

DIPLOMATURA EN

NUEVAS TECNOLOGÍAS

MÓDULO 1

Introducción a la Programación

Docente:

Esp. Ing. Martín Polliotto





Repaso de Semana anterior

- ✓ **Api de Java**
- ✓ **Paquetes**
- ✓ **Clases de envoltorio**
- ✓ **Strings, Date y Calendar**
- ✓ **Interfaces gráficas con Swing**



DIPLOMATURA EN

**NUEVAS
TECNOLOGÍAS**



Ministerio de
**PROMOCIÓN DEL EMPLEO
Y DE LA ECONOMÍA FAMILIAR**

Ministerio de
**CIENCIA Y
TECNOLOGÍA**



Con el apoyo de
Oficina de Montevideo
Oficina Regional de Genios
para América Latina y el Caribe

Contenidos Semana 08:

- 8.1 ArrayList
- 8.2 **static** y **final**
- 8.3 Componentes: JComboBox y JTable



DIPLOMATURA EN

**NUEVAS
TECNOLOGÍAS**



Ministerio de
**PROMOCIÓN DEL EMPLEO
Y DE LA ECONOMÍA FAMILIAR**

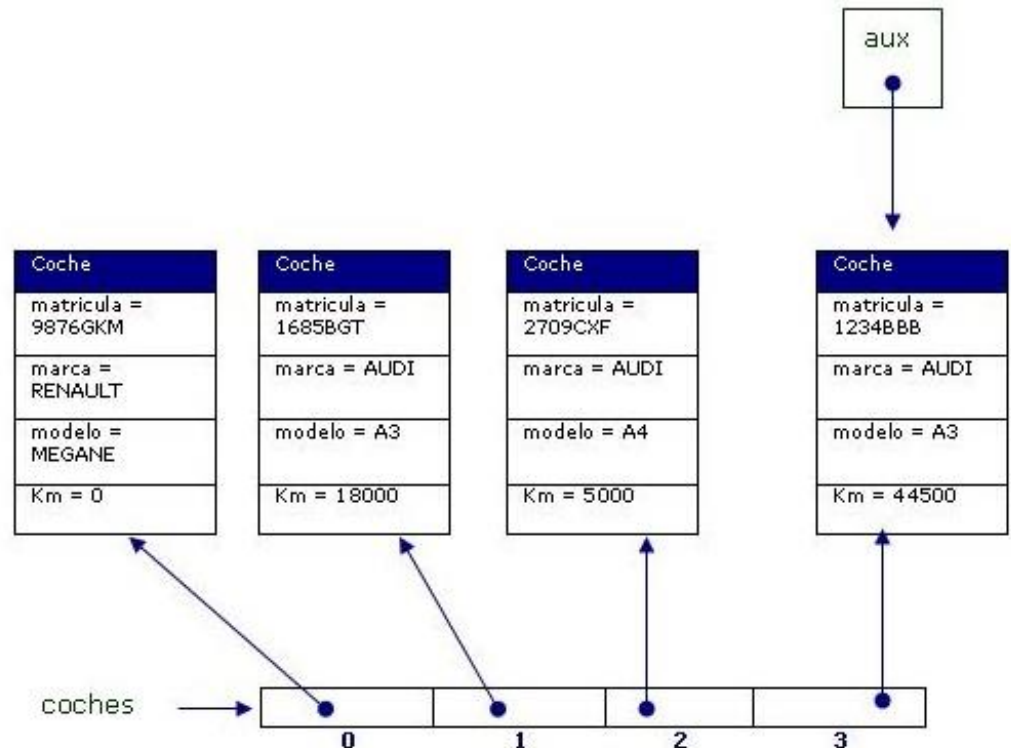
Ministerio de
**CIENCIA Y
TECNOLOGÍA**



Con el apoyo de
Oficina de Montevideo
Oficina Regional de Genios
para América Latina y el Caribe

8.1 ArrayList

- ✓ Esta clase representa una **lista** implementada sobre un arreglo (los nodos de la lista están almacenados dentro de un arreglo)
- ✓ Permite trabajar con un conjunto **dinámico de** datos (tanto primitivos como objetos).
- ✓ Paquete **java.util**

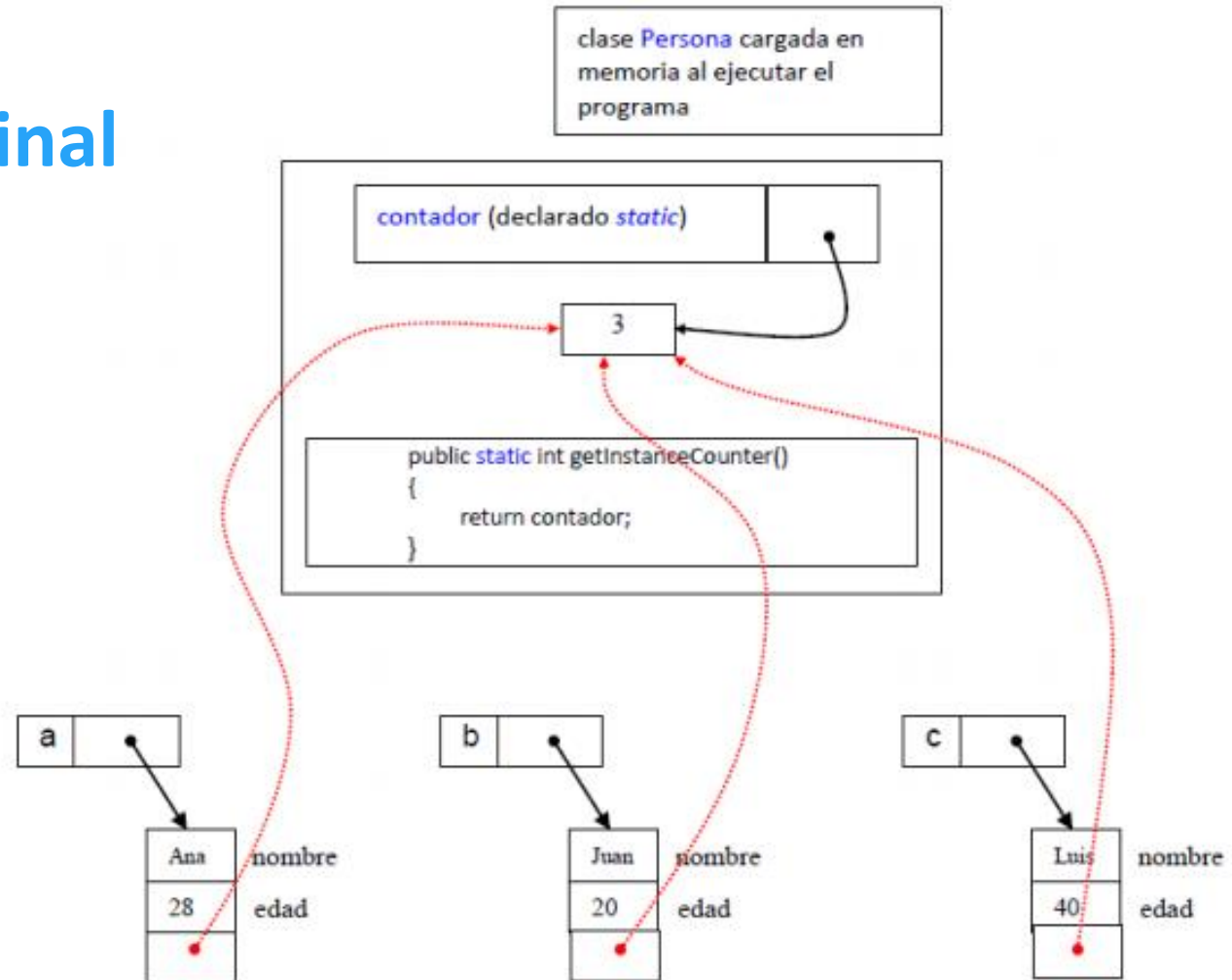


8.1 ArrayList

✓ Métodos principales:

MÉTODO	DESCRIPCIÓN
size()	Devuelve el número de elementos (int)
add(X)	Añade el objeto X al final. Devuelve true.
add(posición, X)	Inserta el objeto X en la posición indicada.
get(posicion)	Devuelve el elemento que está en la posición indicada.
remove(posicion)	Elimina el elemento que se encuentra en la posición indicada. Devuelve el elemento eliminado.
remove(X)	Elimina la primera ocurrencia del objeto X. Devuelve true si el elemento está en la lista.
clear()	Elimina todos los elementos.
set(posición, X)	Sustituye el elemento que se encuentra en la posición indicada por el objeto X. Devuelve el elemento sustituido.
contains(X)	Comprueba si la colección contiene al objeto X. Devuelve true o false.
indexOf(X)	Devuelve la posición del objeto X. Si no existe devuelve -1

8.2 Static y final



- **static** permite definir atributos y método compartidos por todos los objetos de una clase
- **Ejemplo: `Math.random()` o bien `Math.PI`**



8.2 Static y final

- ✓ Las variables indicadas con **final** son consideradas **constantes**.
- ✓ Una vez asignado un valor inicial a una constante, el mismo no puede ser modificado durante el resto del programa
- ✓ Un atributo declarado final (sin que haya combinación con static) puede ser asignado en el momento de la declaración, o dentro de un constructor de la clase
- ✓ Si un atributo se declara static final, debe ser asignado al declararse.

- Por convención **los nombres de las constantes se escriben en mayúsculas, separado por _**
- Ejemplo **MiClase.CONSTANTE**



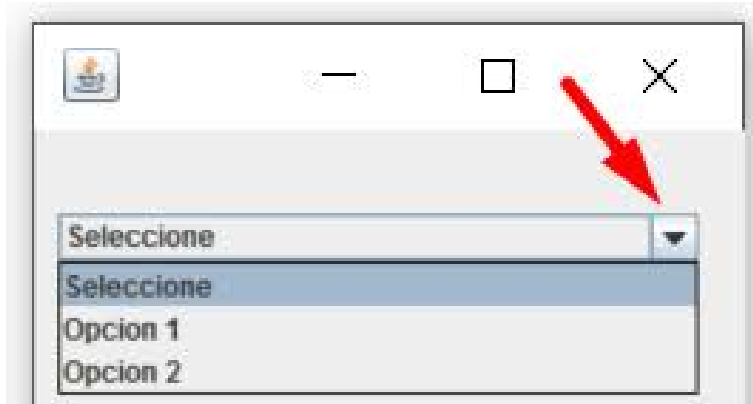
8.2 Static y final

- ✓ Los métodos indicados como **final** no se pueden sobrescribir.
- ✓ Las clases marcadas como final no se pueden **HEREDAR**.
- ✓ **En general** los atributos finales y estáticos suelen utilizarse como **públicos**.

Por ejemplo:

```
public class Test{  
    public static final String POR_DEFECTO = "Nok";  
}  
  
public class Ejecutable{  
  
    public static void main(String args[]){  
        System.out.println(Test.POR_DEFECTO);  
    }  
}
```


8.3 JComboBox



- **Comportamientos:**
 - `getSelectedIndex()`
 - `getSelectedItem()`
- **Eventos:**
 - `ActionPerformed`

Opción 1)

//Luego de inicializar los componentes:

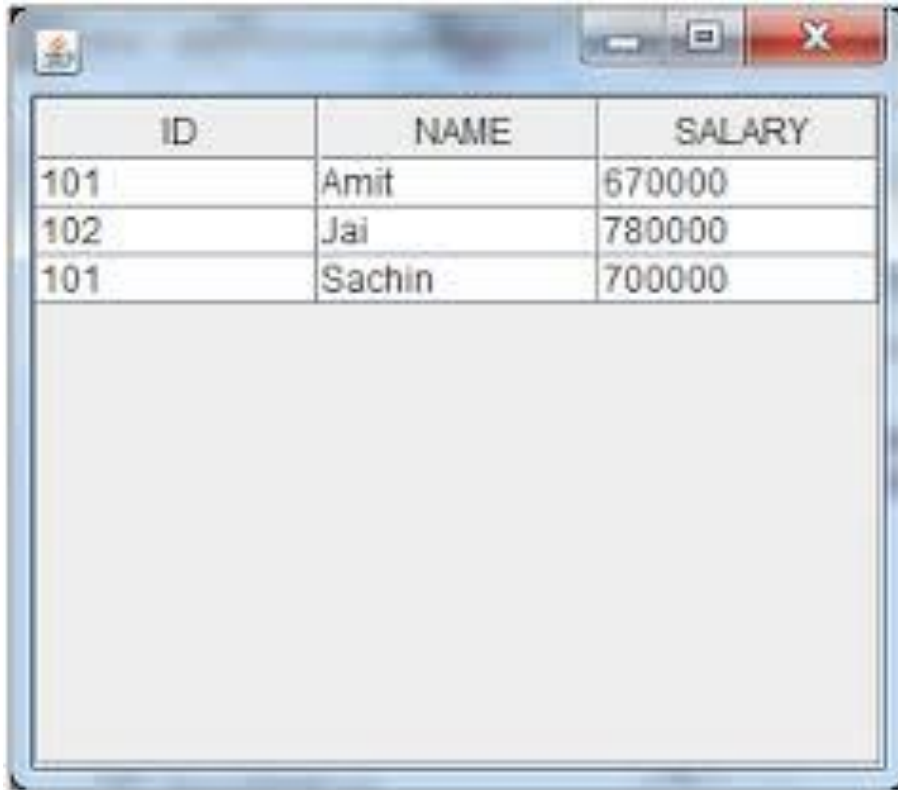
//Creamos un arreglo con los valores:
`String valores[] = {"Seleccione", "Opción 1", "Opción 2"};`
`jcMiCombo.setModel(new DefaultComboBoxModel<>(valores));`

Opción 2)

`jcMiCombo.addItem("Seleccione");`
`jcMiCombo.addItem("Opcion 1");`
`jcMiCombo.addItem("Opcion 2");`
`jcMiCombo.setSelectedIndex(0);`

- Por defecto la lista de elementos es de tipo **String**, pero en realidad puede contener **cualquier tipo de objetos**.

8.3 JTable



ID	NAME	SALARY
101	Amit	670000
102	Jai	780000
101	Sachin	700000

```
Multa[] filas = new Multa[3];  
    multas[0] = new Multa(1, 1, 300);  
    multas[1] = new Multa(2, 9, 1000);  
    multas[2] = new Multa(3, 9, 1000);
```

```
DefaultTableModel modelo = new  
DefaultTableModel();
```

```
modelo.setColumnIdentifiers(new  
String[]{"Acta", "Código", "Monto"});
```

```
for (Multa multa : multas) {  
    modelo.addRow(new  
Object[]{multa.getActa(), multa.getCodigo(),  
multa.calcularMonto()});  
}
```

```
jTable1.setModel(modelo);
```

- Es posible definir un **modelo de datos propio** mediante una clase que extienda de **AbstractTableModel**

Caso práctico



Vamos a NetBeans
de nuevo!