Algoritmos y Estructura de Datos

Unidad 3: Clase ArrayList

Tema 10: Operaciones variadas

Semana 10

Tema 10: Operaciones variadas

Índice

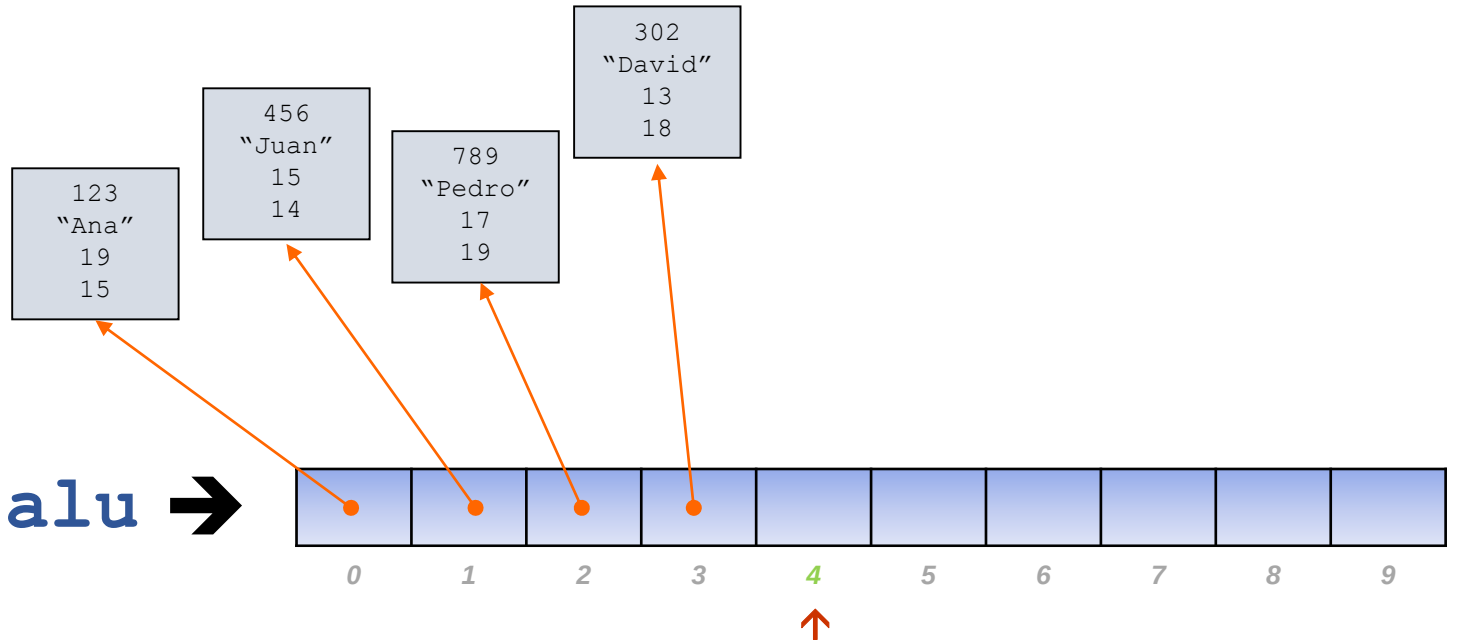
Capacidades

- Identifica los nuevos recursos en programación.
- Diseña programas utilizando el utilitario ArrayList.

3.2.1 Métodos adicionales de la clase ArrayList

```
private ArrayList <Alumno> alu = new ArrayList <Alumno> ();

    alu.add(new Alumno(123, "Ana", 19, 15));
    alu.add(new Alumno(456, "Juan", 15, 14));
    alu.add(new Alumno(789, "Pedro", 17, 19));
    alu.add(new Alumno(302, "David", 13, 18));
```



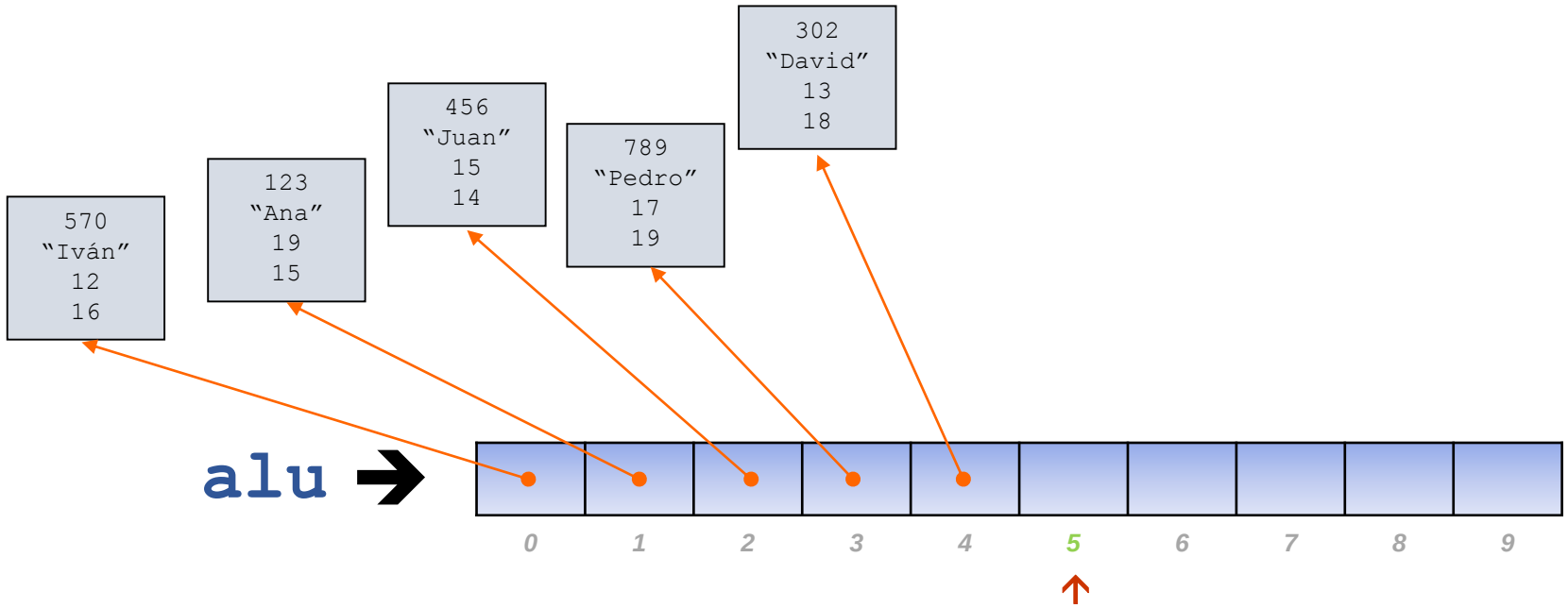
3.2.1 Métodos adicionales de la clase ArrayList

1) `public void add(int, Object) {`
`}`

Inserta una dirección de memoria en la posición indicada.

Ej:

```
alu.add(0, new Alumno(570, "Iván", 12, 16));
```



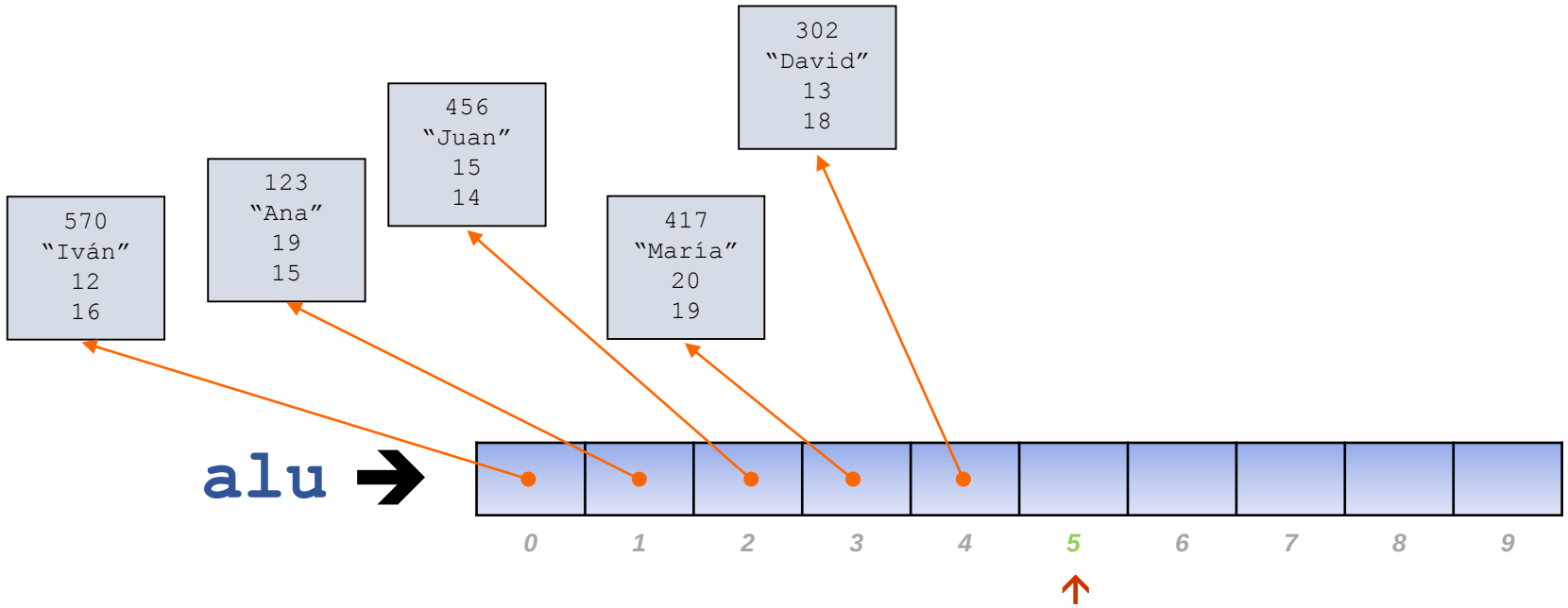
3.2.1 Métodos adicionales de la clase ArrayList

2) `public void set(int, Object) { }`

Impone una dirección de memoria en la posición indicada.

Ej:

```
alu.set(3, new Alumno(417, "María", 20, 19));
```



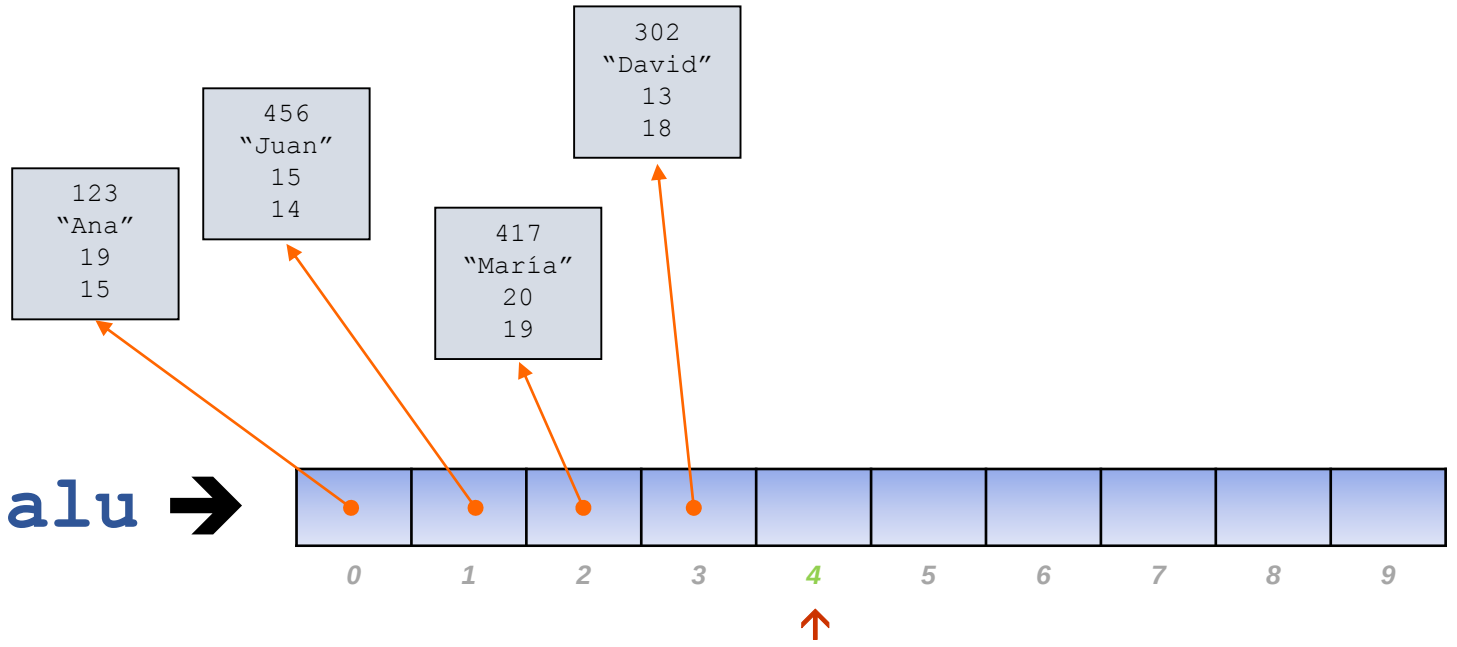
3.2.1 Métodos adicionales de la clase ArrayList

3) `public void remove(int) {`
`}`

Retira del arreglo la dirección de memoria de la posición indicada.

Ej:

```
alu.remove(0);
```



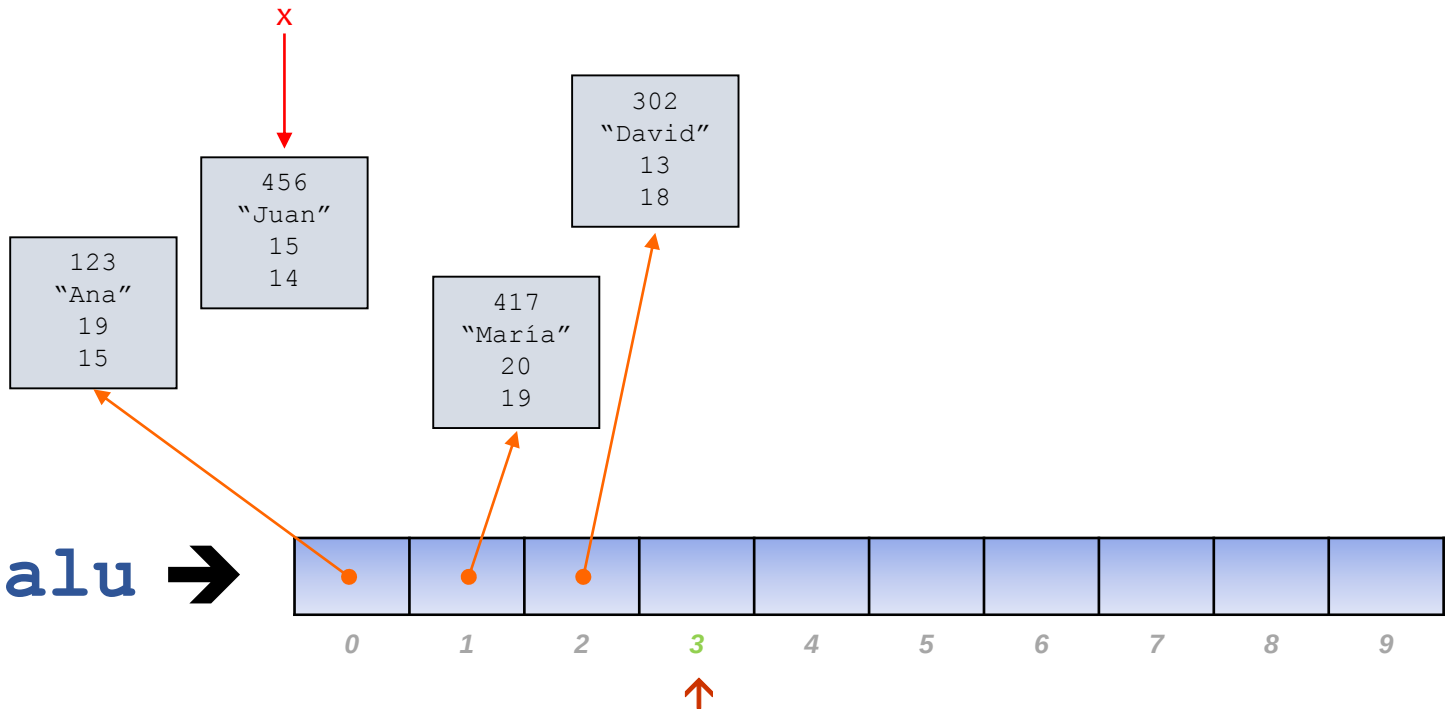
3.2.1 Métodos adicionales de la clase ArrayList

4) `public void remove(Object) {`
`}`

Retira del arreglo la dirección de memoria del objeto referenciado.

Ej:

```
Alumno x = alu.get(1); // x captura la DirMem del segundo alumno
alu.remove(x); // el segundo alumno acaba de ser retirado del arreglo
```



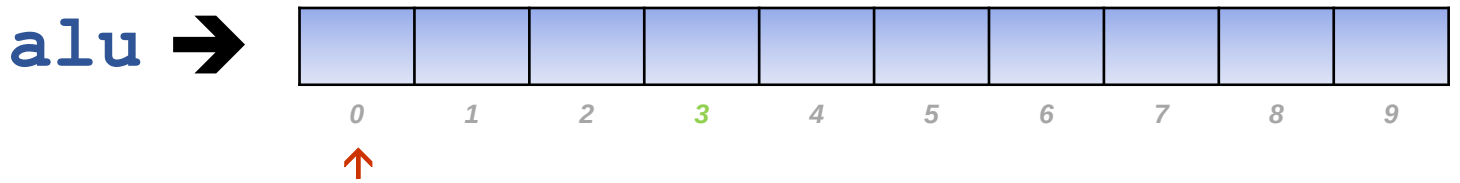
3.2.1 Métodos adicionales de la clase ArrayList

```
5) public void clear() {
   }
```

Retira del arreglo todas las direcciones de memoria.

Ej:

```
alu.clear(); // Reinicializa el ArrayList (Capacidad inicial: diez alumnos)
```



3.2.2 Operaciones públicas básicas

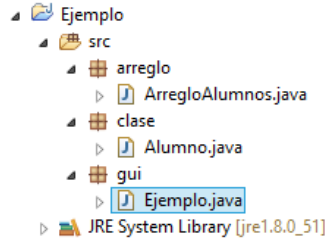
Ejemplo: método que busca un código y retorna la dirección de memoria del objeto que lo contiene. En caso no exista retorna null.

```
public Alumno buscar(int codigo) {
    for (int i=0; i<tamanio(); i++)
        if (obtener(i).getCodigo() == codigo)
            return obtener(i);
    return null;
}
```

Ejemplo: método que recibe la dirección de memoria de un objeto Alumno y lo retira del ArrayList.

```
public void eliminar(Alumno x) {
    alu.remove(x);
}
```

3.2.3 Ejemplo



a) Implementa la clase **Alumno** en el paquete ***clase*** con los atributos privados: código, nombre, nota1 y nota2; un constructor, los métodos de acceso público set/get y el método promedio.

b) Implementa la clase **ArregloAlumnos** en el paquete ***arreglo*** con el atributo privado ArrayList **alu** de tipo **Alumno**.
Implementa como públicos:
- Un constructor que crea el ArrayList **alu** de tipo **Alumno** y autogenera ocho objetos.
- Un método **adicionar** que recibe la dirección de memoria de un nuevo alumno y lo adiciona al ArrayList.
- Un método **tamanio** que retorna la cantidad de alumnos registrados hasta ese momento.
- Un método **obtener** que recibe una posición y retorna la dirección de memoria del alumno respectivo.
- Un método **buscar** que busca un código y retorna la dirección de memoria del objeto que lo contiene. En caso no existe retorna null.
- Un método **eliminar** que recibe la dirección de memoria de un objeto Alumno y lo retira del ArrayList.

c) En la clase **Ejemplo** declara y crea como variable global un objeto de tipo **ArregloAlumnos** e implementa el método listar que visualiza todos los alumnos ingresados.
Implementa la pulsación de los botones:
- **Adicionar** : adiciona un nuevo alumno verificando que el código no se repita.
- **Consultar** : busca un código y si existe edita los datos del alumno.
- **Modificar** : busca un código y si existe modifica los datos de un alumno.
- **Eliminar** : busca un código y si existe retira al alumno del arreglo.

Conclusiones

- La clase ***ArrayList*** implementada por Java nos permite administrar una colección de objetos.
- Con creatividad podemos desarrollar artificios y aprovechar al máximo la clase ***ArrayList***.

Referencias bibliográficas

- ***Joyanes Aguilar Luis***. *Fundamentos de programación: algoritmos, estructuras de datos y objetos. Madrid, España: McGraw-Hill (005.1 JOYA/A 2021)*
- ***Lewis John***. *Estructuras de datos con Java: diseño de estructuras y algoritmos. Madrid, Pearson Educación (005.73 LEWI/E 2021)*
- ***Deitel Harvey***. *Cómo programar en Java. México, D.F.: Pearson Educación (005.133J DEIT 2021)*

GRACIAS

SEDE MIRAFLORES
Calle Diez Canseco Cdra 2 / Pasaje Tello
Miraflores – Lima
Teléfono: 633-5555

SEDE INDEPENDENCIA
Av. Carlos Izaguirre 233
Independencia – Lima
Teléfono: 633-5555

SEDE BREÑA
Av. Brasil 714 – 792
(CC La Rambla – Piso 3)
Breña – Lima
Teléfono: 633-5555

SEDE TRUJILLO
Calle Borgoño 361
Trujillo
Teléfono: (044) 60-2000

SEDE SAN JUAN DE LURIGANCHO
Av. Próceres de la Independencia 3023-3043
San Juan de Lurigancho – Lima
Teléfono: 633-5555

SEDE LIMA CENTRO
Av. Uruguay 514
Cercado – Lima
Teléfono: 419-2900

SEDE BELLAVISTA
Av. Mariscal Oscar R. Benavides 3866 – 4070
(CC Mall Aventura Plaza)
Bellavista – Callao
Teléfono: 633-5555

SEDE AREQUIPA
Av. Porongoche 500
(CC Mall Aventura Plaza)
Paucarpata - Arequipa
Teléfono: (054) 60-3535