# Trabajo 1

## Estructura de datos

Universidad de Cartagena



Carlos Esteban Lara Gil

Diego Andrés Arias Hernandez

Yeshua Martínez Chavez

Jeraldo de Jesus Morales Morales

Hawer Morelos Pitalua

Prof. John Arrieta

Cartagena de indias 31 de Marzo

Ingeniería de Software sede Piedra Bolívar

[**Trabajo 1 1**](#_pc6lzrlco4dk)

[Estructura de datos 1](#_abhlkl7v2duf)

[Introducción. 3](#_bk6tr0enl2kn)

[Objetivos 3](#_1mrberf06u66)

[Justificación. 3](#_3kx2l7ioq93k)

[Desarrollo 4](#_16ig8v4h244m)

[1. Declarar y definir arreglos. 4](#_wl1c5iy0x6ej)

[2. Manipular los elementos dentro de arreglos: 12](#_t0fk35dxnw7)

[3. Buscar elementos en un arreglo: 14](#_va15tlcsdo9m)

[4. Ordenar arreglos 16](#_37sfxw7y0960)

[5. Trabajar con Matrices (tablas o arreglos multidimensionales) 19](#_90h0hzxzaw73)

## **Introducción.**

Con este primer trabajo se busca abordar lo principal de las estructuras de datos, lo cuál son arreglos y matrices que le dan sentido a nuestra estructura de datos usando el lenguaje de programación tipado fuerte más popular y estructurado que es Java ya que al ser un lenguaje cuyo paradigma es la programación orientada a objetos la hace más sencillo y legible la estructura de datos.

## **Objetivos**

En este trabajo se abordan: *Las funciones, las matrices y los arreglos*. Se busca que el estudiante alcance la plena comprensión de estos temas ya que es importante para en el futuro poder generar bases de datos estables y fuertes.

Aprender a documentar y llevar un proyecto en el semestre nos ayuda también a que estemos en un ambiente de programación más acercado a lo que es un trabajo en el mundo real con los retos que implica una organización empresarial.

El estudiante será capaz de implementar funciones y arreglos para poder arreglar un código o crearlo acorde a las necesidades del proyecto donde se encuentre a futuro. Por lo tanto, el estudiante será capaz de usar operadores del mundo real para representarlos en el mundo simbólico dando la posibilidad de aplicar el conocimiento adquirido para resolver problemas de programación.

## **Justificación**.

Las matrices y arreglos son fundamentales para la programación y dan la posibilidad de que el estudiante después del trabajo pueda crear aplicaciones robustas y más eficientes a la altura de la ingeniería que aprenden.

La práctica de la programación poniendo manos a la obra permite que por medio de la práctica ejercitar el talento del estudiante mediante el enfoque práctico la educación se vuelve más disciplinada, lo que permite la mejora de las habilidades y la confianza de trabajar en un proyecto grande

## Desarrollo

### Declarar y definir arreglos.

1. Declarar un arreglo por cada uno de los tipos de datos primitivos (byte, Byte, char, Character, short, Short, int, Integer, long, Long, float, Float, double, Double, Object, String.



byte[] arregloByte; // Declarar un arreglo de bytes

Byte[] arregloByteObjeto; // Declarar un arreglo de objetos Byte

char[] arregloChar; // Declarar un arreglo de caracteres

Character[] arregloCharObjeto; // Declarar un arreglo de objetos Character

short[] arregloShort; // Declarar un arreglo de shorts

Short[] arregloShortObjeto; // Declarar un arreglo de objetos Short

int[] arregloInt; // Declarar un arreglo de enteros

Integer[] arregloIntObjeto; // Declarar un arreglo de objetos Integer

long[] arregloLong; // Declarar un arreglo de longs

Long[] arregloLongObjeto; // Declarar un arreglo de objetos Long

float[] arregloFloat; // Declarar un arreglo de floats

Float[] arregloFloatObjeto; // Declarar un arreglo de objetos Float

double[] arregloDouble; // Declarar un arreglo de doubles

Double[] arregloDoubleObjeto; // Declarar un arreglo de objetos Double

Object[] arregloObject; // Declarar un arreglo de objetos

String[] arregloString; // Declarar un arreglo de cadenas de texto



1. Definir cada uno de los arreglos anteriores con tamaño de 5 elementos.

arregloByte = new byte[5]; // Definir un arreglo de bytes

arregloByteObjeto = new Byte[5]; // Definir un arreglo de objetos Byte

arregloChar = new char[5]; // Definir un arreglo de caracteres

arregloCharObjeto = new Character[5]; // Definir un arreglo de objetos Character

arregloShort = new short[5]; // Definir un arreglo de shorts

arregloShortObjeto = new Short[5]; // Definir un arreglo de objetos Short

arregloInt = new int[5]; // Definir un arreglo de enteros

arregloIntObjeto = new Integer[5]; // Definir un arreglo de objetos Integer

arregloLong = new long[5]; // Definir un arreglo de longs

arregloLongObjeto = new Long[5]; // Definir un arreglo de objetos Long

arregloFloat = new float[5]; // Definir un arreglo de floats

arregloFloatObjeto = new Float[5]; // Definir un arreglo de objetos Float

arregloDouble = new double[5]; // Definir un arreglo de doubles

arregloDoubleObjeto = new Double[5]; // Definir un arreglo de objetos Double

arregloObject = new Object[5]; // Definir un arreglo de objetos

arregloString = new String[5]; // Definir un arreglo de cadenas de texto



1. Asignar valores a cada arreglo usando la notación de posiciones o índices



arregloByte[0] = 1;

arregloByte[1] = 2;

arregloByte[2] = 3;

arregloByte[3] = 4;

arregloByte[4] = 5;

arregloByteObjeto[0] = Byte.valueOf("1");

arregloByteObjeto[1] = Byte.valueOf("2");

arregloByteObjeto[2] = Byte.valueOf("3");

arregloByteObjeto[3] = Byte.valueOf("4");

arregloByteObjeto[4] = Byte.valueOf("5");

arregloChar[0] = 'a';

arregloChar[1] = 'b';

arregloChar[2] = 'c';

arregloChar[3] = 'd';

arregloChar[4] = 'e';

arregloCharObjeto[0] = Character.valueOf('a');

arregloCharObjeto[1] = Character.valueOf('b');

arregloCharObjeto[2] = Character.valueOf('c');

arregloCharObjeto[3] = Character.valueOf('d');

arregloCharObjeto[4] = Character.valueOf('e');

arregloShort[0] = 1;

arregloShort[1] = 2;

arregloShort[2] = 3;

arregloShort[3] = 4;

arregloShort[4] = 5;

arregloShortObjeto[0] = Short.valueOf("1");

arregloShortObjeto[1] = Short.valueOf("2");

arregloShortObjeto[2] = Short.valueOf("3");

arregloShortObjeto[3] = Short.valueOf("4");

arregloShortObjeto[4] = Short.valueOf("5");

arregloInt[0] = 1;

arregloInt[1] = 2;

arregloInt[2] = 3;

arregloInt[3] = 4;

arregloInt[4] = 5;

arregloIntObjeto[0] = Integer.valueOf("1");

arregloIntObjeto[1] = Integer.valueOf("2");

arregloIntObjeto[2] = Integer.valueOf("3");

arregloIntObjeto[3] = Integer.valueOf("4");

arregloIntObjeto[4] = Integer.valueOf("5");

arregloLong[0] = 1L;

arregloLong[1] = 2L;

arregloLong[2] = 3L;

arregloLong[3] = 4L;

arregloLong[4] = 5L;

arregloLongObjeto[0] = Long.valueOf("1");

arregloLongObjeto[1] = Long.valueOf("2");

arregloLongObjeto[2] = Long.valueOf("3");

arregloLongObjeto[3] = Long.valueOf("4");

arregloLongObjeto[4] = Long.valueOf("5");

arregloFloat[0] = 1.0f;

arregloFloat[1] = 2.0f;

arregloFloat[2] = 3.0f;

arregloFloat[3] = 4.0f;

arregloFloat[4] = 5.0f;

arregloFloatObjeto[0] = Float.valueOf("1.0");

arregloFloatObjeto[1] = Float.valueOf("2.0");

arregloFloatObjeto[2] = Float.valueOf("3.0");

arregloFloatObjeto[3] = Float.valueOf("4.0");

arregloFloatObjeto[4] = Float.valueOf("5.0");

arregloDouble[0] = 1.0;

arregloDouble[1] = 2.0;

arregloDouble[2] = 3.0;

arregloDouble[3] = 4.0;

arregloDouble[4] = 5.0;

arregloDoubleObjeto[0] = Double.valueOf("1.0");

arregloDoubleObjeto[1] = Double.valueOf("2.0");

arregloDoubleObjeto[2] = Double.valueOf("3.0");

arregloDoubleObjeto[3] = Double.valueOf("4.0");

arregloDoubleObjeto[4] = Double.valueOf("5.0");

arregloObject[0] = new Object();

arregloObject[1] = new Object();

arregloObject[2] = new Object();

arregloObject[3] = new Object();

arregloObject[4] = new Object();

arregloString[0] = "Hello";

arregloString[1] = "World";

arregloString[2] = "GitHub";

arregloString[3] = "Copilot";

arregloString[4] = "Java";

doublesObj[3] = 6.2831853072;

objects[4] = new Object();

strings[0] = "Hola mundo";



1. Mostrar los valores de cada Arreglo

 public static void main(String[] args) {

// Imprime los valores en consola

DyD\_4 objeto = new DyD\_4(); // nombre del archivo

objeto.definirArreglos();// le paso el metodo de los arreglos definidos

objeto.mostrarValores(); // le paso el método de mostrar valores

}

public void definirArreglos() {

arregloByte = new byte[5]; // Definir un arreglo de bytes

arregloByteObjeto = new Byte[5]; // Definir un arreglo de objetos Byte

arregloChar = new char[5]; // Definir un arreglo de caracteres

arregloCharObjeto = new Character[5]; // Definir un arreglo de objetos Character

arregloShort = new short[5]; // Definir un arreglo de shorts

arregloShortObjeto = new Short[5]; // Definir un arreglo de objetos Short

arregloInt = new int[5]; // Definir un arreglo de enteros

arregloIntObjeto = new Integer[5]; // Definir un arreglo de objetos Integer

arregloLong = new long[5]; // Definir un arreglo de longs

arregloLongObjeto = new Long[5]; // Definir un arreglo de objetos Long

arregloFloat = new float[5]; // Definir un arreglo de floats

arregloFloatObjeto = new Float[5]; // Definir un arreglo de objetos Float

arregloDouble = new double[5]; // Definir un arreglo de doubles

arregloDoubleObjeto = new Double[5]; // Definir un arreglo de objetos Double

arregloObject = new Object[5]; // Definir un arreglo de objetos

arregloString = new String[5]; // Definir un arreglo de cadenas de texto

}

public void mostrarValores() {

System.out.println("Arreglo de bytes");

for (int i = 0; i < arregloByte.length; i++) {

System.out.println(arregloByte[i]);

}

System.out.println("Arreglo de objetos Byte");

for (int i = 0; i < arregloByteObjeto.length; i++) {

System.out.println(arregloByteObjeto[i]);

}

System.out.println("Arreglo de caracteres");

for (int i = 0; i < arregloChar.length; i++) {

System.out.println(arregloChar[i]);

}

System.out.println("Arreglo de objetos Character");

for (int i = 0; i < arregloCharObjeto.length; i++) {

System.out.println(arregloCharObjeto[i]);

}

System.out.println("Arreglo de shorts");

for (int i = 0; i < arregloShort.length; i++) {

System.out.println(arregloShort[i]);

}

System.out.println("Arreglo de objetos Short");

for (int i = 0; i < arregloShortObjeto.length; i++) {

System.out.println(arregloShortObjeto[i]);

}

System.out.println("Arreglo de enteros");

for (int i = 0; i < arregloInt.length; i++) {

System.out.println(arregloInt[i]);

}

System.out.println("Arreglo de objetos Integer");

for (int i = 0; i < arregloIntObjeto.length; i++) {

System.out.println(arregloIntObjeto[i]);

}

System.out.println("Arreglo de longs");

for (int i = 0; i < arregloLong.length; i++) {

System.out.println(arregloLong[i]);

}

System.out.println("Arreglo de objetos Long");

for (int i = 0; i < arregloLongObjeto.length; i++) {

System.out.println(arregloLongObjeto[i]);

}

System.out.println("Arreglo de floats");

for (int i = 0; i < arregloFloat.length; i++) {

System.out.println(arregloFloat[i]);

}

System.out.println("Arreglo de objetos Float");

for (int i = 0; i < arregloFloatObjeto.length; i++) {

System.out.println(arregloFloatObjeto[i]);

}

}



1. Declarar y definir otro arreglo de tipo String en una variable llamada dataStructs, e inicializarlo con los siguientes datos, usando la sintaxis de llaves.
   1. Lista
   2. Cola
   3. Pilas
   4. Mapas
   5. Conjuntos

// Arreglo ya definido e inicializado:

String[] dataStructs = { "Lista", "Cola", "Pilas", "Mapas", "Conjuntos" };



1. Declarar y definir otro arreglo de tipo String en una variable llamada características, con 5 elementos.

// Arreglo vacío para características:

String[] caracteristicas= new String[5];



1. Ingresar los siguientes datos por teclado:
   1. Simples, Circulares y Enlazadas
   2. Primero en entrar, Primero en Salir. Último en entrar, Primero en salir
   3. Primero en entrar, Último en salir
   4. Parejas de clave y valor
   5. Elementos no repetidos

package com.example.DeclararyDefinirArreglos;

import java.util.Scanner;

public class DyD\_7 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Select data structure type

System.out.println("Seleccione el tipo de estructura de datos (1-3):");

System.out.println("1. Simples");

System.out.println("2. Circulares");

System.out.println("3. Enlazadas");

int estructuraTipoDato = scanner.nextInt();

// Selecciona la estructura y ordena si es aplicable

int tipoOrden = 0;

if (estructuraTipoDato == 1 || estructuraTipoDato == 2) {

System.out.println("Seleccione el tipo de ordenamiento (1-3):");

System.out.println("1. Primero en entrar, Primero en Salir (FIFO)");

System.out.println("2. Último en entrar, Primero en salir (LIFO)");

System.out.println("3. Primero en entrar, Último en salir (FILO)");

tipoOrden = scanner.nextInt();

}

// Estructura clave-valor si esta aplicada

int estructuraClaveValor = 0;

if (estructuraTipoDato == 3) {

System.out.println("Seleccione la estructura para pares clave-valor (1-2):");

System.out.println("1. Diccionario");

System.out.println("2. Tabla hash");

estructuraClaveValor = scanner.nextInt();

}

// Estructuras no repetidas

System.out.println("Seleccione la estructura para elementos no repetidos (1-2):");

System.out.println("1. Conjunto");

System.out.println("2. Lista ordenada");

int estructuraNoRepetida = scanner.nextInt();

// Resultado que se meustra en pantalla

System.out.println("\nResultados:");

System.out.println("Tipo de estructura de datos: " + estructuraTipoDato);

if (estructuraTipoDato == 1 || estructuraTipoDato == 2) {

System.out.println("Tipo de ordenamiento: " + tipoOrden);

}

if (estructuraTipoDato == 3) {

System.out.println("Tipo de estructura para pares clave-valor: " + estructuraClaveValor);

}

System.out.println("Tipo de estructura para elementos no repetidos: " + estructuraNoRepetida);

scanner.close();

}

}



### 2. Manipular los elementos dentro de arreglos:

1. Mostrar los elementos del arreglo *dataStructs* con su equivalente elemento del arreglo características.

 public static void main(String[] args) {

// Arreglo de dataStrucs

String[] dataStructs = { "Arreglo", "Lista", "Pila", "Cola", "Árbol", "Grafo" };

// Las características de cada estructura de datos

String[] caracteristicas = { "Estático", "Dinámico", "LIFO", "FIFO", "Jerárquico", "No lineal" };

// Revisa si ambos arreglos tienen la misma longitud

if (dataStructs.length != caracteristicas.length) {

System.out.println("Los arreglos no tienen la misma longitud");

return;

}

// Imprimir elementos de ambos arreglos

for (int i = 0; i < dataStructs.length; i++) {

System.out.println(dataStructs[i] + " es " + caracteristicas[i]);

}

}



1. Declarar un arreglo de enteros en una variable llamada ventas

// Declarar un arreglo de enteros

int[] ventas;



1. Inicializar el arreglo ventas con una cantidad aleatoria de elementos.

// Importar la clase Random

import java.util.Random;

// Crear un objeto Random

Random random = new Random();

int cantidad = random.nextInt(100);

int[ventas] = new int[cantidad];



1. Llenar el arreglo ventas con valores aleatorios entre 1 y 1000

// Llenar el arreglo ventas con valores aleatorios

for (int i = 0; i < ventas.length; i++) {

ventas[i] = random.nextInt(1000) + 1;

}



1. Mostrar la cantidad de ventas

// Mostrar la cantidad de ventas

System.out.println("Cantidad de ventas: " + ventas.length);



1. Mostrar el valor de cada una de las ventas

// Mostrar el valor de cada venta

for (int i = 0; i < ventas.length; i++) {

ventas[i] = random.nextInt(1000) + 1;

//Imprimir los valores de cada venta

System.out.println("venta # " + (i + 1) + ": " + ventas[i]);

}



1. Mostrar el total de ventas

// imprimir el total de ventas

System.out.println("Total de ventas: " + ventas.length);

// Suma de cada una de las ventas

int suma = 0;

for (int i = 0; i < ventas.length; i++) {

suma += ventas[i];

}

System.out.println("Suma de ventas: " + suma);



1. Mostrar el promedio de las ventas.



System.out.println("Promedio de ventas: " + (suma / ventas.length));



### 3. Buscar elementos en un arreglo:

1. Diseñar una función que permita buscar la venta más alta e invocar la función para probarla.

 public static int buscarVentaMasAlta(int[] ventas) {

int ventaMasAlta = 0;

for (int i = 0; i < ventas.length; i++) {

if (ventas[i] > ventaMasAlta) {

ventaMasAlta = ventas[i];

}

}

System.out.println("La venta más alta es: " + ventaMasAlta);

return ventaMasAlta;

}



1. Diseñar una función que permita buscar la venta más baja e invocar la función para probarla.

public static void buscarVentaMasBaja(int[] ventas) {

int ventaMasBaja = ventas[0];

for (int i = 1; i < ventas.length; i++) {

if (ventas[i] < ventaMasBaja) {

ventaMasBaja = ventas[i];

}

}

System.out.println("La venta más baja es: " + ventaMasBaja);

}



1. Diseñar una función que permita obtener el total de las ventas e invocar la función para probarla.

public static int obtenerTotalVentas(int[] ventas) {

int totalVentas = 0;

for (int venta : ventas) {

totalVentas += venta;

}

return totalVentas;

}

// Probar la función

int totalVentas = obtenerTotalVentas(ventas);

System.out.println("Total de ventas: " + totalVentas);



1. Diseñar una función que permita obtener el promedio de las ventas e invocar la función para probarla.

 public static void obtenerPromedioVentas(int[] ventas) {

int totalVentas = 0;

for (int i = 0; i < ventas.length; i++) {

totalVentas += ventas[i];

}

System.out.println("Promedio de ventas: " + totalVentas / ventas.length);

}

}



1. Diseñar una función que devuelva la venta cuyo valor se acerque más al valor que le sigue al promedio e invocar la función para probarla.

 public static void obtenerVentaCercanaPromedio(int[] ventas) {

int totalVentas = 0;

for (int i = 0; i < ventas.length; i++) {

totalVentas += ventas[i];

}

int promedio = totalVentas / ventas.length;

int ventaCercana = ventas[0];

int diferencia = Math.abs(ventas[0] - promedio);

for (int i = 1; i < ventas.length; i++) {

if (Math.abs(ventas[i] - promedio) < diferencia) {

ventaCercana = ventas[i];

diferencia = Math.abs(ventas[i] - promedio);

}

}

System.out.println("Venta cercana al promedio: " + ventaCercana);

}



### 4. Ordenar arreglos

1. Diseñar una función que permita ordenar las ventas de forma descendente e invocar la función para probarla.

 public static void ordenarVentasDescendente(int[] ventas) {

for (int i = 0; i < ventas.length - 1; i++) {

for (int j = i + 1; j < ventas.length; j++) {

if (ventas[i] < ventas[j]) {

int aux = ventas[i];

ventas[i] = ventas[j];

ventas[j] = aux;

}

}

}

System.out.println("Ventas ordenadas de forma descendente:");

for (int i = 0; i < ventas.length; i++) {

System.out.println(ventas[i]);

}

}



1. Diseñar una función que permita ordenamiento que permita ordenar las ventas de forma ascendente e invocar la función para probarla.

public static void ordenarVentasAscendente(int[] ventas) {

for (int i = 0; i < ventas.length - 1; i++) {

for (int j = i + 1; j < ventas.length; j++) {

if (ventas[i] > ventas[j]) {

int aux = ventas[i];

ventas[i] = ventas[j];

ventas[j] = aux;

}

}

}

System.out.println("Ventas ordenadas de forma ascendente:");

for (int i = 0; i < ventas.length; i++) {

System.out.println(ventas[i]);

}

}



1. Diseñar una función que permite desordenar las ventas e invocar la función para probarla.

 public static void desordenarVentas(int[] ventas) {

for (int i = 0; i < ventas.length - 1; i++) {

int j = (int) (Math.random() \* ventas.length);

int aux = ventas[i];

ventas[i] = ventas[j];

ventas[j] = aux;

}

System.out.println("Ventas desordenadas:");

for (int i = 0; i < ventas.length; i++) {

System.out.println(ventas[i]);

}

}



1. Diseñar una función que permita ordenar las ventas primero las partes y luego en impares e invocar la función para probarla

public static void ordenarVentasParesImpares(int[] ventas) {

int[] ventasPares = new int[ventas.length / 2];

int[] ventasImpares = new int[ventas.length - ventasPares.length];

int indicePares = 0, indiceImpares = 0;

for (int venta : ventas) {

if (venta % 2 == 0) {

ventasPares[indicePares++] = venta;

} else {

ventasImpares[indiceImpares++] = venta;

}

}

// Ordenar pares e impares

ordenarVentasAscendente(ventasPares);

ordenarVentasAscendente(ventasImpares);

// Combinar ventas

indicePares = 0;

indiceImpares = 0;

for (int i = 0; i < ventas.length; i++) {

if (i % 2 == 0) {

ventas[i] = ventasPares[indicePares++];

} else {

ventas[i] = ventasImpares[indiceImpares++];

}

}

}

// Probar la función

ordenarVentasParesImpares(ventas);

System.out.println("Ventas ordenadas pares e impares:");

for (int venta : ventas) {

System.out.println(venta);

}

### 5. Trabajar con Matrices (tablas o arreglos multidimensionales)

1. Declarar e inicializar una matriz de 5 x 8 de tipo Object en una variable llamada lenguajes

public static void declararMatriz() {

Object[][] lenguajes = { { "Java", 1, "Oracle" }, { "Python", 2, "Guido van Rossum" },

{ "JavaScript", 3, "Netscape" }, { "C", 4, "Dennis Ritchie" }, { "PHP", 5, "Rasmus Lerdorf" } };

System.out.println("Matriz de lenguajes:");

for (int i = 0; i < lenguajes.length; i++) {

for (int j = 0; j < lenguajes[i].length; j++) {

System.out.print(lenguajes[i][j] + " ");

}

System.out.println();

}



1. A la primera línea de la matriz lenguajes se debe asignar los siguientes valores en ese orden: [NOMBRE, AÑO, AUTOR, DETALLES, FRAMEWORKS]

 public static void asignarValores() {

Object[][] lenguajes = new Object[5][5];

lenguajes[0] = new Object[] { "Java", 1995, "Oracle", "Lenguaje de programación orientado a objetos",

"Spring, Hibernate" };

lenguajes[1] = new Object[] { "Python", 1991, "Guido van Rossum", "Lenguaje de programación interpretado",

"Django, Flask" };

lenguajes[2] = new Object[] { "JavaScript", 1995, "Netscape", "Lenguaje de programación interpretado",

"React, Angular" };

lenguajes[3] = new Object[] { "C", 1972, "Dennis Ritchie", "Lenguaje de programación estructurado",

"Ninguno" };

lenguajes[4] = new Object[] { "PHP", 1995, "Rasmus Lerdorf", "Lenguaje de programación interpretado",

"Laravel, Symfony" };

System.out.println("Matriz de lenguajes:");

for (int i = 0; i < lenguajes.length; i++) {

for (int j = 0; j < lenguajes[i].length; j++) {

System.out.print(lenguajes[i][j] + " ");

}

System.out.println();

}

}

}



1. Realizar una función llamada *infoLenguajes* que permita llenar la matriz lenguajes con datos por teclado a partir de la segunda fila.

 public static void infoLenguajes() {

Object[][] lenguajes = new Object[5][5];

lenguajes[0] = new Object[] { "Java", 1995, "Oracle", "Lenguaje de programación orientado a objetos",

"Spring, Hibernate" };

System.out.println("Matriz de lenguajes:");

for (int i = 0; i < lenguajes.length; i++) {

for (int j = 0; j < lenguajes[i].length; j++) {

System.out.print(lenguajes[i][j] + " ");

}

System.out.println();

}

}



1. Realizar una función que reciba como parámetro el nombre de un lenguaje y retorna verdadero si este se encuentra dentro de la matriz lenguajes, de lo contrario retorna falso.

public static boolean buscarLenguaje(String nombreLenguaje) {

Object[][] lenguajes = new Object[5][5];

lenguajes[0] = new Object[] { "Java", 1995, "Oracle", "Lenguaje de programación orientado a objetos",

"Spring, Hibernate" };

lenguajes[1] = new Object[] { "Python", 1991, "Guido van Rossum", "Lenguaje de programación interpretado",

"Django, Flask" };

lenguajes[2] = new Object[] { "JavaScript", 1995, "Netscape", "Lenguaje de programación interpretado",

"React, Angular" };

lenguajes[3] = new Object[] { "C", 1972, "Dennis Ritchie", "Lenguaje de programación estructurado", "Ninguno" };

lenguajes[4] = new Object[] { "PHP", 1995, "Rasmus Lerdorf", "Lenguaje de programación interpretado",

"Laravel, Symfony" };

for (int i = 0; i < lenguajes.length; i++) {

if (lenguajes[i][0].equals(nombreLenguaje)) {

return true;

}

}

return false;

}



1. Realizar una función que reciba como parámetro el nombre de un lenguaje y retorna las coordenadas (fila, columna) si este se encuentra encuentra dentro de la matriz lenguajes, de lo contrario retorna null.

public static int[] buscarLenguaje(String nombreLenguaje) {

Object[][] lenguajes = new Object[5][5];

lenguajes[0] = new Object[] { "Java", 1995, "Oracle", "Lenguaje de programación orientado a objetos",

"Spring, Hibernate" };

lenguajes[1] = new Object[] { "Python", 1991, "Guido van Rossum", "Lenguaje de programación interpretado",

"Django, Flask" };

lenguajes[2] = new Object[] { "JavaScript", 1995, "Netscape", "Lenguaje de programación interpretado",

"React, Angular" };

lenguajes[3] = new Object[] { "C", 1972, "Dennis Ritchie", "Lenguaje de programación estructurado", "Ninguno" };

lenguajes[4] = new Object[] { "PHP", 1995, "Rasmus Lerdorf", "Lenguaje de programación interpretado",

"Laravel, Symfony" };

for (int i = 0; i < lenguajes.length; i++) {

if (lenguajes[i][0].equals(nombreLenguaje)) {

return new int[] { i, 0 };

}

}

return null;

}



1. Realizar una función que reciba como parámetro una fila y retorne el registro completo (datos en la fila) del lenguaje en esa fila. Si la fila está errada (no existe en la matriz lenguajes) entonces, debe retornar una excepción con un mensaje informando el problema

 public static Object[] buscarLenguaje(int fila) {

Object[][] lenguajes = new Object[5][5];

lenguajes[0] = new Object[] { "Java", 1995, "Oracle", "Lenguaje de programación orientado a objetos",

"Spring, Hibernate" };

lenguajes[1] = new Object[] { "Python", 1991, "Guido van Rossum", "Lenguaje de programación interpretado",

"Django, Flask" };

lenguajes[2] = new Object[] { "JavaScript", 1995, "Netscape", "Lenguaje de programación interpretado",

"React, Angular" };

lenguajes[3] = new Object[] { "C", 1972, "Dennis Ritchie", "Lenguaje de programación estructurado", "Ninguno" };

lenguajes[4] = new Object[] { "PHP", 1995, "Rasmus Lerdorf", "Lenguaje de programación interpretado",

"Laravel, Symfony" };

if (fila < 0 || fila >= lenguajes.length) {

throw new RuntimeException("La fila no existe en la matriz lenguajes");

}

return lenguajes[fila];

}



1. Realizar una función que reciba como parámetro una columna y retorne un todos los datos en esa columna. Si la columna está errada (no existe en la matriz lenguajes) entonces, debe retornar una excepción con un mensaje informando el problema.

 public static Object[] buscarColumna(int columna) {

Object[][] lenguajes = new Object[5][5];

lenguajes[0] = new Object[] { "Java", 1995, "Oracle", "Lenguaje de programación orientado a objetos",

"Spring, Hibernate" };

lenguajes[1] = new Object[] { "Python", 1991, "Guido van Rossum", "Lenguaje de programación interpretado",

"Django, Flask" };

lenguajes[2] = new Object[] { "JavaScript", 1995, "Netscape", "Lenguaje de programación interpretado",

"React, Angular" };

lenguajes[3] = new Object[] { "C", 1972, "Dennis Ritchie", "Lenguaje de programación estructurado", "Ninguno" };

lenguajes[4] = new Object[] { "PHP", 1995, "Rasmus Lerdorf", "Lenguaje de programación interpretado",

"Laravel, Symfony" };

if (columna < 0 || columna >= lenguajes[0].length) {

throw new RuntimeException("La columna no existe en la matriz lenguajes");

}

Object[] datosColumna = new Object[lenguajes.length];

for (int i = 0; i < lenguajes.length; i++) {

datosColumna[i] = lenguajes[i][columna];

}

return datosColumna;



1. Realizar una función que reciba como parámetro las coordenadas (fila, columna) y retorne el valor en esa coordinada y el registro (fila) completo con ojos datos de ese lenguaje, si las coordinadas son erradas (no existen en la matriz lenguajes) entonces, debe retornar una excepción con un mensaje informando el problema.

 public static Object[] buscarLenguaje(int fila, int columna) {

Object[][] lenguajes = new Object[5][5];

lenguajes[0] = new Object[] { "Java", 1995, "Oracle", "Lenguaje de programación orientado a objetos",

"Spring, Hibernate" };

lenguajes[1] = new Object[] { "Python", 1991, "Guido van Rossum", "Lenguaje de programación interpretado",

"Django, Flask" };

lenguajes[2] = new Object[] { "JavaScript", 1995, "Netscape", "Lenguaje de programación interpretado",

"React, Angular" };

lenguajes[3] = new Object[] { "C", 1972, "Dennis Ritchie", "Lenguaje de programación estructurado", "Ninguno" };

lenguajes[4] = new Object[] { "PHP", 1995, "Rasmus Lerdorf", "Lenguaje de programación interpretado",

"Laravel, Symfony" };

if (fila < 0 || fila >= lenguajes.length || columna < 0 || columna >= lenguajes[0].length) {

throw new RuntimeException("Las coordenadas no existen en la matriz lenguajes");

}

return new Object[] { lenguajes[fila][columna], lenguajes[fila] };

}



1. lenguaje y retorna las coordenadas (fila, columna) si este se encuentra dentro de la matriz lenguajes, de lo contrario retorna null.

public static int[] buscarLenguaje(Object[][] lenguajes, String lenguaje) {

for (int i = 0; i < lenguajes.length; i++) {

for (int j = 0; j < lenguajes[0].length; j++) {

if (lenguajes[i][0].equals(lenguaje)) {

return new int[]{i, j};

}

}

}

return null;

}

// Ejemplo de uso

String lenguaje = "Python";

int[] coordenadas = buscarLenguaje(lenguajes, lenguaje);

if (coordenadas != null) {

System.out.println("Lenguaje " + lenguaje + " encontrado en la coordenada (" + coordenadas[0] + ", " + coordenadas[1] + ")");

} else {

System.out.println("Lenguaje " + lenguaje + " no encontrado");

}



1. Realizar una función que permita mostrar los datos de la matriz lenguajes según la siguiente logia:
   1. La función de debe recibir como parámetro un arreglo con 5 elementos de tipo boolean
   2. Al invocar la función se debe mostrar solo la información de la matriz lenguajes de acuerdo al valor (false o true) que tenga el arreglo que recibe como parámetro. Por ejemplo:

public static int[] buscarLenguaje(String lenguaje) {

Object[][] lenguajes = new Object[5][5];

lenguajes[0] = new Object[] { "Java", 1995, "Oracle", "Lenguaje de programación orientado a objetos",

"Spring, Hibernate" };

lenguajes[1] = new Object[] { "Python", 1991, "Guido van Rossum", "Lenguaje de programación interpretado",

"Django, Flask" };

lenguajes[2] = new Object[] { "JavaScript", 1995, "Netscape", "Lenguaje de programación interpretado",

"React, Angular" };

lenguajes[3] = new Object[] { "C", 1972, "Dennis Ritchie", "Lenguaje de programación estructurado", "Ninguno" };

lenguajes[4] = new Object[] { "PHP", 1995, "Rasmus Lerdorf", "Lenguaje de programación interpretado",

"Laravel, Symfony" };

for (int i = 0; i < lenguajes.length; i++) {

if (lenguajes[i][0].equals(lenguaje)) {

return new int[] { i, 0 };

}

}

return null;

}

Debe mostrar los nombres de los lenguajes, los autores, los frameworks

 public static void mostrarLenguajes(boolean[] mostrar) {

Object[][] lenguajes = new Object[5][5];

lenguajes[0] = new Object[] { "Java", 1995, "Oracle", "Lenguaje de programación orientado a objetos",

"Spring, Hibernate" };

lenguajes[1] = new Object[] { "Python", 1991, "Guido van Rossum", "Lenguaje de programación interpretado",

"Django, Flask" };

lenguajes[2] = new Object[] { "JavaScript", 1995, "Netscape", "Lenguaje de programación interpretado",

"React, Angular" };

lenguajes[3] = new Object[] { "C", 1972, "Dennis Ritchie", "Lenguaje de programación estructurado", "Ninguno" };

lenguajes[4] = new Object[] { "PHP", 1995, "Rasmus Lerdorf", "Lenguaje de programación interpretado",

"Laravel, Symfony" };

for (int i = 0; i < lenguajes.length; i++) {

if (mostrar[i]) {

System.out.println("Lenguaje: " + lenguajes[i][0]);

System.out.println("Año de creación: " + lenguajes[i][1]);

System.out.println("Creador: " + lenguajes[i][2]);

System.out.println("Descripción: " + lenguajes[i][3]);

System.out.println("Frameworks: " + lenguajes[i][4]);

System.out.println();

}

}

}



[REPOSITORIO EN GITHUB](https://github.com/herculeskan/estructuraDeDatos_Macabritos_2024)