*Sección: Clases de uso general*

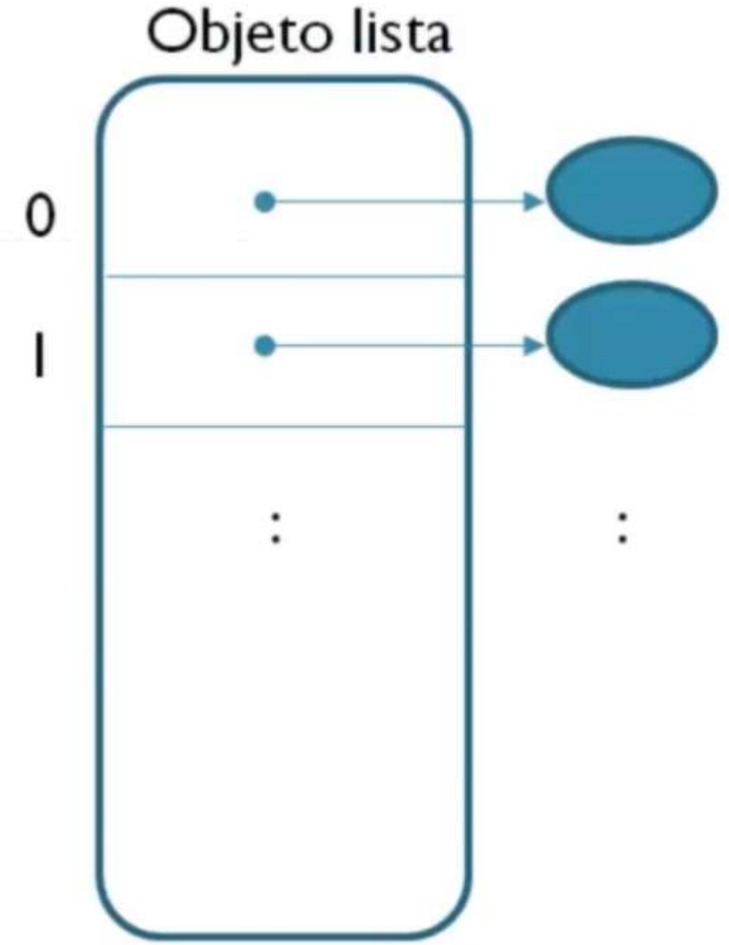
# Colecciones I

# Introducción

- Una colección es una agrupación de objetos sin tamaño fijo
- Se puede añadir y eliminar objetos de una colección dinámicamente
- Para gestionar colecciones disponemos de clases específicas en java.util
- Tipos:
  - Listas
  - Tablas
  - Conjuntos

# Colección tipo lista

- Cada elemento de la colección tiene una posición asociada, al igual que los arrays.
- La clase de colección más importante de este tipo es ArrayList.
- Es una colección de tipo genérico, lo que significa que se pueden crear colecciones de cualquier tipo Java, indicando durante la creación del objeto de colección el tipo específico con el que se va a trabajar.



# La clase ArrayList

- Es la clase de colección más utilizada.
- Para crear un objeto de la misma:

```
ArrayList<Tipo> variable=new ArrayList<>();
```

- *Tipo* es el tipo de objeto que se va a almacenar en la colección.
- Ejemplos:

```
//lista de cadenas de caracteres
ArrayList<String> cadenas=new ArrayList<>();
//lista de números enteros (objetos numéricos)
ArrayList<Integer> nums=new ArrayList<>();
```

- Una vez creado el objeto de colección, ésta se encuentra vacía

# Métodos ArrayList I

- **boolean add(T dato). Añade el dato a la colección y lo coloca al final de la misma. T representa el tipo indicado al crear el objeto de colección:**

```
ArrayList<String> nombres=new ArrayList<>();
nombres.add("Maria");//elemento en posición 0
nombres.add("Angel"); //elemento en posición 1
```

- **boolean add(int pos, T dato). Añade el elemento en la posición indicada, desplazando hacia adelante los que se encuentren en dicha posición**

```
nombres.add("Luis", 1); //desplaza a Angel a la posición 2
```

# Métodos ArrayList II

- **T set(int pos, T dato). Sustituye el elemento existente en la posición indicada por el nuevo dato suministrado como parámetro. Devuelve el elemento sustituido**

```
nombres.set(0,"Laura");//sustituye María por Laura
```

- **int size(). Devuelve el total de elementos de la colección**
- **T get(int pos). Devuelve el elemento que ocupa la posición indicada. Si la posición es menor que 0 o mayor o igual que *size()*, se producirá una excepción.**
- **T remove (int pos). Elimina el elemento que ocupa la posición indicada y lo devuelve. Si la posición es menor que 0 o mayor o igual que *size()*, se producirá una excepción**

# Recorrido de un ArrayList

- **Se puede recorrer con un bucle for estándar:**

```
//muestra el contenido del ArrayList de nombres
for(int i=0;i<nombres.size();i++){
    System.out.println(nombres.get(i));
}
```

- **También mediante un for each:**

```
//misma función que el bloque anterior
for(String s:nombres){
    System.out.println(s);
}
```

# Enunciado

- **Realizar un programa para la gestión de notas que al iniciarse muestre el siguiente menú:**

1.- Agregar nota
2.- Ver nota media
3.- Ver aprobados
4.- Salir

- **A elegir la opción 1, se solicitará la introducción de una nota y se guardará, volviendo a mostrar de nuevo el menú. Con las opciones 2 y 3 se mostrará, respectivamente, la nota media registrada hasta el momento y el número de aprobados. A elegir 4 se abandonará el programa**