

AWAKELAB

#programmingbootcamp

ARREGLOS Y COLECCIONES EN JAVA

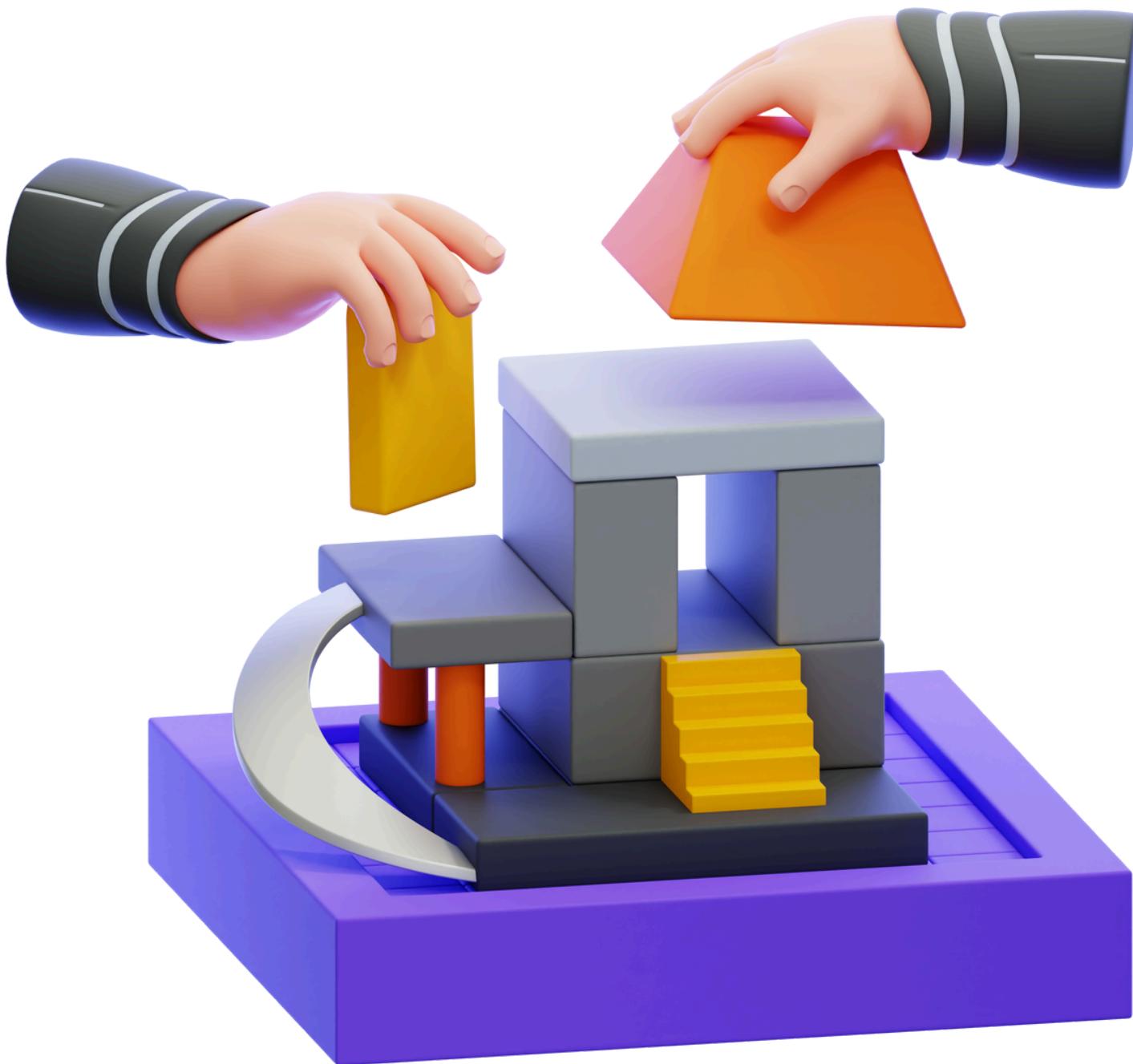




ARREGLOS Y COLECCIONES EN JAVA

¿QUÉ VAMOS A VER?

- Arreglos.
- Matrices.
- Colecciones.





ARREGLOS

- Tal y como aprendimos en los fundamentos de la programación. Los arreglos o arrays son un conjunto de N datos almacenados y que corresponden al mismo tipo.
- En Java es necesario declarar el tipo de arreglo y su nombre, para luego inicializarlo con la palabra reservada new, tal y como aprendimos con Scanner.
- También existe la posibilidad de simplificar la sintaxis, declarando e inicializando el array en una misma línea de código.



```
int edad[];  
edad = new int[10];
```

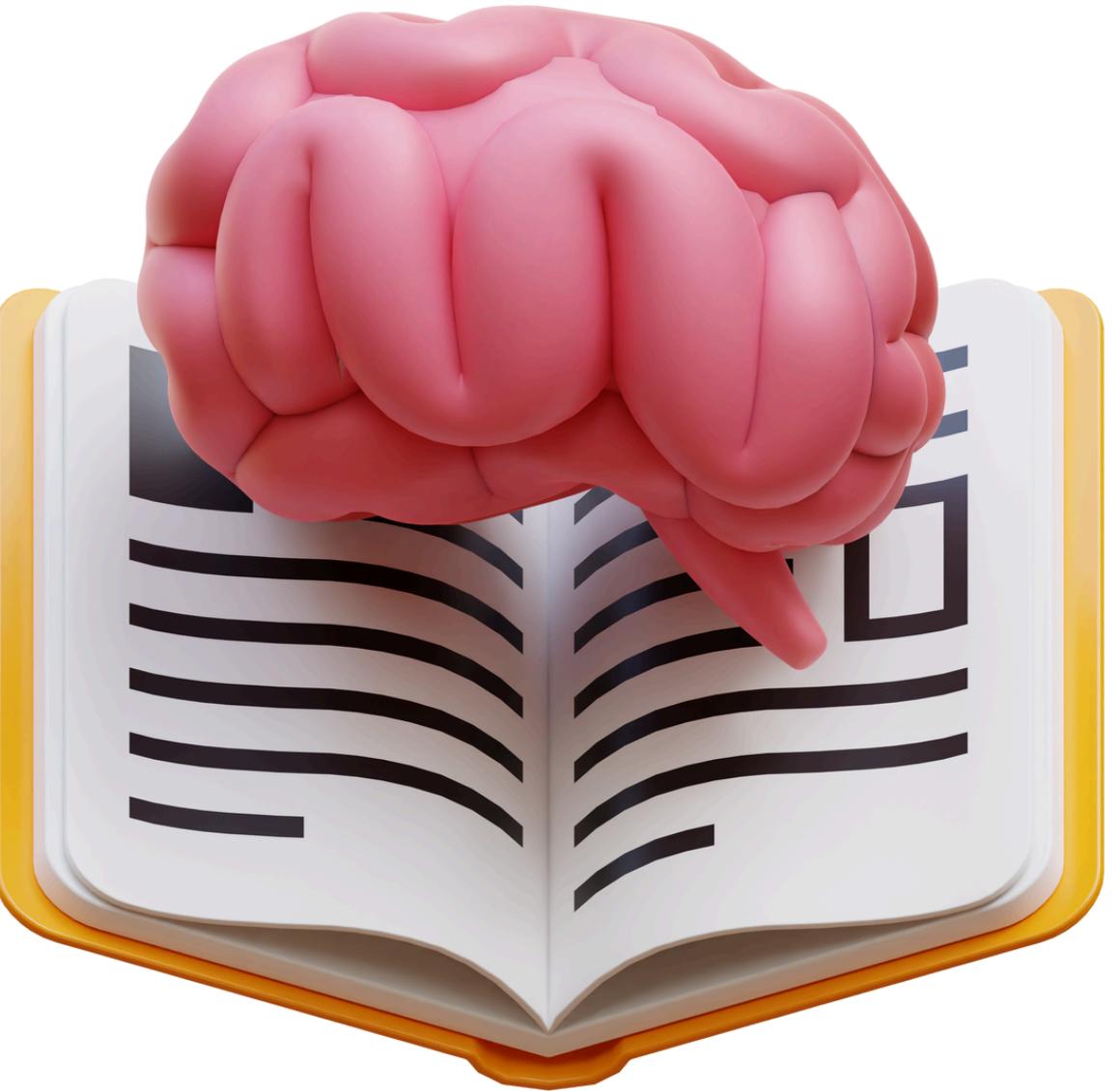
```
int edad[] = new int[10];
```



ARREGLOS

```
int [ ] donacionesPorMes = new int [12];
```

- Analicemos el array creado con este otro tipo de sintaxis.
- ¿Sus índices comienzan en 0 y terminan en 12?
- ¿Tiene en total 11 elementos?
- ¿Podemos acceder al último elemento por medio de donacionesPorMes[12]?





EJERCICIO

- Desarrolle un programa que muestre la declaración, creación, inicialización y consulta de un arreglo llamado edad.
- El arreglo debe tener 3 posiciones de tipo entero.
- El ingreso de datos puede ser manual o con números al azar.





EJERCICIO

- Desarrolle un programa que cree e inicialice un arreglo de 5 posiciones de tipo String.
- Debe contener por defecto cinco nombres.
- ¿Recuerdas For-Each? Este es buen momento de ponerlo en práctica para recorrer y mostrar los elementos del array.





MATRICES

- Corresponde a un conjunto de datos en dos dimensiones $N \times M$. En dónde N corresponde a las filas y M a las columnas.
- La sintaxis de Java exige declarar el tipo de dato que se podrá almacenar dentro de la matriz.

```
int[][] matriz = new int[3][3];
```





COLECCIONES

- Ya sabemos que los arreglos son perfectos para almacenar una cantidad limitada de datos, pero no son dinámicos.
- Ese problema de ineficiencia que se genera con los arreglos, lo vienen a solucionar diferentes estructuras dinámicas que se denominan colecciones.
- Es decir, nos permiten aumentar o disminuir la cantidad de datos según la necesidad que tengamos.





COLECCIONES

LISTAS

- Estructura de datos dinámicos que permite almacenar y organizar los elementos en una secuencia ordenada.
- Permite acceder a los elementos mediante índices.
- Es posible la inserción y eliminación de elementos.
- Se puede modificar el orden de los elementos.

```
// Crear una lista
List<String> ciudades = new ArrayList<>();

// Agregar elementos
ciudades.add(e: "Talca");
ciudades.add(e: "Santiago");
ciudades.add(e: "Iquique");

// Eliminar un elemento
ciudades.remove(index: 1);

// Obtener un elemento
System.out.println(x: ciudades.get(index: 1));

// Modificar un elemento
ciudades.set(index: 1, element: "Antofagasta");

// Recorrer la lista
for (String e : ciudades) {
    System.out.println(x: e);
}

// Ordenar la lista
Collections.sort(list: ciudades);
```



EJERCICIO LISTAS

- Crea una lista de 10 números ingresados al azar.
- Ordena la lista.
- Pide al usuario que ingrese un número límite inferior y otro superior.
- Considerando el límite, elimina todos los elementos de la lista que no se encuentren dentro de su rango.





COLECCIONES

CONJUNTOS

- Se enfocan en almacenar elementos distintos entre sí.
- No permiten elementos duplicados.
- Son útiles cuando se necesita exclusivamente datos únicos.
- Su capacidad de agregar, eliminar y modificar datos es más eficiente que en las listas (Se nota cuando la cantidad de datos es enorme).

```
// Crear un conjunto
Set<String> ciudades = new HashSet<>();

// Agregar un elemento
ciudades.add("Santiago");
ciudades.add("Rancagua");
ciudades.add("Temuco");

// Verificar si existe un elemento;
System.out.println(ciudades.contains("Talca"));

// Eliminar un elemento
ciudades.remove("Rancagua");

// Recorrer el conjunto
for (String e : ciudades) {
    System.out.println(e);
}

// Vaciar conjunto
ciudades.clear();
```



EJERCICIO CONJUNTOS

- Crea dos conjuntos con datos ingresados manualmente.
- Encuentra los elementos comunes entre ambos conjuntos y almacénalos en uno nuevo.
- **Tip:** Prueba utilizando la función *retainAll*





COLECCIONES

MAPAS

- A diferencia de las listas y los conjuntos, los mapas almacenan pares de clave-valor. Esto los vuelve ideales para estructurar datos en forma de diccionario o tabla asociativa.
- Permite acceder a los valores a través de sus claves.
- Se obtiene una mejor organización de los datos. Por ende, mayor flexibilidad a la hora de almacenar datos y acceso eficiente al obtenerlos.

```
// Crear un conjunto
Map<Integer, String> ciudades = new HashMap<>();

// Agregar dato
ciudades.put(key: 1, value: "Talca");
ciudades.put(key: 2, value: "Santiago");
ciudades.put(key: 3, value: "Linares");

// Obtener un valor
ciudades.get(key: 1);

// Verificar si existe una clave
ciudades.containsKey(key: 1);

// Eliminar dato
ciudades.remove(key: 3);

// Recorrer un mapa
for (Map.Entry<Integer, String> e : ciudades.entrySet()) {
    System.out.println(x: e);
}
```



AWAKELAB

#programmingbootcamp

nodovirtual.awakelab.cl

 jELOU futurejob by  Adalid Chile