****

Estructuras de Datos

Presentado por

Luis David Palacio Diaz

7502210079

Docente

Jhon Arrieta Arrieta

Universidad de Cartagena

Facultad de Ingeniería – Ingeniería de Software

III Semestre – Cartagena

2023

TABLA DE CONTENIDO

[**Introducción** 6](#_Toc133177582)

[**Objetivos** 7](#_Toc133177583)

[**Justificación** 8](#_Toc133177584)

[**DESARROLLO DE LA ACTIVIDAD:** 9](#_Toc133177585)

[**PUNTO 1** 10](#_Toc133177586)

[Declarar y definir arreglos: 10](#_Toc133177587)

[**a) Declaración de arreglos de los tipos de datos primitivos:** 11](#_Toc133177588)

[**b) Definir cada uno de los arreglos anteriores con tamaño de 5 elementos.** 11](#_Toc133177589)

[**c) Asignar valores a cada arreglo usando la notación de posiciones o índices** 12](#_Toc133177590)

[**d) Mostrar los valores de cada Arreglo** 12](#_Toc133177591)

[**e) Declarar y definir otro arreglo de tipo String en una variable llamada dataStructs, e inicializarlo con los siguientes datos, usando la sintaxis de llaves.** 20](#_Toc133177592)

[**f) Declarar y definir otro arreglo de tipo String en una variable llamada características, con 5 elementos.** 21](#_Toc133177593)

[**g) Ingresar los siguientes datos por teclado:** 22](#_Toc133177594)

[**PUNTO 2** 24](#_Toc133177596)

[Manipular los elementos dentro de arreglos: 24](#_Toc133177597)

[**a) Mostrar los elementos del arreglo dataStructs con su equivalente elemento del arreglo características.** 24](#_Toc133177598)

[**b) Declarar un arreglo de enteros en una variable llamada ventas**. 25](#_Toc133177599)

[**c) Inicializar el arreglo ventas con una cantidad aleatoria de elementos.** 25](#_Toc133177600)

[**d) Llenar el arreglo ventas con valores aleatorios entre 1 y 1000**. 26](#_Toc133177601)

[**e) Mostrar la cantidad de ventas** 26](#_Toc133177602)

[**f) Mostrar el valor de cada una de las ventas** 26](#_Toc133177603)

[**g) Mostrar el total de ventas** 27](#_Toc133177604)

[**h) Mostrar el promedio de las ventas**. 29](#_Toc133177605)

[**PUNTO 3** 30](#_Toc133177606)

[Buscar elementos en un arreglo: 30](#_Toc133177607)

[**a) Diseñar una función que permita buscar la venta más alta e invocar la función para probarla.** 30](#_Toc133177608)

[**b) Diseñar una función que permita buscar la venta más baja e invocar la función para probarla**. 32](#_Toc133177609)

[**c) Diseñar una función que permita obtener el total de las ventas e invocar la función para probarla**. 34](#_Toc133177610)

[**d) Diseñar una función que permita obtener el promedio de las ventas e invocar la función para probarla.** 35](#_Toc133177611)

[**e) 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**. 36](#_Toc133177612)

[**PUNTO 4** 38](#_Toc133177613)

[Ordenar arreglos: 38](#_Toc133177614)

[**a) Diseñar una función que permita ordenar las ventas de forma descendente e invocar la función para probarla.** 38](#_Toc133177615)

[**b) Diseñar una función que permita ordenamiento que permita ordenar las ventas de forma ascendente e invocar la función para probarla.** 38](#_Toc133177616)

[**c) Diseñar una función que permite desordenar las ventas e invocar la función para probarla.** 38](#_Toc133177617)

[**d) Diseñar una función que permita ordenar las ventas primero las partes y luego en impares e invocar la función para probarla.** 38](#_Toc133177618)

[**PUNTO 5** 39](#_Toc133177619)

[Trabajar con Matrices (tablas o arreglos multidimensionales) 39](#_Toc133177620)

[**a) Declarar e inicializar una matriz de 5 x 8 de tipo Object en una variable llamada lenguajes** 39](#_Toc133177621)

[**b) A la primera línea de la matriz lenguajes se debe asignar los siguientes valores en ese orden:** 39](#_Toc133177622)

[**[NOMBRE, AÑO, AUTOR, DETALLES, FRAMEWORKS]** 39](#_Toc133177623)

[**c) Realizar una función llamada infoLenguajes que permita llenar la matriz lenguajes con datos por teclado a partir de la segunda fila.** 39](#_Toc133177624)

[**d) 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.** 39](#_Toc133177625)

[**e) Realizar una función que reciba como parámetro el nombre de un lenguaje y retorna las coordenadas (fila, columna) si este se encuentra dentro de la matriz lenguajes, de lo contrario retorna null.** 39](#_Toc133177626)

[**f) 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.** 39](#_Toc133177627)

[**g) Realizar una función que reciba como parámetro una columna y retorne 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.** 39](#_Toc133177628)

[**h) 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.** 39](#_Toc133177629)

[**i) lenguaje y retorna las coordenadas (fila, columna) si este se encuentra dentro de la matriz lenguajes, de lo contrario retorna null.** 40](#_Toc133177630)

[**j) Realizar una función que permita mostrar los datos de la matriz lenguajes según la siguiente lógica:** 40](#_Toc133177631)

[i) La función de debe recibir como parámetro un arreglo con 5 elementos de tipo boolean 40](#_Toc133177632)

[ii) 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: 40](#_Toc133177633)

## **Introducción**

En este trabajo se profundiza sobre los arreglos unidimensionales y bidimensionales en Java, sus características, ventajas y desventajas, y su uso en la programación.

Los arreglos son estructuras de datos esenciales que permiten almacenar y manipular conjuntos de elementos del mismo tipo. En Java, los arreglos unidimensionales y bidimensionales son comunes para almacenar y acceder a datos secuenciales y datos organizados en una tabla o matriz, respectivamente. Los arreglos unidimensionales son lineales y contienen elementos indexados de forma consecutiva y ordenada, mientras que los arreglos bidimensionales contienen filas y columnas indexadas por dos valores: índice de fila y columna. En el mundo de la programación, los arreglos son ampliamente utilizados debido a su estructura y eficiencia.

## **Objetivos**

Investigar sobre el rendimiento y la eficiencia de los arreglos unidimensionales y bidimensionales en Java, en términos de tiempo y espacio de almacenamiento, para determinar cuál estructura de datos es más adecuada para diferentes tipos de aplicaciones en el desarrollo de software, además debemos investigar sobre como aplicar técnicas de manipulación de arreglos unidimensionales y bidimensionales en Java, como la búsqueda, ordenamiento y filtrado de datos.

## **Justificación**

Los arreglos unidimensionales y bidimensionales son estructuras de datos esenciales en el mundo de la programación, por lo que es importante conocer sus características y aplicaciones, además de saber elegir estructura de datos adecuada, esto influye mucho ya que afecta de manera directa en el rendimiento y eficiencia del software, lo que puede traducirse en ahorro de recursos y sobre todo tiempo, al investigar mas y realizar comparaciones sobre ambas estructuras se podrá determinar que es mejor en los diferentes tipos de problemas

Los arreglos son una base fundamental en el aprendizaje del area de programación, por lo que el adquirir conocimiento en este trabajo puede ser útil para la formación de futuros profesionales de la ingeniería de software.

# **DESARROLLO DE LA ACTIVIDAD:**

Esta actividad evaluativa tiene como objetivo evidenciar el nivel de estudio y práctica que usted ha realizado sobre los fundamentos de Arreglos y Algoritmos de Ordenamiento y algoritmos de búsqueda.

Usted debe desarrollar y documentar detalladamente los siguientes puntos, cada respuesta debe ir acompañada de su respectiva explicación y ejemplo en Java.

# **PUNTO 1**

# Declarar y definir arreglos:

## **a) Declarar un arreglo por cada uno de los tipos de datos primitivos**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Byte** | **Char** | **Short** | **Int** | **Long** |
| **Float** | **Double** | **Object** | **String** |  |

En este punto, discutiremos sobre **arreglos**, Después de investigar, leer y debatir en clases sobre este tema, hemos concluido que en el lenguaje de programación java un array (arreglo) es una colección de datos del mismo tipo.

Después de haber realizado varias investigaciones sobre el tema, hemos evidenciado que un array se compone de elementos como el tipo de dato, tamaño, elemento y los índices.

En este punto **a** se nos habla sobre la necesidad de realizar arreglos con tipos de datos primitivos y por tal razón deberíamos utilizar solo los tipos de datos resaltados en negro (Datos primitivos), ya que los otros 2 son tipos de dato objeto; sabemos que **Object** representa la base principal de las clases que utilizamos en java, mientras que un String se puede decir que es una subclase de Object.

Con el conocimiento que tenemos podemos concluir que un String es un objeto y es representado con caracteres Unicode, pero debemos tener mucho cuidado, ya que al crear un objeto String no podemos modificar el contenido del arreglo ya que cualquier tipo de situación que incluya una modificación en este lo que en realidad puede suceder es que creara una nueva cadena, por lo tanto, se dice que los objetos String son estáticos o inmodificables.

**Arrays Unidimensionales**

Estos son arreglos de una sola dimensión, es decir de una sola línea y permiten guardar los elementos en una manera ordenada uno al lado del otro en la memoria.

**Punto a**

Declarar un arreglo por cada uno de los tipos de datos primitivos

Para declarar un arreglo se puede realizar de la siguiente manera:

**Tipo\_de\_variable** [ ] **Nombre\_del\_array** = new **Tipo\_de\_variable** [**dimensiòn**];

Un breve ejemplo para explicarlo mejor puede ser este:

**int** [ ] **numerosEnteros** = new **int** [ **5** ] ;

* **int:** Este hace referencia al tipo de variable (double, float, String, etc…) que se desea usar en nuestro arreglo. Ejemplo si se declara una variable del tipo int [ ] solo podremos asignar valores del tipo int en esa respectiva variable.
* **numerosEnteros:** Hace referencia a la asignación de nuestro arreglo, le podemos colocar el nombre que deseemos, es recomendable siempre utilizar el estilo Camelcase, como el ejemplo que se presenta a continuación: numerosEnteros, edadesMenores, animalesAcuaticos, esto se hace por muchos motivos, pero uno de ellos es utilizar buenas practicas en el código.
* **[ 5 ]:** Este espacio ocupado por el numero 5 le llamamos dimensión, ya que este hace alusión a la cantidad de valores que necesitamos ingresar para nuestro array, en este caso le asignamos la cantidad de 5 valores pero en realidad le podemos ingresar los que sean necesarios, normalmente siempre se le pide al usuario por consola la cantidad de datos a ingresar.

## **a) Declaración de arreglos de los tipos de datos primitivos:**

## **b) Definir cada uno de los arreglos anteriores con tamaño de 5 elementos.**

A continuación, se presenta la declaración de cada tipo de arreglo con sus respectivos datos primitivos y adicional se define la dimensión o tamaño de los elementos que en este caso nos indica que sean de 5 elementos.

Se procedió hacer directamente desde Word y no una captura de pantalla para que sea vea mejor estéticamente, más adelante se realizaran capturas del código en NetBeans.

* **Arreglo de bytes:**
* byte [ ] fechasDePago = new byte [5];
* **Arreglo de caracteres:**
* char [ ] inicialesDeNombresMasUsados = new char [5];
* **Arreglo de números cortos:**
* short [ ] numerosMasJugadosDeLoterias = new short [5];
* **Arreglo de enteros:**
* int [ ] NumerosEnteros = new int [5]; //Arreglo de enteros.
* **Arreglo de números largos.**
* long [ ] numerosDeCuentasEmpleados = new long [5];
* **Arreglo de números flotantes**
* float [ ] temperatura\_en\_fahrenheit = new float [5];
* **Arreglo de números dobles**
* double [ ] coordenadasTopograficas = new double [5];
* **Arreglo de tipo objeto**
* Object [ ] deportistasFamosos = new Object [5];
* **Arreglo de tipo cadenas de texto**
* String [ ] animalesAcuaticos = new String [5];

## **c) Asignar valores a cada arreglo usando la notación de posiciones o índices**

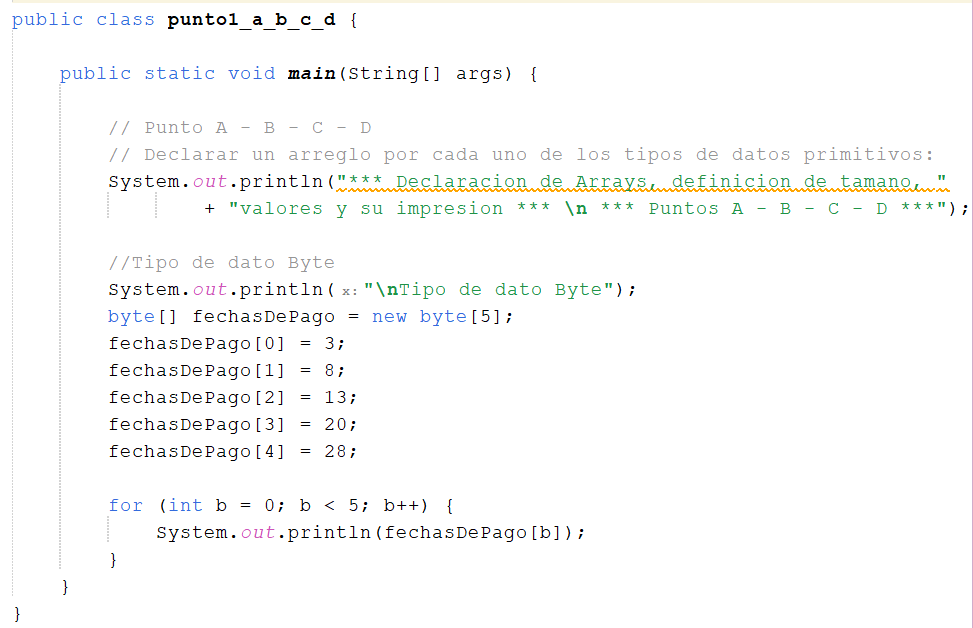
## **d) Mostrar los valores de cada Arreglo**

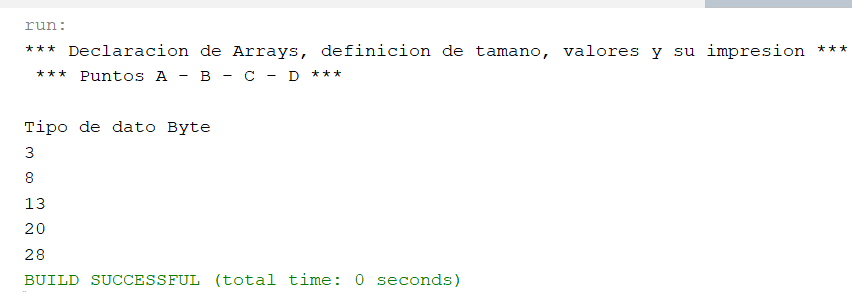
A continuación, se anexan diferentes capturas del código donde se aprecia por cada tipo de dato los valores que se le asigno a cada arreglo, adicional se hará una explicación de lo que se está realizando en el código.

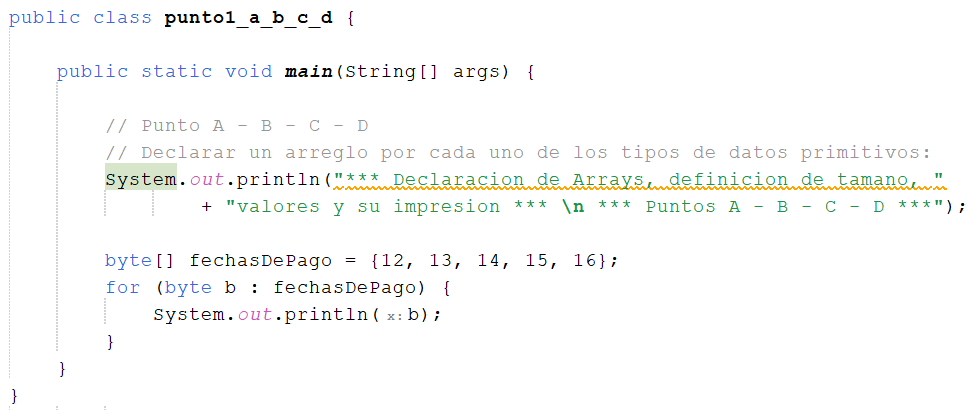
Para este primer tipo de dato Byte, hemos creado un arreglo con el nombre fechasDePago, este tiene una dimensión de 5 elementos, luego podemos realizar la asignación de valores para los 5 elementos ya que es lo que nos solicita el enunciado.

En cada una de las líneas de código desde el índice 0 hasta el índice 4 se le han asignado valores de tipo byte, seguido de este paso se nos solicita el mostrar o imprimir los valores de cada arreglo para lo cual hemos empleado un bucle llamado **for** el cual utilizamos para recorrer cada una de las posiciones de nuestro arreglo, esto realizara el recorrido desde el índice 0 hasta el último índice de la dimensión del arreglo, sabemos que la variable de control “b” se inicializa en cero ya que los índice de los arreglos siempre empiezan en 0 siendo este quien representa el primer elemento de este arreglo, cabe destacar que la expresión **b < 5;** es una condición que hace parte del bucle la cual nos está indicando que en cada iteración el bucle va continuar mientras que nuestro variable “**b**” sea menor a 5, la expresión “**b++”** incrementara cada vez su valor en 1 unidad después de cada repetición en el bloque, apenas se cumpla la condición de que llegue a 5 el bucle finalizara de no ser así continuara.

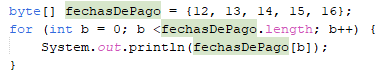
**TIPO DE DATO BYTE**



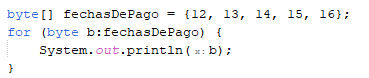
**Impresión en consola:**

Adicional a la manera mencionada anteriormente se puede asignar valores en una misma línea y luego para imprimir los valores se imprimiría con el anterior bucle de **for** los valores fila a fila, como podemos apreciar en la siguiente imagen.

Podemos observar también de esta otra manera, en la cual cuando no sabemos la cantidad de valores que trae nuestro arreglo podemos utilizar el nombre del arreglo seguido de **.length** para que el arroje absolutamente todo lo que este contenido dentro del arreglo, esto se hace especialmente cuando vemos arreglos muy larguísimos y seria tedioso imprimir 1 por 1, este bucle se ejecutara la cantidad de veces necesarias hasta asegurarse que ya no tiene más valores que mostrar.

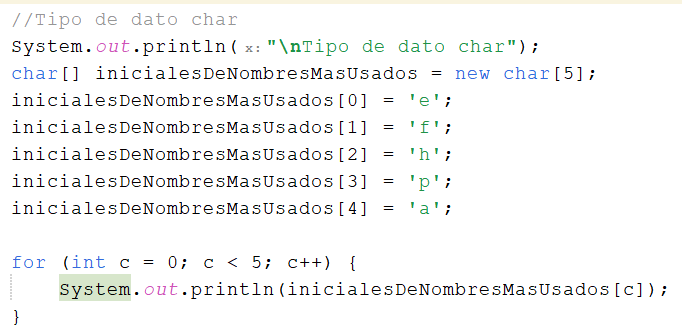


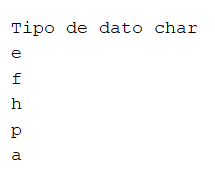
Otra manera de imprimir los valores es mediante el bucle **for-each**, normalmente se suele utilizar para hacer el recorrido de todos los elementos de un array, esto manera de imprimir el arreglo es de las más simples para un código limpio y fácil de usar.

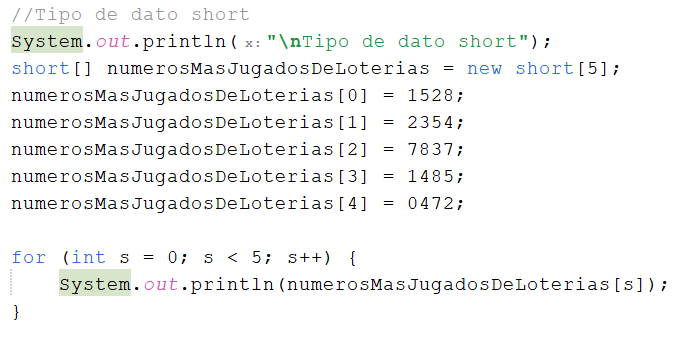


A continuación, se anexa captura de pantalla de los demás tipos de datos y su impresión en consola, solo se realiza la explicación con el primer tipo de dato ya que básicamente estaríamos trabajando con lo mismo en los demás tipos de datos, lo único que puede variar es el propio tipo de dato o la manera de escribir nuestro código.

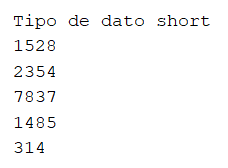
**TIPO DE DATO CHAR**

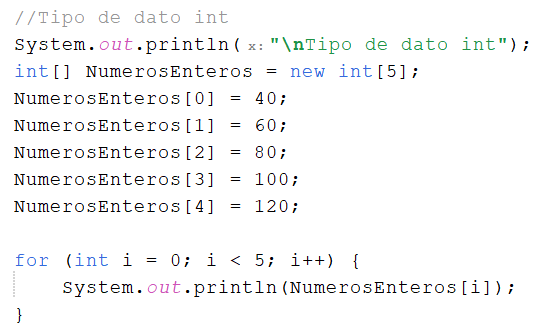


 **Impresión en consola:**

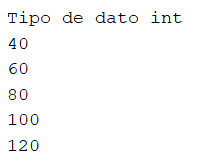
 **TIPO DE DATO SHORT**

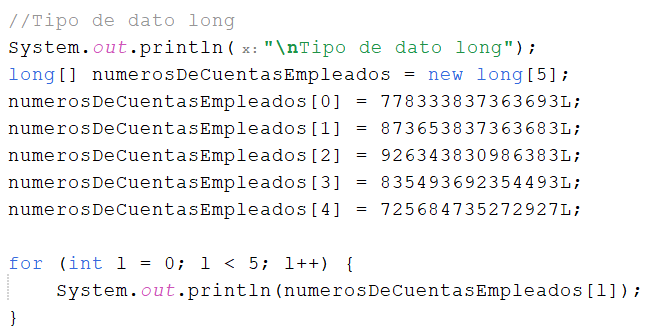
**Impresión en consola:**



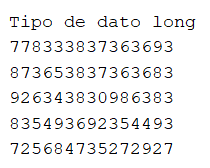
**TIPO DE DATO INT**

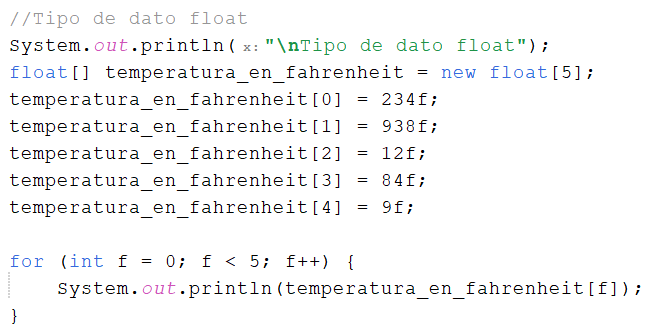
**Impresión en consola:**



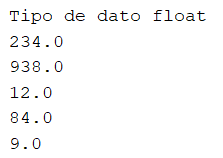
**TIPO DE DATO LONG**

**Impresión en consola:**

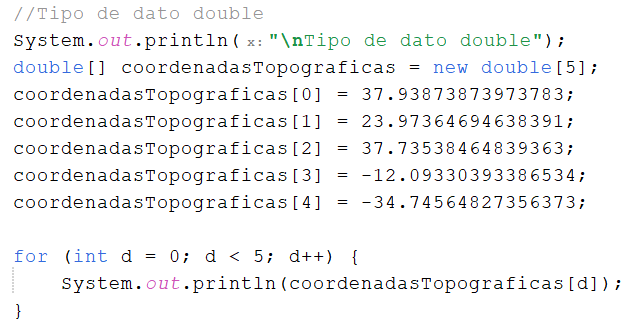


**TIPO DE DATO FLOAT**

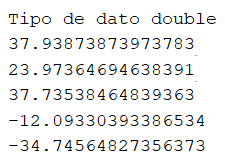
**Impresión en consola:**



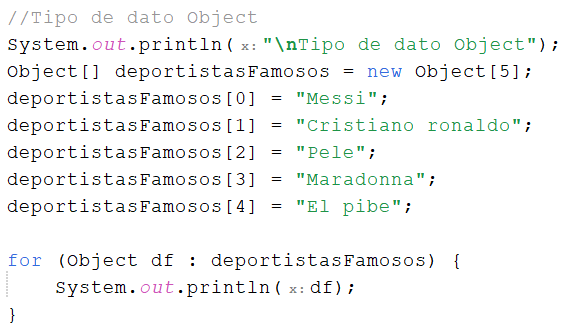
**TIPO DE DATO DOUBLE**



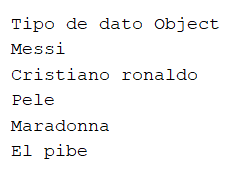
**Impresión en consola:**



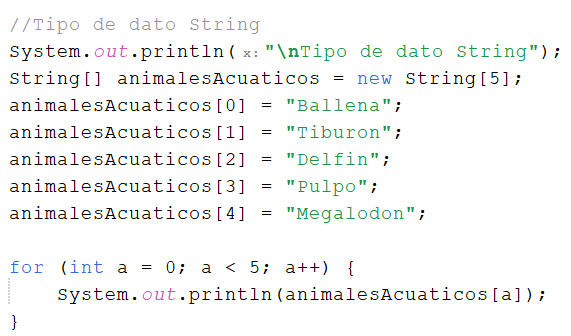
**TIPO DE DATO OBJECT**



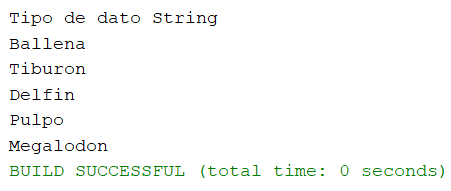
**Impresión en consola:**



**TIPO DE DATO STRING**



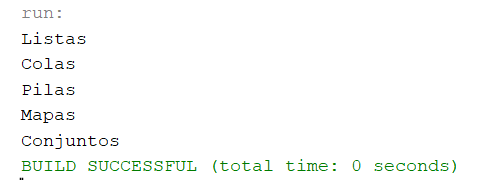
**Impresión en consola:**



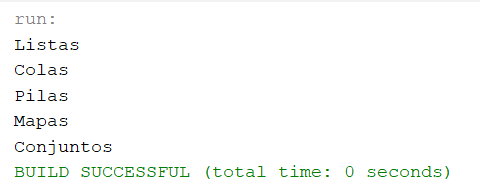
## **e) Declarar y definir otro arreglo de tipo String en una variable llamada dataStructs, e inicializarlo con los siguientes datos, usando la sintaxis de llaves.**

* + 1. Listas
    2. Colas
    3. Pilas
    4. Mapas
    5. Conjuntos



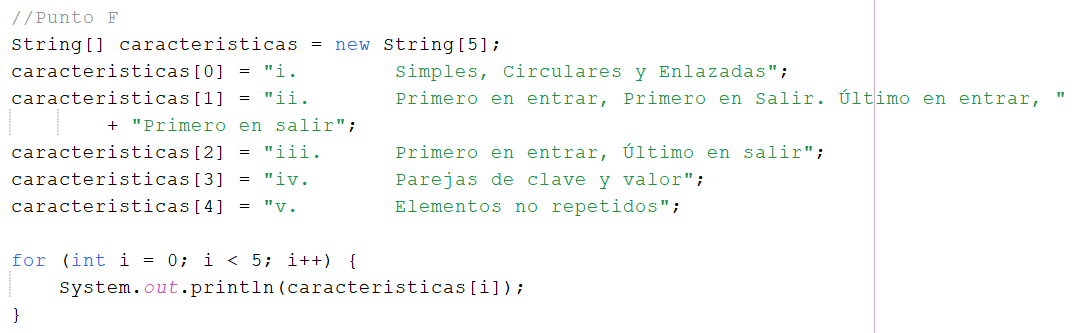


**Impresión en consola:**

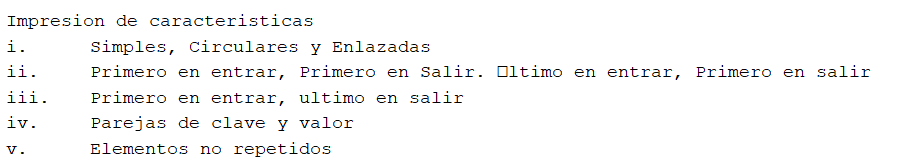


En el anterior código hemos definido un arreglo de tipo cadena llamado dataStructs y lo hemos inicializado con los respectivos valores que se indican en el punto **e** mediante la sintaxis de llave, en este caso de todas las formas para imprimir los valores se me ha hecho más fácil utilizar un ciclo for-each, de esta manera puedo recorrer todo el arreglo **e** imprimir los valores que contiene sin necesidad de hacerlo por medio del índice o de algún otro método, este en lo personal me parece muy útil y sencillo.

## **f) Declarar y definir otro arreglo de tipo String en una variable llamada características, con 5 elementos.**



**Impresión en consola:**



Aquí en este código hemos definido un arreglo de cadenas al cual le hemos asignador el nombre de características como se pide en el enunciado, hemos ingresado 5 valores en cada índice, sabemos que el tamaño del arreglo es de una dimensión de 5 y hemos utilizado el bucle **for** para poder recorrer el arreglo e imprimir sus valores, como hemos utilizado antes este bucle sabemos que ira recorriendo cada posición desde el índice 0 y se ira ejecutando hasta que el iterador sea menor que su dimensión, por lo tanto ira imprimiendo cada valor hasta finalizar el bucle.

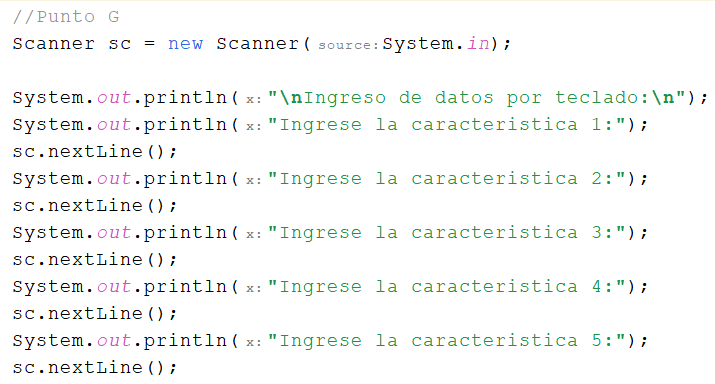
## **g) Ingresar los siguientes datos por teclado:**

* + 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

En este código se nos solicita que ingresemos unos valores por teclado, para la cual en este caso hemos ingresado los valores asignados anteriormente, en este ocasión utilizaremos el método **sc.nextLine()** de la clase Scanner con lo cual al ingresarlo en cada línea del código haremos que se solicite un valor a ingresar y este método esperara a que se ingresen los respectivos valores e ira avanzando cuando presionemos la tecla enter, así de esta manera devuelve la cadena que hemos ingresado para cada una de estas.

Anexo se encuentran capturas del código.

**CAPTURA DE PANTALLA DEL CODIGO DEL PUNTO G**



**Impresión en consola:**

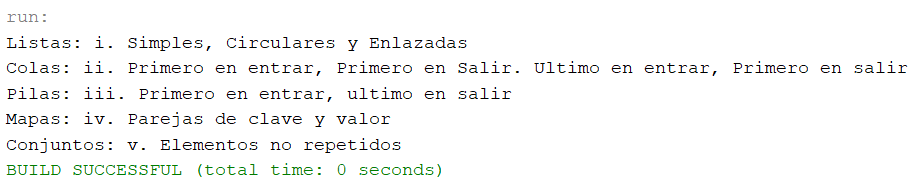
# 

# **PUNTO 2**

# Manipular los elementos dentro de arreglos:

## **a) Mostrar los elementos del arreglo dataStructs con su equivalente elemento del arreglo características.**

En este código declaramos el arreglo **dataStructs** con 5 elementos y hemos declarado otro arreglo de tipo **String** con el nombre características de igual cantidad de elementos que el primer arreglo y solo cambiando sus valores, para imprimir los valores de los arreglos **dataStruct**s y característica hemos optado por usar un bucle for para hacer el recorrido de los dos arreglos e imprimir por consola ambos valores, en este caso podemos utilizar una concatenación dentro del bucle con el operador +, de esta manera lograremos imprimir el arreglo de **dataStruct** y característica en una sola línea de código, eso es básicamente lo que se hace en esta sección del código.

**Impresión en consola:**

Como se puede apreciar vemos ambos arreglos en una sola línea de código.

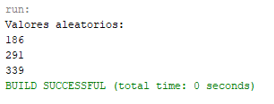
## **b) Declarar un arreglo de enteros en una variable llamada ventas**.

En este código hemos declarado una variable llamada ventas, en la cual hemos declarado un arreglo de enteros y que aun no tiene una dimensión asignada.

## **c) Inicializar el arreglo ventas con una cantidad aleatoria de elementos.**

Después de crear la variable ventas, se utiliza la clase random para generar un numero aleatorio entre 1 y 17 que indica la cantidad de eleementos que tendrá el arreglo, adicional se agrega 1 al resultado del método nextInt para asegurarnos que el arreglo nunca se nos quede vacío, adicional hemos creado un ciclo for para crear valores aleatorios entre 0 y 1000 a cada elemento del arreglo, y por último pasamos a imprimir todos los valores que se generaron aleatoriamente.

## **d) Llenar el arreglo ventas con valores aleatorios entre 1 y 1000**.

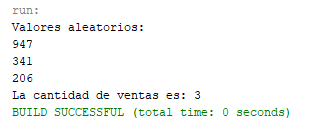
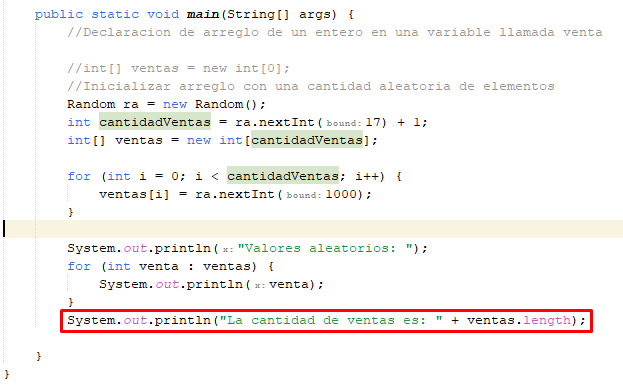
Como podemos apreciar hemos creado un ciclo for en el cual recorremos el arreglo de ventas y inicializamos cada elemento del arreglo con valores aleatorios como se pide el enunciado, los rangos irán desde el valor 0 hasta el valor 999, todo esto lo logramos con la clase random que anteriormente habíamos mencionado, luego utilizamos el ciclo for-each para recorrer el arreglo e imprimir cada uno de los elementos aleatorios en consola, anexo una pequeña captura de la impresión en consola.

## **e) Mostrar la cantidad de ventas**

## **f) Mostrar el valor de cada una de las ventas**

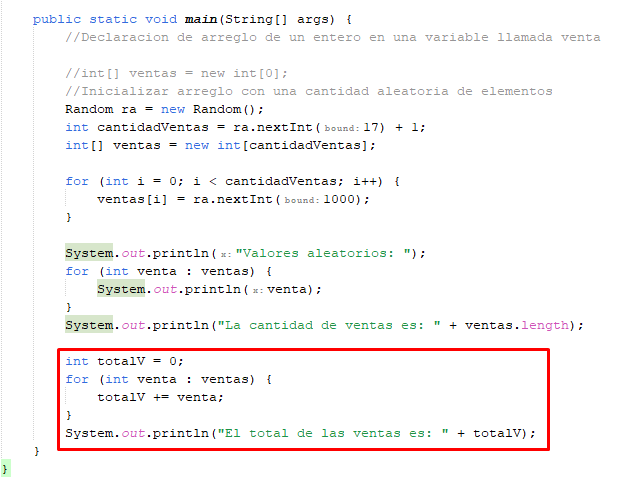
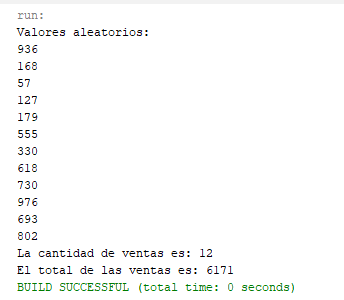
Con la siguiente línea de código hemos pasado a mostrar la cantidad total de ventas mediante la propiedad **length** para imprimir el total de elementos que tiene el arreglo ventas, como sabemos la propiedad length sirve para contar la cantidad de elementos que trae nuestro arreglo y nos regresa el valor mediante un numero entero, anexo imagen del resultado en consola.

Los valores de las ventas se están imprimiendo siempre en consola:

****

## **g) Mostrar el total de ventas**

En este código estamos calculando la sumatoria de todas las ventas de nuestro arreglo, para lo cual hemos utilizado una variable llamada **totalV** haciendo alusión al total de las ventas; nuestra variable la hemos inicializado en “cero” y luego hemos utilizado el ciclo **for-each** para recorrer nuestro arreglo de ventas e ir sumando cada uno de los valores aleatorios de nuestro arreglo ventas mediante nuestro operador += y luego pasárselos a la variable **totalV**, para entenderlo mejor se sabe que en cada iteración de nuestro ciclo se tomara el valor de la venta y se ira agregando a nuestra variable que calculara el total de la venta (**totalV**) lo que resultara al final en la sumatoria de todas nuestras ventas en el arreglo, anexo se deja imagen de la impresión en consola.

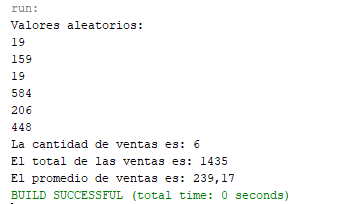


## **h) Mostrar el promedio de las ventas**.

En este punto se nos pide mostrar el promedio de ventas, en matemáticas lo mas fácil seria hacer la sumatoria de los valores y el resultado total dividirlo entre la cantidad de elementos que tenemos, en el lenguaje de programación java lo podemos realizar o resolver de la siguiente manera.

En la primera línea de código (resaltado en rojo), declaramos la variable promVentas para almacenar el promedio de ventas, donde totalV contiene la suma total de todos los elementos de nuestro arreglo y ventas.length contiene la cantidad total de nuestros elementos en el arreglo ventas, en nuestro caso le asignamos máximo un valor de 17 elementos.

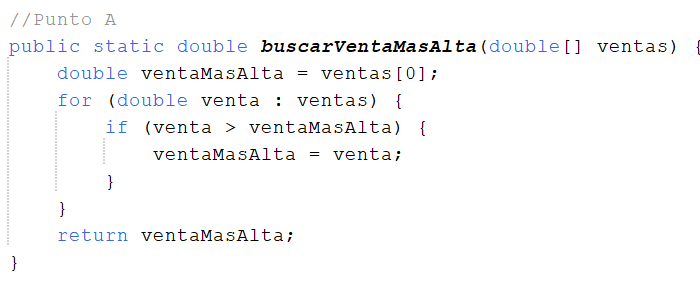
Para pasar a calcular lo hacemos tipo matemáticas, hacemos una división y utilizamos la variable totalV (que contiene la sumatoria de todas las ventas) entre ventas.length que contiene la cantidad de 1 a 17 elementos y como en la división casi siempre se obtienen números decimales se debe hacer la conversión con el tipo de dato double para almacenar el resultado de la división entre las 2 variables para obtener un numero decimal que seria almacenado en la variable promVentas, luego de esto con el objeto creado DecimalFormat que es algo nuevo que he hallado mientras miraba videos y paginas webs para limitar la cantidad de decimales que tiene el número, eso se puede realizar con **DecimalFormat df = new DecimalFormat (“#.##”)** donde el hash(#) después del punto representa la cantidad de decimales que queremos que se limite en el resultado en consola.

Anexo se deja captura de pantalla del resultado en consola.

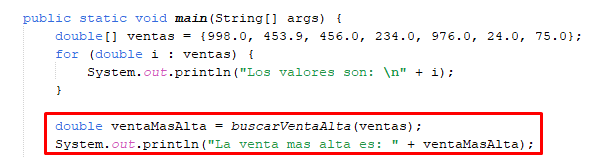
# **PUNTO 3**

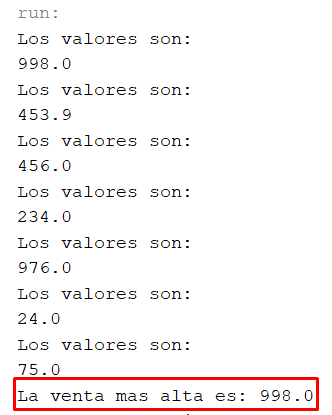
# Buscar elementos en un arreglo:

## **a) Diseñar una función que permita buscar la venta más alta e invocar la función para probarla.**



En este punto como nos pide el ejercicio hemos pasado a definir la función que nos va permitir buscar nuestra venta más alta, nuestra función la hemos definido buscarVentaMasAlta, con un tipo de dato de entrada tipo double y al retornar nos regrese el valor de la venta más alta en nuestro arreglo, en este caso nuestra función buscarVentaMasAlta recibe un parámetro de entrada de tipo double y le hemos asignado el nombre de ventas, luego al declarar nuestra variable ventaMasAlta le asignamos el valor del primer elemento del arreglo de ventas, y pasamos a utilizar el bucle for para realizar iteraciones sobre nuestro arreglo ventas, con este bucle lo que hacemos es que en cada iteración se compara si el valor actual de nuestro arreglo es mayor que ventaMasAlta y si es así se actualizara el valor de ventaMasAlta con el valor actual de nuestro arreglo, una vez finalizado nuestro bucle se retorna el valor de la ventaMasAlta y ahora si podremos pasar a probar la función, para esto creamos lo siguiente:

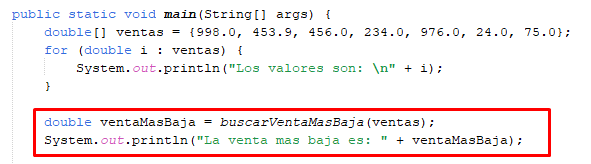


Como podemos apreciar declaramos un arreglo llamado ventas como se nos pide inicialmente, le asignamos valores a nuestro arreglo y se imprime en consola, luego mediante un bucle for-each pasara a recorrer cada elemento de nuestro arreglo ventas y se asignara en nuestra variable i, la cual es de tipo double y luego la pasaremos a imprimir para poder seguir trabajando en la siguiente línea de código, en la cual estaremos llamando a nuestro método buscarVentaMasAlta en la cual estará recibiendo un arreglo de tipo double llamado ventas, en nuestra variable ventaMasAlta se almacenara el valor que se regresa del método buscarVentaMasAlta, que en nuestro caso corresponderá al valor de nuestra venta mas alta en nuestro arreglo de ventas, para finalizar imprimimos nuestro mensaje seguido del valor de nuestra variable llamada ventaMasAlta para que nos quede de la siguiente manera:

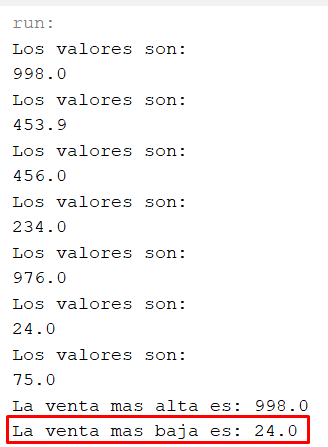
Como podemos apreciar nos arroja el valor de la venta mas alta y con esto finalizamos este punto.

## **b) Diseñar una función que permita buscar la venta más baja e invocar la función para probarla**.

En este punto b como nos pide el ejercicio hemos pasado a definir la función que nos va permitir buscar nuestra venta más baja, nuestra función la hemos definido buscarVentaMasBaja, con un tipo de dato de entrada tipo double y al retornar nos regrese el valor de la venta más baja en nuestro arreglo, en este caso nuestra función buscarVentaMasBaja recibe un parámetro de entrada de tipo double y le hemos asignado el nombre de ventas, luego al declarar nuestra variable ventaMasBaja le asignamos el valor del primer elemento del arreglo de ventas, y pasamos a utilizar el bucle for para realizar iteraciones sobre nuestro arreglo ventas, con este bucle lo que hacemos es que en cada iteración se compara si el valor actual de nuestro arreglo es menor que ventaMasBaja y si es así se actualizara el valor de ventaMasBaja con el valor actual de nuestro arreglo, una vez finalizado nuestro bucle se retorna el valor de la ventaMasBaja y ahora si podremos pasar a probar la función, para esto creamos lo siguiente:

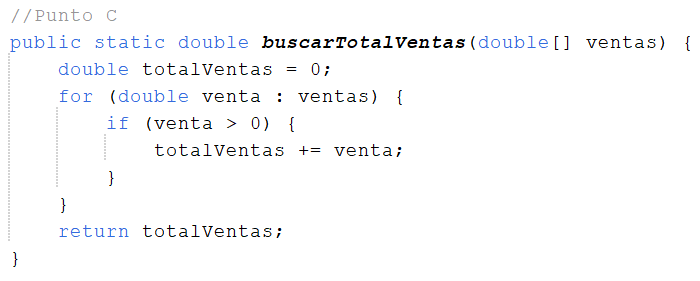


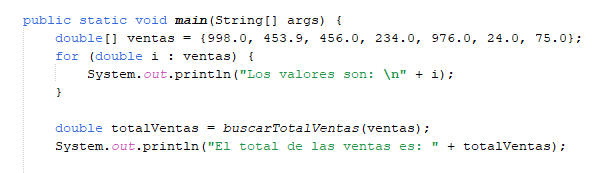
Como podemos apreciar declaramos un arreglo llamado ventas como se nos pide inicialmente, le asignamos valores a nuestro arreglo y se imprime en consola, luego mediante un bucle for-each pasara a recorrer cada elemento de nuestro arreglo ventas y se asignara en nuestra variable i, la cual es de tipo double y luego la pasaremos a imprimir para poder seguir trabajando en la siguiente línea de código, en la cual estaremos llamando a nuestro método buscarVentaMasBaja en la cual estará recibiendo un arreglo de tipo double llamado ventas, en nuestra variable ventaMasBaja se almacenara el valor que se regresa del método buscarVentaMasBaja, que en nuestro caso corresponderá al valor de nuestra venta más baja en nuestro arreglo de ventas, para finalizar imprimimos nuestro mensaje seguido del valor de nuestra variable llamada ventaMasBaja para que nos quede de la siguiente manera:

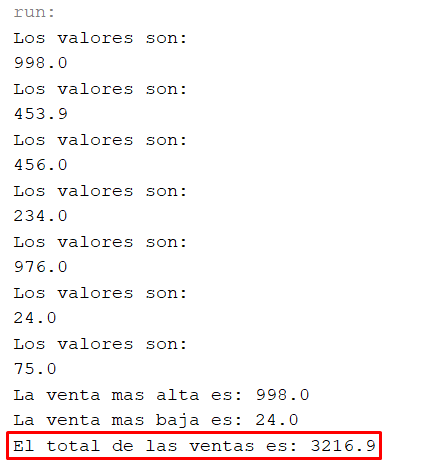


Como podemos apreciar nos arroja el valor de la venta más baja y con esto finalizamos este punto.

## **c) Diseñar una función que permita obtener el total de las ventas e invocar la función para probarla**.



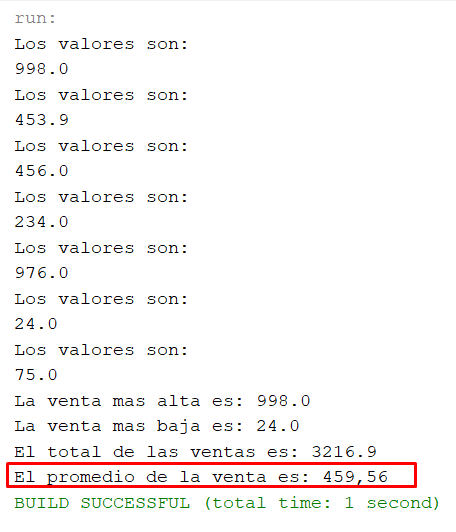
En este punto c como nos pide el ejercicio hemos pasado a definir la función que nos va permitir buscar nuestro total de ventas, nuestra función la hemos definido buscarTotalVentas, con un tipo de dato de entrada tipo double y al retornar nos regrese el valor total de las ventas en nuestro arreglo, en este caso nuestra función buscarTotalVentas recibe un parámetro de entrada de tipo double y le hemos asignado el nombre de ventas, luego al declarar nuestra variable totalVentas le asignamos el valor del primer elemento del arreglo de ventas (Cabe destacar que utilizamos una condición del ir para que evalué si la venta es mayor que cero, esto como lo habíamos realizados en anteriores trabajos se hace para validar que no se nos sumen ventas negativas o que sea totalmente nulas) , y pasamos a utilizar el bucle for para realizar iteraciones sobre nuestro arreglo ventas, con este bucle lo que hacemos es que en cada iteración se compara si el valor actual de nuestra venta actual es mayor que cero, en caso de ser así se sumara el valor de nuestra venta actual a la variable totalVentas, en caso de ser cero no se sumara, quedara el valor igual hasta seguir al siguiente número, finalmente nos devolverá un valor de totalVentas que pasaría a representar el total de todas las ventas de nuestro array, a continuación se deja captura de la impresión en consola.



Como podemos apreciar nos arroja la sumatoria total de las ventas.

## **d) Diseñar una función que permita obtener el promedio de las ventas e invocar la función para probarla.**

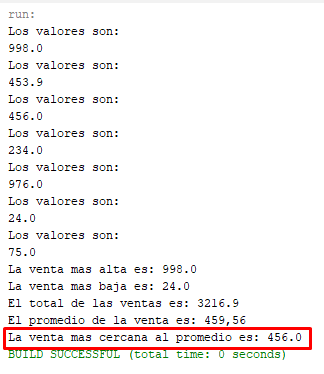
Para obtener el promedio hemos creado el método promVentas el cual nos recibe un parámetro de tipo double que representa nuestras ventas, en esta sección del código básicamente lo que hacemos es buscar la sumatoria de todas las ventas almacenadas en totalVentas para luego dividirla entre la cantidad total de ventas usando la propiedad .length del arreglo con lo cual al imprimir el resultado estaría arrojando el promedio de las ventas, adicional hemos utilizado una instancia de la clase DecimalFormat la cual utilizamos para formatear los valores de los decimales, colocando por medio de # la cantidad decimales que queremos agregar, al final cuando imprimimos por consola nos arrojara el siguiente código:



## **e) 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**.

En nuestro método buscarVentaMasCercanaProm buscaremos en nuestro arreglo de ventas el valor que se encuentra mas cercano a nuestro promedio del total de las ventas, para esto tendríamos que inicializar dos variables, ventaMasCercanaProm y difActual, la primera variable se inicializa con el valor de nuestra primera venta en el arreglo ventas y la segunda variable se inicializara con la diferencia entre esa venta y el promedio de todas nuestras ventas, luego utilizaremos un bucle while para iterar a través del resto de nuestras ventas del arreglo, en cada iteración del bucle se calculara la diferencia entre la venta actual y el promedio de las ventas y se valida si la diferencia es menor que la diferencia actual, si es así se actualizarán el valor en la variable difActual y se actualizara nuestra variable ventaMasCercanaProm con el valor de la venta actual y finalmente retornara el valor que esta más cercano al promedio.

Por otro lado, utilizamos math.abs para poder obtener el valor absoluto de un número, en caso de la diferencia ser negativa se tomará el valor positivo y el resultado de nuestra operación sea almacenado en la variable difActual, de esta manera implementamos math.abs en nuestro arreglo.

Este último punto ha resultado un poco más complejo y se investigó mucho y se observaron varios videos y documentación para poder realizarlo, pero aun así quedo con la duda de poder resolverlo de una manera más sencilla.

# **PUNTO 4**

# Ordenar arreglos:

## **a) Diseñar una función que permita ordenar las ventas de forma descendente e invocar la función para probarla.**

En este punto el objetivo principal es ordenar el arreglo de ventas en orden descendente, para lo cual hemos definido un método llamado ordenDescenten que nos recibe un parámetro de nuestro arreglo ventas de tipo double.

En este caso mediante el método public hacemos que el método pueda ser accedido desde cualquier parte del programa y con la palabra static indicamos que el método pertenece a la clase en la cual esta definido y por ultimo con la palabra void nos indica que el método no nos retornara ningún valor, por lo tanto para proceder a resolver este código hemos utilizado el método Arrays.sort que nos sirve para ordenar los elementos de nuestro array en forma ascendente, para que esto funcione nuestro array debe tener elementos que se puedan comparar entre ellos para seguir ejecutándose y no nos arroje error, una vez que los elementos de nuestro arreglo ventas se organice de forma ascendente pasamos a crear un bucle for para realizar iteraciones en nuestro arreglo, en este paso solo lo haremos hasta la posición n2 (Que sería la mitad del arreglo) en este solo se organizara de formar descendente los elementos de mayor valor y luego desde la posición n2 hasta la última posición se organizar de manera descendente pero serán los elementos más pequeños de nuestro arreglo, de esta manera lo que haremos al final es intercambiar elementos de nuestra primera mitad del arreglo con la otra segunda mitad, de esta manera se irán comparan entre sí y los elementos quedaran organizados de manera mayor a menor.

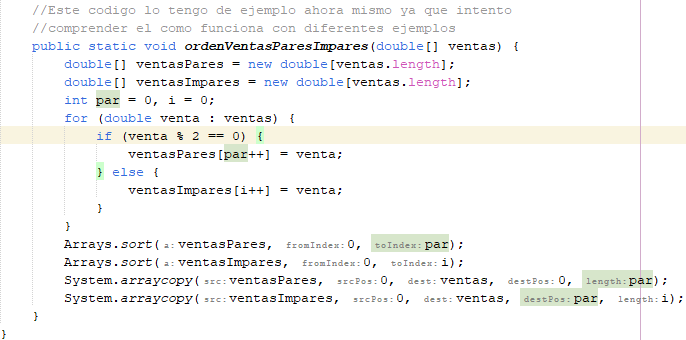
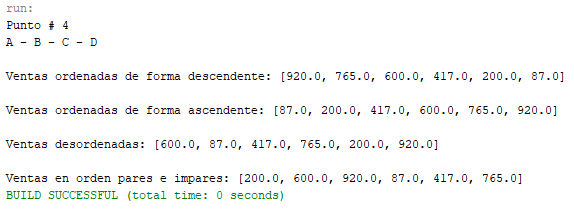
## **b) Diseñar una función que permita ordenamiento que permita ordenar las ventas de forma ascendente e invocar la función para probarla.**

En esta sección del código es un poco mas sencilla ya que como anteriormente hemos utilizado el método Arrays.sort se nos hace mas fácil ya que mediante este podemos organizar el código de manera ascendente, de esta manera no necesitamos emplear ningún bucle ya que mediante este método se realizará la organización automáticamente.

## **c) Diseñar una función que permite desordenar las ventas e invocar la función para probarla.**

En este código lo que implementaremos será un algoritmo que nos permita desordenar de manera aleatoria nuestro array que únicamente contiene números de tipo double, en nuestro array ventas hemos utilizado la clase Random que anteriormente la hemos utilizado y sabemos como funciona, con esto generaremos números aleatorios y cada vez que se imprima se mostraran los números de manera aleatoria y no de la misma manera que hemos colocado los valores en el arreglo, luego en el bucle for que hemos creado en la siguiente línea de código iremos recorriendo cada uno de los elementos de nuestro arreglo ventas y en este mismo se generara un un numero aleatorio utilizando el método nextInt de nuestra clase Random, este lo utilizamos para generar un elemento aleatorio y ira intercambiando con el elemento actual en la iteración de nuestro bucle, como tal el numero que se genera de manera aleatoria se ira guardando en nuestra variable vlrTemp, y en cada iteración se ira intercambiando el elemento que se genera con el actual, es decir que este proceso de generar los números de manera aleatoria se realizara gracias al método random y a nuestro bucle.

## **d) Diseñar una función que permita ordenar las ventas primero las pares y luego en impares e invocar la función para probarla.**



En este caso se tiene como tarea, ordenar el arreglo de ventas en 2 arreglos, para de esta manera poder separar los partes y los impares y luego pasar a copiar los 2 arreglos que están ordenados para crear uno que sea el original y colocar los números pares primero y luego los impares después de estos, creamos dos arreglos de tipo double, uno llamado ventasPares y otro llamado ventasImpares para poder almacenar las ventas pares y las ventas impares, en la siguiente línea de código hemos creado un bucle for-each para realizar iteraciones en nuestro arreglo principal de ventas, de esta manera se puede validar si el numero que se va guardar es par o impar, aquí utilizaremos el operador modulo %, gracias a esto regresara el resultado de una división de dos números, en nuestro caso si el resultado de la división de nuestro arreglo ventas x 2 es igual a cero se considerara que nuestro numero es para y si arroja un numero diferente de cero se puede considerar que este es un numero impar, así de esta manera en cada iteración se agregar el resultado de cada iteración a uno de los dos arreglos que hemos creado, una vez que se han separado las ventas en pares e impartes, se ordenan los dos arreglos por separado utilizado el método Arrays.sort, es de aclarar que se ordenan en cada arreglo de manera independiente, para poder combinar estos arreglos utilizaremos un método llamado arraycopy de la clase system de esta manera ambos arreglos pasaran a ser uno solo y se iran agregando al arreglo de la siguiente manera, el primer arreglo de ventasPares se colocara en las primeras par posiciones del arreglo ventas y las otras posiciones las ocupara el arreglo ventasImpares.

# **PUNTO 5**

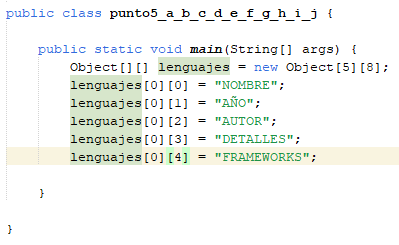
# Trabajar con Matrices (tablas o arreglos multidimensionales)

## **a) Declarar e inicializar una matriz de 5 x 8 de tipo Object en una variable llamada lenguajes**

Aquí hemos declarado e inicializado una matriz de tipo object con 5 filas y 8 columnas

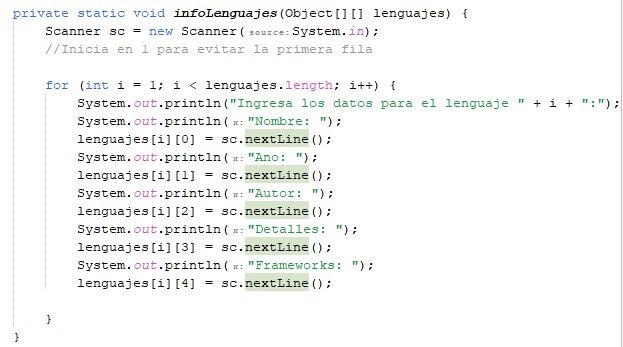
## **b) A la primera línea de la matriz lenguajes se debe asignar los siguientes valores en ese orden:**

**[NOMBRE, AÑO, AUTOR, DETALLES, FRAMEWORKS]**

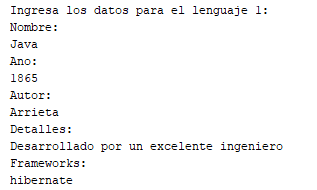


En este punto hemos asignado valores a la primera fila de nuestra matriz lenguajes, la cual tiene un tamaño de 5 filas y 8 columnas como se había mencionado anteriormente, en la primera fila y primera columna se asignó el nombre, luego en la primera fila y segunda columna se asigno el año, luego en la primera fila y tercera columna se agrego autor y así hasta terminar, de esta manera hemos inicializado la primera fila de nuestra matriz con sus respectivos valores.

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



En este punto se nos pide llenar la matriz lenguajes con datos ingresados por teclado, lo primero que debemos hacer es crear un método que se llame infoLenguajes tal cual como se pide en el enunciado, en este caso se toma una matriz bidimensional es decir una estructura de datos que esta compuesta por elementos organizados en filas y columnas, después de saber esto avanzamos e ingresamos la clase Scanner la cual utilizamos para leer la entrada de los datos los cual se ingresaran, adicional debemos crear un bucle for para realizar iteraciones por cada fila en nuestra matriz comenzando desde nuestra segunda fila que en este caso seria el índice numero uno, luego dentro de nuestro bucle imprimimos un mensaje solicitando al usuario que ingrese los datos para el lenguaje actual, adicional debemos utilizar un método llamado nextLine el cual nos sirve para leer la respuesta del usuario que ingresa los datos y luego guardarlas en nuestra matriz de lenguajes, se colocara en cada columna correspondiente como lo es nombre, año, autor, etc, después de esto imprimimos para que nos muestre lo siguiente en consola:

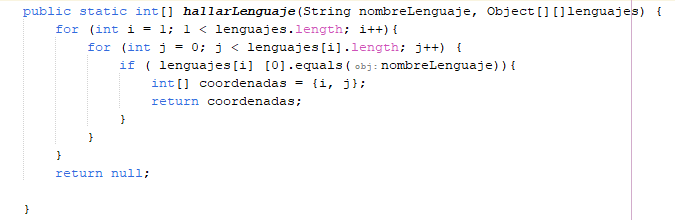


De esta manera se nos solicitara algunos datos y luego se irán ingresando uno a uno hasta completarlos todos, todo esto se guardara en una matriz.

## **d) 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.**

En este punto debemos utilizar el tipo de dato boolean para evaluar si una condiciones es verdadera o falsa, lo primero que debemos hacer es crear nuestro método buscarLenguaje el cual recibirá dos parámetros uno de tipo string y otro de tipo objeto, en este caso seria nombreLenguaje y lenguajes, con este código lo que queremos es buscar el nombre de un lenguaje que estará alojado en nuestra matriz lenguajes; implementaremos un bucle de tipo for para realizar iteraciones en nuestra matriz de lenguajes atrás de sus filas pero mas precisamente arrancando en la segunda fila de esta en la posición uno, lo siguiente es comparar dentro del bucle el valor de nuestra primera columna de cada fila el valor del nombreLenguaje, si el valor es exactamente igual retornara un valor verdadero, indicado que se hallo el lenguaje que estábamos buscando, en caso que no se cumpla lo condición con exactitud se retornara un false indicando que en la matriz lenguaje no se hallo el lenguaje que estábamos buscando.

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



En nuestro código recorreremos la matriz de lenguajes la cual esta representada por un arreglo bidimensional (Organizados en filas y columnas) llamado lenguajes, mediante el cual se pasara un nombre a nuestro parámetro encontrarLenguaje, como anteriormente se había mencionado y plasmado en código nuestra matriz lenguaje tiene una fila superior o fila cero la cual contiene todos los datos del tipo de lenguaje como por ejemplo nombre, año, autor entre otros datos, nuestra función lo que hará es recorrer las filas excluyendo únicamente la primeras, además de recorrer las filas también va recorrer las columnas de nuestra matriz, como se ha realizado en otros puntos se realizara una iteración en la cual se evaluara si el nombre del lenguaje actual ubicado en cierta posición es igual a la búsqueda del lenguaje que se esta realizando, en caso de encontrar una coincidencia nuestra función va regresar la coordenadas de la celda en la matriz como un arreglo de enteros el cual contiene dos elementos, donde el primer el elemento corresponderá al numero de fila y el segunda corresponde al numero de la columna.

## **f) 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.**

En este punto hemos creado un método obtenerRegistro que recibirá el parámetro fila y la matriz lenguaje, como tal se quiere validar la información de la matriz lenguaje y se validara si el numero de la fila es válido, en caso de no ser valido se lanzara lo que se le llama como excepciones la cual arrojara un mensaje de error, en caso de ser correcto se regresara el arreglo que contiene todos los valores de la fila de nuestra matriz lenguajes.

# Bibliografía

Java - Trabajo con arrays - Píldoras Informáticas

Programación en Java || Arreglos || Definición de Arreglos

<https://www.youtube.com/watch?v=AZc3hxMxl54&ab_channel=Programaci%C3%B3nATS>

Programación en Java || Arreglos || Llenar un arreglo

<https://www.youtube.com/watch?v=79LOsYYZm_8&ab_channel=Programaci%C3%B3nATS>

Programación en Java || Arreglos || Bucle for each

<https://www.youtube.com/watch?v=jRSQVeQRvKc&ab_channel=Programaci%C3%B3nATS>

Programación en Java || Arreglos || Ejercicio - Imprimir en orden 5 números de un arreglo

<https://www.youtube.com/watch?v=ftmfV3Yj8so&ab_channel=Programaci%C3%B3nATS>

Programación en Java || Arreglos || Ejercicio - Promedios en un arreglo

<https://www.youtube.com/watch?v=Bb6KAQFBcdc&ab_channel=Programaci%C3%B3nATS>

Programación en Java || Arreglos || Ejercicio - Mezclar 2 arreglos

<https://www.youtube.com/watch?v=0oTbM03P0Vw&ab_channel=Programaci%C3%B3nATS>

Programación en Java || Arreglos || Ejercicio - Arreglo en forma creciente o decreciente

<https://www.youtube.com/watch?v=AvajxzPiXy8&ab_channel=Programaci%C3%B3nATS>

ORDENAR un ARREGLO de forma ASCENDENTE en Java | Ordenar ARREGLO de forma DESCENDENTE en Java

<https://www.youtube.com/watch?v=UL4fZpn6I-8&ab_channel=ByspelTech>

Estructuras de Datos - ARREGLOS Ejemplo en Java parte 1 - Modelo

<https://www.youtube.com/watch?v=7piqgA62oe0&ab_channel=JohnCarlosArrietaArrieta>