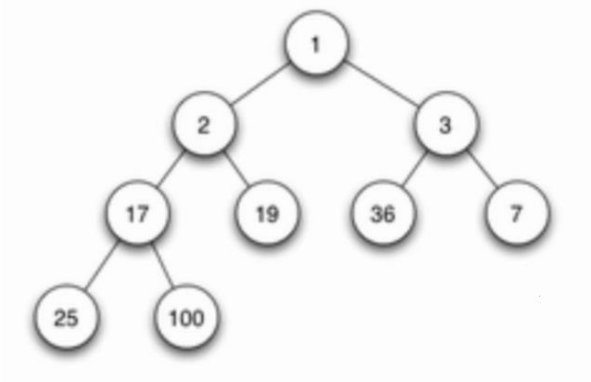
Las estructuras de datos son una forma de organizar los datos en la computadora, de tal manera que nos permita realizar unas operaciones con ellas de forma **muy eficiente**.
Es decir, igual que un array introducimos un dato y eso es prácticamente inmediato, no siempre lo es, según qué estructuras de datos y qué operaciones.
Depende que algoritmo queramos ejecutar, habrá veces que sea mejor utilizar una estructura de datos u otra estructura que nos permita más velocidad.

- **Arrays**



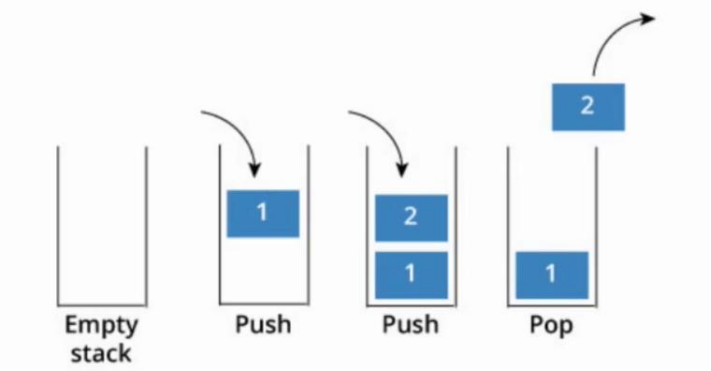
Constan de un **índice** para acceder a una posición concreta y del valor que el mismo almacena.

- **Montículos binarios**



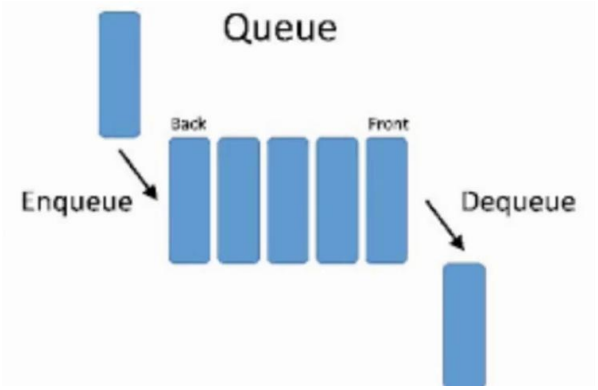
Es una forma de guardar los datos de tal manera, que, aunque no estén ordenados, se puedan retirar de ese conjunto datos **de forma ordenada**.
Esto permite una **gran velocidad**, por ejemplo, a la hora de implementar una cola de prioridades donde queremos que cada elemento que insertemos, si insertamos de repente muchos elementos con una prioridad, el primero que se coja sea el que tenga más o menos prioridad, depende del **tipo de montículo**.

- **Pilas**



Sirven, por ejemplo, para implementar el **proceso de deshacer**, como cuando escribimos en un editor de texto y pulsamos CTRL+Z, lo que podemos implementar con una pila.
Como vemos en la imagen, tenemos una pila vacía, el bloque 1 sería equivalente a escribir algo. El bloque 2 sería el equivalente a borrar una letra, por ejemplo.
Cuando utilizamos deshacer, lo que haría sería coger la última acción realizada, que tendría una función que haría ciertas operaciones con el hecho de haber borrado una letra y la volvería a poner.

- **Colas**

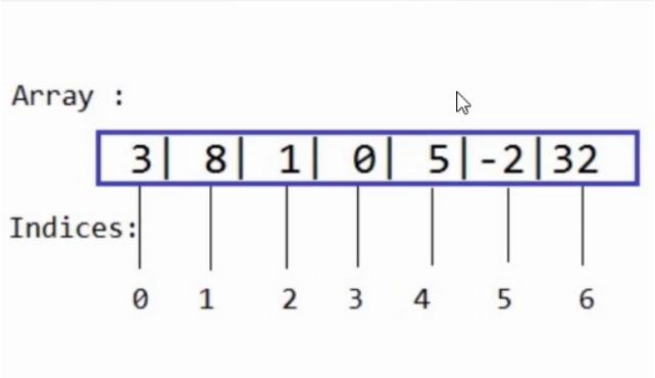


Es otra estructura de datos muy útil, que sirve, entre otras cosas, para implementar una cola o para comunicar procesos asíncronos.

Tipos abstractos de datos (TAD)

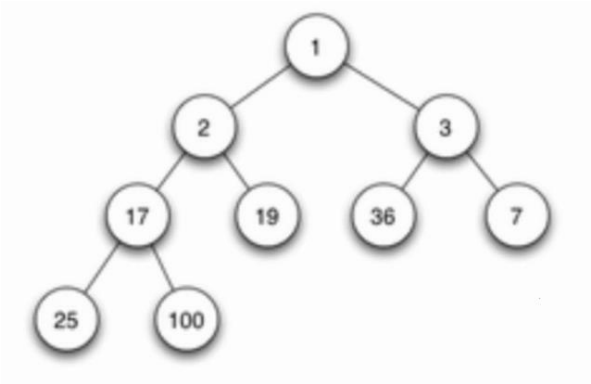
Abstracción. Consiste en ignorar los detalles de la manera particular en que está hecha una cosa, quedándonos solamente con su visión general. -Un tipo de dato abstracto (TDA) o Tipo abstracto de datos (TAD) es un modelo matemático compuesto por una colección de operaciones definidas sobre un conjunto de datos para el modelo y un nombre que lo identifica. Se define como un conjunto de valores que pueden tomar los datos de ese tipo, junto a las operaciones que los manipulan.

- **Arrays**



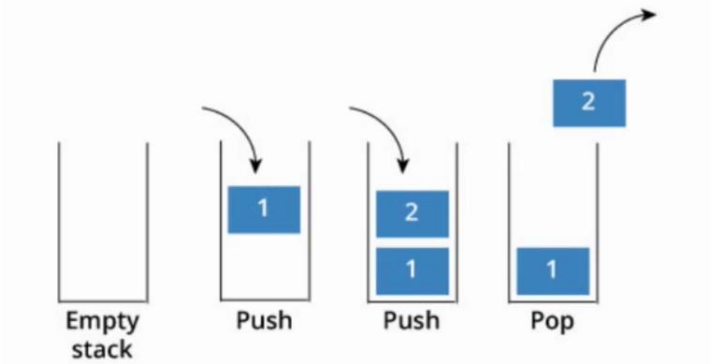
Constan de un **índice** para acceder a una posición concreta y del valor que el mismo almacena.

- **Montículos binarios**



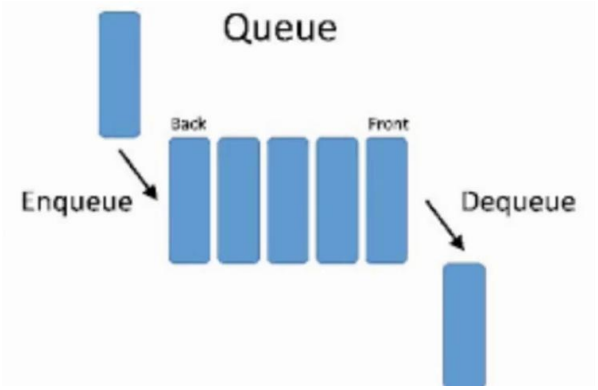
Es una forma de guardar los datos de tal manera, que, aunque no estén ordenados, se puedan retirar de ese conjunto datos **de forma ordenada**.
Esto permite una **gran velocidad**, por ejemplo, a la hora de implementar una cola de prioridades donde queremos que cada elemento que insertemos, si insertamos de repente muchos elementos con una prioridad, el primero que se coja sea el que tenga más o menos prioridad, depende del **tipo de montículo**.

- **Pilas**



Sirven, por ejemplo, para implementar el **proceso de deshacer**, como cuando escribimos en un editor de texto y pulsamos CTRL+Z, lo que podemos implementar con una pila.
Como vemos en la imagen, tenemos una pila vacía, el bloque 1 sería equivalente a escribir algo. El bloque 2 sería el equivalente a borrar una letra, por ejemplo.
Cuando utilizamos deshacer, lo que haría sería coger la última acción realizada, que tendría una función que haría ciertas operaciones con el hecho de haber borrado una letra y la volvería a poner.

- **Colas**

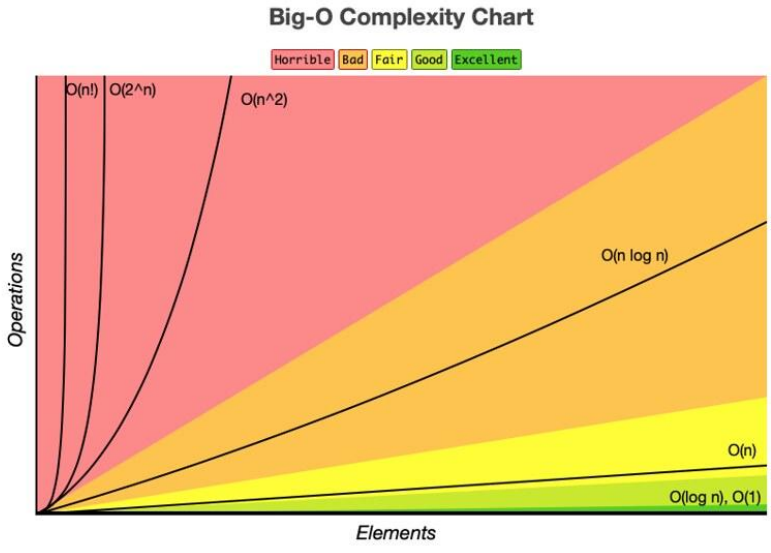


Es otra estructura de datos muy útil, que sirve, entre otras cosas, para implementar una cola o para comunicar procesos asíncronos.

Complejidad Algorítmica (Big-O)

La complejidad de un algoritmo es una medida de cuán eficiente es el algoritmo para resolver el problema. En otras palabras, es una medida de cuánto tiempo y espacio (memoria) requiere el algoritmo para producir una solución.
En matemáticas, la complejidad se estudia a menudo en el contexto de los algoritmos. Esto se debe a que muchos problemas matemáticos se pueden resolver mediante algoritmos, y la eficiencia de estos algoritmos es un factor importante en su utilidad en la práctica.

Otra forma de medir la complejidad es contar la cantidad de memoria (*en bytes o bits*) que requiere el algoritmo. Esto se conoce como la complejidad espacial del algoritmo.
Las complejidades de tiempo y espacio de un algoritmo se pueden expresar utilizando la notación O grande. Esta notación proporciona una forma de comparar la complejidad de diferentes algoritmos. Por ejemplo, un algoritmo con una complejidad temporal de O(n) se considera más eficiente que un algoritmo con una complejidad temporal de O(n^2), porque crece a un ritmo más lento a medida que aumenta el tamaño de entrada.



- **Clases**. -

En Java, las clases son una parte fundamental de la POO y son la base para la creación de objetos, que son instancias de esas clases. Una clase en Java es un plano o un modelo que define la estructura y el comportamiento de los objetos que se pueden crear a partir de ella.

Ejemplo. –

Imagina que estás desarrollando un videojuego de carreras de coches. En este juego, tienes diferentes tipos de coches, como deportivos, todoterrenos y coches de rally. Cada coche tiene ciertas características y habilidades únicas. En Java, puedes utilizar clases para representar estos coches.

Una clase en Java sería como el plano para crear un coche en tu juego. La clase define qué características tiene el coche (como su modelo, velocidad máxima, potencia del motor, etc.) y qué puede hacer (como acelerar, frenar, girar, etc.).

Conceptos fundamentales de las clases

- **Atributos**: Las clases pueden tener atributos, también conocidos como variables de instancia, que representan las características o propiedades del objeto. En el ejemplo del coche los atributos serian el modelo, la velocidad máxima, etc.

- **Métodos**: Las clases contienen métodos que representan el comportamiento del objeto. Los métodos son funciones que pueden realizar acciones y manipular los atributos de la clase. En nuestro caso sería acelerar frenar, girar, etc.

- **Constructores**: Las clases pueden tener constructores, que son métodos especiales utilizados para inicializar objetos cuando se crean. Los constructores tienen el mismo nombre que la clase y pueden tener diferentes parámetros para configurar el estado inicial del objeto.

- **Encapsulación**: La encapsulación es un principio de la POO que se refiere a la ocultación de los detalles de implementación de una clase y la exposición controlada de sus atributos y métodos. En Java, puedes utilizar modificadores de acceso como private, protected, public, etc., para controlar el acceso a los miembros de una clase.

- **Herencia**: Las clases pueden heredar atributos y métodos de otras clases. La herencia permite la creación de jerarquías de clases, donde una clase (subclase o clase derivada) puede extender o heredar características de otra clase (superclase o clase base).

- **Instanciación**: Para utilizar una clase, primero debes crear una instancia u objeto de esa clase. Puedes hacerlo utilizando el operador new, seguido del nombre de la clase y los argumentos necesarios para el constructor si lo tiene.

- **Objetos**. –

En Java, un objeto es una instancia específica de una clase que encapsula datos y comportamientos relacionados. Los objetos se crean a partir de clases, que como hemos comentado definen la estructura y el comportamiento de dichos objetos. Cada objeto tiene atributos que almacenan datos y métodos que definen las operaciones que puede realizar.

Ejemplo. –

En el contexto del videojuego de carreras de coches, los objetos son las representaciones concretas de los coches que existen en el juego. Cada objeto coche corresponde a un tipo específico de coche, como un deportivo, un todoterreno o un coche de rally.

Conceptos fundamentales de los objetos

Los objetos tienen atributos y métodos accesibles, para acceder a los métodos y atributos de un objeto, utilizamos el nombre del objeto seguido de un punto (.) y el nombre del método o atributo.

```
MiClase miObjeto = new MiClase(); // Crear un objeto
miObjeto.miMetodo(); // Acceder a un método
int valor = miObjeto.miAtributo; // Acceder a un atributo
```

Los atributos son variables que almacenan datos en una clase u objeto. Pueden ser públicos, privados o protegidos, y se pueden acceder utilizando métodos públicos conocidos como getters y setters. Para acceder a un atributo, puedes utilizar estos métodos o, si el atributo es público, acceder directamente al atributo.

- **Listas**. –

Las listas son un tipo de colección que hereda de la interface collection, son una estructura de datos que respeta el orden en el cual fueron agregados los elementos, también permiten registros repetidos.

Esta interface o los objetos de esta, representan una colección ordenada de elementos, en la cual se tiene un control absoluto y preciso del lugar en el que se quiere insertar.

Por medio de la Lista nosotros tenemos la posibilidad de decir en que posición (al inicio, al final o cualquier otro lugar) puede ser insertado o eliminado un elemento.

También es posible acceder a sus elementos a través de su índice; el cual representa la posición del elemento en la lista y así se podría buscar un elemento en dicha lista.

La interface List se encuentra en el paquete java.util, algunas de las clases que implementa esta interface son ArrayList, LinkedList, Vector.