



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



Día, Fecha:

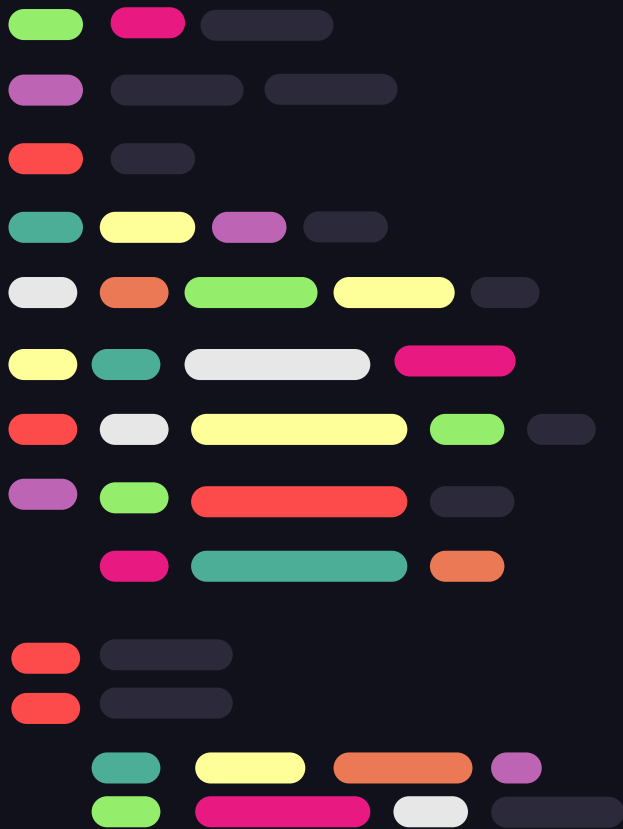
Jueves, 01/02/2024

Hora de inicio:

17:20

Introducción a la Programación y Computación 1 [B]

Josué Rodolfo Morales Castillo



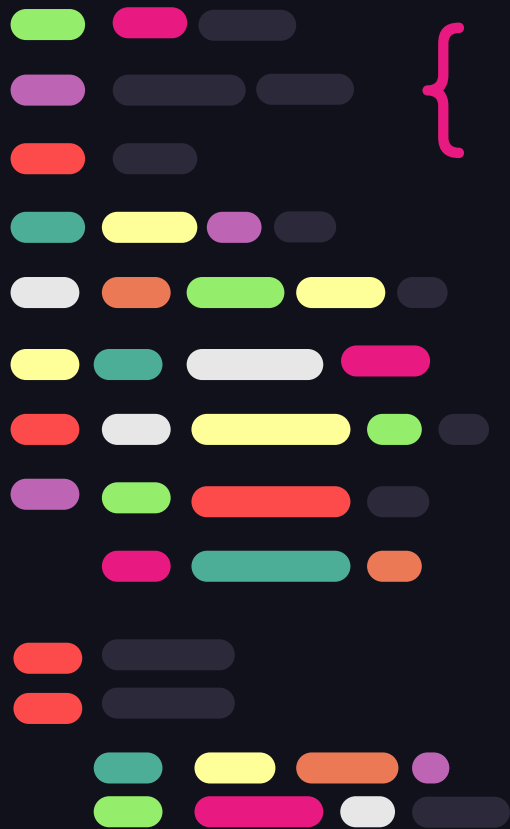
{ ..



Clase 2- Agenda

- Foro No. 2
- Asignación DTT (Formulario)
- Recordatorio tarea 1
- Fundamentos de Programación y JAVA
- Aviso Práctica 1

} ..



Fundamentos de Programación y JAVA

Parte 2

Arreglos

Es un objeto que almacena una colección de elementos del mismo tipo. Estos elementos pueden ser variables primitivas o no primitivos. Son estructuras de datos que te permiten almacenar y manipular múltiples valores bajo un solo nombre de variable.

Unidimensionales

Almacena elementos del mismo tipo en una secuencia lineal. Podemos pensar en un arreglo unidimensional como una lista ordenada de elementos, donde cada elemento tiene un índice único que comienza desde cero.

Bidimensionales

Almacena elementos en filas y columnas, formando una especie de cuadrícula o matriz. Podemos pensar en un arreglo bidimensional como una tabla en la que cada elemento tiene dos índices: uno para la fila y otro para la columna.

Unidimensionales

Tipo de dato
de los elementos del
vector

Nombre del vector

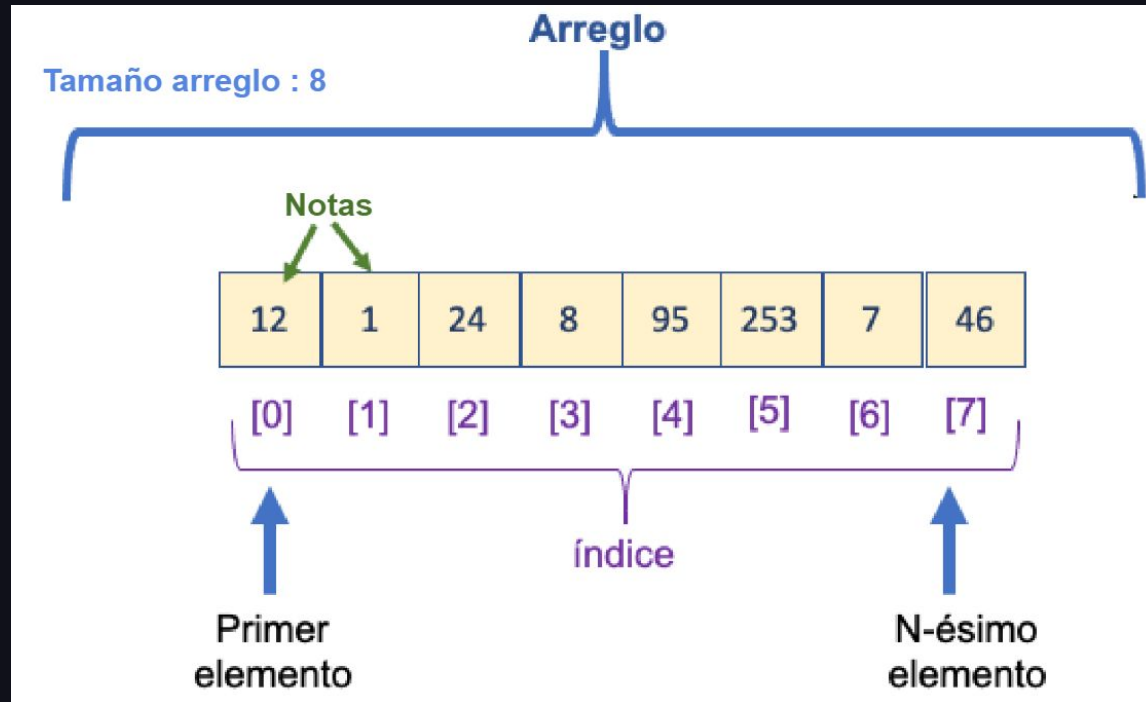
```
int[] notas = new int[8];
```

Número de
elementos del vector

```
notas[0] = 12 ;
```

Asignación de valores

Unidimensionales



Bidimensionales

Tipo de dato
de los elementos de la
matriz

Nombre de la matriz

Filas y columnas de la
matriz

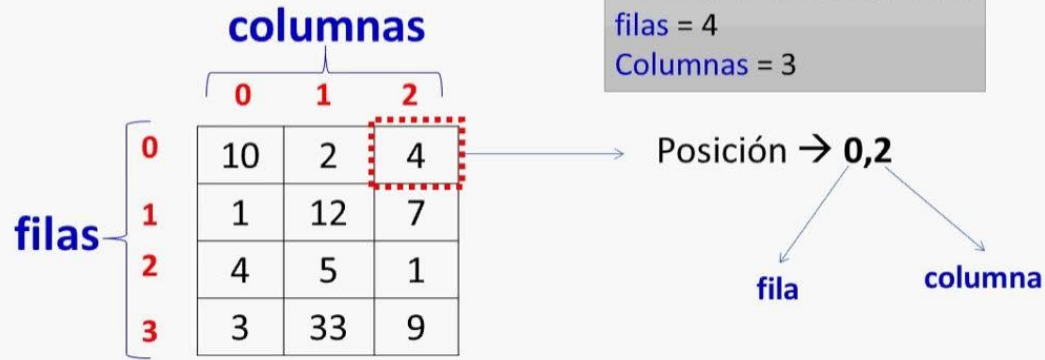
Asignación de valores

```
int[][] notas = new int[filas][columnas];  
  
notas[0][0] = 14;
```

Bidimensionales

Matriz

- Arreglo bidimensional que se compone de **filas** y **columnas**
- En cada posición se almacena un elemento de un tipo
- Todos los elementos de la matriz son del mismo tipo



Listas Dinámicas

Las listas dinámicas son estructuras de datos que pueden cambiar de tamaño durante la ejecución del programa, lo que las hace útiles cuando no conocemos de antemano la cantidad exacta de elementos que contendrá la lista. Se implementan generalmente mediante la interfaz `List` del paquete **java.util**.

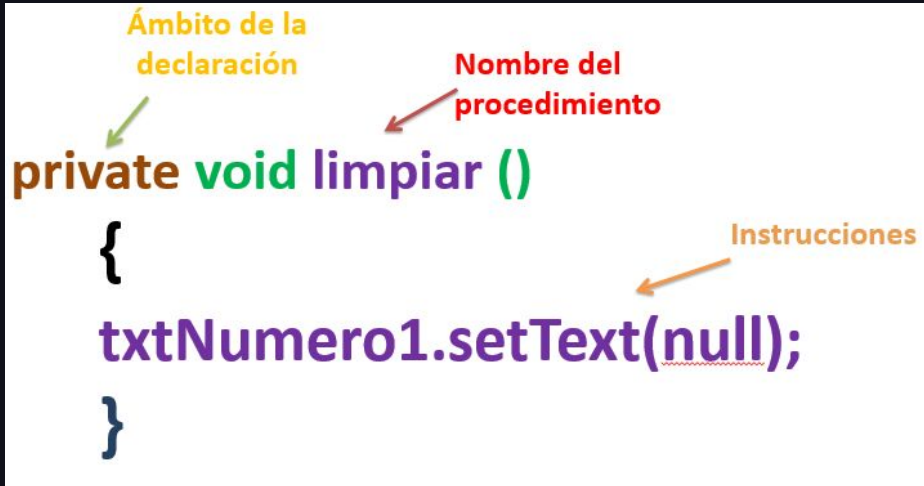
```
import java.util.ArrayList;
public class ArrayListExample {
    public static void main(String[] args) {
        // Crear un ArrayList de tipo String
        ArrayList<String> listaDinamica = new ArrayList<>();
        // Agregar elementos
        listaDinamica.add("Juan");

        // Acceder a elementos
        System.out.println("Elemento en la posición 1: " + listaDinamica.get(0));
        //Resultado: Elemento en la posición 1: Juan
    }
}
```

Procedimientos en Java

Es un bloque de código que realiza una tarea específica y se ejecuta cuando es llamado. Pueden recibir (no siempre) datos como entrada (parámetros), procesar estos datos y realizar acciones, pero no generan un resultado que se pueda utilizar directamente en la llamada al procedimiento, es decir no retorna nada.

Permite la reutilización y organización eficiente del código



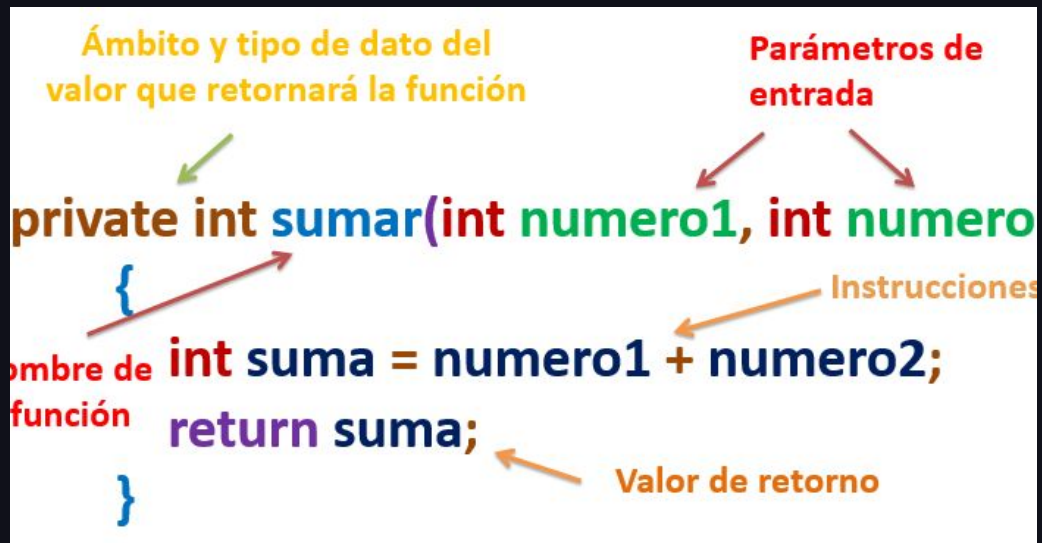
```
private void limpiar ()  
{  
    txtNumero1.setText(null);  
}
```

The diagram illustrates the components of a Java procedure declaration. It shows the code: `private void limpiar ()`, `{`, `txtNumero1.setText(null);`, and `}`. Annotations with arrows point to specific parts: 'Ámbito de la declaración' (Scope of the declaration) points to 'private'; 'Nombre del procedimiento' (Name of the procedure) points to 'limpiar'; and 'Instrucciones' (Instructions) points to the line `txtNumero1.setText(null);`.

Funciones en Java

Es un bloque de código que realiza una tarea específica y se ejecuta cuando es llamado. Las funciones pueden (no siempre) recibir datos como entrada (parámetros), procesar estos datos, la diferencia con los procedimientos es que las funciones devuelven un resultado (valor de retorno).

En java, las funciones deben definirse con un tipo de Dato, y el dato que retorna es el mismo tipo con el que se definió.



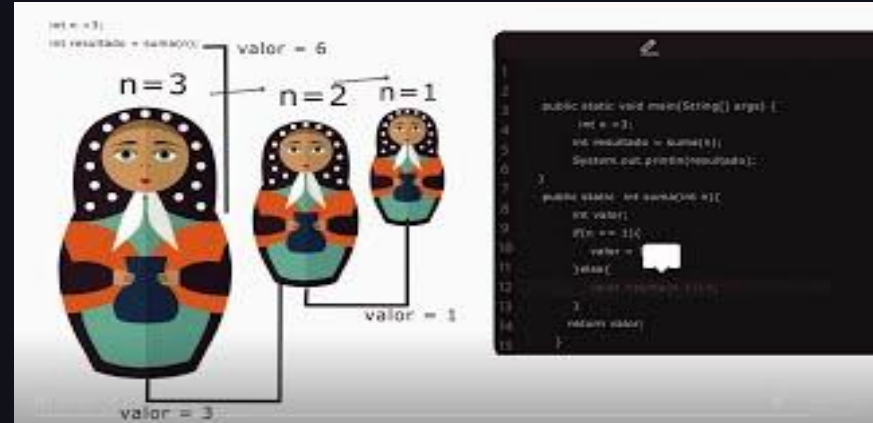
Recursividad

Recursividad Simple

La recursividad simple se refiere al uso de una función que se llama a sí misma directamente, sin la intervención de ninguna función auxiliar. Es decir, en un contexto más sencillo, una función es recursiva simple si se llama a sí misma directamente en su definición.

1 reference

```
public int EjemploRecursividad(int numero)
{
    if (numero == 0) return 1;
    return numero * EjemploRecursividad(numero - 1);
}
```



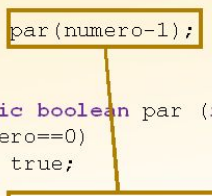
Recursividad

Recursividad Indirecta

La recursividad indirecta ocurre cuando una función A llama a otra función B, y la función B, a su vez, vuelve a llamar a la función A o a alguna otra función que finalmente llama de nuevo a la función original. En otras palabras, las llamadas recursivas se producen entre dos o más funciones de manera circular.

```
public static boolean impar (int numero){
    if (numero==0)
        return false;
    else
        return par(numero-1);
}

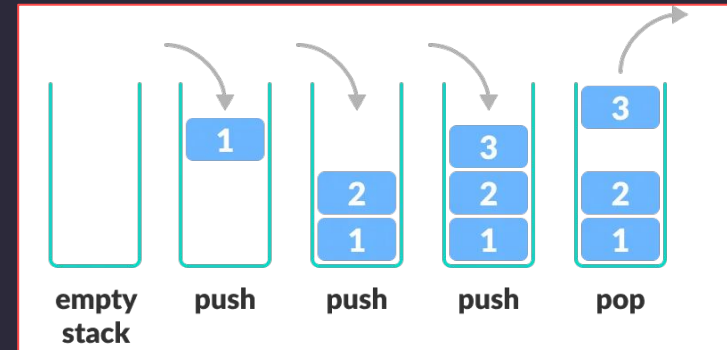
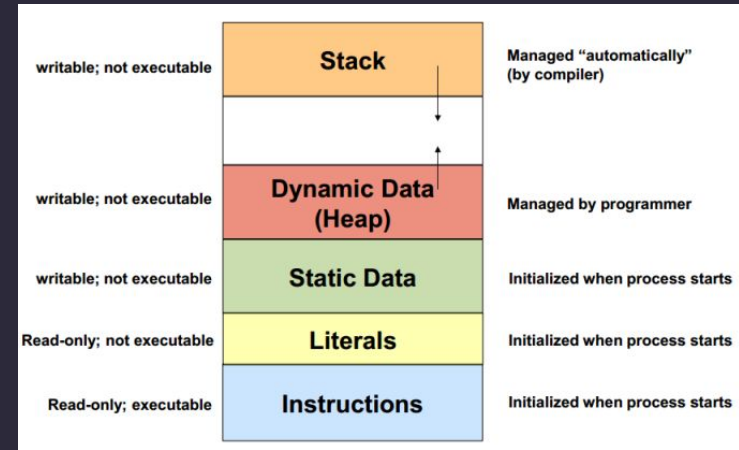
public static boolean par (int numero){
    if (numero==0)
        return true;
    else
        return impar(numero-1);
}
```



Manejo de memoria

Memoria Stack

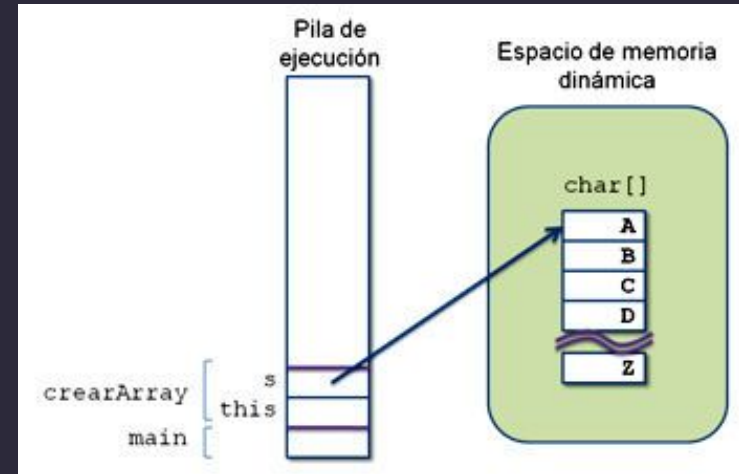
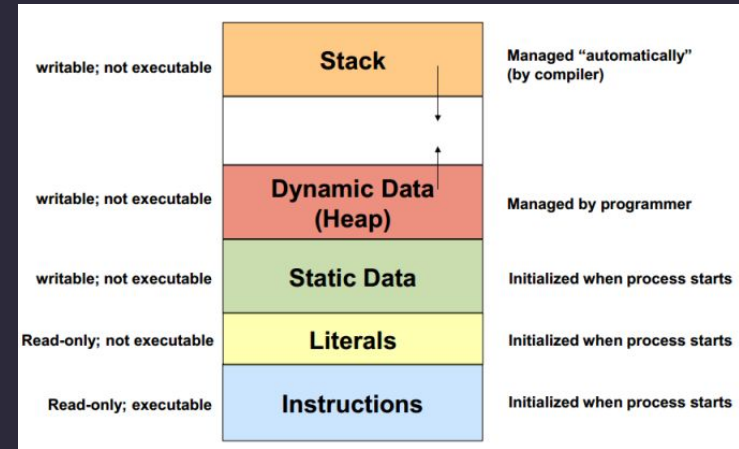
En programación, la memoria de la pila (stack) es una región de la memoria que se utiliza para almacenar variables locales y gestionar llamadas a funciones. Es una estructura de datos tipo pila (stack), lo que significa que las operaciones de inserción y eliminación de datos se realizan en el extremo superior de la pila.



Manejo de memoria

Memoria Heap

La memoria heap es una región de memoria dinámica que se utiliza para la asignación de memoria durante la ejecución de un programa. Es un área de memoria que se reserva para almacenar objetos y estructuras de datos que se crean y eliminan durante la ejecución del programa.



Metodos de Ordenamiento



Son algoritmos diseñados para organizar elementos en una secuencia en un orden específico, como ascendente o descendente. Estos métodos son fundamentales en ciencias de la computación y son utilizados en una variedad de aplicaciones para organizar datos de manera eficiente.

Sorting Algorithms





{ .. Burbuja (Bubble Sort)

Compara pares de elementos adyacentes y los intercambia si están en el orden incorrecto. Este proceso se repite hasta que no se requieren más intercambios.

8 5 3 1 4 7 9





{ .. Por Inserción (Insertion Sort)

Construye una secuencia ordenada uno a uno tomando elementos de la lista de entrada y colocándolos en la posición correcta.



6 5 3 1 8 7 2 4





{ .. Por Selección (Selection Sort)

Encuentra el elemento más pequeño y lo coloca en la primera posición, luego encuentra el siguiente elemento más pequeño y lo coloca en la segunda posición, y así sucesivamente.



	8
	5
	2
	6
	9
	3
	1
	4
	0
	7





Quick Sort

Utiliza el enfoque de "dividir y conquistar". Selecciona un elemento como "pivote" y organiza los demás elementos alrededor del pivote, dividiendo la lista en dos subconjuntos. Luego, se repite el proceso para los subconjuntos.

6 5 3 1 8 7 2 4





Comparación

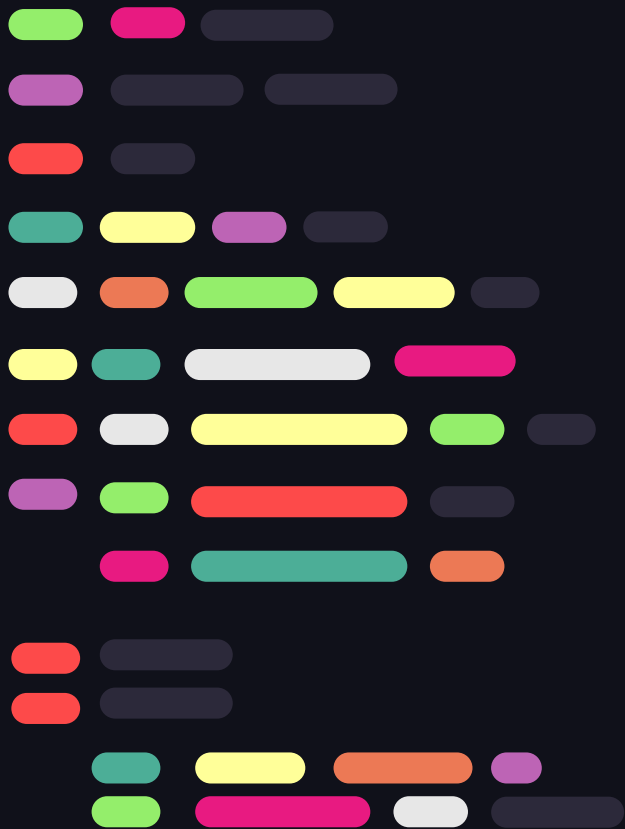


Método de Ordenamiento	Complejidad Temporal	Detalles
Bubble Sort	$O(n^2)$	Simple y fácil de entender, ineficiente para grandes conjuntos de datos.
Insertion Sort	$O(n^2)$	Eficiente para conjuntos pequeños o parcialmente ordenados.
Selection Sort	$O(n^2)$	Ineficiente para grandes conjuntos de datos.
Quick Sort	$O(n \log n)$	Eficiente para grandes conjuntos de datos, pero puede ser inestable.



¿Dudas?





{ ..



Ejemplo

} ..