Actividad 1 - Arreglo y algoritmos de ordenamiento

Reinaldo David Quintana Moreno

Universidad de Cartagena

Estructura de datos

John Carlos Arrieta Arrieta

23 de abril de 2023

Contenido

[Introducción 3](#_Toc133081703)

[Objetivos 4](#_Toc133081704)

[Objetivo general 4](#_Toc133081705)

[Objetivos específicos 4](#_Toc133081706)

[Justificación 5](#_Toc133081707)

[Desarrollo de la actividad 6](#_Toc133081708)

[Código fuente y explicación 6](#_Toc133081709)

[Ejecución del programa 16](#_Toc133081710)

# Introducción

En el presente trabajo se adentra un poco raen el amplio mundo de los arreglos y algoritmos de ordenamiento en el entorno del IDE Apache NetBeans, se denotan las grandes posibilidades que se tienen al implementar arreglos, establecer ciertas cantidades de elementos y manipularlos aleatoriamente por medio de métodos como “random”, hace ver la infinidad de cosas que se pueden crear y experimentar. Como por ejemplo obtener cierta cantidad de números al azar y realizar operaciones aritméticas con estos.

# Objetivos

## Objetivo general

Implementar arreglos y algoritmos de ordenamiento.

## Objetivos específicos

* Declarar y definir arreglos con datos primitivos.
* Inicializar arreglos con cantidades de elementos aleatorios.
* Obtener resultados de operaciones aritméticas con arreglos.

# Justificación

El presente trabajo alberga datos importantes en cuanto al código JAVA y sintaxis que se puede implementar al momento de elaborar un programa en el cual se requieran arreglos en los que se pidan pequeñas o grandes cantidades de elementos y que además se soliciten operaciones como calcular promedio, total, entre otras.

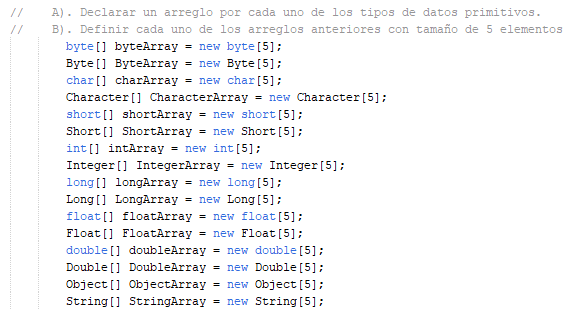
# Desarrollo de la actividad

## Código fuente y explicación

**1)** Declarar y definir arreglos:

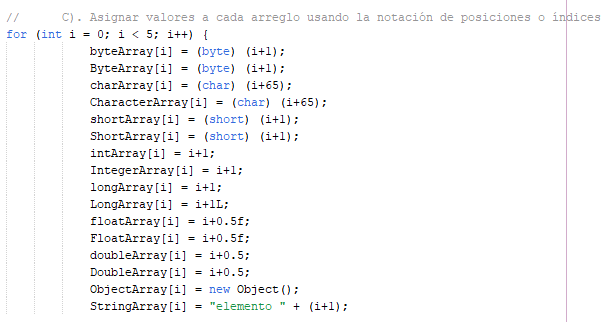
**A.** Declarar un arreglo por cada uno de los tipos de datos primitivos (byte, Byte, char, Character, short, Short, int, Integer, long, Long, float, Float, double, Double, Object, String.

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



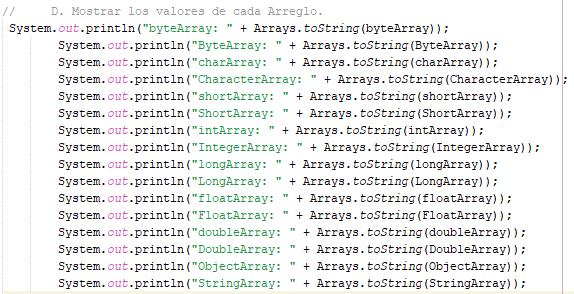
Para declarar un arreglo de los tipos de datos primitivos invocamos el dato agregando corchetes, he indicamos el tamaño que almacenará, es este caso 5, tal como vemos en la imagen.

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



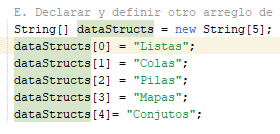
Se usa el buche “for”, inicializando los arreglos con el iterador en 0 colocándolo hasta una posición antes de la cantidad de elementos que les asignamos inicialmente e incrementándolo en 1. Posteriormente invocamos los arreglos iterados, añadiéndole algunos valores extras.

**D.** Mostrar los valores de cada Arreglo.



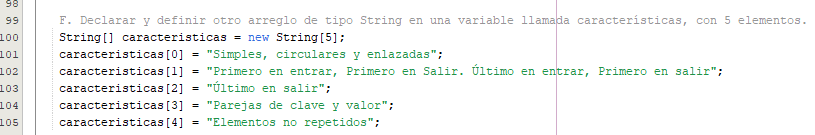
Se coloca la instrucción “System.out.println” para indicar que se imprimirá un texto en pantalla, El método Arrays.toString() convierte el arreglo a una cadena de caracteres y la concatena con la etiqueta del arreglo, especificándose el arreglo a imprimir.

**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.



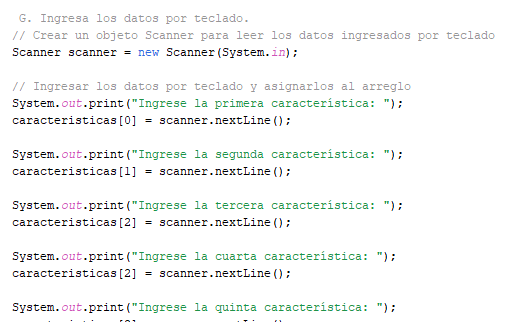
Se declara y define un arreglo de tipo String, tal como se especifica en el punto uno, pero empleando el nombre de variable dataStructs, e inicializándose con los datos especificados (listas, colas, pilas, mapas y conjuntos).

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



Se declara y define un arreglo de tipo “String”, tal como se especifica en el punto uno, pero empleando el nombre de variable “caracteristicas”, con tamaño de 5 elementos.

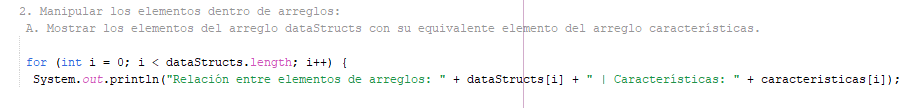
**G.** Ingresar los siguientes datos por teclado.



Se invoca la clase "Scanner scanner = new Scanner();" para poder obtener la información solicitada, ya que es por teclado se indica "System.in". Posteriormente indicamos un "System.out.print" para mostrara en pantalla la solicitud que se le hace al usuario, se llama el arreglo con el que se está trabajando e indicamos el metodo "nextLine()" para leer cadenas de textos.

**2)** Manipular los elementos dentro de arreglos:

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



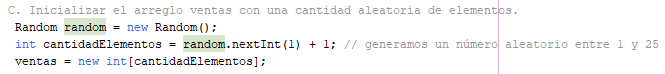
Indicamos el nombre de la variable en este caso es "dataStructs" y elegimo el metodo ".length" que nos permite averiguar la lungitud de una cadena de caracteres, para que se pueda ver en pantalla invocamos "System.out.print" agregando un mensaje y seguidamente colocamos las variables con su respectivo iterador.

**B.** Declarar un arreglo de enteros en una variable llamada ventas.



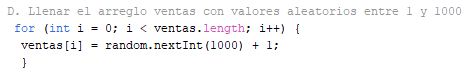
Se coloca "int" que es el dato primitivo tipo entero colocando corchetes para que se entienda que es un arreglo y se coloca el nombre de la variable "ventas".

**C.** Inicializar el arreglo ventas con una cantidad aleatoria de elementos.



Invocamos la clase "Random" para que nos devuelva un aleatorio, seguido del nombre del objeto y “= new Random();” se crea un entero con “int” y nombrando el objeto el cual por si mismo no devolverá un valor por lo que se debe invocar el método "nextLine()"

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



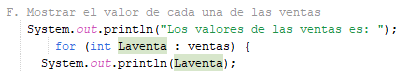
Usamos el bucle “for” para recorrer el arreglo ventas, el vual contiene “i < ventas.length” que en este caso sirve para recorrer el arreglo ya menionado, la iteración “i” va incrementando de uno en uno por medio de “++”

**E.** Mostrar la cantidad de ventas.



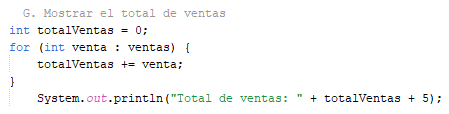
Se imprimen los elementos del arreglo ventas empleando “System.out.println” y la función “Arrays.toString()” para convertir el arreglo en una cadena de texto que se puede imprimir.

**F.** Mostrar el valor de cada una de las ventas.



Se muestra en pantalla el valor de cada venta, usando “for (int Laventa : ventas)” el cual recorre el arreglo asigna valores a las variables, finalmente se imprime.

**G.** Mostrar el total de ventas



Para mostrar el total de ventas empleamos “+=” lo cual se usa para sumar el valor de la variable a la derecha del operador al valor de la que se encuentra a la izquierda y asignar el resultado, en este caso, la variable es “totalVentas”. Finalmente se suma el número 5 y se imprime.

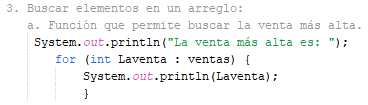
**H.** Mostrar el promedio de las ventas.



Se crea la variable “promedioVentas”, de tipo “double” para poder tener un resultado decimal, la cual se divide con “totalVentas” para ello se implementó “.length”. Por último se imprime.

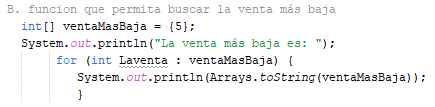
**3)** Buscar elementos en un arreglo:

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



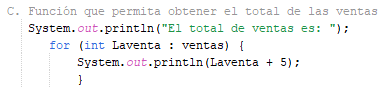
Se imprime por pantalla la venta más alta usando “System.out.println” implementando el bucle “for” en ello.

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



Se imprime por pantalla la venta más baja usando “System.out.println” implementando el bucle “for” en ello y el método Arrays.toString() que convierte el arreglo a una cadena de caracteres y la concatena con la etiqueta del arreglo, especificándose la variable a imprimir.

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



Se imprime por pantalla el total de ventas usando “System.out.println” implementando el bucle “for” en ello, sumándose el valor de todas las ventas.

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



Se imprime por pantalla el promedio de ventas usando “System.out.println” e invocando la variable promedioVentas.

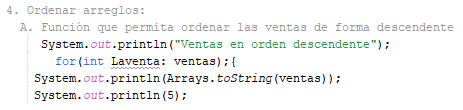
**E.** Diseñar una función que devuelva la venta cuyo valor se acerque más al valor que le sigue al promedio e invocar la función para probarla.



En este caso la venta cuyo valor se acerca más al valor que le sigue al promedio es el mismo promedio de venta, por lo que se imprime en pantalla usando “System.out.println” e invocando la variable promedioVentas.

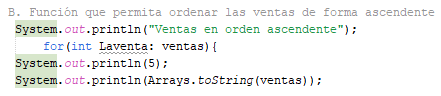
**4)** Ordenar arreglos:

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



Se invocan los arreglos de las ventas en el orden pedido, se usa el buche “for” y se imprime en pantalla.

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



Se invocan los arreglos de las ventas en el orden pedido, se usa el buche “for” y se imprime en pantalla.

## Ejecución del programa

