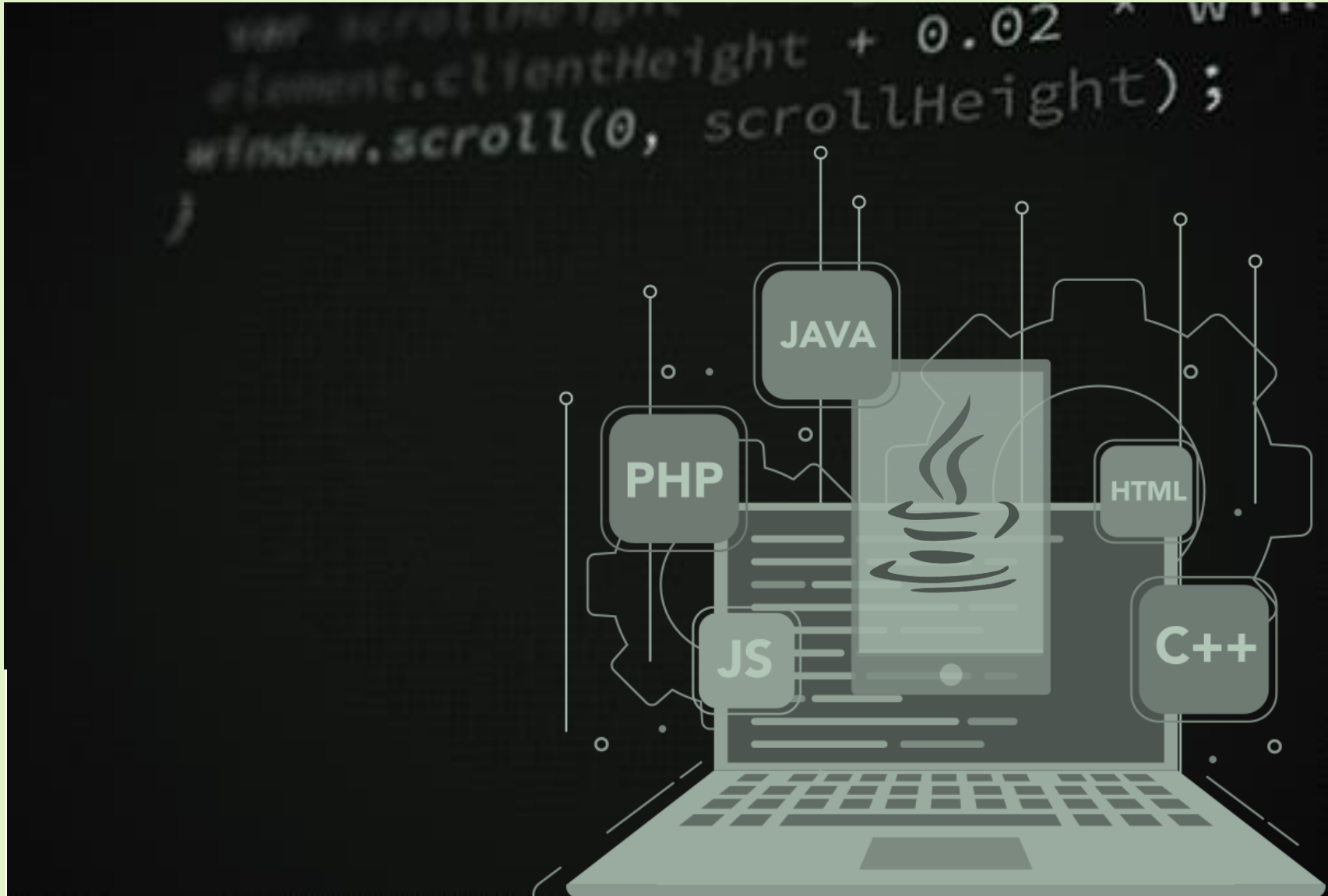
EFA
MORATALAZ

1º CFGS Desarrollo de Aplicaciones Web

PROGRAMACIÓN

DANIEL GONZÁLEZ-CALERO JIMÉNEZ

UT4 – ESTRUCTURAS DE ALMACENAMIENTO DINÁMICAS



EFA MORATALAZ

1º CFGS Desarrollo de Aplicaciones Web

PROGRAMACIÓN

UT4 – ESTRUCTURAS DE ALMACENAMIENTO DINÁMICAS

INDICE

ARRAYLIST

Almacena los elementos en un **array dinámico** que va variando su tamaño en función del número de elementos a contener. Es la implementación de *List* más eficiente en la mayoría de los casos y permite valores null. El coste computacional de utilizar esta colección se dispara al eliminar o insertar elementos, ya que esto implica el desplazamiento de varios de ellos. Añade las funcionalidades siguientes:

FUNCIÓN	DESCRIPCIÓN
add (int posición, E elemento)	Añade el elemento que se le pasa por argumento en la posición indicada.
get (int posición)	Devuelve el elemento situado en la posición indicada.
remove (int posición)	Elimina el elemento situado en la posición indicada y lo devuelve.
set (int posición, E elemento)	Sustituye el elemento situado en la posición especificada con el elemento que se pasa por argumento.
sort (arrayList)	Ordena los elementos del array
indexOf (Object o) / lastIndexOf (Object o)	Devuelve la posición de la primera/ última ocurrencia del elemento dentro de la colección

DECLARACIÓN DE ARRAYLIST

```
import java.util.ArrayList; // import the ArrayList class

ArrayList<String> cars = new ArrayList<String>(); // Create an ArrayList object
```

AÑADIR ITEM AL ARRAYLIST

```
cars.add("Volvo");
cars.add("BMW");
cars.add("Ford");
cars.add("Mazda");
System.out.println(cars);
```

ACCEDER A UN ITEM DEL ARRAYLIST

```
cars.get(0);
```

CAMBIAR UN ITEM (UPDATE)

```
cars.set(0, "Opel");
```

TAMAÑO DEL ARRAYLIST

```
cars.size();
```

ELIMINAR UN ITEM

```
cars.remove(0);
```

LIMPIAR/VACIAR EL ARRAYLIST

```
cars.clear();
```

ORDENAR EL ARRAYLIST

```
Collections.sort(cars);
```

OBTENER POSICIONES

```
for (int i = 0; i < cars.size(); i++) {
  System.out.println(cars.get(i));
}
```

EJEMPLO 1: Declarar un ArrayList unidimensional con los números 22, 77 y 11. Muestra individualmente cada uno de los elementos. Después, cambia el 77 por un 33 y muestra el ArrayList completo.

```
Primer elemento: 22
Segundo elemento: 77
Tercer elemento: 11
[22, 33, 11]
```

EJEMPLO 2: Declarar un ArrayList unidimensional y realiza las siguientes operaciones:

- **A. Añadir 10 números enteros.**
- **B. Acceder a las posiciones 0 y 7**
- **C. Reemplazar la cuarta posición por un “100”.**
- **D. Eliminar la posición 5**
- **Mostrar el arraylist resultante**

```
A. [22, 77, 11, 20, 70, 10, 29, 79, 19, 5]
B. La posicion 0 es: 22
La posicion 7 es: 79
C. [22, 77, 11, 20, 100, 10, 29, 79, 19, 5]
D. [22, 77, 11, 20, 100, 29, 79, 19, 5]
```

EJEMPLO 3: Declarar un ArrayList unidimensional de tamaño 20 con números aleatorios y:

- **A. Mostrar el arraylist resultante**
- **B. Obtener el tamaño del arraylist**
- **C. Eliminar la posición 10 y obtener el nuevo tamaño**
- **D. Obtener el número mínimo**
- **E. Obtener el número máximo**

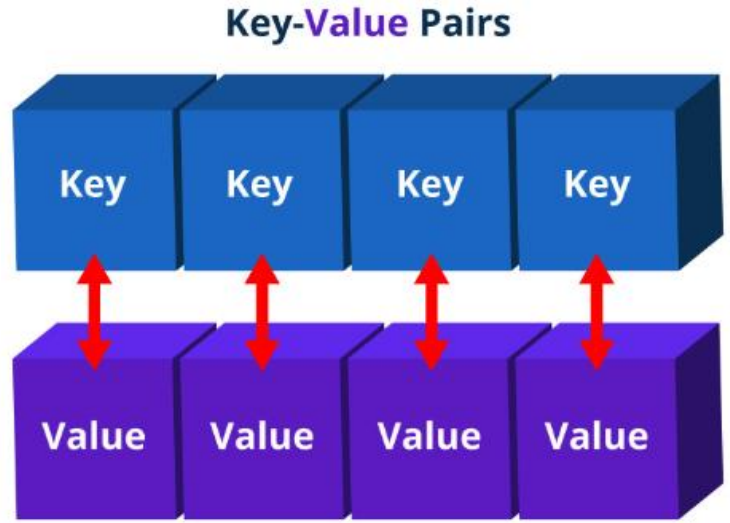
```
[37, 20, 62, 95, 64, 27, 37, 47, 94, 28, 54, 56, 77, 20, 6, 10, 5, 16, 41, 75]
El tamaño del array es: 20
Borrrando... [37, 20, 62, 95, 64, 27, 37, 47, 94, 28, 56, 77, 20, 6, 10, 5, 16, 41, 75]
El tamaño del array ahora es: 19
[5, 6, 10, 16, 20, 20, 27, 28, 37, 37, 41, 47, 56, 62, 64, 75, 77, 94, 95]
El número más pequeño es: 5
El número más pequeño es: 95
```

HASHMAP

Un **hashmap** designa claves únicas para los valores correspondientes que se pueden recuperar en cualquier punto dado.

- Los valores se pueden almacenar en un mapa formando un par **clave-valor**. El valor se puede recuperar usando la clave pasándola al método correcto.
- Hashmap almacena solo referencias de objetos. Por eso, es imposible utilizar tipos de datos primitivos como double o int. Utilice la clase contenedora (como Integer o Double) en su lugar

```
import java.util.HashMap; // import the HashMap class
```



DECLARACIÓN DEL HASHMAP

```
HashMap<String, String> capitalCities = new HashMap<String, String>();
```

AÑADIR ITEM AL ARRAYLIST

```
// Add keys and values (Country, City)
capitalCities.put("England", "London");
capitalCities.put("Germany", "Berlin");
capitalCities.put("Norway", "Oslo");
capitalCities.put("USA", "Washington DC");
System.out.println(capitalCities);
```

ACCEDER A UN ITEM

```
capitalCities.get("England");
```

VACIAR EL HASHMAP

```
capitalCities.clear();
```

ELIMINAR UN ITEM

```
capitalCities.remove("England");
```

OBTENER EL TAMAÑO

```
capitalCities.size();
```

RECORRER EL HASHMAP POSICIÓN POR POSICIÓN

- Para recorrer los hashmap es necesario el uso de los bucles **for-each.**
- Utiliza el método **keySet()** si quieres obtener sólo las claves y el método **values()** si sólo quieres obtener los valores.

```
for (String i : capitalCities.keySet()) {
  System.out.println(i);
}
```

```
for (String i : capitalCities.values()) {
  System.out.println(i);
}
```

EJEMPLO: Crea un hashmap llamado “personas” que almacenará el nombre como clave y la edad como valores.

```java
// Import the HashMap class
import java.util.HashMap;

public class Main {
  public static void main(String[] args) {

    // Create a HashMap object called people
    HashMap<String, Integer> people = new HashMap<String, Integer>();


    // Add keys and values (Name, Age)
    people.put("John", 32);
    people.put("Steve", 30);
    people.put("Angie", 33);

    for (String i : people.keySet()) {
      System.out.println("key: " + i + " value: " + people.get(i));
    }
  }
}
```

Salida:

```
Name: Angie Age: 33
Name: Steve Age: 30
Name: John Age: 32
```