/* Arreglos y archivos */

Sesión conceptual 01

{desafío} latam_

- Objetivo de la sesión
- Activación de conceptos clave

- Conceptualización
- Ejercitación

- Reflexión

Inicio

{desafío}
latam_

/* Introducción a Arrays */

{desafío}
latam_

¿Para qué sirven los arrays?

Los arrays son muy utilizados dentro de la programación. Nos permiten resolver diversos tipos de problemas. Algunos posibles usos son:

- En una aplicación web podemos traer los datos de la base de datos en un array y luego mostrar los datos en la página
- Al traer los datos desde una APO podría venir una colección o podríamos guardarlo dentro de un arreglo
- Traer información guardada en uno o más archivos
- Crear gráficos

Tipos de arreglos

En java existen dos clasificaciones para los arreglos:

- **De tamaño fijo**: Son arreglos de tamaño estático.
- **De tamaño flexible**: En tiempo de ejecución de nuestro programa pueden ir cambiando su tamaño.

Creando arrays estáticos

Definición:

```
int[] a;

int b[];
```

Tamaño del arreglo

```
a = new int [4];  //arreglo llamado a, de tipo enteros, de tamaño 4
```

Para introducir un valor:

```
a[0] = 4; // en la posición 0 agrega el valor 4
```

o bien, al momento de declarar un arreglo, asignarle los valores iniciales:

```
int[] ba = {2,4,5,6};//arreglo llamado ba, de tipo enteros, tamaño 4.
```

Creando arrays dinámicos

Agregar la librería, `import java.util.ArrayList;`

Su sintaxis es la siguiente:

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

ArrayList permite tipos de datos como objetos, no datos primitivos!

Para crear un ArrayList con enteros:

```
ArrayList<Integer> arrayInt = new ArrayList<Integer>();
```

Mostrando un array

```
int[] ba = {2,4,5,6};
```

Existe el método Arrays.toString(), que transforma el array a String

```
System.out.println(Arrays.toString(ba));
```

```
[2, 4, 5, 6]
```

Recordar que debemos agregar la librería java.util.Arrays. Si escribimos

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

Índices

Cada elemento del arreglo tiene una posición determinada, a la cual se le denomina índice

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

Si queremos acceder al primer elemento, debemos acceder a la posición 0.

```
System.out.printf("%d\n",a[0]) // 1
```

Notar que los índices comienzan desde cero.

Recorrido

Para el mismo caso anterior, si queremos obtener cada uno de los elementos, debemos recorrer con un ciclo desde 0 hasta n-1, donde n es el tamaño del arreglo.

Para obtener el tamaño del arreglo, podemos obtenerlo con la propiedad `length` de un arreglo.

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

Índices más allá de los límites

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

```
------------------------------------------------------------------------
java.lang.ArrayIndexOutOfBoundsException: Index 6 out of bounds for length 5
        at .(#35:1)
```

```
System.out.printf("%d\n", a[-1]);
```

```
------------------------------------------------------------------------
java.lang.ArrayIndexOutOfBoundsException: Index -1 out of bounds for length 5
        at .(#36:1)
```

El Array ARGV

En el método main, podemos ver que éste recibe un parámetro, que tiene la forma de un arreglo de `String[]` llamado `args`.

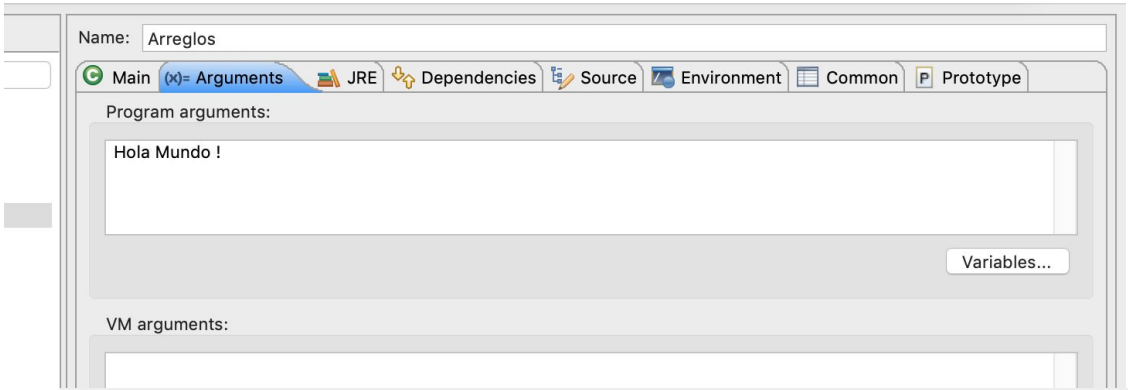
```
public static void main(String[] args) {

}
```

```
public static void main(String[] args) {
    int i,n = args.length;
    for(i=0;i<n;i++) {
       System.out.printf("%s\n", args[i]);
    }
 }
```

argv en Eclipse

Vamos a la pestaña Run-> Run Configurations -> Arguments En la ventana de Program arguments: agregaremos los parámetros que se desean ingresar.



En la consola veremos:
hola
mundo
!

argv en la consola

```
>javac Arreglos.java
>java Arreglos.java esta es una prueba
esta
es
una
prueba
```

Quiz

{desafío}
latam_

/* Operaciones básicas en un Array */

{desafío}
latam_

Agregando un elemento

Modifier and Type	Method	Description
void	add(int index, E element)	Inserts the specified element at the specified position in this list.
boolean	add(E e)	Appends the specified element to the end of this list.

```
ArrayList<Integer> a = new
ArrayList<Integer>();
a.add(1);
a.add(2);
a.add(3);
```

```
System.out.println(a)
```

```
[1, 2, 3]
```

```
a.add(1,4);  //[1, 4, 2]
```

Cuidado al ingresar en un índice que no exista

```
a.add(10,5);
```

```
----------------------------------------------------------------
java.lang.IndexOutOfBoundsException: Index: 10, Size: 4
```

{desafío} latam_

Método contains()

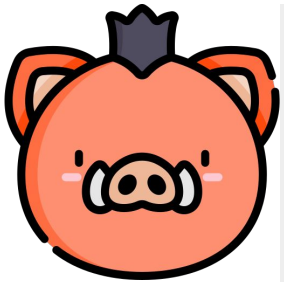
Método contains revela si un elemento está o no dentro del arreglo

```
System.out.println(a.contains(11)); //false
System.out.println(a.contains(1)); //true
```

Ejemplo de uso del método contains()

Tenemos el siguiente ArrayList llamado ingredientes, y se nos pide crear un programa donde el usuario ingrese un ingrediente y se muestre en pantalla si el ingrediente existe o no.

ingredientes // [piña, jamón, salsa, queso]



```
ArrayList<String> ingredientes = new ArrayList<String>();
ingredientes.add("piña");
ingredientes.add("jamón");
ingredientes.add("salsa");
ingredientes.add("queso");

Scanner sc = new Scanner(System.in);
String ingrediente = sc.nextLine();
if(ingredientes.contains(ingrediente)){
   System.out.printf("Tiene el ingrediente\n");

}
else System.out.printf("No lo tiene\n");
```

Ejercicio agregar elemento

Dado un arreglo llamado ingredientes se nos pide crear un programa donde el usuario pueda consultar si un ingrediente existe en la pizza, y si no existe debe ser añadido a la lista de ingredientes.

`ingredientes // [piña, jamón, salsa, queso]`

```
Scanner sc = new Scanner(System.in);
String ingrediente = sc.nextLine();

if(ingredientes.contains(ingrediente)){
        System.out.printf("El ingrediente ya se encuentra dentro de la pizza\n");

}
else {
        ingredientes.add(ingrediente);
        System.out.printf("El ingrediente %s fue agregado\n",ingrediente);
}
System.out.println(ingredientes);
```

Remover elementos

- Eliminar todos los elementos de un arreglo

```
ArrayList<String> a = new ArrayList <String>();
a.add("a");
a.add("b");
a.add("c");
a.add("d");
a.clear();
```

- Eliminar según el índice

```
ArrayList<String> a = new ArrayList <String>();
a.add("a");
a.add("b");
a.add("c");
a.add("d");
a.remove(1); // "b"       //opción 1
```

```
String borrado = a.remove(1); // "b"  opción 2
```

Remover elementos

- Eliminar elemento que coincide con el valor entregado

```
ArrayList<String> a = new ArrayList <String>();
a.add("a");
a.add("b");
a.add("c");
a.add("d");
System.out.println(a);          //[a,b,c,d]
a.remove("a");
System.out.println(a);          //[b,c,d]
```

Elimina solo la primera ocurrencia del elemento

Remover elementos

- Eliminar todos los elementos dentro de una colección.

```
ArrayList<String> a = new ArrayList <String>();
a.add("a");
a.add("b");
a.add("c");
a.add("c");
a.add("c");
a.add("c");
a.add("a");
a.add("d");
System.out.println(a);  //[a, b, c, c, c, c, a, d]
ArrayList<String> elementosABorrar = new ArrayList<String>();
elementosABorrar.add("a");
elementosABorrar.add("c");
a.removeAll(elementosABorrar );
System.out.println(a);  //[b, d]
```

Reemplazar un elemento de un arreglo

- set(int index, E element)

```
ArrayList<String> a = new ArrayList <String>();
a.add("a");
a.add("b");
a.add("c");
a.add("d");
System.out.println(a);  //[a, b, c, d]
a.set(1, "k");
System.out.println(a);  //[a, k, c, d]
```

```
String elementoCambiado = a.set(0,"j");
System.out.println(elementoCambiado); //a
```

Contar elementos

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

a.add("a");

a.add("b");

a.add("c");

a.add("d");

System.out.println(a.size());  //4
```

Otras operaciones

java.util.Collections.

- sort(ArrayList): Ordenar elementos del ArrayList
- reverse(ArrayList): Invertir elementos del ArrayList
- min(ArrayList): obtiene el elemento menor del ArrayList
- max(ArrayList): obtiene el elemento mayor del ArrayList
- frecuency(ArrayList, elemento): retorna la frecuencia del elemento en el ArrayList
- shuffle(ArrayList): desordena el ArrayList

Otro método para crear arreglos

```
ArrayList<Integer> ejemplo = new ArrayList<Integer>(Arrays.asList(1,2,3,4,5));

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

```
[1, 2, 3, 4, 5]
```

importando la librería java.util.Arrays;

Quiz

{desafío}
latam_

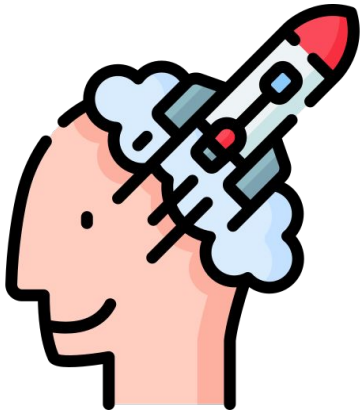
/* Iterando a partir del índice */

¿Qué tipos de problemas podemos resolver recorriendo un arreglo?

Recorrer los elementos de un array nos permite

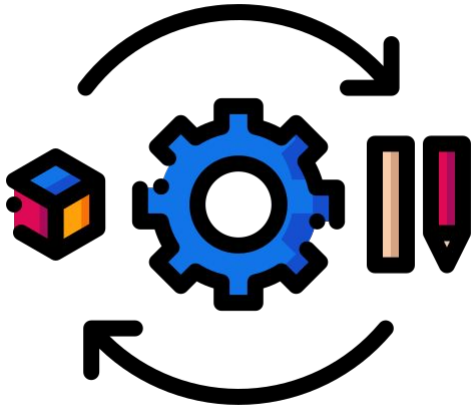
- operar sobre sus valores
- para filtrar valores
- transformar valores

Por ejemplo: Dado un arreglo de precios, podemos encontrar el valor promedio, incrementar todos los valores o aplicar un descuento.



Formas de iterar

- **Ciclo for usando índice****
- Ciclo for avanzado
- **Ciclo while****
- Usando iterador
- Streams



Ciclo for usando índice

```
int i;
for(i=0;i<arreglo.size();i++) {
        System.out.printf("%d\n",arreglo.get(i));

}
```

Ciclo while usando índice

```
int i=0;
while(i<arreglo.size()){
        System.out.printf("%d\n",arreglo.get(i));

        i+=1;
}
```

Hardcode

Mala práctica

```
int i;
for(i=0;i<5;i++) {
        System.out.printf("%d\n",arreglo.get(i));

}
```

Buena práctica

```
int i;
for(i=0;i<arreglo.size();i++) {
        System.out.printf("%d\n",arreglo.get(i)
);
}
```

¿Qué tipo de problemas podemos resolver?

Aplicando lo aprendido podemos:

- Tomar todos los elementos y transformarlos, por ejemplo, de entero a string.
- Filtrar, o sea seleccionar o mostrar solo los elementos que cumplen algún criterio
- Reducir: Por ejemplo, sumar todos los elementos, concatenarlos o multiplicarlos.

Ejemplo de transformación

Recibir los datos como parámetros, como Strings y pasarlos a enteros para operar sobre ellos.

```
ArrayList<Integer> numerosArgs = new ArrayList<Integer>();
int i;
for(i=0;i<args.length;i++){}
        numerosArgs.add(Integer.parseInt(args[i]));
}
System.out.println(numerosArgs);
```

Filtrar elementos de un arreglo

```
ArrayList<Integer> numeros = new ArrayList<Integer>

(Arrays.asList(100,200,500,1000,500,2000,10));

System.out.println(numeros);

int i;

int n = numeros.size();

ArrayList<Integer> numerosFiltrados = new
ArrayList<Integer>();

for(i=0;i<n;i++)

{

        if(numeros.get(i)>=1000) {

                numerosFiltrados.add(numeros.get(i));

        }

}

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

```
[100, 200, 500, 1000, 500, 2000, 10]
[1000, 2000]
```

Ejercicio de arreglos

`AdictoALasRedes.java`

Se pide crear un método `scanAddicts` que reciba un arreglo con los minutos de uso y como resultado entregue un nuevo arreglo cambiando todas las medidas inferiores a 90 minutos como `"bien"` y todas las mayores o iguales a 90 como `"mal"`. Los minutos de uso se deben entregar como parámetro al programa.

Ejercicio transformando segundos a minutos

Se tiene un arreglo con la cantidad de segundos que demoraron algunos procesos y se necesita un método para transformar todos los datos a minutos (las fracciones de minuto serán ignoradas).

El método debe llamarse `toMinutes`. Debe recibir el arreglo con los tiempos en segundos y devolverlo con los tiempos en minutos.

Iterando con for avanzado / for each

```
ArrayList<Integer> arreglo = new ArrayList <Integer>(Arrays.asList(1,2,3,4,5));


    for(int temp : arreglo) {
            System.out.printf("%d\n",temp);
    }
```

```
1
2
3
4
5
```

Usando iterador

Utilizando la libreria java.util.Iterator;

- hasNext() retorna true cuando existe un elemento a continuación. En caso contrario retorna false cuando ya no quedan más elementos a revisar.
- Next() retorna el elemento del arreglo que se va a revisar.

```
ArrayList<Integer> arreglo = new ArrayList <Integer>(Arrays.asList(1,2,3,4,5));

Iterator<Integer> it = arreglo.iterator();

while(it.hasNext()){

    Integer numero = it.next();

    System.out.println(numero);

}
```

Transformando con for avanzado

El for avanzado o for each nos permite recorrer un arreglo y operar sobre cada uno de sus elementos. Si por ejemplo tenemos un listado de precios y queremos aumentar cada uno de ellos en un 20% (es decir, multiplicarlos por 1.2 cada uno de los precios) podemos hacer lo siguiente.

```
ArrayList<Integer> precios = new ArrayList<Integer>(Arrays.asList(120, 210, 309, 104, 192));
ArrayList<Integer> preciosModificados = new ArrayList<Integer>();

for(int temp : precios) {
        preciosModificados.add((int)(temp*1.2f));
}
System.out.println(precios);
System.out.println(preciosModificados);
```

```
[120, 210, 309, 104, 192]
[144, 252, 370, 124, 230]
```

Limitación importante

Al acceder solo a los elementos y no a la posición de estos, no podemos modificarlos directamente dentro del arreglo, por lo que necesitamos un arreglo nuevo para guardar los resultados

```
ArrayList<Integer> numeros = new
ArrayList<Integer>(Arrays.asList(1,3,0,-4,5,2,-3,-4));

for(int temp : numeros) {
   if(temp < 0) temp = 0;
}
System.out.println(numeros);
```

```
[1, 3, 0, -4, 5, 2, -3, -4]
```

Ejercicio

Crear un método llamado `augment()` que reciba un arreglo y un multiplicador, y que como resultado de un arreglo con todos sus valores aumentados.

Ejercicio 2

Supongamos que tenemos un caso donde tenemos un arreglo de notas y queremos calcular el promedio, pero dentro de este arreglo tenemos alumnos que no dieron la prueba, y tienen como nota 0. Para complicar el ejercicio, diremos que nada nota 0 tendrá una nota base 2.

```
notas // [5, 7, 1, 3, 5, 8, 9, 0, 0, 3]
```

Filtrando con for each

Supongamos que tenemos un arreglo de notas y queremos mostrar todas las notas superiores a 5.

```
ArrayList<Integer> notas = new ArrayList<Integer>(Arrays.asList(8,2,5,7,2,9,9,6));
for(int temp : notas) {
        if(temp > 5)
                System.out.println(temp);
}
```

o generar un nuevo arreglo con estos elementos

```
ArrayList<Integer> notasFiltro = new ArrayList<Integer>();
for (int temp : notas) {

        if(temp > 5) {
                notasFiltro.add(temp);
        }
}
System.out.println(notasFiltro);
```

Filtrando con iterador

```
ArrayList<Integer> nums = new ArrayList<Integer>(Arrays.asList(10,4,14,12,9,3,8));
Iterator<Integer> it = nums.iterator();
while(it.hasNext()) {
        if(it.next() > 5) {
                it.remove();
        }
}
System.out.println(nums);
```

```
[4, 3]
```

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