ESCUELA DE
INGENIERÍA EN CIENCIAS Y SISTEMAS
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

ECYS | DIDT
DESARROLLO DE TRANSFERENCIA TECNOLÓGICA

Día, Fecha:	03/08/2023
Hora de inicio:	09:00

Introducción a la Programación y Computación 1 [Sección B]

Diego Fernando Cortez Lopez

Agenda

- Anuncio
- Lectura de Enunciado
- Clase 2
- Ejemplo Práctico

Si están interesados llenar el siguiente formulario:
https://forms.gle/v3nhu4cf8HCGjcy17
(Ingresar solo con su correo institucional)

Fundamentos de la Programación y Java - Parte 2

Clase 2

Operadores en JAVA

Operadores Aritméticos

Operador	Descripción
+	Operador de Suma
-	Operador de Resta
*	Operador de Multiplicación
/	Operador de División
%	Operador de Resto

Operadores Unarios

Operador	Descripción
+	Operador unario Suma
-	Operador unario Resta
++	Operador de Incremento
--	Operador de Decremento
!	Operador de Complemento Lógico

Operadores Relacionales

Operador	**Descripción**
==	Igual a
!=	No igual a
>	Mayor que
>=	Mayor o igual que
<	Menor que
<=	Menor o igual que

Operadores Condicionales

Operador	**Descripción**
&&	Operador Condicional AND
||	Operador Condicional OR
?:	Operador Ternario
instanceof	Operador instanceof

Arreglos

- Es una estructura de datos que nos permite un conjunto de datos del mismo tipo.
- El tamaño se los arreglos de declarar una única vez al momento de su creación y no se puede modificar después como si se pudiera con un ArrayList
- Cada elemento se puede acceder por medio de un índice el cual empieza desde el número 0.

Formas de declaración:

1. Sin valores:

   <tipo>[ ] <nombre> = new <tipo>[<tamaño>]

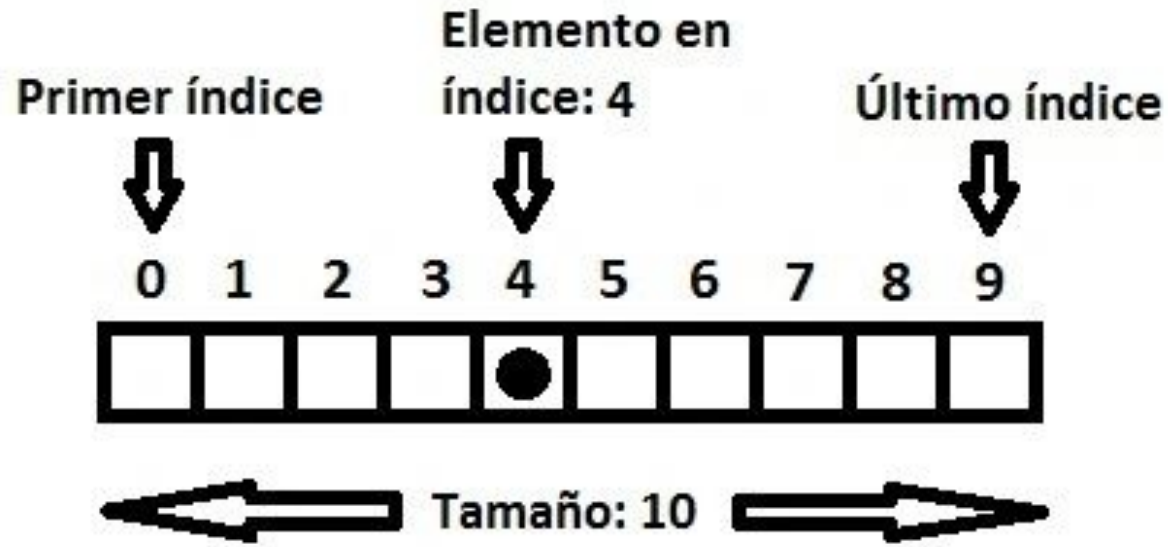
```
int[] nums = new int[10];
```

2. Con valores:

   <tipo>[ ] nums = {<Datos almacenados>};

```
int[] nums = { 1, 7, 8, 2, 3, 10, 3, 8, 2, 10};
```

Índice de cada elemento de los arreglos



Nota: Para obtener el tamaño del arreglo se utiliza la propiedad

length, nums.length

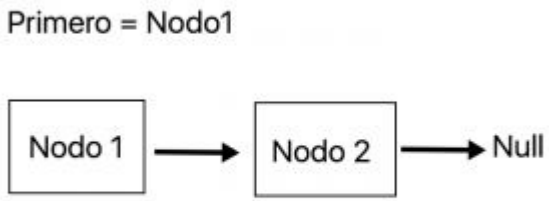
Metodos de Ordenamiento

- Para poder ordenar una cantidad determinada de números almacenados en un vector o matriz, existen distintos métodos con distintas características y complejidad.

- Existe desde métodos más simples, como:
  - Metodo Burbuja
  - Método por Inserción
  - Método por Selección
  - Método Rápido

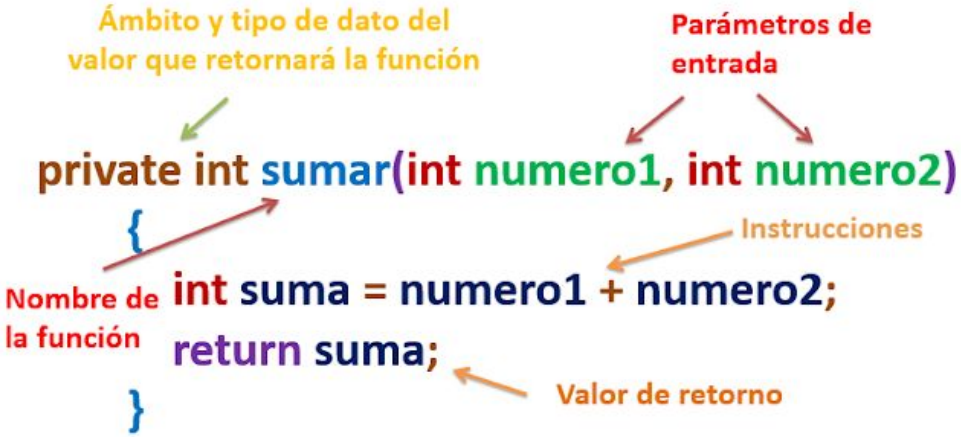
Listas Dinámicas

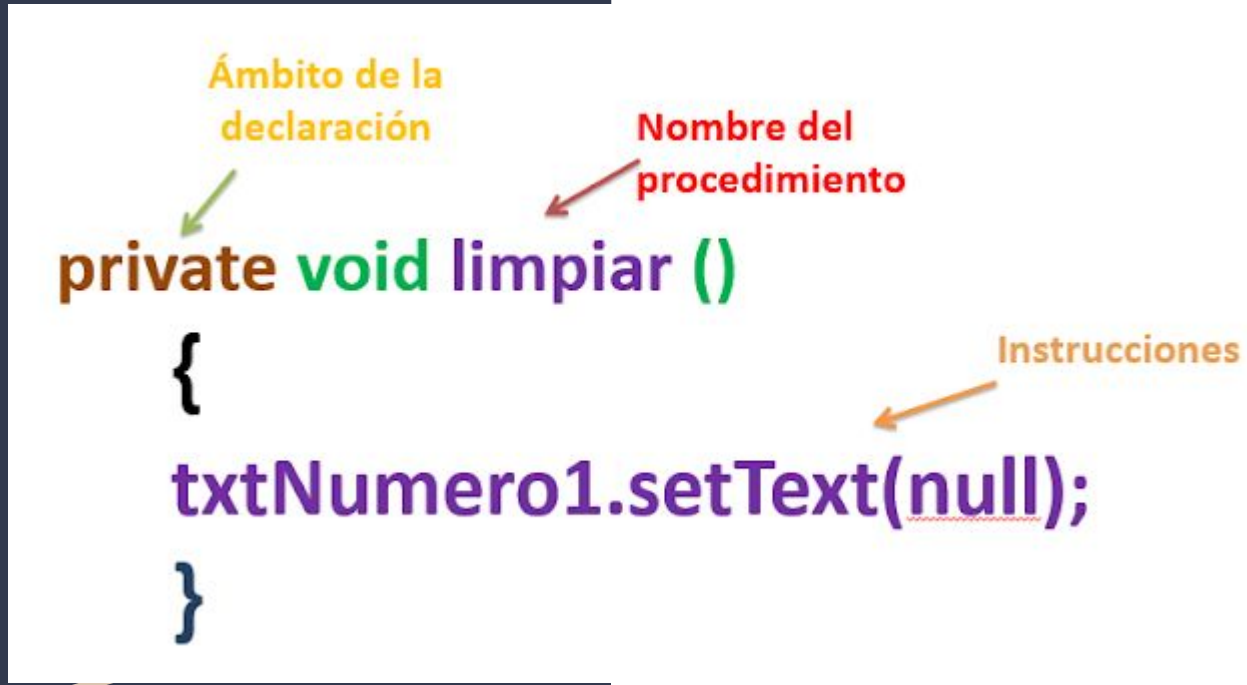
- Una lista dinámica es una colección de objetos enlazados entre sí.
- Estas pueden crecer tanto como se quiera.
- Está compuesta por una clase que maneja la lista y otra que contiene información de los objetos de la lista, llamados nodos.



Funciones

- Se indica el tipo de retorno indicado en caso de tener uno.
- Se debe indicar el tipo de cada parámetro.
- Si hay más de un parámetro de un parámetro se separan por medio de comas.
- Se utiliza la palabra reservada return para devolver un valor.
- Después del retorno no se ejecutará el código que esté debajo de él dentro de la función.




Ámbito de la declaración
Nombre del procedimiento
private void limpiar ()
{
Instrucciones
txtNumero1.setText(null);
}

Ámbito de variables

El ámbito de una variable define su alcance de uso, es decir, indica en qué secciones de código una variable estará disponible.

Fuera de este ámbito, una variable no podrá ser accedida.

Tipos de ámbito:

- Local
- Global
- Estático

Main.java

```java
class Main {
    public static boolean verificarEdad(int edad) {
        if (edad >= 18) {
            return true;
        }
        return false;
    }

    public static void main(String[] args) {
        boolean respuesta = verificarEdad(18);
        System.out.println(respuesta);
        // Salidad: true
    }
}
```

Recursividad

La recursividad es una característica que permite que un método se invoque a sí mismo. La recursividad es útil para resolver problemas definibles en sus propios términos.

- Directa o simple: El método se llama a sí mismo dentro de él.
- Indirecta, cruza o doble: Una función llama a otra y esta otra que también llama a la primera.

Recursividad Directa

Main.java

```java
class Main {
    public static int multiplicacion(int num1, int num2) {
        // Caso base:
        if (num2 == 1) {
            return num1;
        }
        // Llamda recursiva:
        return num1 + multiplicacion(num1, num2 - 1);
    }

    public static void main(String[] args) {
        int resultado = multiplicacion(3, 2);
        System.out.println(resultado);
        // Salidad: 6
    }
}
```

Recursividad Indirecta

Main.java

```java
class Main {
    static boolean esPar(int n) {
        if (n == 0) return true;
        return esImpar(n - 1);
    }

    static boolean esImpar(int n) {
        if (n == 0) return false;
        return esPar(n - 1);
    }

    public static void main(String[] args) {
        boolean resultado = esPar(3);
        System.out.println(resultado);
        // Salidad: falso
    }
}
```

INPUT y OUTPUT JAVA

Output

```
System.out.println(); or

System.out.print(); or

System.out.printf();
```

Input

```
import java.util.Scanner;


// create an object of Scanner
Scanner input = new Scanner(System.in);

// take input from the user
int number = input.nextInt();
```

¿Dudas?

Tarea No. 1

Describir con sus palabras la funcionalidad de los modificadores de acceso:

- Public
- Private
- Protected

Fecha de entrega: 11/02/2023