/* Arreglos y archivos */

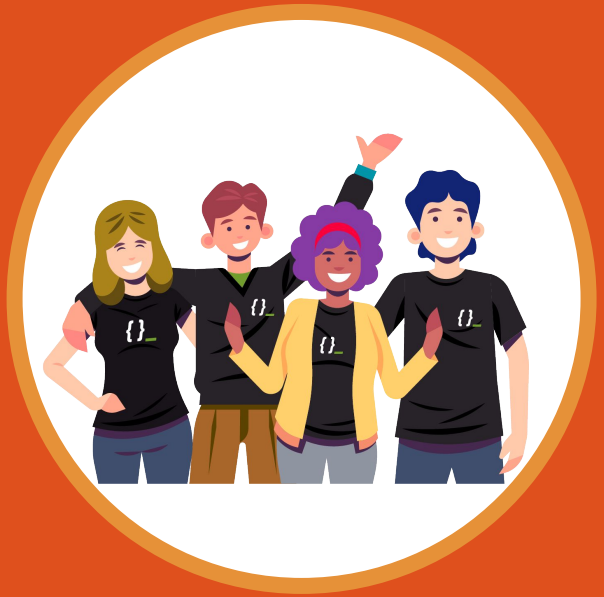
Sesión conceptual 02

{desafío} latam_

- Objetivo de la sesión
- Activación de conceptos clave

- Conceptualización
- Ejercitación

- Reflexión



Inicio

{desafío}
latam_

Introducción

Para operar sobre arreglos, o colecciones de datos en general, tenemos la API Stream, que nos permite realizar operaciones sobre nuestro conjunto de datos.

Streams: secuencia de datos

Mostrar en Stream

Para ver los elementos

```
List<Integer> numeros = Arrays.asList(1,4,8,5,5,10,2);
```

Mostrar en Stream

```
List<Integer> numeros = Arrays.asList(1,4,8,5,5,10,2);
```

Para ver los elementos

```
numeros.stream().forEach(System.out::println);
```

Usando map

```
numeros.stream().map(n -> n*3).forEach(System.out::println);
```

```
3
12
24
15
15
30
6
```

Cada elementos se multiplicó por 3, y luego se mostró por pantalla, y si vemos que paso con el arreglo

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

ningún elemento fue modificado.

```
[1, 4, 8, 5, 5, 10, 2]
```

Creando lista para almacenar el resultado

```
List<Integer> numeros = Arrays.asList(1,4,8,5,5,10,2);
List<Integer> numerosAumentados = new ArrayList<Integer>();
numeros.stream().map(n -> n*3).forEach(numerosAumentados::add);

System.out.println(numeros);
System.out.println(numerosAumentados);
```

```
[1, 4, 8, 5, 5, 10, 2]

[3, 12, 24, 15, 15, 30, 6]
```

Usando Collectors

```
import  java.util.stream.Collectors;
```

```
import java.util.stream.Collectors;
List<Integer> numeros = Arrays.asList(1,4,8,5,5,10,2);
List<Integer> numerosAumentados2 = numeros.stream().map(n -> n*n).collect(Collectors.toList());
```

Predicado

reciben un argumento y devuelven un valor luego de realizar una operación

`e -> operación`

Otro ejemplo

```
List<Integer> numeros = Arrays.asList(1,4,8,5,5,10,2);
List<Integer> numerosAUno =
numeros.stream().map(n -> 1).collect(Collectors.toList());
System.out.println(numerosAUno);
```

```
[1, 1, 1, 1, 1, 1, 1]
```

Creando filtros

filter(predicado)

```
List<Integer> numeros =
Arrays.asList(1,4,8,5,5,10,2);
List<Integer> numFiltrados = numeros.stream()
        .filter(x -> x>=4)
        .collect(Collectors.toList());
System.out.println(numFiltrados);
```

```
[4, 8, 5, 5, 10]
```

y si queremos el cuadrado de cada uno de ellos

```
List<Integer> numFiltrados = numeros.stream()
        .filter(x -> x>=4)
        .map(x -> x*x)
        .collect(Collectors.toList());
System.out.println(numFiltrados);
```

```
[16, 64, 25, 25, 100]
```

Podemos concatenar operaciones!

{desafío} latam_

Filtros

distinct()

```
List<Integer> numerosDistintos =
numeros.stream()
        .distinct()
        .collect(Collectors.toList());

System.out.println(numerosDistintos); //[1, 4, 8, 5, 10, 2]
```

limit()

```
List<Integer> numerosLimit =
numeros.stream()
        .limit(2)
        .collect(Collectors.toList());

System.out.println(numerosLimit); //[1,4]
```

Filtros

limit(n)

```
List<Integer> numerosLimit =
numeros.stream()
        .limit(2)
        .collect(Collectors.toList());

System.out.println(numerosLimit); //[1,4]
```

skip(n)

```
List<Integer> numerosSkip =
numeros.stream()
        .skip(2)
        .collect(Collectors.toList());

System.out.println(numerosSkip); //[5, 5, 10, 2]
```

Reduciendo los datos

Si queremos obtener por ejemplo, la **suma** de todos los elemento del arreglo:

```
int suma = numeros.stream().reduce(0, (a,b) ->a+b);
System.out.println(9suma);
```

reduce() -> es un método acumulador, donde se define en valor de inicio y la operación a realizar.

```
int multiplicacion = numeros.stream().reduce(1, (a,b) ->a*b);
System.out.println(multiplicacion);
```

para **multiplicar** todos los elementos del arreglo

```
int max = numeros.stream().reduce(1, Integer::max);
int min = numeros.stream().reduce(1000,Integer::min);
System.out.println("min: " + min + "\nmax: "+ max);  //10
```

Operando con Streams numéricos

Existen Streams que operan sobre números:

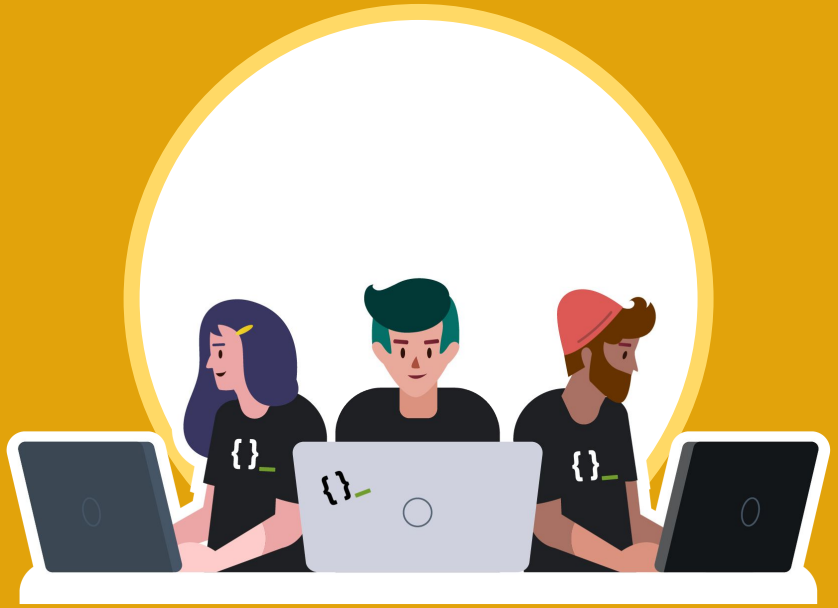
- mapToInt()
- mapToDouble()
- mapToLong().

```
int suma =
numeros.stream().mapToInt(Integer::intValue).sum();

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

y tienen métodos como

- sum()
- average()
- count()
- max()
- min()
- summaryStatistics()



Desarrollo

{desafío}
latam_

Ejercicio 1

Dado un arreglo de enteros

```
List<Integer> numeros = Arrays.asList(1,9,2,10,2,4,7,4,7,1,4);
```

realizar las siguientes operaciones

1. Utilizando reduce, sumar todos los valores del array
2. Utilizando reduce, sumar todos los valores no repetidos del array
3. Convertir todos los datos al tipo float
4. Filtrar todos los elementos menores a 5 sin repetir
5. Utilizando mapToInt, sumar todos los valores del arreglo
6. Utilizando .count() contar todos los elementos menores que 5

Ejercicio 2

Dado un arreglo con nombres

```
List<String> nombres = Arrays
.asList("Anastasia","Beatriz","Clara",

            "Carla","Marianela","Paula", "Pia");
```

1. Obtener todos los elementos que excedan los 5 caracteres
2. Utilizar map para transformar todos los nombres a minúscula
3. Crear un arreglo con todos los nombres que comiencen con P
4. Utilizando .count, contar los elementos que empiecen con 'A', 'B' o 'C'.
5. Utilizando .map, crear un arreglo con la cantidad de letras que tiene cada nombre.

Archivos, Arrays y Strings

{desafío}
latam_

Métodos para facilitar el trabajo con archivos

String

```
String letras = "a,b,c,d,e,f";
ArrayList<String> letrasSeparadas = new
ArrayList<String>(Arrays.asList(letras.split(",")));
System.out.println(letrasSeparadas);
```

```
[a, b, c, d, e, f]
```

Join

```
String letrasJuntas = String.join(" -> ", letrasSeparadas);
System.out.println(letrasJuntas);
```

```
a -> b -> c -> d -> e -> f
```

Crear archivo

En Eclipse:

1. Clic derecho sobre el proyecto
2. New
3. File
4. Ingresamos el nombre del archivo

Dentro del archivo escribiremos la siguiente información

```
1,2,3,4,5,6,7,8,9,10
```

Abrir archivo

Librería FileReader

```
String nombre = "data";
try {

        FileReader fr = new FileReader(nombre);
    //código donde haremos lectura del archivo

    fr.close();
}
catch (Exception e){
        System.out.println("Excepcion leyendo fichero "+ nombre + ": " + e);
}
```

Leer archivo

usando método read()

```
String nombre = "data";
try {

        FileReader fr = new FileReader(nombre);
        int data=fr.read();
        String lectura = "";
        while(data != -1) {
                lectura += (char)data;
                data = fr.read();  //leemos el siguiente caracter
        }
        System.out.println(lectura); //1,2,3,4,5,6,7,8,9,10
        fr.close();
}
catch (Exception e){
        System.out.println("Excepcion leyendo fichero "+ nombre + ": " + e);
}
```

Que pasa si queremos leer multiples lineas y almacenarlas de manera separada?

```
if((char)data == '\n')
```

Leer archivos por líneas

Agregamos la librería BufferedReader

```
String nombre = "data";
try {
        FileReader fr = new FileReader(nombre);
        BufferedReader br = new BufferedReader(fr);

        String data = br.readLine();
        while(data != null) {//retorna -1 cuando no hay más caracteres por leer
                System.out.println(data); //1,2,3,4,5,6,7,8,9,10
                data = br.readLine();
        }
        br.close();
        fr.close();
}
catch (Exception e){
        System.out.println("Excepcion leyendo fichero "+ nombre + ": " + e);
}
```

Transformando los datos a un array

```
String nombre = "data";

FileReader fr = null;

BufferedReader br = null;

String data = "";

ArrayList<String> numerosString;
ArrayList<Integer> numeros = new ArrayList<Integer>();
try {
        fr = new FileReader(nombre);
        br=new BufferedReader(fr);
        data = br.readLine();
        numerosString = new ArrayList<String>(Arrays.asList(data.split(",")));
        for(String temp : numerosString) {
                numeros.add(Integer.parseInt(temp));
        }
        System.out.println(numeros);
        br.close();
        fr.close();
}
catch (Exception e){

        System.out.println("Excepcion leyendo fichero "+ nombre + ": " + e);
}
```

Leyendo múltiples líneas

data2:

```
21
10
6
9
11
0
2
3
50
```

```
String nombreArchivo = "data2";
FileReader fr = null;
BufferedReader br = null;
String data = "";
ArrayList<Integer> numeros = new ArrayList<Integer>();
try {
        fr = new FileReader(nombreArchivo);
        br=new BufferedReader(fr);

        data = br.readLine();
        while (data != null) {
                numeros.add(Integer.parseInt(data));
                data = br.readLine();
        }
        br.close();
        fr.close();
}
catch (Exception e){

        System.out.println("Excepcion leyendo fichero "+ nombreArchivo + ": " + e);
}
System.out.println(numeros); //[21, 10, 6, 9, 11, 0, 2, 3, 50]
```

Ejercicio resuelto

Dado el data2 que tiene los siguientes datos

```
21
10
6
9
11
0
2
3
50
```

Se pide un crear un programa que tome los datos de ese archivo y construya un arreglo con los mismos pero transformando todos los valores mayores de 20 a un máximo de 20.

Solución

```
String nombre = "data2";
ArrayList<Integer> numeros = readFile(nombre);
int i;
for(i=0;i<numeros.size();i++) {
        if(numeros.get(i) >20) {
                numeros.set(i, 20);
        }
}
System.out.println(numeros);
```

Guardando los resultados

Incorporando las librerías FileWriter y PrintWriter. Creamos un método para escribir el archivo

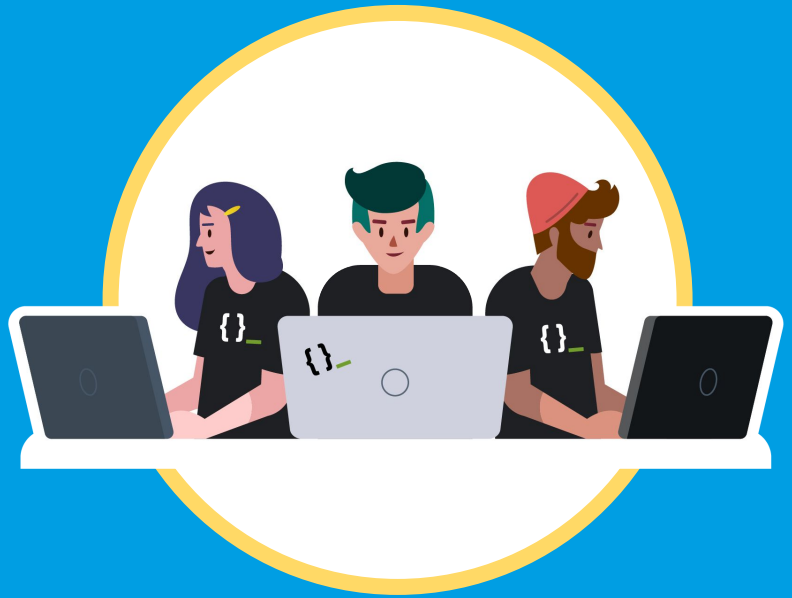
```
void writeFile(String nombreArchivo, ArrayList<Integer> numeros) {
        FileWriter archivo = null;
        PrintWriter pw = null;
        try {
                archivo = new FileWriter(nombreArchivo);
                pw = new PrintWriter(archivo);
                int i;
                for(i = 0 ; i<numeros.size() ; i++) {
                        pw.println(numeros.get(i));
                }
                pw.close();
                archivo.close();
        }
        catch(Exception e){
                System.out.println("Fichero " + nombreArchivo + "no se pudo crear" +
e);
        }
}
```

Cuidado!

- Si el archivo no existe, lo creará
- Si el archivo existe, lo sobreescribirá!!

Para solucionarlo:

```
FileWriter archivo = new FileWriter(nombre,true);
```



Problema con dos Array

{desafío} latam_

Ejercicio

```
ArrayList<Integer> notas = new ArrayList<Integer>(Arrays.asList(5,9,6,8,4));

ArrayList<String> alumno = new ArrayList<String>(Arrays.asList("Julia","María","Teresa","Diego","Pedro"));
```

Algoritmo

- El usuario ingresa un String
- Buscamos el índice en el arreglo de alumnos de ese String
- Si existe el índice
  - buscamos la nota dentro del arreglo de notas
- Sí no:
  - Mostramos que no pudimos encontrar la nota.

Código

```
Scanner sc = new Scanner(System.in);

ArrayList<Integer> notas = new ArrayList<Integer>(Arrays.asList(5,9,6,8,4));
ArrayList<String> alumnos = new ArrayList<String>(Arrays.asList("Julia","María","Teresa","Diego","Pedro"));


String nombreABuscar = sc.nextLine();
int indice = alumnos.indexOf(nombreABuscar);
if(indice != -1)
        System.out.printf ("La nota de %s es: %d\n",nombreABuscar,notas.get(indice));
else
        System.out.println("Alumno no encontrado");
```

Versión con métodos

Se nos pide construir un método que reciba los arrays, el nombre de la persona y de como resultado su nota. En caso de no existir, retornar -1.

```
static int busqueda(ArrayList<Integer> notas, ArrayList<String> alumnos, String nombre) {

	int indice = alumnos.indexOf(nombre);
	if(indice != -1)
		return notas.get(indice);
	else
		return -1;
}
```

Operando sobre valores en dos arrays

Ejemplo de ventas de una empresa: datos de ventas de una empresa en un arreglo y la de la otra empresa en el otro arreglo.
Donde cada posición indica las ventas diarias.

```
ArrayList<Integer> v1 = new ArrayList<Integer>(Arrays.asList(100,20,50,70,90));
ArrayList<Integer> v2 = new ArrayList<Integer>(Arrays.asList(150,30,50,20,30));
```

¿Cómo podemos obtener las ventas diarias de las 2 tiendas en conjunto? ¿por elemento o por índice?

```
ArrayList<Integer> vt = new ArrayList<Integer>();
int n = v1.size();
int i;
for(i=0;i<n;i++){
    vt.add(v1.get(i)+v2.get(i));
}
System.out.println(vt);
```

Ejemplo de torneo

Un ejemplo parecido es construir un listado de todos contra todos. Por ejemplo, si hay 3 equipos: e1, e2, e3, sería:

```
e1 v.s e2
e1 v.s e3
e2 v.s e1
e2 v.s e3
e3 v.s e1
e3 v.s e2
```

Solución 1

```
ArrayList<String> a = new ArrayList<String>(Arrays.asList("Equipo 1","Equipo 2","Equipo 3","Equipo 4","Equipo 5"));
ArrayList<String> b = new ArrayList<String>(Arrays.asList("Equipo 1","Equipo 2","Equipo 3","Equipo 4","Equipo 5"));
ArrayList<String> t = new ArrayList<String>();

for(String tempA : a){
    for(String tempB : b){
        t.add(tempA + " v.s " + tempB);
    }
}
System.out.println(String.join("\n",t));
```

Solución 2 - eliminando repetidos

```
ArrayList<String> a = new ArrayList<String>(Arrays.asList("Equipo 1","Equipo 2","Equipo 3","Equipo 4","Equipo 5"));
ArrayList<String> b = new ArrayList<String>(Arrays.asList("Equipo 1","Equipo 2","Equipo 3","Equipo 4","Equipo 5"));
ArrayList<String> t = new ArrayList<String>();

for(String tempA : a){
    for(String tempB : b){
        if(tempA != tempB){
            t.add(tempA + " v.s " + tempB);
        }
    }
}
System.out.println(String.join("\n",t));
```

Solución Bonus

```
ArrayList<String> a = new ArrayList<String>(Arrays.asList("Equipo 1","Equipo 2","Equipo 3","Equipo
4","Equipo 5"));
ArrayList<String> t = new ArrayList<String>();

for(String tempA : a){
    for(String tempB : a){
        if(tempA != tempB){
            t.add(tempA + " v.s " + tempB);
        }
    }
}
System.out.println(String.join("\n",t));
```

Arrays dentro de arrays

Introducción

Los arrays pueden contener otros arrays en su interior:

```
int[][] a = {{2, 4, 1}, {6, 8}, {7, 3, 6, 5, 1}};
```

Veremos cómo operar sobre ellos utilizando:

- Arreglos estáticos
- Arreglos dinámicos

Arreglos estáticos

```
int[][] arrayDeArray;
```

- Primer par de corchetes: indica las filas que tendrá el arreglo
- Segundo par de corchetes: indica la cantidad de columnas

```
int[][] a = new int [4][2];
a[0][0] = 4;
System.out.println(a[0][0]);
```

Iterando un arreglo estático

```
int [][] a = {{1,2,3},{4,5,6}};
int i,j;
for(i=0;i<2;i++){
      for(j=0;j<3;j++) {
        System.out.printf("%d ", a[i][j]);
      }
      System.out.printf("\n");
}
```

Cuidado con el tamaño del array

```
int[][] a = {{2, 4, 1}, {6, 8}, {7, 3, 6, 5, 1}};
int i,j;
for(i=0;i<a.length;i++){
      for(j=0;j<a[i].length;j++) {
        System.out.printf("%d ", a[i][j]);
      }
      System.out.printf("\n");
}
```

a.length nos retorna 3, que corresponde a los 3 sub arreglos que tenemos, y:

- a[0].length retorna 3
- a[1].length retorna 2
- a[2].length retorna 5 que corresponde a la cantidad de elementos de cada subarreglo.

Creando arreglos estáticos de dimensiones variadas

```
int num[][] = new int[3][];
num[0] = new int[3];
num[1] = new int[2];
num[2] = new int[5];

for(i=0;i<num.length;i++){
      for(j=0;j<num[i].length;j++) {
        System.out.printf("%d ", num[i][i]);
      }
      System.out.printf("\n");
}
```

Creando arreglos estáticos de dimensiones variadas

`int[][] arrayDeArray = new int [4][2];` Creamos que el arreglo tendrá 4 filas, pero todas de tamaño 2.

```
int num[][] = new int[3][];
num[0] = new int[3];
num[1] = new int[2];
num[2] = new int[5];

for(i=0;i<num.length;i++){
      for(j=0;j<num[i].length;j++) {
        System.out.printf("%d ", num[i][i]);
      }
      System.out.printf("\n");
}
```

Ejercicios

{desafío}
latam_

Ejercicio

Crear un programa Piramide.java donde reciba como parámetro un número `filas` al momento de ejecutarse, y cree una pirámide de **n** filas, con la secuencia de números que se muestra a continuación: con `n = 5`.

Podemos observar que:

- en la fila 0 tenemos 1 elemento
- en la fila 1 tenemos 2 elementos
- en la fila 2 tenemos 3 elementos
- en la fila n tenemos n+1 elementos

Por lo que recorreremos las filas del arreglo, creando un arreglo de tamaño `n+1`

Solución

```
int n = Integer.parseInt(args[0]);
int i,j;
int[][] piramide ;

piramide = new int [n][];
//sabemos que nuestra piramide tiene n filas


// creamos las columnas
for(i=0;i<n;i++) {
        piramide[i] = new int [i+1];
}
```

Veamos que tenemos creado

```
for(i=0;i<n;i++) {
        for(j=0;j<piramide[i].length;j++) {
                System.out.printf("%d\t", piramide[i][j]);
        }
        System.out.printf("\n");
}
```

Solución

Vemos que cada elemento va incrementando en 1 a medida que lo recorremos.

```
int numero = 1;
for(i=0;i<n;i++) {
	for(j=0;j<piramide[i].length;j++) {
		piramide[i][j]=numero;
	  numero ++;
	}
}

//mostrando en pantalla
for(i=0;i<n;i++) {
	for(j=0;j<piramide[i].length;j++) {
		System.out.printf("%d\t", piramide[i][j]);
	}
	System.out.printf("\n");
}
```

```
1
2	3
4	5	6
7	8	9	10
11	12	13	14	15
```

Solución - bonus

```
int i,j;
int numero=1;
int filas = Integer.parseInt(args[0]);

int[][] piramide = new int [filas][];
for(i=0;i<n;i++) {
      piramide[i]= new int [i+1];
 //creamos la fila

//asignamos el valor a la fila recien creada
      for(j=0;j<piramide[i].length;j++) {
            piramide[i][j] = numero;
            numero++;
      }
}
```

```
//mostrando en pantalla
for(i=0;i<n;i++) {
      for(j=0;j<piramide[i].length;j++) {
            System.out.printf("%d\t", piramide[i][j]);
      }
      System.out.printf("\n");
}
```

```
1
2     3
4     5     6
7     8     9     10
11    12    13    14    15
```

Arreglos dinámicos

```
ArrayList<ArrayList<String>> arregloDeArreglos = new ArrayList<ArrayList<String>>();
```

Con esta estructura podemos ir agregando a él nuevos arreglos.

```
ArrayList<String> animamalesConA = new ArrayList<String>(Arrays.asList("Asno", "Abeja", "Alacrán",
"Armadillo"));
arregloDeArreglos.add(animalesConA);

ArrayList<String> animamalesConB = new ArrayList<String>(Arrays.asList("Burro", "Boa", "Buey"));
arregloDeArreglos.add(animamalesConB);
```

o agregarlos sin crear variable animalesConC

```
arregloDeArreglos.add(new ArrayList<String>(Arrays.asList("Castor", "Conejo", "Camello", "Canguro", "Caracol", "Calamar")));
```

En la variable, System.out.println(arregloDeArreglos)

```
[[Asno, Abeja, Alacrán, Armadillo], [Burro, Boa, Buey], [Castor, Conejo, Camello, Canguro, Caracol, Calamar]]
```

Agregando elementos

```
int i;
String animal = "Cobra";
for(i=0;i<arregloDeArreglos.size();i++){ //el método size() retornara 3, ya que nuestro
arreglo tiene 3 arreglos
    if(animal.toLowerCase().charAt(0) ==
arregloDeArreglos.get(i).get(0).toLowerCase().charAt(0) ){  //queremos acceder a la primera
letra de la primera palabra
        arregloDeArreglos.get(i).add(animal);
    }

}
System.out.println(arregloDeArreglos);
```

```
[[Asno, Abeja, Alacrán, Armadillo], [Burro, Boa, Buey], [Castor, Conejo, Camello, Canguro, Caracol, Calamar, Cobra]]
```

Agregando elementos

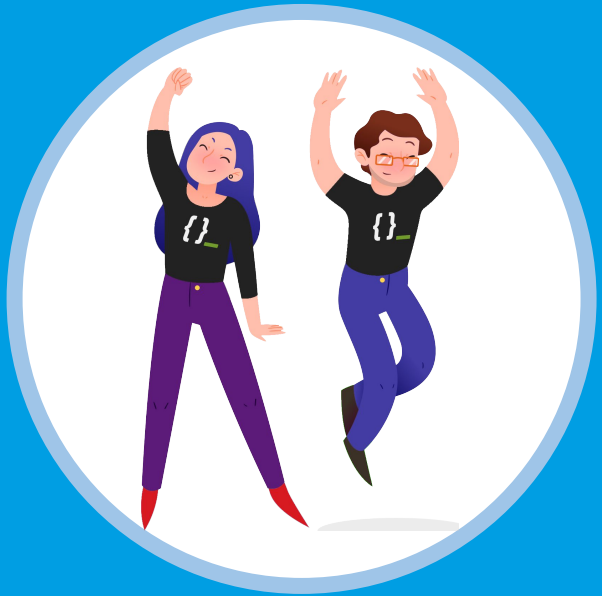
Si queremos ahora agregar un animal con D, como Delfín,

Creemos un método que reciba como parámetros el arreglo de arreglos y el animal a ingresar.

```
static ArrayList<ArrayList<String>> agregarAnimal(ArrayList<ArrayList<String>> animales, String animal){

      int i;
      for(i=0;i<animales.size();i++){ //el método size()
          if(animal.toLowerCase().charAt(0) == animales.get(i).get(0).toLowerCase().charAt(0) ){
              animales.get(i).add(animal);
          }
      }
      return animales;
}
```

Hasta el momento, solo agrega a animales



Cierre

{desafío}
latam_

¿Existe algún concepto que no hayas comprendido?

Volvamos a revisar los conceptos que más te hayan costado antes de seguir adelante

Reflexionemos

{desafío} latam_

{desafío} latam_

Academia de talentos digitales